Report No. 75-0004
Contract No. NAS8-25621

(NASA-CR-120703) VORTEX INFORMATION DISPLAY SYSTEM PROGRAM DESCRIPTION MANUAL (M&S Computing, Inc.) 249 p HC $7.50 CSCL 09B Unclas G3/61. 14583

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>
<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>1. INTRODUCTION</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>2. DATA ACQUISITION</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>3. FORTRAN ROUTINES</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>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>5.1.1 Display Index</td>
<td>128</td>
</tr>
<tr>
<td>5.1.2 Display Chapters</td>
<td>128</td>
</tr>
<tr>
<td>5.2 Librarian Processing Flow</td>
<td>133</td>
</tr>
<tr>
<td>5.2.1 Control Card Processing</td>
<td>136</td>
</tr>
<tr>
<td>5.2.2 Pen, Compose, and Line Card Processing</td>
<td>136</td>
</tr>
<tr>
<td>5.2.3 Text Card Processing</td>
<td>141</td>
</tr>
<tr>
<td>5.2.4 Output Formatting</td>
<td>144</td>
</tr>
<tr>
<td>5.2.5 Cleanup</td>
<td>144</td>
</tr>
<tr>
<td>6. PROGRAM DISPLAYS AND OUTPUT</td>
<td>147</td>
</tr>
<tr>
<td>6.1 Program Displays</td>
<td>147</td>
</tr>
<tr>
<td>6.2 Program Output</td>
<td>147</td>
</tr>
<tr>
<td>6.2.1 Magnetic Tape Output</td>
<td>147</td>
</tr>
<tr>
<td>6.2.2 Time-Based Plots</td>
<td>158</td>
</tr>
<tr>
<td>6.2.3 X-Y Plots</td>
<td>159</td>
</tr>
<tr>
<td>6.2.4 Tabular Data</td>
<td>159</td>
</tr>
<tr>
<td>6.2.5 Scatter Plots</td>
<td>160</td>
</tr>
<tr>
<td>APPENDIX A ATVWS Listings</td>
<td>163</td>
</tr>
<tr>
<td>APPENDIX B READ Listings</td>
<td>175</td>
</tr>
<tr>
<td>APPENDIX C FILL Listings</td>
<td>191</td>
</tr>
<tr>
<td>APPENDIX D FORTRAN Listings</td>
<td>204</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

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.
Figure 2-1
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:

- IFLD = 0, both LDV's
- IFLD = 1, LDV 2 only
- IFLD = 2, LDV 1 only

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 LDVIS 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

ATVWS

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

HALT
Display error message

EXIT TO DC

EXIT

Error

N

LKTRAN
Read first sector to position disk

B

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

Y

Initialize DR-11 A only

N

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 DISAB1

Purpose

The DISAB1 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, DISAB1

or CALL DISAB1

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

DISABLE INTERRUPT 2

ENABLE FLYBY INTERRUPT

---

**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

1. **INITBF**
   - Free all buffers
   - Set initial buffer for LDV 1&2
   - Clear and/or initialize all flags, etc.
   - **EXIT**

ERROR PROCESS

2. **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>

First Time of Buffer

Last Time of Buffer

248 Words of Data

<table>
<thead>
<tr>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
</tr>
<tr>
<td>Byte 0</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

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

Figure 2-9
Figure 2-10
Figure 2-11

INTR

Go to level 7 and save R0-R5

Bring in data words and discard

First EOF

Set not first EOF flag

EXIT

Initialize B interrupt routine

EXIT

Figure 2-12

INTBA3

Any writes waiting

N

Y

Is read in progress

Y

N

Is it locked

Y

N

Lock and go to level 2

WRITE

Write block to disk

Restore R0-R5

EXIT
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 LDV1 indicator (R2 = 0)

Set first time for block

Reset first time flag

Go to level 7 and save R0 - R5

Bring in 4 data words

Set LDV2 indicator (R2 = 2)

Check for disk write

Increment data word count

Figure 2-13
common (see flowchart) and the code is therefore shared.

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
SAW

Set buffer count, start, and store

Check buffer N

Buffer busy

N

Y

N = N+1

All buffers

N

A

2-8

HALT

Go display error message

EXIT

Set first sector flag

EXIT

Set sector in buffer

Set header, frame, and write ready

Go get sector number

CALBLK

Set store and start of new buffer

Set new buffer busy

Save pointer to full buffer

Figure 2-17

ORIGINAL PAGE IS OF POOR QUALITY

-23-
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

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

WAITR
  Test last write

Error in last

HALT
  Display error
  error message

Set sector address in calling sequence

Last buffer free

Return buffer to pool

Set LDV indicator
  R0=0 LDV1
  R0=2 LDV2

Start with search for last LDV

Any buffers ready

EOF
  buffer

Decrement write requests

Increment count to read

Both LDV's

Go to next LDV

Reset lock

EXIT

Figure 2-18
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
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 stripped 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

FILL PROCESS

READ
Start read for first buffer

Initialize store address and data count

PROCESS 1
Process buffer 1

End of frame
Y → B
N → PROCESS 2
Process buffer 2

End of frame
N → A
Y → B

EXIT

Figure 2-21
Figure 2-22
PROCESS BUFFER 2

Figure 2-23

-32-
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).
Figure 2-24

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

\[ \text{JSR PC, MAXV} \]

(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

\[ \text{JSR PC, TIFR} \]

(cannot be called via FORTRAN routine).

Upon entry:  \( R2 = \) current buffer address

Upon exit:  \( R2 = \) unchanged
SET MAXIMUM DATA

Figure 2-26

MAXV

Higher than previous max. 1

Higher than previous max. 2

Replace previous with this one

EXIT

TIFR

Set time for block

Set frame number

EXIT

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.
Figure 3-1
TAPE INITIALIZATION

Figure 3-2
FLYBY INITIALIZATION

1. Initialize variables for first data frame.
2. Check if plots are XY plots.
   - Yes: Establish screen to which plots directed.
   - No: Establish screen to which plots directed.
3. Check if plots are time plots.
   - Yes: Establish screen to which plots directed.
   - No: Establish screen to which plots directed.
4. Initialize for tabular output and headers.
5. RETURN

Figure 3-3

-42-
Figure 3-4
COMPLETE INITIALIZATION FOR FLYBY

(DUMMY3)

Continuous flyby

Abort flyby

Y

N

Flyby interrupt

Y

N

Reset start flyby interrupt

Output scatter plots

DUM3A

Determine aircraft index and fill header

3-6

3-27

SCAT

Generate and display scatter plots

RETURN

Figure 3-5

-44-
PLANE INDEX DETERMINATION

Figure 3-6

-45-
DEPENDENT PARAMETERS

(DUMMY4)

DISP IO
Establish output in large characters

DISP IO
Establish output to both screens

DISP IO
Fill in headers on both screens

DISP IO
Establish output in small characters

Frame number > 55
Y
N

Post-processing
Y
N

Decode flyby no. & day & set plane index for output

RETURN

Figure 3-7
ABORT FLYBY

- Reset abort flyby flag
- Real time
  - Y: Disable start flyby interrupts
  - N: Set flag to place system parameter display on screen
    - Mag. tape unit 0 open
      - Y: Post-processing
        - N: SPUN Write end of file
          - SPUN Write end of file
          - SFUN Backspace 1 record
        - Y: DISPIO
          - Establish screen 1 as primary screen
          - DISPIO Bring up first display
          - RETURN

Figure 3-8
Reset end of flyby flag

Data recorded on tape

Real time

Increment flyby number by 1

Set keyboard request code to 0

Auto hard copy

SFUN Write end of file on tape

DISABLE Disable data & end of file interrupts

Post-analysis

Set flags to bring up real time display

RETURN

DISPIO Hard copy screen

Set flags to bring up post-analysis display

Figure 3-9
PROCESS FRAME OF DATA

**Flowchart Description**

1. \(\text{DUMMY?}\)
   - \(\text{Real time:} Y\) \(\rightarrow\) \(\text{FILL}\)
     - Fill data buffers with raw input
   - \(\text{Real time:} N\)
     - \(\text{Stop flyby:} Y\) \(\rightarrow\) \(\text{A}\)
     - \(\text{Stop flyby:} N\)
       - \(\text{Abort flyby:} Y\) \(\rightarrow\) \(\text{A}\)
       - \(\text{Abort flyby:} N\)
         - \(\text{Record data on tape:} Y\) \(\rightarrow\) \(\text{PUT}\)
           - Write data record on tape
         - \(\text{Record data on tape:} N\)
           - \(\text{IEOFF} = 1\)
             - \(A\)
             - \(\text{IEOFF = -1}\)
               - \(B\)
             - \(\text{IEOFF = 0}\)
               - \(A\)
     - \(\text{Partial frame:} Y\) \(\rightarrow\) \(\text{Reset flag:} Y\)
       - Reset flag that indicates too many data points
   - \(\text{Partial frame:} N\)
     - \(\text{CENTRD}\)
       - Locate vortex centers

2. \(\text{VREAD}\)
   - \(\text{Read a record of data}\)
   - \(\text{End of file:} Y\) \(\rightarrow\) \(\text{A}\)
   - \(\text{End of file:} N\)
     - \(\text{DUM7A}\)
       - Determine tape status
     - \(\text{DUM7B}\)
       - Determine page status

**Figure 3-10**

*Original page is of poor quality*
DETERMINE TAPE STATUS

DUM7A

Data recorded on tape

WAIT
Wait for completion of data transfer

Tape error

DISPIO
Display error message

RETURN

Figure 3-11
DETERMINE PAGE STATUS

Figure 3-12

-51-
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
COMPOSE FIELD INPUTS FOR DAY AND FLYBY NUMBER

Figure 3-14

Search data tape for flyby with input flyby no.

Search data tape for flyby with input date

Real time

YN

A

A

COMPOSE

Place input in flyby number

Place input in date

Date input

Real time

YN

RETURN
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.
- PLTS1 which contains an *.
- PLTS3 which contains an o.
- IDATA which is an output buffer array.
- IL1 which is the buffer that contains bit positions for write-through lines showing default values for system parameters.
- IL2 which is the buffer that contains bit 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

DISPJO
Establish output in large characters

SETAD
Get address of blank

SETAD
Get addresses of alphabet

SETAD
Get address of 

SETAD
Get address of flyby date

SETAD
Get address of flyby number

RETURN
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

RTV

DISPIO
Disable display controller interface

Real time

Magnetic tape unit 1 open

Set write through lines for system parameters

Set real time flag

DISPIO
Enable display controller interface

DISPIO
Place day & flyby no. on screen

DISPIO
Bring up system parameter display

OPEN
Enable interface to magnetic tape unit 1

Set flag to indicate magnetic tape unit 1 is open

SFUN
Forward space a record

VREAD
Read first data frame

Set flags to fill header buffer

PTYP
Place day and flyby no. in header buffer

Reset continuous flyby flag

RETURN

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

Figure 3-19
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 rightmost 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

DECD

Encode number

RETURN

Figure 3-20
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 post-analysis, PTYP can also search the data tape to find a desired plane type.

External References

DISPIO, TIME, VREAD, and SFUN.
GET ADDRESS

Figure 3-21
SET PLANE TYPE

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 ) count</td>
</tr>
<tr>
<td>70 ( \leq ) 75</td>
<td>1.8 ( \times ) 69 + (count-69) ( \times ) 3.6</td>
</tr>
<tr>
<td>( &gt; 75 )</td>
<td>1.8 ( \times ) 69 + 6 ( \times ) 3.6 + (count-75) ( \times ) 7.2</td>
</tr>
</tbody>
</table>

Table 3-1
FIND VORTEX CENTER

CENTRD

Real time

Find maximum
Vpeak

Find points in
area & compute terms
for calculating vortex
center

Vpeak meet criteria

Increase counter for
noise spikes

Vortex defined

Calculate vortex center
and set descriptive
parameters

Second vortex for
frame

Calibrate
test

Noise spikes
limit

Set descriptive
parameters for
missing vortex

Calculate
statistics

Output
statistics

Output
data

Add to totals
for statistics

RETURN

RETURN

RETURN

Figure 3-24
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):

$$X = \frac{\sum_{i=1}^{k} I_i \cdot V_{\text{peak},i} \cdot X_i}{\sum_{i=1}^{k} I_i \cdot V_{\text{peak},i}} \quad \text{and} \quad Y = \frac{\sum_{i=1}^{k} I_i \cdot V_{\text{peak},i} \cdot Y_i}{\sum_{i=1}^{k} I_i \cdot V_{\text{peak},i}}$$

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, and
### DEFINITION OF TERMS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MNEMONIC</th>
<th>DEFINITION</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Input from LDV's</strong></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**                                                       |       |
| 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      | IRADI          | 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 * R. | None  |

Table 3-2

-73-
## DEFINITION OF TERMS
(continued)

### Variables Calculated in CENTRD

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MNEMONIC</th>
<th>DEFINITION</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>KOUNT</td>
<td>Number of points in a correlation region.</td>
<td>None</td>
</tr>
</tbody>
</table>

### Descriptive Parameters Determined in CENTRD

<table>
<thead>
<tr>
<th>X</th>
<th>XCG</th>
<th>A two element array containing the horizontal distances between the vortex centers and the LDV.</th>
<th>feet</th>
</tr>
</thead>
<tbody>
<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>NVORTX</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
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, DEBUG, and OPEN.
TERMINATE PROGRAM

Figure 3-26
SCATTER PLOT GENERATOR

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.
DISPLAY CONTROLLER

Data Tablet

Keyboard

Display Library

Input Interrupt Processor

User Input Processor

Common Utilities

Application Program Request Processor

Output Interrupt Processor

Operator Input Interface Parameters

Application Program Request Interface Parameters

Application Program

Display Terminal

Figure 4-1
USER INPUT PROCESSOR COMPONENTS

Init Loop 1 Term

DC User input executive

4-3

Init

DC Initialization

4-4

Loop 1

Process user input

4-6

Loop 50

Process SUB command

4-7

Loop 75

Process keyboard input

4-8

Loop 58

Process GS command

4-13

Loop 80

Process key-in field CR

4-9

Loop 91

Position at next key-in field

4-10

Loop 85

Process key-in field rubout

4-11

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
Figure 4-3
DC INITIALIZATION

**Figure 4-4**

1. **INIT**
   - Save DOS processor status
   - Set processor priority to 2
   - **INITDS** Initialize disk
   - Save DOS input interrupt
   - Set up DC input interrupt

2. **Save DOS output interrupt**
3. **Set up DC output interrupt**
4. **ATVWS** Initialize DR11 interrupt handler
5. **ONCE** Initialize address buffers
6. **NXTDSP** Bring up first display
7. **DSPOUT** Initiate data tablet burst mode

**RETURN**
Figure 4-5
PROCESS USER INPUT

Figure 4-6
PROCESS SUB COMMAND

LOOP50

Activate graphic cursor

Calculate beginning horizontal bar position

Set beginning position to left edge of screen

Calculate ending horizontal bar position

Set position to right edge of screen

Calculate beginning vertical bar position

Set position to bottom edge of screen

Calculate ending vertical bar position

Set position to top edge of screen

Intersection position screen

Deactivate graphic cursor

RETURN

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

-91-
Figure 4-10
PROCESS KEY-IN FIELD RUBOUT

Figure 4-11
### OPERATOR INPUT INTERFACE PARAMETER LIST

<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>

*Figure 4-12*
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,
PROCESS GS COMMAND

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.
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</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tabular Output</td>
<td></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></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></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></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></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></td>
<td>A(7)</td>
<td>A(screen #)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Refresh Message</td>
<td></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></td>
<td>A(10)</td>
<td></td>
<td>A(screen #)</td>
</tr>
<tr>
<td>11</td>
<td>Auxiliary Screen</td>
<td></td>
<td>A(11)</td>
<td></td>
<td>A(screen #)</td>
</tr>
<tr>
<td>13</td>
<td>Reset</td>
<td></td>
<td>A(13)</td>
<td></td>
<td>A(option #)</td>
</tr>
<tr>
<td>14</td>
<td>High-Speed Output</td>
<td></td>
<td>A(14)</td>
<td></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**

### Format A

<table>
<thead>
<tr>
<th>Char 2</th>
<th>Char 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Char 4</th>
<th>Char 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Char 6</th>
<th>Char 5</th>
</tr>
</thead>
<tbody>
<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>
<th>Screen #</th>
<th>Character Size</th>
<th>Initial X</th>
<th>Initial Y</th>
<th># Characters</th>
<th>A (Characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Format C

<table>
<thead>
<tr>
<th>Screen #</th>
<th>Plot Type</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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.
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)).
VECTOR PLOT

PLVECT

Set up vector plot output string

Disconnect Y

More vector plot input

Set up point in output string

More vector plot input

Disconnect N

More vector plot input

Y

A

4-31

DSPOUT
Send vector plot

RETURN

Figure 4-23
ERASE SCREENS

- Select screen(s)
- to be erased

Send erase command

RETURN

Figure 4-24
REFRESH MESSAGE

Figure 4-25
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.

