

AD-A146 249

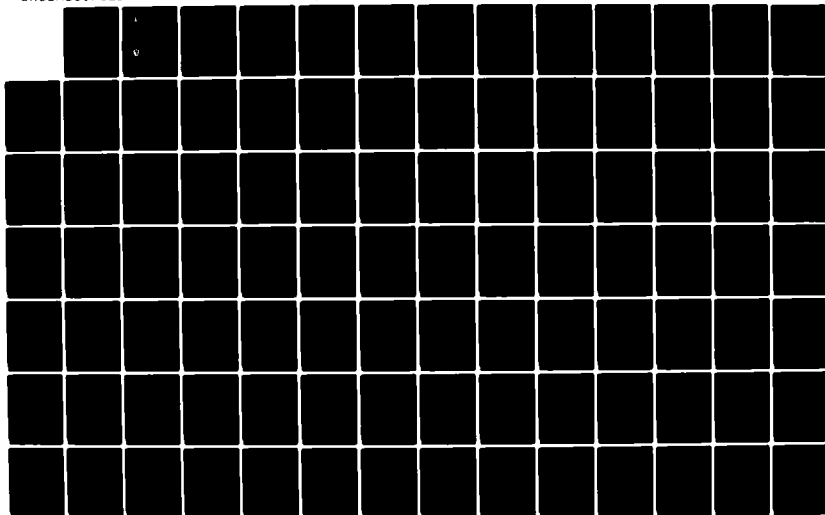
ON LINE DIGITIZER SOFTWARE(U) ARMY MISSILE COMMAND  
REDSTONE ARSENAL AL ADVANCED SENSORS DIRECTORATE  
S R SIMS JUN 84 DRSM1/RE-84-17-TR SBI-AD-E950 550

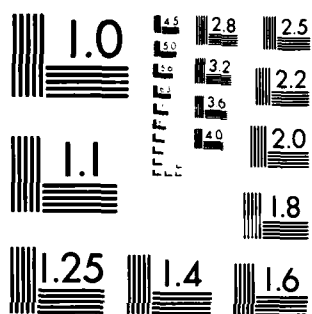
1/2

UNCLASSIFIED

F/G 9/2

NL





12

AD-A146 249



TECHNICAL REPORT RE-84-17

ON LINE DIGITIZER SOFTWARE

S. Richard F. Sims  
Advanced Sensors Directorate  
US Army Missile Laboratory

JUNE 1984



**U.S. ARMY MISSILE COMMAND**

*Redstone Arsenal, Alabama 35898*

Approved for public release; distribution unlimited.

DTIC FILE COPY

#### **DISPOSITION INSTRUCTIONS**

**DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT  
RETURN IT TO THE ORIGINATOR.**

#### **DISCLAIMER**

**THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN  
OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.**

#### **TRADE NAMES**

**USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES  
NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF  
THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.**

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                  | READ INSTRUCTIONS<br>BEFORE COMPLETING FORM                        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|--------------------------------------------------------------------|
| 1. REPORT NUMBER<br>RE-84-17                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 2. GOVT ACCESSION NO.<br>A57 005 | 3. RECIPIENT'S CATALOG NUMBER                                      |
| 4. TITLE (and Subtitle)<br><br>On Line Digitizer Software                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                  | 5. TYPE OF REPORT & PERIOD COVERED<br><br>Interim Technical Report |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                  | 6. PERFORMING ORG. REPORT NUMBER                                   |
| 7. AUTHOR(s)<br><br>S. Richard F. Sims                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                  | 8. CONTRACT OR GRANT NUMBER(s)                                     |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Commander, US Army Missile Command<br>ATTN: DRSMI-RES<br>Redstone Arsenal, AL 35898                                                                                                                                                                                                                                                                                                                                                                                                                        |                                  | 10. PROGRAM ELEMENT, PROJECT, TASK<br>AREA & WORK UNIT NUMBERS     |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Commander, US Army Missile Command<br>ATTN: DRSMI-RPT<br>Redstone Arsenal, AL 35898                                                                                                                                                                                                                                                                                                                                                                                                                            |                                  | 12. REPORT DATE<br>June 1984                                       |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                  | 13. NUMBER OF PAGES<br>165                                         |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                  | 15. SECURITY CLASS. (of this report)<br><br>UNCLASSIFIED           |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                  | 15a. DECLASSIFICATION/DOWNGRADING<br>SCHEDULE                      |
| 16. DISTRIBUTION STATEMENT (of this Report)<br><br>Approved for public release; distribution unlimited                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                  |                                                                    |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                  |                                                                    |
| 18. SUPPLEMENTARY NOTES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                  |                                                                    |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number)<br>Video digitizing and recording<br>Digital video recording and playback                                                                                                                                                                                                                                                                                                                                                                                              |                                  |                                                                    |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number)<br>This report describes the software developed to date for the On Line digitizer system. The software provides means to transfer digital imagery and auxiliary data to a host computer hard disk, to test system components and display the imagery and auxiliary data in various formats. The system digitizes analog video (RS-170 or RS-343) at a 10 MHz rate to eight bits and records 16 auxiliary analog channels for approximately 15 minutes on 28 track tape. |                                  |                                                                    |

DD FORM 1 JAN 73 1473

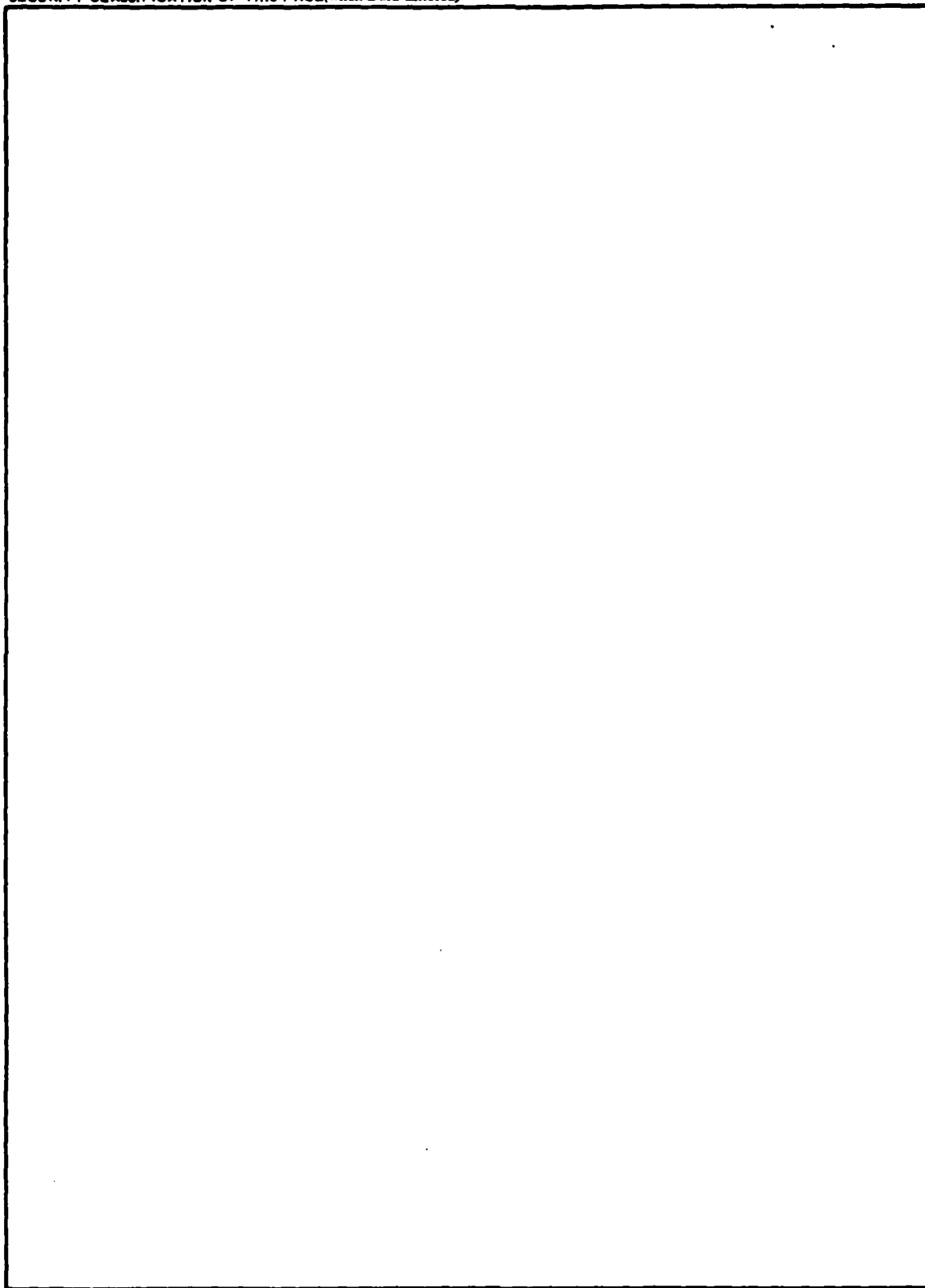
EDITION OF 1 NOV 65 IS OBSOLETE

1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

# TABLE OF CONTENTS

|                                                               | <u>PAGE</u> |
|---------------------------------------------------------------|-------------|
| 1. INTRODUCTION.....                                          | 1           |
| 2. GENERAL USAGE AND PROCESS QUOTAS.....                      | 2           |
| 3. HARDWARE/SOFTWARE INTERFACES.....                          | 2           |
| 4. SOFTWARE UTILITIES FOR IMAGE TRANSFER.....                 | 2           |
| 4.1 Images to Disk.....                                       | 2           |
| 4.2 Sequential Images to Disk: [MAXDISK]MTODSK3.....          | 3           |
| 4.3 Sequential Images to Disk: [MAXDISK]MTODSK3A.....         | 3           |
| 4.4 Sequential Images to Disk Plus: [MAXDISK]MTODSK4.....     | 4           |
| 5. UTILITIES FOR DISPLAY OF IMAGES ON DISK.....               | 5           |
| 5.1 Display of a Single Image from Disk: @SUBIMAGE.....       | 5           |
| 5.2 Display of IMAGES.DAT.....                                | 5           |
| 6. IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTOFIL..... | 5           |
| 7. IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTFIL2..... | 6           |
| 8. AVA FRAME BUFFER UTILITIES.....                            | 6           |
| 8.1 AVA Fields to GRINNELL.[AVA]AVAFIELDS.....                | 6           |
| 8.2 Writing to AVA Field Memory [AVA]AVAFWRITE.....           | 6           |
| 9. INPUT/OUTPUT TIMING.....                                   | 7           |
| 9.1 Writing a Ramp Pattern to AVA Memory [AVA]RAMPMAX.....    | 7           |
| 9.2 Reading AVA Fields [AVA]RAMPMAX2.....                     | 7           |
| 9.3 HBR - 3000 Data Transfer Timing [MAXDISK]MTODSKT.....     | 7           |
| 10. AVA FRAME BUFFER MEMORY DIAGNOSTICS.....                  | 7           |
| 11. UTILITIES FOR AUXILIARY DATA TRANSFER.....                | 8           |
| 11.1 Reading the 16 Auxiliary Channels (First Set Only).....  | 8           |
| 11.2 Plotting the 16 Auxiliary Channels.....                  | 8           |
| 11.3 Utilities for Reading the IRIG Time.....                 | 9           |
| 12. UTILITIES FOR HBR-3000 TAPE CONTROL AND SEARCH.....       | 9           |
| 12.1 Reading IRIG Time from the Tape Search Unit.....         | 9           |
| 12.2 HBR-3000 Drive Control.....                              | 9           |
| 12.3 IRIG List Generation for Image Transfers.....            | 9           |
| 12.4 Transferring IRIG List Images to Disk.....               | 9           |
| 12.5 Utilities for the Tape Search Unit.....                  | 9           |

# APPENDIXES

|                                          | <u>PAGE</u> |
|------------------------------------------|-------------|
| APPENDIX A [AVA.DISK]AVATODSK2 .....     | 11          |
| APPENDIX B [AVA.MAXDISK]MTODSK3 .....    | 17          |
| APPENDIX C [AVA.MAXDISK]MTODSK3A .....   | 22          |
| APPENDIX D [AVA.MAXDISK]MTODSK4 .....    | 28          |
| APPENDIX E [AVA.MAXDISK]MDSKTOGRN .....  | 33          |
| APPENDIX F [AVA.MAXDISK]MDOGRN .....     | 37          |
| APPENDIX G [AVA.MAXDISK]FIELDSEGRN ..... | 41          |
| APPENDIX H [AVA.MAXDISK]MDSKTOFIL .....  | 45          |
| APPENDIX I [AVA.MAXDISK]MDSKTFIL2 .....  | 52          |
| APPENDIX J [AVA]JAVAFIELDS .....         | 59          |
| APPENDIX K [AVA]JAVAGROUP8 .....         | 62          |
| APPENDIX L [AVA]JAVAGROUP9 .....         | 65          |
| APPENDIX M [AVA]JAVAFWRITE .....         | 68          |
| APPENDIX N [AVA]RAMPMAX .....            | 71          |
| APPENDIX O [AVA]RAMPMAX2 .....           | 74          |
| APPENDIX P [AVA.MAXDISK]MTODSKT .....    | 77          |
| APPENDIX Q [AVA]JAVAMENT .....           | 83          |
| APPENDIX R [AVA]JAVAMENT2 .....          | 88          |
| APPENDIX S [AVA]JAVAMENT .....           | 93          |
| APPENDIX T [AVA]JAVAMENT2 .....          | 98          |
| APPENDIX U [AVA]JAVAMENT .....           | 103         |
| APPENDIX V [AVA]JAV .....                | 107         |
| APPENDIX W [AVA]JAV2 .....               | 109         |

# APPENDIXES (CONTINUED)

|                                                           | <u>PAGE</u> |
|-----------------------------------------------------------|-------------|
| APPENDIX X [AVA]AUXPLOT.....                              | 111         |
| APPENDIX Y [AVA]AUXPLOT A.....                            | 114         |
| APPENDIX Z [AVA]AUXIRIG.....                              | 117         |
| APPENDIX AA [AVA]AUXIRIG2.....                            | 119         |
| APPENDIX AB [AVA.TAPEDRIVE]IRIGREAD.....                  | 121         |
| APPENDIX AC [AVA.TAPEDRIVE]COMMAND.....                   | 124         |
| APPENDIX AD [AVA.TAPEDRIVE]REVIEW.....                    | 126         |
| APPENDIX AE [AVA.TAPEDRIVE]IRTO DISK.....                 | 129         |
| APPENDIX AF [AVA.TAPEDRIVE]ISTO START.....                | 134         |
| APPENDIX AG [AVA.TAPEDRIVE]STATUS R.....                  | 137         |
| APPENDIX AH [AVA.TAPEDRIVE]IOTEST.....                    | 140         |
| APPENDIX AI AVA FRAME BUFFER I/O DRIVER.....              | 142         |
| APPENDIX AJ ON LINE DIGITIZER TAPE CONTROLLER DRIVER..... | 155         |



## 1.5 INTRODUCTION

The ON LINE DIGITIZER is a digital data recording/playback system capable of recording real time video for approximately 15 minutes continuously along with 16 auxiliary channels and one digital channel. The main input is either standard RS-170 or RS-343 video which is digitized at 10.08 MHz and phase locked to video horizontal sync. The first 512 samples of the 521 digitized on each line are stored in the ON LINE DIGITIZER frame buffer (AVA). The 16 auxiliary channels are -10. to +10. volt analog input that are digitized to 12 bits and sampled at the (video horizontal sync rate)/16. The digital channel is RS-232 input. All data other than the video is stored during the field into a memory and transferred to AVA frame buffer memory during the video vertical interval.

The ON LINE DIGITIZER consists of two major subsystems, the airborne unit and the ground unit. The airborne unit contains an Ampex AR-1700 28 track digital tape recorder with a custom digital processing unit. The airborne unit is for use in aircraft, range, or laboratory data recording. The ground unit contains an Ampex HBR-3000 28 track digital tape recorder with Datum IRIG search unit and digital image frame buffer. The ground unit takes the recorded 28 track tapes and plays back the data at selected rates for review and transfer of data to the computer. The image frame buffer contains memory area to hold 4 sequential video fields (512 pixels by 240 lines), 256 auxiliary words and frame IRIG time. This report describes and lists software utilities for use with the ON LINE DIGITIZER ground unit which are necessary to transfer data in various modes of operation and diagnostic utilities for hardware testing.

The two subsystems of the ON LINE DIGITIZER can be connected together in a real time mode bypassing the tape drives. This mode is used in diagnostic tests and for single snap shot digitization of images.

The ON LINE DIGITIZER has been in a continuous state of hardware upgrade and software development since its installation in the Sensor Signal Processing System (SSPS) and will continue to be modified to add additional improvements and capability which may impact execution of the software described in this report.

## 2.8 GENERAL USAGE AND PROCESS QUOTAS

This following software can ONLY be run under the AVA username. The process quotas under AVA have been set to allow large buffered I/O transfers.

The AVA frame buffer has four fields stored at a time. The dip switch on card 13 in the frame buffer housing allows display of fields 0 and 1 individually in realtime. Field 2 and 3 can also be displayed but only combined with 0 and 1 respectively. This dip switch along with the master toggle switch should be specifically set to allow viewing of the desired data during playback or in real time direct connect mode.

## 3.8 HARDWARE/SOFTWARE INTERFACES

The On line digitizer frame buffer (AVA) uses a programed I/O interface to the UNIBUS on a VAX 11/780. This unique device does not have DMA capability. The interface also requires total bus control during the I/O transfer. It will not tolerate things like interval clock interrupts, etc. and therefore the driver raises the interrupt priority level to "handle" this. The only after effect is the CPU clock and other things waiting for the I/O driver to release the CPU and UNIBUS do not get serviced and of course all other processes have to wait until the I/O is complete. The software driver AVDRIVER is listed in the appendix.

The On line digitizer search unit is interfaced to the VAX UNIBUS with a standard DEC DR11-C. This unit can control all ground unit functions or can be remotely under software control. The software driver ODDRIVER is listed in the appendix.

## 4.8 SOFTWARE UTILITIES FOR IMAGE TRANSFER

### 4.1 Images To Disk

The following programs are to be used when large amounts of disk space are available. Large being defined as enough contiguous space to hold the number of desired images which can be calculated as follows:

$$\text{BLOCKS} = \text{NI} + ((\text{NI} * \text{NC} * \text{NR} * \text{NB}) / \text{S12})$$

where BLOCKS=NUMBER OF CONTIGUOUS BLOCKS OF DISK SPACE NEEDED (512 BYTES PER BLOCK)  
NI=NUMBER OF IMAGES  
NC=NUMBER OF COLUMNS IN THE IMAGE (Horizontal Picture elements)  
NR=NUMBER OF ROWS IN THE IMAGE (Vertical Picture elements)  
NB=NUMBER OF BYTES PER PICTURE ELEMENT (normally 8 bits/pixel)

#### 4.1.1 Single Image To Disk : [DISK]AVATODSK2 -

The image to disk program will transfer the current image in the AVA frame buffers 0 and 1 to a complete frame in contiguous image disk format. Note that this is true if the frame is "frozen" or not. The normal mode is to freeze the frame using the STOP control on the ground unit and then execute the program to transfer the image. The disk format is transferred easily to tape with @DISK\$USERDISK:[SUBIMAGE]SUBNATO. The user will be asked to enter the disk storage name. The standard convention used for disk image names is described as follows:

XyyyyZZZZ.IMG

where

X= Alpha character

y= Sequence number

Z= Subimage sequence number (i.e. Subimage of Xyyyy)

#### 4.2 Sequential Images To Disk : [MAXDISK]MTODSK3

Sequential images can be transferred from the HBR-3000 during 32 to 1 playback or 3 3/4 speed to disk. MTODSK3 checks the disk and allows transfer only after it knows how many contiguous blocks are available for image storage. The images are placed in IMAGES.DAT. Only fields 0 and 1 are transferred to disk. Fields 2 and 3 are skipped to allow time for field 0 and 1 transfer completion. This of course means only every other frame is transferred to disk.

#### 4.3 Sequential Images To Disk : [MAXDISK]MTODSK3A

This program performs the same functions as MTODSK3 however in addition the frame buffer IRIG time is also stored in memory for each image. After all images have been transferred to disk the stored IRIG times are written to DISK\$AVA:[AVA]IRIGS.DAT.

#### 4.4 Sequential Images To Disk Plus : [MAXDISK]MTODSK4

Sequential images can be transferred from the HBR-3000 during 32 to 1 playback or 3 3/4 speed to disk. MTODSK4 checks the disk and allows transfer only after it knows how many contiguous blocks are available for image storage. The images are placed in IMAGES.DAT. The program puts out the maximum number of fields to disk possible. This is more data than MTODSK3 will transfer since it checks to see if I/O is complete and then transfers the next available field regardless of which one it is.

##### 4.4.1 Initial And Subsequent Runs Of MTODSK3,MTODSK3A Or MTODSK4 -

The initial run of MTODSK3,MTODSK3A or MTODSK4 will not start I/O transmission immediately after the beginning of the run. The disk space is interrogated to decide what is the maximum contiguous space available. In order for the entire disk space to be usable contiguously, the disk pack must be initialized with the /INDEX-BEGINNING qualifier. Interrogation may take several seconds before the search is complete. After the largest space is found the file is opened and the area is allocated. The size of this allocated area and hence the number of images which can be written is highly dependant on the specific medium (disk pack) used in the disk drive. Individual disk packs have different characteristics one of which is where the bad blocks, if any, are located. If very long sequences of data are needed to be transferred several packs may have to be checked before actual execution. After the IMAGES.DAT file has been created by the initial run of MTODSK3,MTODSK3A or MTODSK4 the file will be over written by any subsequent running of these programs. Therefore if the data on the disk in IMAGES.DAT is needed in this form another disk pack is required before the subsequent runs.

#### 5.0 UTILITIES FOR DISPLAY OF IMAGES ON DISK

### 5.1 Display Of A Single Image From Disk : @SUBIMAGE

The subimage data base software uses the SSPS standard image file format which consists of a contiguous image file and associated header file with the same name. The subimage data base software is executed by the command @SUBIMAGE. The operator enters the following answers to program questions:

1. Is a new image list file required? NO
2. Is image from disk or tape? (D or T) D

The operator then enters the following commands:

1. L - tells the program you want to load an image file
2. - "enter the image file name here XYYYYZZZ.IMG"
3. Z - tells the program you want to display loaded image
4. <CR> - just enter carriage return for the AGC value

.....The image is now displayed on the Grinnell .....

5. N - tells the program you want to go to the next image
6. GO TO 1.

### 5.2 Display Of IMAGES.DAT

IMAGES.DAT is generated by MTODSK3, MTODSK3A, or MTODSK4 on DISK\$AVA:[AVA]. The file is one large contiguous set of images 512 by 248 pixels per field. To transfer IMAGES.DAT to the GRINNELL one of the following routines can be used:

- \* [MAXDISK]MDSKTOGRN - Transfers full frames of images to the GRINNELL field at a time.
- \* [MAXDISK]MDTOGRN - Transfers full frames of images to the GRINNELL frame at a time.
- \* [MAXDISK]FIELDSEGRN - Transfers fields as one image to the GRINNELL field at a time.

### 6.8 IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTOFIL

MDSKTOFIL takes the IMAGES.DAT file and transfers, from the desired starting image, each frame to a unique image file in SSPS standard image format with a header file. The number

of images to transfer is an input as well as the increment between frames. The IMAGES.DAT can be created by MTODSK3, MTODSK3A, or MTODSK4.

#### 7.8 IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTFIL2

This program performs the same function as MDSKTOFIL but in addition reads the DISK\$AVA:[AVA]IRIGS.DAT file and places the IRIG time in the NATO header two for each image header file. The IRIGS.DAT file is created by MTODSK3A automatically, however, if the IRIG times are not available the IRIGS.DAT file can also be generated by "other" means.

#### 8.8 AVA FRAME BUFFER UTILITIES

##### 8.1 AVA FIELDS TO GRINNELL. [AVA]AVAFIELDS

AVAFIELDS displays the current field desired residing in the AVA frame buffer on the GRINNELL. The operator input is the field number 0, 1, 2, or 3. The operator in most cases will "freeze" the video before transferring the field image by using the STOP control on the ground unit. The fields are loaded by the hardware in sequence first field 0 then 1, 2 and 3 and the process is repeated as long as input data continues.

[AVA]AVAGROUP8 reads the current AVA fields 0 and 1 and puts them on the GRINNELL updated field at a time continuously.

[AVA]AVAGROUP9 is the same program as AVAGROUP8 except the block sizes are as large as possible for AVA frame buffer reads.

##### 8.2 Writing To AVA Field Memory. [AVA]AVAFWRITE

AVAFWRITE allows the user to write to a specific field memory area in the AVA frame buffer. The data written is an input to the program therefore 256 values can be designated. The same value is written over the entire specified field memory area.

## 9.8 INPUT/OUTPUT TIMING

### 9.1 Writing A Ramp Pattern To AVA Memory [AVA]RAMPMAX

RAMPMAX writes a double ramp grayscale to the AVA memory in all four fields. This routine demonstrates the write timing to the AVA memory from the VAX 11/780. The frame buffer needs to be stopped to allow only the computer to write into AVA field memory.

### 9.2 Reading AVA Fields [AVA]RAMPMAX2

RAMPMAX2 reads frame 1 of the AVA MEMORY or fields 0 and 1. This routine demonstrates the read timing from the AVA memory to the VAX 11/780.

### 9.3 HBR - 3000 Data Transfer Timing. [MAXDISK]MTODSKT

MTODSKT will time AVA reads with a variable write delay, for simulation of the disk write time, for N fields. The HBR-3000 speed for reasonable data rates must be at 3 3/4 ips or 1 7/8 ips.

## 10.8 AVA FRAME BUFFER MEMORY DIAGNOSTICS

1. FAVAMENT - 4 pattern test on video memory
2. FAVAMENT2 - 4 pattern test on video memory \*
3. AVAMENT - user entered pattern test
4. AVAMENT2 - user entered pattern test \*
5. AAVAMENT - 4 pattern test on ALL AVA memory

\* - Specific error printouts

## 11.8 UTILITIES FOR AUXILIARY DATA TRANSFER

### 11.1 Reading The 16 Auxiliary Channels (First Set Only)

[AVA]AUX will display the channel number and the voltage input on each of the 16 auxiliary input channels. The display will be up dated by direct cursor addressing of the screen using the channel address in the auxiliary word and no scrolling will occur. The input channels are sampled at the ( video horizontal sync rate)/16 and can range from -10. to +10. volts. This program reads only the first 16 words from the auxiliary memory area and therefore will not reflect the actual signal frequency response recorded or being sampled in real time.

[AVA]AUX2 will display the 16 auxiliary channels as does AUX except the screen will scroll and reflect the staggering positions of the data as actually stored in the auxiliary memory. The channel addresses are not used for display of the data and the format is in hexadecimal only. This routine like AUX reads only the first 16 words from the auxiliary memory area.

### 11.2 Plotting The 16 Auxiliary Channels

[AVA]AUXPLOT will plot all sixteen samples of the requested auxiliary channel. The scale is +10. to -10. volts vertically with the samples plotted horizontally in sequence repeatedly. The data in the AVA frame buffer is read before each plot is generated. The plot instructions use VT-52 escape sequences and therefore require a compatible terminal. Prior to execution of this routine it is necessary to ensure that the terminal is in VT52 mode by executing the following DCL command. SET TERM/VT52. This process should also be followed in AUXPLOT.

[AVA]AUXPLOT will plot all sixteen samples of all channels five times across the plot. This is useful in testing auxiliary channel frequency response by connecting all channels to the same varying signal and adjusting the amplitude and rate of the variation.

### 11.3 Utilities For Reading The IRIG Time

[AVA]AUXIRIG will display the IRIG time placed in the AVA frame buffer. This IRIG time is updated either in real time or by HBR-3000 playback. The display places the IRIG on the screen without scrolling.

[AVA]AUXIRIG2 will scroll the IRIG time on the screen and is used for checks of video sync stability and IRIG read timing.

## 12.0 UTILITIES FOR HBR-3000 TAPE CONTROL AND SEARCH

### 12.1 Reading IRIG Time From The Tape Search Unit

[AVA.TAPEDRIVE]IRIGREAD will freeze the IRIG output register at each read and transfer the IRIG to the display and repeat continuously.

### 12.2 HBR-3000 Drive Control

[AVA.TAPEDRIVE]COMMAND lets the knowledgeable user remotely control the HBR-3000. The search unit REMOTE/LOCAL switch must be in REMOTE otherwise this program has no effect. It is suggested that the user be familiar with the functions in the Datum Search Unit Manual before running this program. Some typical commands are shown below:

NOTE: all commands are in octal for decoding the bit functions

|        |                                                      |
|--------|------------------------------------------------------|
| 157000 | STOP                                                 |
| 150001 | TRANSLATE IRIG A WITH ZERO FRAME BYPASS              |
| 156400 | UPDATE TIME, RESET RECORD ENABLE,<br>RESET INTERRUPT |
| 157476 | SET THE FILTERS TO 120 IPS                           |
| 157201 | DRIVE FORWARD AT 120 IPS                             |
| 157221 | DRIVE FORWARD AT 240 IPS (FAST FOWARD)               |
| 157061 | DRIVE FORWARD AT 3 3/4 IPS (32 TO 1)                 |
| 157222 | DRIVE REVERSE AT 240 IPS (FAST REVERSE)              |
| 157202 | DRIVE REVERSE AT 120 IPS                             |
| 157206 | SINGLE CYCLE SEARCH MODE AT 120 IPS                  |

### 12.3 IRIG List Generation For Image Transfers

[AVA.TAPEDRIVE]REVIEW is for generating an IRIG list file of the single images the user wants on disk in disk image format. The user would start REVIEW and then play the tape on the HBR-3000 at 120 ips (normal speed). When a desired image appears the user presses a return at the terminal. When a return is pressed the current IRIG time is read from the AVA frame buffer and is written to disk. This is repeated as many times as needed. After reviewing the portion of tape required by the user typically several IRIG times would be in the REVIEW.IGR file, and the user would then type in Z to terminate program execution.

### 12.4 Transferring IRIG List Images To Disk

[AVA.TAPEDRIVE]RTODISK will use the IRIG list file REVIEW.IGR to scan the tape on the HBR-3000 and transfer the images to disk in disk image format. The user must enter the beginning file name for the first image and there after the file name sequence number will be automatically incremented. Note that there must be enough contiguous disk space available in the default file directory else the program will abort.

### 12.5 Utilities For The Tape Search Unit

1. [AVA.TAPEDRIVE]TOSTART allows the user to enter an IRIG time to search for. The tape will be transferred to the position where the IRIG time occurs.
2. [AVA.TAPEDRIVE]STATUSR displays the current status of the HBR-3000 repeatedly.
3. [AVA.TAPEDRIVE]IOTEST allows the user to perform a fully functional test on the DR11-C interface tied to the Tape Search Unit. The maintenance cable or equivalent must be connected from the output port to the input port. This program tests all data bits on the DR11-C and the "A" interrupt hardware.

