VORTEX INFORMATION DISPLAY SYSTEM PROGRAM DESCRIPTION MANUAL

February 11, 1975

M&S COMPUTING, INC.
PREFACE

The Vortex Information Display System (VIDS) Program Description Manual explains the design of VIDS software that is used to collect and process wing-tip-trailing vortex data received from Laser Doppler Velocimeter Systems. VIDS was developed for NASA at Marshall Space Flight Center (Contract No. NAS8-25621, Mods. 17 and 18), under a joint NASA/FAA venture to study the effects of air disturbances created by moving aircraft.

Prepared by:

R. Conway
G. N. Matuck
J. M. Roe
J. Taylor
A. Turner

Project Manager:

E. I. Eastin

Approved by:

J. W. Meadlock
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td><strong>1. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 PDP-11 Support Software</td>
<td>1</td>
</tr>
<tr>
<td>1.2 M&amp;S Computing Application Software</td>
<td>1</td>
</tr>
<tr>
<td>1.3 M&amp;S Computing Support Software</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Hardware Configuration</td>
<td>2</td>
</tr>
<tr>
<td><strong>2. DATA ACQUISITION</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 ATVWS Overview</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Interaction</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Critical Parameters</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3 Design Parameters</td>
<td>7</td>
</tr>
<tr>
<td>2.1.4 Module Descriptions</td>
<td>8</td>
</tr>
<tr>
<td>2.2 READ Overview</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Interaction</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2 Critical Parameters</td>
<td>15</td>
</tr>
<tr>
<td>2.2.3 Design Parameters</td>
<td>15</td>
</tr>
<tr>
<td>2.2.4 Module Descriptions</td>
<td>15</td>
</tr>
<tr>
<td>2.3 FILL Overview</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1 Interaction</td>
<td>28</td>
</tr>
<tr>
<td>2.3.2 Critical Parameters</td>
<td>28</td>
</tr>
<tr>
<td>2.3.3 Module Descriptions</td>
<td>28</td>
</tr>
<tr>
<td><strong>3. FORTRAN ROUTINES</strong></td>
<td>39</td>
</tr>
<tr>
<td>3.1 Start Flyby</td>
<td>39</td>
</tr>
<tr>
<td>3.2 Device Selection</td>
<td>52</td>
</tr>
<tr>
<td>3.3 Get Addresses for Output</td>
<td>52</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS
(continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4 System Initialization</td>
<td>55</td>
</tr>
<tr>
<td>3.5 Update System Parameters</td>
<td>57</td>
</tr>
<tr>
<td>3.6 Raw Data Dump</td>
<td>59</td>
</tr>
<tr>
<td>3.7 Read Data From Tape</td>
<td>59</td>
</tr>
<tr>
<td>3.8 Encode Integers</td>
<td>63</td>
</tr>
<tr>
<td>3.9 Get Address</td>
<td>63</td>
</tr>
<tr>
<td>3.10 Set Plane Type</td>
<td>65</td>
</tr>
<tr>
<td>3.11 Calculate Velocity</td>
<td>68</td>
</tr>
<tr>
<td>3.12 Find Vortex Center</td>
<td>68</td>
</tr>
<tr>
<td>3.13 Display Output Data</td>
<td>75</td>
</tr>
<tr>
<td>3.14 Terminate Program</td>
<td>75</td>
</tr>
<tr>
<td>3.15 Scatter Plot Generation</td>
<td>77</td>
</tr>
<tr>
<td>4. DISPLAY CONTROLLER</td>
<td>81</td>
</tr>
<tr>
<td>4.1 User Input Processor</td>
<td>81</td>
</tr>
<tr>
<td>4.2 Application Program Request Processor</td>
<td>98</td>
</tr>
<tr>
<td>4.3 Command Routines</td>
<td>111</td>
</tr>
<tr>
<td>4.4 Input/Output Interrupt Processors</td>
<td>119</td>
</tr>
<tr>
<td>4.5 Disk I/O Handling</td>
<td>123</td>
</tr>
<tr>
<td>4.6 Executive</td>
<td>123</td>
</tr>
<tr>
<td>5. DISPLAY LIBRARIAN</td>
<td>127</td>
</tr>
<tr>
<td>5.1 Display Book</td>
<td>128</td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS

(continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1</td>
<td>Display Index</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Display Chapters</td>
</tr>
<tr>
<td>5.2</td>
<td>Librarian Processing Flow</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Control Card Processing</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Pen, Compose, and Line Card Processing</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Text Card Processing</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Output Formatting</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Cleanup</td>
</tr>
<tr>
<td>6.</td>
<td>PROGRAM DISPLAYS AND OUTPUT</td>
</tr>
<tr>
<td>6.1</td>
<td>Program Displays</td>
</tr>
<tr>
<td>6.2</td>
<td>Program Output</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Magnetic Tape Output</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Time-Based Plots</td>
</tr>
<tr>
<td>6.2.3</td>
<td>X-Y Plots</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Tabular Data</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Scatter Plots</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>ATVWS Listings</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>READ Listings</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>FILL Listings</td>
</tr>
<tr>
<td>APPENDIX D</td>
<td>FORTRAN Listings</td>
</tr>
<tr>
<td>Figure No.</td>
<td>Title</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1-1</td>
<td>VIDS Hardware System Components</td>
</tr>
<tr>
<td>2-1</td>
<td>Data Acquisition (Banning Vortex)</td>
</tr>
<tr>
<td>2-2</td>
<td>Enable Interrupts</td>
</tr>
<tr>
<td>2-3</td>
<td>Initialize</td>
</tr>
<tr>
<td>2-4</td>
<td>Disable Interrupt 1</td>
</tr>
<tr>
<td>2-5</td>
<td>Disable Interrupt 2</td>
</tr>
<tr>
<td>2-6</td>
<td>Enable Flyby Interrupt</td>
</tr>
<tr>
<td>2-7</td>
<td>Initialize Buffers</td>
</tr>
<tr>
<td>2-8</td>
<td>Error Process</td>
</tr>
<tr>
<td>2-9</td>
<td>Data Buffer Format</td>
</tr>
<tr>
<td>2-10</td>
<td>LDV1 EOF, LDV2 EOF</td>
</tr>
<tr>
<td>2-11</td>
<td>Process Data</td>
</tr>
<tr>
<td>2-12</td>
<td>Disk Write Preamble</td>
</tr>
<tr>
<td>2-13</td>
<td>LDV1 Input, LDV2 Input</td>
</tr>
<tr>
<td>2-14</td>
<td>LDV1 Wait</td>
</tr>
<tr>
<td>2-15</td>
<td>LDV2 Wait</td>
</tr>
<tr>
<td>2-16</td>
<td>Calculate Disk Sector</td>
</tr>
<tr>
<td>2-17</td>
<td>Process Full Buffer</td>
</tr>
<tr>
<td>2-18</td>
<td>Disk Write</td>
</tr>
<tr>
<td>2-19</td>
<td>Fill Process</td>
</tr>
<tr>
<td>2-20</td>
<td>Process Display Data</td>
</tr>
<tr>
<td>2-21</td>
<td>Read and Process Data</td>
</tr>
<tr>
<td>2-22</td>
<td>Process Buffer 1</td>
</tr>
<tr>
<td>2-23</td>
<td>Process Buffer 2</td>
</tr>
<tr>
<td>2-24</td>
<td>Disk Read</td>
</tr>
<tr>
<td>2-25</td>
<td>Set Data in Process Buffer</td>
</tr>
<tr>
<td>2-26</td>
<td>Set Maximum Data</td>
</tr>
<tr>
<td>2-27</td>
<td>Process Times</td>
</tr>
<tr>
<td>3-1</td>
<td>Start Flyby</td>
</tr>
<tr>
<td>3-2</td>
<td>Dummy 1/Tape Initialization</td>
</tr>
<tr>
<td>3-3</td>
<td>Dummy 2/Flyby Initialization</td>
</tr>
<tr>
<td>3-4</td>
<td>Initialize for Tabular Output</td>
</tr>
<tr>
<td>3-5</td>
<td>Complete Initialization for Flyby</td>
</tr>
<tr>
<td>3-6</td>
<td>Plane Index Determination</td>
</tr>
<tr>
<td>3-7</td>
<td>Dependent Parameters</td>
</tr>
<tr>
<td>3-8</td>
<td>Abort Flyby</td>
</tr>
<tr>
<td>3-9</td>
<td>Terminate Flyby</td>
</tr>
<tr>
<td>3-10</td>
<td>Process Frame of Data</td>
</tr>
<tr>
<td>3-11</td>
<td>Determine Tape Status</td>
</tr>
<tr>
<td>3-12</td>
<td>Determine Page Status</td>
</tr>
<tr>
<td>3-13</td>
<td>Device Selection</td>
</tr>
<tr>
<td>3-14</td>
<td>Compose Field Inputs for Day and Flyby Number</td>
</tr>
<tr>
<td>3-15</td>
<td>Get Addresses for Output</td>
</tr>
<tr>
<td>Figure No.</td>
<td>Title</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>3-16</td>
<td>RTV/System Initialization</td>
</tr>
<tr>
<td>3-17</td>
<td>Update System Parameters</td>
</tr>
<tr>
<td>3-18</td>
<td>Raw Data</td>
</tr>
<tr>
<td>3-19</td>
<td>Read Data From Tape</td>
</tr>
<tr>
<td>3-20</td>
<td>DECD/Encode Integers</td>
</tr>
<tr>
<td>3-21</td>
<td>SETAD/Get Address</td>
</tr>
<tr>
<td>3-22</td>
<td>Set Plane Type</td>
</tr>
<tr>
<td>3-23</td>
<td>Calculate Velocity</td>
</tr>
<tr>
<td>3-24</td>
<td>Find Vortex Center</td>
</tr>
<tr>
<td>3-25</td>
<td>Display Output Data</td>
</tr>
<tr>
<td>3-26</td>
<td>Terminate Program</td>
</tr>
<tr>
<td>3-27</td>
<td>SCAT/Scatter Plot Generator</td>
</tr>
<tr>
<td>4-1</td>
<td>Display Controller</td>
</tr>
<tr>
<td>4-2</td>
<td>User Input Processor Components</td>
</tr>
<tr>
<td>4-3</td>
<td>User Input Executive</td>
</tr>
<tr>
<td>4-4</td>
<td>DC Initialization</td>
</tr>
<tr>
<td>4-5</td>
<td>DC Termination</td>
</tr>
<tr>
<td>4-6</td>
<td>Process User Input</td>
</tr>
<tr>
<td>4-7</td>
<td>Process SUB Command</td>
</tr>
<tr>
<td>4-8</td>
<td>Process Keyboard Input</td>
</tr>
<tr>
<td>4-9</td>
<td>Process Key-In Field Carriage Return</td>
</tr>
<tr>
<td>4-10</td>
<td>Position Alphanumeric Cursor at Next Key-In Field</td>
</tr>
<tr>
<td>4-11</td>
<td>Process Key-In Field Rubout</td>
</tr>
<tr>
<td>4-12</td>
<td>Operator Input Interface Parameter List</td>
</tr>
<tr>
<td>4-13</td>
<td>Process GS Command</td>
</tr>
<tr>
<td>4-14</td>
<td>Refresh Screen</td>
</tr>
<tr>
<td>4-15</td>
<td>Application Program Request Processor Components</td>
</tr>
<tr>
<td>4-16</td>
<td>Application Program Request to Display Controller</td>
</tr>
<tr>
<td></td>
<td>Parameter Formats</td>
</tr>
<tr>
<td>4-17</td>
<td>Parameter Buffer Formats</td>
</tr>
<tr>
<td>4-18</td>
<td>Display Input/Output Executive</td>
</tr>
<tr>
<td>4-19</td>
<td>Tabular Output</td>
</tr>
<tr>
<td>4-20</td>
<td>New Display</td>
</tr>
<tr>
<td>4-21</td>
<td>One Line Message</td>
</tr>
<tr>
<td>4-22</td>
<td>Character Plot</td>
</tr>
<tr>
<td>4-23</td>
<td>Vector Plot</td>
</tr>
<tr>
<td>4-24</td>
<td>Erase Screens</td>
</tr>
<tr>
<td>4-25</td>
<td>Refresh Message</td>
</tr>
<tr>
<td>4-26</td>
<td>Hard Copy Screens</td>
</tr>
<tr>
<td>4-27</td>
<td>Auxiliary Screens</td>
</tr>
<tr>
<td>4-28</td>
<td>Reset Options</td>
</tr>
<tr>
<td>4-29</td>
<td>High-Speed Output</td>
</tr>
<tr>
<td>4-30</td>
<td>Position Alphanumeric Cursor</td>
</tr>
<tr>
<td>Figure No.</td>
<td>Title</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>4-31</td>
<td>Send Command String to Display</td>
</tr>
<tr>
<td>4-32</td>
<td>Call Next Program</td>
</tr>
<tr>
<td>4-33</td>
<td>Read Display Library</td>
</tr>
<tr>
<td>4-34</td>
<td>Bring Up Next Display</td>
</tr>
<tr>
<td>4-35</td>
<td>Set Up First Key-In Field</td>
</tr>
<tr>
<td>4-36</td>
<td>Output Interrupt Processor</td>
</tr>
<tr>
<td>4-37</td>
<td>Input Interrupt Processor</td>
</tr>
<tr>
<td>4-38</td>
<td>Initialize Data Set</td>
</tr>
<tr>
<td>4-39</td>
<td>Perform Disk I/O</td>
</tr>
<tr>
<td>4-40</td>
<td>Executive</td>
</tr>
<tr>
<td>5-1</td>
<td>Display Generation</td>
</tr>
<tr>
<td>5-2</td>
<td>Display Index Format</td>
</tr>
<tr>
<td>5-3</td>
<td>Pen Page Format</td>
</tr>
<tr>
<td>5-4</td>
<td>Keyboard Page Format</td>
</tr>
<tr>
<td>5-5</td>
<td>Fill Page Format</td>
</tr>
<tr>
<td>5-6</td>
<td>Display Librarian</td>
</tr>
<tr>
<td>5-7</td>
<td>Determine Record Type</td>
</tr>
<tr>
<td>5-8</td>
<td>Process Control Records</td>
</tr>
<tr>
<td>5-9</td>
<td>Pen, Compose, and Line Record Processing</td>
</tr>
<tr>
<td>5-10</td>
<td>Text Record Processing</td>
</tr>
<tr>
<td>5-11</td>
<td>Verify Text Record Contents</td>
</tr>
<tr>
<td>5-12</td>
<td>Create Output</td>
</tr>
<tr>
<td>5-13</td>
<td>Terminate Processing</td>
</tr>
<tr>
<td>6-1</td>
<td>Mode Selection</td>
</tr>
<tr>
<td>6-2</td>
<td>System Parameter Selection</td>
</tr>
<tr>
<td>6-3</td>
<td>System Parameter Selection (Post-Analysis)</td>
</tr>
<tr>
<td>6-4</td>
<td>Aircraft Selection</td>
</tr>
<tr>
<td>6-5</td>
<td>Aircraft Dependent Parameter Selection</td>
</tr>
<tr>
<td>6-6</td>
<td>Aircraft Dependent Parameter Selection (Post-Analysis)</td>
</tr>
<tr>
<td>6-7</td>
<td>Display of Vortex Information</td>
</tr>
<tr>
<td>6-8</td>
<td>Display of Vortex Position</td>
</tr>
<tr>
<td>6-9</td>
<td>Display of Vortex Location</td>
</tr>
<tr>
<td>6-10</td>
<td>Sample Scatter Plot</td>
</tr>
<tr>
<td>Table No.</td>
<td>Title</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>3-1</td>
<td>Count to Velocity Conversion</td>
</tr>
<tr>
<td>3-2</td>
<td>Definition of Terms</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The Vortex Information Display System (VIDS) provides flexible control through system-user interaction for collecting wing-tip-trailing vortex data, processing this data in real time, displaying the processed data, storing raw data on magnetic tape, and post processing raw data. The data is received from two asynchronous Laser Doppler Velocimeters (LDV's) and includes position, velocity, and intensity information. The raw data is written onto magnetic tape for permanent storage and is also processed in real time to locate vortices and plot their positions as a function of time.

The interactive capability enables the user to make real time adjustments in processing data and thereby provides a better definition of vortex behavior. Displaying the vortex information in real time produces a feedback capability to the LDV system operator allowing adjustments to be made in the collection of raw data. Therefore, both raw data and processing can be continually upgraded during flyby testing to improve vortex behavior studies. The post-analysis capability permits the analyst to perform in-depth studies of test data and modify vortex behavior models to improve transport predictions.

VIDS is composed of both PDP-11 support software and M&S Computing application software running under control of the PDP-11 operating system.

1.1 PDP-11 Support Software

The PDP-11 support software includes system programs and utilities designed by Digital Equipment Corporation (DEC) to support the PDP-11 user during execution of application programs. Specifically, the PDP-11 software components are:

- Disk Operating System (DOS)
- Verification Program (VERIFY)

Sections 2 and 4 of the Vortex Information Display System User's Manual discuss these PDP-11 components in detail.

1.2 M&S Computing Application Software

The M&S Computing application software is composed of routines that control program flow and that collect, store, process, and display data. These routines are grouped as follows:

- Data Acquisition routines which input and output raw data.
- FORTRAN routines which initialize variables, locate vortex centers, and provide interfaces between the assembly language I/O routines, the vortex location routines, and the display controller.
1.3 M&S Computing Support Software

The M&S Computing support software is composed of the display librarian which creates predefined displays in a library to be used and displayed by the application software.

1.4 Hardware Configuration

The hardware configuration (see Figure 1-1) required by VIDS is:

- Digital Computer, PDP-11 series model 35 with 32K memory and 16 bit words - DEC.
- Magnetic Disk Unit, RK05 - DEC.
- Magnetic Tape Unit, TU10 - DEC.
- Graphics Display Terminal, 4014 series terminal and 613 monitor - Tektronix, Inc.
- Hard Copy Device, 4610 series - Tektronix, Inc.
- Graphics Data Tablet, HW-1-11, Summagraphics.

Operation of VIDS hardware is discussed in Section 2, Vortex Information Display System User's Manual.
VIDS HARDWARE SYSTEM COMPONENTS

Tape Transport

Disk Drive

Processor Unit

4014 Series Terminal

613 Monitor

4610 Hard Copy Unit

Graphics Data Tablet

Figure 1-1
2. DATA ACQUISITION

The Data Acquisition portion of the Banning Vortex software system is written in assembly language to handle interrupts as quickly as possible. It is divided into three major portions:

1. **ATVWS** - initialization.
2. **READ** - acquire the data from the LDV's and write the data on the disk.
3. **FILL** - read the data from the disk and format it as required by the data processing functions.

Figure 2-1 depicts the overall data flow for the Data Acquisition function and illustrates the role of each software portion. Each of the three Data Acquisition portions will be described in this section, and the section will be organized as follows:

- Overview of function
- Interaction with other functions
- Critical parameters
- Parameters which define function capability
- Module descriptions, which will include:
  - Purpose of module
  - Calling sequence
  - Other modules required
  - Critical parameters

Listings of the Data Acquisition software are contained in the Appendices.

If any of the specific items are not required for a particular portion of module description, then that heading and item will be omitted.

2.1 **ATVWS Overview**

This function is responsible for initialization of the hardware and software components which make up the Data Acquisition portion of the Banning Vortex system.
In particular, modules within this function perform the following:

- Initialize the hardware interfaces (DR-11's).
- Start flyby or test.
- Stop flyby or test.
- Start data collection.
- Stop data collection.
- Read plane type.
- Initialize pointers and flags required by the software modules.

2.1.1 Interaction

This function interacts with the other two functions (READ and FILL) in that:

- ATVWS initializes flags and indicators for those functions.
- ATVWS contains the central logic to display all error messages which describe error conditions detected by any of the three Data Acquisition functions.

2.1.2 Critical Parameters

ATVWS relies on the flag IFLD which is common to Data Acquisition and Data Processing functions, and which indicates the LDV's required for any particular test. This flag controls routine INIT and must be set prior to entry, as follows:

\[ \text{IFLD} = \begin{cases} 0, & \text{both LDV's} \\ 1, & \text{LDV 2 only} \\ 2, & \text{LDV 1 only} \end{cases} \]

2.1.3 Design Parameters

Additional error messages may easily be added simply by entering the address of the message in the list MSGAD. A particular message is displayed by executing a Jump (JMP) instruction to location HALT (which is GLOBAL) with the appropriate message index in R0. (Each index is a multiple of 2 which provides for word indexing.)
2.1.4 Module Descriptions

**Module ATVWS**

**Purpose**

The ATVWS module (Figure 2-2) initializes the interrupt vectors for the three DR-11's. It also initializes the disk file LDVS which is allocated to store the accumulated data. ATVWS is called by the Display Controller at the start of the system execution.

**Calling Sequence**

```
JSR R5, ATVWS
```

(cannot be called via FORTRAN routine)

**Modules Required**

LKTRAN

INITDS

DISPIO

**Module INIT**

**Purpose**

The INIT module (Figure 2-3) initializes all software pointers and flags and enables the DR-11 interrupts. The latter permits detection of the request A and request B interrupts.

**Calling Sequence**

```
JSR R5, INIT
```

or

```
CALL INIT
```

**Modules Required**

INITBF

**Critical Parameters**

IFLDV is a control indicator which must be set prior to INIT call:

```
IFLD = 0, both LDV's
```
ENABLE INTERRUPTS

**Figure 2-2**

- Set up vector for DR-11 A
- Set up vector for DR-11 B
- Set up vector for DR-11 C
- INITDS Initialize disk file
- Error (Y) leads to HALT Display error message
- Error (N) leads to LKTRAN Read first sector to position disk

N

B

EXIT

EXIT TO DC

A
INITIALIZE

INIT

Clear read frame counter and FIN flag

2-7

INITBF
Initialize input buffers

Both LDV's

N

Y

Initialize DR-11 A LDV 1

Initialize DR-11 A only

A

Initialize DR-11 B LDV 2

EXIT

Figure 2-3
IFLD = 1, LDV 2 only
IFLD = 2, LDV 1 only

IFLD is the first word in common IFLDV.

Module DISABL

Purpose

The DISABL module (Figure 2-4) disarms the hardware interrupts for the two DR-11's. A software flag FIN (normally zero) is set to a positive one to denote end of test.

Calling Sequence

    JSR  R5, DISABL
or    CALL DISABL

Module DISABI

Purpose

The DISABI module (Figure 2-5) disarms the two interrupts for the DR-11 used to input the Start-of-Flyby signal and the plane type.

Calling Sequence

    JSR  R5, DISABI
or    CALL DISABI

Module INTCA

Purpose

The INTCA module (Figure 2-6) responds to the interrupt generated by the DR-11 that specifies the Start-of-Flyby and plane identification. A Start-of-Flyby signal ends any current test and initializes the hardware and software in preparation for the start of another flyby.

Modules Required

    DISABL
DISABLE INTERRUPT 1

Disarm interrupt B for LDV 1 & 2

Set FIN flag (test complete)

Clear DR-11 status words

EXIT

DISABLE INTERRUPT 2

Clear status register for DR-11 C

EXIT

ENABLE FLYBY INTERRUPT

INTCA

2-4

DISABL

Stop data input

Read plane ID

Set test start flag

EXIT

Figure 2-4
Figure 2-5
Figure 2-6
Module INITBF

Purpose

The INITBF module (Figure 2-7) initializes all the data input buffers and assigns a buffer to each of the LDV's. This module also initializes all software pointers and flags.

Calling Sequence

JSR PC, INITBF

(Module cannot be called via FORTRAN routine).

Module HALT

Purpose

The purpose of the HALT module (Figure 2-8) is to stop or abort the current flyby, to display an appropriate message on the CRT, and to initialize the system for further test or flybys. This module is not a subroutine; it is the central error handling function.

Calling Sequence

MOV ERR, R0  (set error message index (see Section 2.1.3)).

JMP HALT    (jump to error routine).

where:

ERR is a unique error number. Error numbers 2-8 are currently used and are assigned such that R0 may be used as a word index. That is, R0 contains an integer which is a multiple of 2.

Modules Required

DISPIO

2.2 READ Overview

This function is responsible for acquiring the data for the Banning Vortex software system.

Modules within this function respond to the DR-11 interrupts, input the data words in response to the interrupts, accumulate the data in memory buffers, and write the data on the disk as the memory buffers are filled.
INITIALIZE BUFFERS

INITBF

Free all buffers

Set initial buffer for LDV 1&2

Clear and/or initialize all flags, etc.

EXIT

ERROR PROCESS

HALT

Go to low level. Set error message address

DISPIO

Go to large characters

DISPIO

Blank screen

DISPIO

Output error message

DISPIO

Request first display

EXIT TO DC

Figure 2-7

Figure 2-8
2.2.1 Interaction

This function interacts with ATVWS in that ATVWS initializes READ pointers and flags and enables the DR-11 interrupts which cause READ function responses. READ interacts with FILL in that FILL functions process data on the disk written by the READ functions. READ functions also maintain a count for the FILL functions to enable them to track the progress of each LDV.

2.2.2 Critical Parameters

Modules within the READ function manage a pool of data input buffers. These buffers each contain 256 words, which is the physical size of a disk sector. Figure 2-9 illustrates the format of these data buffers. As can be seen by this figure, each buffer contains 248 data words and 8 control words. The number of buffers assigned to this buffer pool is critical to the system. Enough buffers to support input functions are required when the rate of input is at a maximum for both LDV's. This rate of input must be sustained during those periods (such as data being FILLed) when data cannot be written on the disk.

2.2.3 Design Parameters

Additional buffers may be added to the system; modules within READ are designed to manage any number. Location BUF contains the current number of buffers and must be changed as buffers are added or removed from the system.

Each buffer consists of two Reserve Block Word operators (.BLKW):

- .BLKW SIZB
- .BLKW SIZD

SIZB (number of control words) has been equated to 8; SIZD (number of data words) to 248. Changing either or both of these parameters will in turn change all data buffers.

2.2.4 Module Descriptions

Module INTAA

Purpose

The INTAA module (Figures 2-10 through 2-12) responds to the request A (end of frame (EOF)) interrupt for DR-11 A (LDV 1). Upon entry, this module inputs four data words (to clear the interrupt signal) and discards the data. If the interrupt is the first EOF for the flyby, the request B interrupt (data input) is enabled. Otherwise, the current buffer is marked to denote an EOF has occurred,
## DATA BUFFER FORMAT

<table>
<thead>
<tr>
<th>Frame No.</th>
<th>Data Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control (see below)</td>
<td></td>
</tr>
<tr>
<td>Header</td>
<td></td>
</tr>
<tr>
<td>Sector Number</td>
<td></td>
</tr>
<tr>
<td>Time 1</td>
<td></td>
</tr>
<tr>
<td>Time 1</td>
<td></td>
</tr>
<tr>
<td>Time 2</td>
<td></td>
</tr>
<tr>
<td>Time 2</td>
<td></td>
</tr>
</tbody>
</table>

8 Words of Control

First Time of Buffer

Last Time of Buffer

248 Words of Data

### Control Byte

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

- Not Used
- End of Test
- End of File
- Buffer Ready for Write
- Buffer Busy

---

Figure 2-9

-16-
LDV1 EOF

INTAA

2-11

INTR

Initialize for interrupt

Set LDV1 indicator (R2 = 0)

Reset first time flag for B interrupt

Set EOF indicator in buffer

SAW

Go switch buffers

Frame count (FRAMES) = FRAMES + 1

EXIT

INTAB

2-11

INTR

Initialize for interrupt

Set LDV2 indicator (R2 = 2)

Figure 2-10

-17-
PROCESS DATA

DISK WRITE PREAMBLE

1. **INTR**
   - Go to level 7 and save R0-R5
   - Bring in data words and discard
   - First EOF
     - Set not first EOF flag
     - Initialize B interrupt routine
     - EXIT

2. **INTBA3**
   - Any writes waiting
     - N
   - Is read in progress
     - N
   - Is it locked
     - N
     - Lock and go to level 2
     - WRITE
       - Write block to disk
     - Restore R0-R5
     - EXIT

3. **EXIT route**

**Figure 2-11**

**Figure 2-12**
a new buffer is assigned to the LDV, and the full buffer is written on disk, if the disk write services are available. If the write services are not available, then the buffer is marked for a "write ready," and the interrupt module is exited. The buffer will be written at the next available time.

Modules Required

SAW

WRITE

Module INTAB

Purpose

The INTAB module (Figures 2-10 through 2-12) performs the same function as INTAA (see X.5.1 above) except INTAB provides support for DR-11 B. Certain functions (see flowchart) are common to both INTAA and INTAB.

Modules Required

SAW

WRITE

Module INTBA

Purpose

The INTBA module (Figures 2-12 and 2-13) inputs the application data via DR-11 A (LDV 1). This module responds to the request B interrupt, brings in the 4 data words, sets the time and increments the word count. If the data buffer is full, the buffer is marked "write read," a new buffer is assigned to the LDV, and the full buffer is written on disk, if the disk services are available. If the disk services are not available, the buffer is written at the next available time. If the buffer is not full, this module is exited after bringing in the data. This module provides code that is common to INTAA, INTAB, and INTBA.