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).

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

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 4-28
HIGH-SPEED OUTPUT

Figure 4-29

-114-
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

Set up parameters

Initiate data tablet point mode

Send high-speed call next

Send alphanumeric cursor

High-speed output

Call next AIDS program

Initiate data tablet burst mode

Clear parameters and buffer for AIDS

RETURN

RETURN

RETURN
READ DISPLAY
LIBRARY

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

KEYSET

Key-in field in display

N

4-33

LIBINP
Read control page

RETURN

Y

4-33

LIBINP
Read key page

Set up first key field counters and pointers

4-30

ALPCRS
Position alphanumeric cursor at key-in field

RETURN

Figure 4-35
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
PERFORM DISK I/O

Figure 4-39

-125-
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
5. 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 preformatted 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 control 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
Figure 5-1
**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
-131-
### 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>$AX$ to End of Field</td>
</tr>
<tr>
<td>$AY$ to End of Field</td>
</tr>
<tr>
<td>Next Display Name</td>
</tr>
<tr>
<td>4-Character Next Program Name</td>
</tr>
</tbody>
</table>

Figure 5-3
A Y 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><strong>X-Coordinate of First Character</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Y-Coordinate of First Character</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Next Display Name</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4-Character Next Program Name</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5-4**

-134-
**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>X-Coordinate of First Character</th>
<th>Y-Coordinate of First Character</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>FILL FIELD ENTRY</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 ΔX and ΔY 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
PROCESS CONTROL RECORDS

CONTROL

- Delete record
  - Check for valid display name
  - Update flag set
    - Y
      - Set flag(s) for output option(s) selected
      - S=source listing
      - P=printed picture
      - L=create new library
      - U=update old library
    - N
      - Update flag set
        - Y
          - LKTRAN Read index of library to be updated
          - N
            - Display not found
              - Y
                - Set error for display not found in library
                - N
                  - Invalidate display entry in index
          - A
        - A
      - A
    - A
  - A
    - RETURN

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

Figure 5-8

-139-
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 user's 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 ensure 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 ensure 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 ΔX and ΔY to the end of the field.

-141-
TEXT RECORD PROCESSING

Figure 5-10

1. **TEXT**
   - Back-scan record and insert carriage return

2. **Text characters > 74**
   - Y: Print "text line exceeds 74 characters"
   - N: Text records > 35

3. **Text records > 35**
   - Y: Print "number of text lines exceeds maximum"
   - N: 5-11

4. **VERIFY**
   - Verify text and complete control page(s)

5. **Write text record to disk**

6. **RETURN**
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.
Figure 5-12
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

**INITIALIZE**

- **DAY** —
- **FLY-BY NO.** —
- **ACQ TIME LIMIT (MIN)**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>NONE</th>
</tr>
</thead>
</table>

#### DATA SELECTION

<table>
<thead>
<tr>
<th></th>
<th>SELECTED</th>
<th>INHIBITED</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UAN 1</strong></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td><strong>UAN 2</strong></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

#### MAG TAPE RECORDING

- [ ]
- [ ]

#### DISPLAY SELECTION

<table>
<thead>
<tr>
<th></th>
<th>PRIMARY</th>
<th>ALTERNATE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X-Y PLOTS</strong></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td><strong>TIME BASED PLOTS</strong></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td><strong>TABULAR DATA</strong></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

#### AUTO HARD COPY

- **YES**
- **NO**

#### OPERATION

- [ ] CONTINUOUS
- [ ] PARAMETER SELECTION

---

Figure 6-3
## System Parameters

### Post-Analysis

<table>
<thead>
<tr>
<th>Fly-by Selection</th>
<th>Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day ___</td>
<td>Rewind</td>
</tr>
<tr>
<td>Fly-by No. _____</td>
<td>Auto</td>
</tr>
<tr>
<td>Van Selection</td>
<td>Processing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Van</th>
<th>Selected</th>
<th>Inhibited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Van 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Display Selection

<table>
<thead>
<tr>
<th>X-Y Plots</th>
<th>Primary</th>
<th>Alternate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Based Plots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabular Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scatter Plots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto Hard Copy</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### Operation

- [ ] Start Fly-by
- [ ] A/C Dependent Parameters

*Figure 6-3*
AIRCRAFT SELECTION

AIRCRAFT TYPE

@ B-707  @ L-1011  @ 2 ENGINE JET
@ B-727  @ C-880 (990)  @ 4 ENGINE JET
@ B-737  @ VC-10  @ 2 ENGINE PROP
@ B-747  @ BAC 111  @ 1 ENGINE PROP
@ DC-08  @ A 300  @ OTHER
@ DC-09  @ ILYUSHIN  @ ______
@ DC-10  @ YAC 40

☐ DEFAULT PLANE TYPE (POST ANALYSIS)

☐ RETURN TO SYSTEM PARAMETER SELECTION

Figure 6-4
AIRCRAFT DEPENDENT PARAMETER SELECTION

<table>
<thead>
<tr>
<th>FLY-BY NO.</th>
<th>DAY</th>
<th>TIME</th>
<th>A/C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AIRCRAFT DEPENDENT PARAMETERS

**CORRELATION CIRCLE**

<table>
<thead>
<tr>
<th>R = RADIUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50 60 70 80 90 100</td>
</tr>
</tbody>
</table>

**NOISE SPIKE FILTER**

- **A = % V-PEAK NOISE TOLERANCE**
- **B = % PTS IN TOLERANCE**
- **NS = VORTEX SEPARATION**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

**MISSING VORTEX CRITERIA**

- **C = % POINTS FOR 2ND VORTEX**
- **D = % V-PEAK FOR NEXT FRAME**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50 60 70 80 90 100</td>
</tr>
<tr>
<td>0 10 20 30 40 50 60 70 80 90 100</td>
</tr>
</tbody>
</table>

- 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 (FT) 0 10 20 30 40 50 60 70 80 90 100
R = RADIUS

NOISE SPIKE FILTER
A = % V-PEAK NOISE TOLERANCE 0 10 20 30 40 50 60 70 80 90 100
B = % PTS IN TOLERANCE
C = VORTEX SEPARATION
D = % V-PEAK FOR NEXT FRAME

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

☐ START FLY-BY
☐ UPDATE DEFAULTS
☐ RETURN TO SYSTEM PARAMETER

Figure 6-6
DISPLAY OF VORTEX INFORMATION

<table>
<thead>
<tr>
<th>FR 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 POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTS</td>
<td>P S</td>
<td>NN</td>
<td>MX</td>
<td>P</td>
<td>S</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>01 0015 008 003 00 00 14 16</td>
<td>039 037 000 2</td>
<td>-038 138</td>
<td>074 137</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02 0032 008 009 00 02 03 16</td>
<td>046 008 002 2</td>
<td>-035 031</td>
<td>033 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 0034 002 003 00 02 03 16</td>
<td>045 036 004 6</td>
<td>-032 076</td>
<td>033 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 0030 019 004 00 03 21</td>
<td>037 032 006 9</td>
<td>-031 076</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05 0037 019 004 00 05 19</td>
<td>033 037 004 9</td>
<td>-031 072</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06 0025 008 007 00 06 12</td>
<td>032 034 011 9</td>
<td>-031 072</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07 0035 019 004 00 05 19</td>
<td>032 034 011 9</td>
<td>-031 072</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08 0037 019 004 00 05 19</td>
<td>032 034 011 9</td>
<td>-031 072</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09 0045 022 013 00 01 03 14</td>
<td>034 036 011 9</td>
<td>-031 072</td>
<td>027 076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 0048 030 009 00 01 03 14</td>
<td>037 036 023 3</td>
<td>-118 045</td>
<td>083 039</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 0039 023 005 00 03 14</td>
<td>037 036 023 3</td>
<td>-118 045</td>
<td>083 039</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 0039 023 005 00 03 14</td>
<td>037 036 023 3</td>
<td>-118 045</td>
<td>083 039</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 0054 026 008 00 05 15</td>
<td>043 036 028 8</td>
<td>-124 053</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 0031 032 003 00 05 15</td>
<td>043 036 028 8</td>
<td>-124 053</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 0041 034 005 00 06 16</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 0040 037 002 00 06 16</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 0036 036 002 00 24 19</td>
<td>043 036 028 8</td>
<td>-143 053</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 0042 037 002 00 06 16</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 0051 031 002 00 06 20</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 0039 036 002 00 06 16</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 0033 013 002 00 06 20</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22 0008 066 000 00 00 22 29</td>
<td>045 036 025 1</td>
<td>-181 045</td>
<td>074 036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DISPLAY OF VORTEX POSITION

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

TIME 00:07:54  A/C B-707
NS = 02  C = 25  D = 50

A = 50  Bv 50
02 2.S
Du 50

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

ALTITUDE IN FT
300
250
200
150
100
50
0

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

ALTITUDE IN FT
300
250
200
150
100
50
0

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

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.
- **ITMIN (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.
o VAN 2 data is displayed in the bottom quadriles.

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

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

o The maximum time that may be displayed is 90 seconds.

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

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

o 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:

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

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

o 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:

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

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

o 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:

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

o 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 starboard 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.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-1</td>
<td>DISABLE INPUT</td>
</tr>
<tr>
<td>5-1</td>
<td>INITIALIZE BUFFERS</td>
</tr>
</tbody>
</table>
.NLST TTM
.TITLE ATUS
.MC ALL PAR AM, EXIT, W AIR
.GOB L LOOPZ
.GOB L DISPIQ
.GOB L LOCK
.GOB L FOR I INITI, DISAB1
.GOB L FIRSTA
.GOB L STORES
.GOB L WAD, FWRT, NEXT, SECON, LSECT
.GOB L BUF
.GOB L BUFST, BUFFER, FRAMES, PLAGA
.GOB L INTAB, INTAA
.GOB L RENAD
.GOB L FILN, BUF1
.GOB L ATUS, MALT, BUSY
.GOB L IN TTDS, LT, TRAN, LNKBLK
.GOB L INTAN, INTAN
.GOB L INTAA, INTAB
.GOB L DISAB, IFLD
.GOB L PONT, PONTW
.GOB L LRSEC
.GOB L INIT
.GOB L PUS T
.GOB L STORA, STORB
.GOB L INTER, F IN, SAI N, WRITE
.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), RI
MOV (SP), R0
. ENDM
**DR-11 DEFINITIONS**

1. `OBUF = 167772` ; DR-11 A OUTPUT BUFFER
2. `SRREG = 167770` ; DR-11 A STATUS REGISTER
3. `VECTA = 300` ; INTERRUPT VECTOR FOR DR-11 A
4. `LEULA = 300` ; INTERRUPT LEVEL FOR DR-11 A
5. `OBUF = 167762` ; DR-11 B OUTPUT BUFFER
6. `SRREG = 167760` ; DR-11 B STATUS REGISTER
7. `VECTB = 310` ; INTERRUPT VECTOR FOR DR-11 B
8. `LEULB = 300` ; INTERRUPT LEVEL FOR DR-11 B
9. `IBUF = 167754` ; DR-11 C INPUT BUFFER
10. `SRDEC = 167752` ; DR-11 C STATUS REGISTER
11. `VECTC = 320` ; INTERRUPT VECTOR FOR DR-11 C
12. `LEULC = 300` ; INTERRUPT LEVEL (6) FOR DR-11 C
13. `LEVS = 100` ; ENABLE A AND B INTERRUPTS
14. `IWRD = 140` ; ENABLE A INTERRUPT DR-11 C
15. `IWRDC = 100` ; WORD COUNT
16. `CNT = 0` ; BLOCK CONTROL WORD
17. `BB = 200` ; BUFFER BUSY
18. `DISARA = 304` ;
19. `DISARB = 314` ;
SET UP INTERRUPT VECTORS FOR DR-11A AND B

MOV #U'ECTA,RO ; INITIALIZE INTERRUPT VECTOR A
MOV #INTAA,(RO)+ ; LOCATION FOR REQUEST A, DR-11 A
MOV #LEUL(A),RO+ ; PSW
MOV #U'ECTB,RO ; INTERRUPT VECTOR B
MOV #INTAB,(RO)+ ; LOCATION FOR REQUEST A, DR-11 B
MOV #LEUL(B),RO+ ; PSW
MOV #INTCA,(RO)+ ; REQUEST A, DR-11 C
MOV #LEULC,(RO)+ ; PSW (LEVEL 6)

INTCA:
JSR RS,DISABL ; INHIBIT DATA FOR THIS FLYBY
IBUC,KPLN ; READ PLANE ID
KPLN,HEADR ; SET PLANE ID IN BLOCK HEADER
KSTFLY #1, =0= WAIT, 1=START 
RTI ; RETURN

INTCB:
NOP
NOP
RTI

HALT:
JSR RS,DISABL ; GO STOP INPUT
MOV #LEV5,PSH

; GO TO LEVEL ONE

HLT1:

MOV MSGAD(R0),MCS1

; GO TO LARGE CHARACTERS

HLT5:

JSR RS,DISPIO

; BLANK SCREEN

HLT4:

JSR RS,DISPIO

; OUTPUT ERROR MESSAGE

HLT2:

JSR RS,DISPIO

; REQUEST DISPLAY 1

HLT3:

JMP LOOPZ
INITIALIZE DR-11 FOR INPUT (A AND/OR B)

**INIT:**

1. PUSH
2. CLR FCNTTR
3. CLR FCNTW
4. CLR FINT
5. JSR PC, INITBF
6. CLR FIIN
7. JSR PC, INITBF
8. MOUV #IWRD, SREGA
9. CLR OBUFRA
10. POP

**INIT3:**

1. DEC R1
2. BEQ INIT2
3. MOUV #IWRD, SREGA
4. CLR OBUFRA
5. RTS RS

**INIT1:**

1. MOUV IBUFC, KPLN
2. CLR KSTFLY
3. MOUV #IWRDC, SREGC
4. RTS RS

**INIT2:**

1. MOUV #IWRD, SREGB
2. CLR OBUFBA
3. BR INIT4

**INIT4:**

1. POP
2. RTS RS

**INIT:**

1. MOUV #IWRD, SREGA
2. CLR OBUFRA
3. BR INIT4

**INIT:**

1. MOUV IBUFC, KPLN
2. CLR KSTFLY
3. MOUV #IWRDC, SREGC
4. RTS RS

**INIT:**

1. MOUV #IWRD, SREGA
2. CLR OBUFRA
3. BR INIT4

**INIT:**

1. MOUV IBUFC, KPLN
2. CLR KSTFLY
3. MOUV #IWRDC, SREGC
4. RTS RS
DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C

DISABLE INTERRUPT FOR DR-11 C
Macros:

**VROS-O1A**

01--JAN-72

00:05 PAGE G

**INITIALIZE BUFFERS**

```
;INITIALIZE BUFFERS (R2=0,LDV1 R2=2,LDV2)

;SAVE REGISTERS
PUSH ;SAVE REGISTERS

MOVBUF R1 ;PICK UP BUFFER COUNT

MOV#BUFST R2 ;SET BUFFER START ADDRESSES

DEC R1 ;CLEAR WORD COUNT

MOUV#BUFSTR2 R3 ;SET INITIAL BUFFER FOR LDV1

MOUV(R2),BUFSAT

MOUV(R2)+,R3

CLR BNBT(R3) ;FREE ALL BUFFERS

CLR CNT(R3)

CLR BNBT(R3) ;FREE ALL BUFFERS

CLR CNT(R3)

CLR FTOR ;CLEAR READ BUSY FLAG

CLR FTOR+2

CLR FIRSTA ;CLEAR FIRSTA

CLR FIRSTA+2

MOV#1,FRAMES ;CLEAR BUSY FLAG

MOUV#1,FRAMES+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST

CLR FURST+2

CLR BUSY

CLR FURST
ATUWS MICRO VR05-01A

01--JAH-72

00:0S PAGE 6-

INITIALIZE BUFFERS

58
59 001050 117 125 124 MSG1: .ASCII /OUT OF BUFFERS /
001053 040 117 106
001056 040 102 125
001071 105 105 105
001074 122 123 040
001077 040

60 001100 104 111 123 MSG2: .ASCII /DISK ERROR WRITE/
001103 113 040 105
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 111 114 106

63
64 001160 000002 MCODE1: .WORD 3
65 001162 000002 MCODE2: .WORD 2
66 001164 000002 MCODE3: .WORD 7
67 001166 000002 MCODE4: .WORD 14.
68 001170 000001 MBIG: .WORD 1
69 001172 000002 MBIG: .WORD 34033
70 001174 000000 MBIG: .WORD -1
71 001176 177777 MLTH1: .WORD 15.
72 001200 000020 MLTH1: .WORD 16.
73 001202 000000 MNULL: .WORD 0
74 001204 000001 MDISP: .WORD 1
75 000000 000000 ESECT KSTFL
76 000000 000000 KSTFL: .WORD 0
77 000002 000000 KPLN: .WORD 0
78
79 000001 .END
### SYMBOL TABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Start Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATUS</td>
<td>000000RG</td>
<td></td>
</tr>
<tr>
<td>BLOCK</td>
<td>000656R</td>
<td></td>
</tr>
<tr>
<td>BUFSAT</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>CNLIC</td>
<td>000100R</td>
<td></td>
</tr>
<tr>
<td>DISA</td>
<td>000304</td>
<td></td>
</tr>
<tr>
<td>FCNTR</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>FIN</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>FTOR</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td>000204RG</td>
<td></td>
</tr>
<tr>
<td>HLT2</td>
<td>000274R</td>
<td></td>
</tr>
<tr>
<td>HLT3</td>
<td>000246R</td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td>000314RG</td>
<td></td>
</tr>
<tr>
<td>INITDS</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>INIT3</td>
<td>000404R</td>
<td></td>
</tr>
<tr>
<td>INITAB</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>INITBB</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>INITCB</td>
<td>000166R</td>
<td></td>
</tr>
<tr>
<td>KFLN</td>
<td>000006R</td>
<td>002</td>
</tr>
<tr>
<td>LEBL</td>
<td>000030</td>
<td>002</td>
</tr>
<tr>
<td>LKTRAN</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>LOOPZ</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>MB16</td>
<td>001174R</td>
<td></td>
</tr>
<tr>
<td>MODE2</td>
<td>001162R</td>
<td></td>
</tr>
<tr>
<td>NC51</td>
<td>000272R</td>
<td></td>
</tr>
<tr>
<td>MULL</td>
<td>001202R</td>
<td></td>
</tr>
<tr>
<td>SBUF0</td>
<td>001300</td>
<td></td>
</tr>
<tr>
<td>SBUF1</td>
<td>167772</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>%0000097</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>%000081</td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td>%000004</td>
<td></td>
</tr>
<tr>
<td>R7</td>
<td>%000007</td>
<td></td>
</tr>
<tr>
<td>SFLAGA</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>SREGC</td>
<td>167760</td>
<td></td>
</tr>
<tr>
<td>STORE</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>VECTA</td>
<td>000030</td>
<td></td>
</tr>
<tr>
<td>VADR</td>
<td>XXXXX G</td>
<td></td>
</tr>
<tr>
<td>%ABS</td>
<td>000000</td>
<td>000</td>
</tr>
<tr>
<td>%KSTFL</td>
<td>000004</td>
<td>002</td>
</tr>
</tbody>
</table>

**ERRORS DETECTED:** 0

**FREE CORE:** 11635. WORDS

ATUS=LS<ATUS.2
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>REQUEST A (END OF FRAME)</td>
</tr>
<tr>
<td>5-1</td>
<td>READ DATA FROM DR-11</td>
</tr>
<tr>
<td>6-1</td>
<td>SWITCH AND WRITE</td>
</tr>
<tr>
<td>7-1</td>
<td>WRITE BUFFERS TO DISK</td>
</tr>
<tr>
<td>8-1</td>
<td>WRITE BLOCK NO.</td>
</tr>
<tr>
<td>9-1</td>
<td>INPUT BUFFERS</td>
</tr>
<tr>
<td>10-1</td>
<td>CONSTANTS</td>
</tr>
</tbody>
</table>
.NLIST TTM
.TITLE READ DATA FROM LDV 1 AND 2
.GLOBL FTOR
.GLOBL HEADR
.GLOBL FIRSTA
.GLOBL HADR, FURIT, NEXT, SECON
.GLOBL FILL, FIN, WRITE
.GLOBL INTAA, INTAB, INTBA, INTBB
.GLOBL SAV
.GLOBL BUFSAT, BUFSAT, FRAMES, SFLAGA
.GLOBL INTAW, INTBH, BUSY
.GLOBL FURST
.GLOBL STORES
.GLOBL LKITFIN, REJUSTR
.GLOBL LKIP: BLK, PCHIW
.GLOBL LOCK
.GLOBL HALT
.GLOBL BUF
.GLOBL LSECT
.GLOBL STORE, STORE
.MCALL .WAITR
.MCALL .PARAM
.PARAM
000000
* * *

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
<table>
<thead>
<tr>
<th>No.</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000340</td>
<td>LEVEL = 340</td>
</tr>
<tr>
<td>2</td>
<td>000100</td>
<td>LEVEL = 100</td>
</tr>
<tr>
<td>3</td>
<td>000000</td>
<td>CNT = 0</td>
</tr>
<tr>
<td>4</td>
<td>000002</td>
<td>BMT = 2</td>
</tr>
<tr>
<td>5</td>
<td>000004</td>
<td>HDRW = 4</td>
</tr>
<tr>
<td>6</td>
<td>000006</td>
<td>FRN = 6</td>
</tr>
<tr>
<td>7</td>
<td>000010</td>
<td>TIM1 = 8</td>
</tr>
<tr>
<td>8</td>
<td>000012</td>
<td>TIM2 = 10</td>
</tr>
<tr>
<td>9</td>
<td>000014</td>
<td>TIMR1 = 12</td>
</tr>
<tr>
<td>10</td>
<td>000015</td>
<td>TIMR2 = 14</td>
</tr>
<tr>
<td>11</td>
<td>000020</td>
<td>EOF = 40</td>
</tr>
<tr>
<td>12</td>
<td>000076</td>
<td>BB = 200</td>
</tr>
<tr>
<td>13</td>
<td>000100</td>
<td>BJ = 100</td>
</tr>
<tr>
<td>14</td>
<td>000030</td>
<td>EDT = 20</td>
</tr>
<tr>
<td>15</td>
<td>000076</td>
<td>DATNO = 62</td>
</tr>
<tr>
<td>16</td>
<td>000100</td>
<td>SIZB = 8</td>
</tr>
<tr>
<td>17</td>
<td>000370</td>
<td>SIZD = 248</td>
</tr>
<tr>
<td>18</td>
<td>IBUFA=167774</td>
<td>IBUFA=167774</td>
</tr>
<tr>
<td>19</td>
<td>VECTA = 304</td>
<td>VECTA = 304</td>
</tr>
<tr>
<td>20</td>
<td>IBUFB=167764</td>
<td>IBUFB=167764</td>
</tr>
<tr>
<td>21</td>
<td>VECTB = 314</td>
<td>VECTB = 314</td>
</tr>
<tr>
<td>22</td>
<td>TIM2 = 6246</td>
<td>TIM2 = 6246</td>
</tr>
<tr>
<td>23</td>
<td>TIM1 = 6244</td>
<td>TIM1 = 6244</td>
</tr>
</tbody>
</table>
READ DATA FROM LDU 1 AND 2
READ DATA FROM DR-11

READ DATA FROM DR-11

; IBUFA, DATA : BRING IN DATA WORDS AND DISCARD
; IBUFA, (R1)+
; BRING IN DATA WORD N
; SET START TIME
; SARE TIME OF DATA READ
; Btale of DATA READ
; BRUTE REQUESTS FOR WRITES
; GO EXIT
; NO, GO EXIT

* READ DATA FROM DR-11

1. MOV #4, CNTRA
   ; SET WORD COUNT
2. MOV IBUFA, DATA
3. DEC CNTRA
4. RTI ; EXIT
5. READ DATA FROM DR-11 A
6. * READ DATA FROM DR-11

7. MOV #4, CNTRA
   ; SET WORD COUNT
8. MOV IBUFA, DATA
9. DEC CNTRA
10. RTI ; EXIT
11. READ DATA FROM DR-11
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

* WRIT2:
  TST CNTR
  BEQ WRITS
  ; Error in last write
  ERROR IN LAST WRITE

* WRIT3:
  MOU #4,R0
  ; Pick up error code
  HALT

* WRIT4:
  WRIT10:
  MOV #2,R4
  ; Pick up loop counter

  WRIT7:
  MOV (R2)+,R3
  TRY BUFFER N

  WRIT6:
  DEC R1
  ; No, finished this loop
  NO, TRY NEXT ONE

  WRIT5:
  CLR CNTR
  ; Yes, set LDV indicator for next loop

  WRIT4:
  TST R0
  ; LDV 1 or 2

  WRIT3:
  MOV (R2)+,R3
  BUFFER WAITING ON WRITE
  YES
READ DATA FROM LDV 1 AND 2
WRITE BUFFERS TO DISK

58 001140 001405 BEQ WRIT3 :LDU 1
59
60 001142 005763 000000 TST CNT(R3) :LDU 2
61 001146 100366 BPL WRIT :WRONG BUFFER
62 001150 001667 000005 JMP WRITS :BUFFER OK, GO WRITE IT
63
64 001154 WRIT9:
65 001154 005763 000000 TST CNT(R3) :RIGHT BUFFER
66 001160 100751 BMI WRITS :WRONG, GO TRY AGAIN
67
68 001162 WRITS:
69 001162 016005 023462' MOU NEXT(R0),RS :FIRST TIME
70 001166 001410 BEQ WRITSA :FIRST TIME FOR LDV 2
71 001170 005305 DEC R5
72 001174 005205 INC R5
73 001176 166305 000006 SUB PNNP(R3),RS
74 001200 005305 INC R5
75 001202 016360 000006 ADD #2,R5
76 001206 001336 BNE WRITS :NOT NEXT IN SEQUENCE, TRY AGAIN
77 001210 WRITSA:
78 001210 016360 000005 023462' MOU PNNP(R3),NEXT(R0)
79 001214 016367 000006 022246 MOU PNNP(R3),SECNO
80 001218 010367 000014 MOU R3,WADR
81 001222 005667 000014 CLR CNTW
82 001226 004567 000005 JSR RS,LYTRAN
83 00122A 023536,' .WORD FILLN
84 00122C 023472' .WORD SECO
85 001244 000000 HADR:
86 001248 000002 MOU RO,R2
87 001250 000000 CLR LOCK
88 001258 000207 RTS PC :RETURN
89 00125A 012767 000001 022176 MOU #1,FWRIT
90 001260 005327 000002 000040 BIT RNET(R3),#EOF
91 001264 001404 BEQ WRITS2
92 001268 005260 000006 INC FTOR(R0)
93 001272 005267 000005 INC FCNTW
94 001300 WRITS2:
95 001300 005367 022170 DEC REQUST
96 001304 010002 MOU R0,R2
97 001306 005067 000024 CLR LOCK
98 001312 002007 RTS PC:RETURN
READ DATA FROM LDV 1 AND 2
WRITE BLOCK NO.

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>001314</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>001314</td>
<td>005750 023524'</td>
</tr>
<tr>
<td>10</td>
<td>001320</td>
<td>001403</td>
</tr>
<tr>
<td>11</td>
<td>001322</td>
<td>002750 000002 023466'</td>
</tr>
<tr>
<td>12</td>
<td>001330</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>001330</td>
<td>015004 023466'</td>
</tr>
<tr>
<td>14</td>
<td>001334</td>
<td>000207</td>
</tr>
</tbody>
</table>

CALCULATE NEXT WRITE BLOCK

RO = 0, LDV 1
RO = 2, LDV 2

TST FURST(RO)
BEQ CAL1
ADD #2,LSECT(RO)
MOV LSECT(RO),R4
RTS PC
### INPUT BUFFERS

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>001336</td>
<td>000000</td>
</tr>
<tr>
<td>5</td>
<td>001340</td>
<td>000000</td>
</tr>
<tr>
<td>6</td>
<td>001342</td>
<td>000000</td>
</tr>
<tr>
<td>7</td>
<td>001344</td>
<td>001456</td>
</tr>
<tr>
<td>10</td>
<td>001344</td>
<td>001456</td>
</tr>
<tr>
<td>11</td>
<td>001346</td>
<td>002456</td>
</tr>
<tr>
<td>12</td>
<td>001350</td>
<td>003456</td>
</tr>
<tr>
<td>13</td>
<td>001352</td>
<td>004456</td>
</tr>
<tr>
<td>14</td>
<td>001354</td>
<td>005456</td>
</tr>
<tr>
<td>15</td>
<td>001356</td>
<td>006456</td>
</tr>
<tr>
<td>16</td>
<td>001358</td>
<td>007456</td>
</tr>
<tr>
<td>17</td>
<td>001360</td>
<td>008456</td>
</tr>
<tr>
<td>18</td>
<td>001362</td>
<td>009456</td>
</tr>
<tr>
<td>19</td>
<td>001364</td>
<td>010456</td>
</tr>
<tr>
<td>20</td>
<td>001366</td>
<td>011456</td>
</tr>
<tr>
<td>21</td>
<td>001368</td>
<td>012456</td>
</tr>
<tr>
<td>22</td>
<td>001370</td>
<td>013456</td>
</tr>
<tr>
<td>23</td>
<td>001372</td>
<td>014456</td>
</tr>
<tr>
<td>24</td>
<td>001374</td>
<td>015456</td>
</tr>
<tr>
<td>25</td>
<td>001376</td>
<td>016456</td>
</tr>
<tr>
<td>26</td>
<td>001378</td>
<td>017456</td>
</tr>
<tr>
<td>27</td>
<td>001380</td>
<td>018456</td>
</tr>
<tr>
<td>28</td>
<td>001382</td>
<td>019456</td>
</tr>
<tr>
<td>29</td>
<td>001384</td>
<td>020456</td>
</tr>
<tr>
<td>30</td>
<td>001386</td>
<td>021456</td>
</tr>
<tr>
<td>31</td>
<td>001388</td>
<td>022456</td>
</tr>
</tbody>
</table>

### STORES:

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>001410</td>