APPENDIX A  
[AVA.DISK]AVATODSK2

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM WRITES THE CURRENT AVA FRAME BUFFER IMAGE
C   FIELDS 0 AND 1 ON TO DISK IN SUBIMAGE DATA BASE FORMAT.
C
C   THE IMAGE NAME XXX IS REQUESTED. THIS NAME IS USED FOR THE
C   XXX.IMG FILE AND THE XXX.HDR FILE.
C
C   THIS FILE CAN THEN BE ACCESSED BY ANY OF THE SUBIMAGE DATA BASE
C   SOFTWARE SET.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM IS DIFFERENT FROM AVAITODSK.FOR IN THAT
C   AVAITODSK.FOR WRITES TO DISK IN A COMPLETELY DIFFERENT FORMAT AND ONLY
C   TO DISK$IMAGES:[AVAJIMAGES.DAT WHEREAS THIS PROGRAM WRITES IT OUT IN
C   THE SUBIMAGE DATA BASE FORMAT AND WILL USE THE NAME INPUT BY THE USER.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   PARAMETER IEVF = 4
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
C   INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
C   INTEGER*4 IMGADR,SYSS$ASSIGN,IMGADR2,SYSS$GETMSG
C   INTEGER*4 LIB$FREEVM,LIB$GETVM,SYSS$DASSGN
C   INTEGER AVACHAN
C   DIMENSION AR(65,65),BR(65,65)
C   CHARACTER*60 TITLE,MSGBUF
C   TITLE='WRITE TO DISK TIME FOR ONE 512X480 IMAGE'
C   DSPSCF=1.0
C   IMGMAPC(1)=1
C   IMGMAPC(2)=1
C   I=SYSS$ASSIGN('TT',IVTC,,)
C   IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
C   I=SYSS$ASSIGN('GRA0',GRCHAN,,)
C   IF(.NOT.I)THEN
C   TYPE *,'ERROR IN GRINNELL CHANNEL ASSIGN'

```

[AVA.DISK]AVATODSK2

```

      STOP
      ENDIF
C      CODE TO READ AVA IMAGE INTO VIRTUAL MEMORY
      ISTATUS=SYSSASSIGN('AVA$',AVACHAN,,)
      IF(.NOT.ISTATUS)TYPE *, ' ERROR IN AVA GRCHANNEL ASSIGN'
C      FNAM = 'BARDOTS.IMG'
C      READ(5,5678)FNAM
5678  FORMAT(A88)
C      TYPE *,FNAM
C      CALL READSUB(ILEN,IWD,IMGADR)   I READ IMAGE FROM DISK
C      IMGMAPC(3)=ILEN
C      IMGMAPC(4)=IWD
C      IMGMAPC(3)=512
C      IMGMAPC(4)=512
C      CALL DSPIMG(XVAL(IMGADR))      I PUT IMAGE ON THE GRINNELL
      NCOL=512
      NROW=488
      ILEN=NROW
      IWD=NCOL
      IMGMAPC(3)=ILEN ILENGTH OF IMAGE
      IMGMAPC(4)=IWD  IWIDTH OF IMAGE
      I=IWD
      NBYT=(I+1)*ILEN*2
      I=LIB$GETVM(NBYT,IMGADR)
      IF(.NOT.I)TYPE *, ' ERROR IN VIRTUAL MEMORY ASSIGNMENT 1'
      CALL AVAREAD(XVAL(IMGADR),AVACHAN)
      HEAD(8)= '      1'      IONE CHARACTER PER CHANNEL

      I = LIB$GETVM(18888,HDR2ADR)
      IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
      CALL ADDHDR2(XVAL(HDR2ADR))
      HDR2LEN=576
      CURRENTNUMFL=8
C      CALL DSPIMG(XVAL(IMGADR))      I PUT IMAGE ON THE GRINNELL
      TYPE *, 'IWD=',IWD,' ILEN=',ILEN
      FNAM='SIMS88881.IMG'
      TYPE *, 'ENTER OUTPUT FILE NAME. (123456789.IMG)'
      FNAM= '
123  READ(5,123)FNAM
      FORMAT(A)
C      CALL TIMRB
C      IWD=512
C      ILEN=512
C      CALL TODISK(XVAL(IMGADR),IWD,ILEN,AVACHAN)
C      CALL TIMRE
C      CALL HEADER(TITLE)
C      I=IWD
C      NBYT=(I+1)*ILEN*2
C      I=LIB$GETVM(NBYT,IMGADR2)
C      IF(.NOT.I)TYPE *, ' ERROR IN VIRTUAL MEMORY ASSIGN FOR OUTPUT IMAGE'
C      I=LIB$GETVM(NBYT,IMGADR3)
C      IF(.NOT.I)TYPE *, ' ERROR IN VIRTUAL MEMORY ASSIGN FOR OUTPUT IMAGE'
C      FNAM(23:31)='A88838881'
C      CALL READSUB(ILEN,IWD,IMGADR3) I READ IMAGE FROM DISK
C      IMGMAPC(3)=ILEN

```

[AVA.DISK]AVATODSK2

```

C      IMGMAPC(4)=IWD
C      TYPE *, 'IWD=', IWD, ' ILEN=', ILEN
      STOP '512X48# IMAGE WRITTEN TO DISK
      1
      END
      SUBROUTINE TODISK(IMAGE, IWD, ILEN, AVACHAN)
      EXTERNAL IOSREADYBLK
      INCLUDE 'DISKSUSERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
      INCLUDE 'DISKSUSERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
      INCLUDE 'DISKSUSERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
      INCLUDE 'DISKSUSERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
      INTEGER*2 US, TS, UM, TM
      INTEGER*2 UH, TH, UD, TD, HD
      INTEGER*2 MS, HMS, TMS, IOSB(4)
      INTEGER*2 IMAGE(NCOL+1, NROW), HDR2LEN, D(8), X, Y
      INTEGER*4 AUTOWRTSB
      INTEGER AVACHAN, SYSSQIOW
      CHARACTER*10000 HDR2ADR
      CHARACTER*3 MONTH, DAY, YEAR*2, WD*8, LEN*8, TIMEA*8
      CHARACTER*5 IFIRST5, ILAST4*4, TNAME*9
      HDR2LEN=HDR2LEN
C      TYPE *, 'HDR2LEN', HDR2LEN
C      CALL CNVRT(XVAL(HDR2ADR), HDR2LEN, HDR2ADR)
C      TYPE *, HEAD
      CALL IDATE(IMONTH, IDAY, IYEAR)
      ENCODE(3, 200, MONTH)IMONTH
      ENCODE(3, 200, DAY )IDAY
200  FORMAT(I3)
      ENCODE(2, 100, YEAR )IYEAR
      HEAD(3)='OLDFAAD '
      HEAD(1)='USAMICOM'
      HEAD(2)=YEAR//MONTH//DAY
100  FORMAT(I2)
      ENCODE(8, 200, WD)IWD
      HEAD(11)(6:8)=WD(1:3)
      ENCODE(8, 200, LEN)ILEN
      HEAD(12)(6:8)=LEN(1:3)
C      TYPE *, HEAD
      IBRACKET=INDEX(FNAM, 'J')
      IPERIOD=INDEX(FNAM, '.')
      TNAME=FNAM(IPERIOD-9:IPERIOD-1)
      IBRACKET=IBRACKET
      IF(IPERIOD-10.LE.IBRACKET)THEN
      IZERO=ABS(IPERIOD-10)
      TNAME(1:IZERO)=' '
      ENDIF
      ILAST4=TNAME(6:9)
      IFIRST5=TNAME(1:5)

      HDR2ADR(1:8)= 'FN*XXXXX'
      HDR2ADR(11:18)='XXXXX.IMG'
      HDR2ADR(4:8)=IFIRST5
      HDR2ADR(11:14)=ILAST4
      HDR2ADR(51:58)='SLREDALA'
      HDR2ADR(41:48)='LTXXXXXX'

```

[AVA.DISK]AVATODSK2

```

HDR2ADR(31:38)='RT#####'
HDR2ADR(21:28)='DT#####'
C HDR2ADR(351:358)=MILISECONDS
CALL TIME(TIMEA)
HDR2ADR(43:44)=TIMEA(1:2)
HDR2ADR(45:46)=TIMEA(4:5)
HDR2ADR(47:48)=TIMEA(7:8)
C
C LETS READ THE RANGE IRIG TIME FROM THE AVA FRAME BUFFER
C
AVAACR='435'O
X='1###O'
Y=2
ISTATUS=SYSSQIOW(XVAL(1),XVAL(AVACHAN),XVAL(XLOC(10$READVBLK)),
1IOSB,,,
C 1D,XVAL(8),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
1D,XVAL(8),XVAL(X),XVAL(Y),XVAL(1),XVAL(AVAACR))
C IF(AVACSR.EQ.8)AVACSR=1
DO I=2,4
D(I)=NOT(D(I))
ENDDO
HMS=IAND(ISHFT(D(2),-8),'F'X)
TMS=IAND(ISHFT(D(2),-4),'F'X)
MS=IAND(D(2),'F'X)
US=IAND(ISHFT(D(2),-12),'F'X)
TS=IAND(D(3),7)
UM=IAND(ISHFT(D(3),-3),'F'X)
TM=IAND(ISHFT(D(3),-7),7)
UH=IAND(ISHFT(D(3),-18),'F'X)
TH=IAND(ISHFT(D(3),-14),3)
UD=IAND(D(4),'F'X)
TD=IAND(ISHFT(D(4),-4),'F'X)
HD=IAND(ISHFT(D(4),-8),'F'X)
HDR2ADR(33:33)=CHAR(TH+48)
HDR2ADR(34:34)=CHAR(UH+48)
HDR2ADR(35:35)=CHAR(TM+48)
HDR2ADR(36:36)=CHAR(UM+48)
HDR2ADR(37:37)=CHAR(TS+48)
HDR2ADR(38:38)=CHAR(US+48)
C
HDR2ADR(351:358)=MILISECONDS
HDR2ADR(279:279)=CHAR(HMS+48)
HDR2ADR(280:280)=CHAR(TMS+48)
HDR2ADR(281:281)=CHAR(MS+48)
C WRITE(6,13)(D(I),I=2,4),HD,TD,UD,TH,UH,TM,UM,TS,US,
C 1HMS,TMS,MS
CC13 FORMAT(1X,3(1X,06),5X,3Z1,':',1X,2Z1,':',Z1,Z1,':',2Z1,
C 1':',3Z1)
C HDR2ADR(33:38)=HDR2ADR(43:48)
C HDR2ADR(23:28)=HDR2ADR(43:48)
HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)
C
CALL UNCNVRT(XVAL(HDR2ADR),HDR2LEN,HDR2ADR)
C TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C TYPE*,' WRITING ',FNAM(1:48)

```

CAVA.DISKJAVATODSK2

```

IHD2=HDR2LEN      !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,XVAL(HDR2ADR))
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
RETURN
END
SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT,IOLINE)
INCLUDE 'DISKSUSERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
BYTE BINPUT(1),BYTE(2)
INTEGER*2 OUT(NCOL+1,NROW),BYTES,SLU
EQUIVALENCE(BYTES,BYTE)
DATA SLU/'34011'O/
I=0
DO 100 IX=1,NUMB
I=I+1
IF(I.EQ.512)THEN
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=BYTES
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
C OUT(I+1,IOLINE)=SLU
WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
I=0
IOLINE=IOLINE+2
GO TO 100
ENDIF
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34 FORMAT(IX,I3,IX,I3,2X,06)
100 CONTINUE
RETURN
END

SUBROUTINE AVAREAD(IMAGE,AVACHAN)
EXTERNAL IOSREADVBLK
INCLUDE 'DISKSUSERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
INTEGER*2 IMAGE(NCOL+1,NROW)
INTEGER AVACHAN,SYSSQIOW,AVACSR,AVAACR,X,Y
INTEGER*2 INPUT(15360),IOSB(4)
BYTE BINPUT(30720)
EQUIVALENCE(BINPUT,INPUT)
AVACSR=0
AVAACR='415'O
Y=6
X=0
ICOUNT=0
IADDR=1
IOLINE=1
1 ISTATUS=SYSSQIOW(XVAL(1),XVAL(AVACHAN),XVAL(XLOC(IOSREADVBLK)),
IOSB,,
1INPUT,XVAL(30720),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0)AVACSR=1
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C WRITE(6,54)BINPUT
54 FORMAT(IX,16(IX,03))

```

[AVA.DISK]AVATODSK2

```
NUMB=38728
CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
Y=Y+38
ICOUNT=ICOUNT+1
IF(ICOUNT.EQ.4)THEN
C ISTATUS=SYSSQIOW(XVAL(1),XVAL(AVACHAN),XVAL(XLOC(IOSREADVBLK)),
C IOSB,,
C IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C NUMB=8192
C CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
IOLINE=2
Y='206'O
X=8
ENDIF
IF(ICOUNT.EQ.8)THEN
C ISTATUS=SYSSQIOW(XVAL(1),XVAL(AVACHAN),XVAL(XLOC(IOSREADVBLK)),
C IOSB,,
C IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C NUMB=8192
C CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
RETURN
ENDIF
GO TO 1
57 CONTINUE
ISTATUS=SYSSGETMSG(XVAL(ISTATUS),MSGLEN,MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG(XVAL(IOSB(1)),MSGLEN,MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:',MSGBUF
STOP
END
```

LAVAL MAXDISKIMTODSK3

17

[AVA.MAXDISK]MTODSK3

```

        ISTATUS=SYSSASSIGN('AVA',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
        TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
        NAME='DISKSAVA:'
        BUFFERL=4
        DEVCODE=DVISFREEBLOCKS
        BUFFERA=XLOC(IFREE)
C      RETURNLA=XLOC(RETURNL)
        ISTATUS=SYSSGETDVI(XVAL(3),.NAME,BUFFERL,...)

        IF(.NOT.ISTATUS)TYPE *, 'PARAMETER ERROR IN GETDVI'
        ISTATUS=SYSSWAITFR(XVAL(3))
        TYPE *, 'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
        MAXIMAGES=IFREE/513
        TYPE *, 'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C      MAXIMAGES=30      ITHIS IS FOR DEGUG ONLY
C
        NIMAGES=MAXIMAGES
7775     INSZ=NIMAGES*480
        FNAME='DISKSAVA:[AVAIIMAGES.DAT'
        TYPE *, 'OPENING',FNAME
        OPEN(UNIT=30.NAME=FNAME,TYPE='UNKNOWN',
        IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        ZRECORDTYPE='FIXED',RECORDSIZE=4096,ERR=777)
        GO TO 776
777     NIMAGES=NIMAGES-10
        IF(NIMAGES.LT.0)STOP 'NIMAGES LESS THAN ZERO!!!!'
        GO TO 7775
776     TYPE *, 'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
        ISTATUS=SYSSASSIGN('AVA',AVACHAN,,)
        IF(.NOT.ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
        INLOCK(1)=XLOC(BINPUT(1))
        INLOCK(2)=XLOC(BINPUT(131072))
        K=SYSSLKWSET(INLOCK,IOLOCK,)
        TYPE *, ' INLOCK(1)= ',INLOCK(1), ' INLOCK(2)= ',INLOCK(2)
        TYPE *, ' IOLOCK(1)= ',IOLOCK(1), ' IOLOCK(2)= ',IOLOCK(2)
        IF(.NOT.K)TYPE *, ' UNABLE TO LOCK BUF'
        AVAACR='415'0
C      K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(105WRITEVBLK)),IOSB,..
C      IBUF(1),XVAL(20),...)
        IEVFO=4
        IEVFO1=5
        IMAGEN=1
        IBLOCK=1
        AVACSR=0
        CALL TIMRB
        X=0
10      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.NE.0)GO TO 10
        ICURR=IFIELD
C      TYPE *, 'ICURR=',ICURR
11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR      ICURRENT FIELD TO PUT ON DISK

```

[AVA.MAXDISK]MTODSK3

```

      ICURR=IFIELD          ICURRENT FIELD BE LOADED INTO THE AVA
C      TYPE =, 'ICURR=', ICURR, 'ISTOREFIELD=', ISTOREFIELD
C      GO TO 11
      ICOUNT=0
1      Y=YA(ISTOREFIELD+1)
      ISTATUS=SYSSWAITFR(XVAL(IEVFO1))
      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
      1IOSB,,,
      1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      IF(AVACSR.EQ.0) AVACSR=1
      ISTATUS=SYSSQIO(XVAL(IEVFO1),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
      1IOSB,,,
      1BINPUT(1),XVAL(32768),XVAL(1BLOCK),,,)
      1BLOCK=1BLOCK+64
      Y=Y+32
2      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
      1IOSB,,,
      1BINPUT(32679),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
      1IOSB,,,
      1BINPUT(32679),XVAL(32768),XVAL(1BLOCK),,,)
      1BLOCK=1BLOCK+64
      Y=Y+32
3      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
      1IOSB,,,
      1BINPUT(65537),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
      1IOSB,,,
      1BINPUT(65537),XVAL(32768),XVAL(1BLOCK),,,)
      1BLOCK=1BLOCK+64
      Y=Y+32
4      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
      1IOSB,,,
      1BINPUT(98305),XVAL(24576),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
      1IOSB,,,
      1BINPUT(98305),XVAL(24576),XVAL(1BLOCK),,,)
      1BLOCK=1BLOCK+48
C      IF(AVACSR.EQ.0)AVACSR=1
C      WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,03))
C      NUMB=32768
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10$WRITEVBLK)),IOSB,,,
C      1BOUT(1),XVAL(65534),,,,)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10$WRITEVBLK)),IOSB,,,
C      1BOUT(65535),XVAL(130),,,,)
C      ICOUNT=ICOUNT+1
C      IF(ICOUNT.NE.4)GO TO 1
C      CALL FIELD(IFIELD,AVACSR)
C      IF(IFIELD.EQ.ICURR)GO TO 11

```

[AVA.MAXDISK]MTODSK3

```

C      IF(IFIELD.NE.IAND(ICURR+1,3))THEN
C      TYPE*, 'FATAL ERROR...*****I/O TO SLOW'
C      TYPE *, 'BLOCK NUMBER=', IBLOCK
C      TYPE *, 'ICURR=', ICURR, ' IFIELD=', IFIELD
C      CALL TIMRE
C      CALL HEADER(TITLE)
C      ISTATUS=SYSSDASSGN(XVAL(IDISK))
C      CLOSE(UNIT=30)
C
C      STOP
C      ENDIF
C      ICURR=IAND(ICURR+1,3)
C      IF(ICOUNT.EQ.0)THEN
C      ICOUNT=1
C      ISTOREFIELD=1
C      GO TO 1
C      ENDIF
C      TYPE*, IMAGEN/2, NIMAGES
C      IF(IMAGEN.GE.NIMAGES)THEN
C      TYPE *, IMAGEN, 'IMAGES WRITTEN TO DISK IN IMAGES.DAT'
C      CALL TIMRE
C      CALL HEADER(TITLE)
C      ISTATUS=SYSSDASSGN(XVAL(IDISK))
C      CLOSE(UNIT=30)
C      STOP 'I/O COMPLETE.....'
C      ENDIF
C      IMAGEN=IMAGEN+1
C      GO TO 10
57 CONTINUE
C      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
C      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
C      MSGBUF='
C      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'I/O STATUS:', MSGBUF
C      STOP
C11 FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C      K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC(IOSWRITEVBLK)), IOSB,,)
C      I1SETUP3, XVAL(4),,,)
C      END
C      SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
C      BYTE BINPUT(1), BYTE(2)
C      INTEGER*2 OUT(513,1), BYTES, SLU
C      EQUIVALENCE(BYTES, BYTE)
C      DATA SLU/'34011'O/
C      I=0
C      IOLINE=1
C      DO 100 IX=1, NUMB
C      I=I+1
C      IF(I.EQ.512)THEN
C      BYTE(1)=BINPUT(IX)
C      OUT(I, IOLINE)=BYTES

```

1AVA.MAXDISKIMTODSK3

```

C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
C      WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
      ENDIF
      BYTE(1)=BINPOT(IX)
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34     FORMAT(1X,I3,1X,I3,2X,O6)
100    CONTINUE
      RETURN
      END

SUBROUTINE FIELD(IFIELD,AVACSR)
  INTEGER AVACSR
  EXTERNAL IOSWRITEBLK,IOSREADVBLK
  INTEGER SYSS$ASSIGN,SYSS$QIOW,SYSS$QIO
  INTEGER SYSS$GETMSG
  INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
  INTEGER*2 INPUT,OUTPUT,INIT(4)
  CHARACTER *80 MSGBUF
  COMMON/AVACHAN/ITCHAN
  DATA IFIRST/1/
  ISAVE=AVACSR
  IF(IFIRST)THEN
    AVACSR='4000'O  ISET MEMORY WINDOW ENABLE AND INITIALIZE AVA
    IFIRST=0
  ELSE
    AVACSR='4001'O
  ENDIF
  ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
    1IOSB,,,
    IOUTPUT,XVAL(2),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(1AVAACR))
C      IF(AVACSR.EQ.'4000'O)AVACSR='4001'O
C      IF(ISTATUS) GO TO 501
C      TYPE *, ' ERROR IN QIOW CALL '
C      ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
C      MSGBUF=' '
C      ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'I/O STATUS:',MSGBUF
501    AVACSR=ISAVE
      IFIELD=IAND(OUTPUT,3)
      RETURN
      END

```

# APPENDIX C

[AVA.MAXDISK]MTODSK3A

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      THIS PROGRAM READS THE CURRENT AVA FRAME FIELDS 0,1 AND WRITES
C      THEM TO DISK$AVA:[AVA]IMAGES.DAT AS THEY ARE PRESENTED BY PLAYING
C      THE HBR-3000 BACK AT 32X1.
C
C      THIS PROGRAM DIFFERS FROM MTODSK3 IN THAT THE IRIG TIME IN THE FRAME
C      BUFFER IS READ FOR EACH IMAGE AND STORED IN A BUFFER.
C      THE BUFFER IS OUTPUT AFTER ALL IMAGES HAVE BEEN TRANSFERED TO A FILE
C      NAMED DISK$AVA:[AVA]IRIGS.DAT. THESE IRIGS WILL CORRESPOND ONE TO ONE
C      WITH THE IMAGES IN IMAGES.DAT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      EXTERNAL IOSWRITEVBLK,IOSREADVBLK,MITLS1
C      INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4),D(4)
C      REAL*8 QD,IRIG(1000)
C      INTEGER      SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
C      INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
C      INTEGER*2 X,Y,VA(4),SYSS$DASSGN
C      INTEGER*2 BYTES
C      INTEGER*2 OUTPUT,INIT(4)
C      INTEGER*2 INPUT(65536)
C      INTEGER*2 US,TS,UM,TM
C      INTEGER*2 UH,TH,UD,TD,HD
C      INTEGER*2 MS,HMS,TMS
C      BYTE BINPUT(32768)
C      BYTE BINPUT(131072)
C      INTEGER AVACSR,AVAACR,SYSS$LKWSET,INLOCK(2),IOLOCK(2)
C      INTEGER*2 ISETUP2(2),ISETUP3(2)
C      CHARACTER *80 MSGBUF
C      CHARACTER*60 TITLE,FNAME*60
C      CHARACTER*60 NAME
C      INTEGER*2 BUFFERL,DEVCODE
C      INTEGER SYSS$GETDVI,DVIS$FREEBLOCKS
C      INTEGER IFREE
C      INTEGER BUFFERA,ZERO
C      COMMON/PRACHAN/IDISK
C      COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
C      COMMON/AVACHAN/ITCHAN
C      EQUIVALENCE(BUF(1),ISETUP(1))

```

[AVA.MAXDISK]MTODSK3A

```

EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(D,QD)
DATA YA/6,'206'O,'406'O,'606'O/
DATA DVISFREEBLOCKS/'0000002A'X/,ZERO/0/
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
NAME='DISKSAVA:'
BUFFERL=4
DEVCODE=DVISFREEBLOCKS
BUFFERA=XLOC(IFREE)
C RETURNLA=XLOC(RETURNL)
ISTATUS=SYSSGETDVI(XVAL(3),,NAME,BUFFERL,,, )

IF(.NOT. ISTATUS)TYPE *, 'PARAMETER ERROR IN GETDVI'
ISTATUS=SYSSWAITFR(XVAL(3))
TYPE *, 'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
MAXIMAGES=IFREE/481
TYPE *, 'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C MAXIMAGES=5      ITHIS IS FOR DEGUG ONLY
C
NIMAGES=MAXIMAGES-2
7775 INSZ=NIMAGES*480
FNAME='DISKSAVA:[AVA]IMAGES.DAT'
TYPE *, 'OPENING',FNAME
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4096,ERR=777)
GO TO 776
777 NIMAGES=NIMAGES-10
IF(NIMAGES.LT.0)STOP 'NIMAGES LESS THAN ZERO!!!!'
GO TO 7775
776 TYPE *, 'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
ISTATUS=SYSSASSIGN('AVA0',AVACHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
INLOCK(1)=XLOC(BINPUT(1))
INLOCK(2)=XLOC(BINPUT(131072))
K=SYSSLKWSET(INLOCK,ILOCK,)
TYPE *, ' INLOCK(1)= ',INLOCK(1),' INLOCK(2)= ',INLOCK(2)
TYPE *, ' ILOCK(1)= ',ILOCK(1),' ILOCK(2)= ',ILOCK(2)
IF(.NOT.K)TYPE *, ' UNABLE TO LOCK BUF'
C K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
C IBUF(1),XVAL(28),,,, )
IEVFO=4
IMAGEN=1
IBLOCK=1
AVACSR=0

```

[AVA.MAXDISK]MTODSK3A

```

CALL TIMRB
X=0
AVAACR='415'0
10 CALL FIELD(IFIELD,AVACSR)
IF(IFIELD.NE.0)GO TO 10
ICURR=IFIELD
C TYPE *,'ICURR=',ICURR
C
C GET IRIG
C
AVAACR=0
AVAACR='435'0
X='1000'0
Y=2
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK))),
1IOSB,,,
ID,XVAL(8),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IRIG(IMAGEN)=QD
C
C GO GET SECOND FIELD IN THIS FRAME
C
AVAACR='415'0
X=0
11 CALL FIELD(IFIELD,AVACSR)
IF(IFIELD.EQ.ICURR)GO TO 11
ISTOREFIELD=ICURR          ICURRENT FIELD TO PUT ON DISK
ICURR=IFIELD              ICURRENT FIELD BE LOADED INTO THE AVA
C TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C GO TO 11
ICOUNT=0
1 Y=Y+(ISTOREFIELD+1)
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK))),
1IOSB,,,
1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0) AVACSR=1
ISTATUS=SYSSQIO(XVAL(10$VFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK))),
1IOSB,,,
1BINPUT(1),XVAL(32768),XVAL(1BLOCK),,,)
1BLOCK=1BLOCK+64
Y=Y+32
ISADR=32769
2 ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK))),
1IOSB,,,
1BINPUT(ISADR),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
ISTATUS=SYSSQIO(XVAL(10$VFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK))),
1IOSB,,,
1BINPUT(ISADR),XVAL(32768),XVAL(1BLOCK),,,)
1BLOCK=1BLOCK+64
Y=Y+32
3 ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK))),
1IOSB,,,
1BINPUT(65537),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
ISTATUS=SYSSQIO(XVAL(10$VFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK))),
1IOSB,,,
1BINPUT(65537),XVAL(32768),XVAL(1BLOCK),,,)

```

[AVA.MAXDISK]MTODSK3A

```

        IBLOCK=IBLOCK+64
        Y=Y+32
4      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
        110SB,,,
        IBINPUT(983#5),XVAL(24576),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(10SWRITEVBLK)),
        110SB,,,
        IBINPUT(983#5),XVAL(24576),XVAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
        IF(1COUNT.EQ.0)THEN
        1COUNT=1
        1STOREFIELD=1
        GO TO 1
        ENDIF
C      TYPE*,IMAGEN/2,NIMAGES
        IF(IMAGEN.GE.NIMAGES)THEN
        TYPE *,IMAGEN, 'IMAGES WRITTEN TO DISK IN IMAGES.DAT'
        CALL TIMRE
        CALL HEADER(TITLE)
        ISTATUS=SYSSDASSGN(XVAL(IDISK))
        CLOSE(UNIT=3#)
C
C      PROCESS IRIG BUFFER
C
        OPEN(UNIT=9,NAME='DISK$AVA:[AVA]IRIGS.DAT',STATUS='NEW')
        DO IRIGX=1,IMAGEN
        QD=IRIG(IRIGX)
        DO I=2,4
        D(I)=NOT(D(I))
        ENDDO
        HMS=IAND(ISHFT(D(2),-8),'F'X)
        TMS=IAND(ISHFT(D(2),-4),'F'X)
        MS=IAND(D(2),'F'X)
        US=IAND(ISHFT(D(2),-12),'F'X)
        TS=IAND(D(3),7)
        UM=IAND(ISHFT(D(3),-3),'F'X)
        TM=IAND(ISHFT(D(3),-7),7)
        UH=IAND(ISHFT(D(3),-10),'F'X)
        TH=IAND(ISHFT(D(3),-14),3)
        UD=IAND(D(4),'F'X)
        TD=IAND(ISHFT(D(4),-4),'F'X)
        HD=IAND(ISHFT(D(4),-8),'F'X)
13      WRITE(9,13)HD,TD,UD,TH,UH,TM,UM,TS,US,HMS,TMS,MS
        FORMAT(3Z1,' ',1X,2Z1,' ',Z1,Z1,' ',2Z1,' ',3Z1)
        ENDDO
        STOP 'I/O COMPLETE.....'
        ENDIF
        IMAGEN=IMAGEN+1
        GO TO 1#
57     CONTINUE
        ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *, ' ISTATUS=',ISTATUS, ' 10SB(1)=' ,10SB(1)
        TYPE *, ' ISTATUS=',ISTATUS, ' 10SB(1)=' ,10SB(1)
        IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'

```

LAJA.MAXDISKIMTODSK3A

```

TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF= ' '
ISTATUS=SYSSGETMSG (XVAL(IOB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
STOP
C11  FORMAT(1X, 'INPUT=', 06, 2X, 'IOB=', 06, 2X, 06, 2X, 06, 2X, 06)
C    K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC( IOSWRITEVBLK)), IOB,...
C    ISETUP3, XVAL(4), ...,)
END
SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
BYTE BINPUT(1), BYTE(2)
INTEGER*2 OUT(513, 1), BYTES, SLU
EQUIVALENCE(BYTES, BYTE)
DATA SLU/'34011'O/
I=0
IOLINE=1
DO 100 IX=1, NUMB
I=I+1
IF(I.EQ.512) THEN
BYTE(1)=BINPUT(IX)
OUT(I, IOLINE)=BYTES
C    WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
C    OUT(I+1, IOLINE)=SLU
C    WRITE(6, 34) I+1, IOLINE, OUT(I+1, IOLINE)
I=0
IOLINE=IOLINE+1
GO TO 100
ENDIF
BYTE(1)=BINPUT(IX)
OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C    WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
C34  FORMAT(1X, I3, 1X, I3, 2X, 06)
100  CONTINUE
RETURN
END

SUBROUTINE FIELD(IFIELD, AVACSR)
INTEGER AVACSR
EXTERNAL IOSWRITEVBLK, IOSREADVBLK
INTEGER SYSSASSIGN, SYSSQIOW, SYSSQIO
INTEGER SYSSGETMSG
INTEGER*2 IOB(4), MSGLEN, NPUT, X, Y
INTEGER*2 INPUT, OUTPUT, INIT(4)
CHARACTER *80 MSGBUF
COMMON/AVACHAN/ITCHAN
DATA IFIRST/1/
ISAVE=AVACSR
IF(IFIRST) THEN
AVACSR='4000'O ISET MEMORY WINDOW ENABLE AND INITIALIZE AVA
IFIRST=0
ELSE
AVACSR='4001'O
ENDIF
ISTATUS=SYSSQIOW(XVAL(1), XVAL(ITCHAN), XVAL(XLOC( IOSREADVBLK)),