Modules Required

SAW

WRITE

Module INTBB

Purpose

The INTBB module (Figures 2-12 and 2-13) performs the same function as INTBA except INTBB provides support for DR-11 B (LDV 2). Certain functions are
Go to level 7 and save R0 - R5

Bring in 4 data words

Set LDV indicator (R2 = 0)

Set LDV2 indicator (R2 = 2)

Set first time for block

Reset first time flag

Set time for this data

Increment data word count

Go switch blocks

Block full

Check for disk write

EXIT

Figure 2-13
Modules Required

SAW
WRITE

Module INTAW

Purpose

The INTAW module (Figure 2-14) responds to the request B interrupt for DR-11 A (LDV 1). This module processes the data prior to receiving the start of flyby (interrupt request A or EOF). The data is read in and discarded.

Module INTBW

Purpose

The INTBW module (Figure 2-15) responds to the request B interrupt for DR-11 A (LDV 2). This module processes the data prior to receiving the start of flyby interrupt. The data is read in and discarded.

Module CALBLK

Purpose

The CALBLK module (Figure 2-16) calculates the sector address for each buffer for both LDV's. A current sector address is maintained for LDV 1 and LDV 2, where LDV 1 buffers are written on even sectors and LDV 2 buffers are written on odd sectors.

Calling Sequence

JSR PC, CALBLK

(cannot be called by FORTRAN routine).

Upon return, the sector number is in R4.

Module SAW

Purpose

The SAW module (Figure 2-17) manages the buffer switching function. When called, this module assigns a new buffer to the LDV and processes the control word for the full buffer, as follows:
Figure 2-14

Figure 2-15

Figure 2-16
PROCESS FULL BUFFER

SAW

Set buffer count, start, and store

Check buffer N

Buffer busy

Y

N = N+1

All buffers

N

A

N

2-16

CALBLK

Go get sector number

Set sector in buffer

Set header, frame, and write ready

HALT

Go display error message

EXIT

Set first sector flag

EXIT

Figure 2-17

ORIGINAL PAGE IS OF POOR QUALITY
Sets store address and start address for new buffer. The store address is used to place data into the buffer; the start address is used to store buffer control information.

Sets new buffer busy (see Figure 2-9).

Calls CALBLK to obtain sector address and places the address in the old buffer control word.

Sets header and frame number in the old buffer control words. Note that the frame number is positive for LDV 1 and negative for LDV 2.

Sets the write ready status (see Figure 2-9) for the old buffer.

Calling Sequence

```assembly
JSR PC, SAW
```

(cannot be called via FORTRAN routine).

Upon entry:

- \( R2 = 0 \), for LDV 1
- \( R2 = 2 \), for LDV 2

Modules Required

- CALBLK
- HALT

Critical Parameters

This module executes on level 7.

Module WRITE

Purpose

The WRITE module (Figure 2-18) performs all functions required to write the data buffers on the disk.

Upon entry, the module searches the buffer pool to find the buffers waiting ("write ready") to be written on the disk. From the buffers ready, this module finds the next logical buffer. The next logical buffer must:

- Be in same sequence as the last buffer written. That is, if the last buffer written was for LDV 1 (even sector number), then all LDV 1 ready buffers will be written before any LDV 2 buffers (odd sector number). Likewise, if the last sector written was
DISK WRITE

WRITE

First write

Y

B

N

C

Test last write

WAITR

Error in last

2-8

HALT

Display error

message

Set sector

address in

calling

sequence

Set sector

address in

calling

sequence

Return buffer to

pool

Last buffer free

N

EXIT

EOF

buffer

Increment count for

LDV

Increment count to

read

Start with

search for

last LDV

Set LDV

indicator

R0=0 LDV1

R0=2 LDV2

Decrement write

requests

Any buffers

ready

Y

C

N

5.

Both

LDV's

N

A

Go to next

LDV

Reset lock

EXIT

Y

B

Figure 2-18

-25-
for LDV 2, then all odd sectors will be written before any even sectors.

- Be in sequence for either even or odd sector numbers. Sectors are written in the order specified by their sector numbers, which progress in increments of 2; i.e., 0, 2, 4, ... for LDV 1 or 1, 3, 5, ... for LDV 2. This method of selection is designed to minimize the disk access times since once the requests are synchronized (after the first sector is written), all other sectors can be written within two sector times, or approximately 6 milliseconds.

After a buffer is written on the disk, its control word is examined to determine if an EOF occurred for the buffer. If an EOF occurred, the number of frames written on the disk is incremented. The buffer is marked as "free" and returned to the buffer pool after it is written on the disk.

Calling Sequence

JSR PC, WRITE

(cannot be called via FORTRAN routine).

Upon entry:   R2 = 0, for LDV 1
               R2 = 2, for LDV 2

Modules Required

WAITR

LKTRAN

HALT

Critical Parameters

This module must be serially executed; it is not reentrant. To insure this, the module is protected by a flag (LOCK) which is set upon entry and reset upon exit. In addition, this module may not be executed while a read disk operation is in progress. BUSY is set by the read disk function and is reset when the read completes. This module is not executed if BUSY is set.

2.3 FILL Overview

This function is responsible for reading the data from disk and placing the data in the process buffer in the specified format. Figure 2-19 provides a
FILL PROCESS

Disk

Input Buffer
(1 sector or 256 words)

Process Buffer

Figure 2-19
A pictorial description of this process. From this figure it can be seen that:

- Data is brought from disk in data buffers which are one sector in length; and
- The data read from disk is striped and placed in the process buffer in such a manner that like data is contiguous.

One fill request results in either:

- All data for one frame being placed in the process buffer,
- A maximum of 992 data points being placed in the buffer, or
- No data being placed in the process buffer because an abort was detected.

2.3.1 Interaction

This function interacts with READ in that a count is maintained for frames written on the disk for each LDV. Basically, READ increments the count as data is written and FILL decrements the count as data is read.

2.3.2 Critical Parameters

This function is given priority over the disk write functions. BUSY is a flag (set = -1) which inhibits the write process. That is, a disk write function will not be processed if a read is in progress. This enables a FILL iteration to be processed with a minimum of disk accesses.

2.3.3 Module Descriptions

Module FILL

Purpose

The FILL module (Figures 2-20 through 2-23) provides the FORTRAN data processing functions with test data read from disk. The data is read from disk and placed in the process buffer in the format required by the calling functions.

Data read in from disk is double-buffered such that one buffer is being processed while the other buffer is being filled by the disk. This technique enables both the data read and process functions to be accomplished in essentially the time required for the disk read.

Calling Sequence

```
JSR R5, FILL
or CALL FILL
```
PROCESS DISPLAY DATA

Figure 2-20
READ AND PROCESS DATA

Figure 2-21

-30-
Figure 2-22
PROCESS BUFFER 2

Figure 2-23
Modules Required

READ
MAXV
STORE
TIFR

Critical Parameters

First word of common IFLDV denotes the test configuration as follows:

IFLD = 0, both LDV
IFLD = 1, LDV 2 only
IFLD = 2, LDV 1 only

First word in common ABTERM is used to signal a test abort:

ABTERM = 33 or 42, test is aborted

Common IHDLI is used to pass the following information:

Word 1 (MAX1) = Integer which denotes the first maximum velocity point found in data.
Word 2 (MAX2) = Integer which denotes the second maximum velocity point found in the data.
Word 3 (IEOFI) = 0, data processed normally
-1, data of frame exceeds 992 points
+1, end of flyby occurred

Common LDVDAT is used to pass the following information:

Word 1 (IFLY) = Used by FORTRAN routines.
Word 2 (IFRM) = Data frame number, where a positive frame denotes LDV1; negative, LDV 2.
Word 3 (TMINT) = Integer value of high-order time for first data word of FILL iteration. Note that each count equals 1/60 second.
Word 4 (TMINT+2) = Integer value of low-order time for first data word of FILL iteration.

Word 5 (TMEND) = Integer value of high-order time for last data word.

Word 6 (TMEND+2) = Integer value of low-order time for last data word

Word 7 (IDAY) = Used by FORTRAN routines.

Word 8 (IPLN) = Used by FORTRAN routines.

Word 9 (NUMPTS) = Integer which denotes number of data points placed in the process buffer.

Module READ

Purpose

The READ module (Figure 2-24) reads in a sector of data from the disk. For a particular FILL iteration, all reads will be performed for a particular LDV. That is, either even or odd sectors will be read.

Calling Sequence

JSR PC, READ

(cannot be called via a FORTRAN routine).

Upon entry: R0 = 0, buffer 1 read

R0 = 2, buffer 2 read

Modules Required

LKTRAN

Module STORE

Purpose

The STORE module (Figure 2-25) extracts four data words from the input buffer and places them in the process buffer in the required format.

Calling Sequence

JSR PC, STORE

(cannot be called via FORTRAN routine).
DISK READ

Figure 2-24

STORE

Move 4 data words

EXIT

Figure 2-25
Upon entry:
R2 = address of first of the four words to be moved
R4 = address of process buffer which will receive words

Upon exit:
R2 = address of next four words
R4 = unchanged

Module MAXV

Purpose

The MAXV module (Figure 2-26) searches for the two maximum velocities of the FILL iteration. An integer (1-992) is set for each of the maximums to denote the data point numbers for the process buffer of the current FILL iteration.

Calling Sequence

JSR PC, MAXV

(cannot be called via FORTRAN routine).

Upon entry:
R4 = address of current velocity in process buffer
R1 = current data point number

Upon exit:
R4 = unchanged
R1 = unchanged

Module TIFR

Purpose

The TIFR module (Figure 2-27) sets the last time and the frame number in the designated COMMON (see module FILL).

Calling Sequence

JSR PC, TIFR

(cannot be called via FORTRAN routine).

Upon entry:
R2 = current buffer address

Upon exit:
R2 = unchanged
SET MAXIMUM DATA

MAXV

 Higher than previous max 1
 Y
 N

 Higher than previous max 2
 Y
 Replace previous with this one
 N

 EXIT

 TIFR

 Set time for block

 Set frame number

 EXIT

Figure 2-26

Figure 2-27
3. FORTRAN ROUTINES

This section describes the routines which initialize variables, locate vortex centers, and provide interfaces between the data acquisition routines, the vortex location routines, and the display controller. These routines are written in FORTRAN to facilitate program modification as more knowledge about vortex behavior and detection is gained.

3.1 Start Flyby

Name

STRT

Calling Sequence

CALL STRT (N)

Where N is a six element array with the elements containing the following:

- N(1) contains the number of the display that was on the screen when STRT was called.
- N(2) contains the option number which was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) contains a flag that indicates type of input field.
- N(5) and N(6) contain data from the input field.

Description of Function

Subroutine STRT (Figures 3-1 through 3-12) provides the interface between the assembly language I/O routines, the vortex location algorithm, and the display controller. The routine initializes the output data tape, directs the placement of the chosen output display backgrounds on the terminal screens, directs the filling of data buffers from either real time LDV input or from tape, directs output of real time data to tape, and calls the vortex location algorithm. STRT also handles flyby termination as directed by LDV input signals, end of files on input data tapes, or operator keyboard input signals and cycles the program to the appropriate starting point for the next data input.

External References

DISPIO, RTV, OPEN, SFUN, INITI, SETAD, DFLT, SCAT, PTYP, DISAB1, DISABL, FILL, PUT, VREAD, CENTRD, and WAIT.
START FLYBY

Figure 3-1
FLYBY INITIALIZATION

Figure 3-3

-42-
Figure 3-4
COMPLETE INITIALIZATION FOR FLYBY

Figure 3-5
PLANE INDEX DETERMINATION

DUM3A

Determine index for plane type

3-22

PTYP
Encode day and flyby number

3-17

PFLT
Place aircraft dependent parameters in header buffer

RETURN

Figure 3-6
Figure 3-7
ABORT FLYBY

DUMMY5

Reset abort flyby flag

Real time

Y

Disab1
start flyby
interrupts

N

Set flag to
place system
parameter
display on
screen

Mag.
tape unit
0 open

Y

Post-
processing

N

A

SFUN
Write end
of file

SFUN
Write end
of file

SFUN
Backspace
1 record

A

DISPLO
Establish
screen 1 as
primary
screen

DISPLO
Bring up
first display

RETURN

Figure 3-8

-47-
TERMINATE FLYBY

Figure 3-9

-48-
PROCESS FRAME OF DATA

START

DUMMY?

Real time

Y  FILL
Fill data
buffers with
raw input

N

Stop
flyby

Y  A

N

Abort
flyby

Y  A

N

Record
data on
tape

Y  PUT
Write data
record on
tape

N

'IEOFF'  = 0

A

= -1

A

= 1

Set flag to
indicate too
many data
points

Partial
frame

Y  Reset flag
that indicates
too many
data points

N  CENTRD
Locate
vortex
centers

RETURN

B

3-19

VREAD
Read a
record of
data

3-11

DUM7A
Determine
tape status

3-12

DUM7B
Determine
page status

N

End of file

Y  A

Figure 3-10

ORIGINAL PAGE IS
OF POOR QUALITY

-49-
DETERMINE TAPE STATUS

Figure 3-11
Figure 3-12
3.2 Device Selection

Name

DSEL

Calling Sequence

CALL DSEL (N)

Where N is an eight element array with the elements containing the following:

- N(1) contains the number of the display that was on the screen when DSEL was called.
- N(2) contains the option number which was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) contains a flag indicating the type of input field.
- N(5) through N(8) contain input field values.

Description of Function

Subroutine DSEL (Figures 3-13 and 3-14) determines which input and output devices were selected by the operator and sets flags to indicate the chosen devices. The routine also is used to place compose field input for flyby number and day in header buffers for displays. In post-analysis, the routine searches the data tape for flyby numbers or day numbers that match the input number.

External References

SFUN, DISPIO, PTYP, and VREAD.

3.3 Get Addresses for Output

Name

ONCE

Calling Sequence

CALL ONCE
Figure 3-13
COMPOSE FIELD INPUTS FOR DAY AND FLYBY NUMBER

Figure 3-14
Description of Function

Subroutine ONCE (Figure 3-15) is entered only when the program is loaded. It calls the display controller to establish high-speed output in large characters and calls SETAD to get the addresses of the variables which follow:

- Blank.
- XYSYM odd elements which contain the letters of the alphabet.
- PLTSI which contains an *.
- PLTS3 which contains an o.
- IDATA which is an output buffer array.
- IL1 which is the buffer that contains dit positions for write-through lines showing default values for system parameters.
- IL2 which is the buffer that contains dit positions for write-through lines showing default values for aircraft dependent parameters.
- DATE which contains flyby date.
- IFLB which contains flyby number.

It also converts the default correlation radius from feet to counts.

External References

DISPIO and SETAD.

3.4 System Initialization

Name

RTV

Calling Sequence

CALL RTV (N)

Where N is a six element array with the elements containing the following:

- N(1) contains the number of the display that was on the screen when RTV was called.
GET ADDRESSES FOR OUTPUT

Once

Convert default R from feet to counts

Display
Establish output in large characters

Setad
Get address of blank

Setad
Get addresses of alphabet

Setad
Get address of '*'

Setad
Get address of '0'

Setad
Get address of flyby date

Setad
Get address of flyby number

Setad
Get starting address of output buffer IDATA

A

RETURN

Figure 3-15

-56-
N(2) contains the number of the option which was selected from the display.

N(3) contains the number of bytes in a compose field.

N(4) contains a flag which indicates the type of input field.

N(5) and N(6) contain data from the input field.

On any given call, all of the elements of N may not be defined.

Description of Function

During Post-Analysis, RTV (Figure 3-16) enables the interface to magnetic tape unit 1 and reads the first data record for each flyby. During both post-analysis and real-time operation, the routine sets the coordinates and calls the display controller to draw write-through lines indicating chosen system parameters. It also places the day and flyby number on the system parameters displays.

External References

DISPIO, OPEN, SFUN, VREAD, and PTYP.

3.5 Update System Parameters

Name

DFLT

Calling Sequence

CALL DFLT (N)

Where N is an eight element array containing the following:

- N(1) contains the number of the display that was on the screen when DFLT was called.
- N(2) contains the option number that was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) contains a flag indicating the type of input field.
- N(5) through N(8) contain input field values.
SYSTEM INITIALIZATION

Figure 3-16
Description of Function

Subroutine DFLT (Figure 3-17) changes the aircraft dependent parameters from a default value to a value input through the keyboard and if the update default option was selected, it updates the default value for the particular aircraft to the input value. The routine also sets the buffer containing the heading that displays aircraft dependent parameters.

External References

None.

3.6 Raw Data Dump

Name

DBUG

Calling Sequence

CALL DBUG

Description of Function

Subroutine DBUG (Figure 3-18) gives a dump of the raw data in counts that is received from the LDV's. The data includes the horizontal and vertical coordinates of a point, the number of filters, the intensity, the peak velocity, and the velocity at maximum intensity.

External References

SSWTCH, DISPIO, and SUBBIT.

3.7 Read Data From Tape

Name

VREAD

Calling Sequence

CALL VREAD

Description of Function

Subroutine VREAD (Figure 3-19) obtains raw position, velocity, and intensity data from tape for post processing and prepares the data for processing.
UPDATE SYSTEM PARAMETERS

Figure 3-17
Figure 3-18
READ DATA FROM TAPE

A

VREAD

GET
Read data tape

WAIT
Wait for I/O completion

SFUN
Get status

End of file
Y
DISPIO
Display EOF message
RETURN

N

N
4008 word record
DISPIO
Display tape error message
RETURN

Y

Process this VAN
Y
Convert velocities to absolute values

3-20
DECD
Encode flyby number

3-20
DECD
Encode day

RETURN

Figure 3-19
External References

GET, WAIT, SFUN, DISPIO, and DECD.

3.8 Encode Integers

Name

DECD

Calling Sequence

CALL DECD (KK, KKK, KP)

Where:

- KK is the number to be encoded.
- KKK is the number of bytes output.
- KP is the address that contains the encoded number.

Description of Function

Subroutine DECD (Figure 3-20) encodes an integer number. The alphanumeric results are stored left-justified. The maximum number of bytes output is five. If the encoded number is greater than five bytes, only the right most bytes are output. Signs are also output and count as one of the five bytes.

External References

None.

3.9 Get Address

Name

SETAD

Calling Sequence

CALL SETAD (I, V)

Where:

- I is the return variable containing the address of V.
ENCODE INTEGERS

Figure 3-20

DECD

Encode number

RETURN
V is an integer or real variable but must begin on an odd word variable.

Description of Function

SETAD (Figure 3-21) gets the address of a variable.

External References

None.

3.10 Set Plane Type

Name

PTYP

Calling Sequence

CALL PTYP (N)

Where N is a six element array defined as follows:

- N(1) contains the number of the display that was on the screen when PTYP was called.
- N(2) contains the option number that was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) indicates the type of input field.
- N(5) and N(6) are input field values.

Description of Function

PTYP (Figure 3-22) determines the name of the aircraft from input data, sets the default values for aircraft dependent parameters, brings up the appropriate aircraft dependent parameters display and fills in the day, flyby number, aircraft type, and default values on this display. In postanalysis, PTYP can also search the data tape to find a desired plane type.

External References

DISPIO, TIME, VREAD, and SFUN.
Figure 3-22
3.11 Calculate Velocity

Name
GETVEL

Calling Sequence
CALL GETVEL (IV, VEL)

Where:
- IV is an integer word that contains the peak velocity in bits 8-14 and the maximum velocity in bits 0-6 in counts.
- VEL is a real variable that contains the peak velocity in feet/second.

Description of Function

Subroutine GETVEL (Figure 3-23) extracts the peak velocity in counts from an integer word that contains both the peak velocity and the maximum velocity for a data point. It then converts the integer velocity in counts to a floating point velocity in feet/second according to Table 3-1.

External References
SUBBIT and FLOAT.

3.12 Find Vortex Center

Name
CENTRD

Calling Sequence
CALL CENTRD

Description of Function

CENTRD (Figure 3-24) processes raw data in an attempt to locate vortex centers. The data is checked to see if it possesses the minimum number of points that are required to define a vortex. The minimum number of points to locate a vortex center is never less than two and will be two for the first vortex of a data frame. For the second vortex the minimum is defined as C% of the number of points contained in the first vortex center. If there
CALCULATE VELOCITY

Figure 3-23
### COUNT TO VELOCITY CONVERSION

<table>
<thead>
<tr>
<th>COUNT</th>
<th>VELOCITY IN FEET PER SECOND</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \leq 69 )</td>
<td>( 1.8 \times \text{count} )</td>
</tr>
<tr>
<td>( 70 \leq 75 )</td>
<td>( 1.8 \times 69 + (\text{count}-69) \times 3.6 )</td>
</tr>
<tr>
<td>( &gt; 75 )</td>
<td>( 1.8 \times 69 + 6 \times 3.6 + (\text{count}-75) \times 7.2 )</td>
</tr>
</tbody>
</table>

**Table 3-1**
FIND VORTEX CENTER

CENTRD

Real time

Find maximum Vpeak

Y

Find points in area & compute terms for calculating vortex center

Y

Vpeak meet criteria

N

B

Increase counter for noise spikes

Vortex defined

Calculate vortex center and set descriptive parameters

Second vortex for frame

Calculate time

No noise spikes limit

Y

A

Noise spikes limit

N

A

Set descriptive parameters for missing vortex

Y

Calculate statistics

Frame number

DISPLA

Output data

N

3-25

C

Output statistics

Add to totals for statistics

RETURN

RETURN
are insufficient points, the description parameters that are used in tabular
data output for the vortices are set to indicate that no vortex center was found.
The assigned values are:

- \( \text{NCORPT}(1) = \text{NUMPTS} \)
- \( \text{NCORPT}(2) = 0 \)
- \( \text{PKVEL}(1) = \text{maximum peak velocity} \)
- \( \text{NOISE}(1) = 0 \)
- The remaining descriptive parameters are not set but will be set equal to a blank in subroutine DISPLA.

If there are sufficient points, the point with the maximum peak velocity is found, and its velocity checked to see that it has a minimum value of \( D\% \) of the maximum peak velocity of the last vortex center found (it is assumed that for the first frame that the previous velocity was zero). If the velocity is less than the minimum, the descriptive parameters are again set as shown above to indicate that no vortex center was found.

If the maximum peak velocity equals or exceeds the minimum, the points in a correlation region defined as all points within a radius \( R \) of the point with maximum velocity are located. If there are a sufficient number of points to define a vortex and if \( B\% \) of the points have a minimum velocity defined as \( A\% \) of the maximum, then a vortex center is determined using the following equation (see Table 3-2):

\[
X = \frac{\sum_{i=1}^{k} I_i \cdot V_{peak_i} \cdot X_i}{\sum_{i=1}^{k} I_i \cdot V_{peak_i}} \quad \text{and} \quad Y = \frac{\sum_{i=1}^{k} I_i \cdot V_{peak_i} \cdot Y_i}{\sum_{i=1}^{k} I_i \cdot V_{peak_i}}
\]

If the above criterion is not met, the point is rejected as a noise spike, the noise spike count is increased by one, and if the noise spike limit has not been exceeded, the process is repeated beginning with a search for the point possessing the maximum peak velocity. If the noise spike limit is exceeded, the descriptive parameters are set as shown above to indicate that no vortex center was located except \( \text{NOISE}(1) \) will be equal to the number of noise spikes and \( \text{NCORPT}(1) \) will be equal to the number of points in the correlation region. Then subroutine DISPLA will be called.

When the first vortex center is found, time is calculated, \( \text{NOISES}(1) \) is set equal to the noise spikes, \( \text{NCORPT}(1) \) is set equal to \( \text{KOUNT} \), \( \text{PKVEL}(1) \) is set equal to the maximum peak velocity in the vortex center, all of the points used in determining the location of this vortex center are rejected,
## DEFINITION OF TERMS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MNEMONIC</th>
<th>DEFINITION</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input from LDV's</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>INTENS (bits 7-14)</td>
<td>intensity</td>
<td>Counts</td>
</tr>
<tr>
<td>Vpeak</td>
<td>IVEL (bits 8-14)</td>
<td>peak velocity</td>
<td>Counts</td>
</tr>
<tr>
<td>X</td>
<td>IX</td>
<td>horizontal coordinate of point wrt LDV</td>
<td>Counts</td>
</tr>
<tr>
<td>Y</td>
<td>IY</td>
<td>vertical coordinate of point wrt LDV</td>
<td>Counts</td>
</tr>
<tr>
<td>N</td>
<td>NUMPTS</td>
<td>number of data points</td>
<td>None</td>
</tr>
</tbody>
</table>

**Variable Input**

| SYMBOL | MNEMONIC | DEFINITION | UNITS | |
|--------|----------|------------|-------|
| A | VELTOL | A defines the minimum peak velocity that B% of the points in a correlation region must possess by requiring them to have A% of the maximum peak velocity. | None | |
| B | PTTOL | B defines the percent of points in a correlation region that must possess a minimum velocity defined as A% the maximum peak velocity. | None | |
| C | VORTOL | Defines the minimum number of points needed to locate a second vortex center in a frame of data since this number is C% of the number of points which defined the first vortex center. | None | |
| D | FRVTOL | The maximum peak velocity for the first vortex center defined in a data frame must be D% of the peak velocity from the last defined vortex center. | None | |
| R | IRA DI | Radius of correlation volume | feet | |
| NS | NOISE | The maximum peak velocity for the definition of the second vortex must be separated from the maximum peak velocity of the first vortex by NS \( \times R \). | None | |

Table 3-2

-73-
### DEFINITION OF TERMS
(continued)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MNEMONIC</th>
<th>DEFINITION</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables Calculated in CENTRD</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>KOUNT</td>
<td>Number of points in a correlation region.</td>
<td>None</td>
</tr>
<tr>
<td><strong>Descriptive Parameters Determined in CENTRD</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>XCG</td>
<td>A two element array containing the horizontal distances between the vortex centers and the LDV.</td>
<td>feet</td>
</tr>
<tr>
<td>Y</td>
<td>YCG</td>
<td>A two element array containing the vertical distances between the vortex centers and the LDV</td>
<td>feet</td>
</tr>
<tr>
<td>n</td>
<td>NCORPT</td>
<td>A two element array containing the number of points in the correlation regions of the vortices.</td>
<td>None</td>
</tr>
<tr>
<td>(Vpeak)max</td>
<td>PKVEL</td>
<td>A two element array containing the maximum peak velocity for each vortex.</td>
<td>ft/sec</td>
</tr>
<tr>
<td>Θmin</td>
<td>ELANG1</td>
<td>Minimum angle at which data point was found.</td>
<td>deg.</td>
</tr>
<tr>
<td>Θmax</td>
<td>ELANG2</td>
<td>Maximum angle at which a data point was found.</td>
<td>deg.</td>
</tr>
<tr>
<td>NVORTEX</td>
<td>NVORTX</td>
<td>Number of vortices found in a frame of data.</td>
<td>None</td>
</tr>
<tr>
<td>t</td>
<td>RTIME</td>
<td>Time at which the vortex center was detected relative to time at which the first data point was detected.</td>
<td>sec.</td>
</tr>
<tr>
<td>NOISE</td>
<td>NOISES</td>
<td>A two element array containing the number of noise spikes that were encountered while locating a vortex center. Maximum number allowed is 5.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3-2
(continued)
a search using the same techniques is used to locate a second vortex center. However, when descriptive parameters are set they will have an index of 2 and there is no velocity check against the last vortex center found. When either the second vortex center has been located or it has been determined that a second vortex center cannot be defined, subroutine DISPLA is called to output the data.

External References

DISPLA, GETVEL, and SUBBIT.

3. 13  Display Output Data

Name

DISPLA

Calling Sequence

CALL DISPLA

Description of Function

Subroutine DISPLA (Figure 3-25) transforms vortex locations from LDV referenced coordinate systems to the center of runway coordinate system, encodes description parameters for output, sets the display controller buffers with the desired output format, and calls the display controller to output data. For a detailed description of output choices and output see Section 6 of this manual.

External References

DISPIO.

3. 14  Terminate Program

Name

TERM

Calling Sequence

CALL TERM (N)

Where: N is a dummy argument.
DISPLAY OUTPUT DATA

Figure 3-25

-76-
Description of Function

Subroutine TERM (Figure 3-26) prepares the data tape for program termination and returns control to the DOS Monitor. When the program is operating in real time and recording data on magnetic tape, TERM writes two consecutive end of files at the end of data. The routine then rewinds the tape for both real time and post-analysis and returns control to the DOS Monitor.

External References

SFUN, WAIT, CLOSE, and DISPIO.