<td>001476</td>
</tr>
<tr>
<td>31</td>
<td>001412</td>
<td>002476</td>
</tr>
<tr>
<td>32</td>
<td>001414</td>
<td>003476</td>
</tr>
<tr>
<td>33</td>
<td>001416</td>
<td>004476</td>
</tr>
<tr>
<td>34</td>
<td>001420</td>
<td>005476</td>
</tr>
<tr>
<td>35</td>
<td>001422</td>
<td>006476</td>
</tr>
<tr>
<td>36</td>
<td>001424</td>
<td>007476</td>
</tr>
<tr>
<td>37</td>
<td>001426</td>
<td>008476</td>
</tr>
<tr>
<td>38</td>
<td>001430</td>
<td>009476</td>
</tr>
<tr>
<td>39</td>
<td>001432</td>
<td>010476</td>
</tr>
<tr>
<td>40</td>
<td>001434</td>
<td>011476</td>
</tr>
<tr>
<td>41</td>
<td>001436</td>
<td>012476</td>
</tr>
<tr>
<td>42</td>
<td>001438</td>
<td>013476</td>
</tr>
<tr>
<td>43</td>
<td>001440</td>
<td>014476</td>
</tr>
<tr>
<td>44</td>
<td>001442</td>
<td>015476</td>
</tr>
<tr>
<td>45</td>
<td>001444</td>
<td>016476</td>
</tr>
<tr>
<td>46</td>
<td>001446</td>
<td>017476</td>
</tr>
<tr>
<td>47</td>
<td>001450</td>
<td>018476</td>
</tr>
<tr>
<td>48</td>
<td>001452</td>
<td>019476</td>
</tr>
</tbody>
</table>

### BUFFER START ADDRESSES

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>001410</td>
<td>001476</td>
</tr>
<tr>
<td>31</td>
<td>001412</td>
<td>002476</td>
</tr>
<tr>
<td>32</td>
<td>001414</td>
<td>003476</td>
</tr>
<tr>
<td>33</td>
<td>001416</td>
<td>004476</td>
</tr>
<tr>
<td>34</td>
<td>001420</td>
<td>005476</td>
</tr>
<tr>
<td>35</td>
<td>001422</td>
<td>006476</td>
</tr>
<tr>
<td>36</td>
<td>001424</td>
<td>007476</td>
</tr>
<tr>
<td>37</td>
<td>001426</td>
<td>008476</td>
</tr>
<tr>
<td>38</td>
<td>001430</td>
<td>009476</td>
</tr>
<tr>
<td>39</td>
<td>001432</td>
<td>010476</td>
</tr>
<tr>
<td>40</td>
<td>001434</td>
<td>011476</td>
</tr>
<tr>
<td>41</td>
<td>001436</td>
<td>012476</td>
</tr>
<tr>
<td>42</td>
<td>001438</td>
<td>013476</td>
</tr>
<tr>
<td>43</td>
<td>001440</td>
<td>014476</td>
</tr>
<tr>
<td>44</td>
<td>001442</td>
<td>015476</td>
</tr>
<tr>
<td>45</td>
<td>001444</td>
<td>016476</td>
</tr>
<tr>
<td>46</td>
<td>001446</td>
<td>017476</td>
</tr>
<tr>
<td>47</td>
<td>001450</td>
<td>018476</td>
</tr>
</tbody>
</table>

### BUFFER STORE ADDRESSES

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>001410</td>
<td>001476</td>
</tr>
<tr>
<td>31</td>
<td>001412</td>
<td>002476</td>
</tr>
<tr>
<td>32</td>
<td>001414</td>
<td>003476</td>
</tr>
<tr>
<td>33</td>
<td>001416</td>
<td>004476</td>
</tr>
<tr>
<td>34</td>
<td>001420</td>
<td>005476</td>
</tr>
<tr>
<td>35</td>
<td>001422</td>
<td>006476</td>
</tr>
<tr>
<td>36</td>
<td>001424</td>
<td>007476</td>
</tr>
<tr>
<td>37</td>
<td>001426</td>
<td>008476</td>
</tr>
<tr>
<td>38</td>
<td>001430</td>
<td>009476</td>
</tr>
<tr>
<td>39</td>
<td>001432</td>
<td>010476</td>
</tr>
<tr>
<td>40</td>
<td>001434</td>
<td>011476</td>
</tr>
<tr>
<td>41</td>
<td>001436</td>
<td>012476</td>
</tr>
<tr>
<td>42</td>
<td>001438</td>
<td>013476</td>
</tr>
<tr>
<td>43</td>
<td>001440</td>
<td>014476</td>
</tr>
<tr>
<td>44</td>
<td>001442</td>
<td>015476</td>
</tr>
<tr>
<td>45</td>
<td>001444</td>
<td>016476</td>
</tr>
<tr>
<td>46</td>
<td>001446</td>
<td>017476</td>
</tr>
<tr>
<td>47</td>
<td>001450</td>
<td>018476</td>
</tr>
</tbody>
</table>

### Locking Information

- **LDV1 Input Buffers**: Lock: .WORD 0, Store: .WORD 0
- **LDV2 Input Buffers**: Lock: .WORD 0, Store: .WORD 0
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>003476</td>
<td>ISA3A</td>
</tr>
<tr>
<td>06</td>
<td>004456</td>
<td>IEA4A</td>
</tr>
<tr>
<td>07</td>
<td>004476</td>
<td>ISA4A</td>
</tr>
<tr>
<td>08</td>
<td>005456</td>
<td>IEA1B</td>
</tr>
<tr>
<td>09</td>
<td>005476</td>
<td>ISA1B</td>
</tr>
<tr>
<td>10</td>
<td>006456</td>
<td>ISA2A</td>
</tr>
<tr>
<td>11</td>
<td>007456</td>
<td>IEA2B</td>
</tr>
<tr>
<td>12</td>
<td>010456</td>
<td>ISA3B</td>
</tr>
<tr>
<td>13</td>
<td>010476</td>
<td>IEA3B</td>
</tr>
<tr>
<td>14</td>
<td>011456</td>
<td>ISA4B</td>
</tr>
<tr>
<td>15</td>
<td>011476</td>
<td>IEA4B</td>
</tr>
<tr>
<td>16</td>
<td>012456</td>
<td>IEA5B</td>
</tr>
<tr>
<td>17</td>
<td>012476</td>
<td>ISA5B</td>
</tr>
<tr>
<td>18</td>
<td>013456</td>
<td>IEA6B</td>
</tr>
<tr>
<td>19</td>
<td>013476</td>
<td>ISA6B</td>
</tr>
<tr>
<td>20</td>
<td>014456</td>
<td>IEA7B</td>
</tr>
<tr>
<td>21</td>
<td>014476</td>
<td>ISA7B</td>
</tr>
<tr>
<td>22</td>
<td>015456</td>
<td>IEA8B</td>
</tr>
<tr>
<td>23</td>
<td>015476</td>
<td>ISA8B</td>
</tr>
<tr>
<td>24</td>
<td>016456</td>
<td>IEA9B</td>
</tr>
<tr>
<td>25</td>
<td>016476</td>
<td>ISA9B</td>
</tr>
<tr>
<td>26</td>
<td>017456</td>
<td>IEB1A</td>
</tr>
<tr>
<td>27</td>
<td>017476</td>
<td>ISB1A</td>
</tr>
<tr>
<td>28</td>
<td>018456</td>
<td>IEB2A</td>
</tr>
<tr>
<td>29</td>
<td>018476</td>
<td>ISB2A</td>
</tr>
<tr>
<td>30</td>
<td>019456</td>
<td>IEB3A</td>
</tr>
<tr>
<td>31</td>
<td>019476</td>
<td>ISB3A</td>
</tr>
<tr>
<td>32</td>
<td>020456</td>
<td>IEB4A</td>
</tr>
<tr>
<td>33</td>
<td>020476</td>
<td>ISB4A</td>
</tr>
<tr>
<td>34</td>
<td>021456</td>
<td>IEB5A</td>
</tr>
<tr>
<td>35</td>
<td>021476</td>
<td>ISB5A</td>
</tr>
<tr>
<td>36</td>
<td>022456</td>
<td>IEB6A</td>
</tr>
<tr>
<td>37</td>
<td>022476</td>
<td>ISB6A</td>
</tr>
<tr>
<td>38</td>
<td>023456</td>
<td>IEB7A</td>
</tr>
<tr>
<td>39</td>
<td>023476</td>
<td>ISB7A</td>
</tr>
<tr>
<td>40</td>
<td>024456</td>
<td>IEB8A</td>
</tr>
<tr>
<td>41</td>
<td>024476</td>
<td>ISB8A</td>
</tr>
<tr>
<td>42</td>
<td>025456</td>
<td>IEB9A</td>
</tr>
<tr>
<td>43</td>
<td>025476</td>
<td>ISB9A</td>
</tr>
<tr>
<td>44</td>
<td>026456</td>
<td>IEB10A</td>
</tr>
<tr>
<td>45</td>
<td>026476</td>
<td>ISB10A</td>
</tr>
<tr>
<td>46</td>
<td>027456</td>
<td>IEC1A</td>
</tr>
<tr>
<td>47</td>
<td>027476</td>
<td>ISC1A</td>
</tr>
<tr>
<td>48</td>
<td>028456</td>
<td>IEC2A</td>
</tr>
<tr>
<td>49</td>
<td>028476</td>
<td>ISC2A</td>
</tr>
<tr>
<td>Line</td>
<td>ASCII Hex</td>
<td>Decimal</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>023456</td>
<td>818254</td>
</tr>
<tr>
<td>2</td>
<td>023460</td>
<td>818258</td>
</tr>
<tr>
<td>3</td>
<td>023462</td>
<td>818260</td>
</tr>
<tr>
<td>4</td>
<td>023466</td>
<td>818262</td>
</tr>
<tr>
<td>5</td>
<td>023468</td>
<td>818264</td>
</tr>
<tr>
<td>6</td>
<td>023472</td>
<td>818268</td>
</tr>
<tr>
<td>7</td>
<td>023474</td>
<td>818270</td>
</tr>
<tr>
<td>8</td>
<td>023476</td>
<td>818272</td>
</tr>
<tr>
<td>9</td>
<td>023480</td>
<td>818276</td>
</tr>
<tr>
<td>10</td>
<td>023484</td>
<td>818280</td>
</tr>
<tr>
<td>11</td>
<td>023488</td>
<td>818284</td>
</tr>
<tr>
<td>12</td>
<td>023492</td>
<td>818288</td>
</tr>
<tr>
<td>13</td>
<td>023496</td>
<td>818292</td>
</tr>
<tr>
<td>14</td>
<td>023500</td>
<td>818296</td>
</tr>
<tr>
<td>15</td>
<td>023504</td>
<td>818300</td>
</tr>
<tr>
<td>16</td>
<td>023508</td>
<td>818304</td>
</tr>
<tr>
<td>17</td>
<td>023510</td>
<td>818308</td>
</tr>
<tr>
<td>18</td>
<td>023514</td>
<td>818312</td>
</tr>
<tr>
<td>19</td>
<td>023520</td>
<td>818318</td>
</tr>
<tr>
<td>20</td>
<td>023530</td>
<td>818328</td>
</tr>
<tr>
<td>21</td>
<td>023532</td>
<td>818330</td>
</tr>
<tr>
<td>22</td>
<td>023534</td>
<td>818332</td>
</tr>
<tr>
<td>23</td>
<td>023536</td>
<td>818334</td>
</tr>
<tr>
<td>24</td>
<td>023540</td>
<td>818338</td>
</tr>
<tr>
<td>25</td>
<td>023542</td>
<td>818340</td>
</tr>
<tr>
<td>26</td>
<td>023544</td>
<td>818342</td>
</tr>
<tr>
<td>27</td>
<td>023540</td>
<td>818348</td>
</tr>
<tr>
<td>28</td>
<td>023542</td>
<td>818350</td>
</tr>
</tbody>
</table>

### Constants

- **FWRIT**: .WORD 0
- **BUSY**: .WORD 0
- **NEXT**: .WORD 0.1
- **LSECT**: .WORD 0.1
- **REQST**: .WORD 0
- **HEADR**: .WORD 0
- **FRAMES**: .WORD 0
- **FRAMEA**: .WORD 0
- **FRAMEB**: .WORD 0
- **SFLAGA**: .WORD 0
- **SFLAGB**: .WORD 0
- **FIRSTA**: .WORD 0
- **FIRSTB**: .WORD 0
- **BUFSAT**: .BLKW 4
- **FURST**: .WORD 0.0
- **DATA**: .WORD 0
- **CNTRB**: .WORD 0
- **FILN**: .RADSO /LDVS/
- **FIN**: .WORD 0
### Symbol Table

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB</td>
<td>000230</td>
</tr>
<tr>
<td>BUFSAT</td>
<td>023514RG</td>
</tr>
<tr>
<td>BU</td>
<td>000100</td>
</tr>
<tr>
<td>CNT</td>
<td>000000</td>
</tr>
<tr>
<td>CNTW</td>
<td>001250R</td>
</tr>
<tr>
<td>EOF</td>
<td>000020</td>
</tr>
<tr>
<td>FILN</td>
<td>023536RG</td>
</tr>
<tr>
<td>FIRSTB</td>
<td>023512R</td>
</tr>
<tr>
<td>FRAMEB</td>
<td>023502R</td>
</tr>
<tr>
<td>FNRST</td>
<td>023524RG</td>
</tr>
<tr>
<td>HOM</td>
<td>000000</td>
</tr>
<tr>
<td>IBUBF</td>
<td>167764</td>
</tr>
<tr>
<td>IECA</td>
<td>002456R</td>
</tr>
<tr>
<td>IEGB</td>
<td>000456R</td>
</tr>
<tr>
<td>IEBA</td>
<td>011456R</td>
</tr>
<tr>
<td>IEBB</td>
<td>016456R</td>
</tr>
<tr>
<td>IEBA</td>
<td>014456R</td>
</tr>
<tr>
<td>IECA</td>
<td>022456R</td>
</tr>
<tr>
<td>INTAB</td>
<td>000126RG</td>
</tr>
<tr>
<td>INTA3</td>
<td>000432R</td>
</tr>
<tr>
<td>INTP</td>
<td>000474R</td>
</tr>
<tr>
<td>INB</td>
<td>000526RG</td>
</tr>
<tr>
<td>ISAB</td>
<td>005476R</td>
</tr>
<tr>
<td>ISA2A</td>
<td>006476R</td>
</tr>
<tr>
<td>ISAA4B</td>
<td>003476R</td>
</tr>
<tr>
<td>ISA3B</td>
<td>013476R</td>
</tr>
<tr>
<td>ISAB4</td>
<td>017456R</td>
</tr>
<tr>
<td>ISCA</td>
<td>021476R</td>
</tr>
<tr>
<td>LEUC</td>
<td>000340R</td>
</tr>
<tr>
<td>LOCK</td>
<td>001336R</td>
</tr>
<tr>
<td>NO3</td>
<td>00074R</td>
</tr>
<tr>
<td>REG</td>
<td>023474R</td>
</tr>
<tr>
<td>RE</td>
<td>0000000</td>
</tr>
<tr>
<td>RE</td>
<td>0000000</td>
</tr>
<tr>
<td>SGW</td>
<td>00562RG</td>
</tr>
<tr>
<td>SGW4</td>
<td>00052R</td>
</tr>
<tr>
<td>SFLAGC</td>
<td>023504RG</td>
</tr>
<tr>
<td>SSSID</td>
<td>000370R</td>
</tr>
<tr>
<td>STORB</td>
<td>001342RG</td>
</tr>
<tr>
<td>SUR</td>
<td>177650</td>
</tr>
<tr>
<td>TMS1</td>
<td>000010</td>
</tr>
<tr>
<td>TMS2</td>
<td>005246</td>
</tr>
<tr>
<td>HAP</td>
<td>006736R</td>
</tr>
<tr>
<td>WRITA</td>
<td>001210R</td>
</tr>
<tr>
<td>WRITX</td>
<td>001034R</td>
</tr>
<tr>
<td>WRIT3</td>
<td>001072R</td>
</tr>
<tr>
<td>WRIT6</td>
<td>00114R</td>
</tr>
<tr>
<td>WRIT9</td>
<td>000000</td>
</tr>
</tbody>
</table>

### Error Status
- **ERRORS DETECTED**: 0
- **FREE CORE**: 11556. WORDS

**READ.L1 < READ.1**
APPENDIX C
FILL LISTINGS
TABLE OF CONTENTS

4-1 READ DATA FROM DISK
5-1 STORE DATA WORDS
6-1 SET MAXIMUM VELOCITY
7-1 READ BLOCK OF DATA
FILL PROCESS BUFFER

.MACRO UR05-01A 01-JAN-72 00:18 PAGE 1

1 .NLIST TIM

2 .TITLE FILL PROCESS BUFFER

3 .GLOBL FTOR

4 .GLOBL NEXT

5 .GLOBL FILL,BUSY,FIN,REQUEST

6 .GLOBL ITRAN,INK,BLK,WRITE

7 .GLOBL IFLD,LRSEC

8 .GLOBL BUF1

9 .GLOBL FCNTW,FCNTF,FILN

10 .MCALL .PARAM,.WAITR

11 000000 .PARAM
1. MACRO PUSH
   MOV R0, -(SP)
   MOV R1, -(SP)
   MOV R2, -(SP)
   MOV R3, -(SP)
   MOV R4, -(SP)
   MOV RS, -(SP)
   .ENDM

   ;SAVE REGISTERS

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

   ;RESTORE REGISTERS
<table>
<thead>
<tr>
<th>Line</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000000</td>