```

[AVA.MAXDISK]MTODSK3A

```
      IOSB...
      IOUTPUT,XVAL(2),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(IAVAACR))
C      IF(AVACSR.EQ.'4000'0)AVACSR='4001'0
C      IF(ISTATUS)      GO TO 501
C      TYPE *, ' ERROR IN QIOW CALL'
C      ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
C      MSGBUF=' '
C      ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'I/O STATUS:',MSGBUF
501    AVACSR=ISAVE
      IFIELD=IAND(OUTPUT,3)
      RETURN
      END
```

APPENDIX D  
[AVA.MAXDISK]MTODSK4

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      THIS PROGRAM READS THE CURRENT AVA FRAME FIELD AND WRITES IT
C      IT TO DISK$IMAGES:[AVA]IMAGES.DAT. THE NEXT FIELD IS THEN WRITTEN
C      TO DISK ALSO IF THERE IS ENOUGH TIME ELSE A FIELD IS SKIPPED AND THE
C      PROCESS CONTINUES.
C
C      THIS PROGRAM IS DIFFERENT FROM MTODSK3 IN THAT IT TRANSFERS THE
C      CURRENT FIELD AND FOLLOWING FIELDS WHERE AS MTODSK3 ALWAYS TRANSFERS
C      FIELD # THEN FIELD 1 AND DOES NOT ATTEMPT TO TRANSFER FIELDS 2 OR 3.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK,MITLS1
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER    SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 X,Y,YA(4),SYSS$DASSGN
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(65536)
C
BYTE BINPUT(32768)
BYTE BINPUT(131072)
INTEGER AVACSR,AVAACR,SYSSLKWSET,INLOCK(2),IOLOCK(2)
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF
CHARACTER*60 TITLE,FNAME*60
CHARACTER*60 NAME
INTEGER*2 BUFFERL,DEVCODE
INTEGER SYSS$GETDVI,DVIS$FREEBLOCKS
INTEGER IFREE
INTEGER BUFFERA,ZERO
COMMON/PRACHAN/IDISK
COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
COMMON/AVACHAN/ITCHAN
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
DATA YA/6,'206'O,'406'O,'606'O/
DATA DVIS$FREEBLOCKS/'0000002A'X/,ZERO/0/
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,

```

[AVA.MAXDISK]MTODSK4

```

2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
NAME='DISK$AVA:'
BUFFERL=4
DEVCODE=DVISFREEBLOCKS
BUFFERA=XLOC(IFREE)
C RETURNLA=XLOC(RETURNL)
ISTATUS=SYSSGETDVI(XVAL(3),,NAME,BUFFERL,,,)

IF(.NOT. ISTATUS)TYPE*, 'PARAMETER ERROR IN GETDVI'
ISTATUS=SYSSWAITFR(XVAL(3))
TYPE *, 'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
MAXIMAGES=IFREE/513
TYPE *, 'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C MAXIMAGES=30      !THIS IS FOR DEGUG ONLY
C
NIMAGES=MAXIMAGES
7775 INSZ=NIMAGES*481
FNAME='DISK$AVA:[AVA]IMAGES.DAT'
TYPE *, 'OPENING',FNAME
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4096,ERR=777)
GO TO 776
777 NIMAGES=NIMAGES-10
IF(NIMAGES.LT.0)STOP 'NIMAGES LESS THAN ZERO!!!!'
GO TO 7775
776 TYPE*, 'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
ISTATUS=SYSSASSIGN('AVA0',AVACHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
INLOCK(1)=XLOC(BINPUT(1))
INLOCK(2)=XLOC(BINPUT(131072))
K=SYSSLKWSET(INLOCK,IOLOCK,)
TYPE *, ' INLOCK(1)= ',INLOCK(1), ' INLOCK(2)= ',INLOCK(2)
TYPE *, ' IOLOCK(1)= ',IOLOCK(1), ' IOLOCK(2)= ',IOLOCK(2)
IF(.NOT.K)TYPE *, ' UNABLE TO LOCK BUF'
AVAACR='415'O
C K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),IOSB,,,
C 1BUF(1),XVAL(20),,,)
IEVFO=4
IMAGEN=1
IBLOCK=1
AVACSR=0
CALL TIMRB
X=0
CALL FIELD(IFIELD,AVACSR)
ICURR=IFIELD
GO TO 11

```

[AVA.MAXDISK]MTODSK4

```

10 CALL FIELD(IFIELD,AVACSR)
   IF(IFIELD.NE.ICURR)GO TO 10
   ICURR=IFIELD
C   TYPE *,'ICURR=',ICURR
11 CALL FIELD(IFIELD,AVACSR)
   IF(IFIELD.EQ.ICURR)GO TO 11
   ISTOREFIELD=ICURR          ICURRENT FIELD TO PUT ON DISK
   ICURR=IFIELD              ICURRENT FIELD BE LOADED INTO THE AVA
C   TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C   GO TO 11
   ICOUNT=0
1   ISTOREF=ISTOREFIELD+1
   Y=YA(ISTOREF)
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...
   IBINPUT(XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
   IF(AVACSR.EQ.0) AVACSR=1
   ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...
   IBINPUT(1),XVAL(32768),XVAL(IBLOCK),...)
   IBLOCK=IBLOCK+64
   Y=Y+32
2   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...
   IBINPUT(32679),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
   ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...
   IBINPUT(32679),XVAL(32768),XVAL(IBLOCK),...)
   IBLOCK=IBLOCK+64
   Y=Y+32
3   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...
   IBINPUT(65537),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
   ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...
   IBINPUT(65537),XVAL(32768),XVAL(IBLOCK),...)
   IBLOCK=IBLOCK+64
   Y=Y+32
4   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...
   IBINPUT(98305),XVAL(24576),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
   ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...
   IBINPUT(98305),XVAL(24576),XVAL(IBLOCK),...)
   IBLOCK=IBLOCK+48
   IF(ICOUNT.EQ.0)THEN
       ICOUNT=1
       ISTOREFIELD=ISTOREFIELD+1
       IF(ISTOREFIELD.GT.3)ISTOREFIELD=0
       GO TO 1
   ENDIF
   IF(IMAGEN.GE.NIMAGES)THEN
       TYPE *,'I/O COMPLETE.....'
       TYPE *,IMAGEN,' IMAGES WRITTEN TO DISKSAVA:[AVA]IMAGES.DAT'
       CALL TIMRE

```

[AVA.MAXDISK]MTODSK4

```

        CALL HEADER(TITLE)
        ISTATUS=SYSSDASSGN(XVAL(IDISK))
        CLOSE(UNIT=30)
        STOP 'IMAGE WRITTEN TO DISK'
        ENDIF
    IMAGEN=IMAGEN+1
    GO TO 10
57  CONTINUE
    ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
    TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
    TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
    MSGBUF=' '
    ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *, 'I/O STATUS:', MSGBUF
    STOP
C11  FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C    K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(%LOC(IOSWRITEVBLK)), IOSB,,)
C    I1SETUP3, XVAL(4),,,)
    END
    SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
    BYTE BINPUT(1), BYTE(2)
    INTEGER*2 OUT(513,1), BYTES, SLU
    EQUIVALENCE(BYTES, BYTE)
    DATA SLU/'34011'0/
    I=0
    IOLINE=1
    DO 100 IX=1, NUMB
    I=I+1
    IF(I.EQ.512) THEN
    BYTE(1)=BINPUT(IX)
    OUT(I, IOLINE)=BYTES
C    WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
C    OUT(I+1, IOLINE)=SLU
C    WRITE(6,34) I+1, IOLINE, OUT(I+1, IOLINE)
    I=0
    IOLINE=IOLINE+1
    GO TO 100
    ENDIF
    BYTE(1)=BINPUT(IX)
    OUT(I, IOLINE)=IAND(NOT(BYTES), '377'0)
C    WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
34  FORMAT(1X, I3, 1X, I3, 2X, 06)
100  CONTINUE
    RETURN
    END

    SUBROUTINE FIELD(IFIELD, AVACSR)
    INTEGER AVACSR
    EXTERNAL IOSWRITEVBLK, IOSREADVBLK
    INTEGER SYSSASSIGN, SYSSQIOW, SYSSQIO
    INTEGER SYSSGETMSG
    INTEGER*2 IOSB(4), MSGLEN, NPUT, X, Y

```

[AVA.MAXDISK]MTODSK4

```

      INTEGER*2 INPUT,OUTPUT,INIT(4)
      CHARACTER *8 MSGBUF
      COMMON/AVACHAN/ITCHAN
      DATA IFIRST/1/
      ISAVE=AVACSR
      IF(IFIRST)THEN
      AVACSR='4888'0  ISET MEMORY WINDOW ENABLE AND INITIALIZE AVA
      IFIRST=8
      ELSE
      AVACSR='4881'0
      ENDIF
      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      IOSB,...
      IOUTPUT,XVAL(2),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(IAVAACR))
C      IF(AVACSR.EQ.'4888'0)AVACSR='4881'0
C      IF(ISTATUS) GO TO 581
C      TYPE *,' ERROR IN QIOW CALL'
C      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C      TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C      MSGBUF=' '
C      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C      TYPE *,'I/O STATUS:',MSGBUF
581  AVACSR=ISAVE
      IFIELD=IAND(OUTPUT,3)
      RETURN
      END
```

APPENDIX E  
[AVA.MAXDISKIMDSKTOGRN

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND DISPLAYS
C   THE IMAGE ON THE GRINNELL.
C
C   THIS IS THE COMPACT IMAGE FORMAT USED WHEN TRYING TO OUTPUT THE AVA
C   IMAGE AS FAST AS POSSIBLE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK,MITLSI
INTEGER*2 BUF(288),ISETUP(14),SLU,IOSB(4)
INTEGER    SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2),IIMAGEB(4)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
BYTE BINPUT(32768)
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF
CHARACTER*60 TITLE,FNAME
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
COMMON/PRACHAN/IDISK
DATA ISETUP/'128848'0,'148881'0,'121888'0,'187777'0,'17777'0,
1 '24861'0,'26882'0,'38888'0,'44888'0,'64777'0,'128888'0,
2 '58881'0,'78776'0,'54888'0/
DATA ISETUP2/'64777'0,'44888'0/
DATA ISETUP3/'64776'0,'44888'0/
I = SYSS$ASSIGN('GRA8',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('AVA8',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=8
AVAACR='415'0
K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(%LOC(IOSWRITEVBLK)),IOSB,..
IBUF(1),XVAL(28),...,)

```

[AVA.MAXDISKIMDSKTOGRN

```

TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
NIMAGES=1
INSZ=NIMAGES*513
FNAME='DISK$AVA:[AVA]IMAGES.DAT'
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4096)
IEVFO=4
Y=6
X=0
TYPE *, 'ENTER STARTING IMAGE NUMBER DESIRED'
ACCEPT*,IBLOCK
IF(IBLOCK.EQ.1)THEN
IBLOCK=1
ISTARTI=1
ELSE
ISTARTI=IBLOCK
IBLOCK=1+(240*(IBLOCK-1))
ENDIF
TYPE *, 'ENTER IMAGE INCREMENT'
ACCEPT*,IBLOCKI
ICOUNT=0
C CALL TIMRB
TYPE *, 'ENTER NUMBER OF IMAGES TO DISPLAY'
ACCEPT*,IFIELDS
C TYPE *,IFIELDS
IFIELDS=IFIELDS*8
1 IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
NBYTES=24576
ELSE
NBYTES=32768
ENDIF
ISTATUS=SYSSQIOW(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(IOSREADVBLK)),
IOSB,,,
IBINPUT(1),XVAL(NBYTES),XVAL(IBLOCK),,,)
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
IBLOCK=IBLOCK+48
NUMB=24576
ELSE
NUMB=32768
IBLOCK=IBLOCK+64
ENDIF
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C IOSB,,,
C IBINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IBINPUT,XVAL(30720),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C
C WRITE (4,54)BINPUT
54 FORMAT(1X,16(1X,03))
C
C CALL BUFCNVT(NUMB,BINPUT,OUT)
TYPE *, 'NUMBER OF LINES TO OUTPUT=',IOLINE
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN

```

[AVA.MAXDISK]MDSKTOGRN

```

IGBYTES=24624
ELSE
IGBYTES=32832
ENDIF

ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
1BOUT(1),XVAL(IGBYTES),,,,
ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
1BOUT(IGBYTES+1),XVAL(IGBYTES),,,,
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
Y=Y+32
ICOUNT=ICOUNT+1
IF(ICOUNT.EQ.4)THEN
K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
11SETUP3(1),XVAL(4),,,,
Y='206'O
X=0
ENDIF
IF(ICOUNT.EQ.8)THEN
IIMAGEB(1)=27
IIMAGEB(2)=89
IIMAGEB(3)=55
IIMAGEB(4)=40
77 WRITE(6,7)IIMAGEB,ISTARTI
FORMAT(1H+,4A1,'IMAGE NUMBER',I5, ' DISPLAYED ON THE GRINNELL NOW.')
ICOUNT=0
ISTARTI=ISTARTI+IBLOCKI
IBLOCK=IBLOCK+(240*(IBLOCKI-1))
K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
11SETUP2(1),XVAL(4),,,,
C CALL TIMRE
C CALL HEADER(TITLE)
C STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
Y=6
ENDIF
IFIELDS=IFIELDS-1
IF(IFIELDS.EQ.0)STOP
GO TO 1
57 CONTINUE
ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:',MSGBUF
STOP
C11 FORMAT(1X,'INPUT=',.06,2X,'IOSB=',.06,2X,.06,2X,.06,2X,.06)
C K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
C 11SETUP3,XVAL(4),,,,
END

```

CAVA.MAXDISK]MDSKTOGRN

```

SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
  BYTE BINPUT(1),BYTE(2)
  INTEGER*2 OUT(513,1),BYTES,SLU
  EQUIVALENCE(BYTES,BYTE)
  DATA SLU/'34011'O/
  I=0
  IOLINE=1
  DO 100 IX=1,NUMB
    I=I+1
    IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES
    C   WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
    C   OUT(I+1,IOLINE)=SLU
    C   WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
    I=0
    IOLINE=IOLINE+1
    GO TO 100
  ENDIF
  BYTE(1)=BINPUT(IX)
  OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
  C   WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
  34  FORMAT(1X,13,1X,13,2X,06)
  100 CONTINUE
  RETURN
  END
```

## APPENDIX F

[illegible]

THIS PROGRAM READS THE DISKSIMAGES:[AVA]IMAGES.DAT FILE AND DISPLAYS THE IMAGE ON THE GRINNELL A FRAME AT A TIME.

THIS IS THE COMPACT IMAGE FORMAT USED WHEN TRYING TO OUTPUT THE AVA  
IMAGE AS FAST AS POSSIBLE.

[illegible]

```

EXTERNAL IOSWRITEVBLK,IOSREADVBLK,MITLS1
INTEGER*2 BUF(288),ISETUP(14),SLU,IOSB(4)
INTEGER SYSS$ASSIGN,SYSS$QIOW,CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,488),X,Y
BYTE BOUT(492488),BYTE(2),IIMAGEB(4)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
BYTE BINPUT(32768),BLINES(512,64)
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER*88 MSGBUF
CHARACTER*68 TITLE,FNAME
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES),(BINPUT,BLINES)
COMMON/PRACHAN/IDISK
DATA ISETUP/'128848'0,'148888'0,'121888'0,'187777'0,'17777'0,
1 '24861'0,'268882'0,'38888'0,'448888'0,'64777'0,'128888'0,
2 '588881'0,'78777'0,'54888'0/
DATA ISETUP2/'64777'0,'44888'0/
DATA ISETUP3/'64776'0,'44888'0/
I = SYSS$ASSIGN('GRA8',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('AVA8',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=8
AVAACR='415'0
K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB
1BUF(1),XVAL(28),...)

```

[AVA.MAXDISK]MDTOGRN

```

TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
NIMAGES=1
INSZ=NIMAGES*513
FNAME='DISK$AVA:[AVA]IMAGES.DAT'
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4096)
IEVFO=4
Y=6
X=0
TYPE *, 'ENTER STARTING IMAGE NUMBER DESIRED'
ACCEPT*,IBLOCK
IF(IBLOCK.EQ.1)THEN
IBLOCK=1
ISTARTI=1
ELSE
ISTARTI=IBLOCK
IBLOCK=1+(400*(IBLOCK-1))
C 400 BLOCKS CONTAINS ONE FULL IMAGE OR TWO FIELDS
ENDIF
TYPE *, 'ENTER IMAGE INCREMENT'
ACCEPT*,IBLOCKI
ICOUNT=0
C CALL TIMRB
TYPE *, 'ENTER NUMBER OF IMAGES TO DISPLAY'
ACCEPT*,IFIELDS
C TYPE *,IFIELDS
IFIELDS=IFIELDS*8
1 IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
NBYTES=24576
ELSE
NBYTES=32768
ENDIF
ISTATUS=SYSSQIOW(XVAL(1),XVAL(IDISK),XVAL(XLOC(10$READVBLK)),
1IOSB,,
1BINPUT(1),XVAL(NBYTES),XVAL(IBLOCK),,,)
IF(.NOT.ISTATUS)TYPE *, 'GIO PARAMETER ERROR ON DISK READ'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN DISK INPUT'
C TYPE *, 'IBLOCK=',IBLOCK
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
IBLOCK=IBLOCK+48
NUMB=24576
ELSE
NUMB=32768
IBLOCK=IBLOCK+64
ENDIF
C ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
C 1IOSB,,
C 1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C 1INPUT,XVAL(30720),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C
C WRITE (4,54)BINPUT
54 FORMAT(1X,16(1X,O3))

```

[AVA.MAXDISK]MDTOGRN

```

C      IF (ICOUNT.EQ.0)THEN
C      DO I=2,12
C      WRITE (6,155)(BLINES(J,I),J=8,16)
C      ENDDO
C      ENDIF
155    FORMAT(1X,10(1X,03))
189    CALL BUFCNV(T(NUB,BINPUT,OUT,ICOUNT)
C      TYPE *,(OUT(I,1),I=1,20)
C      TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C      IF(ICOUNT.EQ.7)THEN
C      K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
11SETUP2(1),XVAL(4),,,)
C      DO IQIO=1,7
C      IADDR=1+(65534*(IQIO-1))
C      ISTATUS = SYSSQIOW(XVAL(1),XVAL(CHAN),
1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
1BOUT(IADDR),XVAL(65534),,,)
C      ENDDO
C      IADDR=1+(65534*(IQIO-1))
C      ISTATUS = SYSSQIOW(XVAL(1),XVAL(CHAN),
1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
1BOUT(IADDR),XVAL(33742),,,)
C      ENDIF
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C      Y=Y+32
C      ICOUNT=ICOUNT+1
C      IF(ICOUNT.EQ.4)THEN
C      K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
C      11SETUP3(1),XVAL(4),,,)
C      Y='206'0
C      X=0
C      ENDIF
C      IF(ICOUNT.EQ.8)THEN
C      IIMAGEB(1)=27
C      IIMAGEB(2)=89
C      IIMAGEB(3)=55
C      IIMAGEB(4)=40
C      WRITE(6,77)IIMAGEB,ISTARTI
77    FORMAT(1H+,4A1,'IMAGE NUMBER',15, ' DISPLAYED ON THE GRINNELL NOW.')
C      ICOUNT=0
C      ISTARTI=ISTARTI+IBLOCKI
C      IBLOCK=IBLOCK+(480*(IBLOCKI-1))
C      480 BLOCKS CONTAINS ONE FULL IMAGE OR TWO FIELDS
C      CALL TIMRE
C      CALL HEADER(TITLE)
C      STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
C      Y=6
C      ENDIF
C      IFIELDS=IFIELDS-1
C      IF(IFIELDS.EQ.0)STOP
C      GO TO 1
57    CONTINUE
C      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C      TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)

```

[AVA.MAXDISK]MDTOGRN

```

TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF= ' '
ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
STOP
C11  FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C    K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC(IOSWRITEVBLK)), IOSB,,
C    1 ISETUP3, XVAL(4), , , ,)
      END
SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT, ICOUNT)
  BYTE BINPUT(1), BYTE(2)
  INTEGER*2 OUT(513, 1), BYTES, SLU
  EQUIVALENCE(BYTES, BYTE)
  DATA SLU/'34811'O/
  I=0
  IF(ICOUNT.EQ.0) IOLINE=1
  IF(ICOUNT.EQ.1) IOLINE=129
  IF(ICOUNT.EQ.2) IOLINE=257
  IF(ICOUNT.EQ.3) IOLINE=385
  IF(ICOUNT.EQ.4) IOLINE=2
  IF(ICOUNT.EQ.5) IOLINE=138
  IF(ICOUNT.EQ.6) IOLINE=258
  IF(ICOUNT.EQ.7) IOLINE=386

C    TYPE *, ' IOLINE=', IOLINE, ICOUNT
      DO 100 IX=1, NUMB
        I=I+1
        IF(I.EQ.512) THEN
          BYTE(1)=BINPUT(IX)
          OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C        WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
          OUT(I+1, IOLINE)=SLU
C        WRITE(6, 34) I+1, IOLINE, OUT(I+1, IOLINE)
          I=0
          IOLINE=IOLINE+2
          GO TO 100
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C        WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
C34  FORMAT(1X, I3, 1X, I3, 2X, 06)
100  CONTINUE
      RETURN
      END

```

## [AVA,MAXDISK]FIELD\$GRN

[illegible]

[AVA.MAXDISK]FIELDSCRN

```

TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
NIMAGES=1
INSZ=NIMAGES*513
FNAME='DISKSAVA:[AVA]IMAGES.DAT'
OPEN(UNIT=38,NAME=FNAME,TYPE='UNKNOWN',
1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4896)
IEVFO=4
Y=6
X=8
TYPE *, 'ENTER STARTING IMAGE NUMBER DESIRED'
ACCEPT*,IBLOCK
IF(IBLOCK.EQ.1)THEN
IBLOCK=1
ISTARTI=1
ELSE
ISTARTI=IBLOCK
IBLOCK=1+(248*(IBLOCK-1))
ENDIF
TYPE*, 'ENTER FIELD INCREMENT'
ACCEPT*,IBLOCKI
ICOUNT=8
C CALL TIMRB
TYPE *, 'ENTER NUMBER OF IMAGES TO DISPLAY'
C ACCEPT*,IFIELDS
TYPE *,IFIELDS
IFIELDS=IFIELDS*8
1 IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
NBYTES=24576
ELSE
NBYTES=32768
ENDIF
ISTATUS=SYSSQIO(XVAL(IEVFO),XVAL(IDISK),XVAL(XLOC(10SREADVBLK)),
1IOSB,...
1BINPUT(1),XVAL(NBYTES),XVAL(IBLOCK),...)
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
IBLOCK=IBLOCK+48
NUMB=24576
ELSE
NUMB=32768
IBLOCK=IBLOCK+64
ENDIF
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK))),
C 1IOSB,...
C 1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C 1INPUT,XVAL(38728),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IF(AVACSR.EQ.8)AVACSR=1
C IF(.NOT.1STATUS.OR..NOT.10SB(1))GO TO 57

C WRITE (4,54)BINPUT
54 FORMAT(1X,16(1X,03))

C CALL BUFFCNVT(NUMB,BINPUT,OUT)
TYPE *, 'NUMBER OF LINES TO OUTPUT=',IOLINE
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN

```

[AVA.MAXDISK]FIELDSEGRN

```

    IGBYTES=24624
    ELSE
    IGBYTES=32832
    ENDIF

    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
    1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
    1BOUT(1),XVAL(IGBYTES),,,,
    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
    1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
    1BOUT(IGBYTES+1),XVAL(IGBYTES),,,,
C    IF(.NOT. ISTATUS.OR..NOT.IOSB(1))GO TO 57
    Y=Y+32
    ICOUNT=ICOUNT+1
    IF(ICOUNT.EQ.4)THEN
    IIMAGEB(1)=27
    IIMAGEB(2)=89
    IIMAGEB(3)=55
    IIMAGEB(4)=48
    WRITE(6,77)IIMAGEB,ISTARTI
    K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
    11SETUP3(1),XVAL(4),,,,
    ISTARTI=ISTARTI+IBLOCKI
    Y='286'O
    X=8
    ENDIF
    IF(ICOUNT.EQ.8)THEN
    IIMAGEB(1)=27
    IIMAGEB(2)=89
    IIMAGEB(3)=55
    IIMAGEB(4)=48
    WRITE(6,77)IIMAGEB,ISTARTI
77    FORMAT(1H+,4A1,'FIELD NUMBER',15, ' DISPLAYED ON THE GRINNELL NOW.')
    ICOUNT=8
    ISTARTI=ISTARTI+IBLOCKI
    IBLOCK=IBLOCK+(248*(IBLOCKI-1))
    K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
    11SETUP3(1),XVAL(4),,,,
C    CALL TIMRE
C    CALL HEADER(TITLE)
C    STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
    Y=6
    ENDIF
    IFIELDS=IFIELDS-1
    IF(IFIELDS.EQ.8)STOP
    GO TO 1
57    CONTINUE
    ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
    TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=',IOSB(1)
    TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=',IOSB(1)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
    TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
    MSGBUF=' '
    ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'

```

[AVA.MAXDISK]FIELDSEGRN

```

      TYPE *, 'I/O STATUS:', MSGBUF
      STOP
C11   FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C     K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC(IOSWRITEVBLK)), IOSB,...
C     I1SETUP3, XVAL(4), ...,)
      END
      SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
      BYTE BINPUT(1), BYTE(2)
      INTEGER*2 OUT(513, 1), BYTES, SLU
      EQUIVALENCE(BYTES, BYTE)
      DATA SLU/'34811'O/
      I=8
      IOLINE=1
      DO 188 IX=1, NUMB
      I=I+1
      IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C     WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
      OUT(I+1, IOLINE)=SLU
C     WRITE(6, 34) I+1, IOLINE, OUT(I+1, IOLINE)
      I=8
      IOLINE=IOLINE+1
      GO TO 188
      ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C     WRITE(6, 34) I, IOLINE, OUT(I, IOLINE)
C34   FORMAT(1X, I3, 1X, I3, 2X, 06)
188   CONTINUE
      RETURN
      END

```

APPENDIX H  
[AVA.MAXDISK]MDSKTOFIL

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND GENERATES
C   NATO FORMATTED DISK FILES FOR AS MANY FILES AS SPECIFIED BY THE USER.
C
C   IMAGES.DAT IS THE COMPACT IMAGE FORMAT.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK,MITLS1,MITLS2
INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*4 LIB$FREEVM,LIB$GETVM,SYSS$DASSGN
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
BYTE BINPUT(32768)
INTEGER AVACSR,AVAACR,OTSS$CVTLTI
INTEGER*2 ISETUP2(2),ISETUP3(2)
C   CHARACTER *80 MSGBUF,NAMNUM*4,DISKSPEC*14
CHARACTER *80 MSGBUF,NAMNUM*4,DISKSPEC*22
CHARACTER*60 TITLE,FNAME,FNAM2
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
COMMON/PRACHAN2/IDISK2
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
DATA IFIRST/1/

```

CAVA.MAXDISKIMDSKTOFIL

```

I=SYSSASSIGN('TT',IVTC,,)
IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='415'0
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
IBUF(1),XVAL(2B),,,,))
TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
NIMAGES=1
INSZ=NIMAGES*513
FNAME='DISKSAVA:[AVA]IMAGES.DAT'
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS2,
ZRECORDTYPE='FIXED',RECORDSIZE=4096)
IEVFO=4
Y=6
X=0
TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
ACCEPT*,IBLOCK
IF(IBLOCK.EQ.1)THEN
IBLOCK=1
ELSE
IBLOCK=1+(400*(IBLOCK-1))
ENDIF
ICOUNT=0
C CALL TIMRB
TYPE *,'ENTER NUMBER IMAGES TO STORE'
ACCEPT*,IFIELDS
I IFIELDS=IFIELDS*8 I CONVERT IMAGES TO TRANSFERS
1 IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
NBYTES=24576
ELSE
NBYTES=32768
ENDIF
ISTATUS=SYSSQIOW(XVAL(IEVFO),XVAL(IDISK2),XVAL(XLOC(IOSREADVBLK)),
IOSB,,,
IBINPUT(1),XVAL(NBYTES),XVAL(IBLOCK),,,)
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
IBLOCK=IBLOCK+48
NUMB=24576
ELSE
NUMB=32768
IBLOCK=IBLOCK+64
ENDIF
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C IOSB,,,
C IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IINPUT,XVAL(30720),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C WRITE (4,54)BINPUT

```

[AVA.MAXDISK]MDSKTOFIL

```
54      FORMAT(1X,16(1X,03))

C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
      TYPE *, 'NUMBER OF LINES TO OUTPUT=', IOLINE
      IF(ICOOUNT.EQ.3.OR.ICOOUNT.EQ.7)THEN
        IGBYTES=24624
      ELSE
        IGBYTES=32832
      ENDIF

      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
        IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
        IBOUT(1),XVAL(IGBYTES),,,,
        ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
        IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
        IBOUT(IGBYTES+1),XVAL(IGBYTES),,,,
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
      IF(IFIRST)THEN
C      TYPE *, 'ENTER DISK NAME AND DIRECTORY. (DISK$IMAGES:[WILLIAMS])'
C      READ(5,6)DISKSPEC
      DISKSPEC='DISK$IMAGES:[WILLIAMS]'
C      DISKSPEC='DISK$AVA:[AVA]'
      TYPE *, 'ENTER FIRST IMAGE FILE NAME. (W00010000.IMG)'
      READ(5,6)FNAM2
C      FORMAT(A)
      FNAM='SIMS00001.IMG'
      NAMNUM=FNAM2(2:5)
      FNAM=DISKSPEC//FNAM2
      DECODE(4,101,NAMNUM)INAMNUM
101     FORMAT(I4)
      NCOL=512
      NROW=480
      ILEN=NROW
      IWD=NCOL
C      IMGMAPC(3)=ILEN ILENGTH OF IMAGE
C      IMGMAPC(4)=IWD IWIDTH OF IMAGE
      IFIRST=0
      I=IWD
      NBYT=(I+1)*ILEN*2
      I=LIB$GETVM(NBYT,IMGADR)
      IF(.NOT.I)TYPE *, ' ERROR IN VIRTUAL MEMORY ASSIGNMENT 1'
      I = LIB$GETVM(10000,HDR2ADR)
      IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
      ENDIF
      HEAD(8)= '          1'          IONE CHARACTER PER CHANNEL
      HDR2LEN=576
      CURRENTNUMFL=0
      CALL ADDHDR2(XVAL(HDR2ADR))
      CALL IMGTODISK(BINPUT,NUMB,XVAL(IMGADR),ICOOUNT)
      IF(ICOOUNT.EQ.7)THEN
C      FNAM='SIMS00001.IMG'
      INAMNUM=INAMNUM+1
      ISTATUS=OTSSCVTLTI(INAMNUM,NAMNUM,XVAL(4),XVAL(4),)
      IF(.NOT.ISTATUS)TYPE *, 'CONVERSION ERROR IN FILE NAME'
      FNAM2(2:5)=NAMNUM
```

[AVA,MAXDISK]MDSKTOFIL