3.15 Scatter Plot Generation

Name

SCAT

Calling Sequence

CALL SCAT

Description of Function

Subroutine SCAT (Figure 3-27) plots the raw data points and the vortex centers in an X-Y coordinate system for each data frame in a flyby. The character which represents each point is determined by the magnitude of the velocity for that point. The points possessing the ten highest velocities are represented by the corresponding first ten letters of the alphabet. Subsequent points are represented according to the following table:

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VELOCITY IN FT./SEC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20-30</td>
</tr>
<tr>
<td>1</td>
<td>30-40</td>
</tr>
<tr>
<td>2</td>
<td>40-50</td>
</tr>
<tr>
<td>3</td>
<td>50-60</td>
</tr>
<tr>
<td>4</td>
<td>60-70</td>
</tr>
<tr>
<td>5</td>
<td>70-80</td>
</tr>
<tr>
<td>6</td>
<td>80-90</td>
</tr>
<tr>
<td>7</td>
<td>90-100</td>
</tr>
<tr>
<td>8</td>
<td>100-110</td>
</tr>
<tr>
<td>9</td>
<td>110-120</td>
</tr>
</tbody>
</table>

External References

CENTRD, DISPIO, GETVEL, DECD, SETAD, VREAD, DBUG, and OPEN.
TERMINATE PROGRAM

Figure 3-26
SCATTER PLOT GENERATOR

Diagram:

A. Select character & plot each (X, Y) point

B. End of file

C. End flyby signal

DISPIO: Bring up plot background

DISPIO: Place header information on screen

DISPIO: Determine vortex centers

CENTRD: Determine vortex centers

SETAD: Get buffer address

DISPIO: Plot vortex centers with large V's

DECD: Encode velocities

DISPIO: Output velocities

VREAD: Read data from next frame

Auto hard copy

DISPIO: Hard copy screen

RETURN

Console switch 0 up

Output octal listing of data

Determine highest 10 velocities & store in their indices

Figure 3-27

-79-
4. DISPLAY CONTROLLER

The Display Controller coordinates all communication between the operator at the display terminal and the application program (Vortex) executing in the PDP-11 computer. In particular, the Display Controller performs the following functions:

1. processes data tablet inputs from the operator,
2. processes keyboard inputs from the operator,
3. processes information outputs from the application program (Vortex),
4. processes display control information from the Display Library, and
5. performs all input and output to the terminal.

Basically, the Display Controller is organized as two processors to perform the primary functions listed above. The two processors are:

- User Input Processor (1 and 2)
- Application Program Request Processor (3)

In addition, there are routines that are essential for the proper execution of these processors but are not a part of either one. They perform all accessing of the Display Library information and the input/output processing. They will be referred to in this section as:

- Common Utility routines (4)
- Input/Output Interrupt Processing (5)

Figure 4-1 illustrates the structure of the Display Controller and how its processors interface with external devices and the application program.

Descriptions and flowcharts of the Display Controller, as categorized above, follow.

4.1 User Input Processor

The User Input Processor (whose component block diagram is depicted in Figure 4-2) processes both data tablet inputs and keyboard inputs from the operator at the display terminal. In response to data tablet inputs, it outputs a graphic cursor that tracks the position of the data tablet cursor on the tablet. This processor also passes information to the application program regarding points selected by the operator through depression of the data tablet cursor.
USER INPUT PROCESSOR COMPONENTS

Figure 4-2
Processing keyboard inputs involves both echoing the keyed-in data back to the operator at the terminal and passing the data to the application program for further processing.

Various application program requests are referenced in this section. The processing of these requests is discussed in Section 4.2.

User Input Executive

The User Input Executive (DC (Figures 4-3 through 4-5)) initiates processing. This routine performs initialization of the DOS disk driver, data tablet input, and certain hardware addresses needed by the controller. It calls ATVWS and ONCE to perform various initialization functions for the application program. DC then brings up the first display on the screen and initiates the input search loop (LOOP1) which continues to process for the duration of execution.

When execution has been terminated, DC then releases the DOS disk driver, terminates data tablet input, and resets those hardware addresses previously initialized by it.

Process User Input

The Process User Input routine (LOOP1 (Figure 4-6)) repeatedly checks for user input from the data tablet and the keyboard which if present has been extracted from the hardware input register and stored in a table for this routine by the Input Interrupt Processor. LOOP1 searches this table for three specific types of input:

- Data tablet SUB command coordinates (data tablet cursor is in proximity of data tablet)
- Keyboard character
- Data tablet GS command (point/option selected via data tablet cursor by operator)

If input is present, it is processed by the appropriate routine. LOOP20 is then executed to refresh the screen(s). LOOP1 then determines if the application program has requested termination through the Reset Request to the controller. As long as termination has not been requested, LOOP1 repeats its search for input.

Process SUB Command

Process SUB Command routine (LOOP50 (Figure 4-7)) processes the SUB command input received from the data tablet whenever the data tablet cursor is in proximity of the tablet. This processing consists of using the coordinates that are transmitted with the SUB command to activate and build an output string that displays the graphic cursor on the screen to indicate the current position of
USER INPUT EXECUTIVE

Figure 4-3
DC INITIALIZATION

Figure 4-4

-86-
DC TERMINATION

Figure 4-5
PROCESS USER INPUT

Figure 4-6
PROCESS SUB COMMAND

LOOP50

Activate graphic cursor

Calculate beginning horizontal bar position

Calculate ending horizontal bar position

Calculate beginning vertical bar position

Calculate ending vertical bar position

Intersection position screen

Set beginning position to left edge of screen

Set beginning position to bottom edge of screen

Set beginning position to top edge of screen

Set ending position to right edge of screen

Set ending position to vertical screen bottom edge of bar position

Select primary screen

Set up graphic cursor output string

Deactivate graphic cursor

RETURN

RETURN

A

A

Figure 4-7
the data tablet cursor. The graphic cursor consists of a horizontal bar and a vertical bar intersecting at the point being tracked.

As the coordinates are being converted to the output string format, any part of the generated cursor that exceeds the range of the data tablet (and therefore the display screen) is clipped to prevent wraparound vectors from appearing on the screen. Also, if the data tablet position being tracked is outside the viewing area of the screen but still in range of the tablet (i.e., in the menu area), the graphic cursor output string is deactivated to eliminate unnecessary output to the display.

The graphic cursor is sent only to the primary screen. The primary screen is that screen so designated by the application program through an Auxiliary Screen request. The default is screen 1 (the screen with the keyboard).

Process Keyboard Input

The Process Keyboard Input (LOOP75 (Figures 4-8 through 4-11)) processes all input keyed-in by the operator at the display terminal.

Key-in field data is that data which is input in response to a key-in field designated by "_" characters in the text of the display. The positioning of the alphanumeric cursor always indicates the next available field position for such input.

There are two characters that have special meaning as keyboard input. These characters are the carriage return and the rubout.

The carriage return character indicates end of input and causes the data already keyed-in to be transmitted to the application program's designated program. This routine must have access to the key page of the Display Library for the display currently on the screen. If this 'page' is not currently in core, the LIBINP routine reads it from disk where the library resides.

If there is no keyboard input prior to the carriage return, the alphanumeric cursor is positioned at the next/first key-in field via the information in the key page. If there is prior data, it is passed to the application program's next program (specified in the key page) via ROOTEX for processing.

The input data is set up along with other control information in the format shown in Figure 4-12. Each parameter is described below:

Terminal ID: Identifies the user terminal through which the input was transmitted. This value is significant only for multi-terminal systems.

Next Program: Contains the 4-character name of the program to be executed within the application program to process the operator input.

Current Display: Contains the 4-digit decimal number of the display that is currently being presented to the operator.
PROCESS KEYBOARD INPUT

Figure 4-8
PROCESS KEY-IN FIELD CARRIAGE RETURN

Figure 4-9
POSITION ALPHANUMERIC CURSOR AT NEXT KEY-IN FIELD

LOOP91

Next key-in field info in core

More key-in fields

LIBINP
Read display library key page

Clear data buffer for application program

Calculate position of next key-in field

ALPCRS
Reposition alphanumeric cursor

KEYSET
Position alphanumeric cursor first field

RETURN

Figure 4-10
PROCESS KEY-IN FIELD RUBOUT

Figure 4-11
<table>
<thead>
<tr>
<th>Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Terminal ID</td>
</tr>
<tr>
<td>Word 2</td>
<td>Next Program</td>
</tr>
<tr>
<td>Word 3</td>
<td>Next Program</td>
</tr>
<tr>
<td>Word 4</td>
<td>Current Display</td>
</tr>
<tr>
<td>Word 5</td>
<td>Option Number</td>
</tr>
<tr>
<td>Word 6</td>
<td>Data Length</td>
</tr>
<tr>
<td>Word 7</td>
<td>Data Type</td>
</tr>
<tr>
<td>Word 8, etc.</td>
<td>Data Buffer</td>
</tr>
</tbody>
</table>
Option Number: Contains the number of the compose field or option selection within the input display associated with this transmission.

Data Length: Contains the number of characters (bytes) of data being sent to the application program.

Data Type: Designates which type of input is being transmitted to the application program.

0 - data tablet option selection
1 - keyboard compose field
2 - data tablet image design point
3 - keyboard image design character

Data Buffer: Contains the input characters (bytes) being transmitted to the application program.

If the key page indicates a next display is associated with the key-in field, NXTDSP routine is executed to bring up the display and then the next program (also indicated in the key page) is called to process the data. If a next display is not specified, the next program is called and the alphanumeric cursor is positioned at the next/first key-in field of the current display.

The rubout character indicates that the character keyed-in previously is to be ignored. If there is no previous data, the rubout is ignored. If there is data, the last character is deleted as input and the alphanumeric cursor is repositioned at the previous character position.

If the keyboard input character is neither the carriage return nor the rubout, it is saved as valid input providing the input field is not already full. The key-in field is limited to the size specified in the key page. The keyboard input character is then echoed on the screen. The character keyed-in to a key-in field is displayed in permanent store mode over the '_' character in the display and the alphanumeric cursor is positioned at the next '_'.

Process GS Command

The Process GS Command routine (LOOP58 (Figure 4-13)) processes the GS command input received from the data tablet whenever a point is selected with the data tablet cursor. The graphic cursor is deactivated and the library pen page is read into core.

The point selected must correspond to a pen option in the pen page for the input to be valid. The pen page is searched until the pen option matching the selected point coordinates is found. If a next program is specified for the selected pen option,
Figure 4-13
the parameter list is set up (Figure 4-12). If a next display is also specified, it is
brought up on the screen by the NXTDSP routine and the next program is executed
through the INVPGM routine. If a next program is not specified but a next display
is, it is brought up by NXTDSP.

Refresh Screen

The Refresh Screen routine (LOOP20 (Figure 4-14)) initiates all refresh
mode output to the screen. This includes:

- graphic cursor in response to SUB command from the data
tablet,
- refresh message in response to Refresh Message application
  program request, and
- keyboard input echo in response to keyboard command input.

4.2 Application Program Request Processor

The Application Program Request Processor (whose block diagram is
depicted in Figure 4-15) processes all display I/O requests from the application
program. There are 11 types of requests that are handled by this processor. They are:

- tabular output
- new display
- one-line message
- character plot
- vector plot
- erase screen(s)
- refresh message
- hard copy screen(s)
- auxiliary screen
- reset options
- high-speed output

The application program communicates its request to the Display Controller
through a standard parameter list as shown in Figure 4-16 with associated buffer
formats shown in Figure 4-17.

-98-
Figure 4-14
APPLICATION PROGRAM REQUEST PROCESSOR COMPONENTS

Figure 4-15
### APPLICATION PROGRAM REQUEST TO DISPLAY CONTROLLER PARAMETER FORMATS

<table>
<thead>
<tr>
<th>Request Code</th>
<th>Function</th>
<th>Parameters 1</th>
<th>Parameters 2</th>
<th>Parameters 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tabular Output</td>
<td>A(1)</td>
<td>A(data length)</td>
<td>A(buffer format A)</td>
</tr>
<tr>
<td>2</td>
<td>New Display</td>
<td>A(2)</td>
<td>A(display name)</td>
<td>A(overlay option)</td>
</tr>
<tr>
<td>3</td>
<td>One-Line Message</td>
<td>A(3)</td>
<td>A(data length)</td>
<td>A(buffer format A)</td>
</tr>
<tr>
<td>4</td>
<td>Character Plot</td>
<td>A(4)</td>
<td>A(data length)</td>
<td>A(buffer format B)</td>
</tr>
<tr>
<td>5</td>
<td>Vector Plot</td>
<td>A(5)</td>
<td>A(data length)</td>
<td>A(buffer format C)</td>
</tr>
<tr>
<td>7</td>
<td>Erase</td>
<td>A(7)</td>
<td>A(screen #)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Refresh Message</td>
<td>A(9)</td>
<td>A(data length)</td>
<td>A(buffer format B)</td>
</tr>
<tr>
<td>10</td>
<td>Hard Copy</td>
<td>A(10)</td>
<td>A(screen #)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Auxiliary Screen</td>
<td>A(11)</td>
<td>A(screen #)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reset</td>
<td>A(13)</td>
<td>A(option #)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>High-Speed Output</td>
<td>A(14)</td>
<td>A(data length)</td>
<td>A(buffer format A)</td>
</tr>
</tbody>
</table>

1. A( ) indicates address of parenthesized element.
2. See Figure 4-17 for formats.
3. Parameter 2
   - 1 = refresh message
   - 16 = terminate to DOS
   - 32 = temporarily return to DOS
   - 64 = return to Display Controller
4. All data lengths are byte lengths.
5. Parameter zeroed when output complete; must be set before each call.
6. 0 = new display replaces current display.
   -1 = new display overlays current display.
   next display = parameter 3 new display overlays parameter 2 new display which overlays current display.

Figure 4-16
PARAMETER BUFFER FORMATS

<table>
<thead>
<tr>
<th>Format A</th>
<th>Format C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char 2</td>
<td>Char 1</td>
</tr>
<tr>
<td>Char 4</td>
<td>Char 3</td>
</tr>
<tr>
<td>Char 6</td>
<td>Char 5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Format B**

<table>
<thead>
<tr>
<th>Screen #</th>
<th>Character Size</th>
<th>Initial X</th>
<th>Initial Y</th>
<th># Characters</th>
<th>A (Characters)</th>
</tr>
</thead>
</table>

**Plot 1**

1. Character = ASCII Character
2. Screen # = 1 screen 1
   2 screen 2
   3 both screens
3. Character Size = -1 for refresh vector plot message; otherwise not used.

**Plot 2**

4. X = 0-1023
5. Y = 0-800
6. Plot Type = -1 solid
7. disconnects within a plot represented by -1.
8. a plot buffer (series of x, y points and disconnects) for refresh vector plot message.

Figure 4-17
Each parameter is set up as the address of the value being passed as the parameter. For all request types the first parameter is the address of the request code number. The second and third parameters vary according to request type.

All requests from the application program for output to the display are directed to the primary screen unless a screen number is specifiable through the parameter list. The primary screen is that screen to which the displays are being directed by the controller. The default is screen 1 (the screen with the keyboard), but is changeable through the application program Auxiliary Screen Request. Additionally, all characters output will be the standard hardware size even where specifiable through the parameter list.

Display Input/Output Executive

The Display Input/Output Executive (DISPIO (Figure 4-18)) is called directly by the application program to execute 1 of the 11 requests listed above. DISPIO determines which type of request is being made and calls the appropriate routine to process it. When the request processing is complete, DISPIO returns to the application program that called it.

Tabular Output - Request Code 1

The Tabular Output routine (TABOUT (Figure 4-19)) processes requests for the output of data to the screen to areas predefined in the Display Library fill page for the display currently on the screen.

The location of the fill page for the current display within the Display Library is calculated from the Display Library index and the appropriate block is read into core from the Display Library by the LIBINP routine. Using the information in the fill page, the first fill field is set up in an output string with the data passed from the application program and sent to the display by the DSPOUT routine. This process is repeated until (1) all the data from the application program has been sent to the display or (2) all the fill fields specified in the fill page have been used. The alphanumeric cursor is then repositioned by the ALPCRS routine to the location it occupied before the tabular output was sent.

New Display - Request Code 2

The New Display routine (NEWDSP (Figure 4-20)) determines if the new display is to overlay the current display or replace it. If it is to overlay, it then determines if a third display is to overlay the second. It then brings up the appropriate displays accordingly by calling the NXTDSP routine.

One-Line Message - Request Code 3

The One-Line Message routine (ONELIN (Figure 4-21)) processes the application program request for output of a message to the left side of the bottom line of the display screen. This message is output in permanent storing mode.
DISPLAY INPUT/OUTPUT EXECUTIVE

Request code

1. TABOUT
   Tabular output
2. NEWDSP
   New display
3. ONELIN
   One line message
4. PLCHAR
   Character plot
5. PLVECT
   Vector plot
6. ERASE
   Erase screen(s)
7. REFMSG
   Refresh message
8. HARDCP
   Hard copy screen(s)
9. DCCNTL
   Control screen
10. RESET
    Reset options
11. HSOUTP
    High-speed output

Figure 4-18
Therefore, if there already was a one-line message on the screen, a fresh copy of the current display is brought up before the newly requested message is output. The DSPOUT routine is called to perform the output. The alphanumeric cursor is then repositioned by the ALPCRS routine to the position it occupied before the one-line message was output.

Character Plot - Request Code 4

The Character Plot routine (PLCHAR (Figure 4-22)) processes the application program request for a series of alphanumeric characters to be output to any specified positions on the screen in permanent storing mode.

Each series of characters specified by the application program is set up in an output string preceded by the x-y position on the screen. The DSPOUT routine is called to output the string to the terminal and the ALPCRS routine repositions the alphanumeric cursor to its position prior to output of the character plot(s).

Vector Plot - Request Code 5

The Vector Plot routine (PLVECT (Figure 4-23)) processes the application program request for a series of vectors to be drawn to any specified positions on the screen in permanent storing mode. The output string is set up as a series of x-y points. If a disconnect is indicated by the application program, a new output string is set up to begin another series of vectors. When the entire output string is set up, the DSPOUT routine is called to send the output to the screen.

Erase Screens - Request Code 7

The Erase Screens routine (ERASE (Figure 4-24)) processes the application program request to erase either or both screens. The erase command string is set up for the requested screen(s) and DSPOUT routine is called to output the command string.

Refresh Message - Request Code 9

The Refresh Message routine (REFMSG (Figure 4-25)) processes the application program request for a series of alphanumeric messages or vector plots to be output to any specified positions on the screen in non-storing or write-through mode.

The output string is set up as a series of character plots and vector plots except they are specified in write-through mode. This routine is only responsible for setting up the output string. The actual output is done by the Refresh Screen routine (LOOP20 (Figure 4-14)).
CHARACTER PLOT

PLCHAR

Set up character plot output string

Add plot characters to output string

More character plots

Y

DSPOUT
Send character plot(s)

N

ALPCRS
Restore alphanumeric cursor

RETURN

Figure 4-22
Figure 4-23
ERASE SCREENS

Figure 4-24

ERASE

Select screen(s) to be erased

DSPOUT
Send erase command

RETURN
REFRESH MESSAGE

Figure 4-25
Hard Copy Screens - Request Code 10

The Hard Copy Screens routine (HARDCP (Figure 4-26)) processes the application program request to hard copy either or both screens. The hard copy command string is set up for the requested screen(s) and the DSPOUT routine is called to output the command string.

Auxiliary Screen - Request Code 11

The Auxiliary Screen routine (DCCNTL (Figure 4-27)) processes the application program request to establish a specific screen to be the primary screen (the screen to which all display information is directed).

Reset Options - Request Code 13

The Reset Options routine (RESET (Figure 4-28)) processes the application program request to reset certain conditions established for Display Controller processing. These options are:

- Deactivate refresh messages
- Terminate execution - return to DOS
- Temporarily return to DOS
- Return to Display Controller

High-Speed Output - Request Code 14

The High-Speed Output routine (HSOUTP (Figure 4-29)) processes the application program request for output of preformatted data directly to the display screen.

If a previous output request is still being processed, the currently requested output is stacked in a Display Controller table for later output. If there is no output in progress, the output of this requested data is set up and begun. The first byte of data is moved to the output register and the output interrupt is enabled. The remainder of the data from the request is output on an interrupt basis by the OUTINT routine.

4.3 Common Routines

There are several routines that perform general functions required by both the User Input Processor (Section 4.1) and the Application Program Request Processor (Section 4.2). These routines are:

- Position Alpha Cursor - ALPCRS
Select screen(s) to be hard copied

DSPOUT
Send hard copy command

RETURN

Set primary screen

RETURN

Figure 4-26

Figure 4-27
Figure 4-29
Send Command String to Display - DSPOUT

Call Next Program - INVPGM

Read Display Library - LIBINP

Bring Up Next Display - NXTDSP

Set Up First Key-In Field - KEYSET

Position Alpha Cursor

The Position Alpha Cursor routine (ALPCRS (Figure 4-30)) sends the alphanumeric cursor to the x-y position set up in the output string by the calling program.

Send Command String to Display

The Send Command String to Display routine (DSPOUT (Figure 4-31)) processes all output requests whose output strings are formatted by the Display Controller (i.e., all output except High-Speed Output requests from the application program). This routine sets the output up as a High-Speed Output request and calls DISPIO to process it as such.

Call Next Program

The Call Next Program routine (INVPGM (Figure 4-32)) calls the application program to execute the next program indicated in the parameter list (Figure 4-12) set up by the routine that called INVPGM. INVPGM initiates the data tablet to send-point mode so that the application program processing will not be interrupted by data tablet input. INVPGM calls the Executive (ROOTEX) which in turn executes the next program.

When the next program has completed processing, INVPGM reestablishes burst mode for data tablet input.

Read Display Library

The Read Display Library routine (LIBINP (Figure 4-33)) reads the Display Library block number as set up by the routine that called LIBINP.

Bring Up Next Display

The Bring Up Next Display routine (NXTDSP (Figure 4-34)) processes the request for a display to be presented on the screen. DSPOUT is called first to erase the primary screen unless the next display is to overlay the current display. If the display to be brought up is specified as PREV, the display history table is accessed to determine the name of the previous display. If the next display
Select primary screen

Send alphanumeri
cursor

Set up parameters

High-speed output

Initiate data
tablet point mode

Call next AIDS program

Initiate data
tablet burst mode

Clear parameters
and buffer for AIDS

Figure 4-30
Figure 4-31
Figure 4-32
LIBINF

Set "wait" parameter

LKTRAN

Read display library

RETURN

Figure 4-33
BRING UP NEXT DISPLAY

Figure 4-34
is not the current display, the Display Library index is read by LIBINP and then searched for the next display entry. (If the next display is the current display, its index entry is already in core.)

Using the index, the location of the text page is determined. The text page(s) is read by LIBINP and output to the screen by DSPOUT. If a second overlay has been requested, this entire sequence is repeated.

KEYSET is called to set up for the first key-in field, and position the alphanumeric cursor accordingly. The graphic cursor, the refresh message, and the keyboard input echo are all deactivated.

**Set Up First Key-In Field**

The Set Up First Key-In Field routine (KEYSET (Figure 4-35)) processes the setup of the first key-in field in the current display (if one is present). The routine first reads in the key page for the current display from the Display Library through LIBINP. Using the key page information, this routine sets up appropriate counters and pointers and then positions the alphanumeric cursor at the beginning of the first key-in field.

4.4 **Input/Output Interrupt Processors**

The Input/Output Interrupt Processors perform all I/O to the terminal through the input/output hardware registers. They both process on an interrupt driven basis.

**Output Interrupt Processor**

The Output Interrupt Processor (OUTINT (Figure 4-36)) moves the next/first byte of output into the output register. (The High-Speed Output routine(HSOUTP (Figure 4-29)) is responsible for initiating the output process and stacking waiting requests.)

When the OUTINT routine determines that all bytes have been sent for the current request, it sets the output string length in the output requestor's area to zero to indicate completion. If another output request is waiting, it removes it from the wait stack and moves the first byte of the output string into the output register, thus initiating the next request.

All entries to this routine are through the output interrupt address set up by the Initialization routine (INIT (Figure 4-4)).

**Input Interrupt Processor**

The Input Interrupt Processor (INPINT (Figure 4-37)) processes all input received through the input register and sets it up for further processing by the User Input Processor (Section 4.1). This input includes data tablet input and keyboard input both received one byte at a time.
SET UP FIRST KEY-IN FIELD

Figure 4-35
OUTINT

Length left to output < 0

Y

Set requestor's output complete

N

Another request waiting

Y

Remove first waiting request from stack

N

RETURN

Move next byte to output register

RETURN

RETURN

Figure 4-36
INPUT INTERRUPT PROCESSOR

Figure 4-37
Data tablet input is received in a series of five bytes, a command byte followed by four coordinate bytes. The command byte is either a SUB, GS, or US. From data tablet input only the SUB command, its coordinates, and the GS command are saved for further processing. All other bytes are ignored.

If the input is not part of one of the valid five byte input strings mentioned above and determined not to be noise input, it is saved as keyboard input.

All entries to this routine are through the input interrupt address set up by the Initialization routine (INIT (Figure 4-4)).

4.5 Disk I/O Handling

The two routines in this area are called to perform all I/O to the disk.

Initialize Data Set Device

The routine (INITDS (Figure 4-38)) which performs this function issues a DOS system macro to initialize the specified device to insure that the device driver is in core for subsequent I/O operations.

Read/Write Data Set

This routine (LKTRAN (Figure 4-39)) performs the basic function of reading and writing records from and to data sets on disk. It performs error checking prior to and following the I/O function performed. This checking is done to determine if the data set to be read or written exists, if the record number to be read is valid, and if any error occurred on the I/O operation. A return parameter is set accordingly.

4.6 Executive

The Executive (ROOTEX), as shown in Figure 4-40, directs the input from the display to the appropriate application "next" program. After determining that the indicated next program is a valid program, ROOTEX gives control to the required routine.
INITIALIZE DATA SET

Figure 4-38

-124-
ROOTEX

Adjust parameter pointer to current display

Next program name valid

N

HALT

Y

XXXXXX Application program

RETURN

PTYP
DFLT
STRT
TERM
RTV
DSEL

Figure 4-40
DISPLAY LIBRARIAN

To minimize the on-line core and time requirements necessary to create each individual application program oriented display and to provide a completely general graphics capability, all displays are pre-formatted by an off-line Display Librarian. The librarian accepts card images of the text and control information defining each display and creates a "book" of displays.

The display book resides on disk and contains a display chapter for each display within the book. A display index is generated by the librarian defining the location of each display chapter within the display book. Each chapter is further sub-divided into two "pages":

- Text Page
- Control Page

The text page of each display chapter contains display text information in an expanded format consisting of embedded graphic control commands. The text page exists in a format that is ready for immediate generation on the display screen and requires no editing, scanning, or unpacking in real time. The control page is made up of the pen, keyboard, and fill pages that provide the control information needed by the real time Display Controller to respond to tablet pen and keyboard inputs and application program fill-in requests.

During real time operation the Display Controller, upon detecting a tablet pen selection or a keyboard input, uses the control information associated with the display being viewed to determine the user specified action to be taken.

The primary purpose of the Display Librarian is the creation of the preformatted display book from user defined input. To insure that the data can be correctly displayed and operated on during real time operations, it is necessary for the Display Librarian to perform extensive error checking on the user's input data prior to creating the display chapter on disk. The librarian can therefore serve as a display assembler and aid the user in defining his displays. During the processing of a display, records that contain errors are listed along with messages describing the errors. Each display must be completely free of errors before it is added to the display book on disk.
5.1 Display Book

The display book is a sequentially organized contiguous file on
disk consisting of the display index and a display chapter for each display
within the book. Figure 5-1 presents the process by which the display
book is generated and the organization of the display chapters and the
display index on disk.

5.1.1 Display Index

The display index is the first record of the display book and defines
the location of each display chapter within the display book file. The
display index is segmented into 256-word blocks; the format of the index
is presented in Figure 5-2. If more than one index record is necessary
to define the display chapters, i.e., there are more than 50 displays in
the "book," additional index records are placed at the end of the display
chapters. The first word of each index record contains the relative block
number of the next index record.

The relative block number of the display chapter is the relative
block number within the display book file of the first text page block. The
blocks within a display chapter are organized sequentially as text blocks
followed by control blocks. If any of the blocks are not required for a
display, the appropriate display index entry is set to zero. The display
name is an integer between 0001 and 9999 defined by the user's input
card.

The Display Controller reads the display index blocks, locates
the appropriate display chapter index by virtue of the display name, and
then uses the relative block number of the display chapter to access the
display within the display book file.

5.1.2 Display Chapters

The display chapters are divided into two "pages": the text page(s)
and the control page(s). The contrbl page(s) is made up of the pen, keyboard,
and fill pages. The text and control pages are segmented into 256-word
blocks.

Text Page

The text page contains the information that is to be displayed to the
operator. This information consists of embedded graphic orders, character
control orders, alphanumeric information and special symbols that have
meaning to the operator and the Display Controller. The "#" symbol defined
by the user input indicates locations where an application program may fill-in
DISPLAY INDEX FORMAT

16-bit word

<table>
<thead>
<tr>
<th>Relative Block # of Next Index Record</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Displays in This Block</td>
</tr>
<tr>
<td>Display Name</td>
</tr>
<tr>
<td>Relative Block # of First Text Page</td>
</tr>
<tr>
<td>Relative Block # of First Control Page</td>
</tr>
<tr>
<td>Pointer to Keyboard Page</td>
</tr>
<tr>
<td>Pointer to Fill Page</td>
</tr>
</tbody>
</table>

Figure 5-2
tabular data. The "#" symbol is replaced by a blank within the text page when a display is presented to eliminate the need to refresh the entire display picture when an application sends data to the screen. The data is displayed in the format initially defined by the "#" symbols. The "|" symbol indicates keyboard input areas and is replaced by an underline (_) character within the text page and display picture. The pen option areas are defined to the Librarian as the area between the two enclosing symbols "<" and ">." The characters within the symbols are displayed on the display picture and comprise one pen option area.