<td>ONE = 0</td>
</tr>
<tr>
<td>2</td>
<td>003720</td>
<td>TWO = 2000.</td>
</tr>
<tr>
<td>3</td>
<td>007640</td>
<td>THREE = 4000.</td>
</tr>
<tr>
<td>4</td>
<td>013956</td>
<td>FOUR = 6000.</td>
</tr>
<tr>
<td>5</td>
<td>001740</td>
<td>MAXTS = 932.</td>
</tr>
<tr>
<td>6</td>
<td>000000</td>
<td>CNT = 0</td>
</tr>
<tr>
<td>7</td>
<td>000002</td>
<td>B4ST = 2</td>
</tr>
<tr>
<td>8</td>
<td>000004</td>
<td>HDRW = 4</td>
</tr>
<tr>
<td>9</td>
<td>000006</td>
<td>MPNTS = 1</td>
</tr>
<tr>
<td>10</td>
<td>000010</td>
<td>EOF = 40</td>
</tr>
<tr>
<td>11</td>
<td>000012</td>
<td>TIMS1 = 8</td>
</tr>
<tr>
<td>12</td>
<td>000014</td>
<td>TIMS2 = 10</td>
</tr>
<tr>
<td>13</td>
<td>000016</td>
<td>TIMR1 = 12</td>
</tr>
<tr>
<td>14</td>
<td>000020</td>
<td>TIMR2 = 14</td>
</tr>
<tr>
<td>15</td>
<td>000040</td>
<td>BB = 200</td>
</tr>
<tr>
<td>16</td>
<td>000010</td>
<td>SIZE = 8</td>
</tr>
<tr>
<td>17</td>
<td>000370</td>
<td>SIZD = 246</td>
</tr>
<tr>
<td>18</td>
<td>000340</td>
<td>LEUW = 340</td>
</tr>
</tbody>
</table>
READ DATA FROM DISK

* READ 1000 WORDS OR FRAME OF DATA
* FORMAT FOR DATA PROCESSING PROGRAM

FILL DATA BUFFER

FILL:
  PUSH

FILL0:
  CLR IEOFI
  CMP FCNTW, FCNTR
  BHI FILL1
  NO, GO READ ONE

  TST FIN
  BNE FILLE
  YES, TEST OVER

  CMP FCNTW, FCNTR
  BHI FILL1
  ALL FRAMES READ
  NO, WAIT FOR A FRAME

  INC IEOFI
  YES, SET END OF TEST FLAG

FILLE:
  JMP FIXT

FILLE1:
  MOV IFLD, R1
  BEQ SECTS
  PICK UP INHIBIT FLAG

  SUB #2, R1
  BEQ SECT2
  BOTH LDVS

  MOU #2, R1
  BEQ SECT2
  LDV 1 ONLY

  MOU FTOH, R2
  BEQ SECT2
  LDV 2 ONLY

SECTS:
  MOV FTOH+2, R2
  BEQ SECT3
  NMO FTOH+2, R2
  SUB R1, R2
  BMI SECT3
  BR SECT6

  BR SECT2

SECT5:
  MOV FTOH, R1
  BEQ SECT2
  MOU FTOH, R1
  BEQ SECT2
  MOU FTOH+2, R2

SECT6:
  BIT READST, #1

SECT3:
  CLR R2

SECT4:
READ DATA FROM DISK

58 000152 016267 003105 002724
59 000150 016267 002726
60 000156 016267 177776
61 000170 016267 003240 177776
62 000176 016267 000000
63 000174 016267 177776
64 000210 005067 002700
65 000214 005067 002767
66 000220 005067 002700
67 000224 005067 002767
68 000230 005067 000000
69 000236 005067 002666
70 000242 005703 000001
71 000246 005703 000002
72 000248 005703 000004
73 000250 005703 000004
74 000252 005703 000005
75 000254 005703 000005
76 000256 005703 000006
77 000258 005703 000006
78 00025a 005703 000007
79 00025c 005703 000007
80 00025e 005703 000008
81 000260 005703 000008
82 000262 005703 000009
83 000264 005703 000009
84 000266 005703 00000a
85 000268 005703 00000a
86 00026a 005703 00000b
87 00026c 005703 00000b
88 00026e 005703 00000c
89 000270 005703 00000c
90 000272 005703 00000d
91 000274 005703 00000d
92 000276 005703 00000e
93 000278 005703 00000e
94 00027a 005703 00000f
95 00027c 005703 00000f
96 00027e 005703 000010
97 000280 005703 000010
98 000282 005703 000011
99 000284 005703 000011
100 000286 005703 000012
101 000288 005703 000012
102 00028a 005703 000013
103 00028c 005703 000013
104 00028e 005703 000014
105 000290 005703 000014
106 000292 005703 000015
107 000294 005703 000015
108 000296 005703 000016
109 000298 005703 000016
110 00029a 005703 000017
111 00029c 005703 000017
112 0002ac 005703 000018
113 0002ac 005703 000018

FILL PROCESS BUFFER MACRO UR06-01A 01-JAN-72 00:18 PAGE 4+
FILL PROCESS BUFFER
READ DATA FROM DISK

MACRO UROS-01A 01-JAN-72 03:18 PAGE 4+

115 000434 001432 000430
116 000436 016702 000430
117 000442 000442
118 000442 004767 000144
119 000442 004767 000162
120 000452 005724
121 000454 005201
122 000456 005303
123 000460 001370
124 000465
125 000465 012702 002100'
126 000466 004767 000225
127 000472 035227 000092 000460
128 000500 001010
129 000502 001740
130 000506 103557
131 000509 012767 177777 000004'
132 000516 000167 000004
133 000522
134 000522 005267 002372
135 000526
136 000526
137 000526 005057 000006
138 000532 016702 002354
139 000536 005724 002342 003106'
140 000544 005362 003100'
141 000550 005301
142 000552 001002
143 000554 001557 177234
144 000554 001557 177234
145 000556
146 000556 010167 000020'
147 000564
148 000564
149 000600 000205
150 000602
151 000602 012700 000005
152 000606 001677 000000'
FILL PROCESS BUFFER
STORE DATA WORDS

1

2

3

4

5 000612

6 000612 012264 000000

7 000616 012264 003720

8 000622 012264 007640

9 000622 012264 013560

10 000622 000207

.SBTTL STORE DATA WORDS

** FORMAT AND STORE DATA WORDS **

STORE:

MOV (R2)+,ONE(R4) ; FIRST DATA WORD X

MOV (R2)+,TWO(R4) ; SECOND DATA WORD Y

MOV (R2)+,THREE(R4) ; THIRD DATA WORD N/I

MOV (R2)+,FOUR(R4) ; FOURTH DATA WORD U/V

RTS PC ; RETURN
SET MAXIMUM VELOCITY

; Set maximum velocity

1 CMP FOUR(R4), MAXV1 ; Velocity higher than previous
2 BLS MAXV1 ; No
3 MOU R4, MAXV1 ; Yes, replace last highest
4 MOU R4, MAXV2 ; Set current max number
5 BR MNAXU2 ;

; Find maximum velocity

6 CMP FOUR(R4), MAXV2 ; Higher than previous
7 BLS MAXV2 ; No
8 MOU R4, MAXV2 ; Yes, replace highest
9 MOU R4, MAX2
10 RTS PC ;

; Timer

11 PUSH TIMR1(R2), TMEND
12 MOU TIMR2(R2), TMEND+2
13 MOU CNT(R2), R4
14 ASH #8, R4
15 MOU R4, IFRM
16 RTS PC

; Filler

17 FILL PROCESS BUFFER
18 SET MAXIMUM VELOCITY
19 MACRO VR05-01A 01-JAN-72 00:18 PAGE 6
20
READ BLOCK OF DATA

* * READ NEXT BLOCK OF DATA FROM DISK

READ:
1. ** TST FFG
2. ** BED R11
3. ** ADD #2, READST
4. SET READ SECTOR NUMBER
5. ** MOV READRS(R0), READAD
6. SET READ ADDRESS
7. ** MOV #1, FFG
8. ** CLR CNTR
9. ** JSR R5, LKTRAN
10. ** CLEAR ERROR/FUNCTION

READ3:
1. ** TST CNTR
2. ** BEQ READ3
3. ** JMP FIXET
4. ** ERROR ON LAST READ

READ1:
1. ** RTS PC

READ2:
1. ** WORD BUF1+16
2. ** READ BUFFERS

READS:
1. ** WORD BUF1+16
2. ** READ BUFFERS

READAD:
1. ** WORD 0
2. ** SET READ ADDRESS
3. ** READ

CNTR:
1. ** WORD 0
2. ** CLEAR ERROR/FUNCTION

TST:
1. ** WORD FILN

R11:
1. ** MOV READRS(R0), READAD
2. ** SET READ ADDRESS
3. ** MOV #1, FFG
4. ** CLR CNTR
5. ** JSR R5, LKTRAN
6. ** CLEAR ERROR/FUNCTION

RTS:
1. ** WORD PRODAT
2. ** PROCESS BUFFER ADDRESS

PC:
1. ** WORD 0
2. ** READ

BUF1:
1. ** BLKW SIZEB
2. ** BLKW SIZD

BUF2:
1. ** BLKW SIZEB
2. ** BLKW SIZD

FTOR:
1. ** WORD 0
2. ** READ

READST:
1. ** READ
2. ** READ

FPG:
1. ** READ
2. ** READ

FCNT1:
1. ** READ
2. ** READ

MAX1:
1. ** READ
2. ** READ

MAX2:
1. ** READ
2. ** READ

FDAT:
1. ** READ
2. ** READ

CSECT:
1. ** CSECT ABTERM
2. ** CSECT ABTERM

CSECT:
1. ** CSECT IFLDV
2. ** CSECT IFLDV

CSECT:
1. ** CSECT IHDL1
2. ** CSECT IHDL1
FILL PROCESS BUFFER  MACRO VR05-01A 01-JAN-72 00:18 PAGE 7+
READ BLOCK OF DATA

58  000002  000000
59  000004  000000
60  000000  0
61  000000  0
62  000000  0
63  000000  0
64  000004  000000  000000
65  000010  000000  000000
66  000014  000000
67  000016  000000
68  000020  000000
69  000001  0

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
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>ADDRESS</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABTRM</td>
<td>000000R</td>
<td>002</td>
<td>BB = 000000</td>
</tr>
<tr>
<td>BUF1</td>
<td>001100R</td>
<td>001</td>
<td>BUF2 = 002100</td>
</tr>
<tr>
<td>CMP</td>
<td>000000R</td>
<td>021</td>
<td>EOF = 000000</td>
</tr>
<tr>
<td>FCNTF</td>
<td>003120R</td>
<td>026</td>
<td>FCNTF = 003122</td>
</tr>
<tr>
<td>FILEG</td>
<td>003115R</td>
<td>022</td>
<td>FILEDEF = 000522</td>
</tr>
<tr>
<td>FILLA</td>
<td>000050R</td>
<td>026</td>
<td>FILLB = 000322</td>
</tr>
<tr>
<td>FILLL</td>
<td>000014R</td>
<td>020</td>
<td>FILL1 = 000070</td>
</tr>
<tr>
<td>FILL3</td>
<td>000322R</td>
<td>020</td>
<td>FILL6 = 000410</td>
</tr>
<tr>
<td>FILN</td>
<td>..........</td>
<td>007</td>
<td>FIN = ******** G</td>
</tr>
<tr>
<td>FIXT</td>
<td>000564R</td>
<td>005</td>
<td>FIXET = 000526</td>
</tr>
<tr>
<td>FOUR</td>
<td>013560R</td>
<td>005</td>
<td>FTR = 003100</td>
</tr>
<tr>
<td>HLT</td>
<td>000050R</td>
<td>005</td>
<td>IDAY = 000014</td>
</tr>
<tr>
<td>IFLD</td>
<td>000000R</td>
<td>003</td>
<td>IFLY = 000000</td>
</tr>
<tr>
<td>IFUN</td>
<td>000016R</td>
<td>005</td>
<td>LDUDAT = 000000</td>
</tr>
<tr>
<td>LKTRAN</td>
<td>********</td>
<td>004</td>
<td>LNKBLK = ******** G</td>
</tr>
<tr>
<td>MAXV</td>
<td>000634R</td>
<td>004</td>
<td>MAX1 = 000674</td>
</tr>
<tr>
<td>MAXW1</td>
<td>000314R</td>
<td>004</td>
<td>MAX12 = 000326</td>
</tr>
<tr>
<td>MAX2</td>
<td>000022R</td>
<td>004</td>
<td>MPNTS = 000074</td>
</tr>
<tr>
<td>NUMPTS</td>
<td>000020R</td>
<td>005</td>
<td>ONE = 000000</td>
</tr>
<tr>
<td>PRODAT</td>
<td>003130R</td>
<td>005</td>
<td>PSW = 177776</td>
</tr>
<tr>
<td>READ0</td>
<td>001046R</td>
<td>005</td>
<td>READ1 = 001070</td>
</tr>
<tr>
<td>READ1</td>
<td>001070R</td>
<td>005</td>
<td>READ2 = 001072</td>
</tr>
<tr>
<td>REDEX</td>
<td>003112R</td>
<td>005</td>
<td>REJDT = ******** G</td>
</tr>
<tr>
<td>R1</td>
<td>000061</td>
<td>005</td>
<td>R11 = 000166</td>
</tr>
<tr>
<td>R3</td>
<td>000063</td>
<td>005</td>
<td>R4 = 000004</td>
</tr>
<tr>
<td>R5</td>
<td>000066</td>
<td>005</td>
<td>R7 = 000007</td>
</tr>
<tr>
<td>SECT3</td>
<td>000150R</td>
<td>005</td>
<td>SECT4 = 000152</td>
</tr>
<tr>
<td>SECT6</td>
<td>000312R</td>
<td>005</td>
<td>SECT5 = 000162</td>
</tr>
<tr>
<td>SP</td>
<td>000066</td>
<td>005</td>
<td>STORE = 000612</td>
</tr>
<tr>
<td>THREE</td>
<td>007640</td>
<td>005</td>
<td>TIFF = 001070</td>
</tr>
<tr>
<td>TMRed</td>
<td>000016</td>
<td>005</td>
<td>TIMS1 = 000010</td>
</tr>
<tr>
<td>WRITE</td>
<td>********</td>
<td>005</td>
<td>TIM2 = 000012</td>
</tr>
<tr>
<td></td>
<td>000000R</td>
<td>005</td>
<td>TINT = 000004</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 PLTSYM(10), IBUF(6)
COMMON XCG(2), YCG(2), INDEX(250), NCRPT(2), NOISES(2), PKVEL(2)
1, ELANG(2), NVORTX, KERM, RTIME
COMMON/BUFFER/IBUF1(20)
COMMON/TMPS/KDFTT(6), JAUTO
COMMON/DSPL/NX1, NX2, NX3, NS4, LDT, KSM, KL, LLDT
COMMON/ABTERM/IASTRM
COMMON/INF/PTTOL, NOISE, VELTOL, VORTOL, IIR, IIR2,
1 NRADI, ONXY, STIME, FRUTOL
COMMON /LOVDAT/IFLY, IFRM, ITMINT, ITMEND, IDAY, IPLN, NUMPTS,
1 IX(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON/HDLIM/MAX1, MAX2, IEDFI
DATA ITXOFF, TYOFF, XSTRT, YSTRT, RLX, RLY, IXL, IYL/
1 140, 173, 200, 0, 700, 400, 784, 442/
DATA IVCHAR/'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'/
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
KTWO=2
CALL DISPIO(14, KTWO, KL)
C
OUTPUT HEADER ON AUXILIARY SCREEN
NST=2
IF(NS4 .EQ. 2)NST=1
CALL DISPIO(11, NST)
CALL DISPIO(1, 36, IBUF1)
C
RE-SELECT SCREEN FOR SCATTER
CALL DISPIO(11, NS4)
C
22 CONTINUE
CALL DBGU
C
BRING UP BACKGROUND
CALL DISPIO(2, 13, 0)
C
ENCOD FRAME FOR SCATTER HEADER
KERM=TABS(IFRM)
CALL DECOD(KERM, 2, IBUF1(19))
IF(IFRM .LT. 0)IBUF1(20)=PLTSYM(3)
IF(IFRM .GE. 0)IBUF1(20)=PLTSYM(2)
C
BRING UP SCATTER HEADER
CALL DISPIO(1, 40, IBUF1)
C
STORE INDICES OF HIGHEST 10 VELOCITIES IN ARRAY IMX
DO 30 I=1, 10
30 IMX(I)=0