```

FNAME=DISKSPEC//FNAME2
ENDIF
Y=Y+32
ICOUNT=ICOUNT+1
IF(ICOUNT.EQ.4)THEN
K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
ISETUP3(1),XVAL(4),,...)
Y='206'O
X=0
ENDIF
IF(ICOUNT.EQ.8)THEN
ICOUNT=0
K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
ISETUP2(1),XVAL(4),,...)
C CALL TIMRE
C CALL HEADER(TITLE)
C STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
Y=6
ENDIF
IFIELDS=IFIELDS-1
IF(IFIELDS.EQ.0)STOP 'ALL IMAGES WRITTEN TO DISK'
GO TO 1
57 CONTINUE
ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:',MSGBUF
STOP
C11 FORMAT(1X,'INPUT=',06,2X,'IOSB=',06,2X,06,2X,06,2X,06)
C K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C ISETUP3,XVAL(4),,...)
END
SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
BYTE BINPUT(1),BYTE(2)
INTEGER*2 OUT(513,1),BYTES,SLU
EQUIVALENCE(BYTES,BYTE)
DATA SLU/'34011'O/
I=0
IOLINE=1
DO 100 IX=1,NUMB
I=I+1
IF(I.EQ.512)THEN
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=BYTES
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
OUT(I+1,IOLINE)=SLU
C WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
I=0
IOLINE=IOLINE+1
GO TO 100

```

[AVA.MAXDISK]MDSKTOFIL

```

ENDIF
BYTE(1)=BINPUT(IX)
OUT(1,IOLINE)=IAND(NOT(BYTES),'377'O)
C   WRITE(6,34) 1,IOLINE,OUT(1,IOLINE)
34  FORMAT(1X,I3,1X,I3,2X,O6)
188 CONTINUE
RETURN
END

SUBROUTINE IMGTODISK(BINPUT,NUMB,IMAGE,ICOUNT)
INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.C.IN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
INTEGER*2 IMAGE(NCOL+1,NROW)
INTEGER*4 IMGADR,SYSSASSIGN,IMGADR2,SYSSGETMSG
BYTE BINPUT(1)
INDEX=1
IF(ICOUNT.LE.3)THEN
  ISTART=1+(ICOUNT*128)
  IF(ICOUNT.LE.2)IEND=ISTART+126
  IF(ICOUNT.EQ.3)IEND=ISTART+94
  DO I=ISTART,IEND,2
    DO J=1,NCOL
      INPUT=BINPUT(INDEX)
      IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C     IMAGE(J,I)=BINPUT(INDEX)
      INDEX=INDEX+1
    ENDDO
  ENDDO
ELSE
  ISTART=2+((ICOUNT-4)*128)
  IF(ICOUNT.LE.6)IEND=ISTART+126
  IF(ICOUNT.EQ.7)IEND=ISTART+94
  DO I=ISTART,IEND,2
    DO J=1,NCOL
      INPUT=BINPUT(INDEX)
      IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C     IMAGE(J,I)=BINPUT(INDEX)
      INDEX=INDEX+1
    ENDDO
  ENDDO
ENDIF
ILEN=NROW
IWD=NCOL
C   TYPE*,'IWD AND ILEN BEFORE TODISK=',IWD,ILEN
  IF(ICOUNT.EQ.7) CALL TODISK(IMAGE,IWD,ILEN)

C   WRITE(6,1)ICOUNT
1   FORMAT(1X,I18)
RETURN
END
SUBROUTINE TODISK(IMAGE,IWD,ILEN)

```

[AVA.MAXDISK]MDSKTOFIL

```

INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
INTEGER*2 IMAGE(NCOL+1,NROW),HDR2LEN
INTEGER*4 AUTOWRTS8
CHARACTER*10000 HDR2ADR
CHARACTER*3 MONTH,DAY,YEAR*2,WD*8,LEN*8,TIMEA*8
CHARACTER*5 IFIRST5,ILAST4*4,TNAME*9
HDR2LEN=HDR2LEN
C TYPE *,'HDR2LEN',HDR2LEN
C TYPE *,'HDR2LEN ',HDR2LEN
CALL CNVRT(XVAL(HDR2ADR),HDR2LEN,HDRZADR)
C TYPE *,HEAD
CALL IDATE(IMONTH,IDAY,IYEAR)
ENCODE(3,200,MONTH)IMONTH
ENCODE(3,200,DAY )IDAY
200 FORMAT(I3)
ENCODE(2,100,YEAR )IYEAR
HEAD(3)='OLDFAAD '
HEAD(1)='USAMICOM'
HEAD(2)=YEAR//MONTH//DAY
100 FORMAT(I2)
ENCODE(8,200,WD)IWD
HEAD(11)(6:8)=WD(1:3)
ENCODE(8,200,LEN)ILEN
HEAD(12)(6:8)=LEN(1:3)
C TYPE *,HEAD
IBRACKET=INDEX(FNAM,'I')
IPERIOD=INDEX(FNAM,'.')
TNAME=FNAM(IPERIOD-9:IPERIOD-1)
IBRACKET=IBRACKET
IF(IPERIOD-10.LT.IBRACKET)THEN
IZERO=ABS(IPERIOD-10)
TNAME(1:IZERO)=' '
ENDIF
ILAST4=TNAME(6:9)
IFIRST5=TNAME(1:5)

HDR2ADR(1:8)= 'FN*X0000'
HDR2ADR(11:18)='0000.IMG'
HDR2ADR(4:8)=IFIRST5
HDR2ADR(11:14)=ILAST4
HDR2ADR(51:58)='SLREDALA'
HDR2ADR(41:48)='LT000000'
HDR2ADR(31:38)='RT000000'
HDR2ADR(21:28)='DT000000'
C HDR2ADR(351:358)=MILISECONDS
CALL TIME(TIMEA)
HDR2ADR(43:44)=TIMEA(1:2)
HDR2ADR(45:46)=TIMEA(4:5)
HDR2ADR(47:48)=TIMEA(7:8)
C HDR2ADR(33:38)=HDR2ADR(43:48)
C HDR2ADR(23:28)=HDR2ADR(43:48)
HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)

```

[AVA.MAXDISK]MDSKTOFIL

```
C      CALL UNCNVRT(XVAL(HDR2ADR),HDR2LEN,HDR2ADR)
C      TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C      TYPE*,' WRITING ',FNAM(1:40)
      IHD2=HDR2LEN      !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
      ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,XVAL(HDR2ADR))
      IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
      RETURN
      END
```

## [AVA.MAXDISK]MDSKTFIL2

C

[AVA.MAXDISKIMDSKTFIL2

```

1 '24861'O,'26882'O,'38888'O,'44888'O,'64777'O,'128888'O,
2 '58881'O,'78776'O,'54888'O/
DATA ISETUP2/'64777'O,'44888'O/
DATA ISETUP3/'64776'O,'44888'O/
DATA IFIRST/1/
I=SYSSASSIGN('TT',IVTC,,)
IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA8',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=8
AVAACR='415'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
IBUF(1),XVAL(28),,,)
OPEN(UNIT=22,NAME='DISKSAVA:[AVA]IRIGS.DAT',STATUS='OLD',READONLY)
TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
NIMAGES=1
INSZ=NIMAGES*513
FNAME='DISKSAVA:[AVA]IMAGES.DAT'
OPEN(UNIT=38,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS2,
2RECORDTYPE='FIXED',RECORDSIZE=4896)
IEVFO=4
Y=6
X=8
TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
ACCEPT*,IBLOCK
IF(IBLOCK.EQ.1)THEN
IBLOCK=1
ELSE
IBLOCK=1+(488*(IBLOCK-1))
ENDIF
ICOUNT=8
CALL TIMRB
C TYPE *,'ENTER NUMBER IMAGES TO STORE'
ACCEPT*,IFIELDS
IFIELDS=IFIELDS*8 ICONVERT IMAGES TO TRANSFERS
1 IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
NBYTES=24576
ELSE
NBYTES=32768
ENDIF
ISTATUS=SYSSQIOW(XVAL(IEVFO),XVAL(IDISK2),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,
IBINPUT(1),XVAL(NBYTES),XVAL(1BLOCK),,,)
IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
IBLOCK=IBLOCK+48
NUMB=24576
ELSE
NUMB=32768
IBLOCK=IBLOCK+64
ENDIF
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C 1IOSB,,

```

[AVA.MAXDISK]MDSKTFIL2

```

C      1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      1INPUT,XVAL(38728),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      IF(AVACSR.EQ.0)AVACSR=1
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C      WRITE (4,54)BINPUT
54     FORMAT(1X,16(1X,03))

C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C      IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
C        IGBYTES=24624
C      ELSE
C        IGBYTES=32832
C      ENDIF

C      ISTATUS = SYS$QIO(XVAL(1),XVAL(CHAN),
C        1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...,
C        IBOUT(1),XVAL(IGBYTES),...,)
C      ISTATUS = SYS$QIO(XVAL(1),XVAL(CHAN),
C        1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...,
C        IBOUT(IGBYTES+1),XVAL(IGBYTES),...,)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C      IF(IFIRST)THEN
C        TYPE *,'ENTER DISK NAME AND DIRECTORY. (DISK$IMAGES:[WILLIAMS])'
C        READ(5,6)DISKSPEC
C        DISKSPEC='DISK$AVA2:[AVA]'
C        DISKSPEC='DISK$AVA:[AVA]'
C        TYPE *,'ENTER FIRST IMAGE FILE NAME. (V00010000.IMG)'
C        READ(5,6)FNAM2
C        FORMAT(A)
C        FNAM='SIMS000001.IMG'
C        NAMNUM=FNAM2(2:5)
C        FNAM=DISKSPEC//FNAM2
C        DECODE(4,101,NAMNUM)INAMNUM
101     FORMAT(I4)
C        NCOL=512
C        NROW=488
C        ILEN=NROW
C        IWD=NCOL
C        IMGMAPC(3)=ILEN ILENGTH OF IMAGE
C        IMGMAPC(4)=IWD IWIDTH OF IMAGE
C        IFIRST=0
C        I=IWD
C        NBYT=(I+1)*ILEN*2
C        I=LIB$GETVM(NBYT,IMGADR)
C        IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MEMORY ASSIGNMENT 1'
C        I = LIB$GETVM(10000,HDR2ADR)
C        IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
C        ENDIF
C        HEAD(8)='          1' IONE CHARACTER PER CHANNEL
C        HDR2LEN=576
C        CURRENTNUMFL=0
C        CALL ADDHDR2(XVAL(HDR2ADR))
C        CALL IMGTODISK(BINPUT,NUMB,XVAL(IMGADR),ICOUNT)

```

[AVA.MAXDISK]MDSKTFIL2

```

C      IF(ICOUNT.EQ.7)THEN
      FNAME='SIMS00001.IMG'
      INAMNUM=INAMNUM+1
      ISTATUS=OTSSCVTLTI(INAMNUM,NAMNUM,XVAL(4),XVAL(4),)
      IF(.NOT.ISTATUS)TYPE *,'CONVERSION ERROR IN FILE NAME'
      FNAME(2:5)=NAMNUM
      FNAME=DISKSPEC//FNAME
      ENDIF
      Y=Y+32
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      ISETUP3(1),XVAL(4),,,)
      Y='206'0
      X=0
      ENDIF
      IF(ICOUNT.EQ.8)THEN
      ICOUNT=0
      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      ISETUP2(1),XVAL(4),,,)
      CALL TIMRE
      CALL HEADER(TITLE)
      STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
      Y=6
      ENDIF
      IFIELDS=IFIELDS-1
      IF(IFIELDS.EQ.0)STOP 'ALL IMAGES WRITTEN TO DISK'
      GO TO 1
57    CONTINUE
      ISTATUS=SYSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
      TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
      IF(.NOT.ISTATUS)TYPE *,'ERROR IN CALL TO $GETMSG'
      TYPE *,'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS)TYPE *,'ERROR IN CALL TO $GETMSG'
      TYPE *,'I/O STATUS:',MSGBUF
      STOP
C11   FORMAT(1X,'INPUT=',06,2X,'IOSB=',06,2X,06,2X,06,2X,06)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
C      ISETUP3,XVAL(4),,,)
      END
      SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
      BYTE BINPUT(1),BYTE(2)
      INTEGER*2 OUT(513,1),BYTES,SLU
      EQUIVALENCE(BYTES,BYTE)
      DATA SLU/'34011'0/
      I=0
      IOLINE=1
      DO 100 IX=1,NUMB
      I=I+1
      IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES

```

IAVA.MAXDISKIMDSKTFIL2

```

C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
C      WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=8
      IOLINE=IOLINE+1
      GO TO 188
    ENDIF
    BYTE(1)=BINPUT(IX)
    OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34     FORMAT(1X,13,1X,13,2X,06)
188    CONTINUE
      RETURN
      END

SUBROUTINE IMGTODISK(BINPUT,NUMB,IMAGE,ICOUNT)
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
  INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
  INTEGER*2 IMAGE(NCOL+1,NROW)
  INTEGER*4 IMGADR,SYSS$ASSIGN,IMGADR2,SYSS$GETMSG
  BYTE BINPUT(1)
  INDEX=1
  IF(ICOUNT.LE.3)THEN
    ISTART=1+(ICOUNT*128)
    IF(ICOUNT.LE.2)IEND=ISTART+126
    IF(ICOUNT.EQ.3)IEND=ISTART+94
    DO I=ISTART,IEND,2
      DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C      IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
      ENDDO
    ENDDO
  ELSE
    ISTART=2+((ICOUNT-4)*128)
    IF(ICOUNT.LE.6)IEND=ISTART+126
    IF(ICOUNT.EQ.7)IEND=ISTART+94
    DO I=ISTART,IEND,2
      DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C      IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
      ENDDO
    ENDDO
  ENDIF
  ILEN=NROW
  IWD=NCOL
C  TYPE*,IWD AND ILEN BEFORE TODISK=',IWD,ILEN
  IF(ICOUNT.EQ.7) CALL TODISK(IMAGE,IWD,ILEN)

```

[AVA.MAXDISK]MDSKTFIL2

```

C      WRITE(6,1)ICOUNT
1      FORMAT(1X,I10)
      RETURN
      END
      SUBROUTINE TODISK(IMAGE,IWD,ILEN)
      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
      INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
      INCLUDE 'DISK$USERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
      INTEGER*2 IMAGE(NCOL+1,NROW),HDR2LEN
      INTEGER*4 AUTOWRTSB
      CHARACTER*10000 HDR2ADR
      CHARACTER*3 MONTH,DAY,YEAR*2,WD*8,LEN*8,TIMEA*8
      CHARACTER*5 IFIRST5,ILAST4*4,TNAME*9
      CHARACTER*2 HOUR,MINUTES,SECONDS,MSECONDS*3,HMS*12
      HDR2LEN=HDR2LEN
C      TYPE *,HDR2LEN',HDR2LEN
C      TYPE *,HDR2LEN ',HDR2LEN
      CALL CNVRT(XVAL(HDR2ADR),HDR2LEN,HDR2ADR)
C      TYPE *,HEAD
      CALL IDATE(IMONTH,IDAY,IYEAR)
      ENCODE(3,200,MONTH)IMONTH
      ENCODE(3,200,DAY )IDAY
200  FORMAT(I3)
      ENCODE(2,100,YEAR )IYEAR
      HEAD(3)='OLDFAAD '
      HEAD(1)='USAMICOM'
      HEAD(2)=YEAR//MONTH//DAY
100  FORMAT(I2)
      ENCODE(8,200,WD)IWD
      HEAD(11)(6:8)=WD(1:3)
      ENCODE(8,200,LEN)ILEN
      HEAD(12)(6:8)=LEN(1:3)
C      TYPE *,HEAD
      IBRACKET=INDEX(FNAM,'1')
      IPERIOD=INDEX(FNAM,'.')
      TNAME=FNAM(IPERIOD-9:IPERIOD-1)
      IBRACKET=IBRACKET
      IF(IPERIOD-10.LT.IBRACKET)THEN
      IZERO=ABS(IPERIOD-10)
      TNAME(1:IZERO)=' '
      ENDIF
      ILAST4=TNAME(6:9)
      IFIRST5=TNAME(1:5)

      HDR2ADR(1:8)= 'FN*X0000'
      HDR2ADR(11:18)='0000.IMG'
      HDR2ADR(4:8)=IFIRST5
      HDR2ADR(11:14)=ILAST4
      HDR2ADR(51:58)='SLREDALA'
      HDR2ADR(41:48)='LT000000'
      HDR2ADR(31:38)='RT000000'
      HDR2ADR(21:28)='DT000000'
      CALL TIME(TIMEA)

```

[AVA,MAXDISK]MDSKTFIL2

```
HDR2ADR(43:44)=TIMEA(1:2)
HDR2ADR(45:46)=TIMEA(4:5)
HDR2ADR(47:48)=TIMEA(7:8)
C HDR2ADR(33:38)=HDR2ADR(43:48)
C HDR2ADR(23:28)=HDR2ADR(43:48)
HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)
READ(22,22)HOUR,MINUTES,SECONDS,MSECONDS
22 FORMAT(5X,A2,1X,A2,1X,A2,1X,A3)
HDR2ADR(33:34)=HOUR
HDR2ADR(35:36)=MINUTES
HDR2ADR(37:38)=SECONDS
HDR2ADR(279:281)=MSECONDS

CALL UNCNVRT(XVAL(HDR2ADR),HDR2LEN,HDR2ADR)
C TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C TYPE*,' WRITING ',FNAM(1:40)
IHD2=HDR2LEN !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,XVAL(HDR2ADR))
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
RETURN
END
```

APPENDIX J

[AVA]AVAFIELDS

12

[AVAJAVAFIELDS

```

        IF(IFIELD.EQ.3)Y='606'O
        IF(IFIELD.LT.0.OR.IFIELD.GT.3)GO TO 12
        X=0
        ICOUNT=0
1       ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
        1IOSB,...
C       1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        IF(AVACSR.EQ.0)AVACSR=1
C       IF(.NOT.1STATUS.OR..NOT.1OSB(1))GO TO 57
C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,03))

        NUMB=32768
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
        1XVAL(XLOC(10$WRITEVBLK)),1OSB,...
        1BOUT(1),XVAL(65534),...,)
        ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
        1XVAL(XLOC(10$WRITEVBLK)),1OSB,...
        1BOUT(65535),XVAL(130),...,)
C       IF(.NOT.1STATUS.OR..NOT.1OSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
            K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10$WRITEVBLK)),1OSB,...
            11SETUP3(1),XVAL(4),...,)
            GO TO 12
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYSSGETMSG (XVAL(1STATUS), MSGLEN, MSGBUF,..)
        TYPE *,' 1STATUS=',1STATUS,' 1OSB(1)=',1OSB(1)
        TYPE *,' 1STATUS=',1STATUS,' 1OSB(1)=',1OSB(1)
        IF(.NOT.1STATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYSSGETMSG (XVAL(1OSB(1)), MSGLEN, MSGBUF,..)
        IF(.NOT.1STATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',06,2X,'1OSB=',06,2X,06,2X,06,2X,06)
C       K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10$WRITEVBLK)),1OSB,...
C       11SETUP3,XVAL(4),...,)
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34011'O/
        I=0
        IOLINE=1
        DO 100 IX=1,NUMB

```

[AVA]AVAFIELDS

```

      I=I+1
      IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
C      WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
      ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
100      CONTINUE
      RETURN
      END

```

APPENDIX K  
[AVA]AVAGROUP8

```

EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,60),X,Y
BYTE BOUT(61560),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(15360)
BYTE BINPUT(30720)
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='415'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
IBUF(1),XVAL(28),...)
Y=6
X=0
ICOUNT=0
1 ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
IOSB,...)
C IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IINPUT,XVAL(30720),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT. ISTATUS.OR..NOT. IOSB(1))GO TO 57
C WRITE (4,54)BINPUT

```

[AVA]AVAGROUP8

```

54      FORMAT(1X,16(1X,03))

      NUMB=38728
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *, 'NUMBER OF LINES TO OUTPUT=', IOLINE
      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      1BOUT(1),XVAL(61568),,,)
C      IF(.NOT. ISTATUS.OR..NOT. IOSB(1))GO TO 57
      Y=Y+38
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      1IOSB,,
      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      NUMB=8192
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      1BOUT(1),XVAL(16384),,,)
      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      11SETUP3(1),XVAL(4),,,)
      Y='286'O
      X=8
      ENDIF
      IF(ICOUNT.EQ.8)THEN
      ICOUNT=8
      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      1IOSB,,
      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      NUMB=8192
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      1BOUT(1),XVAL(16384),,,)
      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
      11SETUP2(1),XVAL(4),,,)
      Y=6
      ENDIF
      GO TO 1
57      CONTINUE
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
      IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'I/O STATUS:',MSGBUF
      STOP
C11     FORMAT(1X,'INPUT=',06,2X,'IOSB=',06,2X,06,2X,06,2X,06)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
C      11SETUP3,XVAL(4),,,)
      END

```

[AVA]AVAGROUP8

```

SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
  BYTE BINPUT(1),BYTE(2)
  INTEGER*2 OUT(513,1),BYTES,SLU
  EQUIVALENCE(BYTES,BYTE)
  DATA SLU/'34011'O'/
  I=0
  IOLINE=1
  DO 100 IX=1,NUMB
    I=I+1
    IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
      C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
      C WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
    ENDIF
    BYTE(1)=BINPUT(IX)
    OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
      C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      C34 FORMAT(1X,I3,1X,I3,2X,O6)
      C100 CONTINUE
      RETURN
  END
```

# APPENDIX L

[AVA]AVAGROUP9

```

EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
BYTE BINPUT(32768)
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='415'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(%LOC(IOSWRITEVBLK)),IOSB,,
1BUF(1),XVAL(28),,,)
Y=6
X=0
ICOUNT=0
1 ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(%LOC(IOSREADVBLK)),
1IOSB,,
C 1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT. ISTATUS.OR..NOT. IOSB(1))GO TO 57
C WRITE (4,54)BINPUT

```

[AVAJAVAGROUP9

```

54      FORMAT(1X,16(1X,03))
      NUMB=32768
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *, 'NUMBER OF LINES TO OUTPUT=', IOLINE
      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
      IBOUT(1),XVAL(65534),,,,
      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
      IBOUT(65535),XVAL(130),,,,
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
      Y=Y+32
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C      IOSB,,,
C      IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
C      IBOUT(1),XVAL(8192),,,,
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
      ISETUP3(1),XVAL(4),,,,
      Y='206'0
      X=0
      ENDIF
      IF(ICOUNT.EQ.8)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C      IOSB,,,
C      IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
C      IBOUT(1),XVAL(8192),,,,
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
      ISETUP2(1),XVAL(4),,,,
      Y=6
      ENDIF
      GO TO 1
57      CONTINUE
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'I/O STATUS:',MSGBUF
      STOP
C11     FORMAT(1X, 'INPUT=' ,06,2X, 'IOSB=' ,06,2X,06,2X,06,2X,06)

```

[AVA]AVAGROUP9

```

C      K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      I1SETUP3,XVAL(4),,,)
      END
      SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
      BYTE BINPUT(1),BYTE(2)
      INTEGER*2 OUT(513,1),BYTES,SLU
      EQUIVALENCE(BYTES,BYTE)
      DATA SLU/'34011'O/
      I=0
      IOLINE=1
      DO 100 IX=1,NUMB
      I=I+1
      IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
C      OUT(I+1,IOLINE)=SLU
      WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
      ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      FORMAT(1X,I3,1X,I3,2X,O6)
C 34      CONTINUE
      RETURN
      END
100

```

APPENDIX M

[AVA]AVAFWRITE

```
C CCCCCCCCCCC
```

```
C      THIS PROGRAM ALLOWS THE USER TO WRITE A SPECIFIED BYTE TO A FIELD  
C IN THE AVA VIDEO MEMORY SELECTED BY HIM.  
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC  
EXTERNAL IOSWRITEVBLK,IOSREADVBLK  
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)  
INTEGER SYSSASSIGN,SYSSQIOW,CHAN,SYSSQIO,SYSSWAITFR  
INTEGER SYS$GETMSG,MGLLEN,ISTATUS  
INTEGER*2 OUT(513,64),X,Y  
BYTE BOUT(65664),BYTE(2)  
INTEGER*2 BYTES  
INTEGER*2 OUTPUT,INIT(4)  
INTEGER*2 INPUT(16384)  
INTEGER IERROR(512)  
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)  
INTEGER*2 IDATA  
INTEGER AVACSR,AVAACR  
INTEGER*2 ISETUP2(2),ISETUP3(2)  
CHARACTER *8 MSGBUF,TITLE  
EQUIVALENCE(BUF(1),ISETUP(1))  
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)  
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)  
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,  
1'24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,  
2'50001'O,'70776'O,'54000'O/  
DATA ISETUP2/'64777'O,'44000'O/  
DATA ISETUP3/'64776'O,'44000'O/  
I = SYSSASSIGN('GRAO',CHAN,..)  
IF(.NOT.I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'  
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,..)  
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'  
AVACSR=0  
AVAACR='435'O  
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...  
IBUF(1),XVAL(20)...)  
ITESTN=1  
IERRORC=0  
DATAINA(1)='-125'O
```

[AVA]AVAFWRITE

```

DATAINA(2)=0
DATAINA(3)='252'O
DATAINA(4)='377'O
DATAIN='125'O
DATAIN=DATAINA(ITESTN)
57  TYPE *, 'ENTER DATA TO BE WRITTEN INTO THE FIELD IN OCTAL. (I.E. 377)'
    READ(5,56)DATAIN
56  FORMAT(03)
    DO I=1,32768
    BINPUT(I)=DATAIN
    ENDDO
    TYPE *, 'ENTER THE FIELD TO WRITE THE DATA INTO. (0,1,2, OR 3)'
    READ(5,56)IFIELD
    IF(IFIELD.EQ.0)Y=6
    IF(IFIELD.EQ.1)Y='206'O
    IF(IFIELD.EQ.2)Y='406'O
    IF(IFIELD.EQ.3)Y='606'O
    IF(IFIELD.LT.0.OR.IFIELD.GT.3)GO TO 57
C177 Y=6
    X=0
    ICOUNT=0
1    ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
    IOSB,...
C    1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
    1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
    IF(AVACSR.EQ.0)AVACSR=1
C    IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C    WRITE (4,54)BINPUT
54  FORMAT(1X,16(1X,03))

C    NUMB=32768
C    CALL BUFFCNVT(NUMB,BINPUT,OUT)
C    TYPE *, 'NUMBER OF LINES TO OUTPUT=',IOLINE
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C    1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C    1BOUT(1),XVAL(65534),...)
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C    1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C    1BOUT(65535),XVAL(130),...)
C    IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
Y=Y+32
ICOUNT=ICOUNT+1
IF(ICOUNT.EQ.4)THEN
GO TO 57
ENDIF
GO TO 1
END
SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
BYTE BINPUT(1),BYTE(2)
INTEGER*2 OUT(513,1),BYTES,SLU
EQUIVALENCE(BYTES,BYTE)
DATA SLU/'34011'O/
I=0
IOLINE=1

```

[AVAJAVAFWRITE

```
DO 100 IX=1,NUMB
I=I+1
IF(I.EQ.512)THEN
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=BYTES
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
OUT(I+1,IOLINE)=SLU
C WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
I=0
IOLINE=IOLINE+1
GO TO 100
ENDIF
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34 FORMAT(IX,I3,IX,I3,2X,O6)
100 CONTINUE
RETURN
END
```

# APPENDIX N

[AVA]RAMPMAX

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
BYTE BINPUT(32768),BDATA(2)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'0,'140001'0,'121000'0,'107777'0,'17777'0,
1 '24061'0,'26002'0,'30000'0,'44000'0,'64777'0,'120000'0,
2 '50001'0,'70776'0,'54000'0/
DATA ISETUP2/'64777'0,'44000'0/
DATA ISETUP3/'64776'0,'44000'0/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVAB',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'0
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
1BUF(1),XVAL(20),,,,)
Y=6
X=0
ICOUNT=0
DO I=1,32768
IDATA=IAND((I-1),'377'0)
BINPUT(I)=BDATA(1)
ENDDO
TITLE='FOUR FIELD WRITE TO AVA'
CALL TIMRB
1 ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
```

[AVA]RAMPMAX

```

C      IOSB,...
C      INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      IF(AVACSR.EQ.0)AVACSR=1
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C      WRITE (4,54)BINPUT
54     FORMAT(1X,16(1X,O3))

C      NUMB=32768
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *, 'NUMBER OF LINES TO OUTPUT=', IOLINE
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      IBOUT(1),XVAL(65534),...,)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      IBOUT(65535),XVAL(130),...,)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C      Y=Y+32
C      ICOUNT=ICOUNT+1
C      IF(ICOUNT.EQ.4)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK))),
C      IOSB,...
C      INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      IBOUT(1),XVAL(8192),...,)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      ISETUP3(1),XVAL(4),...,)
C      Y='206'O
C      X=0
C      ENDIF
C      IF(ICOUNT.EQ.8)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK))),
C      IOSB,...
C      INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      IXVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      IBOUT(1),XVAL(8192),...,)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      ISETUP2(1),XVAL(4),...,)
C      Y='406'O
C      ENDIF
C      IF(ICOUNT.EQ.16)THEN
C      Y='606'O
C      ENDIF
C      IF(ICOUNT.EQ.24)THEN
C      ICOUNT=0
C      CALL TIMRE
C      CALL HEADER(TITLE)

```

[AVA]RAMPMAX

```

      Y=6
      CALL TIMRB
      ENDIF

57      GO TO 1
      CONTINUE
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
      IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
      TYPE *, 'I/O STATUS:', MSGBUF
      STOP
C11     FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C       K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC(IOSWRITEVBLK)), IOSB,,)
C       ISETUP3, XVAL(4),,,)
      END
      SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
      BYTE BINPUT(1), BYTE(2)
      INTEGER*2 OUT(513,1), BYTES, SLU
      EQUIVALENCE(BYTES, BYTE)
      DATA SLU/'34011'O/
      I=0
      IOLINE=1
      DO 100 IX=1, NUMB
      I=I+1
      IF(I.EQ.512) THEN
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=BYTES
C       WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
      OUT(I+1, IOLINE)=SLU
C       WRITE(6,34) I+1, IOLINE, OUT(I+1, IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
      ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=IAND(NOT(BYTES), '377'O)
C       WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
      FORMAT(1X, I3, 1X, I3, 2X, 06)
C34     CONTINUE
      RETURN
      END
100