In addition, four character sizes are supported by the Display Librarian and character control orders are defined within the text page to display the different sizes. Any combination of the character sizes may be defined for a display and the Librarian will insure the correct spacing both horizontally and vertically. Although multiple character sizes are supported by the Librarian, they are only functional when the hardware capability exists.

Control Pages

The pen, keyboard, and fill pages are grouped together under the general category of control pages because they supply the control information used by the Display Controller during real time operation to process inputs to the respective fields.

Pen Page

The pen page of the display chapter contains the control information necessary to define the areas of the display text that may be selected with the graphics tablet pen. The format of the pen page is depicted in Figure 5-3. Each pen entry is 7-words in length consisting of:

- the X-, Y-coordinate of the first character within the pen field,
- the ΔX and ΔY of the last character within the pen field,
- the next display to be presented to the user when this pen field option is selected with the tablet pen, and
- an optional 4-character application program name to be given control when this pen field is selected.

The Display Controller determines the dimensions of the pen field from the starting X-, Y-coordinates of the first character and the ΔX and ΔY of the last character.
PEN PAGE FORMAT

16-bit word

<table>
<thead>
<tr>
<th># Pen Fields in the Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Coordinate for Start of Field</td>
</tr>
<tr>
<td>Y-Coordinate for Start of Field</td>
</tr>
<tr>
<td>Δ X to End of Field</td>
</tr>
<tr>
<td>Δ Y to End of Field</td>
</tr>
<tr>
<td>Next Display Name</td>
</tr>
</tbody>
</table>

4-Character Next Program Name

.. . .

Figure 5-3
of the last character in the field. When the tablet pen input is received by the Display Controller, its coordinates are checked against the dimensions of each pen field within the pen page to determine which option was selected. When the selected option is found, the next display and the next program associated with the pen field are displayed and executed.

Keyboard Page

The keyboard page contains the following control information for each compose field within the display text (see Figure 5-4):

- the total number and size of characters within the compose field,
- the X-, Y-coordinate of the first character within the field,
- the next display name to be presented to the operator, and
- the next program name to receive the compose data.

The Tektronix keyboard permits the user to enter alphanumeric characters into computer storage for transmission to the application program. The cursor keys on the keyboard control the compose field where the data will be placed. As each character is entered, it is displayed to the operator in one of the character slots indicated by the underline (_) character for verification and editing. After entering the data, the user presses the transmission key to pass the data to the application program associated with the compose field.

Fill Page

All areas of the display text that are available for application program tabular data output must be predefined to the Display Librarian by the special symbol ".#." The librarian constructs a fill page entry for each of these areas defining their location within the display text. Each fill entry is delimited by either a non-# symbol or a new display line. The format of the fill page is depicted in Figure 5-5.

5.2 Librarian Processing Flow

In creating the user's display book, the Display Librarian program executes five levels of processing. These five levels include:

1. Control card processing
KEYBOARD PAGE FORMAT

16-bit words

<table>
<thead>
<tr>
<th># Compose Fields in the Display</th>
<th>Character Size</th>
<th># of Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Coordinate of First Character</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y-Coordinate of First Character</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Display Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Character Next Program Name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-4
FILL PAGE FORMAT

16-bit words

<table>
<thead>
<tr>
<th># of Fields in the Display</th>
<th>Character Size</th>
<th># of Characters</th>
<th>FILL FIELD ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Coordinate of First Character</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y-Coordinate of First Character</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-5
2. Pen, compose, and line card processing

3. Text card processing

4. Output formatting

5. Cleanup

Levels two, three, and four are executed for each display being defined whereas level one is executed only when display control cards are detected in the input stream and level five only upon receiving an end-of-fbm indication. Figures 5-6 and 5-7 present a general flow of the Display Librarian from job initiation to job end.

The following subsections describe briefly and depict in flow-charts the processing done by the librarian program on each of the above mentioned levels.

5.2.1 Control Card Processing

The output option card and delete cards are processed on this level (Figure 5-8). If an output option card is received, the appropriate option flags are set; if the update option ("U") is specified, the index of the old display book is read and maintained in core. The program then returns to read the next input card.

If the control card read is a delete, the display name specified in the card is compared against the display names in the index until the display is found or all entries are checked. If the display is found, the index entry for the display is deleted and the message DISPLAY xxxx HAS BEEN DELETED FROM THE LIBRARY is written to the printer. If the display name is not in the index, the message DISPLAY xxxx NOT FOUND IN LIBRARY is written.

5.2.2 Pen, Compose, and Line Card Processing

It is on this level of processing (Figure 5-9) that the librarian program begins building the pen and key entries (Figures 5-3 and 5-4) in the display control page(s).

In processing a pen card, the names of the next display and next program are stored in the pen page. If the X-, Y-coordinate positions are present on the input record, the AX and AY are calculated and all are placed in the pen page completing the entry for the pen field. In the cases where the X-, Y-coordinates are not on the pen card, the pen
DETERMINE RECORD TYPE

Figure 5-7
CONTROL

Delete record

Y
Check for valid display name

N
Update flag set

Y
Delete display records are valid only when "U" option is specified

N
Set flag(s) for output option(s) selected

S=source listing
P=printed picture
L=create new library
U=update old library

Y
Check index for corresponding display name

N
LKTRAN Read index of library to be updated

Y
Display not found

N
Set error for display not found in library

A
Invalid display entry in index

A
RETURN

A

Figure 5-8
Figure 5-9
field entries are completed during the text processing (level 3) when the pen field characters "<" and ">" are encountered in the display text.

On receiving a compose card, the librarian saves the next display name, the next program name, and the compose field length in the keyboard page. The remaining information is supplied to the compose field entry during the text processing.

If the input record received, is a line card, the program determines the type of line being defined and compares the values specified against the limits allowed. If there is no error, it enters the specified values into either a coordinate table or a character/line table. Values entered in the coordinate tablet are not operated on again until the fourth level of processing when they are placed in the display text page in the form of vector commands. The values placed in the character/line table are converted to X-, Y-coordinate positions during text processing when the specified character number and line number are encountered.

5.2.3 Text Card Processing

During the text card processing (Figures 5-10 and 5-11), the librarian program performs three functions. These include verifying that the users text data can be successfully generated on the display screen completing the pen, key, and fill page entries and converting the vector entries in the character/line table to X-, Y-coordinates.

In determining whether the text can be generated on the screen, the librarian verifies that each text input record does not contain more characters than can fit on a single display line (i.e., 74 size 1 characters, 37 size 2 characters etc.), checks each character on the line to insure that it is displayable and verifies that the number of text lines does not exceed 35 (size 1). The pen field defining characters "<" and ">" and the change character size specifications are not included in the text character count.

On detecting a compose or fill-in character in the text, the Display Librarian stores the X-, Y-coordinates of the field in the key or fill page along with the present character size and, for fill field, the number of characters in the field. For compose fields the number of compose characters contained on the input record is compared against the value specified on the compose card to insure that there is no error in the field definition.

When pen field characters are encountered in the text, the X-, Y-coordinates are saved in the pen page along with the calculated $\Delta X$ and $\Delta Y$ to the end of the field.
TEXT RECORD PROCESSING

Figure 5-10
VERIFY TEXT RECORD CONTENTS

Figure 5-11
While the text characters are being "verified," the Librarian determines from the character/line table if a vector initiation or termination is specified for the present character/line position. If so, the character/line specification is replaced in the table, with the actual X-, Y-coordinates.

After each text record is read and verified, it is written to a scratch file on disk to be later operated upon by the output formatting routines.

5.2.4 Output Formatting

Upon receiving a display End record, the librarian enters the fourth level of display processing (Figure 5-12). It is on this level that the printed picture of the display is generated; if there are no errors in the display definition, the preformatted text pages are created and added to the display book along with the control pages.

Each text record for the display is retrieved from the librarian scratch file, formatted to effectively represent how the data will appear on the screen, and printed on the user assigned output device. If there are no errors, the display data is stored in the text page along with the necessary control commands to display on the screen. After all the text records have been retrieved, preformatted in the text pages, and written to the printer, the librarian uses the X-, Y-coordinates from both the coordinate table and character/line table and creates the display control commands to cause the generation of the specified vectors. These commands are stored following the user defined text data, thus completing the preformatted text page.

Upon completion of the text page, the Display Librarian adds the newly defined display to the display book on disk and updates the index to reflect its presence.

5.2.5 Cleanup

When attempting to read the next user input record the Display Librarian receives an end-of-job indication, this final level of processing is executed (Figure 5-13). If a new display book was created from the user's display data or an old book was updated without any of the old displays being deleted or replaced, the librarian adds the display index to the book on disk and terminates. If, however, during update of a display book, one or more of the old displays was deleted or replaced, the librarian at this time compresses the display chapters and the display index to eliminate all unused areas and then terminates.
TERMINATE PROCESSING

CLEANUP

Library created or updated

New library

Displays deleted or replaced

LKTRAN
Write index to disk

RETURN

Figure 5-13
6. PROGRAM DISPLAYS AND OUTPUT

6.1 Program Displays

The user interacts with VIDS and controls the VIDS operation by responding to a set of displays which are placed on the screen. These displays are:

- Mode Selection (see Figure 6-1)
- System Parameter Selection (see Figure 6-2)
- System Parameter Selection (Post-Analysis) (see Figure 6-3)
- Aircraft Selection (see Figure 6-4)
- Aircraft Dependent Parameters Selection (see Figure 6-5)
- Aircraft Dependent Parameters Selection (Post-Analysis) (see Figure 6-6)

6.2 Program Output

Depending on which options are selected from the above displays, data will be output in one or a combination of the following methods:

- Unprocessed data recorded on magnetic tape.
- Display of vortex information in tabular form (see Figure 6-7).
- Display of vortex positions as a function of time (see Figure 6-8).
- Display of vortex locations in an X-Y coordinate system (see Figure 6-9).
- Display of raw data with vortex centers marked (Scatter Plots) (see Figure 6-10).
- Listing of raw data.

6.2.1 Magnetic Tape Output

Data is recorded on the magnetic tape in the following format:

- One file per flyby.
- One or more fixed length records per frame (1 record for each 1000 data points or portion of 1000) of LDV data.
MODE SELECTION

REAL TIME

POST-ANALYSIS

TERMINATE PROGRAM

Figure 6-1
SYSTEM PARAMETER SELECTION

SYSTEM PARAMETERS

INITIALIZATION
DAY ___
FLY-BY NO. _____
ACQ TIME LIMIT (MIN) 0 1 2 3 4 5 NONE

DATA SELECTION
VAN 1 SELECTED INHIBITED
VAN 2
MAG TAPE RECORDING

DISPLAY SELECTION
X-Y PLOTS PRIMARY ALTERNATE
TIME BASED PLOTS
TABULAR DATA
AUTO HARD COPY YES NO

OPERATION
CONTINUOUS PARAMETER SELECTION
SYSTEM PARAMETERS
(POST-ANALYSIS)

FLY-BY SELECTION

TAPE
REWIND

DAY ___

FLY-BY NO. _____

AUTO
PROCESSING

VAN SELECTION

SELECTED
INHIBITED

VAN 1

VAN 2

DISPLAY SELECTION

PRIMARY
ALTERNATE

X-Y PLOTS

TIME-BASED PLOTS

TABULAR DATA

SCATTER PLOTS

AUTO HARD COPY

YES
NO

OPERATION

START FLY-BY

A/C DEPENDENT PARAMETERS

Figure 6-3
## AIRCRAFT SELECTION

### AIRCRAFT TYPE

<table>
<thead>
<tr>
<th>Aircraft</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-707</td>
<td>2 ENGINE JET</td>
</tr>
<tr>
<td>B-727</td>
<td>4 ENGINE JET</td>
</tr>
<tr>
<td>B-737</td>
<td>2 ENGINE PROP</td>
</tr>
<tr>
<td>B-747</td>
<td>1 ENGINE PROP</td>
</tr>
<tr>
<td>DC-02</td>
<td>OTHER</td>
</tr>
<tr>
<td>DC-09</td>
<td>ILYUSHIN</td>
</tr>
<tr>
<td>DC-10</td>
<td>YAC 40</td>
</tr>
</tbody>
</table>

- DEFAULT PLANE TYPE (POST ANALYSIS)
- RETURN TO SYSTEM PARAMETER SELECTION

Figure 6-4
<table>
<thead>
<tr>
<th>FLiBY NO.</th>
<th>DAY</th>
<th>TIME</th>
<th>A/C</th>
</tr>
</thead>
</table>

**AIRCRAFT DEPENDENT PARAMETERS**

**CORRELATION CIRCLE**

\[ R = \text{RADIUS} \]

**NOISE SPIKE FILTER**

\[ A = \% \text{V-PEAK NOISE TOLERANCE} \]

\[ B = \% \text{PTS IN TOLERANCE} \]

\[ N_S = \text{VORTEX SEPARATION} \]

**MISSING VORTEX CRITERIA**

\[ C = \% \text{POINTS FOR 2ND VORTEX} \]

\[ D = \% \text{V-PEAK FOR NEXT FRAME} \]

- **START FLY-BY (OPERATOR CONTROLLED)**
- **START FLY-BY (CONTINUOUS)**
- **UPDATE DEFAULTS**
- **RETURN TO AIRCRAFT SELECTION**

*Figure 6-5*
AIRCRAFT DEPENDENT PARAMETER SELECTION (POST-ANALYSIS)

FLY-BY NO. DAY TIME A/C

AIRCRAFT DEPENDENT PARAMETERS (POST ANALYSIS)

CORRELATION CIRCLE
R = RADIUS

NOISE SPIKE FILTER
A=% U-PEAK NOISE TOLERANCE
B=% PTS IN TOLERANCE
NS= VORTEX SEPARATION

LOST VORTEX CRITERIA
C=% POINTS FOR 2ND VORTEX
D=% U-PEAK FOR NEXT FRAME

START FLY-BY
UPDATE DEFAULTS
RETURN TO SYSTEM PARAMETER

Figure 6-6
### Display of Vortex Information

**Fly-by:** 00029  **Day:** 310  **Time:** 00:06:37  **A/C:** B-707  

<table>
<thead>
<tr>
<th>FR</th>
<th>R = 49</th>
<th>A = 50</th>
<th>B = 50</th>
<th>NS = 02</th>
<th>C = 25</th>
<th>D = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display of Vortex Information</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>COR PT</th>
<th>NOISE</th>
<th>ANGLE</th>
<th>PK</th>
<th>VEL TIME</th>
<th>PORT POS</th>
<th>STAR D POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTS</td>
<td>P N</td>
<td>M</td>
<td>P X</td>
<td>X</td>
<td>Y</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>01</td>
<td>0015</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>14</td>
</tr>
<tr>
<td>02</td>
<td>0023</td>
<td>008</td>
<td>009</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>08</td>
</tr>
<tr>
<td>03</td>
<td>0034</td>
<td>006</td>
<td>008</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>03</td>
</tr>
<tr>
<td>04</td>
<td>0039</td>
<td>009</td>
<td>004</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>07</td>
</tr>
<tr>
<td>05</td>
<td>0087</td>
<td>012</td>
<td>013</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>06</td>
<td>0089</td>
<td>007</td>
<td>008</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>0094</td>
<td>013</td>
<td>014</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>08</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>09</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>10</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>11</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>12</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>13</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>14</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>15</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>16</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>17</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>18</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>19</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>20</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>21</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
<tr>
<td>22</td>
<td>0097</td>
<td>002</td>
<td>003</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
</tr>
</tbody>
</table>

**Figure 6-7**
DISPLAY OF VORTEX POSITION

FLY-BY 00029
R = 49  A = 50  DAY 310
B = 50
VAN 1

TIME 20:07:54  A/C B-707
NS = 02  C = 25  D = 50

A = 50
Bv
50 NS
02
Du 50

A
300
VAN 1

TIME IN SECONDS
DISTANCE IN HUNDREDS OF FT

-5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5

VAN 2

Figure 6-8
DISPLAY OF VORTEX LOCATION

FLY-BY 00029  DAY 310  TIME 00:09:20  A/C B-707
R = 49  A = 50  B = 50  NS = 02  C = 25  D = 50

300
250
200
150
100
50
0

-5 -4 -3 -2 -1 0 1 2 3 4 5
HORIZONTAL POSITION FROM CENTER OF RUNWAY IN HUNDREDS OF FT

300
250
200
150
100
50
0

-5 -4 -3 -2 -1 0 1 2 3 4 5
HORIZONTAL POSITION FROM CENTER OF RUNWAY IN HUNDREDS OF FEET

Figure 6-9
SAMPLE SCATTER PLOT

FLY-BY 00029  DAY 310  TIME 00:15:50  A/C B-707
R= 69  A= 50  B= 50  NS= 02  C= 25  D= 50

FRAME NUMBER 09

VELOCITY PLOTS

DISTANCE IN FEET FROM VAN 1

A=039  B=037  C=037  D=037  E=037  F=037  G=039  H=036  I=036  J=036

Figure 6-10
Data from each LDV system is contained in separate records.

There is no fixed order of records corresponding to a given LDV system.

Each record contains 4009 integer words defined as follows:

- IFLY - flyby number.
- IFRM - frame number. A positive value indicates that the data was received from VAN 1. A negative value indicates that the data was received from VAN 2.
- ITMINT (2) - two words containing the time that the first data point was received for this frame.
- ITMEND (2) - two words containing the time that the last data point was received for this frame.
- IDAY - number of the day of the year.
- IPLN - index for plane type.
- NUMPTS - number of data points in a frame.
- IX - 1000 word array containing the X-coordinates for the data points in counts.
- IY - 1000 word array containing the Y-coordinates for the data points in counts.
- INTENS - 1000 word array containing intensity and filter information for the data points in counts. Bits 0 through 6 contain the number of filters and bits 7 through 14 contain the intensity of the point.
- IVEL - 1000 word array containing velocity information for the data points in counts. Bits 0 through 6 contain the maximum velocity. Bits 8 through 14 contain the peak velocity.

6.2.2 Time-Based Plots

Time-based plots (see Figure 6-8) are generated according to the following format:

- The screen is divided into four quadriles.
- VAN 1 data is displayed in the top quadriles.
VAN 2 data is displayed in the bottom quadriles.

Height of the vortex centroid as a function of time is displayed in the left quadriles.

Horizontal location of the vortex centroid as a function of time is displayed in the right quadriles.

The maximum time that may be displayed is 90 seconds.

An "*" is used to represent the port vortex.

An "o" is used to represent the starboard vortex.

An "s" is used to represent a single vortex.

6.2.3 X-Y Plots

X-Y plots (see Figure 6-9) are generated according to the following format:

VAN 1 data is displayed on the top half of the screen.

VAN 2 data is displayed on the bottom half of the screen.

The vortex centroid is located in an X-Y coordinate system with frame 1 data represented by an A, frame 2 data by a B, etc., for the first 26 frames. After 26 frames the cycle is repeated.

6.2.4 Tabular Data

Tabular data (see Figure 6-7) is displayed according to the following format:

VAN 1 data is displayed on the left half of the screen.

VAN 2 data is displayed on the right half of the screen.

A frame of data is entered as a blank line if there are 0 data points or more than 1000 data points.

The headings of the columns are:

FR - 2-digit frame number. If the frame number is greater than 99, only the right 2 digits of the number are displayed.

DATA PTS - number of points contained in this frame of data.
COR PT - number of data points in the correlation area for a vortex. P represents port and S represents starboard.

NOISE - the number of noise spikes which were found in processing the vortex information. P represents the number found while processing the port vortex and S represents the number found while processing the starboard vortex.

ANGLE - MN represents the minimum angle at which a data point was found during the scan. MX represents the maximum angle at which a data point was found during the scan.

PK VEL - P represents the peak velocity found in the port vortex. S represents the peak velocity found in the stbboard vortex.

TIME - represents the time at which the vortex centroid was found with respect to the time that the first data point was received from this flyby.

PORT POS - gives the X and Y centroid values for the port vortex.

STARB POS - gives the X and Y centroid values for the starboard vortex.

6.2.5 Scatter Plots

Scatter plots (see Figure 6-10) display raw data points showing their location in an X-Y coordinate system and their velocities within ten ft./sec. and also show the vortex centroid. The points are represented by numerals 0-9 with the following velocity correspondence:

-0 20-30 ft./sec.
-1 30-40 ft./sec.
-2 40-50 ft./sec.
-3 50-60 ft./sec.
-4 60-70 ft./sec.
-5 70-80 ft./sec.
-6 80-90 ft./sec.
-7 90-100 ft./sec.
-8 100-110 ft./sec.
-9 110-120 ft./sec.

The points having the ten highest values are represented by the corresponding first ten letters of the alphabet and their values are printed on the bottom of the display.

The centroids are represented by two large V's.
APPENDIX A

ATVWS LISTINGS
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>DISABLE INPUT</td>
</tr>
<tr>
<td>6</td>
<td>INITIALIZE BUFFERS</td>
</tr>
</tbody>
</table>
1. NLIST TTMM
2. TITLE AIIJIS
3. MCALL . PARAM. EX IT, AI TR
4. GLOBL LOOPZ
5. GLOBL DISPS
6. GLOBL LOCK
7. GLOBL FTOR, INIT1, DISAB
8. GLOBL FIRSTA
9. GLOBL STORES
10. GLOBL WADR, FWRIT, NEXT, SECNO, LSECT
11. GLOBL BUF
12. GLOBL BUFSHT, BUFST, FRAMES, SFLAGA
13. GLOBL INTRAB, I I TAA
14. GLOBL HEADR
15. GLOBL FILN, BUF1
16. GLOBL ATUWLS, HALT, BUSY
17. GLOBL INITDS, LKTRN, LHNBKL
18. GLOBL I TAW, INTBW
19. GLOBL INTBA, INTBB
20. GLOBL DISABL, IFLD
21. GLOBL FCNTR, FCNTW
22. GLOBL LRSEC
23. GLOBL INIT
24. GLOBL FURST
25. GLOBL STORA, STORB
26. GLOBL INITBF, FIN, SAW, WRITE
27. pH -000000
28. ;
29. l
30. .MACRO
31. PUSH ; SS AVE REGISTERS
32. MOU RO, -(SP)
33. MOV R1, -(SP)
34. MOV R2, -(SP)
35. MOV R3, -(SP)
36. MOV R4, -(SP)
37. MOV RS, -(SP)
38. ENDM
39. 38 .MACRO POP ; RESTORE REGISTERS
40. MOV (SP)+, RS
41. MOV (SP)+, R4
42. MOV (SP)+, R3
43. MOV (SP)+, R2
44. MOV (SP)+, R1
45. MOV (SP)+, RO
46. ENDM
ATUWS

MACRO VR05-01A 01-JAN-72 00:05 PAGE 2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

>:DR-11 DEFINITIONS
*: OBUFA=167772
*: SREGR=167770
*: VECTA= 300
*: LEULA= 300
*: OBUFB=167762
*: SREGB=167760
*: VECTB= 310
*: LEULB= 300
*: IBUFC = 167754
*: SREGC = 167750
*: VECTC = 320
*: LEULC= 300
*: LEUS = 100
*: IWRD = 140
*: IWRDC= 100
*: CNT = 0
*: BNST = 2
*: BB = 200
*: DISABA = 304
*: DISABB = 314

:DR-11 A OUTPUT BUFFER
:DR-11 A STATUS REGISTER
:INTERRUPT VECTOR FOR DR-11 A
:INTERRUPT LEVEL FOR DR-11 A
:DR-11 B OUTPUT BUFFER
:DR-11 B STATUS REGISTER
:INTERRUPT VECTOR FOR DR-11 B
:INTERRUPT LEVEL FOR DR-11 B
:DR-11 C INPUT BUFFER
:DR-11 C OUTPUT BUFFER
:DR-11 C STATUS REGISTER
:INTERRUPT VECTOR FOR DR-11 C
:INTERRUPT LEVEL (6) FOR DR-11 C

:ENABLE A AND B INTERRUPTS
:ENABLE A INTERRUPT DR-11 C
:WORD COUNT
:BLOCK CONTROL WORD
:BUFFER BUSY
SET UP INTERRUPT VECTORS FOR DR-11A AND B

000000 012700 000000

MOV #U'ECTA,R0 ;INITIALIZE INTERRUPT VECTOR A

MOU #INTAA,(R0)+ ;LOCATION FOR REQUEST A, DR-11 A

MOU #LEUL4,(R0)+  ;PSW

MOU #INTBA,(R0)+ ;LOCATION FOR REQUEST B, DR-11 A

MOU #LEUL2,(R0)+  ;PSW

MOU #INTBC,R0 ;INITIALIZE INTERRUPT VECTOR B

MOV #INTAB,(R0)+ ;LOCATION FOR REQUEST A, DR-11 B

MOV #LEUL3,(R0)+ ;PSW

MOV #INTBB,(R0)+ ;LOCATION FOR REQUEST B, DR-11 B

MOV #LEUL1,(R0)+  ;PSW

MOV #UECTD,RO  ;INITIALIZE INTERRUPT VECTOR C

MOV #INTCA,(R0)+ ;REQUEST A, DR-11 C

MOV #LEUL5,(R0)+ ;PSW

MOV #INTCB,(R0)+ ;REQUEST B, DR-11 C

MOV #LEUL5,(R0)+  ;PSW

JSR RS,INITDS

JSR RS,LKTRAN

MNO IBUFC,KPLN ;READ PLANE ID

MOU KPLN,HEADR ;SET PLANE ID IN BLOCK HEADER

NMO #1,KSTFLY ;SET START FLAG (0=WAIT, 1=START)

RTI

INTCA:

JSR RS,DISABL ;INHIBIT DATA FOR THIS FLYBY

JSR IBUF,KPLN ;READ PLANE ID

MOV KPLN,HEADR ;SET PLANE ID IN BLOCK HEADER

MOV #1,KSTFLY ;SET START FLAG (0=WAIT, 1=START)

RETURN

INTCB:

NOP

NOP

RTI

HALT:

JSR RS,DISABL ;GO STOP INPUT
MACRO UR05-01A 01-JAN-72 00:05 PAGE 3+

55 000210 012767 000100 177776' MOV #REV5,PSW
56 000216 HLT1: MOU MSQAD(R0),MCS1
57 000216 016057 001046' 000046 MOV #2,MTWO
58 000224 012767 000002 000740 JSR RS,DISP1O
59 000226 000000G ER HLT5
60 000240 001166' .WORD MCODE4
61 000242 001172' .WORD MTWO
62 000244 001174' .WORD MBIG
63 000246 004567 000006G JSR RS,DISP1O
64 000249 001166' .WORD MCODE3
65 000252 000040 ER HLT4
66 000254 001154' .WORD MBLK
67 000256 001170' .WORD MCODE3
68 000258 004567 000006G JSR RS,DISP1O
69 000260 001150' .WORD MCODE1
70 000262 001160' .WORD MCODE1
71 000264 004567 000006G JSR RS,DISP1O
72 000266 001200' .WORD MCODE1
73 000268 001200' .WORD MCODE1
74 000270 001060' .WORD MCODE1
75 000272 001060' .WORD MCODE1
76 000274 HLT2: .WORD MCODE1
77 000274 004567 000006G JSR RS,DISP1O
78 000277 000030 ER HLT3
79 000279 001152' .WORD MCODE2
80 000281 001152' .WORD MCODE2
81 000283 000030 ER HLT3
82 000286 001204' .WORD MCODE3
83 000288 001204' .WORD MCODE3
84 000290 001176' .WORD MCODE3
85 000292 001176' .WORD MCODE3
86 000294 001176' .WORD MCODE3
87 000296 001176' .WORD MCODE3
88 000300 001167' .WORD MCODE3
89 000302 001167' .WORD MCODE3
90 000304 001167' .WORD MCODE3
91 000306 001167' .WORD MCODE3
92 000308 001167' .WORD MCODE3
93 000310 HLT3: JMP LOOP2
* * * INITIALIZE DR-11 FOR INPUT (A AND/OR B)  
* * *

INIT:  

PUSH  
CLR FCNTR  
CLR FCN1  
CLR FIN  
JSR PC, INITBF  
MOV IWRD, SREGA  
CLR OBUFRA  
MOV #IWRD, SREGB  
CLR OBUFB  
MOV #IWRD, SREGC  
CLR KSTFLY  
MOV #IWRDC, SREGC  
ENABLE A INTERRUPT FOR DR-11 C  

INIT2:  

MOV #IWRD, SREGA  
CLR OBUFRA  
MOV #IWRD, SREGB  
CLR OBUFB  
BR INIT4  

INIT3:  
DEC R1  
BEQ INIT2  
MOV #IWRD, SREGA  
CLR OBUFRA  

INIT4:  
POP  
FTS RS  

INIT1:  
MOV IBUFC, KPLN  
CLR KSTFLY  
MOV #IWRDC, SREGC  

* * *
DISABLE INPUT

1
2
3
4
5 000462
6 000462
7 000476 012700 000304
8 000502 012710 00000G
9 000506 012700 000314
10 000512 012710 00000G
11 000516 005267 00000G
12 000522 005067 167770' INC SREGC
13 000526 005067 167760' CLR SREGC
14 000532 000006
15 000546 000205
16 000550 000507 167750' CLR SREGC
17 000554 000006
18 000556 000000