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(1.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 DECD(IDUM,4,LBUF(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 DECD(IDUM,3,LBUF(K))
40 CONTINUE
C OUTPUT VELOCITIES
CALL DISPIO(2,24,-1)
CALL DISPIO(1,30,LBUF)
C SELECT CHARACTER TO BE OUTPUT FOR EACH POINT AND OUTPUT
Ktwo=2
CALL DISPIO(14,KTWO,KSM)
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.6,6,2)
2 IF(VEL.LE.BK(J))GO TO 9
3 J=J+1
IF(J-11)4,4,7
4 IF(VEL.GT.BK(J))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)\times CONVY 
VF=IV(T)\times CONVY 
IBUF(5)=1 
IBUF(3)=IXOFF+(XF-XSTRT)\times XSCAL 
IBUF(4)=IVOFF+(VF-YSTRT)\times YSCAL 
CALL SETAD(IBUF(6), ICHAR) 
CALL DISPI0(4, 12, IBUF) 
1 CONTINUE 
C 
C CALCULATE CENTROID AND PLOT VORTEX POSITION WITH V 
KTWO=2 
CALL DISPI0(14, KTWO, KSM) 
CALL CENTRD 
KTWO=2 
CALL DISPI0(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)\times XSCAL 
IBUF(4)=IVOFF+(YCG(I)-YSTRT)\times YSCAL 
CALL SETAD(IBUF(6), ICHAR) 
CALL DISPI0(4, 12, IBUF) 
61 CONTINUE 
60 CONTINUE 
C 
C HARD COPY IF REQUESTED 
IF(JAUTO EQ. 1)GO TO 65 
CALL DISPI0(10, NS4) 
C 
C DELAY 
DO 11 I=1, 65 
DO 11 J=1, 2000 
11 BDUMB=(ADUMB+2.5)/3.6 
65 CONTINUE 
C 
C READ RECORD FROM TAPE 
CALL VREAD 
C 
C EXIT IF END OF FILE 
IF(IEOFI EQ. 1)GO TO 20 
C 
C EXIT IF 
IF(IABTRM. EQ, 33)GO TO 20 
GO TO 22 
20 RETURN 
END
SUBROUTINE DECO (KK, KKK, KP)
BYTE KP(5)
BYTE NEG
DATA NEG
II=0
K=0
KNNM=KKK
IF (KK) 10, 25, 25
10 KK=-KK
KP(1)=NEG
II=!
KNNM=KNNM-1
25 IF (KNNM<5) 25, 50, 200
35 K=KK/10**KNNM
50 DO 100 I=1, KNNM
100 KP(I+II)=K-KT*10+48
200 RETURN
END
SUBROUTINE VREAD

C THIS SUBROUTINE READS THE VORTEX DATA TAPE FOR POST-PROCESSING

COMMON BUFFER/ IBUF1(18)
COMMON/IDUMMY/NBA,NFL
COMMON /IFLDV/IFLDV
COMMON /CFL1/ CFL1
COMMON /IHOL1/ MAXI,MAX2,IEOFI
COMMON/TNFP/ RTTOL,NOISE, VELTOL, VORTOL, IRADI, IRADI2, NRADI,
 1 CONVX, STIME, FRVTOL
COMMON /LDUDAT/IFLY,IFRM, ITMINT(2),ITMEND(2), IDAY, IPLN, NUNPTS,
 1 IY(1000), IV(1000), INTENS(1000), IVEL(1000)
COMMON /PLNAME/ IDAT(60)
COMMON /ROUT/ ROUT, KRB
COMMON /SETUP/ DATE, JDEFT(20,6), JPLTP(3), IPLIND, IL1(60), TIM(2)
 1 IFLB(3), IFLD, TAHG, TL2(90), KRB(6)
INTEGER CFL1
DIMENSION BUF1(2), BUF2(2)

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

READ PHYSICAL RECORD FROM TAPE MT1
20 CONTINUE
CALL GET (1,1,IFLY,4010)
    WAIT FOR I/O OPERATION TO COMPLETE ON MT1
    CALL WAIT (1.0,N)
    GET RETURN STATUS FOR END-OF-FILE INDICATOR
CALL SFUN(1,7,NBA,IEOFI)
IF (IEOFI) 35,35,30
30 CONTINUE
    CALL DISPIO(3,8,BUF2)
    RETURN
35 CONTINUE
    CHECK FOR 4008 WORD PHYSICAL RECORD READ
IF(N-1) 50,50,40
40 CALL DISPIO(3,8,BUF1)
    RETURN
50 CONTINUE
    IF (((IFRM.GT.0) AND (IFLDV.EQ.1)) OR ((IFRM.LT.0) AND
1 (IFLDV.EQ.2))) GO TO 20
    DO 60 I=1,NUNPTS
60 IVEL(I)= IABS(VEL(I))
    ENCODE FLY-BY-NO
    CALL DECOD (IFLY,5,IFLB)
    ENCODE DAY
    CALL DECOD (IDAY,3,DAT)
    RETURN
END
SUBROUTINE DRUG
COMMON /ABTERM/ IABTRM
COMMON XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
PKUEL(2), ELVANG(2), NVORTX, KFRM, RTIME
COMMON /ATLP/ IBASE, DIMX
COMMON /BUFFER/ IBUF1(12), IBUF2(6), IBUF3(2)
COMMON /BUFFER/ IPER(12)
COMMON /CRTL/ KPROC, TACOLT, MTIFR
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /IADRS/ IADDY(26), IADPTS(2), IDATA(64), IADDAT, BLANK,
TABNK
COMMON /IHDLI/ MAX1, MAX2, IEGFI
COMMON /IFLDV/ IFLDV
COMMON/INPT/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRADII, NRADI,
CONXY, STIME, FRVTOL
COMMON /INITL/ ITPLT, PORTS(2), STARTS(2), VELMNF
COMMON /KSTFL/KSTFL, KPLN
COMMON /LDVAT/ IFLV, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 ITY(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON /OUT/ IOWT(6)
COMMON /PLNAME/ IDAT(60)
COMMON /ROUT/ ROUT, KGBF
COMMON /SEARCH/ ISFBN, ISDAY, ISP N
COMMON /SETUP/ DATE, JDFT(20, 6), JPLTP(3), IPLND, IL1(60), TIM(2),
1 IFLR(3), IFLD, IAHC, IL2(90), KREVF(6)
COMMON /TMPS/ KDFT(6), JAUTO
COMMON /XCORDT/KX1, KX2, KX3, KY4
BYTE IDATA
INTEGER BLANK
DATA M/6/
CALL SSWITCH(0, K)
IF (K NE 1) RETURN
CALL DISPIO(7, 1)
KTWO=2
CALL DISPIO(14, KTWO, KSM)
CALL DISPIO(13, 32)
WRITE(M, 100) IFLV, IFRM, ITMINT, ITMEND, IDAY, IPLN, NUMPTS
100 FORMAT(5X, 'IFLY=', I6, 5X, 'IFRM=', I6, 5X, 'FIRST TIME =', 2(I6, 1X), 4X,
1 'LAST TIME=', 2(I6, 1X), '/5X,' 'IDAY=', I6, ' IPLN=', I6, ' NUMPTS=', I6, /
2 //13X, 'X' Y I/N &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, K K4
344 CONTINUE
101 FORMAT (1X, 16, 13X, I6, 11X, I6, 4X, I6, 4X, I6, 4X, I6, 4X, I6, 4X, 16)
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

-212-
BLOCK DATA
COMMON XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
PKVEL(2), ELVANG(2), NVORTX, KFRM, RTIME
COMMON /ABTERM/ IABTERM
COMMON /ATLP/ IATLP, DMX
COMMON /BUFFER/ IBUF1(12), IBUF2(6), IBUF3(2)
COMMON /BUFFER/ IBFR(12)
COMMON /CNTL/ KPROC, TACOTL, MTIRE
COMMON /DSP/ NS1, NS2, NS3, NS4, LDT, KSM, KL, LLDT
COMMON /IADDRS/ IADDXY(26), IADPTS(2), IData(66), IADAT, BLANK,
IABLNK
COMMON /THDL1/ MAX1, MAX2, IEMFL
COMMON /IFLDV/ IFLDV
COMMON/NP/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRAD2, NRADI,
CONV, STIME, FRVOTL
COMMON /INIT/ IFLPT, PORTS(2), STARBS(2), VELMNF
COMMON/KSL/ KSTEL, KPLN
COMMON /LDDAT/ IFLV, FERM, ITMINT(2), ITMEND(2), IDAY, IPLN; NUMPTS,
I(1000), IY(1000), INTENS(1000), IVEL(1000)
COMMON /DWT/ IDWT(6)
COMMON /PLNAME/ IPLAT(60)
COMMON /ROUT/ IROUT, KGBF
COMMON /SETUP/ DATE, JDT(20, 6), JDLTP(3), IFLIND, IL1(60), TIM(2),
IFL6(3), ILDV, IAHC, IL2(90), KFBUF(6)
COMMON /TMPS/ KBT(6), KAUTO
COMMON /XORDT/ KX1, KX2, KX3, KY4
COMMON /RREAD/ RREAD
COMMON /MTWFLG/ MTWFLG
BYTE IDATA
INTEGER BLANK
CONVY = 2099 /1023 (640 METERS)
DATA CONVY/2 0518084/
DATA ITMINT, ITMEND/0, 0, 150/
DATA KSM, KL, K/035433, 034033/
DATA IREAD/0/
DATA MTWFLG/0/
DATA IFLDV/0/
DATA IDATA/ 66" " /
DATA BLANK/ " "
DATA ROUT/ BL " /
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, IRADI, FRVTL/0. 50, 2, 0. 50, 0. 50, 40,
1 0 50/
DATA ILDV, IAHC/0, 1/
DATA IFLIND/12/
DATA JDT(20*40, 20*50, 20*50, 20*20, 20*25, 20*50/
DATA NS1, NS2, NS3, NS4/2, 1, 3, 3/
DATA LLDT/12/
DATA KX1, KX2, KX3, KY4/560, 700, 770, 910/
DATA TACOTL/1 0E+06/
DATA MTIRE/0/

-213-
DATA ILU2 /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, 3, 0, 1, 1, 406, 614, 5, 0/
DATA IBASE, RTMX/566, 280, 0/
DATA KRBUF/1, -1, 0, 0, 180, 0/
1 "7", "C", "88", "0", "4", "E", "JT", "R", "73", "7", "VC", "-1",
2 "0", "2", "E", "PR", "B", "74", "7", "BA", "C1", "11", 1, "E",
3 "", "IL", "YU", "SN", "", "", "", "DC", "-1", "0", "YA", "C",
4 "40"/
END

ORIGINAL PAGE IS
OF POOR QUALITY

-214-
SUBROUTINE STRT (N)
COMM.OM XCF(2), YCF(2), INDEX(250), NCFRPT(2), NOISES(2),
PKVEL(2), ELYANG(2), NVORY, KFRM, RTIME
COMMON /ABTERM/ IABTERM
COMMON /BUFFER/ IBUF(20)
COMMON/CFLI/CFLI
COMMON /CNTL1/ KPROC, TACOTL, MTIRE
COMMON /DSCAL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /INFLI/ MAX1, MAX2, IEQFI
COMMON/INITL/ IFLFL, PORTS(2), STARTS(2), VELMNF
COMMON/TPNT/. PTTL, NOISE, VELTOL, VORTOL, IRADI, IRADI2, NRADI
1 CONXY, STIME, FRVTOL
COMM.OM /READ/ IREAD
COMMON/KSTFL/KSTFL, KPLN
COMMON/KTFILL/KTFILG
COMMON /LDVFAT/IFLY, IFRM, ITMIN(2), ITMEND(2), IDAY, IFLN, NUMPTS,
IY(1000), IY(1000), INTENS(1000), TVEL(1000),
COMMON /MTWFLG/MTWFLG
COMM.OM /ROUT/ ROUT, KGBF
COMMON /SETUP/ DATE, JOPT(20, 4), JPLTP(3), IFLN, IL1(10), TIM(2),
IPLFL(3), ILOV, ISHC, IL2(50), KRB(6)
COMMON /STST/ TOT(2), KVAN1, KVAN2
COMMON /TMPS/ KDP1(6), JAUTO
BYTE CHAR(4)
BYTE IDATE(4)
INTEGER CFLI
DIMENSION ERMSG(3)
DIMENSION IBUF(A)
DIMENSION ISTM(2)
DIMENSION N(A)
DIMENSION NNN(2)
DIMENSION TEXT(16), TEXT1(16)
EQUIVALENCE (CHAR(1), JFLB(1))
EQUIVALENCE (IDATE(1), DATE)
DATA DATS/'STRU'/'
DATA ERMSG/'OPT', 'TAPE', 'ERR'/'
DATA IBUF/1, 1, 0, 0, 44, 0/
DATA KSGFS /0/'
DATA TEXT/'FR O', 'ATA ', 'COR', 'PT', 'NOI', 'SE', 'ANGL', 'E P',
1 'K', 'E', 'TM', 'PO', 'RT', 'OS S', 'TARR', 'POS'/'
DATA TEXT1/ 'P', 'R', 'T', 'S', 'S', 'P', 'S', 'MN M', 'Y'
1, 'P', 'S', 'X', 'Y', 'Y', 'Y'/'

C STRT SUBROUTINE PROCESSING
C
C VORTEX - COMMON VARIABLES DEFINITION
C
C CFLI - CONTINUOUS MODE FLAG (0-OPERATOR CONTROLLED), 1-CONTINUOUS
C CHAR - BYTE ARRAY, EQUIVALENCE (CHAR, JFLB)
C DATS - MNEMONIC OF SUBROUTINE CODE
C IABTERM = FLAG TO INDICATE KEYBOARD REQUEST
C + (ABORT) [42]
C 1 (END FLV-RY+ [38]
C IEQFI = FLAG FOR END OF FLV-BY (0-NORMAL FRAME, 1-ENDED-OF-FLY-BY)
C IFLY = FLV-BY NUMBER
C IRADI = CORRELATION CIRCLE RADIUS
C IRADI2 = SQUARE OR CORRELATION CIRCLE RADIUS

-215-
TERMINATE REQUEST MODE

ROUT = DATA

CONTINUE

IF (CELL EO) CALL INIT

IF CONTINUOUS MODE ENABLE START PLAY-BY-PLAY INTERRUPTS

IF (N1) EO 16) CELL=0
IF (N1) EO 10) CELL=0

CONTINUE

IF (N2) EO 8) CELL=1
IF (N2) EO 7) CELL=0
IF (N1) EO 6) GO TO 2
IF (N1) EO 3) CELL=1

CONTINUE

IF (CELL Z2) Z2.11

SET CONTINUOUS OR OPERATOR CONTROLLED START PLAY-BY-PLAY

READ=0

CONTINUE

3002 IF(MITLE)=0 M1NUTE 10, 300.10

3000 IF(MITLE)=0 M1NUTE 10, 300.10

M1NUTE=0

IABITM=0

TNAME

TNAME - TIME LIMIT FOR PLAY-BY-PLAY (SECONDS)
TIME - START PLAY-BY TIME
STAFES - ARRAY
PLACE - TIME SINCE START OF PLAY-BY
MEMO MNU OF CURRENT SUBROUTINE
PORT - ARRAY
PORTS - ARRAY
SNSA - SCREEN ASSIGNMENT FOR SCATTER PLATS
NSMB - SCREEN ASSIGNMENT FOR TIME BASED PLATS
NSTA - SCREEN ASSIGNMENT FOR X-Y PLATS
NSU - NEGATIVE OF CORRELATION CIRCLE RADILES
N(2) - OPTION NUMBER
(1) - DISPLAY NUMBER
(0) - ARRAY: TABLED OPTION SELECTION

M1NIF - MAXIMUM FOR CURRENT FRAME PLAY-BY-PLAY
L101 - INSERTION OF Y-UNIT POSITION OF TABULAR DATA
L100 - INSERTION OF X-UNIT POSITION OF TABULAR DATA
L(1) - 2-POST ANALYSIS
L(0) - 2-POST ANALYSIS
L000 - INSERT OF INVERSE NUMBER OF DATA PLUS EXCEED 992
1511M - ARRAY START PLAY-BY TIME
CALL DISP (13, 1)
CALL DISP (13, 1)
C INITIALIZE PORTS AND STARBBS FOR 1ST FRAME
C INITIALIZE PORTS AND STARBBS FOR 1ST FRAME
PORTS(1) = 0
PORTS(2) = 800
STARBBS(1) = 800
STARBBS(2) = 0
VELMN = 0.0
TFLST = 0
IRAD = IRAD * IRAD
NRAD = -IRAD
DO 4317 I = 1, 8
4317 TOT(I) = 0
    KVAN = 0
    KVAN2 = 0
C C BRING UP DISPLAY BACKGROUNDS
C IF (NS1 EQ 3) GO TO 75
C C BRING UP X-Y PLOT BACKGROUND
C SELECT SCREEN NS1
C CALL DISP (11, NS1)
C CALL DISP (2, 7, 8)
75 CONTINUE
C IF (NS2 EQ 3) GO TO 80
C C BRING UP TIME PLOT BACKGROUND
C SELECT SCREEN NS2
C CALL DISP (11, NS2)
C CALL DISP (2, 11, 12)
80 CONTINUE
C CONTINUE
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
C CALL DISP (11, NS3)
C CALL DISP (2, 19, 0)
LOT = 712
C PRINT TABULAR HEADING
IBUF(1) = NS3
IBUF(4) = LOT
CALL SETAD (IBUF(6), TEXT)
IBUF(3) = 512
CALL DISP (4, 12, IBUF)
C C
IBUF(5) = 64
LOT = LOT - LLDT
IBUF(4) = LOT
CALL SETAD (IBUF(6), TEXT1)
CALL DISP (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 (CFLT) 891,891,857
857 CONTINUE
C IF CONTINUOUS MODE LOOK FOR START FLYBY INTERRUPT
C STOP -- WAIT FOR START FLY-BY INTERRUPT
C IF TABTRM=4 ABORT FLY-BY
89 IF (TABTRM EQ. 42) GO TO 91
IF (KSTFL .NE. 1) GO TO 89
C DECODE BCD-CODED PLANE TYPE
11=KPLN/16
IF (KPLN-16*I1 .GT. 9) KPLN=16*I1+9