```

APPENDIX O  
[AVALRAMP]MAX2

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
PARAMETER NLINES=64
INTEGER*2 OUT(513,NLINES),X,Y
BYTE BOUT(1026*NLINES),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(256*NLINES)
BYTE BINPUT(512*NLINES),BDATA(2)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
1BUF(1),XVAL(28),...)
Y=6
X=0
ICOUNT=0
TITLE='AVA READ FRAME TIME'
DO I=1,512*NLINES
IDATA=IAND((I-1),'377'O)
BINPUT(I)=BDATA(I)
ENDDO
CALL TIMRB
```

[AVA]RAMPX2

```

1      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
      1IOSB,...
C      1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      1INPUT,XVAL(512*NLINES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      IF(AVACSR.EQ.0)AVACSR=1
      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C
C      WRITE (4,54)BINPUT
54     FORMAT(1X,16(1X,03))
C
C      NUMB=512*NLINES
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      1BOUT(1),XVAL(65534),,...)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      1BOUT(65535),XVAL(36866),,...)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
      Y=Y+(NLINES/2)
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
C      1IOSB,...
C      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      1BOUT(1),XVAL(8192),,...)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      11SETUP3(1),XVAL(4),,...)
      Y='206'0
      X=0
      ENDIF
      IF(ICOUNT.EQ.8)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
C      1IOSB,...
C      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      1BOUT(1),XVAL(8192),,...)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C      11SETUP2(1),XVAL(4),,...)
      CALL TIMRE
      CALL HEADER(TITLE)
      CALL TIMRB
      Y=6
      ENDIF
      GO TO 1
57     CONTINUE

```

[AVA]RAMPMAX2

```

      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
      TYPE *, 'I/O STATUS:', MSGBUF
      STOP
C11  FORMAT(1X, 'INPUT=', 06, 2X, 'IOSB=', 06, 2X, 06, 2X, 06, 2X, 06)
C    K = SYSSQIOW(XVAL(1), XVAL(CHAN), XVAL(XLOC(IOSWRITEVBLK)), IOSB,,)
C    I1SETUP3, XVAL(4), ..., )
      END
      SUBROUTINE BUFFCNVT(NUMB, BINPUT, OUT)
      BYTE BINPUT(1), BYTE(2)
      INTEGER*2 OUT(513,1), BYTES, SLU
      EQUIVALENCE(BYTES, BYTE)
      DATA SLU/'34011'0/
      I=0
      IOLINE=1
      DO 100 IX=1, NUMB
      I=I+1
      IF(I.EQ.512) THEN
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=BYTES
C    WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
      OUT(I+1, IOLINE)=SLU
C    WRITE(6,34) I+1, IOLINE, OUT(I+1, IOLINE)
      I=0
      IOLINE=IOLINE+1
      GO TO 100
      ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I, IOLINE)=IAND(NOT(BYTES), '377'0)
C    WRITE(6,34) I, IOLINE, OUT(I, IOLINE)
34  FORMAT(1X, I3, 1X, I3, 2X, 06)
100  CONTINUE
      RETURN
      END

```

APPENDIX P

[AVA.MAXDISK]MTODSKT

[illegible]

[AVA.MAXDISK]MTODSKT

```

DATA DVISFREEBLOCKS/'0000002A'X/,ZERO/0/
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
NAME='DISKSAVA:'
BUFFERL=4
DEVCODE=DVISFREEBLOCKS
BUFFERA=XLOC(IFREE)
RETURNLA=XLOC(RETURNL)
ISTATUS=SYSSGETDVI(XVAL(3),,NAME,BUFFERL,,, )

IF(.NOT. ISTATUS)TYPE *, 'PARAMETER ERROR IN GETDVI'

ISTATUS=SYSSWAITFR(XVAL(3))
TYPE *, 'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
MAXIMAGES=IFREE/513

TYPE *, 'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES

MAXIMAGES=50      !THIS IS FOR DEGUG ONLY

NIMAGES=MAXIMAGES
INSZ=NIMAGES*513
FNAME='DISKSAVA:[AVA]IMAGES.DAT'
OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
IFORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
2RECORDTYPE='FIXED',RECORDSIZE=4096)
ISTATUS=SYSSASSIGN('AVA0',AVACHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
INLOCK(1)=XLOC(BINPUT(1))
INLOCK(2)=XLOC(BINPUT(131072))
K=SYSSLKWSET(INLOCK,IOLOCK,)
TYPE *, ' INLOCK(1)= ',INLOCK(1), ' INLOCK(2)= ',INLOCK(2)
TYPE *, ' IOLOCK(1)= ',IOLOCK(1), ' IOLOCK(2)= ',IOLOCK(2)
IF(.NOT. K)TYPE *, ' UNABLE TO LOCK BUF'
AVACSR=0
AVAACR='415'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
1BUF(1),XVAL(28),,,, )
IEVFO=4
IMAGEN=1
IBLOCK=1
CALL TIMRB
X=0
CALL FIELD(IFIELD,AVACSR)
ICURR=IFIELD
TYPE *, 'ICURR=',ICURR

```

[AVA.MAXDISK]MTODSKT

```

11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR          ICURRENT FIELD TO PUT ON DISK
        ICURR=IFIELD              ICURRENT FIELD BE LOADED INTO THE AVA
C      TYPE *, 'ICURR=', ICURR, 'ISTOREFIELD=', ISTOREFIELD
C      GO TO 11
        Y=VA(ISTOREFIELD+1)
C      ICOUNT=#
1      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
        1IOSB,,,
        1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        IF(AVACSR.EQ.#)AVACSR=1
        CALL DELAY(TIME)
C      ISTATUS=SYSSQIO(XVAL(1EVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
C      1IOSB,,,
C      1INPUT(1),XVAL(32768),XVAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
2      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
        1IOSB,,,
        1INPUT(32679),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        CALL DELAY(TIME)
C      ISTATUS=SYSSQIO(XVAL(1EVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
C      1IOSB,,,
C      1INPUT(32679),XVAL(32768),XVAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
3      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
        1IOSB,,,
        1INPUT(65537),XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        CALL DELAY(TIME)
C      ISTATUS=SYSSQIO(XVAL(1EVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
C      1IOSB,,,
C      1INPUT(65537),XVAL(32768),XVAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
4      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
        1IOSB,,,
        1INPUT(98305),XVAL(24576),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        CALL DELAY(TIME2)
C      ISTATUS=SYSSQIO(XVAL(1EVFO),XVAL(IDISK),XVAL(XLOC(10$WRITEVBLK)),
C      1IOSB,,,
C      1INPUT(98305),XVAL(24576),XVAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
C      IF(AVACSR.EQ.#)AVACSR=1
C      WRITE (4,54)BINPUT
54      FORMAT(1X 16(1X,03))

C      NUMB=32768
C      CALL BUFCNV(TNUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(10$WRITEVBLK)),IOSB,,,
C      1BOUT(1),XVAL(65534),,,)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),

```

[AVA.MAXDISK]MTODSKT .

```

C      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      1BOUT(65535),XVAL(130),...,)
C      ICOUNT=ICOUNT+1
C      IF(ICOUNT.NE.4)GO TO 1
      CALL FIELD(IFIELD,AVACSR)
           IF(IFIELD.EQ.ICURR)THEN
      IF(IMAGEN.EQ.MAXIMAGES)THEN
      ISTATUS=SYSS$WAITFR(XVAL(1))
      IF(.NOT.ISTATUS)TYPE *, ' ERROR IN TIME DELAY'
      TYPE *, ' I/O COMPLETE.....'
      CALL TIMRE
      CALL HEADER(TITLE)
      ISTATUS=SYSS$DASSGN(XVAL(IDISK))
      CLOSE(UNIT=30)
      TYPE *,IMAGEN, ' FIELDS WRITTEN TO DISK'
      STOP 'ALL IMAGES WRITTEN TO DISK'
      ENDIF
      IMAGEN=IMAGEN+1
           GO TO 11
           ELSE
      TYPE *, 'FATAL ERROR....***....I/O TO SLOW'
      TYPE *, 'BLOCK NUMBER=',IBLOCK
      TYPE *, 'ICURR=',ICURR, ' IFIELD=',IFIELD
      TYPE *,IMAGEN, ' FIELDS WRITTEN TO DISK'
      CALL TIMRE
      CALL HEADER(TITLE)
      ISTATUS=SYSS$DASSGN(XVAL(IDISK))
      CLOSE(UNIT=30)
      STOP
      ENDIF
57      CONTINUE
      ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=',IOSB(1)
      TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=',IOSB(1)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'I/O STATUS:',MSGBUF
      STOP
C11     FORMAT(1X, 'INPUT=',06,2X, 'IOSB=',06,2X,06,2X,06,2X,06)
C      K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
C      I1SETUP3,XVAL(4),...,)
           END
           SUBROUTINE DELAY(TIME)
           DOUBLE PRECISION QUAD
           INTEGER SYSS$BINTIM,SYSS$SETIMR,SYSS$WAITFR
           CHARACTER*16 TIME
C      RETURN
C      TIME='0000 00:00:00.50'
      ISTATUS=SYSS$BINTIM(XDESCR(TIME),QUAD)
      IF(.NOT.ISTATUS)TYPE *, ' ERROR IN TIME DELAY'
      ISTATUS=SYSS$SETIMR(XVAL(6),QUAD,,)
      IF(.NOT.ISTATUS)TYPE *, ' ERROR IN TIME DELAY'

```

[AVA.MAXDISK]MTODSKT

```

ISTATUS=SYSSWAITFR(XVAL(6))
IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
RETURN
END
SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
BYTE BINPUT(1),BYTE(2)
INTEGER*2 OUT(513,1),BYTES,SLU
EQUIVALENCE(BYTES,BYTE)
DATA SLU/'34011'O/
I=0
IOLINE=1
DO 100 IX=1,NUMB
I=I+1
IF(I.EQ.512)THEN
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=BYTES
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
OUT(I+1,IOLINE)=SLU
C WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
I=0
IOLINE=IOLINE+1
GO TO 100
ENDIF
BYTE(1)=BINPUT(IX)
OUT(I,IOLINE)=IAND(.NOT.(BYTES),'377'O)
C WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34 FORMAT(1X,I3,1X,I3,2X,O6)
100 CONTINUE
RETURN
END

SUBROUTINE FIELD(IFIELD,AVACSR)
INTEGER AVACSR
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER SYSSASSIGN,SYSSQIOW,SYSSQIO
INTEGER SYSSGETMSG
INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
INTEGER*2 INPUT,OUTPUT,INIT(4)
CHARACTER *80 MSGBUF
COMMON/AVACHAN/ITCHAN
DATA IFIRST/1/
ISAVE=AVACSR
IF(IFIRST)THEN
AVACSR='4000'O ISET MEMORY WINDOW ENABLE AND INITIALIZE AVA
IFIRST=0
ELSE
AVACSR='4001'O
ENDIF
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,
C IOUTPUT,XVAL(2),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(IAVAACR))
C IF(AVACSR.EQ.'4000'O)AVACSR='4001'O
C IF(ISTATUS) GO TO 501
C TYPE *,' ERROR IN QIOW CALL'
C ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)

```

[AVA.MAXDISK]MTODSKT

```
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
C      MSGBUF=' '
C      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
C      TYPE *, 'I/O STATUS:', MSGBUF
501    AVACSR=ISAVE
      IFIELD=IAND(OUTPUT,3)
      RETURN
      END
```

APPENDIX Q  
[AVA]FAVAMENT

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSS$ASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...
1BUF(1),XVAL(28),...,)
ITESTN=1
IERRORC=0
DATAINA(1)='125'O
DATAINA(2)=0
DATAINA(3)='252'O
DATAINA(4)='377'O
DATAIN='125'O
DATAIN=DATAINA(ITESTN)
DO I=1,32768
```

[AVA]FAVAMEMT

```

      BINPUT(I)=DATAIN
      ENDDO
177  Y=6
      X=0
      ICOUNT=0
      TITLE='FOUR FIELD WRITE TO AVA'
C    CALL TIMRB
I    ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
      IOSB,...)
C    IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C    IF(AVACSR.EQ.0)AVACSR=1
      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C
C    WRITE (4,54)BINPUT
54   FORMAT(1X,16(1X,03))
C
C    NUMB=32768
      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C    TYPE = 'NUMBER OF LINES TO OUTPUT=',IOLINE
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    1BOUT(1),XVAL(65534),,...)
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    1BOUT(65535),XVAL(130),,...)
C    IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
      Y=Y+32
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
C    ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      1IOSB,...)
C    1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      NUMB=8192
C    CALL BUFFCNVT(NUMB,BINPUT,OUT)
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    1BOUT(1),XVAL(8192),,...)
C    K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    11SETUP3(1),XVAL(4),,...)
      Y='206'0
      X=0
      ENDIF
      IF(ICOUNT.EQ.8)THEN
C    ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      1IOSB,...)
C    1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      NUMB=8192
C    CALL BUFFCNVT(NUMB,BINPUT,OUT)
C    ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    1BOUT(1),XVAL(8192),,...)
C    K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C    11SETUP2(1),XVAL(4),,...)
      Y='406'0

```

(AVA)FAVAMENT

```

      ENDIF
      IF(ICOUNT.EQ.16)THEN
        Y='606'O
      ENDIF
      IF(ICOUNT.EQ.24)THEN
CCCCCCCCC READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME CCCCC
C
        ICOUNT=0
        Y=6
        X=0
51      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
        IIOSB,,
        IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
        IF(AVACSR.EQ.0)AVACSR=1
        DO IADD=1,32768
          IF(BINPUT(IADD).NE.DATAIN)THEN
            IADD=IAND(IADD,'777'O)
C          WRITE(6,12)DATAIN,BINPUT(IADD),
C          IAND(IADD,'777'O)
            IERROR(IADD)=IERROR(IADD)+1
            IERRORC=IERRORC+1
          ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',03,2X,'OUTPUT= ',03,
            15X,'COLUMN= ',15)
        ENDDO
C      TYPE *, ' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT

        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
          WRITE(6,56)IERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',15,' DATA IN= ',03)
          IERRORC=0
          Y='206'O
          X=0
          ENDIF
          IF(ICOUNT.EQ.8)THEN
            WRITE(6,157)IERRORC,DATAIN
157      FORMAT(1X,' ERROR COUNT IN FIELD 2=',15,' DATA IN= ',03)
            IERRORC=0
            Y='406'O
            ENDIF
            IF(ICOUNT.EQ.16)THEN
              Y='606'O
              WRITE(6,58)IERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',15,' DATA IN= ',03)
              IERRORC=0
              ENDIF
              IF(ICOUNT.EQ.24)THEN
                WRITE(6,59)IERRORC,DATAIN
59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',15,' DATA IN= ',03)
                IERRORC=0
                DO IADD=1,512
                  IF(IERROR(IADD).NE.0)WRITE(6,233)IADD,

```

[AVA]FAVAMENT

```

233      IERROR(IADD),DATAIN
        FORMAT(IX,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=0
        DO IADD=1,512
          IERROR(IADD)=0
        ENDDO
        ICOUNT=0
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
          BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
      ENDIF
      GO TO 51
    ENDIF
    GO TO 1
57      CONTINUE
        ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
        TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
        IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'I/O STATUS:',MSGBUF
        STOP
C11      FORMAT(IX,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C        K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,)
C        I1SETUP3,XVAL(4);,,)
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34011'O/
        I=0
        IOLINE=1
        DO 100 IX=1,NUMB
          I=I+1
          IF(I.EQ.512)THEN
            BYTE(1)=BINPUT(IX)
            OUT(I,IOLINE)=BYTES
            WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
            OUT(I+1,IOLINE)=SLU
            WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
            I=0
            IOLINE=IOLINE+1
            GO TO 100
          ENDIF
          BYTE(1)=BINPUT(IX)

```

[AVA]FAVAMENT

```
C      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
34     WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
100    FORMAT(1X,I3,1X,I3,2X,O6)
      CONTINUE
      RETURN
      END
```

APPENDIX R  
[AVA]FAVAMEMT2

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),PDATA(2),DATAIN,DATAINA(4)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'0,'140001'0,'121000'0,'107777'0,'17777'0,
1 '24061'0,'26002'0,'30000'0,'44000'0,'64777'0,'120000'0,
2 '50001'0,'70776'0,'54000'0/
DATA ISETUP2/'64777'0,'44000'0/
DATA ISETUP3/'64776'0,'44000'0/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'0
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,,
1 BUF(1),XVAL(28),...,)
ITESTN=1
IERRORC=0
DATAINA(1)='125'0
DATAINA(2)=0
DATAINA(3)='252'0
DATAINA(4)='377'0
DATAIN='125'0
DATAIN=DATAINA(ITESTN)
DO I=1,32768
```

[AVAJFAVAMEMT2

```

BINPUT(1)=DATAIN
ENDDO
177 Y=6
X=0
ICOUNT=0
TITLE='FOUR FIELD WRITE TO AVA'
C CALL TIMRB
I ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SWRITEVBLK)),
IOSB,...
C IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C WRITE (4,54)BINPUT
54 FORMAT(1X,16(1X,03))

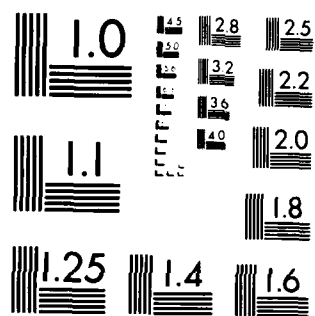
C NUMB=32768
C CALL BUFFCNVT(NUMB,BINPUT,OUT)
C TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 1BOUT(1),XVAL(65534),...,)
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 1BOUT(65535),XVAL(130),...,)
C IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
Y=Y+32
ICOUNT=ICOUNT+1
IF(ICOUNT.EQ.4)THEN
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
C 1IOSB,...
C IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C NUMB=8192
C CALL BUFFCNVT(NUMB,BINPUT,OUT)
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 1BOUT(1),XVAL(8192),...,)
C K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 11SETUP3(1),XVAL(4),...,)
Y='206'0
X=0
ENDIF
IF(ICOUNT.EQ.8)THEN
C ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10SREADVBLK)),
C 1IOSB,...
C IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C NUMB=8192
C CALL BUFFCNVT(NUMB,BINPUT,OUT)
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 1BOUT(1),XVAL(8192),...,)
C K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),IOSB,...
C 11SETUP2(1),XVAL(4),...,)
Y='406'0

```

2/2

NL

END  
DATE  
FILMED  
10-84  
DTIC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS 1963-A

[AVA]FAVAMENT2

```

        ENDIF
        IF(ICOUNT.EQ.16)THEN
            Y='686'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
CCCCCCCCCCCC READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME CCCCCC
C
            ICOUNT=8
            Y=6
            X=8
51      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
            IOSB,,,
            IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
            IF(AVACSR.EQ.8)AVACSR=1
            DO IADD=1,32768
            IF(BINPUT(IADD).NE.DATAIN)THEN
                IADD=IAND(IADD,'777'O)
                WRITE(6,12)DATAIN,BINPUT(IADD),
                IAND(IADD,'777'O)
                IERROR(IADD)=IERROR(IADD)+1
                IERRORC=IERRORC+1
            ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',03,2X,'OUTPUT= ',03,
            15X,'COLUMN= ',15)
            ENDDO
C      TYPE *, ' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT

            Y=Y+32
            ICOUNT=ICOUNT+1
            IF(ICOUNT.EQ.4)THEN
56      WRITE(6,56)IERRORC,DATAIN
            FORMAT(1X,' ERROR COUNT IN FIELD 1=',15,' DATA IN= ',03)
            IERRORC=8
            Y='286'O
            X=8
            ENDIF
            IF(ICOUNT.EQ.8)THEN
157     WRITE(6,157)IERRORC,DATAIN
            FORMAT(1X,' ERROR COUNT IN FIELD 2=',15,' DATA IN= ',03)
            IERRORC=8
            Y='486'O
            ENDIF
            IF(ICOUNT.EQ.16)THEN
            Y='686'O
58      WRITE(6,58)IERRORC,DATAIN
            FORMAT(1X,' ERROR COUNT IN FIELD 3=',15,' DATA IN= ',03)
            IERRORC=8
            ENDIF
            IF(ICOUNT.EQ.24)THEN
59      WRITE(6,59)IERRORC,DATAIN
            FORMAT(1X,' ERROR COUNT IN FIELD 4=',15,' DATA IN= ',03)
            IERRORC=8
            DO IADD=1,512
            IF(IERROR(IADD).NE.8)WRITE(6,233)IADD,

```

CAVA1FAVAMEMT2

```

233      IERROR(IADD),DATAIN
        FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=0
        DO IADD=1,512
          IERROR(IADD)=0
        ENDDO
        ICOUNT=0
        V=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
          BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
      ENDIF
      GO TO 51
    ENDIF
    GO TO 1
57      CONTINUE
        ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
        TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
        IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'I/O STATUS:',MSGBUF
        STOP
C11      FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C        K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(%LOC(IOSWRITEVBLK)),IOSB,,
C        I1SETUP3,XVAL(4),,,)
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34011'O/
        I=0
        IOLINE=1
        DO 100 IX=1,NUMB
          I=I+1
          IF(I.EQ.512)THEN
            BYTE(1)=BINPUT(IX)
            OUT(I,IOLINE)=BYTES
C          WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
            OUT(I+1,IOLINE)=SLU
C          WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
            I=0
            IOLINE=IOLINE+1
            GO TO 100
          ENDIF
          BYTE(1)=BINPUT(IX)

```

[AVAJFAVAMENT2

```
C      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
34     WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
188    FORMAT(1X,I3,1X,I3,2X,O6)
      CONTINUE
      RETURN
      END
```

APPENDIX S  
[AVAJAVAMENT

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(255),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
BYTE BYTEDAT
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'125545'O,'145551'O,'121555'O,'157777'O,'17777'O,
1 '24561'O,'26552'O,'35555'O,'44555'O,'64777'O,'125555'O,
2 '55551'O,'75776'O,'54555'O/
DATA ISETUP2/'64777'O,'44555'O/
DATA ISETUP3/'64776'O,'44555'O/
I = SYSSASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA5',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=5
AVAACR='435'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(%LOC(IOSWRITEVBLK)),IOSB,,,
IBUF(1),XVAL(28),,,,)
ITESTN=1
IERRORC=5
TYPE *,' ENTER MEMORY DATA FOR TESTING'
READ(5,78)BYTEDAT
FORMAT(O3)
DATAINA(1)=BYTEDAT
DATAINA(2)=BYTEDAT
DATAINA(3)=BYTEDAT
```

78

[AVA]AVAMEMT

```

C      DATAINA(4)=BYTEDAT
      DATAIN='125'O
      DATAIN=DATAINA( ITESTN)
      DO I=1,32768
      BINPUT(I)=DATAIN
177    ENDDO
      Y=6
      X=8
      ICOUNT=8
      TITLE='FOUR FIELD WRITE TO AVA'
C      CALL TIMRB
1      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC( IOSWRITEVBLK)),
      IOSB,...)
C      IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      IF(AVACSR.EQ.8)AVACSR=1
      IF(.NOT. ISTATUS.OR..NOT. IOSB(1))GO TO 57

C      WRITE (4,54)BINPUT
54     FORMAT(1X,16(1X,03))

C      NUMB=32768
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *, 'NUMBER OF LINES TO OUTPUT=', IOLINE
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC( IOSWRITEVBLK)),IOSB,...)
C      IBOUT(1),XVAL(65534),...)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC( IOSWRITEVBLK)),IOSB,...)
C      IBOUT(65535),XVAL(138),...)
C      IF(.NOT. ISTATUS.OR..NOT. IOSB(1))GO TO 57
      Y=Y+32
      ICOUNT=ICOUNT+1
      IF( ICOUNT.EQ.4)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC( IOSREADVBLK)),
C      IOSB,...)
C      IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC( IOSWRITEVBLK)),IOSB,...)
C      IBOUT(1),XVAL(8192),...)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC( IOSWRITEVBLK)),IOSB,...)
C      ISETUP3(1),XVAL(4),...)
      Y='286'O
      X=8
      ENDIF
      IF( ICOUNT.EQ.8)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC( IOSREADVBLK)),
C      IOSB,...)
C      IINPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC( IOSWRITEVBLK)),IOSB,...)

```

[AVAJAVAMENT

```

C      1BOUT(1),XVAL(8192),,,,
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(10SWRITEVBLK)),10SB,,,
C      11SETUP2(1),XVAL(4),,,,
      Y='406'O
      ENDIF
      IF(1COUNT.EQ.16)THEN
      Y='606'O
      ENDIF
      IF(1COUNT.EQ.24)THEN
CCCCCCCCC READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME CCCCC
C
      1COUNT=0
      Y=6
      X=0
51      1STATUS=SYSSQIO(XVAL(1),XVAL(1TCHAN),XVAL(XLOC(10SREADVBLK)),
      110SB,,,
      1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      IF(AVACSR.EQ.0)AVACSR=1
      DO 1ADD=1,32768
      IF(1INPUT(1ADD).NE.DATAIN)THEN
      11ADD=1AND(1ADD,'777'O)
C      WRITE(6,12)DATAIN,1INPUT(1ADD),
C      11AND(1ADD,'777'O)
      1ERROR(11ADD)=1ERROR(11ADD)+1
      1ERRORC=1ERRORC+1
      ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',03,2X,'OUTPUT= ',03,
      15X,'COLUMN= ',15)
      ENDDO
C      TYPE *, ' ERROR COUNT AFTER FIELD ***=',1ERRORC,1COUNT

      Y=Y+32
      1COUNT=1COUNT+1
      IF(1COUNT.EQ.4)THEN
      WRITE(6,56)1ERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',15,' DATA IN= ',03)
      1ERRORC=0
      Y='206'O
      X=0
      ENDIF
      IF(1COUNT.EQ.8)THEN
      WRITE(6,157)1ERRORC,DATAIN
157      FORMAT(1X,' ERROR COUNT IN FIELD 2=',15,' DATA IN= ',03)
      1ERRORC=0
      Y='406'O
      ENDIF
      IF(1COUNT.EQ.16)THEN
      Y='606'O
      WRITE(6,58)1ERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',15,' DATA IN= ',03)
      1ERRORC=0
      ENDIF
      IF(1COUNT.EQ.24)THEN
      WRITE(6,59)1ERRORC,DATAIN

```

[AVAJAVAMENT

```

59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',15,' DATA IN= ',03)
      IERRORC=0
      DO IADD=1,512
      IF(IERROR(IADD).NE.0)WRITE(6,233)IADD,
233      IERROR(IADD),DATAIN
      FORMAT(1X,'COLUMN',14,' ERRORS= ',16,' DATA IN= ',03)
      ENDDO
      IERRORC=0
      DO IADD=1,512
      IERROR(IADD)=0
      ENDDO
      ICOUNT=0
      Y=6
      ITESTN=ITESTN+1
      IF(ITESTN.GT.4)ITESTN=1
      DATAIN=DATAINA(ITESTN)
      DO I=1,32768
      BINPUT(I)=DATAIN
      ENDDO
      GO TO 177
      ENDIF
      GO TO 51
      ENDIF
      GO TO 1
57      CONTINUE
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
      TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
      TYPE *, 'I/O STATUS:',MSGBUF
      STOP
C11     FORMAT(1X,' INPUT=',06,2X,' IOSB=',06,2X,06,2X,06,2X,06)
C       K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
C       ISETUP3,XVAL(4),,,)
      END
      SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
      BYTE BINPUT(1),BYTE(2)
      INTEGER*2 OUT(513,1),BYTES,SLU
      EQUIVALENCE(BYTES,BYTE)
      DATA SLU/'34011'O/
      I=0
      IOLINE=1
      DO 100 IX=1,NUMB
      I=I+1
      IF(I.EQ.512)THEN
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=0

```

[AVA]AVAMENT

```
      IOLINE=IOLINE+1
      GO TO 100
    ENDIF
    BYTE(1)=BINPUT(IX)
    OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34     FORMAT(1X,I3,1X,I3,2X,O6)
100    CONTINUE
      RETURN
      END
```

APPENDIX T  
[AVA]AVAMENT2

```
EXTERNAL IOSWRITEBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSS$ASSIGN, SYSS$QIOW, CHAN,SYSS$QIO,SYSS$WAITFR
INTEGER SYSS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
BYTE BYTEDAT
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSS$ASSIGN('GRAB',CHAN,,)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('AVA0',ITCHAN,,)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEBLK)),IOSB,...
1BUF(1),XVAL(28),,...)
ITESTN=1
IERRORC=0
TYPE *, ' ENTER MEMORY DATA FOR TESTING'
READ(5,78)BYTEDAT
FORMAT(O3)
DATAINA(1)=BYTEDAT
DATAINA(2)=BYTEDAT
DATAINA(3)=BYTEDAT
```

78

[AVA]AVAMENT2

```

C      DATAINA(4'-BYTEDAT
C      DATAIN='125'O
C      DATAIN=DATAINA(1TESTN)
C      DO I=1,32768
C      BINPUT(I)=DATAIN
177    ENDDO
C      Y=6
C      X=8
C      ICOUNT=8
C      TITLE='FOUR FIELD WRITE TO AVA'
C      CALL TIMRB
1      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
C      1IOSB,...)
C      1INPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      IF(AVACSR.EQ.8)AVACSR=1
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C
C      WRITE (4,54)BINPUT
54    FORMAT(1X,16(1X,03))
C
C      NUMB=32768
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C      1BOUT(1),XVAL(65534),...)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C      1BOUT(65535),XVAL(138),...)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
C      Y=Y+32
C      ICOUNT=ICOUNT+1
C      IF(ICOUNT.EQ.4)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C      1IOSB,...)
C      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C      1BOUT(1),XVAL(8192),...)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C      11SETUP3(1),XVAL(4),...)
C      Y='286'O
C      X=8
C      ENDIF
C      IF(ICOUNT.EQ.8)THEN
C      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
C      1IOSB,...)
C      1INPUT,XVAL(8192),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C      NUMB=8192
C      CALL BUFFCNVT(NUMB,BINPUT,OUT)
C      ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C      1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)