;SBTL DISAB:  ;SAVE REGISTERS
;** INHIBIT DATA INPUT
;** DISABL:
PUSH ;SAVE REGISTERS
MOV $DISARA,R0 ;DISARM DR-11 A
MOV $INTAW,(R0)
MOV $DISARB,R0 ;DISARM DR-11 B
MOV $INTBW,(R0)
INC FIN ;SET TEST OVER FLAG
CLR SREGA ;CLEAR ENABLE DR-11 A
CLR SREGB ;CLEAR ENABLE DR-11 B
POP RTS $5 ;RESTORE REGISTERS ;RETURN

;** DISABLE INTERRUPT FOR DR-11 C
;** DISAB1:
CLR SREGC RTS $5

;** BLOCK: .WORD 0
**ATVWS**

**MACRO**

**VROS-O1A**

**01--JAN-72**

**00:05**

**PAGE**

**G**

**INITIALIZE BUFFERS**

### .SBTL INITIALIZE BUFFERS

1. **INITIALIZE BUFFERS (R2=0,LDV1 R2=2,LDV2)**

2. **SAVE REGISTERS**

3. **PICK UP BUFFER COUNT**

4. **SET BUFFER START ADDRESSES**

5. **FREE ALL BUFFERS**

6. **CLEAR WORD COUNT**

7. **FREE ALL BUFFERS**

8. **CLEAR BUFFER SPACE**

9. **CLEAR INITIAL BUFFER**

10. **CLEAR BUSY FLAG**

11. **CLEAR LDV1**

12. **CLEAR LDV2**

13. **CLEAR STORES**

14. **CLEAR FRAMES**

15. **CLEAR NEXT**

16. **CLEAR FURST**

17. **CLEAR SECNO**

18. **CLEAR LSECT**

19. **CLEAR LRSEC**

20. **POP**

21. **RETURN**

---

**ERROR MESSAGES**

---

**MSGAD**: .WORD MSG1

**MSG2**: .WORD MSG3

**MSG3**: .WORD MSG4
ATWS MICRO V05-01A 01-JAN-72 00:05 PAGE 6

INITIALIZE BUFFERS

58
59 001050 117 125 124 MSG1: .ASCII /OUT OF BUFFERS /
001053 040 117 106
001056 040 102 106
001071 102 123 040
001074 122 123 040
001077 040
60 001100 104 111 123 MSG2: .ASCII /DISK ERROR WRITE/
001103 113 040 106
001106 122 122 117
001111 122 040 127
001114 122 111 124
001117 105
61 001120 104 111 123 MSG3: .ASCII /DISK ERROR READ /
001123 113 040 106
001126 122 122 117
001131 122 040 122
001134 105 101 104
001137 040
62 001140 111 116 111 MSG4: .ASCII /INITIALIZE FILE /
001143 124 111 101
001146 114 111 130
001151 105 040 106
001154 111 114 108
001157 040

63
64 001160 000002
65 001162 000002
66 001164 000007
67 001166 000016
68 001170 000001
69 001172 000002
70 001174 034033
71 001176 177777
72 001200 000020
73 001202 000000
74 001204 000001
75 000000 000000
76 000000 000000
77 000000 000000
78 000000 000000
79 000000 000000

64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79

-172-

ORIGINAL PAGE IS OF POOR QUALITY.
## SYMBOL TABLE

<table>
<thead>
<tr>
<th>ATUS</th>
<th>000000R</th>
<th>ATUV1</th>
<th>000174R</th>
<th>BB</th>
<th>000200</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>000566R</td>
<td>CNCL</td>
<td>000100R</td>
<td>DISA</td>
<td>000304</td>
</tr>
<tr>
<td>BUFST</td>
<td>XXXXXG</td>
<td>CNT</td>
<td>000000</td>
<td>DISAB</td>
<td>000562R</td>
</tr>
<tr>
<td>BUSY</td>
<td>XXXXXG</td>
<td>DISAP</td>
<td>000314</td>
<td>DISP1</td>
<td>000562R</td>
</tr>
<tr>
<td>CNLIC</td>
<td>000100R</td>
<td>FCNT</td>
<td>XXXXXG</td>
<td>FCNTW</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>DISARP</td>
<td>000304</td>
<td>FILN</td>
<td>XXXXXG</td>
<td>FILE5</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>FCTR</td>
<td>XXXXXG</td>
<td>FLIT</td>
<td>XXXXXG</td>
<td>FLIT3</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>HALT</td>
<td>000204R</td>
<td>HLT2</td>
<td>000274R</td>
<td>HLT3</td>
<td>000310R</td>
</tr>
<tr>
<td>HLT5</td>
<td>000246R</td>
<td>ICHFB</td>
<td>167754</td>
<td>IFLD</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>INIT</td>
<td>000314R</td>
<td>INITB</td>
<td>000560R</td>
<td>INIT1</td>
<td>000440R</td>
</tr>
<tr>
<td>INITDS</td>
<td>XXXXXG</td>
<td>INIT2</td>
<td>000370R</td>
<td>INIT5</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>INIT3</td>
<td>000404R</td>
<td>INIT4</td>
<td>000422R</td>
<td>INITA</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>INITAB</td>
<td>XXXXXG</td>
<td>INTA4</td>
<td>XXXXXG</td>
<td>INTAB</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>INIT4</td>
<td>000404R</td>
<td>INTB4</td>
<td>XXXXXG</td>
<td>INTCB</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>INTP</td>
<td>XXXXXG</td>
<td>INTD</td>
<td>XXXXXG</td>
<td>KFNL</td>
<td>000002R</td>
</tr>
<tr>
<td>INTF</td>
<td>XXXXXG</td>
<td>KFNL3</td>
<td>000300</td>
<td>LEUBL</td>
<td>000002R</td>
</tr>
<tr>
<td>INTG</td>
<td>XXXXXG</td>
<td>KFNL4</td>
<td>000300</td>
<td>LEUS</td>
<td>000002R</td>
</tr>
<tr>
<td>INTAB</td>
<td>XXXXXG</td>
<td>LKIR4</td>
<td>XXXXXG</td>
<td>LKIR6</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>LOOP2</td>
<td>XXXXXG</td>
<td>LOOP2</td>
<td>XXXXXG</td>
<td>LISTC</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>MB16</td>
<td>001174R</td>
<td>MB16</td>
<td>001174R</td>
<td>MB2</td>
<td>001174R</td>
</tr>
<tr>
<td>MODE2</td>
<td>001162R</td>
<td>MDISP</td>
<td>001204R</td>
<td>MDISP</td>
<td>001204R</td>
</tr>
<tr>
<td>MCS1</td>
<td>000272R</td>
<td>MDISP</td>
<td>001204R</td>
<td>MDISP</td>
<td>001204R</td>
</tr>
<tr>
<td>MOD1</td>
<td>001162R</td>
<td>MODE1</td>
<td>001164R</td>
<td>MODE2</td>
<td>001164R</td>
</tr>
<tr>
<td>MS1</td>
<td>000608R</td>
<td>MONE</td>
<td>001176R</td>
<td>MONE</td>
<td>001176R</td>
</tr>
<tr>
<td>MSG1</td>
<td>000608R</td>
<td>MSG1</td>
<td>001100R</td>
<td>MSG1</td>
<td>001100R</td>
</tr>
<tr>
<td>MSG2</td>
<td>000608R</td>
<td>MSG2</td>
<td>001100R</td>
<td>MSG2</td>
<td>001100R</td>
</tr>
<tr>
<td>MS3</td>
<td>000608R</td>
<td>MSG3</td>
<td>001100R</td>
<td>MSG3</td>
<td>001100R</td>
</tr>
<tr>
<td>MS4</td>
<td>000608R</td>
<td>MS4</td>
<td>001100R</td>
<td>MS4</td>
<td>001100R</td>
</tr>
<tr>
<td>OBFA</td>
<td>016772</td>
<td>OBUF</td>
<td>016772</td>
<td>OBUF2</td>
<td>016772</td>
</tr>
<tr>
<td>FC</td>
<td>0000007</td>
<td>PSW</td>
<td>177776</td>
<td>R0</td>
<td>0000000</td>
</tr>
<tr>
<td>R1</td>
<td>0000001</td>
<td>R1</td>
<td>0000002</td>
<td>R2</td>
<td>0000002</td>
</tr>
<tr>
<td>R2</td>
<td>0000004</td>
<td>R2</td>
<td>0000005</td>
<td>R3</td>
<td>0000005</td>
</tr>
<tr>
<td>R3</td>
<td>0000007</td>
<td>R3</td>
<td>0000008</td>
<td>R4</td>
<td>0000008</td>
</tr>
<tr>
<td>SFLG1</td>
<td>XXXXXG</td>
<td>SP</td>
<td>0000006</td>
<td>SREGA</td>
<td>167770</td>
</tr>
<tr>
<td>SREGA</td>
<td>016770</td>
<td>SREGA</td>
<td>016770</td>
<td>SREGA</td>
<td>016770</td>
</tr>
<tr>
<td>STOREA</td>
<td>XXXXXG</td>
<td>STOREA</td>
<td>XXXXXG</td>
<td>STOREA</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>VECTA</td>
<td>000300</td>
<td>VECTA</td>
<td>000300</td>
<td>VECTA</td>
<td>000300</td>
</tr>
<tr>
<td>WRD</td>
<td>XXXXXG</td>
<td>WRD</td>
<td>XXXXXG</td>
<td>WRD</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>SFLAGA</td>
<td>XXXXXG</td>
<td>SFLAGA</td>
<td>XXXXXG</td>
<td>SFLAGA</td>
<td>XXXXXG</td>
</tr>
<tr>
<td>SREGA</td>
<td>001206</td>
<td>SREGA</td>
<td>001206</td>
<td>SREGA</td>
<td>001206</td>
</tr>
<tr>
<td>ADDR</td>
<td>000000</td>
<td>ADDR</td>
<td>000000</td>
<td>ADDR</td>
<td>000000</td>
</tr>
</tbody>
</table>

**ERRORS DETECTED:** 0  
**FREE CORE:** 11635 WORDS  
**ATUS=13<ATUS.2**
APPENDIX B

READ LISTINGS
# Table of Contents

4- 1  REQUEST A (END OF FRAME)
5- 1  READ DATA FROM DR-11
6- 1  SWITCH AND WRITE
7- 1  WRITE BUFFERS TO DISK
8- 1  WRITE BLOCK NO.
9- 1  INPUT BUFFERS
10- 1  CONSTANTS
.MACRO PUSH
MOV R0, -(SP)
MOV R1, -(SP)
MOV R2, -(SP)
MOV R3, -(SP)
MOV R4, -(SP)
MOV R5, -(SP)
.ENDM

;SAVE REGISTERS

;RESTORE REGISTERS

.MACRO POP
MOV (SP)+, R0
MOV (SP)+, R1
MOV (SP)+, R2
MOV (SP)+, R3
MOV (SP)+, R4
MOV (SP)+, R5
.ENDM
READ DATA FROM DR-11

;SBTTL READ DATA FROM DR-11

10000026 012767 000004 023275
10000234 MOV #4,CNTRA ;SET WORD COUNT
10000234 IBUFA,DATA ;BRING IN DATA WORDS AND DISCARD
10000246 DEC CNTRA
10000250 RETI ;EXIT

READ DATA FROM DR-11 A

10000276 012767 000034 177776' MOV #LEUX,PSW ;SAVE REGISTER (R0-R4)
10000304 PUSH STORA.R1 ;PICK UP STORE ADDRESS POINTER
10000346 MOU #4,R2 ;SET WORD COUNTER N
10000366 INTBA1: MOV IBUFA.(RI)+ ;BRING IN DATA WORD N
10000386 DEC R2 ;N=N+1 FINISHED
10000390 BNE INTBA1

10000390 016761 006244' 000010 MOV TImX1,TIMS1(RI) ;SET START TIME
100003A8 016761 006246' 000012 MOU TIMX2,TIMR1(RI)
100003BA 005272 023514' INC @BUFSRT(R2) ;INCREMENT WORD COUNT
100003C6 000076 CMP @BUFSAT(R2),#DATNO ;BUFFER FULL
100003D4 103402 BLO INTBA3 ;NO, GO EXIT
100003D4 W05S767 023036 TST REQUST ~ RHY REQUESRS FOR WRITES
100003FF 001416 BEQ INTBA7 ; NO, GO EXIT
10000400 005767 023014 TST BUSY ;SWITCH BUFFERS AND WRITE FULL
10000416 080413 Jsr PC,SW
10000426 000130 INTBA3: TST REQUEST ;ANY REQUESTS FOR WRITES
10000436 091416 RETI ;EXIT
READ DATA FROM LDV 1 AND 2
MACRO VR05-01A 01-JAN-72 00:13 PAGE 6

SWITCH AND WRITE

1 READ DATA FROM LDU I AND 2

2 MACRO UROS-01A

3 01-JAN-72

4 00 13 PAGE 6

5

6 SWITCH AND WRITE

7 *.

8 *.

9 *.

10 *.

11 *.

12 *.

13 *.

14 *.

15 *.

16 *.

17 *.

18 *.

19 *.

20 *.

21 *.

22 *.

23 *.

24 *.

25 *.

26 *.

27 *.

28 *.

29 *.

30 *.

31 *.

32 *.

33 *.

34 *.

35 *.

36 *.

37 *.

38 *.

39 *.

40 *.

41 *.

42 *.

43 *.

44 *.

45 *.

46 *.

47 *.

48 *.

49 *.

50 *.

51 *.

52 *.

53 *.

54 *.

55 *.

56 *.

57 *.

58 *.

59 *.

60 *.

61 *.

62 *.

63 *.

64 *.

65 *.

66 *.

67 *.

68 *.

69 *.

70 *.

71 *.

72 *.

73 *.

74 *.

75 *.

76 *.

77 *.

78 *.

79 *.

80 *.

81 *.

82 *.

83 *.

84 *.

85 *.

86 *.

87 *.

88 *.

89 *.

90 *.

91 *.

92 *.

93 *.

94 *.

95 *.

96 *.

97 *.

98 *.

99 *.

100 *.

101 *.

102 *.

103 *.

104 *.

105 *.

106 *.

107 *.

108 *.

109 *.

110 *.

111 *.

112 *.

113 *.

114 *.

115 *.

116 *.

117 *.

118 *.

119 *.

120 *.

121 *.

122 *.

123 *.

124 *.

125 *.

126 *.

127 *.

128 *.

129 *.

130 *.

131 *.

132 *.

133 *.

134 *.

135 *.

136 *.

137 *.

138 *.

139 *.

140 *.

141 *.

142 *.

143 *.

144 *.

145 *.

146 *.

147 *.

148 *.

149 *.

150 *.

151 *.

152 *.

153 *.

154 *.

155 *.

156 *.

157 *.

158 *.

159 *.

160 *.

161 *.

162 *.

163 *.

164 *.

165 *.

166 *.

167 *.

168 *.

169 *.

170 *.

171 *.

172 *.

173 *.

174 *.

175 *.

176 *.

177 *.

178 *.

179 *.

180 *.

181 *.
READ DATA FROM LDV 1 AND 2
WRITE BUFFERS TO DISK

;SBTL: WRITE BUFFERS TO DISK
*WRITE:
    MOV R2,R0

*WRIT1:
    TST FWRIT
    BEQ WRIT10

*HAS WRITE BEEN REQUESTED
*NO

*ERROR IN LAST WRITE

*WRIT2:
    MOV #4,R0
    JMP #HALT

*RO

*WRIT3:
    MOV #2,R0
    SUB R0,RO

*ALL LOOPS FINISHED
*NO, TRY NEXT

*WRIT4:
    SUB $000000,RO

*LDV 1 OR 2
READ DATA FROM LDV 1 AND 2

WRITE BUFFERS TO DISK

58 001140 001405
** BEQ WRIT3
59 001142 005763 000000
** TST CNT(R3)
60 001146 100356
** BPI WRIT2
61 001150 000167 000005
** JMP WRIT5
62 001154 005763 000000
** TST CNT(R3)
63 001158 100751
** WRIT9:
64 00115d 001140 001405
** BEQ WRIT9:
65 00115d 001146 100356
** BPI WRIT5:
66 00115d 000167 000005
** WRIT:
67 00115d 100751
** WRIT:
68 001162 016005 023462'
** WRIT:
69 001166 001410
** BEQ WRITSA
70 001167 005305
** BEQ WRITSA
71 001167 001406
** BEQ WRITSA
72 001167 005205
** INC RS
73 001167 166305 000006
** SUB FNNP(R3),RS
74 001170 002705 000002
** ADD #2,RS
75 001170 001336
** BNE WRIT6
76 001170 010367 000014
** MOU R3,WADR:
77 001170 005067 000000
** CLR CNTW:
78 001174 002362'
** WRITSA:
79 00117d 001763 000000
** MOU FNNP(R3),NEXT(R0)
80 00117d 001763 000000
** MOU FNNP(R3),NEXT(R0)
81 00117d 000014
** MOU FNNP(R3),SECNO
82 00117d 004567 000000
** CLR CNTW:
83 00117d 036327
** DEC REQUEST
84 00117d 000000
** CLR LOCK:
85 00117d 000002
** RTS PC
86 00117d 000000
** RTS PC
87 00117d 005307 000000
** RTS PC
88 00117d 000000
** RTS PC
89 00117d 012176 000001 022176
** MOU #1,WRIT
90 00117d 003327 000002 000040
** BIT BNET(R3),#EOF
91 00117d 001404
** BEQ WRIT5:
92 00117d 002567 000000
** INC FTOR(R3):
93 00117d 002567 000000
** INC FCNTW:
94 00117d 005367 022170
** WRIT:
95 00117d 000000
** WRIT:
96 00117d 000002
** RTS PC
97 00117d 000002
** RTS PC
98 00117d 000000
** RTS PC
99 00117d 012176 000001 022176
** MOU #1,WRIT
READ DATA FROM LDV 1 AND 2

WRITE BLOCK NO.

MACRO VR05-01A 01-JAN-72 00:13 PAGE 8

1 SBTL WRITE BLOCK NO.

; K
; K
; K
; K
; K
; K

CALBLK:

8 001314 005750 023524
9 001320 001403
10 001322 002750 000002 023466
11 001330 CALI: TST FURST(R0)
12 001332 000107 000207 RTS PC
13 001334 015004 023466
14 001336 000207 ADD #2, LSECT(R0)

001314 001320 001322 001330 001334

001314 001320 001322 001330 001334

005750 001403 002750 000002 015004

005750 001403 002750 000002 015004

023524 023466 023466

023524 023466 023466

023524 023466 023466
READ DATA FROM LDV 1 AND 2

INPUT BUFFERS

1. SBITL INPUT BUFFERS

2. DATA INPUT BUFFERS

LOCK: .WORD 0

WRITE LOCK (0=FREE, 2=LOCKED)

STOR: .WORD 0

CURRENT STORE ADDRESS FOR LDV 1

STORE: .WORD 0

CURRENT STORE ADDRESS FOR LDV 2

BUFST: .WORD IEA1A

BUFFER START ADDRESSES

STORES: .WORD ISA1A

BUFFER STORE ADDRESSES

BUF: .WORD 18

IEA1A: .BLKW SIZE

IEA2A: .BLKW SIZE

IEA1B: .BLKW SIZE

IEA2B: .BLKW SIZE

IEB1A: .BLKW SIZE

IEB2A: .BLKW SIZE

IEB1B: .BLKW SIZE

IEB2B: .BLKW SIZE

ISC1A: .BLKW SIZE

ISC2A: .BLKW SIZE
READ DATA FROM LDV 1 AND 2

INPUT BUFFERS

ISA3A: .BLKW SIZE D

IEA4A: .BLKW SIZE B

ISA4A: .BLKW SIZE D

IEA1B: .BLKW SIZE B

ISA1B: .BLKW SIZE D

IEA2B: .BLKW SIZE B

ISA2B: .BLKW SIZE D

IEA3B: .BLKW SIZE B

ISA3B: .BLKW SIZE D

IEA4B: .BLKW SIZE B

ISA4B: .BLKW SIZE D

IEB1A: .BLKW SIZE B

ISB1A: .BLKW SIZE D

IEB2A: .BLKW SIZE B

ISB2A: .BLKW SIZE D

IEB3A: .BLKW SIZE B

ISB3A: .BLKW SIZE D

IEB4A: .BLKW SIZE B

ISB4A: .BLKW SIZE D

IEB1B: .BLKW SIZE B

ISB1B: .BLKW SIZE D

IEB2B: .BLKW SIZE B

ISB2B: .BLKW SIZE D

IEB3B: .BLKW SIZE B

ISB3B: .BLKW SIZE D

IEB4B: .BLKW SIZE B

ISB4B: .BLKW SIZE D

ISC1A: .BLKW SIZE B

ISC1A: .BLKW SIZE D

ISC2A: .BLKW SIZE B

ISC2A: .BLKW SIZE D
<table>
<thead>
<tr>
<th>1</th>
<th>023456</th>
<th>000000</th>
<th>WRIT:</th>
<th>.WORD</th>
<th>0</th>
<th>WRITE BUSY FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>023460</td>
<td>000000</td>
<td>BUSY:</td>
<td>.WORD</td>
<td>0</td>
<td>NEXT SECTOR FOR LDV 1 AND LDV 2</td>
</tr>
<tr>
<td>3</td>
<td>023462</td>
<td>000000</td>
<td>000001</td>
<td>NOOP</td>
<td></td>
<td>LAST SECTOR FOR LDV 1 AND LDV 2</td>
</tr>
<tr>
<td>4</td>
<td>023466</td>
<td>000000</td>
<td>000001</td>
<td>NOOP</td>
<td></td>
<td>LAST SECTOR WRITTEN</td>
</tr>
<tr>
<td>5</td>
<td>023472</td>
<td>000000</td>
<td>NEXT:</td>
<td>.WORD</td>
<td>0.1</td>
<td>WRITE REQUEST COUNT</td>
</tr>
<tr>
<td>6</td>
<td>023474</td>
<td>000000</td>
<td>LSECT:</td>
<td>.WORD</td>
<td>0.1</td>
<td>HEADER (AIRCRAFT TYPE)</td>
</tr>
<tr>
<td>7</td>
<td>023476</td>
<td>000000</td>
<td>SECD:</td>
<td>.WORD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>023500</td>
<td>000000</td>
<td>REQUEST: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>023500</td>
<td>000000</td>
<td>HEADR: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>023502</td>
<td>000000</td>
<td>FRAMES:</td>
<td>.WORD 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>023504</td>
<td>000000</td>
<td>FRAMEA: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>023506</td>
<td>000000</td>
<td>FRAMEB: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>023510</td>
<td>000000</td>
<td>SFLAGA: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>023512</td>
<td>000000</td>
<td>SFLAGB: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>023514</td>
<td>000000</td>
<td>FIRSTA: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>023516</td>
<td>000000</td>
<td>FIRSTB: .WORD 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>BUFSAT:</td>
<td>.BLK 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>023524</td>
<td>000000</td>
<td>000000</td>
<td>FURST:</td>
<td>.WORD 0,0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>023530</td>
<td>000000</td>
<td>DATA:</td>
<td>.WORD 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>023532</td>
<td>000000</td>
<td>CNTRA:</td>
<td>.WORD 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>023534</td>
<td>000000</td>
<td>CNTRB:</td>
<td>.WORD 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>023536</td>
<td>000000</td>
<td>FILN:</td>
<td>.RADSO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>023536</td>
<td>045666</td>
<td>073300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>023542</td>
<td>000000</td>
<td>RADSO:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>023544</td>
<td>000000</td>
<td>RADSO:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>023544</td>
<td>000001</td>
<td>FIN:</td>
<td>.WORD 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ERRORS DETECTED: 0
FREE CORE: 11556 WORDS
READ.L1<READ.1
APPENDIX C

FILL LISTINGS
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Read Data From Disk</td>
</tr>
<tr>
<td>5-1</td>
<td>Store Data Words</td>
</tr>
<tr>
<td>6-1</td>
<td>Set Maximum Velocity</td>
</tr>
<tr>
<td>7-1</td>
<td>Read Block Of Data</td>
</tr>
</tbody>
</table>
.NLIST
.TITLE FILL PROCESS BUFFER
.GLOBL FTOR
.GLOBL NEXT
.GLOBL FILL,BUSY,FIN,REQUEST
.GLOBL LTRAN,LRSEC
.GLOBL WRITE
.GLOBL IFLD,LRSEC
.GLOBL BUF1
.GLOBL FCNTL,FCNT,R,FILN
.MCALL .PARAM .PARAM .PARAM .PARAM .PARAM
.PARAM
MACRO PUSH
        MOV R0,-(SP)
        MOV R1,-(SP)
        MOV R2,-(SP)
        MOV R3,-(SP)
        MOV R4,-(SP)
        MOV R5,-(SP)
        .ENDM

* MACRO POP
        MOV (SP)+,R5
        MOV (SP)+,R4
        MOV (SP)+,R3
        MOV (SP)+,R2
        MOV (SP)+,R1
        MOV (SP)+,R0
        .ENDM

; SAVE REGISTERS

; RESTORE REGISTERS
1  000000  ONE  =  0  ;START OF PROCESS BUFFER X
2  003720  TWO  =  2000. ;START OF Y
3  007540  THREE  =  4000. ;START OF N/I
4  013560  FOUR  =  6000. ;START OF U/U
5  001740  MNTS  =  932. ;MAXIMUM NUMBER OF POINTS
6  000000  CNT  =  0 ;WORD COUNT
7  000002  B4ST  =  2 ;BLOCK CONTROL
8  000004  HDRW  =  4 ;HEADER WORD
9  000006  P/NF  =  6 ;FRAME NUMBER,SECTOR NUMBER
10  000010  TINS1  =  8. ;START TIME, LOW ORDER
11  000012  TINS2  =  10. ;START TIME, HIGH ORDER
12  000014  TIMR1  =  12. ;READ TIME, LOW ORDER
13  000016  TIMR2  =  14. ;READ TIME, HIGH ORDER
14  000200  BB  =  200 ;BUFFER BUSY
15  000040  EOF  =  40 ;END OF FRAME
16  000010  SIZEB  =  8. ;WORDS PER CONTROL BLOCK
17  000370  SIZD  =  248. ;DATA WORDS PER BLOCK
18  000340  LEUW  =  340
FILL PROCESS BUFFER
READ DATA FROM DISK

1. SBTL
2. * READ 1000 WORDS OR FRAME OF DATA
3. * FORMAT FOR DATA PROCESSING PROGRAM
4. 
5. FILL DATA BUFFER
6. FILL:
7. PUSH
8. FILL1:
9. CLR EIOFI
10. CMP FCNTW, FCNTR
11. BHI FILL1
12. ALL FRAMES READ
13. NO, GO READ ONE
14. YES, TEST OVER
15. ALL FRAMES READ
16. NO, WAIT FOR A FRAME
17. YES, SET END OF TEST FLAG
18. FILL1:
19. INC EIOFI
20. JMP FIXT
21. SEC:
22. MOV IFLD,R1
23. PICK UP INHIBIT FLAG
24. BOTH LDVS
25. SECT:
26. MOU FTOR,R1
27. LDV 1 ONLY
28. SECT:
29. MOV FTOR+2,R2
30. SECT:
31. MOV #2,R2
32. SECT:
33. BR SECT4
34. SECT:
35. CLR R2
36. SECT4:
FILL PROCESS BUFFER

READ DATA FROM DISK

MACRO UROS-i1A 01-JAN-72 00:18 PAGE 4+

115 000434 001432
116 000436 016792 000430
117 000442
118 000442 004767 000144
119 000442 004767 000162
120 000452 005724
121 000454 005201
122 000456 005303
123 000460 001370
124
125 000462
126 000462 012762 002100'
127 000466 004767 000328
128 000472 036227 000002 000040
129 000500 001010
130 000502 002017 001740
131 000506 103657
132 000510 012767 177777 00004'
133 000516 000187 000004
134 000522
135 000522 005267 002372
136 000526
137 000526
138 000526 005067 0000006
139 000536 016702 002354
140 000536 015762 002342 003100'
141 000544 005362 003100'
142 000550 005301
143 000552 001002
144 000554 000167 177234
145 000560
146 000560 010167 000020'
147 000564
148 000564
149 000600 000205
150 000602
151 000602 012760 000005
152 000606 000167 000000'

;SET 4 DATA WORDS
;GO FIND MAX VELOCITY
;INCREMENT TO NEXT PROCESS WORD
;INCREMENT MAX WORD COUNT
;FINISHED BLOCK

;END OF FRAME
;YES
;NO, MAXIMUM POINTS
;NO, GO READ NEXT BLOCK
;YES, GO EXIT

;CLEAR WRITE BUSY FLAG
.SBTTL  STORE DATA WORDS

1*  STORE:
2*  FORMAT AND STORE DATA WORDS
3*  
4  000612
5  000612 012264 000000  MOV  (R2)+,ONE(R4)  FIRST DATA WORD  X
6  000616 012264 003720  MOV  (R2)+,TWO(R4)  SECOND DATA WORD  Y
7  000616 012264 003720  MOV  (R2)+,THREE(R4)  THIRD DATA WORD  N/I
8  000622 012264 013560  MOV  (R2)+,FOUR(R4)  FOURTH DATA WORD  U/U
9  000626 000207  RTS  PC  RETURN
10  000632 000207  RTS  PC  RETURN
FILL PROCESS BUFFER

SET MAXIMUM VELOCITY

.SETTL SET MAXIMUM VELOCITY

.* FIND MAXIMUM VELOCITY

:MAXU:

CMP FOUR(R4),MAXWI ;VELOCITY HIGHER THAN PREVIOUS

BLOS MAXW1 ;NO

MOV MAXWI,MAXW2 ;YES, REPLACE LAST HIGHEST

MOU FOUR(R4),MAXW2 ;SET CURRENT MAX NUMBER

BR MAXW2

[MAXW1:

CMP FOUR(R4),MAXW1 ;VELOCITY HIGHER THAN PREVIOUS

BLOS MAXW2 ;NO

MOV MAXWI,MAXW2 ;YES, REPLACE LAST HIGHEST

MOU R1,MAXW1

RTS PC

:[MAXW2:

RTS PC

TIFR:

PUSH TIMR1(R2),TMEND

MOV TIMRE(R2),TMEND+2

MOU CNT(R2),R4

ASH #-8,R4

MOU R4,IFRM

POP

RTS PC
READ BLOCk OF DATA

;SBTL READ BLOCK OF DATA FROM DISK

* READ:

001002 005767 000105
001006 001403
001010 002787 000002 002066
001016 016067 001074 000022
001024 012767 000001 002062
001032 005067 000014

* READ:

TST FFG
BEQ R11
ADD #2,READST
MOV READRS(R0),READAD
SET READ SECTOR NUMBER
CLR CNTR
CLEAR ERROR/FUNCTION

* READ3:

TST CNTR
BEQ R13
ADD #4,READST
SET READ SECTOR NUMBER
CLR CNTR
CLEAR ERROR/FUNCTION

READ1: .WORD BUF1+16.
READ2: .WORD BUF2+16.
READS: .WORD BUF1
READR: .WORD BUF2

BUFI: .BLKW SIZEB
BUF2: .BLKW SIZEB

FTOR: .WORD 0,0
READST: .WORD 0
REDXX: .WORD 0
FFG: .WORD 0
FFLG: .WORD 0
FCTR1: .WORD 0
FCTR2: .WORD 0
MAX1: .WORD 0
MAX2: .WORD 0
PRODAT: .WORD LDUDAT+18.