N(2)=KPLN-6*I1+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 KTW0=2
CALL DISPIO(14,KTWO,KLG)
C BRING UP HEADER ON SCREENS 1 AND 2
C SELECT SCREEN J
CALL DISPIO (11,3)
C PRINT HEADER-FILL IN
893 CALL DISPIO(1,3A,TRUE1)
C SELECT SMALL CHARACTER SIZE
KTWO0=2
CALL DISPIO(14,KTWO,KSM)
C IF OTHER THAN FIRST PAGE OF TABULAR DATA SKIP INITIALIZATION
IF (LFRAME .GT. 55) GO TO 90
KTFGLG=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
CFLI=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 DISP0 (11,1)
BRING UP INITIAL DISPLAY
CALL DISP0 (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
IF (CHR$(K) = 48) 

IF (CHR$(K) < 57) GO TO 97 

$8 CONTINUE 

INCREMEMT RELEV AN BY 10 BY 1 (ASCII)

IF (KPOPO = 0) CALL DISABLE

IF REAL-TIME M O D E DISABLE S DATA & END-OF-FRAME INTERRUPTS

IF (MIPER. EQ 0) CALL PRINT (O.2, N8A. NCR)

IF TAPE WRITE OPTION SELECTED WRITE END ON TAPE

$95 INDUCT = 0

RESET END-OF-FRAME-BY-FLAG

IF (RTIME. LT TACOTT) GO TO 90

CHECK FOR FRAME-BY-EXCEEDING TIME LIMIT

1094 CONTINUE 

GO TO 81 

IF (COUNTER. NE I) CALL DISPLAY (1, 0, 3)

IF (COUNTER. LT LEBAN) GO TO 1094 

CHECK FOR MAXIMUM PAGE COUNT ON TA BULAR DATA

IF (K NE 0) CALL PANEL (3, 12, EBSC)

CHECK FOR TAPE ERROR ON TAPE WRITE

IF (MIPER. EQ 0) CALL WAIT O, O)

GO TO PREVIOUS I/O OPERATION ON MTO TO COMPLETE

GO CONTINUE 

IF (LACER = 0) GO TO 95 

CALL VREAD 

IF (XPOQCE. EQ 0) GO TO 60T 

$4 CONTINUE 

985 K R E M = 1 

SET FLAG TO INDICATE EXCESSIVE NUMBER OF DATA POINTS

GO TO 94 

CALL CENTROID 

CALL CENTROID TO COMPUTE AND OUTPUT VORTEX CENTROIDS

IF (LEBMB. NE 1) GO TO 93 

IF (XGPM = 0)

931 CONTINUE 

GO TO 94 

KNAME = 0 

IF (LEBMB. LE 0) CALL PRINT (O. I, IVAY. 409)

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
         IRAFRM=0
C    DELAY
C    HARD COPY 4014 IF REQUESTED
         IF (.NOT.AUTO 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 (CFLI) 22.99.22
         99 CONTINUE
C    BRING UP REAL-TIME OPTION SELECTION DISPLAY
C    BRING UP POST-ANALYSIS OPTION SELECTION DISPLAY
         CALL DISPIO (2,KT,0)
         CALL RTV (NNN)
950 RETURN
END
SUBROUTINE TERM (N)
COMMON /CNTRL/ KPROC, TACOLT, 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 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/ILV,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
COMMON /ABTERM/IABTRM
COMMON /ATLP/IRBASE,DIMX
COMMON /CTRL/KPROC,TACTRL,MTRF
COMMON /DPL/NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
COMMON /FDRV/FDRV
COMMON /SETUP/DATE,JDFT(20,6),JPLTP(3),TPLND,IL1(60),TIM(2),
1 JFLB(3),JLUV,IAHC,IL2(90),KREUF(6)
COMMON /XORDT/KX1,KX2,KX3,KX4
COMMON /TMPS/KDFT(K),JAUTO
COMMON /CFL/CRF
INTEGER CFL
BYTE CHAR(4),KDABT(4)
DIMENSION N(8)
DIMENSION KDATE(2)
DIMENSION NS(4)
DIMENSION NNN(4)
EQUIVALENCE (JFLB(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 (IL81,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)=N5
KDATE(2)=N(4)
IF (KPROC EQ. 1) GO TO 950
ISDAY=0
DO 33 I=1,3
33 ISDAY=ISDAY+10+KDABT(I)-48
2000 IF(ISDAY-IDAY) 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 FOREWORD SPACE A FILE
NBA=1
CALL SFUN(1,10,NBA,NFL)
IF(NFL) 975,2600,975
34 CONTINUE
TPLND=TPLN
CFLI=1
NNN(2)=1PLIND
NNN(4)=0
CALL PTYP (NNN)
CFL1=0
GO TO 950
36 CONTINUE
IF(LB(1)=N5
IF(LB(2)=N(4))
IF(LB(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, 46000, 975
4600 CONTINUE
CALL VRNAD
GO TO 4000
5000 CONTINUE
C NEED FOREWARD SPACE A FILE
NBA=1
CALL SFUN(1, 10, NBA, NFL)
IF(NFL) 975, 46000, 975
6000 CONTINUE
N2=1
GO TO 34
35 CONTINUE
C IBASE IS BIT POSITION OF BAR ORIGIN
C 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, 300, 300, 300, 300, 300, 300, 300)
1 300, 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 VRNAD
GO TO 6000
44 CONTINUE
IF (N5 LT. IBASE) N5=IBASE
TACOTL = ((N5-IBASE)*300.0)/DIMX
IL1=IBASE
IL3=IBASE+(TACOTL*DIMX/300)
IL6=IBASE
IL8=IL3
GO TO 950
45 CONTINUE
IF (KPROC NE 2) GO TO 47
CFL1=1
GO TO 950
47 CONTINUE
TACOTL=1 0E+6
IL01=910
IL3=980
IL6=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
MTIFR=0
GO TO 435

275 CONTINUE
MTIFR=1
GO TO 460

300 CONTINUE
K1=1
IF ((N2 .EQ. 10) OR (N2 .EQ. 12) OR (N2 .EQ. 14) OR (N2 .EQ. 16)):
1 K1=2
ASSIGN 300 TO KST
DO 400 I=1,4
IF (N3(I) .NE. K1) GO TO 400
N3(I)=8
K1=KY1
K2=KY1
KT=10*I+31
GO TO 700

400 CONTINUE
ASSIGN 950 TO KST
IF (N2-2=NT) 425, 450, 450

425 N3(NT-4)=1

435 K1=KY1
K2=KY2
GO TO 475

450 N3(NT-4)=2

460 K1=KY2
K2=KY4

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
JAUTO=2
IL81=630
IL83=700
IL86=630
IL88=700
GO TO 950

ORIGINAL PAGE IS OF POOR QUALITY
925 CONTINUE
J AUTO = 1
IL81 = 840
IL83 = 910
IL86 = 840
IL88 = 910
950 CONTINUE
CALL DISPI0 (9, 12, KREUF)
RETURN
975 TABTRM = 0
CALL DISPI0 (11, 1)
CALL DISPI0 (2, 1, 0)
RETURN
END
SUBROUTINE CENTRD

THIS SUBROUTINE PERFORMS A SEARCH TO LOCATE THE POINTS WHICH
define a vortex and calculates the centroid of these points.
CONXY = SCALE FACTOR TO CONVERT IX AND IV TO FEET
ELVANG(1) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
ELVANG(2) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
INTENS = INTENSITY OF DATA POINT
IRADI2 = SQUARE OF RADIUS FOR CORRELATION CIRCLE
IVEL = VELOCITY OF DATA POINT
IX = X COORDINATE OF DATA POINT
ITY = Y COORDINATE OF DATA POINT
KNT = NUMBER OF POINTS WHICH LIE IN CORRELATION CIRCLE
LSEARC = MAXIMUM NUMBER OF SEARCHES ALLOWED FOR VORTEX LOCATION
MAX1 = INDEX OF MAXIMUM VELOCITY IN A FRAME
MAX2 = INDEX OF SECOND HIGHEST VELOCITY IN A FRAME
MINCNT = NUMBER OF POINTS IN CORRELATION CIRCLE POSSESSING A PEAK VELOCITY
EQUAL TO OR GREATER THAN MINVEL
MINVEL = ITOL2*PEAK VELOCITY
NOISES = NOISE SPIKES THAT WERE DISCARDED IN PROCESSING DATA FOR A
SPECIFIC VORTEX -- ARRAY OF 2
NNS = NUMBER OF SEARCHES THAT HAVE BEEN MADE IN AN ATTEMPT TO
LOCATE A VORTEX
NP = NUMBER OF DATA POINTS IN A FRAME
NVORTEX = NUMBER OF VORTICES THAT HAVE BEEN LOCATED IE. NVORTEX + 1
IS THE VORTEX NO. THAT IS CURRENTLY BEING PROCESSED
PTTOL = PER CENT OF POINTS IN A CORRELATION AREA THAT MUST HAVE A
VELOCITY GT VELTOL*PEAK VELOCITY
SUM1 = SUM OF PRODUCTS OF INTENSITY AND VELOCITY OF ALL POINTS
IN THE CORRELATION CIRCLE
SUM2 = SUM OF THE PRODUCTS OF INTENSITY, VELOCITY, AND X COORDINATE
OF ALL POINTS IN THE CORRELATION CIRCLE
SUM3 = SUM OF THE PRODUCTS OF INTENSITY, VELOCITY, AND Y COORDINATE
OF ALL POINTS IN THE CORRELATION CIRCLE
VELMN2 = MINIMUM ACCEPTABLE PEAK VELOCITY
VELTOL = VELOCITY TOLERANCE -- PER CENT OF PEAK VELOCITY THAT SOME
PER CENT OF POINTS IN A CORRELATION AREA MUST POSSESS
VORTOL = PER CENT OF PEAK VELOCITY FROM 1ST VORTEX THAT PEAK VELOCITY FROM
2ND VORTEX MUST MEET
XCG = X CENTROID OF VORTEX
YCG = Y CENTROID OF VORTEX
COMMON XCG(2), YCG(2), INDEX(250), NOISES(2), MINCNT(2), MINVEL(2), ELANG1, ELANG2, NVORTEX, KFRM, RTIME
COMMON /CNTRL/ KPROC, TACGTL, MTFK
COMMON /DISPL/ NS1, NS2, NS3, NS4, LDT, KSM, KL1, LL1, LLD
COMMON /IDUMMY/NBA, NFL
COMMON /IHDL1/ MAX1, MAX2, IEQF1
COMMON /INTL/ ITFL, PORTS(2), STARBS(2), VELMN1, VELMN2
COMMON /INPT/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRADI2, NRADI,
1 CONXY, STIME, FRVTOL
COMMON /KTFLG/KTFLG
COMMON /LDOVAT/ IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IX(1000), IV(1000), INTENS(1000), IVEL(1000)
COMMON /ROUTE/ ROUT, KGBF
COMMON /SETUP/ DATE, INPT(20, 6), JPLTR(3), IPLIND, IL1(50), TIM(2),
1 TFLN(3), TLDV, TAHO, IL2(90), KRBUF(6)

-227-
COMMON /ABTERM/ IABTBM
COMMON /STAT/ TOTX1, TOTX2, TOTY1, TOTY2, SSOX1, SSOX2, SSOY1, SSOY2.
1 KVAN1, KVAN2
INTEGER KBUFOT(4)
INTEGER MSTAT(16)
REAL STT(5)
EQUIVALENCE (XCG1, XCG(1)), (YCG1, YCG(1))
EQUIVALENCE (MSTAT, STTT)
EQUIVALENCE (MSTAT4, MSTAT(4)), (MSTAT7, MSTAT(7)), (MSTA13, MSTAT(13))
EQUIVALENCE (MSTA15, MSTAT(15))
EQUIVALENCE (KBUF6, KBUFOT(6))
EQUIVALENCE (WTEST, JPLTP)
DATA SNAME/ 'CENT'/
DATA KBUFOT /4*1.32, 0/.
DATA STTT / AVG: '=( , , , , , ) &', 'TD=(' , , , , , , , /.
DATA WHEEL / 'WHEE'/
DATA LNOISE/5/
ROUT= SNAME
TMINT= 32768 0*ITMINT(1) + ITMINT(2)
IF (KTFLG) 428, 428, 429
428 CONTINUE
STIME=TMINT
KTFLG=1
429 CONTINUE
NSPIKE=0
C INITIALIZE NOISE(1) FOR COMPUTATION OF NOISE FOR EACH VORTEX
NOISES(1)=0
NVORTX=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
NCOORPT(1)=NUMPTS
CALL GETVEL(VEL(1), VELMAX)
GO TO 5020
5010 CONTINUE
GO TO (1,2000), KPROC
1 CALL GETVEL(VEL(MAX1), VELMAX)
2 CONTINUE
IF (VELMAX-VELMIN) 9, 10, 10
9 CONTINUE
NCOORPT(1) = 0
5020 CONTINUE
NCOORPT(2) = 0
PKVEL(1) = VELMAX
GO TO 2500
10 CONTINUE
C INITIALIZE SUM TERMS AND COUNTERS
SUM1 = 0
SUM2 = 0
1. Calculate centroid of all points in correlation circle.

2. Continue if minimum-velocity point is not found, search another.

3. Call subroutine (lens(l), 2, idum).

4. Continue if minimum = index + 1.

5. Continue if velocity = minimum.

6. Continue if (velocity - minimum > 0).

7. Call subroutine (lens(l), 1, velocity).

8. If (r1 - rand1 > 0.4, 0.1).

9. If (r1 = rand1, 100).

10. If (r1 > rand1, 100).

11. If (r1 < rand1, 0.8).

12. Do 100 = l, numpts.

13. Continue if (l > lens(max)).

14. Continue if (l < lens(min)).

15. Compute minimum velocity for correlation circle.

16. Determine which points lie in correlation circle and which do not.

17. Compute sum terms.

18. If not enough points have the minimum velocity, try another.
YCG(NVORTX) = CONXY*DUM2
NOISES(NVORTX) = NSPIKE - NOISES(1)
NCRPT(NVORTX) = KOUNT
VELMNF = FRVTOL*VELMAX
PKVEL(NVORTX) = VELMAX
IF(NVORTX-2)425,2500,2500

425 CONTINUE
C COMPUTE TIME
TMEND= 32768 0*ITMEND(1) + ITMEND(2)
ANGLE = ATAN2(YCG(1),XCG(11)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(FNLANG2-ELANG1) 430, 460, 445
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
MINPTS=VORTOL*KOUNT
530 CONTINUE
NBA=NUMPTS-KOUNT
IF(NBA-MINPTS)2280,540,540
540 CONTINUE
   DO 550 L=1,KOUNT
      J=INDEX(L)
      IVEL(J)=- IVEL(J)
550 CONTINUE
GO TO 2000
C TRY ANOTHER PT WITH A MAX VELOCITY
1000 CONTINUE
NSPIKE=NSPIKE + 1
IF(NUMPTS-2)2280,2280,1001
1001 CONTINUE
IF(LNOISE-NSPIKE)2300,2300,1010
PROCEDURE (NFL) = VELMAX
COUNT (NFL) = KOUNT

NOTE (NFL) = NSPIKE - NOTES (1)

XCO (NFL) = 0
XCO(NFL) = 0
NFL = NFL + 1

2380 CONTINUE
COUNT = 0
VELMAX = 0
CONTINUE

2280 IF (ONRTY) 2, 10
CONTINUE

2248 GO TO 2000
INVL (MAX) = INVL (MAX1)

2280 CONTINUE
IF (SORT (MIN) - RANDMN (2230, 2230, 2248)
RANDMN = NOTES (1)
BMIN = BMIN + 1
DINV = DINV + 1

2280 CONTINUE
IF (ONRTY) 1, 2248, 2230, 2220

2220 CONTINUE
CALL GETVL (LEVEL (MAX1), VELMAX)

2200 60 TO 2200
MAVL = 0
WMAX = MAX1
MAX2 = MAX1
CONTINUE

2140 IF (LEVEL (MAX) - LEVEL (L) =) 140, 2150
CONTINUE

2130 IF (LEVEL (MAX2) - LEVEL (L)) = 130, 2200
DO 2200 L = 3, NUMPR
MAX2 = 1
MAX2 = 2
MAX2 = MAX1
CONTINUE

2050 IF (LEVEL (MAX1) - LEVEL (MAX2)) = 2050, 2100, 2100
CONTINUE
E

FIND NEW MAXIMA FOR VORTEX SEARCH

1060 0 CONTINUE
IF (ONRTY) 2, 10
CONTINUE

1050 IF (VELMAX (2280, 2280, 1060)
CALL GETVL (LEVEL (MAX1), VELMAX)
CONTINUE

1050 IF (MAX2 < 2000, 2000, 1050
MAX2 = 0
MAX2 = MAX2