```

[AVA]AVAMENT2

```

C      18OUT(1),XVAL(8192),...)
C      K = SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,,
C      I1SETUP2(1),XVAL(4),...)
      Y='406'O
      ENDIF
      IF(ICOUNT.EQ.16)THEN
      Y='606'O
      ENDIF
      IF(ICOUNT.EQ.24)THEN
CCCCCCCCC READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME CCCCC
C
      ICOUNT=0
      Y=6
      X=0
51      ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      IOSB,,
      I1INPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
      IF(AVACSR.EQ.0)AVACSR=1
      DO IADD=1,32768
      IF(BINPUT(IADD).NE.DATIN)THEN
      IADD=IAND(IADD,'777'O)
      WRITE(6,12)DATIN,BINPUT(IADD),
      IAND(IADD,'777'O)
      IERROR(IADD)=IERROR(IADD)+1
      IERRORC=IERRORC+1
      ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',03,2X,'OUTPUT= ',03,
      15X,'COLUMN= ',15)
      ENDDO
C      TYPE *, ' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT

      Y=Y+32
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.4)THEN
      WRITE(6,56)IERRORC,DATIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',15,' DATA IN= ',03)
      IERRORC=0
      Y='206'O
      X=0
      ENDIF
      IF(ICOUNT.EQ.8)THEN
      WRITE(6,157)IERRORC,DATIN
157      FORMAT(1X,' ERROR COUNT IN FIELD 2=',15,' DATA IN= ',03)
      IERRORC=0
      Y='406'O
      ENDIF
      IF(ICOUNT.EQ.16)THEN
      Y='606'O
      WRITE(6,58)IERRORC,DATIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',15,' DATA IN= ',03)
      IERRORC=0
      ENDIF
      IF(ICOUNT.EQ.24)THEN
      WRITE(6,59)IERRORC,DATIN

```

CAVAJAVAMEMT2

```

59      FORMAT(IX,' ERROR COUNT IN FIELD 4=',I5,' DATA IN= ',O3)
        IERRC=0
        DO IADD=1,512
          IF(IERROR(IADD).NE.0)WRITE(6,233)IADD,
1      IERROR(IADD),DATAIN
233     FORMAT(IX,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRC=0
        DO IADD=1,512
          IERROR(IADD)=0
        ENDDO
        ICOUNT=0
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
          BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
      ENDIF
      GO TO 51
    ENDIF
    GO TO 1
57      CONTINUE
        ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
        TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
        IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF= ' '
        ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
        TYPE *, 'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(IX,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEBLK)),IOSB,,)
C       I1SETUP3,XVAL(4),,,)
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34011'O/
        I=0
        IOLINE=1
        DO 100 IX=1,NUMB
          I=I+1
          IF(I.EQ.512)THEN
            BYTE(1)=BINPUT(IX)
            OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
            OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
          I=0

```

[AVA]AVAMENT2

```
      IOLINE=IOLINE+1
      GO TO 188
    ENDIF
    BYTE(1)=BINPUT(IX)
    OUT(1,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) 1,IOLINE,OUT(1,IOLINE)
34     FORMAT(IX,I3,1X,I3,2X,O6)
188    CONTINUE
      RETURN
      END
```

APPENDIX U  
[AVAJAAVAMEMT

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      THIS PROGRAM TESTS ALL OF THE AVA MEMORY (NOT JUST WHERE VIDEO IS STORED)
C      UP TO Y='777'O AND X='1777'O
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER SYSSASSIGN, SYSSQIOW, CHAN,SYSSQIO,SYSSWAITFR
INTEGER SYSSGETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYSSASSIGN('GRAB',CHAN,..)
IF(.NOT. I)TYPE *, ' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYSSASSIGN('AVA0',ITCHAN,..)
IF(.NOT. ISTATUS)TYPE *, ' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYSSQIOW(XVAL(1),XVAL(CHAN),XVAL(XLOC(IOSWRITEVBLK)),IOSB,..
1BUF(1),XVAL(28),...)
ITESTN=1
IERRORC=0
DATAINA(1)='125'O

```

[AVA]AAVAMEMT

```

DATAINA(2)=0
DATAINA(3)='252'0
DATAINA(4)='377'0
DATAIN='125'0
DATAIN=DATAINA(ITESTN)
DO I=1,32768
BINPUT(I)=DATAIN
ENDDO
177 Y=0
X=0
ICOUNT=0
TITLE='FOUR FIELD WRITE TO AVA'
C CALL TIMRB
1 ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
IOSB,...)
C IINPUT,XVAL(NBYTES),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
C IF(AVACSR.EQ.0)AVACSR=1
C IF(.NOT. ISTATUS.OR..NOT.IOSB(1))GO TO 57

C WRITE (4,54)BINPUT
54 FORMAT(1X,16(1X,03))

C NUMB=32768
C CALL BUFFCNVT(NUMB,BINPUT,OUT)
C TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C IBOUT(1),XVAL(65534),...,)
C ISTATUS = SYSSQIO(XVAL(1),XVAL(CHAN),
C 1XVAL(XLOC(IOSWRITEVBLK)),IOSB,...)
C IBOUT(65535),XVAL(130),...,)
C IF(.NOT. ISTATUS.OR..NOT.IOSB(1))GO TO 57
C Y=Y+32
IF(Y.GT.'777'0)THEN
CCCCCCCCCCCC READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME CCCCCC
C
ICOUNT=0
Y=0
X=0
51 ISTATUS=SYSSQIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
IOSB,...)
IINPUT,XVAL(32768),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(AVACSR.EQ.0)AVACSR=1
DO IADD=1,32768
IF(BINPUT(IADD).NE.DATAIN)THEN
IADD=IAND(IADD,'777'0)
C WRITE(6,12)DATAIN,BINPUT(IADD),
C IAND(IADD,'777'0)
IERROR(IADD)=IERROR(IADD)+1
IERRORC=IERRORC+1
ENDIF
12 FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',03,2X,'OUTPUT= ',03,
15X,'COLUMN= ',15)
ENDDO

```

[AVA]AAVAMENT

```

C      TYPE *, ' ERROR COUNT AFTER FIELD ***=', IERRORC, ICOUNT

      Y=Y+32
      IF(Y.GT.'777'0)THEN
        WRITE(6,59)IERRORC,DATAIN
59      FORMAT(1X,' ERROR COUNT =',I5,' DATA IN= ',03)
        IERRORC=0
        DO IADD=1,512
          IF(IERROR(IADD).NE.0)WRITE(6,233)IADD,
            IERROR(IADD),DATAIN
233      FORMAT(1X,' COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',03)
        ENDDO
        IERRORC=0
        DO IADD=1,512
          IERROR(IADD)=0
        ENDDO
        ICOUNT=0
        Y=0
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAIN(ITESTN)
        DO I=1,32768
          BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
      ENDIF
      GO TO 51
    ENDIF
    GO TO 1
57    CONTINUE
      ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
      TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'I/O STATUS:',MSGBUF
      STOP
C11    FORMAT(1X,' INPUT=',06,2X,' IOSB=',06,2X,06,2X,06,2X,06)
C      K = SYSS$QIOW(XVAL(1),XVAL(CHAN),XVAL(%LOC(IOSWRITEVBLK)),IOSB,...
C      I$SETUP3,XVAL(4),...)
      END
      SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
      BYTE BINPUT(1),BYTE(2)
      INTEGER*2 OUT(513,1),BYTES,SLU
      EQUIVALENCE(BYTE,BYTE)
      DATA SLU/'34011'0/
      I=0
      IOLINE=1
      DO 100 IX=1,NUMB
      I=I+1
      IF(I.EQ.512)THEN

```

[AVA]AAVAMENT

```
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=BYTES
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
      OUT(I+1,IOLINE)=SLU
C      WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
      I=8
      IOLINE=IOLINE+1
      GO TO 188
    ENDIF
      BYTE(1)=BINPUT(IX)
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C      WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34     FORMAT(1X,I3,1X,I3,2X,O6)
188    CONTINUE
      RETURN
      END
```

# APPENDIX V

[AVAIAUX

```

EXTERNAL IOSREADVBLK,IOSWRITEVBLK
BYTE IX(100),IY(100),N,IFLAG
INTEGER*2 D(100),X,Y,IOSB(4)
INTEGER*2 OUT(16),US,TS,UM,TM,UH21
DIMENSION VOLTS(16)
BYTE BD(200),TEMP,IMAGE8(4)
INTEGER*2 UH84,UH,TH,UD,TD,HD
CHARACTER*80 MSGBUF
INTEGER*2 TEMS,MS,HMS,TMS
INTEGER AVACR
INTEGER AVACSR,SYSSASSIGN,SYSSGETMSG,SYSSQIOW,SYSSQIO
EQUIVALENCE(D,BD)
ISTATUS=SYSSASSIGN('AVA0',IAVAC,,)
IF(.NOT.ISTATUS)THEN
  TYPE *,'AVA CHANNEL ASSIGN ERROR'
  STOP
ENDIF
IFLAG=1
AVAACR='415'0
AVACSR=0
Y=1
X=0
ISTATUS=SYSSQIOW(XVAL(1),XVAL(IAVAC),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,)
1D,XVAL(36),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVACR))
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
    ISTATUS=SYSSGETMSG(XVAL(ISTATUS),MSGLEN,MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO SGETMSG'
    TYPE *,MSGBUF
  MSGBUF=' '
    ISTATUS=SYSSGETMSG(XVAL(IOSB(1)),MSGLEN,MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO SGETMSG'
    TYPE *,MSGBUF
    STOP
  ENDIF
  IF(AVACSR.EQ.0)AVACSR=1
  DO I=3,35,2
    TEMP=BD(I)
    BD(I)=BD(I+1)
    BD(I+1)=TEMP
  
```

3

[AVAJAUX

```

C      ENDDO
111    WRITE(6,111)(D(I),I=1,16)
      FORMAT(1X,16(1X,Z4))
      IF(IFLAG)THEN
        IFLAG=#
        IIMAGEB(1)=27   IESC
        IIMAGEB(2)=72   IH      CURSOR HOME
        IIMAGEB(3)=27   IESC
        IIMAGEB(4)=74   IJ      ERASE TO END OF SCREEN
        WRITE(6,77)IIMAGEB
        IIMAGEB(1)=27   IESC
        IIMAGEB(2)=89   IY
        IIMAGEB(4)=32   ICOLUMN
        IIMAGEB(3)=37   ILINE
177    WRITE(6,177)IIMAGEB
      FORMAT(1H+,4A1,' CHANNEL      OCTAL      HEX',
        1'      VOLTS')
      ENDIF
      DO I=2,17
        ICHAN=IAND(NOT(ISHFT(D(I),-12))-1,'F'X)
        IS=IAND(ISHFT(D(I),-11),1)
        IF(IS.EQ.#)THEN
          II=FLOAT(D(I))+1.
        ELSE
          II=D(I)
        ENDIF
        OUT(ICHAN+1)=IAND(II,'7777'O)
        VOLTS(ICHAN+1)=FLOAT(IAND(II,'3777'O))/2#47.
        VOLTS(ICHAN+1)=VOLTS(ICHAN+1)*1#.
        IF(IS.NE.#)VOLTS(ICHAN+1)=-VOLTS(ICHAN+1)
        IF(VOLTS(ICHAN+1).GT.#.)VOLTS(ICHAN+1)=ABS(VOLTS(ICHAN+1)-1#.)
      ENDDO
      DO I=1,16
        IIMAGEB(3)=I+39 ILINE
C      WRITE(6,77)IIMAGEB,(OUT(I),I=1,16)
      WRITE(6,77)IIMAGEB,I,NOT(OUT(I)),NOT(OUT(I)),VOLTS(I)
C77    FORMAT(1H+,4A1,16(Z4,1X))
77     FORMAT(1H+,4A1,I5,1#X,06,'      =      ',Z4,1#X,F6.2)
      ENDDO
      GO TO 3
      END

```

# APPENDIX W

[AVA]AUX2

```

EXTERNAL IOSREADVBLK,IOSWRITEVBLK
BYTE IX(100),IY(100),N,IFLAG
INTEGER*2 D(100),X,Y,IOSB(4)
INTEGER*2 US,TS,UM,TM,UH21
BYTE BD(200),TEMP
INTEGER*2 UH84,UH,TH,UD,TD,HD
CHARACTER*80 MSGBUF
INTEGER*2 TEMS,MS,HMS,TMS
INTEGER AVAACR
INTEGER AVACSR,SYSS$ASSIGN,SYSS$GETMSG,SYSS$QIOW,SYSS$QIO
EQUIVALENCE(D,BD)
ISTATUS=SYSS$ASSIGN('AVA0',IAVAC,,)
IF(.NOT.ISTATUS)THEN
  TYPE *, 'AVA CHANNEL ASSIGN ERROR'
  STOP
ENDIF
IFLAG=1
AVAACR='415'0
AVACSR=0
Y=1
X=0
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(IAVAC),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,
1D,XVAL(36),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
    ISTATUS=SYSS$GETMSG(XVAL(ISTATUS),MSGLEN,MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *,MSGBUF
MSGBUF=' '
    ISTATUS=SYSS$GETMSG(XVAL(IOSB(1)),MSGLEN,MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *,MSGBUF
    STOP
  ENDIF
IF(AVACSR.EQ.0)AVACSR=1
DO I=3,35,2
  TEMP=BD(I)
  BD(I)=BD(I+1)
  BD(I+1)=TEMP
ENDDO

```

CAVAJAUX2

```
111  WRITE(6,111)(D(I),I=2,17)
      FORMAT(1X,16(1X,Z4))
      GO TO 3
      END
```

APPENDIX X  
[AVA]AUXPLOT

```

EXTERNAL IOSREADVBLK,IOSWRITEVBLK
BYTE IX(100),IY(100),N,IFLAG
INTEGER*2 D(260),X,Y,IOSB(4)
INTEGER*2 OUT(16),US,TS,UM,TM,UH21
DIMENSION VOLTS(16)
BYTE BD(520),TEMP,IIMAGEB(4)
INTEGER*2 UH04,UH,TH,UD,TD,HD
CHARACTER*80 MSGBUF
INTEGER*2 TEMS,MS,HMS,TMS
INTEGER AVAACR
INTEGER AVACSR,SYSSASSIGN,SYSSGETMSG,SYSSQIOW,SYSSQIO
EQUIVALENCE(D,BD)
ISTATUS=SYSSASSIGN('AVA0',IAVAC,,)
IF(.NOT.ISTATUS)THEN
  TYPE *, 'AVA CHANNEL ASSIGN ERROR'
  STOP
ENDIF
IFLAG=1
AVAACR='415'0
AVACSR=0
TYPE *, 'ENTER CHANNEL TO BE PLOTTED. (1...16)'
ACCEPT*,IWCHAN
Y=1
X=0
ISTATUS=SYSSQIOW(XVAL(1),XVAL(IAVAC),XVAL(XLOC(IOSREADVBLK)),
IOSB,...
1D,XVAL(520),XVAL(X),XVAL(Y),XVAL(AVACSR),XVAL(AVAACR))
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
  TYPE *,MSGBUF
MSGBUF=' '
  ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
  TYPE *,MSGBUF
  STOP
ENDIF
IF(AVACSR.EQ.0)AVACSR=1
DO I=3,519,2
  TEMP=BD(I)

```

3

CAVAJAUXPLOT

```

      BD(I)=BD(I+1)
      BD(I+1)=TEMP
      ENDDO
C
C      THE FIRST 16 BITS WORD IS TRASH.
C
      IRESET=1
      DO I=2,256,15
C      WRITE(6,111)(D(J),J=I,I+15)
      DO IJ=I,I+15
      ICHAN=IAND(NOT(ISHFT(D(IJ),-12))-1,'F'X)
      IS=IAND(ISHFT(D(IJ),-11),1)
      IF(IS.EQ.0)THEN
      II=FLOAT(D(IJ))+1.
      ELSE
      II=D(IJ)
      ENDIF
      OUT(ICHAN+1)=IAND(II,'7777'0)
      VOLTS(ICHAN+1)=FLOAT(IAND(II,'3777'0))/2047.
      VOLTS(ICHAN+1)=VOLTS(ICHAN+1)*10.
      IF(IS.NE.0)VOLTS(ICHAN+1)=-VOLTS(ICHAN+1)
      IF(VOLTS(ICHAN+1).GT.0)VOLTS(ICHAN+1)=ABS(VOLTS(ICHAN+1)-10.)
      ENDDO
C      TYPE *, 'VOLTS(16)=',VOLTS(16)
      CALL AVT52P(VOLTS,IWCHAN,IRESET)
      ENDDO
111      IRESET=1
      FORMAT(1X,16(1X,Z4))
      GO TO 3
      END
      SUBROUTINE AVT52P(VOLTS,IWCHAN,IRESET)
C
C      THIS IS AUXILIARY DATA VT52 PLOT SUBROUTINE
C
C      IWCHAN IS THE CHANNEL TO BE PLOTTED
C      VOLTS IS THE VOLTAGE
C
      BYTE IIMAGEB(4)
      DIMENSION VOLTS(16)
      DATA IFIRST/1/,IXPOS/32/
      IF(IRESET)THEN
      IXPOS=32
      IRESET=0
      IFIRST=1
      ENDIF
      IF(IFIRST)THEN
      IFIRST=0
      IIMAGEB(1)=27      IESC
      IIMAGEB(2)=72      IH      CURSOR HOME
      IIMAGEB(3)=27      IESC
      IIMAGEB(4)=74      IJ      ERASE TO END OF SCREEN
      WRITE(6,77)IIMAGEB
77      FORMAT(1H+,4A1)
      IIMAGEB(1)=27      IESC
      IIMAGEB(2)=89      IV

```

CAVAJAUXPLOT

```

      IIMAGEB(4)=32   ICOLUMN
      IIMAGEB(3)=44   ILINE
      WRITE(6,177)IIMAGEB
177   FORMAT(1H+,4A1,
      1'-----',
      1'-----')
      IIMAGEB(1)=27   IESC
      IIMAGEB(2)=89   IY
      IIMAGEB(4)=32   ICOLUMN
      DO I=32,56
      IIMAGEB(3)=I     ILINE
178   WRITE(6,178)IIMAGEB
      FORMAT(1H+,4A1,'+')
      ENDDO
      ENDIF
      IIMAGEB(4)=IXPOS
      Y=24.*((VOLTS(IWCHAN)+10.)/20.)
      IIMAGEB(3)=32.+24.-Y
      WRITE(6,179)IIMAGEB
C     TYPE*,'X,Y=',IIMAGEB(4),IIMAGEB(3)
179   FORMAT(1H+,4A1,'*')
      IXPOS=IXPOS+1
      IF (IXPOS.GE.120)THEN
      IXPOS=32
      IFIRST=1
      ENDIF
      RETURN
      END

```

APPENDIX V  
[AVA]AUXPLOT

```

EXTERNAL IOSREADVBLK, IOSWRITEVBLK
BYTE IX(100), IY(100), N, IFLAG
INTEGER*2 D(260), X, Y, IOSB(4)
INTEGER*2 OUT(16), US, TS, UM, TM, UH21
DIMENSION VOLTS(16)
BYTE BD(520), TEMP, IIMAGEB(4)
INTEGER*2 UH84, UH, TH, UD, TD, HD
CHARACTER*80 MSGBUF
INTEGER*2 TEMS, MS, HMS, TMS
INTEGER AVAOCR
INTEGER AVACSR, SYSS$ASSIGN, SYSS$GETMSG, SYSS$QIOW, SYSS$QIO
EQUIVALENCE(D, BD)
ISTATUS=SYSS$ASSIGN('AVA0', IAVAC, .)
IF(.NOT. ISTATUS) THEN
  TYPE *, 'AVA CHANNEL ASSIGN ERROR'
  STOP
ENDIF
IFLAG=1
AVAOCR='415'0
AVACSR=0
Y=1
X=0
ISTATUS=SYSS$QIOW(XVAL(1), XVAL(IAVAC), XVAL(XLOC(IOSREADVBLK)),
  1IOSB, .,
  1D, XVAL(520), XVAL(X), XVAL(Y), XVAL(AVACSR), XVAL(AVAOCR))
  IF(.NOT. ISTATUS.OR. .NOT. IOSB(1)) THEN
    ISTATUS=SYSS$GETMSG(XVAL(ISTATUS), MSGLEN, MSGBUF, .)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *, MSGBUF
  MSGBUF=' '
    ISTATUS=SYSS$GETMSG(XVAL(IOSB(1)), MSGLEN, MSGBUF, .)
    IF(.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
    TYPE *, MSGBUF
    STOP
  ENDIF
  IF(AVACSR.EQ.0) AVACSR=1
  DO I=3, 519, 2
    TEMP=BD(I)
    BD(I)=BD(I+1)
    BD(I+1)=TEMP
  
```

3

CAVAJAUXPLOT

```

      ENDDO
C      THE FIRST 16 BITS WORD IS TRASH.
C
      IRESET=1
      DO I=2,80,15
C      WRITE(6,111)(D(I),I=1,I+15)
      DO IJ=1,I+15
      ICHAN=IAND(NOT(ISHFT(D(IJ),-12))-1,'F'X)
      IS=IAND(ISHFT(D(IJ),-11),1)
      IF(IS.EQ.0)THEN
      II=FLOAT(D(IJ))+1.
      ELSE
      II=D(IJ)
      ENDIF
      OUT(ICCHAN+1)=IAND(II,'7777'O)
      VOLTS(ICCHAN+1)=FLOAT(IAND(II,'3777'O))/2047.
      VOLTS(ICCHAN+1)=VOLTS(ICCHAN+1)*10.
      IF(IS.NE.0)VOLTS(ICCHAN+1)=-VOLTS(ICCHAN+1)
      IF(VOLTS(ICCHAN+1).GT.0)VOLTS(ICCHAN+1)=ABS(VOLTS(ICCHAN+1)-10.)
      ENDDO
C      TYPE *, 'VOLTS(16)=',VOLTS(16)
      CALL AVT52P(VOLTS,IWCHAN,IRESET)
      ENDDO
      IRESET=1
111  FORMAT(1X,16(1X,Z4))
      GO TO 3
      END
      SUBROUTINE AVT52P(VOLTS,IWCHAN,IRESET)
C
C      THIS IS AUXILIARY DATA VT52 PLOT SUBROUTINE
C
C      IWCHAN IS THE CHANNEL TO BE PLOTTED
C      VOLTS IS THE VOLTAGE
C
      BYTE IIMAGEB(4)
      DIMENSION VOLTS(16)
      DATA IFIRST/1/,IXPOS/32/
      IF(IRESET)THEN
      IXPOS=32
      IRESET=0
      IFIRST=1
      ENDIF
      IF(IFIRST)THEN
      IFIRST=0
      IIMAGEB(1)=27  IESC
      IIMAGEB(2)=72  IH      CURSOR HOME
      IIMAGEB(3)=27  IESC
      IIMAGEB(4)=74  IJ      ERASE TO END OF SCREEN
      WRITE(6,77)IIMAGEB
77  FORMAT(1H+,4A1)
      IIMAGEB(1)=27  IESC
      IIMAGEB(2)=89  IY
      IIMAGEB(4)=32  ICOLUMN
      IIMAGEB(3)=44  ILINE

```

[AVAJAUXPLOTA

```
177 WRITE(6,177)IIMAGEB
   FORMAT(1H+,4A1,
   1'-----',
   1'-----')
   IIMAGEB(1)=27 IESC
   IIMAGEB(2)=89 IV
   IIMAGEB(4)=32 ICOLUMN
   DO I=32,56
   IIMAGEB(3)=I ILINE
178 WRITE(6,178)IIMAGEB
   FORMAT(1H+,4A1,'+')
   ENDDO
   ENDIF
   DO JJ=1,16
   IIMAGEB(4)=IXPOS
   Y=24.*((VOLTS(JJ)+18.)/28.)
   IIMAGEB(3)=32.+24.-Y
   WRITE(6,179)IIMAGEB
C TYPE*, 'X,Y=',IIMAGEB(4),IIMAGEB(3)
179 FORMAT(1H+,4A1,'*')
   IXPOS=IXPOS+1
   IF (IXPOS.GE.128) THEN
   IXPOS=32
   IFIRST=1
   ENDIF
   ENDDO
   RETURN
   END
```

APPENDIX Z  
CAVAJAUXIRIG

```

EXTERNAL IOSREADVBLK, IOSWRITEVBLK
BYTE IX(100), IY(100), N, IFLAG
INTEGER*2 D(100), X, Y
INTEGER*2 US, TS, UM, TM, UH21
INTEGER*2 UH84, UH, TH, UD, TD, HD
BYTE IIMAGEB(4)
INTEGER*2 TEMS, MS, HMS, TMS
INTEGER AVACSR, SYSSASSIGN, SYSSQIOW, SYSSQIO
INTEGER AVAACR
ISTATUS=SYSSASSIGN('AVA0', IAVAC, )
IF(.NOT. ISTATUS) THEN
  TYPE *, 'AVA CHANNEL ASSIGN ERROR'
  STOP
ENDIF
IFLAG=1
AVACSR=0
AVAACR='435'0
IIMAGEB(1)=27 IESC
IIMAGEB(2)=72 IH CURSOR HOME
IIMAGEB(3)=27 IESC
IIMAGEB(4)=74 IJ ERASE TO END OF SCREEN
WRITE(6,77) IIMAGEB
IIMAGEB(1)=27 IESC
IIMAGEB(2)=89 IY
IIMAGEB(4)=32 ICOLUMN
IIMAGEB(3)=37 ILINE
X='1000'0
Y=2
ISTATUS=SYSSQIOW(XVAL(1), XVAL(IAVAC), XVAL(XLOC(IOSREADVBLK)),
  1IOSB, ,
  1D, XVAL(8), XVAL(X), XVAL(Y), XVAL(AVACSR), XVAL(AVAACR))
IF(AVACSR.EQ.0) AVACSR=1
DO I=2,4
  D(I)=NOT(D(I))
ENDDO
HMS=IAND(ISHFT(D(2),-8), 'F'X)
TMS=IAND(ISHFT(D(2),-4), 'F'X)
MS=IAND(D(2), 'F'X)
US=IAND(ISHFT(D(2),-12), 'F'X)
TS=IAND(D(3),7)

```

3

[AVA]AUXIRIG

```

      UM=IAND(ISHFT(D(3),-3),'F'X)
      TM=IAND(ISHFT(D(3),-7),7)
      UH=IAND(ISHFT(D(3),-10),'F'X)
      TH=IAND(ISHFT(D(3),-14),3)
      UD=IAND(D(4),'F'X)
      TD=IAND(ISHFT(D(4),-4),'F'X)
      HD=IAND(ISHFT(D(4),-8),'F'X)

      WRITE(6,13)IIMAGEB,(NOT(D(I)),I=2,4),HD,TD,UD,TH,UH,TM,UM,TS,US,
      IHMS,TMS,MS
77     FORMAT(1H+,4A1,15,10X,06,'      =      ',Z4,10X,F6.2)
13     FORMAT(1H+,4A1,3(1X,06),5X,3Z1,'::',1X,2Z1,'::',Z1,Z1,'::',2Z1,
      1'::',3Z1)
      GO TO 3
      END

```

APPENDIX AA  
[AVA]AUXIRIG2

```

EXTERNAL IOSREADVBLK, IOSWRITEVBLK
BYTE IX(100), IY(100), N, IFLAG
INTEGER*2 D(100), X, Y
INTEGER*2 US, TS, UM, TM, UH21
INTEGER*2 UH84, UH, TH, UD, TD, HD
INTEGER*2 TEMS, MS, HMS, TMS
INTEGER AVACSR, SYSSASSIGN, SYSSQIOW, SYSSQIO
INTEGER AVAACR
ISTATUS=SYSSASSIGN('AVA0', IAVAC, .)
IF(.NOT. ISTATUS) THEN
TYPE *, 'AVA CHANNEL ASSIGN ERROR'
STOP
ENDIF
IFLAG=1
AVACSR=0
AVAACR='435'0
3 X='1000'0
Y=2
ISTATUS=SYSSQIOW(XVAL(1), XVAL(IAVAC), XVAL(XLOC(IOSREADVBLK)),
1IOSB, .,
1D, XVAL(8), XVAL(X), XVAL(Y), XVAL(AVACSR), XVAL(AVAACR))
IF(AVACSR.EQ.0) AVACSR=1
DO I=2, 4
D(I)=NOT(D(I))
ENDDO
HMS=IAND(ISHFT(D(2), -8), 'F'X)
TMS=IAND(ISHFT(D(2), -4), 'F'X)
MS=IAND(D(2), 'F'X)
US=IAND(ISHFT(D(2), -12), 'F'X)
TS=IAND(D(3), 7)
UM=IAND(ISHFT(D(3), -3), 'F'X)
TM=IAND(ISHFT(D(3), -7), 7)
UH=IAND(ISHFT(D(3), -10), 'F'X)
TH=IAND(ISHFT(D(3), -14), 3)
UD=IAND(D(4), 'F'X)
TD=IAND(ISHFT(D(4), -4), 'F'X)
HD=IAND(ISHFT(D(4), -8), 'F'X)

WRITE(6, 13) (NOT(D(I)), I=1, 4), HD, TD, UD, TH, UH, TM, UM, TS, US,
1HMS, TMS, MS

```

[AVA]AUXIRIG2

13      FORMAT(1X,4(1X,06),5X,3Z1,' ',1X,2Z1,' ',Z1,Z1,' ',2Z1,  
             1' ',3Z1)  
         GO TO 3  
         END

APPENDIX AB  
[AVA.TAPEDRIVE]IRIGREAD

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM CONTINUOUSLY READS THE IRIG FROM THE SEARCH UNIT
C   AND DISPLAYS IT ON THE TERMINAL.
C
C .....
C
C   THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C       150001  I TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C       156400  I UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C       157000  I STOP
C       157447  I THE FILTERS ARE SET TO 1 AND 10000 HZ
C .....
C
C   SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER SYSSASSIGN,SYSSQIOW,SYSSQIO
INTEGER SYSSGETMSG
INTEGER*2 IOSB(4),MSGLEN
INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)
INTEGER*2 US,TS,UM,TM,UH21
INTEGER*2 UH84,UH,TH,UD,TD,HD
INTEGER*2 TEMS,MS,HMS,TMS
CHARACTER *80 MSGBUF
DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O,'157201'O/

```

[AVA.TAPEDRIVE]IRIGREAD

```

DATA CONT/'156403'O,'156405'O,'156407'O,'156411'O,
1'156400'O/
ISTATUS=SYSS$ASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN DR11-C CHANNEL ASSIGN'
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
1IOSB,,,
1INIT,XVAL(10),,,,
IF(ISTATUS.AND.IOSB(1)) GO TO 1
TYPE *,'ERROR IN QIOW CALL'
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
TYPE *,'I/O STATUS:',MSGBUF
1
2
45
1CONTINUE
FORMAT(06)
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
1IOSB,,,
1CONT(1CONT),XVAL(2),,,,
IF(ISTATUS) GO TO 500
TYPE *,'ERROR IN QIOW CALL'
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
500
TYPE *,'IOSB(1)=' ,IOSB(1),' IOSB(2)=' ,IOSB(2)
CONTINUE
IF(1CONT.EQ.5)GO TO 501
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,,
1OUTPUT(1CONT),XVAL(2),,,,
IF(ISTATUS) GO TO 501
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
MSBBUG=' '
ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
TYPE *,'I/O STATUS:',MSGBUF
GO TO 2
501
IF(1CONT.EQ.1)THEN
C
C IST4=IAND(OUTPUT(1),'000010'O)
C
C IST5=IAND(OUTPUT(1),'000020'O)
C
C IST6=IAND(OUTPUT(1),'000040'O)
C
C IST7=IAND(OUTPUT(1),'000100'O)
C
C IST8=IAND(OUTPUT(1),'000200'O)
C
C IST9=IAND(OUTPUT(1),'000400'O)
C
C IST10=IAND(OUTPUT(1),'001000'O)
C
C IST11=IAND(OUTPUT(1),'002000'O)
C
C IF(IST4.NE.0)TYPE*,'START TIME FOUND'
C
C IF(IST5.NE.0)TYPE*,'STOP TIME FOUND'
C
C IF(IST6.NE.0)TYPE*,'PLAYBACK CYCLE BEGAN'
C
C IF(IST7.NE.0)TYPE*,'STOPPED'
C
C IF(IST8.NE.0)TYPE*,'PLAYBACK INTERVAL'
C
C IF(IST9.NE.0)TYPE*,'SEARCHING'