CSECT ABTERM

IFLDV

IFLD: .BLKW 1

MAX1 .WORD 0

PRODAT: .WORD LDDAT+18.

PROCESS BUFFER ADDRESS

ABTERM .WORD 0
IFLDV .BLKW 1
READ BLOCK OF DATA

MAX2: .WORD 0
IEOF1: .WORD 0
LDUDAT =.
IFLY: .WORD 0
IFRM: .WORD 0
TMINT: .WORD 0.0
TMEND: .WORD 0.0
IDAY: .WORD 0
IPLN: .WORD 0
NUMPTS: .WORD 0
.END
### SYMBOL TABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABTRM</td>
<td>000000R</td>
<td>002</td>
<td>000200</td>
</tr>
<tr>
<td>BUF1</td>
<td>001100R</td>
<td>002</td>
<td>002100R</td>
</tr>
<tr>
<td>CNTR</td>
<td>000000R</td>
<td>002</td>
<td>001052R</td>
</tr>
<tr>
<td>FCNTW</td>
<td>003120R</td>
<td>002</td>
<td>003122R</td>
</tr>
<tr>
<td>FILLG</td>
<td>003115R</td>
<td>002</td>
<td>000522R</td>
</tr>
<tr>
<td>FILLA</td>
<td>000300R</td>
<td>002</td>
<td>000322R</td>
</tr>
<tr>
<td>FILLE</td>
<td>000060R</td>
<td>002</td>
<td>00064R</td>
</tr>
<tr>
<td>FILL0</td>
<td>000014R</td>
<td>002</td>
<td>00070R</td>
</tr>
<tr>
<td>FILL3</td>
<td>000322R</td>
<td>002</td>
<td>00410R</td>
</tr>
<tr>
<td>FILN</td>
<td>000002</td>
<td>002</td>
<td>00020G</td>
</tr>
<tr>
<td>FIXT</td>
<td>000560R</td>
<td>002</td>
<td>000560R</td>
</tr>
<tr>
<td>FOUR</td>
<td>012550</td>
<td>002</td>
<td>03100R</td>
</tr>
<tr>
<td>HLT</td>
<td>000602R</td>
<td>003</td>
<td>00014R</td>
</tr>
<tr>
<td>IFLD</td>
<td>000000R</td>
<td>003</td>
<td>00000R</td>
</tr>
<tr>
<td>IFLUN</td>
<td>000015R</td>
<td>005</td>
<td>0000R</td>
</tr>
<tr>
<td>LKTRAN</td>
<td>000002</td>
<td>004</td>
<td>00044R</td>
</tr>
<tr>
<td>MAXV</td>
<td>000040R</td>
<td>005</td>
<td>00074R</td>
</tr>
<tr>
<td>MAX4</td>
<td>000312R</td>
<td>005</td>
<td>003126R</td>
</tr>
<tr>
<td>MAX2</td>
<td>000002R</td>
<td>004</td>
<td>00174R</td>
</tr>
<tr>
<td>NUMPTS</td>
<td>000000R</td>
<td>005</td>
<td>000000</td>
</tr>
<tr>
<td>PRODAT</td>
<td>003130R</td>
<td>002</td>
<td>00130R</td>
</tr>
<tr>
<td>READ1</td>
<td>001070R</td>
<td>005</td>
<td>00174R</td>
</tr>
<tr>
<td>REDEX</td>
<td>003112R</td>
<td>005</td>
<td>003126R</td>
</tr>
<tr>
<td>R1</td>
<td>%200001</td>
<td>%200001</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>%200002</td>
<td>%200002</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>%200003</td>
<td>%200003</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>%200004</td>
<td>%200004</td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td>%200006</td>
<td>%200006</td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td>%200007</td>
<td>%200007</td>
<td></td>
</tr>
<tr>
<td>SECT3</td>
<td>000150R</td>
<td>005</td>
<td>00152R</td>
</tr>
<tr>
<td>SECT4</td>
<td>000120R</td>
<td>005</td>
<td>00121R</td>
</tr>
<tr>
<td>SF</td>
<td>%200000</td>
<td>%200000</td>
<td></td>
</tr>
<tr>
<td>THREE</td>
<td>000016</td>
<td>005</td>
<td>000114R</td>
</tr>
<tr>
<td>TIMR2</td>
<td>000016</td>
<td>005</td>
<td>000114R</td>
</tr>
<tr>
<td>TIMN</td>
<td>000010R</td>
<td>005</td>
<td>00004R</td>
</tr>
<tr>
<td>WRITE</td>
<td>000002</td>
<td>005</td>
<td>00000R</td>
</tr>
<tr>
<td>ABTERM</td>
<td>000000</td>
<td>005</td>
<td>00000R</td>
</tr>
<tr>
<td>IFLDV</td>
<td>000002</td>
<td>005</td>
<td>00000R</td>
</tr>
<tr>
<td>IHDL1</td>
<td>000002</td>
<td>005</td>
<td>00000R</td>
</tr>
<tr>
<td>LDUDAT</td>
<td>000002</td>
<td>005</td>
<td>00000R</td>
</tr>
</tbody>
</table>

**ERRORS DETECTED:** 0

**FREE CORE:** 11744. WORDS

**FILL.L2:** FILL.2
APPENDIX D

FORTRAN LISTINGS
SUBROUTINE SCAT
INTEGER PLTSYM
DIMENSION LBUF(15)
DIMENSION IMX(10), ISYM(10)
DIMENSION BK(11)
DIMENSION LIM(10), IMX(10)
DIMENSION PLTSYM(10), IMX(10)
COMMON XCOG(2), YCOG(2), INDEX(250), NCORPT(2), NOISES(2), PKVEL(2)
1. ELVANG(2), NVORTX, KFRM, RTIME
COMMON/BUFFER/IBUF(20)
COMMON/TMPS/KDFTT(6), JPEG
COMMON/DSP/LNS1, LNS2, LNS3, LNS4, LDT, KSM, KL3, LLDT
COMMON/ABTERM/IABTERM
COMMON/INF/PRTOL, NOISE, VELTOL, VORTOL, IRADI, IRADI2,
1. NRADI, CONV, STIME, PRVTOL
COMMON/LOD/BUFFER/IBUF(20)
COMMON/INF/PRTOL, NOISE, VELTOL, VORTOL, IRADI, IRADI2,
1. NRADI, CONV, STIME, PRVTOL

DATA ITXOFF, TYOFF, XSTRT, YSTRT, RLX, RLY, IXL, IYL/
1. 140, 173, 200, 0, 700, 400, 784, 442/
DATA ICHAR/'V'
DATA PLTSYM/'O', '1', '2', '3', '4', '5', '6', '7', '8', '9'/
DATA BK/20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120/
DATA IBUF(1,1,0,0,1,0)/
CALL OPEN(1,1)
XSCAL=IXL/RLX
YSCAL=IYL/RLY
IBUF(1)=NS4
KTRY=2
CALL DISPIO(14, KTRY, KL3)

C
C OUTPUT HEADER ON AUXILIARY SCREEN
NMT=2
IF(NMT.EQ.2)NMT=1
CALL DISPIO(11, NMT)
CALL DISPIO(1, 36, IBUF1)
C
C RE-SELECT SCREEN FOR SCATTER
CALL DISPIO(11, NS4)
C
C 22 CONTINUE
CALL DEBUG
C
C BRING UP BACKGROUND
CALL DISPIO(2, 13, 0)
C
C ENCODE FRAME FOR SCATTER HEADER
KFRM=TABS(IFRM)
CALL DECO(KFRM, 2, IBUF1(19))
IF(IFRM.LT.0)IBUF1(20)=PLTSYM(3)
IF(IFRM.GE.0)IBUF1(20)=PLTSYM(2)
C
C BRING UP SCATTER HEADER
CALL DISPIO(1, 40, IBUF1)
C
C STORE INDICES OF HIGHEST 10 VELOCITIES IN ARRAY IMX
DO 30 I=1, 10
IMX(I)=0
30
DO 31 K=1,10
IVLM=0
DO 31 I=1,NUMPTS
IF(IVEL(I) LE IVLM)GO TO 31
IF(K.EQ.1)GO TO 33
KM1=K-1
DO 32 J=1,KM1
IF(I.EQ.IMX(J))GO TO 31
32 CONTINUE
33 IMX(K)=I
IVLM=IVEL(I)
31 CONTINUE
C ENCODE VELOCITIES
DO 40 I=1,10,2
J=11-I
J=IMX(J)
IDUM=0
IF(J.EQ.0)GO TO 300
CALL GETVEL(IVEL(J),VEL)
IDUM=VEL
300 CONTINUE
K=(31-3*I)/2
CALL DECOD(IDUM,4,LBLF(K))
J=10-I
J=IMX(J)
IDUM=0
IF(J.EQ.0)GO TO 301
CALL GETVEL(IVEL(J),VEL)
IDUM=VEL
301 CONTINUE
K=K-1
CALL DECOD(IDUM,3,LBLF(K))
40 CONTINUE
C OUTPUT VELOCITIES
CALL DISPIO(2,24,-1)
CALL DISPIO(1,30,LBLF)
C SELECT CHARACTER TO BE OUTPUT FOR EACH POINT AND OUTPUT
KTWO=2
CALL DISPIO(14,KTWO,KS)
J=2
DO 1 I=1,NUMPTS
CALL GETVEL(IVEL(I),VEL)
DO 34 L=1,10
IF(1 NE IMX(L))GO TO 34
ICHAR=ISYM(L)
GO TO 5
34 CONTINUE
9 J=J-1
IF(J.EQ.6,GO TO 9
2 IF(VEL.LE.BK(J))GO TO 9
3 J=J+1
4 IF(J.EQ.4,GO TO 3
8 ICHAR=PLTSYM(J-1)
GO TO 5
6 J=2
GO TO 1
7 J=11
GO TO 8
5 CONTINUE
XF=IX(T)*CONXY
YF=IV(T)*CONXY
IBUF(5)=1
IBUF(3)=IXOFF+(XF-XSTRT)*XSCA
IBUF(4)=IYOFF+(YF-YSTRT)*YSCA
CALL SETAD(IBUF(6), ICHAR)
CALL DISPIO(4, 12, IBUF)
CONTINUE

C     CALCULATE CENTROID AND PLOT VORTEX POSITION WITH V
KTWO=2
CALL DISPIO(14, KTWO, KSM)
CALL CENTRD
KTWO=2
CALL DISPIO(14, KTWO, KLG)
IF(NVORTX .EQ. 0) GO TO 60
DO 61 I=1, 2
IF(XCG(I) .EQ. 0.) GO TO 61
IBUF(3)=IXOFF+(XCG(I)-XSTRT)*XSCA
IBUF(4)=IYOFF+(YCG(I)-YSTRT)*YSCA
CALL SETAD(IBUF(6), ICHAR)
CALL DISPIO(4, 12, IBUF)
61 CONTINUE
60 CONTINUE

C     HARD COPY IF REQUESTED
IF(JACUTO .EQ. 1) GO TO 65
CALL DISPIO(10, NS4)

C     DELAY
DO 11 I=1, 65
DO 11 J=1, 2000
11 BDUMB=(ADUMB+2.5)/3.6
65 CONTINUE

C     READ RECORD FROM TAPE
CALL VREAD

C     EXIT IF END OF FILE
IF(IEOFI .EQ. 1) GO TO 20

C     EXIT IF
IF(TABTRM .EQ. 33) GO TO 20
GO TO 22

20 RETURN
END
SUBROUTINE DECD (KK, KKK, KP)
BYTE KP(5)
BYTE NEG
DATA NEG/**/**
II=0
K=0
KNM=KKK
IF (KK) 10, 25, 25
10 KK=-KK
KP(1)=NEG
II=I
KNM=KNM-1
25 IF(KNM-5) 35, 50, 200
35 K=KK/10**KNM
50 DO 100 I=1, KNM
70 KT=K
K=KK/10**(KNM-I)
100 KP(I+II)=K-KT*10+48
200 RETURN
END
SUBROUTINE VREAD

COMMON/ BUFFER/ IBUF1(18)
COMMON/ TDUMMY/NRA, NFL
COMMON /IFLOV/IFLOV
COMMON /CFL1/ CFL1
COMMON /THDL1/ MAX1, MAX2, IEOFI
COMMON/ TINPT/ RTO, NOISE, VELTO, VORTOL, IRADI, IRAD2, NRADI,
1 CONVY, STIME, FRVTOL
COMMON /LODUAT/IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IY(1000), IV(1000), INTENS(1000), IVEL(1000)
COMMON/ PLNAME/ IDAT(60)
COMMON /ROUT/ ROUT, KGBF
COMMON /SETUP/ DATE, JDFT(20, 6), JPLTR(3), IPLINC, IL1(60), TIM(2),
1 IFLB(3), IFLV, TAHC, TLZ(90), KREVF(6)
INTEGER CFL1
DIMENSION BUF1(2), BUF2(2)

DATA BUF1/'TAPE', 'ERR'/
DATA BUF2/'EOF', 'READ'/
DATA DATS/'VREAD'/
ROUT = DAT

READ PHYSICAL RECORD FROM TAPE MT1
20 CONTINUE
CALL GET (1, 1, IFLY, 4010)
C WAIT FOR I/O OPERATION TO COMPLETE ON MT1
CALL WAIT (1, 0, N)
C GET RETURN STATUS FOR END-OF-FILE INDICATOR
CALL SFUN(1, 7, NBA, IEOFI)
IF (IEOFI) 35, 35, 30
30 CONTINUE
CALL DISPO(3, 8, BUF2)
RETURN
35 CONTINUE
C CHECK FOR 4008 WORD PHYSICAL RECORD READ
IF(N-1) 50, 50, 40
40 CALL DISPO(3, 8, BUF1)
RETURN
50 CONTINUE
IF (((IFRM, GT 0) AND (IFLOV, EO, 1)) OR ((IFRM, LT 0) AND
1 (IFLOV, EO, 2))) GO TO 20
DO 60 I=1, NUMPTS
60 IVEL(I) = IABS(IVEL(I))
C ENCODE FLY-BY-NO
CALL DECOH (IFLY, 5, IFLB)
C ENCODE DAY
CALL DECOH(IDAY, 3, DATE)
RETURN
END
SUBROUTINE DRUG
COMMON /ATTERM/ IABTRM
COMMON XCG(2), YCG(2), INDEX(250), N robot(2), NOISES(2),
PKVEI(2), ELVANG(2), NVORTX, KFRM, RTIME
COMMON /DLF/ IBASE, DIMX
COMMON /BUFFER/ IBUF1(12), IBUF2(6), IBUF3(2)
COMMON /BUFFER/ IFER(12)
COMMON /CTRL/ KPROC, TACQTL, MTIRF
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /IADDE/ IADDIY(26), IADPNTS(2), IDATA(64), IADDAT, BLANK
COMMON /IHD/ MAX1, MAX2, IEEFI
COMMON /IIFDV/ IIFDV
COMMON /INPT/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRADII, NRAD
COMMON /INIT/ ITPLT, PORTS(2), STARBS(2), VELMNF
COMMON /KSTFL/ KSTFL, KPLN
COMMON /LDVAT/ IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
COMMON /IOIT/ IOIT(6)
COMMON /PLNAME/ IDAT(60)
COMMON /ROUT/ ROUT, KGBF
COMMON /SEARCH/ ISFBN, ISDAY, ISP
COMMON /SETUP/ DATE, JDFT(20, 6), JPLTP(3), IFIND, IL1(60), TIM(2),
COMMON /TMPS/ KDFTT(6), JAUTO
COMMON /XCOND/KK1, KK2, KK3, KK4
BYTE IDATA
INTEGER BLANK
DATA M/6/
CALL SSWITCH (0, K)
IF (K NE 1) RETURN
CALL DISPIO (7, 1)
KTO=2
CALL DISPIO (14, KTO, KSM)
CALL DISPIO (13, 32)
WRITE (M, 100) IFLY, IFRM, ITMINT, ITMEND, IDAY, IPLN, NUMPTS
100 FORMAT (5X, 'IFLY=', I6, 5X, 'IFRM=', I6, 5X, 'FIRST TIME =', 2(I6, 1X), 4X,
'LAST TIME=', 2(I6, 1X), 5X, 'IDAY=', I6, ' IPLN=', I6, ' NUMPTS=', I6,
' I/N VP/VM/')
K1=NUMPTS
DO 344 I=1, K1
CALL SUBBIT (INTENS(I), 2, 8, KK1)
CALL SUBBIT (INTENS(I), 10, 7, KK2)
CALL SUBBIT (IABS(IVEL(I)), 2, 7, KK3)
CALL SUBBIT (IABS(IVEL(I)), 10, 7, KK4)
WRITE (M, 101) I, IX(I), IY(I), KK1, KK2, KK3, KK4
344 CONTINUE
101 FORMAT (1X, I6, 13X, I6, 11X, I6, 4X, I6, 4X, I6, 4X, I6, 4X, I6, 4X, I6)
999 CONTINUE
CALL SSWITCH (1, K)
IF (K EQ 1) GO TO 850
ENDFILE M
850 CONTINUE
CALL DISPIO (13, 64)
IF (JAUTO NE 1) AND (K NE 1) CALL DISPIO(10, NS4)
RETURN
END
BLOCK DATA
COMMON XCG(2), YCG(2), INDEX(250), NCORRT(2), NOISES(2),
PKVEL(2), ELVANG(2), NVORTX, KFRM, RTIME
COMMON /ABTERM/ IAABRM
COMMON /ATL/ IBASE, DMX
COMMON /BUFFER/ IBUF1(12), IBUF2(6), IBUF3(2)
COMMON /BUFFER/ IFR(12)
COMMON /CNTRL/ KPROD, TACDVT, MTIF
COMMON /DISPLAY/ NS1, NS2, NS3, NS4, LTD, KSM, KL, LLD
COMMON /TADDRS/ IADDRV(26), IADPTS(2), IDATA(66), IADDAT, BLANK,
1 IABLKN
COMMON /THDL1/ MAX1, MAX2, IENFT
COMMON /IFLDV/ IFLDV
COMMON /INPT/ IPTTOL, NOISE, VELTOL, VORTOL, IRAD1, IRAD2, NRAD1,
1 CONXY, STIME, FRVTO
COMMON /INIT/ IITLT, PORTS(2), STARBS(2), VELMN, F
COMMON /KSTFL/KSTFL, KPLN
COMMON /LVDAT/ IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IX(1000), IY(1000), INTEN(1000), IVEL(1000)
COMMON /IWTT/ IWTT(6)
COMMON /PLNAME/ IDAT(60)
COMMON /ROUT/ ROUT, KGEB
COMMON /SETUP/ DATE, JDFT(20, 6), JPLTP(3), IFLIND, IL1(60), TIM(2),
1 IFLB(3), ILDV, IAHC, ILZ(90), KRFBU(6)
COMMON /TMPS/ KFTTT(6), IJAN
COMMON /XCORDT/ KX1, KY2, KX3, KY4
COMMNE/RAD TEREAD
COMMON /MTFLEG/MTFLEG
BYTE IDATA
INTEGER BLANK

CONVY = 2099. / 1023 (440 METERS)
DATA CONVY/ 2 0518054/
DATA IITMINT, ITMEND/ 0, 0, 150/
DATA KSM, KL/ 03543, 034033/
DATA IREAD/0/
DATA MTWFLG/0/
DATA IFLDV/ 0/
DATA IDATA/ 064* " "/
DATA BLANK/ " "
DATA ROUT/ " "
DATA IFLB/2*00", 0 "/
DATA DATE/ " "
DATA IL1/420, 627, 644, 627, -1, 420, 621, 644, 621, -1, 420, 517, 700, 517,
1 -1, 420, 510, 700, 510, -1, 420, 473, 700, 473, -1, 420, 466, 700, 466, -1,
2 420, 429, 700, 429, -1, 420, 422, 700, 422, -1, 420, 319, 700, 319, -1,
3 420, 312, 700, 312, -1, 420, 275, 700, 275, -1, 420, 268, 700, 268, -1/
DATA PTTOL, NOISE, VELTOL, VORTOL, IRAD1, FRVTO/0, 50, 2, 0, 50, 0, 50, 40,
1 0 50/
DATA ILDV, IAHC/0, 1/
DATA IPLIND/12/
DATA JDFT/20*40, 20*50, 20*50, 20*20, 20*25, 20*50/
DATA NS1, NS2, NS3, NS4/2, 1, 3, 3/
DATA LDLT/12/
DATA KX1, KY2, KX3, KY4/560, 700, 770, 910/
DATA TACDVT/1 0E+06/
DATA MTIFR/0/

-213-
```
DATA 11,2 /-910,583,980,583,-1,910,576,980,576,-1,560,473,700,473,
1 -1,560,466,700,466,-1,560,429,700,429,-1,560,422,700,422,-1,
2 560,385,700,385,-1,560,378,700,378,-1,770,275,910,275,-1,
3 770,268,910,268,-1,560,231,700,231,-1,560,224,700,224,-1,
4 560,187,560,187,-1,560,180,560,180,-1,560,143,560,143,-1,
5 560,136,560,136,-1,630,143,700,143,-1,630,136,700,136,-1
DATA TOWT/1, -1, 0, 0, 120, 0/
DATA IBFR/1, 1, 308, 657, 6, 0, 1, 1, 406, 614, 5, 0/
DATA BASE, BTMX/566, 280, 0/
DATA KRBUF/1, -1, 0, 0, 180, 0/
DATA IDAT/'B-', '70', '7', 'L-', '10', '11', '2', 'E', 'JT', 'B-', '72',
1 '7', 'C-', '88', '0', '4', 'E', 'JT', 'B-', '73', '7', 'VC', '1',
2 '0', '2', 'E', 'PR', 'B-', '74', '7', 'BA', 'C1', '11', '1', 'E',
3 'PR', 'DC', '1-', 'A-', '30', '0', '0T', 'HE', 'R', 'DC', '1',
4 'IL', 'YU', 'SN', '1', 'DC', '1-', '0', 'YA', 'C-',
END
```
SUBROUTINE STRT (N)
COMMON XCG(2), YCG(2), INDEX(250), NCGORT(2), NOISE(2),
1 PKVEL(2), ELVANG(2), NVECT, KFRM, RTIME
COMMON /ABTERM/ IABTRM
COMMON /BUFFER/ IBUF(20)
COMMON /CFLI/CFLI
COMMON /CRTL/ KPROC, TAC:CELL, MTllib
COMMON /CPOL/ NS1, NS2, NS3, NPS, LDOT, KSM, KLG, LLDT
COMMON /HDLT/ MAX1, MAX2, IEOF1
COMMON /INIT/ IITFLT, FORTS(2), STARS(2), VELOMF
COMMON /TNPT/, PTOL, NOISE, VELTOL, VORTOL, ITRAD, ITRAD2, NTRAD,
1 CONXY, STIME, FRTOL
COMMON /READ/ IREAD
COMMON /KSTFL/KSTFL, KPLN
COMMON /KTF/L/G, KTF/L
COMMON /LDDAT/ IFLY, IFRM, ITRINT(2), ITRMEND(2), IDAY, IPLN, NUMPTS,
1 IY(1000), IY(1000), INTENS(1000), TVEL(1000),
COMMON /MTWFLG/MTWFLG
COMMON /ROUT/ ROUT, KGF
COMMON /SETUP/ DATE, IDF(20, 4), JPLTP(3), IPLIND, IL1(100), TIM(2),
1 FFLR(3), UIC, UAC, IL2(90), KREUF(6)
COMMON /STAT/ TOT(2), KVAN1, KVAN2
COMMON /TMPS/ KDFIT(3), JAUTQ
BYTE CHAR(4)
BYTE IDATE(4)
INTEGER CFLI
DIMENSION ERMSG(3)
DIMENSION IBUF(3)
DIMENSION ITRINT(2)
DIMENSION N(3)
DIMENSION NUMPTS(2)
DIMENSION TEXT(16), TEXTI(16)
EQUIVALENCE (CHAR(1), IFLR(1))
EQUIVALENCE (IDATE(1), DATE)
DATA DATS/"STRT"
DATA DERSG/"OPT", "TAPE", "ERR"
DATA IBUF/1,1, 0, 0, 44, 0/
DATA KGF/40/
1 "K", "FL T", "IMR", "PO", "R", "DP S", "TARR", "POS"
1, "P", "S", "X", "V", "Y", "Y" /

STRT SUBROUTINE PROCESSING

VORTEX - COMMON VARIABLES DEFINITION

CFLI - CONTINUOUS MODE FLAG (0-OPERATOR CONTROLLED), 1-CONTINUOUS
CHAR - BYTE ARRAY; EQUIVALENCE (CHAR, IFLR)
DATS - MNEMONIC OF SUBROUTINE CODE
TABRM - FLAG TO INDICATE KEYBOARD REQUEST
+ (ABORT) (42)
1 (END FLY-BY) (33)
IEOF1 - FLAG FOR END OF FLY-BY (0-NORMAL FRAME, 1-END-OF-FLY-BY)
IFLY - FLY-BY NUMBER
ITRAD - CORRELATION CIRCLE RADIUS
ITRAD2 - SQUARE OR CORRELATION CIRCLE RADIUS
TERMINATE REQUEST MODE

0

ROUT = 0 AS
CONTINUE

1 if (FAIL EO 1) CALL INIT

BLE CONTINUOUS MODE ENABLE START PLY-BY-PLAY INTERRUPTS

2

(let(N1) EO 15) CIEL=0

(let(N1) EO 10) CIEL=0

..continue

(let(N2) EO 7) CIEL=1

(let(N2) EO 6) GO TO 2

(let(N1) EO 3) CIEL=1

..continue

1 if (FAIL) 2.1.1 if

SET CONTINUOUS AS OPERATION CONTROLLED START PLY-BY

(READ=0

CONTINUE

already

CALL STPNT(0.5, NPA, NEL)

NPA=1

CALL STPNT(1.1, NPA, NEL)

CALL STPNT (2.0)