LEVEL (MAX1) = 0
CONTINUE
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=TOTY2 + YCG1
     TOTY2=TOTY2+YCG1
     SSQX2=SSQX2+XCG1*YCG1
     SSQY2=SSQY2+YCG1*YCG1
     GO TO 3600
3150 IF (KVAN1=30) 3155, 3250, 3600
3155 CONTINUE
     KVAN1=KVAN1 + 1
     TOTY1 = TOTY1 + YCG1
     TOTY1=TOTY1+YCG1
     SSQX1=SSQX1+XCG1*YCG1
     SSQY1=SSQY1+YCG1*YCG1
     GO TO 3600
3225 AVYCG=TOTY2
     AVYCG=TOTY2
     STDX=SSQX2
     STDY=SSQY2
     KBUFOT(3)=517
     GO TO 3275
3250 AVYCG=TOTY1
     AVYCG=TOTY1
     STDX=SSQX1
     STDY=SSQY1
     KBUFOT(3)=5
3275 CALL SETAD (KBUF6A, MSTAT)
     KBUFOT(1)=N83
     KBUFOT(4)=LDT-LLDT*IFRM
     LAVXCG=AVXCG/30.0
     LAVYCG=AVYCG/30.0
     KSSTDX=SORT((30.0*STDX-AVXCG*AVXCG)/870.0)
     KSSTDY=SORT((30.0*STDY-AVYCG*AVYCG)/870.0)
     CALL DECD (LAXXCG, 5, MSTAT4)
     CALL DECD (LAVXCG, 5, MSTAT7)
     CALL DECD (KSTDX, 5, MSTA13)
     CALL DECD (KSTDY, 5, MSTA15)
     CALL DISP10 (4, 12, KBUFOT)
     GO TO 3700
3600 CONTINUE
     CALL DISPLA
3700 RETURN
END
SUBROUTINE RTV (N)
COMMON /ATLP/ IBASE, DIMX
COMMON /BUFFER/ IBFR(12)
COMMON /CTRL/ KPROC, TACOTL, MTIRF
COMMON /DPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
COMMON /DATE/ JDATE (20, 6), JPLTP(3), IPLIND, IL1(60), TIM(2),
I IFPB(3), ILOV, IAHG, IL2(90), KRBUF(6)
COMMON /READ/ IREAD
COMMON /IDUMMY/NBA, NFL
COMMON /LEVEL/ IFLY, IFRM, ITMIN(2), ITMEND(2), IDAY, IPLN, NUMPTS,
I TX(100), IY(100), INTENS(100), IVEL(100)
COMMON /CFII/ CFII
INTEGER CFII
DIMENSION N(4)
DIMENSION NN(4)
EQUIVALENCE (IBFR4, IBFR(4)), (IBFR10, IBFR(10))
EQUIVALENCE (IL01, IL2(1)), (IL3, IL2(3)), (IL6, IL2(6)), (IL9, IL2(9))
EQUIVALENCE (IL31, IL2(31)), (IL33, IL2(33)), (IL36, IL2(36)),
I (IL38, IL2(38)))
EQUIVALENCE (IL42, IL2(42)), (IL44, IL2(44)), (IL47, IL2(47)),
1 (IL49, IL2(49)), (IL52, IL2(52)), (IL54, IL2(54)), (IL57, IL2(57)),
2 (IL59, IL2(59)), (IL62, IL2(62)), (IL64, IL2(64)), (IL67, IL2(67)),
3 (IL69, IL2(69)), (IL72, IL2(72)), (IL74, IL2(74)), (IL77, IL2(77)),
4 (IL79, IL2(77))
DATA IND */0/
DATA KM1/-1/
CALL DISP(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 )

ORIGINAL PAGE IS OF POOR QUALITY
IL6=ibase
IL8=IL3
200 CONTINUE
KT=3
IF (MTTRF EQ. 1) GO TO 300
IL31=560
IL33=700
IL36=560
IL38=700
GO TO 500
300 CONTINUE
IL31=770
IL33=910
IL36=770
IL38=910
GO TO 500
400 CONTINUE
IFRF4=613
IFRF10=571
IF (IND EQ. 1) GO TO 410
IND=1
CALL OPEN (1,1)
NBA=1
CALL SEUN (1,4,NBA,NFL)
410 CONTINUE
MTTRF=1
KPROC=2
IF (IREAD) 425, 420, 425
420 CONTINUE
IFRED=1
CALL VREAD
IPLIND=IPLN
CFL1=1
NNN(2)=IPLIND
NNN(4)=0
CALL PTYP (NNN)
CFL1=0
425 CONTINUE
IL01=400
IL03=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
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, KBUFF)
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 LOCVAN(1) = LOCATION OF VAN1 WRT CENTER OF RUNWAY
C LOCVAN(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 YRATXY = RATIO FOR Y DIT POS FOR XY PLOTS
C YRATXT = RATIO FOR X DIT POS FOR X-TIME PLOTS
C TRATYT = RATIO FOR TIME DIT POS FOR X-TIME PLOTS
C YRATYT = RATIO FOR Y DIT POS FOR Y-TIME PLOTS
C TRATYR = RATIO FOR TIME DIT POS FOR Y-TIME PLOTS
C COMMON/INITL/ ITPLT, PORTS(2), STARB(2), VELMN
C COMMON /DISPL/ NS1, NS2, NS3, N1, NS4, LDT, KSM, K2, LDIT
C COMMON /DUDAT/ IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
C 1 IX(1000), IX(1000), INTENS(1000), IVEL(1000)
C COMMON/ IADORS/ IADDY(26), IADPTS(2), IDATA(66), IADDAT, BLANK,
C 1 IAR W
C COMMON XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
C 1 PKVEL(2), ELUANG(2), NUORTX, KFRM, RTIME
C COMMON /ROUT/ ROUT, KGBF
C DIMENSION IDITTB(2), IVDIT(2), IDITY(2), IDITTY(2), LOCVAN(2)
C DIMENSION IBBUFF(24), PORTS(2), STARB(2)
C INTEGER BLANK
C BYTE IDATA, DECIML, KJWK, NSEG
C DATA DECIML/ 5, 6, 7 /
C DATA NEG/ -5, -6, -7 /
C DATA LOCVAN/ 400, 400 /
C DATA IBBUFF(2), IBBUFF(3) /1, 0 /
C DATA DSMF/ "DATA /
C DATA SNAME/ "DISP /
C REMOVE A/N TEST ****
C DATA IRUNXY/ 518 /
C DATA XRATXY/ 7 /
C DATA YRATXY/ 88 /
C DATA IVDIT/ 459, 85 /
C DATA IDITTB/ 0, 512 /
C DATA IDITY/ 459, 63 /
C DATA IDITTY/ 712, 316 /
C DATA IRUNXT/ 784 /
C DATA IDITTM/ 70 /
C DATA XRATYR/ 42 /
C DATA TRATYT/ 3, 125 /
C DATA YRATYT/ 88 /
C DATA TRATYT/ 4, 2 /
C REMOVE ********************+

ROUT= 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
LVEL=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(IVAN 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)

-237-
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
IVC1=YCG(1)
IVC2=YCG(2)
LEVEL=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
IDUM= 1*MINANG
IDATA(25)= IDUM + 48
IDATA(26)= MINANG - 10*IDUM + 48
IDUM= 1*MAXANG
IDATA(28)=IDUM + 48
IDATA(29)= MAXANG - 10*IDUM + 48
IDUM= .01*LVEL
IDATA(32)=IDUM + 48
IDUM1= LVEL - 100*IDUM
IDUM= 1*IDUM1
IDATA(33)=IDUM + 48
IDATA(34)= IDUM1 - 10*IDUM + 48
IDUM= .01*IRVEL
IDATA(36)=IDUM + 48
IDUM1=IRVEL - 100*IDUM
IDUM= 1*IDUM1
IDATA(37)=IDUM + 48
IDATA(38)= IDUM1 - 10*IDUM + 48
KTIME= RTIME
IDUM= 01*KTIME
IDATA(40)= IDUM + 48
IDUM1=KTIME-100*IDUM
IDUM= 1*IDUM1
IDATA(41)=IDUM + 48
IDATA(42)= IDUM1 - 10*IDUM + 48
IDATA(43)=DECIMAL
IDUM= RTIME
IDUM= RTIME-10 0 - 10*IDUM
IDATA(44)=IDUM + 48
IF (IXC1 .GE. 0) GO TO 1200.
KDUM= IABS(IXC1)
IDATA(46)=NEG
GO TO 1250
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= IABS(IXC2)
IDATA(56)=NEG
GO TO 1270
1260 CONTINUE
KDUM= IXC2
IDATA(56)=BLANK
1270 CONTINUE
IDUM= 01*KDUM

-239-
IDATA(57) = IDUM + 48
IDUM1 = KDUM - 100*IDUM.
IDUM = 1*IDUM1
IDATA(58) = IDUM + 48
IDATA(59) = IDUM1 - 10*IDUM + 48
IDUM = 01*TYC2
IDATA(61) = IDUM + 48
IDUM1 = TYC2 - 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) = 33
IBUFF(3) = IDITTR(IVAN)
IBUFF(4) = LDT-LLDT*KFRM
IF(KFRM .GT. 55) IBUFF(4) = LDT-LLDT*(KFRM-55)
IF(KFRM .GT. 110) IBUFF(4) = LDT-LLDT*(KFRM-55)
TBBUFF(5)=63
TBBUFF(6)=1
CALL DISPLO(4,12,1BUFF)
1500 CONTINUE
IF(NUORTX.EQ.0) GO TO 4000
1F(NS1.EQ.3) GO TO 2000
C  YY PLOTS WERE CHOSEN AS A DISPLAY OPTION
1BBUFF(1)=NS1
TBBUFF(5)=1
1BBUFF(7)=NS1
TBBUFF(8)=TBBUFF(2)
1BBUFF(11)=1
1F(YCG(1).EQ.0.0) GO TO 1700
C  THERE IS A PORT VORTEX TO BE PLOTTED
1BBUFF(3)= TRUNYX + IXC1*YAXTV
TBBUFF(4)= IYDIT(IVAN) + YCG(1)*YAXTV
TBBUFF(6)= IADDXY(ITPLT)
GO TO 1800
1700 CONTINUE
C  THERE IS NO PORT VORTEX
1BBUFF(6)=TABLNK
1800 CONTINUE
1F(YCG(2).EQ.0.0) GO TO 1900
C  THERE IS A STARBOARD VORTEX
1BBUFF(9)=TRUNYX + IXC2*YAXTV
1BBUFF(10)=IYDIT(IVAN) + YCG(2)*YAXTV
1BBUFF(12)=IADDXY(ITPLT)
GO TO 1950
1900 CONTINUE
C  THERE IS NO STARBOARD VORTEX
1BBUFF(12)=TABLNK
1950 CONTINUE
CALL DISPLO(4,24,1BUFF)
2000 CONTINUE
1F(NS2.EQ.3) GO TO 4000
C  X-TIME AND Y-TIME PLOTS WERE CHOSEN AS A DISPLAY OPTION
1F(RTIME.GE.81.) GO TO 4000
1BBUFF(1)= NS2
1BBUFF(7)= NS2
1BBUFF(13)=NS2
1BBUFF(19)=NS2
1BBUFF(5) = 1
1BBUFF(11) = 1
1BBUFF(17) = 1
1BBUFF(23) = 1
1BBUFF(4) = IDITTY(IVAN) - RTIME*TRATXT
1BBUFF(16) = TBBUFF(4)
1BBUFF(9) = IDITTM + RTIME*TRATVT
1BBUFF(21) = TBBUFF(9)
1F(YCG(1).EQ.0.0) GO TO 2500
C  THERE IS A PORT VORTEX
1BBUFF(3)= TRUNYX + IXC1*XAXTV
1BBUFF(6)= IADPTS(1)
1BBUFF(10)=IDITTV(IVAN) + YCG(1)*XAXTVT
1BBUFF(12)=TBBUFF(6)
GO TO 3000

-241-
2500 CONTINUE
C THERE WAS NO PORT VORTEX
IBUFF(6)=IABLNNK
IBUFF(12)=IABLNNK
3000 CONTINUE
IBUFF(14)=IBUFF(2)
IBUFF(8)=IBUFF(2)
IBUFF(20)=IBUFF(2)
IF(XCG(2).EQ.0.0) GO TO 3500
C THERE IS A STARBOARD VORTEX
IBUFF(15)= IRUNXT + IXCZ*XMRXT
IBUFF(18)= IADPTS(2)
IBUFF(22)= IDITY(IVAN) + YCG(2)*YMRATY
IBUFF(24)= IADPTS(2)
IF(XCG(1).NE.0.)GO TO 3600
IBUFF(18)=IADDXY(19)
IBUFF(24)=IADDXY(19)
3500 CONTINUE
C THERE IS NO STARBOARD VORTEX
IBUFF(6)=IADDXY(19)
IBUFF(12)=IADDXY(19)
IBUFF(18)=IABLNNK
IBUFF(24)=IABLNNK
3600 CONTINUE
CALL DISPIO(4,48,IBUFF)
4000 CONTINUE
IF(NVORTX.LT.2)RETURN
PORTS(IVAN)=XCG(1)
STARBS(IVAN)=XCG(2)
RETURN
END
SUBROUTINE GETVEL(IV, VEL)
IF(IV LE 0) GO TO 1000
CALL SUBRT(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 /BUFP/ IBUF1(12), IBUF2(12), IBUFF(2)
COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LDLT
COMMON /IFLDV/ IFLDV
COMMON /INPT/ PTTOL, NOISE, VELTOL, VORTOL, IRADI, IRADI2, NRADI,
1 CONXY, STIME, FRVTOL
COMMON /LDVDAT/ IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
1 IX(1000), IV(1000), INTENS(1000), IVEL(1000)
COMMON /OUT/ IOWT(6)
COMMON /ROUT/ ROUT, KGBF
COMMON /SETUP/ DATE, JDFT(20, 4), JPLTP(3), IPLIND, IL1(60), TIM(2),
1 IFLB(3), ILDU, IAIAC, IL2(90), KRFBUF(6)
COMMON /TPS/ KDFTT(6), JAUTO
BYTE IBUF2
DIMENSION N(8)
EQUIVALENCE (IBUF01, IBUF2(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
JDFT(IPLIND, I)=KDFTT(I)
525 CONTINUE
GO TO 950
900 CONTINUE
KI=IRADI*CONXY
CALL DEC0D (KI, 2, IBUF01)
KI=VELTOL*100.0
CALL DEC0D (KI, 2, IBUF3)
KI=PTTOL*100.0
CALL DEC0D (KI, 2, IBUF5)
CALL DEC0D (NOISE, 2, IBUF7)
KI=VORTOL*100.0
CALL DEC0D (KI, 2, IBUF9)
KI=FRVTOL*100.0
CALL DEC0D (KI, 2, IBUF11)
CALL DISP60(9, 12, IOWT)
950 CONTINUE
RETURN
END

-244-
SUBROUTINE PTVP(N)
COMMON /BUFFER/ TRUE(12), TRUE2(6), TRUE3(3)
COMMON /CELL/ CELI
COMMON /CTRL/ KPROC, TACOLT, MTIF
COMMON /DELTP/ KDEL
COMMON /TNST/ TST, NOISE, VELTOL, VORTOL, TRAIT, TRAIT2, NTRAIT,
1 COMMON, STIME, FRU10
COMMON /IDUBAT/ IFLV, IERM, ITMINT(2), ITMENR(2), ITDAY, ITLN, NUMETS,
1 TXY(1000), TV(1000), INTENS(1000), IVFL(1000)
COMMON /IOUT/ IOUT(4)
COMMON /PNAME/ IDAT(40)
COMMON /ROU1/ ROUT, KRF
COMMON /SETUP/ NDATE, INDT(20, 6), JPLTP(3), JPLIND, TL1(40), TIME(2),
1 TFLR(3), ITN, IACF, ITZ(90), KREUF(6)
COMMON /TIMEPS/ KDEFT(4), JAHUTO
INTEGER CIEL1
INTEGER Y1, YM, YT, IV, RVY
DIMENSION M(2), N(6)
DIMENSION IDATE(2), EDIVAIENCE (IDATE(1), DATE)
EDIVAIENCE (JPLTP1, JPLTP(1)), (JPLTP2, JPLTP2)), (JPLTP3, JPLTP3))
C
Y1 IS LOW X-COORDINATE
YM IS HIGH X-COORDINATE
DATA X1, YM/A20, 980/
C
DATA KM1/-1/
DATA DATS/'PTVP'/
ROUT = DATS
N2=N(2)
IF (N2 EQ 22) GO TO 100
IF (N2 NE 21) GO TO 75
IF (KPROC EQ 1) GO TO 995
KDEL=1
GO TO 930
75 CONTINUE
IF (N(4) EQ 1) GO TO 200
JPLIND=N2
T=3*TPLIND-2
JPLTP1=IDAT(1)
JPLTP2=IDAT(1+1)
JPLTP3=IDAT(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, I0)
CALL RTV (M)
RETURN

ORIGINAL PAGE IS OF POOR QUALITY
200 CONTINUE
  IPLIND=18
  JPLTP1=N(5)
  JPLTP2=N(6)
  JPLTP3=N(7)
900 CONTINUE
  IF (KPROC=1) 930, 930, 910
910 CONTINUE
912 IF (IPLIND = IPLN) 915, 930, 915
915 CONTINUE
  NBA=1
  CALL SFUN(1, 10, NBA, NFL)
  CALL UREAD
  GO TO 912
930 CONTINUE
  DX = XM-X1
  DO 950 I=1, 6
  KDFTT(I) = JNDT(IPLIND, I)
  KY=X1+(DX*KDFTT(I))*O 01
  ILI(10*I-7)=KY
  ILI(10*I-2)=KY
950 CONTINUE
  TRADT = KDFTT(1)/CONXY
  VELTOL = KDFTT(2)*O 01
  PTTOL = KDFTT(3)*O 01
  NOISE = KDFTT(4)*10
  VORTOL = KDFTT(5)*O 01
  FRVTOL = KDFTT(6)*O 01
  J=5
  IF (KPROC EQ 2) J=15
  DO 960 I=1, 9
    TRUEI(I)=TFLB(I)
    TRUEI(I+3)=IDATE(I)
960 TRUEI(I+9)=JPLTP(I)
  CALL TIME (TRUE(6))
  IF (CFLI EQ 1) GO TO 995
  CALL DISPIO(2, J, 0)
  CALL DISPIO(1, 24, TRUE)
C
C   CALL DISPLAY CONTROLLER TO DISPLAY DEFAULT IN WRITE THROUGH
   CALL DISPIO (2, J+1, KM1)
   CALL DISPIO (9, 12, IOWT)
995 CONTINUE
  RETURN
END