```

[AVA.TAPEDRIVE]IRIGREAD

```

C      C      IF(IST10.NE.0)TYPE*,'POWER OFF'
C      C      IF(IST11.NE.0)TYPE*,'REMOTE SELECTED'
      ENDIF
      IF(ICONT.LT.5)THEN
        ICONT=ICONT+1
        GO TO 2
      ENDIF
      ICONT=1
C
C      AFTER ALL STATUS AND IRIG HAVE BEEN READ IN LETS PRINT THEM OUT
C
C      IRIG WORD 1 DIGIT DECODING
      US=IAND(OUTPUT(2),'F'X)
      TS=IAND(ISHFT(OUTPUT(2),-4),'7'X)
      UM=IAND(ISHFT(OUTPUT(2),-7),'F'X)
      TM=IAND(ISHFT(OUTPUT(2),-11),'7'X)
      UH21=ISHFT(OUTPUT(2),-14)
C
C      IRIG WORD 2 DIGIT DECODING
      UH84=ISHFT(IAND(OUTPUT(3),'3'X),2)
      UH=IOR(UH84,UH21)
      TH=IAND(ISHFT(OUTPUT(3),-2),'3'X)
      UD=IAND(ISHFT(OUTPUT(3),-4),'F'X)
      TD=IAND(ISHFT(OUTPUT(3),-8),'F'X)
      HD=ISHFT(OUTPUT(3),-12)
C
C      IRIG WORD 3 DIGIT DECODING
      TEMS=IAND(OUTPUT(4),'F'X)
      MS=IAND(ISHFT(OUTPUT(4),-4),'F'X)
      HMS=IAND(ISHFT(OUTPUT(4),-8),'F'X)
      TMS=ISHFT(OUTPUT(4),-12)
      WRITE(6,71)OUTPUT(1),HD,TD,UD,TH,UH,
71      1TM,UM,TS,US,TMS,HMS,MS,TEMS
      FORMAT(1X,'HEX STATUS= ',Z4,4X,'IRIG TIME= ',
      13I1,' ',2I1,' ',2I1,' ',2I1,' ',4I1)
      GO TO 2
      END

```

APPENDIX AC  
[AVA.TAPEDRIVE]COMMAND

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM LETS YOU ENTER A SIX DIGIT OCTAL COMMAND (16 BITS)
C   TO THE ON LINE DIGITIZER.
C
C .....
C
C   THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C       150001  I TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C       156400  I UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C       157000  I STOP
C       157476  I THE FILTERS ARE SET TO 120 ips I HOPE
C .....
C
C   SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   EXTERNAL IOSWRITEVBLK,IOSREADVBLK
C   INTEGER SYSS$ASSIGN,SYSS$QIOW,SYSS$QIO
C   INTEGER SYSS$GETMSG
C   INTEGER*2 IOSB(4),MSGLEN
C   INTEGER*2 INPUT,OUTPUT,INIT(4)
C   CHARACTER *80 MSGBUF
C   DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O/
C   ISTATUS=SYSS$ASSIGN('ODAB',ITCHAN,,)
C   IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
C   ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(%LOC(IOSWRITEVBLK)),

```

[AVA.TAPEDRIVE]COMMAND

```

1IOSB,,,
1INIT,XVAL(8),,,,
2  TYPE *,'INPUT CONTROL WORD'
  READ(5,45)INPUT
45  FORMAT(06)
1  ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
  IOSB,,,
  INPUT,XVAL(2),,,,
  IF(ISTATUS)      GO TO 500
  TYPE *,' ERROR IN QIOW CALL'
  ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO PARAMETER STATUS:',MSGBUF
  TYPE *,' IOSB(1)=' ,IOSB(1), ' IOSB(2)=' ,IOSB(2)
500 CONTINUE
11  FORMAT(1X,'INPUT=' ,06,2X,'IOSB=' ,06,2X,06,2X,06,2X,06)
C   ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C   IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C   TYPE *,'QIOW IO-STATUS RETURN:',MSGBUF
  WRITE(6,11)INPUT,IOSB(1),IOSB(2),IOSB(3),IOSB(4)
  GO TO 2
END

```

APPENDIX AD  
[AVA.TAPEDRIVE]REVIEW

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   THIS PROGRAM CONTINUOUSLY READS THE IRIG FROM THE SEARCH UNIT
C   BUT DISPLAYS ONLY THE DESIRED "SAVE" IRIG DESIGNATED BY THE
C   USER WHEN HE HITS THE RETURN KEY.
C
C   THE "SAVE" IRIGS ARE
C   WRITTEN TO DISK IN REVIEW.IRG AS THEY ARE COLLECTED.
C
C .....
C
C   THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C       150001 I TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C       156400 I UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C       157000 I STOP
C       157447 I THE FILTERS ARE SET TO 1 AND 10000 HZ
C .....
C
C   SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER SYSS$ASSIGN,SYSS$QIOW,SYSS$QIO
INTEGER SYSS$GETMSG
INTEGER*2 IOSB(4),MSGLEN
INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)

```

[AVA.TAPEDRIVE]REVIEW

```

INTEGER*2 US,TS,UM,TM,UH21
INTEGER*2 UH84,UH,TH,UD,TD,HD
INTEGER*2 TEMS,MS,HMS,TMS
CHARACTER *88 MSGBUF,GETIRIG*1
DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O,'157201'O/
DATA CONT/'156403'O,'156405'O,'156407'O,'156411'O,
1'156400'O/
OPEN(UNIT=8,NAME='REVIEW.IRG',TYPE='NEW')
ISTATUS=SYSS$ASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *, ' ERROR IN DR11-C CHANNEL ASSIGN'
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
1IOSB,,,
1INIT,XVAL(10),,,,
IF(ISTATUS.AND.IOSB(1)) GO TO 1
TYPE *, ' ERROR IN QIOW CALL'
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
TYPE *, 'I/O STATUS:',MSGBUF
1 ICONT=1
TYPE *, ' TO SAVE IRIG HIT RETURN WHEN DESIRED SCENE APPEARS.'
5 READ(5,5)GETIRIG
5 FORMAT(A)
2 CONTINUE
45 FORMAT(O6)
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
1IOSB,,,
1ICONT(ICONT),XVAL(2),,,,
IF(ISTATUS) GO TO 500
TYPE *, ' ERROR IN QIOW CALL'
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
TYPE *, ' IOSB(1)=' ,IOSB(1), ' IOSB(2)=' ,IOSB(2)
500 CONTINUE
IF(ICONT.EQ.5)GO TO 501
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
1IOSB,,,
1OUTPUT(ICONT),XVAL(2),,,,
IF(ISTATUS) GO TO 501
ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUG=' '
ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
TYPE *, 'I/O STATUS:',MSGBUF
GO TO 2
501 IF(ICONT.EQ.1)THEN
C C IST4=IAND(OUTPUT(1),'000010'O)
C C IST5=IAND(OUTPUT(1),'000020'O)
C C IST6=IAND(OUTPUT(1),'000040'O)
C C IST7=IAND(OUTPUT(1),'000100'O)
C C IST8=IAND(OUTPUT(1),'000200'O)

```

[AVA.TAPEDRIVE]REVIEW

```

C      C      IST9=IAND(OUTPUT(1),'000400'O)
C      C      IST10=IAND(OUTPUT(1),'001000'O)
C      C      IST11=IAND(OUTPUT(1),'002000'O)
C      C      IF(IST4.NE.0)TYPE*, 'START TIME FOUND'
C      C      IF(IST5.NE.0)TYPE*, 'STOP TIME FOUND'
C      C      IF(IST6.NE.0)TYPE*, 'PLAYBACK CYCLE BEGAN'
C      C      IF(IST7.NE.0)TYPE*, 'STOPPED'
C      C      IF(IST8.NE.0)TYPE*, 'PLAYBACK INTERVAL'
C      C      IF(IST9.NE.0)TYPE*, 'SEARCHING'
C      C      IF(IST10.NE.0)TYPE*, 'POWER OFF'
C      C      IF(IST11.NE.0)TYPE*, 'REMOTE SELECTED'
C      C      ENDIF
C      C      IF(ICON.T.5)THEN
C      C      ICON=ICON+1
C      C      GO TO 2
C      C      ENDIF
C      C      ICON=1
C
C      AFTER ALL STATUS AND IRIG HAVE BEEN READ IN LETS PRINT THEM OUT
C
C      IRIG WORD 1 DIGIT DECODING
C
C      US=IAND(OUTPUT(2),'F'X)
C      TS=IAND(ISHFT(OUTPUT(2),-4),'7'X)
C      UM=IAND(ISHFT(OUTPUT(2),-7),'F'X)
C      TM=IAND(ISHFT(OUTPUT(2),-11),'7'X)
C      UH21=ISHFT(OUTPUT(2),-14)
C
C      IRIG WORD 2 DIGIT DECODING
C
C      UH84=ISHFT(IAND(OUTPUT(3),'3'X),2)
C      UH=IOR(UH84,UH21)
C      TH=IAND(ISHFT(OUTPUT(3),-2),'3'X)
C      UD=IAND(ISHFT(OUTPUT(3),-4),'F'X)
C      TD=IAND(ISHFT(OUTPUT(3),-8),'F'X)
C      HD=ISHFT(OUTPUT(3),-12)
C
C      IRIG WORD 3 DIGIT DECODING
C
C      TEMS=IAND(OUTPUT(4),'F'X)
C      MS=IAND(ISHFT(OUTPUT(4),-4),'F'X)
C      HMS=IAND(ISHFT(OUTPUT(4),-8),'F'X)
C      TMS=ISHFT(OUTPUT(4),-12)
C      WRITE(6,71)OUTPUT(1),HD,TD,UD,TH,UH,
C      1TM,UM,TS,US,TMS,HMS,MS,TEMS
C      WRITE(8,71)OUTPUT(1),HD,TD,UD,TH,UH,
C      1TM,UM,TS,US,TMS,HMS,MS,TEMS
71      FORMAT(1X,'HEX STATUS= ',Z4.4X,'IRIG TIME= ',
C      13I1,'.',2I1,'.',2I1,'.',2I1,'.',4I1)
C      READ(5,5,END=7777)GETIRIG
C      GO TO 2
7777      CLOSE(UNIT=8)
C      STOP ' REVIEW. IRG GENERATED'
C      END

```

APPENDIX AE

[AVA.TAPEDRIVE]RTODISK

```

THIS PROGRAM USES THE REVIEW.IRG FILE TO FIND SEARCH TIMES.

THE FOLLOWING INSTRUCTIONS ARE SENT TO THE IRIG SEARCH UNIT
ON THE ON LINE DIGITIZER AND THEN THE REVIEW.IRG FILE IS READ FOR THE
FIRST SEARCH TIME.

150001 I TRANSLATE IRIG A WITH ZERO FRAME BYPASS
156400 I UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
157000 I STOP
157447 I THE FILTERS ARE SET TO 120 Ips I HOPE, NO CARRIER FILTER

ENTER IRIG INPUT:

THE FOLLOWING IS THE SEQUENCE OF CONTROL WORDS THAT WOULD BE SENT
IF THE OPERATOR ENTERS A START IRIG TIME OF 000:00:01:00.0000

150400 I SEARCH START TIME DAYS "00X" WHERE X IS NOT SET IN THIS WORD
151000 I SEARCH START TIME DAYS "SS0", HOURS "00" WHERE SS WAS SET ABOVE
151401 I SEARCH START TIME MINUTES "01" MINUTES
152000 I SEARCH START TIME SECONDS "00" SECONDS
152400 I SEARCH START TIME MILLISECONDS .00XX
153000 I SEARCH START TIME MILLISECONDS .SS00 WHERE SS WAS SET ABOVE

THE PROGRAM AFTER THE IRIG IS ENTERED TRANSFERS TO THE IRIG SEARCH
UNIT TRANSFERS THE FOLLOWING CONTROL WORD AND THE SEARCH PROCESS IS
INITIATED.

157227 I SEARCH TO START TIME 000:00:01:00.0000 OR USER ENTERED

```

[AVA.TAPEDRIVE]RTODISK

```

DOUBLE PRECISION QUAD
CHARACTER*16 TIME,FILENAME*60
INTEGER*2 TM,UM,TS,US,ITMS,HMS,TMS
INTEGER*2 HD,ITD,TD,UD,UH,TH
INTEGER*2 IDAY,IHR,IMIN,ISEC,IMSEC
INTEGER*2 TEMS,IUMS,UMS
CHARACTER *80 MSGBUF
EQUIVALENCE(SW1,INPUT(5)),(SW2,INPUT(6)),(SW3,INPUT(7))
EQUIVALENCE(SW4,INPUT(8)),(SW5,INPUT(9)),(SW6,INPUT(10))
COMMON/AVACHAN/IAVACHAN
DATA INPUT/'150001'O,'156400'O,'157000'O,'157447'O,'150400'O,
1'/'151000'O,'151401'O,'152000'O,'152400'O,'153000'O,'157227'O/
DATA READCNTRL/'157442'O,'157201'O,'157061'O/
ISTATUS=SYSS$ASSIGN('AVA0',IAVACHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN ON LINE DIGITIZER CHANNEL ASSIGN'
TYPE *,'ENTER BEGINNING FILE NAME TO BE USED FOR DISK FILES',
1'.' (1.e. X00320000)'
READ(5,45)FILENAME
45 FORMAT(A)
C2 TYPE *,'ENTER SEARCH START IRIG TIME IN THE FOLLOWING FORMAT:'
C TYPE *,'DAY:HR:MN:SC.MSEC'
C TYPE *,'000:00:01:00.0000 FOR EXAMPLE'
C READ(5,45)IDAY,IHR,IMIN,ISEC,IMSEC
C45 FORMAT(I3,1X,I2,1X,I2,1X,I2,1X,I4)
OPEN(UNIT=8,NAME='REVIEW.IRG',STATUS='OLD')
2 READ(8,71,END=7777)OUTPUT(1),HD,TD,UD,TH,UH,
1TM,UM,TS,US,TMS,HMS,MS,TEMS
71 FORMAT(I3X,Z4,15X,
13I1,1X,2I1,1X,2I1,1X,2I1,1X,4I1)
C HD=IDAY/100
C ITD=IDAY-(HD*100)
C TD=ITD/10
C UD=ITD-(TD*10)
C SW1='006420'O
C SW1=ISHFT(IOR(SW1,TD),4)
C SW1=IOR(SW1,UD)
C SW2='001510'O
C SW2=ISHFT(IOR(IAND(HD,3),SW2),2)
C TH=IHR/10
C SW2=ISHFT(IOR(IAND(TH,3),SW2),4)
C UH=IHR-(TH*10)
C SW2=IOR(SW2,UH)
C SW3='006460'O
C TM=IMIN/10
C UM=IMIN-(TM*10)
C SW3=ISHFT(IOR(SW3,TM),4)
C SW3=IOR(SW3,UM)
C TS=ISEC/10
C US=ISEC-(TS*10)
C SW4='006500'O
C SW4=ISHFT(IOR(SW4,TS),4)
C SW4=IOR(SW4,US)
C SW5='006520'O

```

[AVA.TAPE DRIVE]RTODISK

```

SW6='006540'O
C HMS=IMSEC/1000
C ITMS=IMSEC-(HMS*1000)
C TMS=ITMS/100
C IUMS=ITMS-(TMS*100)
C UMS=IUMS/10
C TEMS=IUMS-(UMS*10)
SW5=ISHFT(IOR(SW5,HMS),4)
SW5=IOR(SW5,TMS)
SW6=ISHFT(IOR(SW6,UMS),4)
SW6=IOR(SW6,TEMS)
C WRITE(6,55) SW1,SW2,SW3,SW4,SW5,SW6
55 FORMAT(1X,'SWX=',6(1X,Z4))
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   INPUT,XVAL(22),...)

WRITE(6,171)OUTPUT(1),HD,TD,UD,TH,UH,
ITM,UM,TS,US,TMS,HMS,MS,TEMS
171 FORMAT(1X,'SEARCHING FOR HEX STATUS= ',Z4,X,'IRIG TIME= ',
13I1,'.',2I1,'.',2I1,'.',2I1,'.',4I1)
   IF(.NOT.ISTATUS.OR..NOT. IOSB(1))GO TO 34
C CHECK TAPE DRIVE STATUS

61 ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   I'156403'O,XVAL(2),...) ITELL DATUM YOU WANT STATUS
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...)
   IDSTATUS,XVAL(2),...)
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   I'156400'O,XVAL(2),...) ICLEAR DATUM
   IF(IAND(DSTATUS,'100'O).EQ.0)GO TO 61 IIS IT STOPPED??
C MOVE FOWARD TO CORRECT IRIG THEN PLAY TAPE AT 32/1 AND TRANSFER IMAGES.
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   IREADCNTRL,XVAL(4),...) ISET UP PLAYBACK FILTER AND MOVE BACK FOWARD 120 IPS
   TIME='0000 00:00:09.60'
   ISTATUS=SYSSBINTIM(XDESCR(TIME),QUAD)
   IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
   ISTATUS=SYSSSETIMR(XVAL(6),QUAD,...)
   IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
   ISTATUS=SYSSWAITFR(XVAL(6))
   IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
C NOW PLAYBACK AT 32/1
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   IREADCNTRL(3),XVAL(2),...) ISET UP PLAYBACK FILTER AND MOVE BACK FOWARD 120 IPS
62 ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,...)
   I'156403'O,XVAL(2),...) ITELL DATUM YOU WANT STATUS
   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
   IOSB,...)
   IDSTATUS,XVAL(2),...)

```

[AVA.TAPEDRIVE]RTODISK

```

C      IF(IAND(DSTATUS,2).EQ.0)GO TO 62      IIS TAPE SYNC ON?
C      GIVE AVA TIME TO LOAD ALL FIELDS
C
      TIME='0000 00:00:03.00'
      ISTATUS=SYSSBINTIM(XDESCR(TIME),QUAD)
      IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
      ISTATUS=SYSSSETIMR(XVAL(6),QUAD,,)
      IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
      ISTATUS=SYSSWAITFR(XVAL(6))
      IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
C
C      IS FIELD 1 BEING LOADED? (0 , 1 , 2 , 3)
C
23     CALL FIELD(IFIELD,IAVACSR)
      IF(IFIELD.NE.1)GO TO 23
C
C      WRITE THE IMAGE TO DISK
C
      CALL AVATODSK2(FILENAME)
      GO TO 2
34     TYPE *,' ERROR IN QIOW CALL'
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO SGETMSG'
      TYPE *,'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      TYPE *,'I/O STATUS:',MSGBUF
500    CONTINUE
11     FORMAT(1X,'INPUT=',06,2X,'IOSB=',06,2X,06,2X,06,2X,06)
      GO TO 2
7777   ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
      1IOSB,,)
      1'157220'O,XVAL(2),,,) ISTOP HBR-3000 TAPE DRIVE
      STOP 'ALL IRIGS HAVE BEEN FOUND'
      END
      SUBROUTINE FIELD(IFIELD,AVACSR)
      INTEGER AVACSR
      EXTERNAL IOSWRITEVBLK,IOSREADVBLK
      INTEGER SYSSASSIGN,SYSSQIOW,SYSSQIO
      INTEGER SYSSGETMSG
      INTEGER*2 IOSB(4),MSGLEN,NPUT,X,V
      INTEGER*2 INPUT,OUTPUT,INIT(4)
      CHARACTER *80 MSGBUF
      COMMON/AVACHAN/ITCHAN
      IAVACSR='4001'O ISET MEMORY WINDOW ENABLE AND INITIALIZE AVA
      ISAVE=AVACSR
      ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSREADVBLK)),
      1IOSB,,)
      1OUTPUT,XVAL(2),XVAL(X),XVAL(Y),XVAL(IAVACSR),XVAL(IAVAACR))
      IF(ISTATUS) GO TO 501
      TYPE *,' ERROR IN QIOW CALL'
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO SGETMSG'

```

CAVA.TAPEDRIVEJRTODISK

```
501      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
        MSGBUF= ' '
        ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF (.NOT. ISTATUS) TYPE *, 'ERROR IN CALL TO SGETMSG'
        TYPE *, 'I/O STATUS:', MSGBUF
        AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

## [AVA.TAPEDRIVE]STO:START

[illegible]

THIS PROGRAM LETS THE OPERATOR ENTER THE START SEARCH TIME FROM THE KEYBOARD AND THEN TELLS THE SEARCH UNIT TO GO AND SEARCH FOR THE IRIG TIME ENTERED.

THE FOLLOWING INSTRUCTIONS ARE SENT TO THE IRIG SEARCH UNIT  
ON THE ON LINE DIGITIZER AND THEN THE OPERATOR IS ASKED FOR START IRIG.

```

150001 I TRANSLATE IRIG A WITH ZERO FRAME BYPASS
156400 I UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
157000 I STOP
157447 I THE FILTERS ARE SET TO 120 ps I HOPE, NO CARRIER FILTER

```

ENTER IRIG INPUT:

THE FOLLOWING IS THE SEQUENCE OF CONTROL WORDS THAT WOULD BE SENT  
IF THE OPERATOR ENTERS A START IRIG TIME OF 000:00:01:00.0000

```

150400 I SEARCH START TIME DAYS "00X" WHERE X IS NOT SET IN THIS WORD
151000 I SEARCH START TIME DAYS "SS0" HOURS "00" WHERE SS WAS SET ABOVE
151401 I SEARCH START TIME MINUTES "01" MINUTES
152000 I SEARCH START TIME SECONDS "00" SECONDS
152400 I SEARCH START TIME MILLISECONDS .00XX
153000 I SEARCH START TIME MILLISECONDS .SS00 WHERE SS WAS SET ABOVE

```

THE PROGRAM AFTER THE IRIG IS ENTERED TRANSFERS TO THE IRIG SEARCH UNIT TRANSFERS THE FOLLOWING CONTROL WORD AND THE SEARCH PROCESS IS INITIATED.

157207 I SEARCH TO START TIME 000:00:01:00.0000 OR USER ENTERED

[illegible]

```
EXTERNAL IOSWRITEBLK,IOSREADBLK
INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
INTEGER SYS$GETMSG
INTEGER*2 IOS(4),MSGLEN
INTEGER*2 INPUT(11)
INTEGER*2 SW1,SW2,SW3,SW4,SW5,SW6,SW7,SW8,SW9,SW10,SW11,SW12
```

[AVA.TAPEDRIVE]STOSTART

```

INTEGER*2 TM,UM,TS,US,ITMS,HMS,TMS
INTEGER*2 HD,ITD,TD,UD,UH,TH
INTEGER*2 IDAY,IHR,IMIN,ISEC,IMSEC
INTEGER*2 TEMS,IUMS,UMS
CHARACTER *80 MSGBUF
EQUIVALENCE(SW1,INPUT(5)),(SW2,INPUT(6)),(SW3,INPUT(7))
EQUIVALENCE(SW4,INPUT(8)),(SW5,INPUT(9)),(SW6,INPUT(10))
DATA INPUT/'150001'O,'156400'O,'157000'O,'157447'O,'150400'O,
1'151000'O,'151401'O,'152000'O,'152400'O,'153000'O,'157207'O/
ISTATUS=SYSS$ASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN ON LINE DIGITIZER CHANNEL ASSIGN'
2 TYPE *,'ENTER SEARCH START IRIG TIME IN THE FOLLOWING FORMAT:'
TYPE *,'DAY:HR:MN:SC.MSEC'
TYPE *,'000:00:01:00.0000 FOR EXAMPLE'
45 READ(5,45)IDAY,IHR,IMIN,ISEC,IMSEC
FORMAT(13,1X,12,1X,12,1X,12,1X,14)
HD=IDAY/100
ITD=IDAY-(HD*100)
TD=ITD/10
UD=ITD-(TD*10)
SW1='006420'O
SW1=ISHFT(IOR(SW1,TD),4)
SW1=IOR(SW1,UD)
SW2='001510'O
SW2=ISHFT(IOR(IAND(HD,3),SW2),2)
TH=IHR/10
SW2=ISHFT(IOR(IAND(TH,3),SW2),4)
UH=IHR-(TH*10)
SW2=IOR(SW2,UH)
SW3='006460'O
TM=IMIN/10
UM=IMIN-(TM*10)
SW3=ISHFT(IOR(SW3,TM),4)
SW3=IOR(SW3,UM)
TS=ISEC/10
US=ISEC-(TS*10)
SW4='006500'O
SW4=ISHFT(IOR(SW4,TS),4)
SW4=IOR(SW4,US)
SW5='006520'O
SW6='006540'O
HMS=IMSEC/1000
ITMS=IMSEC-(HMS*1000)
TMS=ITMS/100
IUMS=ITMS-(TMS*100)
UMS=IUMS/10
TEMS=IUMS-(UMS*10)
SW5=ISHFT(IOR(SW5,HMS),4)
SW5=IOR(SW5,TMS)
SW6=ISHFT(IOR(SW6,UMS),4)
SW6=IOR(SW6,TEMS)
C WRITE(6,55) SW1,SW2,SW3,SW4,SW5,SW6
55 FORMAT(1X,'SWX=',6(1X,Z4))
ISTATUS=SYSS$QIOW(XVAL(1),XVAL(ITCHAN),XVAL(%LOC(IOWRITEVBLK)),
1IOSB,...

```

[AVA.TAPEDRIVE]STO START

```
      1INPUT,XVAL(22),,,)
      IF(.NOT.ISTATUS.OR..NOT. IOSB(1))GO TO 34
      GO TO 2
34     TYPE *, ' ERROR IN QIOW CALL '
      ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
      MSGBUF=' '
      ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
      TYPE *, 'I/O STATUS:',MSGBUF
588    CONTINUE
11     FORMAT(1X,'INPUT=',06,2X,'IOSB=',06,2X,06,2X,06,2X,06)
      GO TO 2
      END
```

## APPENDIX AG

[AVA.TAPEDRIVE]STATUSR

THIS PROGRAM CONTINUOUSLY READS THE STATUS FROM THE SEARCH UNIT  
AND DISPLAYS IT ON THE TERMINAL.

THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS

```

150001 | TRANSLATE IRIG A WITH ZERO FRAME BYPASS
156400 | UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
157000 | STOP
157447 | THE FILTERS ARE SET TO 1 AND 10000 HZ

```

SOME TYPICAL COMMANDS ARE AS FOLLOWS:

150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS

156400 = UPDATE TIME. RESET RECORD ENABLE. RESET INTERRUPT ENABLE

```

157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
157000 = STOP
157222 = DRIVE REVERSE AT 240 ips
157222 = DRIVE REVERSE AT 120 ips
157206 = SINGLE CYCLE SEARCH MODE AT 120 ips

```

```

EXTERNAL IOSWRITEBLK,IOSREADVBLK
INTEGER SYSS$ASSIGN,SYSS$QIOW,SYSS$QIO
INTEGER SYSS$GETMSG
INTEGER*2 IOSB(4),MSGLEN
INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)
INTEGER*2 US,TS,UM,TM,UH21
INTEGER*2 UH84,UH,TH,UD,TD,HD
INTEGER*2 TEMS,MS,HMS,TMS
CHARACTER *80 MSGBUF
DATA INIT/'150000'0,'156400'0,'157000'0,'157447'0,'157201'0/

```

[AVA.TAPEDRIVE]STATUSR

```

DATA CONT/'156403'O,'156405'O,'156407'O,'156411'O,
1'156400'O/
ISTATUS=SYSSASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$WRITEVBLK)),
1IOSB,,,
1INIT,XVAL(8),,,,
C IF(ISTATUS.AND.IOSB(1)) GO TO 1
C TYPE *,' ERROR IN QIOW CALL'
C ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
C IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C MSGBUF=' '
C ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C TYPE *,'I/O STATUS:',MSGBUF
1 ICONT=1
2 CONTINUE
45 FORMAT(06)
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$WRITEVBLK)),
1IOSB,,,
ICONT(ICONT),XVAL(2),,,,
IF(ISTATUS) GO TO 500
TYPE *,' ERROR IN QIOW CALL'
ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
TYPE *,' IOSB(1)=' ,IOSB(1),' IOSB(2)=' ,IOSB(2)
500 CONTINUE
ISTATUS=SYSSQIOW(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(10$READVBLK)),
1IOSB,,,
1OUTPUT(ICONT),XVAL(2),,,,
IF(ISTATUS) GO TO 501
ISTATUS=SYSSGETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
MSGBUG=' '
ISTATUS=SYSSGETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
TYPE *,'I/O STATUS:',MSGBUF
GO TO 2
501 IST4=IAND(OUTPUT(1),'000010'O)
IST5=IAND(OUTPUT(1),'000020'O)
IST6=IAND(OUTPUT(1),'000040'O)
IST7=IAND(OUTPUT(1),'000100'O)
IST8=IAND(OUTPUT(1),'000200'O)
IST9=IAND(OUTPUT(1),'000400'O)
IST10=IAND(OUTPUT(1),'001000'O)
IST11=IAND(OUTPUT(1),'002000'O)
IST1= IAND(OUTPUT(1),'000002'O)
TYPE *,'.....STATUS IS AS FOLLOWS.....',
1'.....'
IF(IST1.NE.0)TYPE *,'SYNC'
IF(IST4.NE.0)TYPE *,'START TIME FOUND'
IF(IST5.NE.0)TYPE *,'STOP TIME FOUND'
IF(IST6.NE.0)TYPE *,'PLAYBACK CYCLE BEGAN'
IF(IST7.NE.0)TYPE *,'STOPPED'

```

CAVA.TAPEDRIVE1STATUSR