OPEN MAG TAPE (MTD) FOR OUTPUT

8000 CONTINUE

10:6:00.0 10:30.0:10

8300 IF (MTD1) 10:30.0:10

osel IF TAPe REC INITIATED IN REAL TIME

MTRP SET=0 IN BLOCK DATA SET=1 FOR POST ANALYSES SET=1 IN

IF (MTD1) 10:30.0:10

MTRP=0 BLOCK DATA SET=1 IN START WHEN TAPE UNIT 0 IS OPENED

IASSERTION -

VALUE1 - TIME LIMIT FOR PLY-BY (SECONDS)
TIME1 - START PLY-BY TIME
STAGES - ARRAY
STAGES - TIME美麗 START OF PLY-BY
STAGES - MUSIC OF CURRENT SUBROUTINE
PORT1 - PORTS - ARRAY
PORT2 - ARRAY
PORT3 - SCREEN ASSIGNMENT FOR SCATTER PLANTS
PORT4 - SCREEN ASSIGNMENT FOR TABULAR DATA
PORT5 - SCREEN ASSIGNMENT FOR TIME BASED PLANTS
PORT6 - SCREEN ASSIGNMENT FOR-X GRAPH
PORT7 - EFFECT OF CORRELATION CIRCLE RADIUS
PORT8 - OPTION NUMBER
PORT9 - Option NUMBER
PORT10 - Option NUMBER
PORT11 - Option NUMBER
PORT11 - ARRAY: TABLE OPTION SELECTION

MTRP - A PLAE FOR MAG TAPE WRITE READY (0-WRITE, 1-INIT)AT
LINE - MAXIMUM FOR CURRENT PAGE OF TABULAR DATA
LET - INDEXATION F-Y-ATT POSITION OF CURRENT LINE OF TABULAR OUTLINE
LET - Y-ATT POSITION OF CURRENT LINE OF TABULAR OUTLINE
LET - X-ATT POSITION OF CURRENT LINE OF TABULAR OUTLINE
IF - IF XIao FOR START PLY-BY INTERRUPT (I-INTERUPT)
CONTROL - IF FOR TYPE OF PROCESSING (1-REAL TIME, 2-POST ANALYSIS
CONTROL - IF TRUE NUMBER OF DATA PLUS EXCEED 32,
CONTROL - POSITIVE FRAME NUMBER
CONTROL - ARRAY: START PLY-BY TIME
CALL DISPIO (13,1)

C INITIALIZE PORTS AND STARS FOR 1ST FRAME
PORTS(1)=0
PORTS(2)=800
STARS(1)=800
STARS(2)=0
VELMNE=0
ITPLT=0
IRAD1=IRAD1*IRAD1
NRAD1=IRAD1
DO 4317 I=1,8
4317 TOT(I)=0
KVA1=0
KVA2=0
C
C BRING UP DISPLAY BACKGROUNDS
IF (NS1 EQ 3) GO TO 75
C
C BRING UP X-Y PLOT BACKGROUND
C SELECT SCREEN NS1
CALL DISPIO (11,NS1)
CALL DISPIO (2,7,8)
75 CONTINUE
IF (NS2 EQ 3) GO TO 80
C
C BRING UP TIME PLOT BACKGROUND
C SELECT SCREEN NS2
CALL DISPIO (11,NS2)
CALL DISPIO (2,11,12)
80 CONTINUE
KFRM=0
81 CONTINUE
C
C INITIALIZE COUNT FOR MAXIMUM TABULAR SIZE
LFRAME=KFRM+55
IF (NS3 EQ 3) GO TO 85
C
C BRING UP TABULAR HEADING
C SELECT SCREEN NS3
CALL DISPIO (11,NS3)
CALL DISPIO (2,19,0)
LOT=.712
C PRINT TABULAR HEADING
IBUF(1)=NS3
IBUF(4)=LOT
CALL SETAD (IBUF(4),TEXT)
IBUF(3)=512
CALL DISPIO(4,12,IBUF)
C
IBUF(5)=64
LOT=LOT-LLDT
IBUF(4)=LOT
CALL SETAD (IBUF(4),TEXT1)
CALL DISPIO(4,12,IBUF)
LOT=LOT-LLDT
85 CONTINUE
TABTRM=0
IF (CFL1 EQ 1) GO TO 850
C IF NOT CONTINUOUS MODE CREATE HEADER FILL-IN  
N(2)=10  
CALL DFLT (N)  
850 CONTINUE  
C PROCESS DATA  
C CHECK FOR ADDITIONAL PAGES OF TABULAR DATA  
IF (LFRAME GT 60) GO TO 894  
C CHECK FOR OPERATOR CONTROLLED MODE  
IF (CFL1) 891, 891, 857  
857 CONTINUE  
C IF CONTINUOUS MODE LOOK FOR START FLYBY INTERRUPT  
C STOP -- WAIT FOR START FLY-BY INTERRUPT  
89 IF (TABTRM EQ. 42) GO TO 91  
IF (KSTFL NE 1) GO TO 89  
C DECODE BCD-CODED PLANE TYPE  
II=KPLN/16  
IF (KPLN-16*II GT 9) KPLN=16*II+9  
N(2)=KPLN-6*II+1  
N(4)=0  
CALL PTYP (N)  
N(2)=10  
CALL DFLT (N)  
C RESET START FLY-BY INTERRUPT  
891 KSTFL=0  
C BRING UP SCATTER PLOTS  
IF (NS4 EQ. 3) GO TO 87  
CALL SCAT  
GO TO 95  
87 CONTINUE  
C SELECT LARGE CHARACTER SIZE  
894 KTWO=2  
CALL DISPLO (14, KTWO, KFLG)  
C BRING UP HEADER ON SCREENS 1 AND 2  
C SELECT SCREEN J  
CALL DISPLO (11, 3)  
C PRINT HEADER-FILL IN  
893 CALL DISPLO (11, 3A, TRUE1)  
C SELECT SMALL CHARACTER SIZE  
KTWO=2  
CALL DISPLO (14, KTWO, KSM)  
C IF OTHER THAN FIRST PAGE OF TABULAR DATA SKIP INITIALIZATION  
IF (LFRAME GT 55) GO TO 90  
KFLG=0  
IF (KPROC EQ. 2) GO TO 90
IF REAL-TIME MODE ENABLE DATA & END-OF-FRAME INTERRUPTS
CALL INIT

CALCULATE FLY-BY NUMBER FOR TAPE
IFLY=0
DO 892 I=1,5
IFLY=IFLY*10+CHAR(I)-48
892 CONTINUE

DECODE DAY NUMBER FOR TAPE
IDAY=0
DO 600 I=1,3
IDAY=IDAY*10+IDATE(I)-48
600 CONTINUE
IFLN=IFLIND

FRAME ENTRY POINT

CONTINUE

CHECK FOR STOP FLY-BY KEYBOARD REQUEST
IF (IABTRM .EQ. 33) GO TO 95

CHECK FOR ABORT KEYBOARD REQUEST
IF (IABTRM .NE. 42) GO TO 92

ABORT CODE
SELECT LARGE CHARACTERS
RESET KEYBOARD REQUEST CODE

IABTRM=0
IF (KPROC .NE. 1) GO TO 911

DISABLE START FLY-BY INTERRUPTS
CALL DISAB1
DISABLE DATA & END-OF-FRAME INTERRUPTS
CALL DISAB1
SELECT OPERATOR CONTROLLED MODE

CONTINUE
CELI=0
IF (MTWFLG .EQ. 0) GO TO 191
IF (KPROC .EQ. 2) GO TO 191
CALL SFUN (0,2,NBA,NFL)
CALL SFUN (0,2,NBA,NFL)
NBA=1
CALL SFUN (0,5,NBA,NFL)
191 CONTINUE
CALL DISP (11,1)
BRING UP INITIAL DISPLAY
CALL DISP (2,1,0)
GO TO 950

CONTINUE

IF REAL-TIME MODE FILL PROCESSING BUFFERS WITH FRAME OF DATA
IF (KPROC .EQ. 1) CALL FILL

CHECK FOR STOP FLY-BY KEYBOARD REQUEST
IF (IABTRM .EQ. 33) GO TO 95

CHECK FOR ABORT KEYBOARD REQUEST
IF (IABTRM .EQ. 42) GO TO 91
CHARK(K)=48
IF (CHARK(K) LE 57) GO TO 97
66 CONTINUE

K5=5
INCREMENT FLY-BY 10 BY 1 (ASCII)

IF (KPOPT.GT.1) CALL DISABLE
IF REAL-TIME MODE DISABLE DATA & END-OF-FRAME INTERRUPTS

IF (MTIRE.EQ.0) CALL SEUN (0,2, NBA, NSF)
IF TAPE WRITE OPTION SELECTED WRITE END ON TAPE

95 IEOCT=0
RESET END-OF-FLY-BY FLAG

1E (ETIME LT TACOOL) GO TO 90
CHECK FOR FLY-BY EXCEEDING TIME LIMIT
1094 CONTINUE
GO TO 81

1E (CAUTO.NE.1) CALL DISPLT (1,3)
1E (FRAME LT LEAMON) GO TO 1094
CHECK FOR MAXIMUM PAGE COUNT ON TABLES DATA

1E (K NE 0) CALL TISP (3,12, RAM)
CHECK FOR TAPE ERROR ON TAPE WRITE

IF (MTIRE.EQ.0) CALL WAIT (0,0,K)

K=0
WAIT FOR PREVIOUS I/O OPERATION ON MTO TO COMPLETE

601 CONTINUE
1E (IEORI.NE.1) GO TO 95
CALL VREAD
1E (XPOLE.EQ.1) GO TO 601
94 CONTINUE
995 KFRED=1
SET FLAG TO INDICATE EXCEEDING NUMBER OF DATA POINTS

GO TO 94
CALL CENTRIO

CALL CENTRIO TO COMPUTE AND OUTPUT VORTICES CENTERINGS

KFRED=ABS(KFRED)
CONTINUE
60 TO 94

IF (LASEM.NE.1) GO TO 93
CHECK FOR FRAME EXCEEDING 93 DATA PLS

63 IF (LETOR.GE.339, 93, 95
CHECK FOR END-OF-FLY-BY

IF (MTIRE.EQ.0) CALL PUT (0,1, IFLA, 4009)
IF TAPE WRITE OPTION SELECTED INITIALIZE TAPE I/O
K=K-1
GO TO 96
97 CHAR(K)=CHAR(K)+1

C
RESET KEYBOARD REQUEST CODE
IHABTRM=0

C
DELAY
HARD COPY 4014 IF REQUESTED
IF (JAUTO NE 1) CALL DISPIO (10;3)

C
REAL-TIME MODE
KT=2
NNN(2)=1

C
CHECK FOR POST-ANALYSIS MODE
IF (KPROC NE 2) GO TO 98
KT=9
NNN(2)=3
98 CONTINUE

C
CHECK FOR CONTINUOUS OPERATION MODE
IF (CFI) 22,99,22
99 CONTINUE

C
BRING UP REAL-TIME OPTION SELECTION DISPLAY
BRING UP POST-ANALYSIS OPTION SELECTION DISPLAY
CALL DISPIO (2,KT,0)
CALL RTV (NNN)
950 RETURN
END
SUBROUTINE TERM (N)
COMMON /CNTRL/ KPROC, TACOTL, MTIRF
COMMON /MTWFLG/ MTWFLG
IF (KPROC EQ 2) GO TO 50
IF (MTWFLG .NE. 1) GO TO 100
CALL SFUN(0, 2, NBA, NFL)
CALL SFUN(0, 2, NBA, NFL)
CALL SFUN(0, 3, NRECD, NFL)
CALL WAIT(0, 0, NW)
CALL CLOSE(2, 0)

100 CONTINUE
CALL DISPIO (13, 16)
RETURN

50 CALL SFUN(1, 3, NRECD, NFL)
CALL WAIT(1, 0, NW)
GO TO 100
END
SUBROUTINE DSEL (N)  
COMMON /LDVAT/IFLV, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
  TX(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON /ABTERM/ IABTRM
COMMON /ATLP/ ATS, DIMX
COMMON /CNTRL/ KPLOC, TACOTL, MTRF
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /IFLV/ IFLDV
COMMON /SETUP/ DATE, JDFT(20, 6), TPLFT(3), IPLIND, IL1(60), TIM(2),
  TFLB(3), IFLV, IAHC, IL2(90), KRBUF(6)
COMMON /XCORDT/KY1, KY2, KY3, KY4
COMMON /TPRS/ KNTFT(4), IAUTO
COMMON /CFL/ CFL1
INTEGER CFL1
BYTE CHAR(6), KDABT(4)
DIMENSION N(8)
DIMENSION KDATE(2)
DIMENSION NS(4)
DIMENSION NNM(4)
EQUIVALENCE (IFLB(1), CHAR(1)), (KDATE(1), KDABT(1))
EQUIVALENCE (DATE, KDATE)
EQUIVALENCE (NS1, NS(1))
EQUIVALENCE (IL01, IL2(1)), (IL3, IL2(3)), (IL6, IL2(6)), (IL8, IL2(8))
EQUIVALENCE (IL01, IL2(81)), (IL83, IL2(83)), (IL86, IL2(86))
1 (IL88, IL2(88))
IF (IABTRM EQ. 42) GO TO 975
N2=N(2)
N5=N(5)
NT=(N2+1)/2
ASSIGN 950 TO KST
IF (N(4) NE 1) GO TO 35
IF (N2 NE 1) GO TO 36
KDATE(1)=NS
KDATE(2)=N(6)
IF (KPLOC EQ. 1) GO TO 950
ISDAY=0
DO 33 I=1,3
33 ISDAY=ISDAY*10+KDABT(I)-48
2000 IF(ISDAY-IDAY) .LT. 2500,34,3000
C NEED TO BACKSPACE FILE
2500 CONTINUE
NBA=2
CALL SFUN(1, 13, NBA, NFL)
IF(NFL) 975, 2600, 975
2600 CONTINUE
CALL VREAD
GO TO 2000
3000 CONTINUE
C FOREWARD SPACE A FILE
NBA=1
CALL SFUN(1, 10, NBA, NFL)
IF(NFL) 975, 2600, 975
34 CONTINUE
IPLIND=1PLN
CFLI=1
NNN(2) = IPLIND
NNN(4) = 0
CALL PTYP (NNN)
CFL = 0
GO TO 950

36 CONTINUE
IF (FLB(1) = N5)
IF (FLB(2) = N(A))
IF (FLB(3) = N(7))
IF (KPROC .EQ. 1) GO TO 950
ISFBN = 0
DO 1501 I = 1, 5
1501 ISFBN = ISFBN * 10 + CHAR(I) - 48
4000 IF (ISFBN = IFLY) 4500, 6000, 5000
4500 CONTINUE
C NEED TO BACKSPACE FILE
NBA = 2
CALL SFUN (1, 13, NBA, NFL)
IF (NFL) 975, 4400, 975
4400 CONTINUE
CALL VREAD
GO TO 4000
5000 CONTINUE
C NEED FORWARD SPACE A FILE
NBA = 1
CALL SFUN (1, 10, NBA, NFL)
IF (NFL) 975, 4400, 975
6000 CONTINUE
N2 = 1
GO TO 34
35 CONTINUE

IBASE IS BIT POSITION OF BAR ORIGIN
DIMX IS DELTA BIT RANGE FOR ENTIRE BAR
GO TO (40, 45, 50, 100, 150, 200, 250, 275, 300, 300, 300, 300, 300, 300, 300, 300, 300,
1300, 900, 925), N2
40 CONTINUE
IF (KPROC .NE. 2) GO TO 44
CALL SFUN (1, 3, NBA, NBA)
CALL SFUN (1, 4, 1, NBA)
CALL VREAD
GO TO 6000
44 CONTINUE
IF (N5 LT IBASE) N5 = IBASE
TACOTL = ((N5 - IBASE) * 300.0) / DIMX
IL01 = IBASE
IL3 = IBASE + (TACOTL * DIMX / 300)
IL6 = IBASE
IL8 = IL3
GO TO 950
45 CONTINUE
IF (KPROC .NE. 2) GO TO 47
CFL = -1
GO TO 950
47 CONTINUE
TACOTL = 1.0E+6
IL01=910
IL3=980
ILA=910
IL8=980
GO TO 950
50 CONTINUE
IF (IFLDV.EQ. 1) IFLDV=0
GO TO 435
100 CONTINUE
IFLDV=1
GO TO 460
150 CONTINUE
IF (IFLDV.EQ. 2) IFLDV=0
GO TO 435
200 CONTINUE
IFLDV=2
GO TO 460
250 CONTINUE
MTIRF=0
GO TO 435
275 CONTINUE
MTIRF=1
GO TO 460
300 CONTINUE
KI=1
IF ((N2.EQ.10) OR (N2.EQ.12) OR (N2.EQ.14) OR (N2.EQ.16)) THEN
KI=2
ASSIGN 300 TO KST
DO 400 I=1,4
IF (NS(I).NE.KI) GO TO 400
NS(I)=8
K1=KX1
K2=KX1
KT=10*I+31
GO TO 700
400 CONTINUE
ASSIGN 950 TO KST
IF (N2-2*NT) 425, 450, 450
425 NS(NT-4)=1
435 K1=KX1
K2=KX2
GO TO 475
450 NS(NT-4)=2
460 K1=KX3
K2=KX4
475 KT=NT*10-9
700 IL2(KT)=K1
IL2(KT+2)=K2
IL2(KT+5)=K1
IL2(KT+7)=K2
GO TO KST, (300, 950)
900 CONTINUE
JAV=2
IL81=630
IL83=700
IL86=630
IL88=700
GO TO 950

ORIGINAL PAGE IS OF POOR QUALITY
925 CONTINUE
   JAUTO=1
   IL81=840
   IL83=910
   IL86=840
   IL88=910
950 CONTINUE
   CALL DISPIO (9, 12, KBLUF)
   RETURN
975 TABTRM=0
   CALL DISPIO (11, 1)
   CALL DISPIO (2, 1, 0)
   RETURN
END
SUBROUTINE CENTRD
C
C THIS SUBROUTINE PERFORMS A SEARCH TO LOCATE THE POINTS WHICH
C DEFINE A VORTEX AND CALCULATES THE CENTROID OF THESE POINTS
C
C CONXY = SCALE FACTOR TO CONVERT IX AND IY TO FEET
C ELVANG(1) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
C ELVANG(2) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
C INTENS= INTENSITY OF DATA POINT
C IRAD12= SQUARE OF RADIUS FOR CORRELATION CIRCLE
C IVEL = VELOCITY OF DATA POINT
C IX = X COORDINATE OF DATA POINT
C IY = Y COORDINATE OF DATA POINT
C KPOINT = NUMBER OF POINTS WHICH LIE IN CORRELATION CIRCLE
C LSEARCH= MAXIMUM NUMBER OF SEARCHES ALLOWED FOR VORTEX LOCATION
C MAX1 = INDEX OF MAXIMUM VELOCITY IN A FRAME
C MAX2 = INDEX OF SECOND HIGHEST VELOCITY IN A FRAME
C MINCNT= NUMBER OF POINTS IN CORRELATION CIRCLE POSING A PEAK VELOCITY
C         EQUAL TO OR GREATER THAN MINVEL
C MINVEL= ITOL2*PEAK VELOCITY
C NOISES= NOISE SPIKES THAT WERE DISCARDED IN PROCESSING DATA FOR A
C         SPECIFIC VORTEX - ARRAY OF 2
C NSEARCH= NUMBER OF SEARCHES THAT HAVE BEEN MADE IN AN ATTEMPT TO
C         LOCATE A VORTEX
C NSPIKE= NOISE SPIKES THAT ARE FOUND IN A FRAME
C NUMPTS= NUMBER OF DATA POINTS IN A FRAME
C NVORTX= NUMBER OF VORTICES THAT HAVE BEEN LOCATED IE. NVORTX+1
C         IS THE VORTEX NO. THAT IS CURRENTLY BEING PROCESSED
C PTIOL = PER CENT OF POINTS IN A CORRELATION AREA THAT MUST HAVE A
C         VELOCITY GT VELTOL*PEAK VELOCITY
C SUM1= SUM OF PRODUCTS OF INTENSITY AND VELOCITY OF ALL POINTS
C         IN THE CORRELATION CIRCLE
C SUM2= SUM OF THE PRODUCTS OF INTENSITY, VELOCITY, AND X COORDINATE
C         OF ALL POINTS IN THE CORRELATION CIRCLE
C SUM3= SUM OF THE PRODUCTS OF INTENSITY, VELOCITY, AND Y COORDINATE
C         OF ALL POINTS IN THE CORRELATION CIRCLE
C VELMN2= MINIMUM ACCEPTABLE PEAK VELOCITY
C VELTOL= VELOCITY TOLERANCE--PER CENT OF PEAK VELOCITY THAT SOME
C         PER CENT OF POINTS IN A CORRELATION AREA MUST POSSESS
C VORTOL= PER CENT OF PEAK VEL FROM 1ST VORTEX THAT PEAK VEL FROM
C         2ND VORTEX MUST MEET
C XCG = X CENTROID OF VORTEX
C YCG = Y CENTROID OF VORTEX
C
COMMON XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
1 PKVEI(2), ELANG1, ELANG2, NVORTX, KFRM, RTIME
COMMON /CNTRL/ KPROC. TACGTL, MTIRF
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KL, LLDT
COMMON /IDUMY/NBA, NFL
COMMON /IHDLI/ MAX1, MAX2, IEQF1
COMMON /INITL/ ITPLT, PORTS(2), STARBS(2), VELMN2
COMMON /INPT/ PTL, NOISE, VELTOL, VORTOL, IRAD, IRAD1, NRAD1
1 CONXY, STIME, FRVTOL
COMMON /KTFLG/KTFVL
COMMON /LDOAT/ IFLY, IFRM, ITMINT(2), ITMN2(2), IDAY, IFLN, NUMPTS,
1 IX(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON /ROUT/ ROUT, KRF
COMMON /SETUP/ DATE, IFMT(20, 6), JPLTP(3), IFLIND, IL1(60), TIM(2),
1 IFILR(3), ILDV, IAH, IL2(90), KRBUE(6)

-227-
COMMON /ABTERM/ IABSRTM
COMMON /STAT/ TOTX1, TOTX2, TOTY1, TOTY2, SSQX1, SSQX2, SSQY1, SSQY2,

INTEGER KBUFO4(6)
INTEGER MSTAT(16)
REAL STT((S))
EQUIVALENCE (XCG1, XCO1), (YCO1, YCO1)
EQUIVALENCE (MSTAT, STT())
EQUIVALENCE (MSTAT4, MSTAT(4)), (MSTAT7, MSTAT(7)), (MSTAT13, MSTAT(13))
EQUIVALENCE (MSTAT15, MSTAT(15))
EQUIVALENCE (KBUFT4, KBUFT(6))
EQUIVALENCE (WTEST, JPLTP)
DATA SNAME/ 'CENT'/
DATA KBUFO(4*1, 32, 0)/
DATA STT(4) / 'AVG', '(', ' ', ' ', ' ', ' ', ' )' $', 'TD=', '(', ' ', ' ', ' ', ' ', ' ', ' ')
DATA WHEEL / 'WHEEL'/
DATA LNOISE/5/
ROUT= SNAME
TIMT= 32768 0*ITMINT(1) + ITMINT(2)
IF(KTFLG) 428, 428, 429
428 CONTINUE
STIME=TIMT
KTFLG=1
429 CONTINUE
NSPIKE=0
C  INITIALIZE NOISE(1) FOR COMPUTATION OF NOISE FOR EA. VORTEX
NOISES(1)=0
NVORTY)=0
MINPTS=2
C  CALCULATE MAXIMUM AND MINIMUM ELATION ANGLES
X=IX(1)
Y=IV(1)
ELANG1 = ATAN2(Y, X)*57 2958
X=IX(NUMPTS)
Y=IV(NUMPTS)
ELANG2 = ATAN2(Y, X)*57 2958
IF(NUMPTS-1)5000, 5000, 5010
5000 CONTINUE
NCORPT(1)=NUMPTS
CALL GETVEL(IVEL(1), VELMAX)
GO TO 5020
5010 CONTINUE
GO TO (1, 2000), KPROC
1 CALL GETVEL(IVEL(MAX1), VELMAX)
2 CONTINUE
IF(VELMAX-VELMN) 9, 10, 10
9 CONTINUE
NCORPT(1)= 0
5020 CONTINUE
NCORPT(2)= 0
PKVEL(1) = VELMAX
GO TO 2500
10 CONTINUE
C  INITIALIZE SUM TERMS AND COUNTERS
SUM1=0 0
SUM2=0 0
1. Calculate centroid of all points in correlation circle. Continue if minimum points have the minimum velocity. If not try another.

IF (COUNT-MNHTS) 1000, 1100, 110

INDEX (COUNT) = L

SUM (SW) = SUM (SW) + DUM (VA)

SUM (SW) = SUM (SW) + DUM (VA)

DUM (VA) = DUM (VA)

CALL subroutine (RENLS) (L: 2, I: DUM)

COUNT = COUNT + 1

IF (MNHT = MINCOUNT) 50, 49, 49

IF (VELOCITY - VELEM) 50, 49, 49

CONTINUE

IF (VELOCITY) 100, 100, 48

CALL subroutine (VELOC) (L: VELOC)

CONTINUE

IF (FP) = HAPDN (4.5, 4.5, 100)

FP = FP + 100 + 100 + 100

CONTINUE

IF (FP) = HAPDN (4.0, 4.3, 43, 43

CONTINUE

IF (FP) = HAPDN (4.0, 4.0, 100

FP = FP + 100 + 98 + 100

CONTINUE

IF (FP) = HAPDN (3.0, 3.0, 3.5, 35

CONTINUE

IF (FP) = HAPDN (3.0, 3.0, 3.0, 100

FP = FP + 100 + 98 + 100

CONTINUE

NR = 1.0 + 1.0, NUMPTS

LAMAX = LAMAX (MAX 1)

NE = LAMAX

DO 40 LDL = 1, NUMPTS

CONTINUE

These possess the minimum velocity, compute sum terms.

Determine which points lie in correlation circle and which do not.
VCG(NVORTX)=CONXY*DUM2
NOISES(NVORTX)= NSPIKE - NOISES(1)
NORPT(NVORTX)= KOUNT
VELMNF=FRVTOL*VELMAX
PKVEL(NVORTX)= VELMAX
IF(NVORTX-2)425, 2500, 2500
425 CONTINUE
C COMPUTE TIME
TMEND= 3276.8*ITMEND(1) + ITMEND(2)
ANGLE=ATAN2(YCG(1),XCG(1))*57.2958
C COMPUTE RTIME FOR START OF THIS SCAN
RTIME= .01666666*(TMINT - STIME)
DTIME = .01666666*(TMEND - TMINT)
C CHECK EVALUATION ANGLES FOR DIRECTION OF SCAN
IF(ANGLE-ELANG1) 430, 460, 440
430 CONTINUE
C SCANNING DOWN SWAP ANGLES
NBA=ELANG1
ELANG1=ELANG2
ELANG2=NBA
IF(ANGLE-ELANG2) 435, 435, 460
435 CONTINUE
IF(ANGLE - ELANG1)500, 500, 438
438 CONTINUE
DTIME= DTIME*(ELANG2 - ANGLE)/ (ELANG2 - ELANG1)
GO TO 500
440 CONTINUE
C SCANNING UP
IF(ANGLE-ELANG1)500, 445, 445
445 CONTINUE
IF(ANGLE - ELANG2)450, 460, 460
450 CONTINUE
DTIME= DTIME*(ANGLE - ELANG1)/ (ELANG2 - ELANG1)
460 CONTINUE
RTIME = RTIME + DTIME
500 CONTINUE
C SET FLAG TO INDICATE THAT WE ARE LOOKING FOR 2ND VORTEX
C SET VELOCITIES NEGATIVE THAT CORRESPOND TO POINTS THAT WERE USED
C TO DEFINE 1ST VORTEX
IF(KOUNT-2)530, 530, 525
525 CONTINUE
MNPTS=VORTOL*KOUNT
530 CONTINUE
NBA=MNPTS-KOUNT
IF(NBA-MNPTS)2280, 540, 540
540 CONTINUE
DO 550 L=1, KOUNT
J=INDEX(L)
VEL(L)= - VEL(J)
550 CONTINUE
GO TO 2000
C TRY ANOTHER PT WITH A MAX VELOCITY
1000 CONTINUE
NSPIKE=NSPIKE + 1
IF(MNPTS-2)2280, 2280, 1001
1001 CONTINUE
IF(LNOISE-NSPIKE)2300, 2300, 1010

-230-
1010 CONTINUE TVEL (MAX1) = -VEL (MAX1)
1050 CONTINUE IF (VEL (MAX1) > 2000, 2000, 1050)
1060 CONTINUE CALL GETVEL (VEL (MAX1), VELMAX)
2000 CONTINUE FIND NEW MAXIMA FOR VORTEX SEARCH
2050 CONTINUE MAX1 = 1
2100 CONTINUE MAX2 = MAX1
2130 CONTINUE IF (VEL (MAX2) > 2200, 2130, 2140)
2140 CONTINUE GO TO 2200
2150 CONTINUE MAX1 = MAX2
2200 CONTINUE MAX2 = MAX1
2220 CONTINUE IF (VEL (MAX1) > TVEL (L), TVEL (L) + 1)
2250 CONTINUE CALL GETVEL (VEL (MAX1), VELMAX)
2280 CONTINUE CONTINUE TVEL (MAX1) = TVEL (MAX1) + 1
2290 CONTINUE CONTINUE IF (NORT (NFL) = 0, 2290, 2300)
2300 CONTINUE CONTINUE NORT (NFL) = 1
2330 CONTINUE CONTINUE TVEL (MAX1) = TVEL (MAX1) + 1
2340 CONTINUE CONTINUE IF (NORT (NFL) = 0, 2340, 2350)
2350 CONTINUE CONTINUE NORT (NFL) = 1
2380 CONTINUE CONTINUE TVEL (MAX1) = TVEL (MAX1) + 1
2500 CONTINUE
3100 IF (WTEST .NE. WHEEL) GO TO 3600
   IF (MVORTX) 3600, 3600, 2700
2700 CONTINUE
3110 IF (IFRM) 3125, 3600, 3150
3125 IF (KVAN2=30) 3130, 3225, 3600
3130 CONTINUE
   KVAN2=KVAN2 + 1
   TOTY2=TOTX2 + XCG1
   TOTY2=TOTY2+XCG1
   SSQX2=SSQX2+XCG1*XCG1
   SSQY2=SSQY2+XCG1*XCG1
   GO TO 3600
3150 IF (KVAN1=30) 3155, 3250, 3600
3155 CONTINUE
   KVAN1=KVAN1 + 1
   TOTY1 = TOTX1 + XCG1
   TOTY1=TOTY1+XCG1
   SSQX1=SSQX1+XCG1*XCG1
   SSQY1=SSQY1+XCG1*XCG1
   GO TO 3600
3225 AVXCG=TOTX2
   AVXCG=TOTX2
   STDX=SSQX2
   STDY=SSQY2
   KBUFOT(3)=517
   GO TO 3275
3250 AVYCG=TOTX1
   AVYCG=TOTX1
   STDX=SSQX1
   STDY=SSQY1
   KBUFOT(3)=5
3275 CALL SETAB (KBUF0, MSTAT)
   KBUF0(1)=NS3
   KBUF0(4)=LDT-LLDT*KFRM
   LAVXCG=AVXCG/30.0
   LAVYCG=AVYCG/30.0
   KSTDX=SORT((30.0*STDX-AVXCG*AVXCG)/870.0)
   KSTDY=SORT((30.0*STDY-AVYCG*AVYCG)/870.0)
   CALL DECD (LAVXCG, 5, MSTAT4)
   CALL DECD (LAVYCG, 5, MSTAT7)
   CALL DECD (KSTDX, 3, MSTA13)
   CALL DECD (KSTDY, 3, MSTA15)
   CALL DISPT0 (4, 12, KBUFOT)
   GO TO 3700
3600 CONTINUE
   CALL DISPRA
3700 RETURN
   END
SUBROUTINE RTV (N)
COMMOM /ATLP/ IBASE, DIMX
COMMOM /BUFFER/ IBFR(12)
COMMOM /CTRL/ KPROC, TACOTL, MTIRF
COMMOM /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMOM /SETUP/ DATE, JDEF(20, 6), JPLTP(3), IPLIND, IL1(60), TIM(2),
1 IPLB(3), ILDV, IAHPC, ILZ(90), KRBUF(6)
COMMOM /READ/IREAD
COMMOM /DUMMY/NBA, NFL
COMMON /LBBAT/IFLY, TFRM, ITMDRT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IX(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMOM /CFLY/ CFLI
INTEGER CFLI
DIMENSION N(6)
DIMENSION NNN(4)
EQUIVALENCE (IBFR4, IBFR(4)), (IBFR10, IBFR(10))
EQUIVALENCE (IL01, ILZ(1)), (IL3, ILZ(3)), (IL6, ILZ(6)), (IL8, ILZ(8))
EQUIVALENCE (IL31, ILZ(31)), (IL33, ILZ(33)), (IL36, ILZ(36)), 1 (IL38, ILZ(38))
EQUIVALENCE (IL42, ILZ(42)), (IL44, ILZ(44)), (IL47, ILZ(47)), 1 (IL49, ILZ(49)), (IL52, ILZ(52)), (IL54, ILZ(54)), (IL57, ILZ(57)), 2 (IL59, ILZ(59)), (IL62, ILZ(62)), (IL64, ILZ(64)), (IL67, ILZ(67)), 3 (IL69, ILZ(69)), (IL72, ILZ(72)), (IL74, ILZ(74)), (IL77, ILZ(77)), 4 (IL79, ILZ(79))
DATA IND /0/
DATA KM1/-1/
CALL DISPIO(13, 32)
N2=N(2)
IF (N2 .GE. 3) GO TO 400
IL42=275
IL44=275
IL47=268
IL49=268
IL52=231
IL54=231
IL57=224
IL59=224
IL62=187
IL64=187
IL67=180
IL69=180
IL72=143
IL74=143
IL77=136
IL79=136
KPROC=1
NS4=3
IF (TACOTL .NE. 1.0E+6) GO TO 100
IL01=910
IL3=980
IL4=910
IL8=980
GO TO 200
100 CONTINUE
IL01=IBASE
IL3=IBASE+(TACOTL*DIMX/300 )
IL6=IBASE
IL8=IL3
200 CONTINUE
KT=3
IF (MTTRF EQ 1) GO TO 300
IL31=560
IL32=700
IL36=560
IL38=700
GO TO 500
300 CONTINUE
IL31=770
IL33=910
IL36=770
IL38=910
GO TO 500
400 CONTINUE
IBFR4=613
IBFR10=571
IF (IND EQ 1) GO TO 410
IND=1
CALL OPEN (1,1)
NBA=1
CALL SFUN (1,4,NBA,NFL)
410 CONTINUE
MTTRF=1
KPROC=2
IF (IREAD)425, 420, 425
420 CONTINUE
TREAD=1
CALL VREAD
IPLIND=IPLN
CFL1=1
NNN(2)=IPLIND
NNN(4)=0
CALL PTYP (NNN)
CFL1=0
425 CONTINUE
IL01=400
IL3=400
IL6=400
IL8=400
IL31=400
IL33=400
IL36=400
IL38=400
IL42=319
IL44=319
IL47=312
IL49=312
IL52=275
IL54=275
IL57=268
IL59=268
IL62=231
IL64=231

-234-
IL67=224
IL69=224
IL72=187
IL74=187
IL77=180
IL79=180
KT=10

500 CONTINUE
CALL DISPIO(13, 64)
CALL DISPIO (4, 24, IBFR)
IBFR4=657
IBFR10=615
CALL DISPIO (2, KT, KM1)
CALL DISPIO (9, 12, KRBUE)
RETURN
END
SUBROUTINE DISPLA
C IDITTB(1) = INITIAL X DIT POS FOR TABLE FOR VAN1 DATA
C IDITTB(2) = INITIAL X DIT POS FOR TABLE FOR VAN2 DATA
C IVDIT(1) = INITIAL Y DIT POS FOR XY PLOT FOR VAN1 DATA
C IVDIT(2) = INITIAL Y DIT POS FOR XY PLOT FOR VAN2 DATA
C IDITY(1) = INITIAL Y DIT POS FOR Y-TIME PLOTS FOR VAN1 DATA
C IDITY(2) = INITIAL Y DIT POS FOR Y-TIME PLOTS FOR VAN2 DATA
C IDITTY(1) = INITIAL TIME DIT POS FOR TIME-X PLOTS FROM VAN1 DATA
C IDITTY(2) = INITIAL TIME DIT POS FOR TIME-X PLOTS FROM VAN2 DATA
C IRUNXY = DIT POS FOR CENTER OF RUNWAY FOR XY PLOTS
C IRUNXT = DIT POS FOR CENTER OF RUNWAY FOR XT PLOTS
C IDITTM = DIT POS FOR INITIAL TIME POS FOR Y-TIME PLOTS
C LOCVA1(1) = LOCATION OF VAN1 WRT CENTER OF RUNWAY
C LOCVA1(2) = LOCATION OF VAN2 WRT CENTER OF RUNWAY
C NS1 SCREEN FLAG FOR XY PLOTS
C NS2 SCREEN FLAG FOR TIME PLOTS
C NS3 SCREEN FLAG FOR TABLE
C YRATXY = RATIO FOR X DIT POS FOR XY PLOTS
C YRATX = RATIO FOR X DIT POS FOR XY PLOTS
C YRATXT = RATIO FOR X DIT POS FOR X-TIME PLOTS
C TRATY = RATIO FOR TIME DIT POS FOR X-TIME PLOTS
C YRATYT = RATIO FOR TIME DIT POS FOR X-TIME PLOTS
C TRATYT = RATIO FOR TIME DIT POS FOR Y-TIME PLOTS
C COMMON/INITI/ ITPLT, PORTS(2), STABRTS(2), VELM1
C COMMON /DISP/ NS1, NS2, NS3, NS4, LBST, KSM, KLD, LLST
C COMMON /TDAT/ ELV, FERM, I1MINT(2), I1MEND(2), IDAY, I1PN, N2MPTS,
C 1 IX(1000), IX(1000), I1TENS(1000), I1VEL(1000)
C COMMON/ PADD/ IADDY(26), IADDITS(2), IDATA(56), IADDAT, BLANK
C 1 IAR
C COMMON XCG(2), YCG(2), INDEX(250), N2CORTS(2), NOISES(2),
C 1 PKV(2), ELVANG(2), NVRCTX, FERM, RTIME
C COMMON /ROUT/ ROUT, KGCF
C DIMENSION IDITTB(2), IVDIT(2), IDITY(2), IDITTY(2), LOCVA1(2)
C DIMENSION IBBUFF(24), PORTS(2), STABRTS(2)
C INTEGER BLANK
C BYTE IDATA, DECIML, KJJK, N2G
C DATA DECII/ /"
C DATA NEG"/""
C DATA LOCVA1/ 400, 400/
C DATA IBBUFF(2), IBBUFF(3)/1, 0/
C DATA DSM/"DATA"/
C DATA SNAME/"DISP"/
C REMOVE AFTER TEST ****
C DATA IRUNXY/ 518/
C DATA YRATXY/ 7/
C DATA YRATY/ 88/
C DATA IDITIT/459.85/
C DATA IDITIT/ 0.512/
C DATA IDITY/459.63/
C DATA IDITTY/712.316/
C DATA IRUNXT/784/
C DATA IDITTM/70/
C DATA YRATXY/ 42/
C DATA YRATXY/ 1625/
C DATA YRATY/ 88/
C DATA TRATYT/4 2/
C REMOVE ********************
ROUTE=SNAME
ITPLT=KFRM
IF(KFRM.LT.26)GO TO 10
ITPLT=ITPLT/26
ITPLT=KFRM-26*ITPLT+1
10 CONTINUE
MINANG=ELVANG(1)
MAXANG=ELVANG(2)
C ORDER ANGLES
IF(MAXANG.GE.MINANG)GO TO 20
IDUM=MINANG
MINANG=MAXANG
MAXANG=IDUM
20 CONTINUE
IF(NVORTX.GT.0)GO TO 100
C NO VORTEX FOUND
IXC1=0
IXC2=0
IVC1=0
IVC2=0
LEVEL=PKVEL(1)
IRVEL=0
IVAN=1
IF(IFRM.LT.0)IVAN=2
GO TO 1000
100 CONTINUE
C TRANSFORM TO CENTER OF RUNWAY COORDINATE SYS
IF(IFRM.GT.0)GO TO 150
C DATA REC FROM VAN2
IVAN=2
IXC1=LOCVAN(IVAN)-XCG(1)
IXC2=LOCVAN(IVAN)-XCG(2)
GO TO 175
150 CONTINUE
C DATA REC FROM VAN1
IVAN=1
IXC1=XCG(1)-LOCVAN(IVAN)
IXC2=XCG(2)-LOCVAN(IVAN)
175 IF(NVORTX.EQ.2)GO TO 200
IF(IIVAN.EQ.1)GO TO 180
C ONLY ONE VORTEX WAS FOUND
IF((PORTS(IVAN)-XCG(1)).LE.(XCG(1)-STARBS(IVAN)))GO TO 300
GO TO 210
180 CONTINUE
IF((PORTS(IVAN)-XCG(1)).LE.(XCG(1)-STARBS(IVAN)))GO TO 210
GO TO 300
C DATA WILL BE ORDERED SO THAT LEFT VORTEX WILL BE IN LOC. 1
200 CONTINUE
IF(IXC1.LE.IXC2)GO TO 300
C VORTEXES IN REV. POS. SO SWITCH
210 CONTINUE
IDUM=IXC1
IXC1=IXC2
IXC2=IDUM
IVC1=YCG(2)
IVC2=YCG(1)
LEVEL=PKVEL(2)
IRVEL=PKVEL(1)
IDUM=NOISES(1)
NOISES(1)=NOISES(2)
NOISES(2)=IDUM
IDUM=NCORPT(1)
NCORPT(1)=NCORPT(2)
NCORPT(2)=IDUM
DUM=XCG(1)
XCG(1)=XCG(2)
XCG(2)=DUM
DUM=YCG(1)
YCG(1)=YCG(2)
YCG(2)=DUM
GO TO 1000
300 CONTINUE
ICYC1=YCG(1)
ICYC2=YCG(2)
LVEL=PKVEL(1)
IRVEL=PKVEL(2)
C CALCULATE DISTANCE FROM THE CENTER OF THE RUNWAY
1000 CONTINUE
C DATA IS NOW IN CORRECT LOCATIONS
IF (NS3 EQ 3) GO TO 1500
C TABLE WAS CHOSEN AS A DISPLAY OPTION
IDUM= 01*KFRM
IDUM1=KFRM-100*IDUM
IDUM= 1*IDUM1
IDATA(1)=IDUM+48
IDATA(2)=IDUM1-10*IDUM+48
IDUM= 001*NUMPTS
IDATA(4)=IDUM + 48
IDUM1=NUMPTS - 1000*IDUM
IDUM= 01*IDUM1
IDATA(5)= IDUM + 48
IDUM1= IDUM1 - 100*IDUM
IDUM= 1*IDUM1
IDATA(6)= IDUM+ 48
IDATA(7)= IDUM1 - 10*IDUM + 48
IDUM= 01*NCORPT(1)
IDATA(9)= IDUM + 48
IDUM1= NCORPT(1) - 100*IDUM
IDUM= 1*IDUM1
IDATA(10)=IDUM + 48
IDATA(11)=IDUM1 - 10*IDUM + 48
IDUM= 01*NCORPT(2)
IDATA(13)=IDUM + 48
IDUM1= NCORPT(2) - 100*IDUM
IDUM= 1*IDUM1
IDATA(14)=IDUM + 48
IDATA(15)=IDUM1 - 10*IDUM + 48
IDUM= 1*NOISES(1)
IDATA(18)=IDUM + 48
IDATA(19)= NOISES(1) - 10*IDUM + 48
IDUM= 1*NOISES(2)
IDATA(21)=IDUM + 48
IDATA(22)= NOISES(2) - 10*IDUM + 48
1200 CONTINUE
KDUM=IXC1
IDATA(46)=BLANK

1250 CONTINUE
IDUM=01*KDUM
IDATA(47)=IDUM + 48
IDUM1=KDUM - 100*IDUM
IDUM=1*IDUM1
IDATA(48)=IDUM + 48
IDATA(49)=IDUM1 - 10*IDUM + 48
IDUM=01*IXC1
IDATA(51)=IDUM + 48
IDUM1=IXC1 - 100*IDUM
IDUM=1*IDUM1
IDATA(52)=IDUM + 48
IDATA(53)=IDUM1 - 10*IDUM + 48
IF (IXC2 GE 0) GO TO 1260
KDUM=ABS(IXC2)
IDATA(56)=NEG
GO TO 1270

1260 CONTINUE
KDUM=IXC2
IDATA(56)=BLANK

1270 CONTINUE
IDUM=01*KDUM
IDATA(57) = IDUM + 48
IDUM1 = KIDUM - 100*IDUM.
IDUM = 1*IDUM1
IDATA(58) = IDUM + 48
IDATA(59) = IDUM1 - 10*IDUM + 48
IDUM = 01*IVC2
IDATA(61) = IDUM + 48
IDUM1 = IVC2 - 100*IDUM
IDUM = 1*IDUM1
IDATA(62) = IDUM + 48
IDATA(63) = IDUM1 - 10*IDUM + 48
IF(NVORTX EQ 2) GO TO 1450
IF(NVORTX EQ 1) GO TO 1300
IDATA(13) = BLANK
IDATA(14) = BLANK
IDATA(15) = BLANK
IDATA(21) = BLANK
IDATA(22) = BLANK
IDATA(36) = BLANK
IDATA(37) = BLANK
IDATA(38) = BLANK
DO 1280 J=40,49
   IDATA(J) = BLANK
1280 CONTINUE
DO 1285 J=51,59
   IDATA(J) = BLANK
1285 CONTINUE
IDATA(61) = BLANK
IDATA(62) = BLANK
IDATA(63) = BLANK
GO TO 1450
1300 CONTINUE
IF(XCG(1) EQ 0) GO TO 1350
IDATA(56) = BLANK
IDATA(57) = BLANK
IDATA(58) = BLANK
IDATA(59) = BLANK
IDATA(61) = BLANK
IDATA(62) = BLANK
IDATA(63) = BLANK
GO TO 1450
1350 CONTINUE
IDATA(46) = BLANK
IDATA(47) = BLANK
IDATA(48) = BLANK
IDATA(49) = BLANK
IDATA(51) = BLANK
IDATA(52) = BLANK
IDATA(53) = BLANK
GO TO 1450
1450 CONTINUE
IBUFF(1) = NS3
IBUFF(3) = ID1TTR(IVAN)
IBUFF(4) = LDT-LD1DT*KFRM
IF(KFRM GT 55)IBUFF(4) = LDT-LD1DT*(KFRM-55)
IF(KFRM GT 110)IBUFF(4) = LDT-LD1DT*(KFRM-55)

ORIGINAL PAGE IS OF POOR QUALITY
IRRBUF(5)=63
IRRBUF(4)=TARDAT
CALL DISP1O(4,12,IRRBUF)

1500 CONTINUE
IF(NVORTX EQ 0) GO TO 4000
IF(NS1 EQ 3) GO TO 2000

C

XY PLOTS WERE CHOSEN AS A DISPLAY OPTION
IRRBUF(1)=NS1
IRRBUF(5)=1
IRRBUF(7)=NS1
IRRBUF(8)=IRRBUF(2)
IRRBUF(11)=1
IF(YCG(1) EQ 0 0) GO TO 1700
C

THERE IS A PORT VORTEX TO BE PLotted
IRRBUF(3)= TRUNxy + IXC1*YRATXY
IRRBUF(4)= IVDIT(TVAN) + YCG(1)*YRATXY
IRRBUF(4)= IADDXY(ITPLT)
GO TO 1800

1700 CONTINUE
C

THERE IS NO PORT VORTEX
IRRBUF(4)=TARBLNK

1800 CONTINUE
IF(YCG(2) EQ 0 0) GO TO 1900
C

THERE IS A STARBOARD VORTEX
IRRBUF(9)=TRUNXY + IXC2*YRATXY
IRRBUF(10)=IVDIT(TVAN) + YCG(2)*YRATXY
IRRBUF(12)=IADDXY(ITPLT)
GO TO 1950

1900 CONTINUE
C

THERE IS NO STARBOARD VORTEX
IRRBUF(12)=TARBLNK

1950 CONTINUE
CALL DISP1O(4,24,IRRBUF)

2000 CONTINUE
IF(NS2 EQ 3) GO TO 4000
C

X-TIME AND Y-TIME PLOTS WERE CHOSEN AS A DISPLAY OPTION
IF(RTIME GE 81.) GO TO 4000
IRRBUF(1)= NS2
IRRBUF(7)= NS2
IRRBUF(13)=NS2
IRRBUF(19)=NS2
IRRBUF(5)= 1
IRRBUF(11)= 1
IRRBUF(17)= 1
IRRBUF(23)= 1
IRRBUF(4)= [IDIYX(TVAN) - RTIME*TRATXT
IRRBUF(16)= IRRBUF(4)
IRRBUF(9)= IDITTM + RTIME*TRATVT
IRRBUF(21)= IRRBUF(9)
IF(YCG(1) EQ 0 0) GO TO 2500
C

THERE IS A PORT VORTEX
IRRBUF(3)= TRUNXY + IXC1*YRATXY
IRRBUF(4)= IDAVTS(1)
IRRBUF(10)=IVDT(TVAN) + YCG(1)*YRATXY
IRRBUF(12)=IRRBUF(6)
GO TO 3000
2500 CONTINUE
C      THERE WAS NO PORT VORTEX
       IBBUFF(6)=IABLNNK
       IBBUFF(12)=IABLNNK
3000 CONTINUE
       IBBUFF(14)=IBBUFF(2)
       IBBUFF(8) =IBBUFF(2)
       IBBUFF(20)=IBBUFF(2)
       IF(XCG(2).EQ.0.0) GO TO 3500
C      THERE IS A STARBOARD VORTEX
       IBBUFF(15)= IRUNXT + IXC2*XRTXT
       IBBUFF(18)= IADPTS(2)
       IBBUFF(22)= IDITY(IVAN) + YCG(2)*YRTYT
       IBBUFF(24)= IADPTS(2)
       IF(XCG(1).NE.0) GO TO 3600
       IBBUFF(18)=IADDXY(19)
       IBBUFF(24)=IADDXY(19)
       GO TO 3600
3500 CONTINUE
C      THERE IS NO STARBOARD VORTEX
       IBBUFF(6)=IADDXY(19)
       IBBUFF(12)=IADDXY(19)
       IBBUFF(18)= IABLNNK
       IBBUFF(24)= IABLNNK
3600 CONTINUE
       CALL DISPIO(4,48,IBBUFF)
4000 CONTINUE
       IF(NVORTX.LT.2)RETURN
       PORTS(IVAN)=XCG(1)
       STARS(IVAN)=XCG(2)
       RETURN
END
SUBROUTINE GETVEL(IV, VEL)
     IF(IV LE 0) GO TO 1000
     CALL SUBRIT(IV, 2, 7, IDUM)
     IF,IDUM, GT, 69) GO TO 100
     VEL = FLOAT(IDUM)*1.8
     RETURN
     100 CONTINUE
     IF,IDUM, GT, 75) GO TO 200
     VEL = 1.8*69.0 + FLOAT(IDUM - 69)*3.6
     RETURN
     200 VEL = 1.8*69.0 + 3.6*6.0 + FLOAT(IDUM-75)*7.2
     RETURN
     1000 VEL = 0.0
     RETURN
     END
SUBROUTINE DFLT (N)
COMMON /BUFFER/ IBUF1(12), IBUF2(12), IBUFF(2)
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /IFLDV/ IFLDV
COMMON /INPT/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRAD2, NRADI,
1 CONXY, STIME, FRVTOL
COMMON /LDVDAT/ IFLV, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IX(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON /QWT/ IOWT(6)
COMMON /ROUT/ ROUT, KGBF
COMMON /SETUP/ DATE, JOFT(20, 6), JPLTP(3), IPLIND, IL1(60), TIM(2),
1 IFLB(3), ILDV, IAHC, IL2(90), KRBUF(6)
COMMON /TMPs/ KDFTT(6), JAUTO
BYTE IBUF2
DIMENSION N(8)
EQUIVALENCE (IBUF01, IBUF5(1)), (IBUF3, IBUF2(3)), (IBUF5, IBUF2(5)),
1 (IBUF7, IBUF2(7)), (IBUF9, IBUF2(9)), (IBUF11, IBUF2(11))
EQUIVALENCE (KDFTT1, KDFTT(1)), (KDFTT2, KDFTT(2)), (KDFTT3, KDFTT(3)),
1 (KDFTT4, KDFTT(4)), (KDFTT5, KDFTT(5)), (KDFTT6, KDFTT(6))
DATA DATS//"DFLT/"
ROUT = DATS
N2=N(2)
IF ((N2 .GE. 10) .OR. (N2 .EQ. 7)) GO TO 900
IF (N2 .GT. 6) GO TO 500
KDFTT(N2)=((N(5)-420.)*100.0)/560.0
IL1(10*N2-7)=N(5)
IL1(10*N2-2)=N(5)
GO TO (25, 50, 75, 100, 125, 150), N2
25 CONTINUE
IRADI =KDFTT1/CONXY
GO TO 900

50 VELTOL = KDFTT2*0.01
GO TO 900
75 PTTOL = KDFTT3*0.01
GO TO 900
100 NOISE = KDFTT4/10
GO TO 900
125 VORTOL = KDFTT5*0.01
GO TO 900
150 FRVTOL = KDFTT6*0.01
GO TO 900
500 DO 525 I=1,6
JOFT(IPLIND, I)=KDFTT(I)
525 CONTINUE
GO TO 950
900 CONTINUE
KI=IRADI*CONXY
CALL DECD (KI, 2, IBUF01)
KI=VELTOL*100.0
CALL DECD (KI, 2, IBUF3)
KI=PTTOL*100.0
CALL DECD (KI, 2, IBUF5)
CALL DECD (NOISE, 2, IBUF7)
KI=VORTOL*100.0
CALL DECD (KI, 2, IBUF9)
KI=FRVTOL*100.0
CALL DECD (KI, 2, IBUF11)
CALL DISPLO(9, 12, IOWT)
950 CONTINUE
RETURN
END
SUBROUTINE PTVP(N)
COMMON /BUFFER/ TRUE1(12), TRUE2(6), TRUE3(3)
COMMON /CELL/ CELL
COMMON /CTRL/ KPROC, TACOLT, MTIME
COMMON /DELTPT/ KDELPT
COMMON /DELUV/ DELUV
COMMON /INFT/ PTFNL, NOISE, VELTOL, VORTOL, TRANT, TRANT2, NRANT.
1 CONVY, STIME, FRATOR
COMMON /LDURAT/ LDUV, LDURM, ITMINT(2), ITMENR(2), ITDVR, ITLRU, NUMETS.
1 ITY(1000), ITY(1000), INTENS(1000), ITVEL(1000)
COMMON /OUT/ INOUT(A)
COMMON /PLNAME/ PLAT(A)
COMMON /ROUTE/ ROUT, KRES
COMMON /SETUP/ DATE, TDAT(2, 6), JPLTP(3), IJPLN0, TL1(A0), TIM(2).
1 TPLR(3), TPLU, IAHW, TI2(90), KRETU(6)
COMMON /TRPS/ KDEFN(A), JAHOT
INTEGER CEL1
INTEGER XL, YM, VI, VI, DNV
DIMENSION M(A), N(A)
DIMENSION ITDATE(A)
EQUIVALENCE (ITDATE(1), DATE)
EQUIVALENCE (JPLTP1, JPLTP(1)), KJPLTP2, JPLTP(2), KJPLTP3, JPLTP(3))
C
C X1 IS LOW X-COORDINATE
C YM IS HIGH Y-COORDINATE
DATA X1, YM/420.980 /
C
DATA M1/-1/
DATA DATS/"PTVP"/
ROUT = DATS
N2=N(2)
IF (N2 EQ 3) GO TO 100
IF (N2 NE 21) GO TO 75
IF (KPROC EQ 1) GO TO 985
KDEFN=1
GO TO 950
75 CONTINUE
IF (N(4) EQ 1) GO TO 200
IJPLN0=N2
I=3*TPLN0-2
JPLTP1=TDAT(1)
JPLTP2=TDAT(1+1)
JPLTP3=TDAT(1+2)
GO TO 900
100 CONTINUE
I=7
M(2)=1
IF (KPROC EQ 1) GO TO 110
I=9
M(2)=3
110 CONTINUE
CALL DISP10 (2, I, 0)
CALL RTV (M)
RETURN

ORIGINAL PAGE IS OF POOR QUALITY
200 CONTINUE
1PLIND=18
1JPLTP1=N(5)
1JPLTP2=N(6)
1JPLTP3=N(7)
900 CONTINUE
1F(KPROC-1) 930, 930, 910
910 CONTINUE
912 IF(IPLIND - IFLN) 915, 930, 915
915 CONTINUE
NBA=1
CALL SFUN(1,10,NBA,NFL)
CALL VREAD
GO TO 912
930 CONTINUE
DX = XM-X1
DO 950 I=1,6
KDFTT(I) = JDEFT(IPLIND, I)
KY=X1+(DX*KDFTT(I))*0.01
IL1(10*I-7)=KY
IL1(10*I-2)=KY
950 CONTINUE
TRADI = KDFTT(1)/CONXY
VELTOL = KDFTT(2)*0.01
PTTOL = KDFTT(3)*0.01
NOISE = KDFTT(4)/10
VORTOL = KDFTT(5)*0.01
FRTTOL = KDFTT(6)*0.01
J=5
IF (KPROC EQ 2) J=15
DO 960 I=1,3
TRUF1(I)=TFLB(I)
TRUF1(I+3)=IDATE(I)
960 TRUF1(I+9)=JPLTP(I)
CALL TIME (TRUF1(6))
IF (CELI EQ. 1) GO TO 995
CALL DISP(2,J,0)
CALL DISP(1,24,TRUF1)
C
C    CALL DISPLAY CONTROLLER TO DISPLAY DEFAULT IN WRITE THROUGH
CALL DISP(2,J+1,1M1)
CALL DISP(9,12,IOWT)
995 CONTINUE
RETURN
END

ORIGINAL PAGE IS
OF POOR QUALITY