```
IF(IST8.NE.0)TYPE*, 'PLAYBACK INTERVAL'
IF(IST9.NE.0)TYPE*, 'SEARCHING'
IF(IST10.NE.0)TYPE*, 'POWER OFF'
IF(IST11.NE.0)TYPE*, 'REMOTE SELECTED'
C  WRITE(6,7)OUTPUT(1)
7  FORMAT(1X,06)
   GO TO 2
   END
```

APPENDIX AH  
[AVA.TAPEDRIVE]IOTEST

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      THIS PROGRAM IS USED TO TEST THE ON LINE DIGITIZER DR11-C INTERFACE
C      IN CONJUNCTION WITH THE ODDRIVER. THE MAINTENANCE CABLE MUST BE HOOKED
C      UP!
C
C      THE ODDRIVER TRANSFERS THE WORD IN INPUT TO THE OUTBUF REGISTER ON
C      THE DR11-C. THE DRIVER THEN SETS CSR0 AND THE IEA BITS. THIS WILL
C      CAUSE AN INTERRUPT AFTER IPL IS LOWERED BELOW DEVICE IPL. AFTER
C      THE INTERRUPT THE INPUTBUF IS THE COPIED INTO IOSB(3) AND IS USED IN
C      THIS PROGRAM TO COMPARE TO WHAT WENT INTO THE OUTBUF.
C
C      THIS PROGRAM TESTS ALL DATA BITS ON THE DR11-C AND THE "A" INTERRUPT
C      HARDWARE.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER SYSS$ASSIGN,SYSS$QIOW,SYSS$QIO
INTEGER SYSS$GETMSG
INTEGER*2 IOSB(4),MSGLEN
INTEGER*2 INPUT,OUTPUT
CHARACTER *80 MSGBUF
ISTATUS=SYSS$ASSIGN('ODA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
INPUT='100000'0
2  TYPE *,'DR11-C BIT TEST STARTING.....'
1  ISTATUS=SYSS$QIO(XVAL(1),XVAL(ITCHAN),XVAL(XLOC(IOSWRITEVBLK)),
   IOSB,,,
   INPUT,XVAL(2),,,,
   IF(ISTATUS) GO TO 500
   TYPE *,' ERROR IN QIOW CALL'
   ISTATUS=SYSS$GETMSG (XVAL(ISTATUS), MSGLEN, MSGBUF,,)
   IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
   TYPE *,'QIO PARAMETER STATUS:',MSGBUF
   TYPE *,' IOSB(1)=' ,IOSB(1), ' IOSB(2)=' ,IOSB(2)
500 CONTINUE
11  FORMAT(1X,06,2X,06,'CSR=' ,06,2X,06)
C   ISTATUS=SYSS$GETMSG (XVAL(IOSB(1)), MSGLEN, MSGBUF,,)
C   IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C   TYPE *,'QIOW IO-STATUS RETURN:',MSGBUF
C   IF(INPUT.NE.IOSB(3))THEN

```

[AVA.TAPEDRIVE]IOTEST

```
12      TYPE *, ' INBUF NOT EQUAL TO OUTBUF'
        WRITE(6,11)IOSB(1),IOSB(2),IOSB(3),IOSB(4)
        WRITE(6,12) INPUT,IOSB(3)
        FORMAT(1X,'OUTPUT WAS=',06,5X,'INPUT WAS=',06)
        ENDIF
        INPUT=INPUT+1
        IF (INPUT.GE.32767) THEN
            INPUT='100000'0
            TYPE *, 'DR11-C BIT TEST COMPLETED'
            GO TO 2
        ENDIF
        GO TO 1
    END
```

APPENDIX A1  
AVA FRAME BUFFER I/O DRIVER

```
.TITLE AVDRIVER - VAX/VMS AVA FRAME BUFFER INTERFACE DRIVER
.IDENT 'V83-001'

; **
; FACILITY:
;
; VAX/VMS ON LINE DIGITIZER AVA FRAME BUFFER
; ABSTRACT:
;
; This module contains the driver:
;
; Tables
; Controller and unit initialization routines
; The FDT routine
; The start I/O routine
; The interrupt service routine
; The cancel I/O routine
; The device register dump routine
;
; AUTHOR:
;
; S. Richard F. Sims January 27, 1983
; REVISION HISTORY:
;
; --
; .SBTTL External and local symbol definitions
;
; External symbols
;
; SCANDEF ; Cancel reason codes
; SCRBDDEF ; Channel request block
; SDCDEF ; Device classes and types
; SDOBDEF ; Device data block
; SDEVDEF ; Device characteristics
; SIDBDEF ; Interrupt data block
; SIODEF ; I/O function codes
; SIPLDEF ; Hardware IPL definitions
; SIRPDEF ; I/O request packet
```

# AVA FRAME BUFFER I/O DRIVER

```

; System status codes
; Unit control block
; Interrupt vector block
; JOB INFO BLOCK OFFSET DEFS
; PROCESS CONTROL BLOCK OFFSET DEFS
SSSDEF
UCBDEF
SVECDEF
SJIBDEF
SPCBDEF

; Local symbols
;
; Argument list (AP) offsets for device-dependent QIO parameters
;
P1      = 0      ; First QIO parameter
P2      = 4      ; Second QIO parameter
P3      = 8      ; Third QIO parameter
P4      = 12     ; Fourth QIO parameter
P5      = 16     ; Fifth QIO parameter
P6      = 20     ; Sixth QIO parameter
;
; Other constants
;
AVDEFBUFSIZ      = 1      ; Default buffer size
AVTIMEOUTSEC     = 5      ; 1/2 second device timeout
AVNUMREGS        = 3      ; Device has 3 registers
BUFOVRHD        = 12     ; SYSTEM BUFFER OVERHEAD FOR BUFFERED I/O
;
; Definitions that follow the standard UCB fields
;
SDEFINI UCB      ; Start of UCB definitions
.=UCBSKLENGTH   ; Position at end of UCB
UCBSWAVCSR      ; UCB Device CSR STORAGE
SDEF            .BLKW 1
UCBSWAVBYTCNT   ; Device's BYTE count register
SDEF            .BLKW 1
UCBSWAVOUTBUF   ; DEVICE OUTBUF REGISTER
SDEF            .BLKW 1
UCBSWAVXADDR    ; STARTING X ADDRESS (P3)
SDEF            .BLKW 1
UCBSWAVYADDR    ; STARTING Y ADDRESS (P4)
SDEF            .BLKW 1
UCBSWAVACR      ; INITIALIZE ACCES CONTROL REGISTER BITS
SDEF            .BLKW 1
UCBSKAVUCBLEN   ; Length of extended UCB
SDEF            .BLKW 1
;
; Bit positions for device-dependent status field in UCB
;
SVIELD UCBCSR,0,<-      ; Device status
<BITZERO,,M>,-        ; First bit
<BITONE,,M>,-         ; Second bit
>
SDEFEND UCB          ; End of UCB definitions
;
; Device register offsets from CSR address
;
SDEFINI AV           ; Start of status definitions

```

# AVA FRAME BUFFER I/O DRIVER

```

SDEF    AVCSR                                ; Control/status
        .BLKW    1
;
; Bit positions for device control/status register
;
        YIELD    AVCSR,0,<-                ; Control/status register
                <AVEN,,M>,-                ; WHEN UNASSERTED INITIALIZES AVA HARDWARE
                <INTEN0,,M>,-              ; ENABLES AVA INTERRUPT 0
                <INTEN1,,M>,-              ; ENABLES AVA INTERRUPT 1
                <INTEN2,,M>,-              ; ENABLES AVA INTERRUPT 2
                <INTEN3,,M>,-              ; ENABLES AVA INTERRUPT 3
                <DMAEN,,M>,-               ; ENABLES AVA UNIBUS MASTERSHIP REQUEST
                <OTAG0,,M>,-               ; GENERAL PURPOSE SENSE LINE
                <ITAG,,M>,-                ; GENERAL PURPOSE SENSE LINE
                <WDR0,,M>,-                ; MEMORY WINDOW ID BIT 0
                <WDR1,,M>,-                ; MEMORY WINDOW ID BIT 1
                <WDR2,,M>,-                ; MEMORY WINDOW ID BIT 2
                <WDEN,,M>,-                ; ENABLES AVA RESPONSE TO MEMORY WINDOW ACCESS
                <OTAG1,,M>,-               ; GENERAL PURPOSE SOFTWARE TAG
                <,3>,-                     ; RESERVED FOR FUTURE USE
        >
SDEFEND AV                                ; End of device register
; definitions.
        FSMCCSR=    AVCSR-0000176        ; FRAME STORE MEMORY CONTROLLER CSR OFFSET

        CPUIOVFLAG=    AVCSR-0000072
        CPUIDATARD0=    AVCSR-0000070
        CPUIDATARD1=    AVCSR-0000066
        CPUIFSTEST=    AVCSR-0000064
        CPUISEQSUB=    AVCSR-0000062
        CPUIMAIN=    AVCSR-0000060        ; MAINTENANCE REGISTER
        CPUIACR=    AVCSR-0000056        ; ACCESS CONTROL REGISTER
        CPUIYSTR=    AVCSR-0000052
        CPUIXSTR=    AVCSR-0000050
        CPUIFYENCE=    AVCSR-0000046
        CPUIXFENCE=    AVCSR-0000044
        CPUIYADDR=    AVCSR-0000042
        CPUIXADDR=    AVCSR-0000040
        CPUICOMP0=    AVCSR-0000036
        CPUICOMP1=    AVCSR-0000034
        CPUICOMP2=    AVCSR-0000032
        CPUICOMP3=    AVCSR-0000030
        CPUICOMP4=    AVCSR-0000026
        CPUICOMP5=    AVCSR-0000024
        CPUICOMP6=    AVCSR-0000022
        CPUICOMP7=    AVCSR-0000020

.SBTTL    Standard tables
;
; Driver prologue table
;
        DPTAB    -                        ; DPT-creation macro
                END=AVEND,-                ; End of driver label
                ADAPTER=UBA,-              ; Adapter type
                UCBSIZE=<UCB$KAVUCBLEN>,- ; Length of UCB
                NAME=AVDRIVER              ; Driver name

```

# AVA FRAME BUFFER I/O DRIVER

```

DPTSTORE INIT ; Start of load
; initialization table
DPTSTORE UCB,UCBSBFIPL,B,8 ; Device fork IPL
DPTSTORE UCB,UCBSBDIPL,B,22 ; Device interrupt IPL=22=8R6
DPTSTORE UCB,UCBSLDEVCHAR,L,- ; Device characteristics
DEVSMIDVI- ; input device
DEVSMAVLI-
DEVSMODV> ; output device
DPTSTORE UCB,UCBSBDEVCLASS,B,DCSSCOM ; Device class?
DPTSTORE UCB,UCBSBDEVTYPE,B,DT$DR11C ; DEVICE TYPE (NOT REALLY)
DPTSTORE UCB,UCBSWDEVBUFSIZ,W,- ; Default buffer size
AVDEFBUFSIZ
DPTSTORE REINIT ; Start of reload
; initialization table
DPTSTORE DDB,DBBSLDDT,D,AVSDDT ; Address of DDT
DPTSTORE CRB,CRBSLINTD+4,D,- ; Address of interrupt
AVINTERRUPT ; service routine
DPTSTORE CRB,- ; Address of controller
CRBSLINTD+VECSLINITIAL,- ; initialization routine
D,AVCONTROLINIT
DPTSTORE CRB,- ; Address of device
CRBSLINTD+VECSLUNITINIT,- ; unit initialization
D,AVUNITINIT ; routine
DPTSTORE END ; End of initialization
; tables

; Driver dispatch table
;
DDTAB - ; DDT-creation macro
DEVNAM=AV,- ; Name of device
START=AVSTART,- ; Start I/O routine
FUNCTB=AVFUNCTABLE,- ; FDT address
CANCEL=AVCANCEL,- ; Cancel I/O routine
REGDMP=AVREGDUMP ; Register dump routine

; Function decision table
;
AVFUNCTABLE: ; FDT for driver
FUNCTAB ,- ; Valid I/O functions
<READVBLK,- ; Read virtual
READLBLK,- ; Read logical
READPBLK,- ; Read physical
WRITEVBLK,- ; Write virtual
WRITELBLK,- ; Write logical
WRITEPBLK> ; Write physical
FUNCTAB ,- ; Buffered functions
<READVBLK,- ; Read virtual
READLBLK,- ; Read logical
READPBLK,- ; Read physical
WRITEVBLK,- ; Write virtual
WRITELBLK,- ; Write logical
WRITEPBLK> ; Write physical
FUNCTAB AVAFDT,-
<READVBLK,- ; Read virtual
READLBLK,- ; Read logical

```

# AVA FRAME BUFFER I/O DRIVER

```

        READPBLK,-                ; Read physical
        WRITEVBLK,-              ; Write virtual
        WRITELBLK,-             ; Write logical
        WRITEPBLK>              ; Write physical
FUNCTAB AVWRITEAVACFDT,-
        <WRITEVBLK,-            ; Write virtual
        WRITELBLK,-            ; Write logical
        WRITEPBLK>             ; Write physical
FUNCTAB AVREADAVACFDT,-
        <READVBLK,-            ; Read virtual
        READLBLK,-             ; Read logical
        READPBLK>              ; Read physical
        .LONG -1                ; SET ALL BITS FOR THE
        .LONG -1                ; FDT CATCH ALL ERROR ROUTINE
        .ADDRESS OOPS
.SBTTL AVCONTROLINIT, Controller initialization routine
;--
; AVCONTROLINIT, Readies controller for I/O operations
;
; Functional description:
;
;     The operating system calls this routine in 3 places:
;
;         at system startup
;         during driver loading and reloading
;         during recovery from a power failure
;
; Inputs:
;
;     R4      - address of the CSR (controller status register)
;     R5      - address of the IDB (interrupt data block)
;     R6      - address of the DDB (device data block)
;     R8      - address of the CRB (channel request block)
;
; Outputs:
;
;     The routine must preserve all registers except R0-R3.
;
;--
AVCONTROLINIT:                ; Initialize controller
        RSB                    ; Return
.SBTTL AVUNITINIT, Unit initialization routine
;--
; AVUNITINIT, Readies unit for I/O operations
;
; Functional description:
;
;     The operating system calls this routine after calling the
;     controller initialization routine:
;
;         at system startup
;         during driver loading
;         during recovery from a power failure
;
; Inputs:

```

# AVA FRAME BUFFER I/O DRIVER

```

;
;      R4      - address of the CSR (controller status register)
;      R5      - address of the UCB (unit control block)
;
; Outputs:
;
;      The routine must preserve all registers except R0-R3.
;
;--
AVUNITINIT:                                ; Initialize unit
        BISW   UCBS$MONLINE, -             ; Set unit online
        UCBS$WSTS(R5)
        CLRW   AVCSR(R4)                   ; INITIALIZE AVA FRAME BUFFER
        MOVW   1,AVCSR(R4)                 ; SET *INIT BIT IN CSR
        RSB                                ; Return
        .SBTTL AVFDTROUTINE, ON LINE DIGITIZER AVA FDT routine
;
;+
; AVFDTROUTINE, ON LINE DIGITIZER AVA FDT routine
;
; Functional description:
;
;      SET UP FOR BUFFERED IO ON THE AVA INTERFACE
;
; Inputs:
;
;      R0-R2    - scratch registers
;      R3      - address of the IRP (I/O request packet)
;      R4      - address of the PCB (process control block)
;      R5      - address of the UCB (unit control block)
;      R6      - address of the CCB (channel control block)
;      R7      - bit number of the I/O function code
;      R8      - address of the FDT table entry for this routine
;      R9-R11   - scratch registers
;      AP      - address of the 1st function dependent QIO parameter
;
; Outputs:
;
;      The routine must preserve all registers except R0-R2, and
;      R9-R11.
;
;--
;
;      CATCH ALL FDT ERROR ROUTINE
;
OOPS:
        MOVL   SSS$ILLIOFUNC,R0            ; ILLEGAL I/O FUNCTION SPECIFIED
        JSB    GEXE$ABORTIO               ; SO LET'S ABORT
;
AVAFDT:
        MOVW   P3(AP),UCBS$WAVXADDR(R5)    ; STARTING X ADDRESS
        MOVW   P4(AP),UCBS$WAVYADDR(R5)    ; STARTING Y ADDRESS
        MOVW   P5(AP),UCBS$WAVCSR(R5)      ; INITIALIZE ACCESS CONTROL REGISTER BITS
        MOVW   P6(AP),UCBS$WAVACR(R5)     ; SET UP ACR
        RSB
;
AVWRITEAVCFDT:
        MOVQ   P1(AP),R0                  ; WRITE FDT routine
;
;      MOVE BUFFER ADDRESS IN R0
;

```

```

TSTL      R1                ; AND BUFFER SIZE      IN R1
BGTR      1$                ; IF BUFFER SIZE <=0 WE HAVE PROBLEMS
MOVL      SSS$IVBUFLN,R0    ; OTHERWISE LETS GET ON WITH IT
JMP        GEXES$FINISHIO    ; BAD BUFFER SIZE
1$: JSB     GEXES$WRITECHK    ; ABORTS AND DOESN'T COME BACK IF IT
                                ; CAN'T WRITE TO BUFFER
;          MOVL      SSS$NOACNT,R0    ; *****ONLY FOR ERROR CHECKING****
;          JMP        GEXES$FINISHIO    ; *****ONLY FOR ERROR CHECKING****
;          PUSHHR    M<R2,R3>          ; SAVE R2 AND R3 FROM BUFFERQUOTA
;          JSB        GEXES$BUFFERQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
;          POPR       M<R2,R3>          ; RESTORE R2 AND R3
11$: BLBS    R0,10$          ; IF ERROR WE EXCEEDED QUOTA
10$: JMP     GEXES$ABORTIO    ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
;          PUSHHR    M<R3>              ; SAVE IRP ADDRESS FROM ALLOCBUF
;          MOVL      R1,R9              ; SAVE BUFFER SIZE IN R9 JUST FOR GRINS
;          ADDL2     12,R1              ; SAVE BUFFER SIZE TO CHARGE PROCESS
;          JSB        GEXES$ALLOCBUF    ; ALLOCATE SOME NON-PAGED POOL FOR THIS
;          POPR       M<R3>              ; RESTORE IRP ADDRESS TO R3
;          BLBC      R0,11$            ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
;          ADDL3     R2, 12,(R2)        ; INIT FIRST LONGWORD OF BUFFER WITH
;                                     ; ADDRESS OF DATA AREA
;          MOVL      R2,IRP$LSVAPTE(R3) ; PUT ADDRESS OF SYSTEM BUFFER
;          MOVL      P1(AP),4(R2)      ; INIT SECOND LONGWORD WITH USER BUFFER
;                                     ; ADDRESS
;          MOVL      PCB$LJIB(R4),R0   ; JET JIB ADDRESS
;          SUBL      R9,JIB$LBYTCNT(R0) ; CHARGE PROCESS FOR BUFFER SPACE USED
;          PUSHHR    M<R1,R2,R3,R4,R5> ; SAVE ALL THESE FOR THE MOV
;          MOVVC3    P2(AP),04(R2),0(R2) ; MOVE USER BUFFER INTO SYSTEM BUFFER
;          POPR       M<R1,R2,R3,R4,R5> ; RESTORE THESE NOW AFTER MOV
;          MOVW      R9,IRP$SWBOFF(R3) ; NUMBER OF BYTES CHARGED AGAINST
;                                     ; USER'S PROCESS QUOTA
;          JMP        GEXES$QIODRVPKT  ; NOW GO QUEUE I/O REQUEST PACKET
AVREADAVACFDT: ; READ FDT routine
SUBW2     IOSREADBLK-IOSREADBLK,-    ; SET I/O FUNCTION CODE IN IRP
;          IRP$WFUNC(R3)
;          MOVQ      P1(AP),R0          ; MOVE BUFFER ADDRESS IN R0
;          ; AND BUFFER SIZE      IN R1
;          TSTL      R1                ; IF BUFFER SIZE <=0 WE HAVE PROBLEMS
;          BGTR      51$                ; OTHERWISE LETS GET ON WITH IT
;          JMP        GEXES$FINISHIO    ; BAD BUFFER SIZE
51$: JSB     GEXES$READCH      ; ABORTS AND DOESN'T COME BACK IF IT
                                ; CAN'T WRITE TO BUFFER
;          PUSHHR    M<R0,R3>          ; SAVE R0 AND R3 FROM BUFFERQUOTA
;          ADDL2     12,R1              ; IF ERROR WE EXCEEDED QUOTA
;          JSB        GEXES$BUFFERQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
;          BLBS      R0,511$           ; IF ERROR WE EXCEEDED QUOTA
511$: JMP     GEXES$ABORTIO    ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
510$: JSB     GEXES$ALLOCBUF    ; ALLOCATE SOME NON-PAGED POOL FOR THIS
;          BLBC      R0,511$           ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
;          POPR       M<R0,R3>          ; RESTORE R0 AND R3
;          MOVL      R2,IRP$LSVAPTE(R3) ; PUT ADDRESS OF SYSTEM BUFFER
;          MOVW      R1,IRP$SWBOFF(R3) ; BYTE QUOTA CHARGED
;          PUSHL     R0
;          MOVL      PCB$LJIB(R4),R0   ; JET JIB ADDRESS

```

# AVA FRAME BUFFER I/O DRIVER

```

        SUBL    R1,JIB$LBTCNT(R0)      ; CHARGE PROCESS FOR BUFFER SPACE USED
        POPL    R0
        MOVAB   12(R2),(R2)+           ; SAVE DATA AREA ADDRESS
        MOVL    R0,(R2)                ; SAVE USER BUFFER ADDRESS
        JMP     GEX$QIODRVPKT          ; NOW GO QUEUE I/O REQUEST PACKET
        .SBTTL  AVSTART, Start I/O routine
;
; ++
; AVSTART - Start a transmit, receive data from or to AVA INTERFACE
;
; Functional description:
;
;     START A READ OR WRITE TO ON LINE DIGITIZER AVA INTERFACE
;
; Inputs:
;
;     R3        - address of the IRP (I/O request packet)
;     R5        - address of the UCB (unit control block)
;
; Outputs:
;
;     R0        - 1st longword of I/O status: contains status code and
;                number of bytes transferred
;     R1        - 2nd longword of I/O status: device-dependent
;
;     The routine must preserve all registers except R0-R2 and R4.
;
; --
AVSTART:
        DSBINT  UCBSBDIPL(R5)          ; Process an I/O packet
        ADDL2   BUFOVRHD,UCBS$SVAPTE(R5); DISABLE INTERRUPTS
        REQPCAN UCBSWBCNT(R5),R1        ; SKIP SYS BUF HEADER
        MOVZWL  UCBSWBCNT(R5),R1        ; PUTS CSR ADDRESS IN R4
        ASHL    -1,R1,R1
        MOVW    R1,UCBSWAVBYTCNT(R5)
        MOVW    UCBSWBCNT(R5),UCBSWAVBYTCNT(R5); MOVE BYTE COUNT TO
                                                ; NEW UCB FIELD
        CLRW    UCBSWBCNT(R5)          ; CLEAR UCB BYTE COUNT
        MOVW    P5(AP),UCBSWAVCSR(R5)  ; INITIALIZE ACCES CONTROL REGISTER BITS
        MOVZWL  UCBSWAVCSR(R5),R1
        BLBS    R1,NOINIT
        BLBS    R1,CHECKSSW
        MOVW    UCBSWAVCSR(R5),AVCSR(R4)
        MOVW    1,AVCSR(R4)
        BISW3   UCBSWAVCSR(R5), 1,AVCSR(R4)
        MOVW    0,CPUIMAIN(R4)
        MOVW    01776,CPUIXFENCE(R4)
        MOVW    0777,CPUIFYFENCE(R4)
        MOVW    0000035,CPUIACR(R4)
        MOVW    P6(AP),UCBSWAVACR(R5)  ; SET UP ACR
        MOVW    UCBSWAVACR(R5),CPUIACR(R4)
        MOVW    UCBSWAVXADDR(R5),CPUIXADDR(R4)
        MOVW    UCBSWAVYADDR(R5),CPUIYADDR(R4)
CHECKSSW:
        BITW    04000,UCBSWAVCSR(R5)  ; TEST FOR MEMORY WINDOW ENABLE
        BEQL    NOINIT

```

# AVA FRAME BUFFER I/O DRIVER

```

MOVW    UCBSWAVCSR(R5),AVCSR(R4)
;
; WE HAVE A REQUEST TO CHECK THE AVA SPECIAL STATUS WORD
;
SSW:
MOVW    @IRP$LSVAPTE(R3),UCB$LSVAPTE(R5) ; GET BUFFER ADDRESS
SETIPL  IPL$POWER ; CHECK FOR POWER FAIL
BBCC    UCB$VPOWER,-
;
; UCBSWSTS(R5),-
; WAITSPREAD
;
WAITSPREAD:
MOVW    CPUICOMP3(R4),@UCB$LSVAPTE(R5); READ INPUT DATA REGISTER
MOVW    1,AVCSR(R4)
ENBINT
;
; THIS MOVW READS FROM THE OVERLAY COMPONENT WHICH WE REALLY DON'T
; HAVE AND SINCE BIT 11 IS SET IN THE AVA CSR MBA16 IS SET WHICH
; PUTS THE SPECIAL AVA STATUS ON THE MASTER BUS DATA BUS
BRW     FINISH
;
NOINIT:
MOVW    P3(AP),UCBSWAVXADDR(R5) ; STARTING X ADDRESS
MOVW    P4(AP),UCBSWAVYADDR(R5) ; STARTING Y ADDRESS
;
MOVW    UCBSWAVXADDR(R5),CPUIXADDR(R4)
MOVW    UCBSWAVYADDR(R5),CPUIYADDR(R4)
ENBINT
CMPZV   IRP$VFCODE, IRP$SFCODE,-
IRP$WFUNC(R3), IOSREADPBLK
BEQL    READD ; WANT TO GO READ AVA
;
DSBINT  UCBSBDIPL(R5) ; DISABLE INTERRUPTS
;
WRITE:
MOVW    @UCB$LSVAPTE(R5),AVCSR(R4) ; CSR BIT TESTING.....
MOVW    @UCB$LSVAPTE(R5),CPUICOMP3(R4)
MOVW    AVCSR(R4),UCBSWAVCSR(R5) ; PUT THE CSR IN IOSB STATUS WORD
SETIPL  IPL$POWER ; CHECK FOR POWER FAIL
BBCC    UCB$VPOWER,-
;
; UCBSWSTS(R5),-
; WAITWRITE
;
ENBINT
RELCHAN
MOVZWL  SSS$POWERFAIL,R#
REQCOM
;
WAITWRITE:
WFIKPC  AVTIMEOUT, AVTIMEOUTSEC
INCL    UCBSWBCNT(R5) ; INCREMENT NUMBER OF WORDS TRANSFERED
MOVW    AVOUTBUF(R4),UCBSWAVOUTBUF(R5); DEVICE OUTPUT REGISTER
MOVW    AVA74(R4),UCBSWAVCSR(R5) ; PUT THE CSR IN IOSB STATUS WORD
IOFORK
INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
INCL    UCB$LSVAPTE(R5)
ADDL2   2,UCB$LSVAPTE(R5)
DECW    UCBSWAVBYTCNT(R5) ; DECREMENT BYTE COUNT TO SEE IF DONE
DECW    UCBSWAVBYTCNT(R5) ; DECREMENT BYTE COUNT TO SEE IF DONE
SUBL2   2,UCBSWAVBYTCNT(R5)

```

# AVA FRAME BUFFER I/O DRIVER

```

        BGTR    WRITE
        ENBINT
FINISH: RELCHAN

;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
        INSV    UCBSWBCNT(R5), 16,-      ; Load number of bytes trans-
        16,R0      ; ferred into high word of R0.
        MOVW    SSSNORMAL,R0           ; Load a success code into R0.
;
        INSV    UCBSWAVOUTBUF(R5), 16,- ; LOAD OUTBUF IN IOSB(4)
        16,R1
;
        INSV    0, 16, 16,R1           ; CLEAR UPPER WORD. IOSB(4)
;
        MOVW    UCBSWAVCSR(R5),R1 ; PUT THE IN6JF IN IOSB STATUS WORD
        MOVW    UCBSWAVOUTBUF(R5),R1
        MOVW    0,R1
;
; Call I/O postprocessing.
;
COMPLETEIO:                                ; Driver processing is finished.
        REQCOM                                ; Complete I/O.

;
; READ LOOP
;
READD:  MOVL    @IRPSLSVAPTE(R3),UCBSLSVAPTE(R5) ; GET BUFFER ADDRESS
;
        DSBINT  UCBSBBDIPL(R5)           ; DISABLE INTERRUPTS
        SETIPL  IPLSPower                ; CHECK FOR POWER FAIL
        BBCC    UCBSVPOWER,-
        UCBSWSTS(R5),-
        READ
;
READ:   WFIKPC  AVTIMEOUT, AVTIMEOUTSEC
;
        MOVW    AVINBUF(R4),@UCBSLSVAPTE(R5); READ INPUT DATA REGISTER
        MOVW    CPUICOMP0(R4),@UCBSLSVAPTE(R5); READ INPUT DATA REGISTER
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
;
        INCL    UCBSWBCNT(R5)           ; INCREMENT NUMBER OF WORDS TRANSFERED
;
        MOVW    AVINBUF(R4),UCBSWAVCSR(R5); PUT THE INBUF IN IOSB STATUS WORD
;
        MOVW    AVOUTBUF(R4),UCBSWAVOUTBUF(R5); DEVICE OUTPUT REGISTER
;
        IOFORK
;
        INCL    UCBSLSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
;
        INCL    UCBSLSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
;
        ADDL2    2,UCBSLSVAPTE(R5)
;
        DECV    UCBSWAVBYTCNT(R5)      ; DECREMENT BYTE COUNT TO SEE IF DONE
;
        DECV    UCBSWAVBYTCNT(R5)      ; DECREMENT BYTE COUNT TO SEE IF DONE
;
        SUBL2    2,UCBSWAVBYTCNT(R5)
;
        BGTR    READ
        ENBINT
        BRW     FINISH
;
; Device timeout handling. Return an error status code.

```

# AVA FRAME BUFFER I/O DRIVER

```

;
AVTIMEOUT:                                ; Timeout handling
; BICW2    <AVCSRMCSR>,AVCSR(R4); SET CONTROL LINE # LOW
; MOVW     #,AVCSR(R4)
; SETIPL   UCBSBFIPL(R5)                ; Lower to driver fork IPL
; MOVZWL   SS$TIMEOUT,R#                ; Return error status.
; MOVL     63,R1                        ; .. STATUS TESTING..
; MOVW     AVCSR(R4),UCBSWAVCSR(R5) ; PUT THE CSR IN IOSB STATUS WORD
; MOVW     UCBSWAVCSR(R5),R1 ; PUT THE CSR IN IOSB STATUS WORD
; INSV     UCBSWAVOUTBUF(R5), 16,- ;LOAD OUTBUF IN IOSB(4)
;          16,R1
; BRW      COMPLETEIO                  ; Call I/O postprocessing.
; .SBTTL   AVINTERRUPT, Interrupt service routine
;
;++
; AVINTERRUPT, Analyzes interrupts, processes solicited interrupts
;
; Functional description:
;
;     The sample code assumes either
;
;         that the driver is for a single-unit controller, and
;         that the unit initialization code has stored the
;         address of the UCB in the IDB; or
;
;         that the driver's start I/O routine acquired the
;         controller's channel with a REQCHANL macro call, and
;         then invoked the WFIKPB macro to keep the channel
;         while waiting for an interrupt.
;
; Inputs:
;
;     # (SP) - pointer to the address of the IDB (interrupt data
;             block)
;     4 (SP) - saved R#
;     8 (SP) - saved R1
;     12 (SP) - saved R2
;     16 (SP) - saved R3
;     20 (SP) - saved R4
;     24 (SP) - saved R5
;     28 (SP) - saved PSL (program status longword)
;     32 (SP) - saved PC
;
;     The IDB contains the CSR address and the UCB address.
;
; Outputs:
;
;     The routine must preserve all registers except R#-R5.
;
;--
AVINTERRUPT:                                ; Service device interrupt
; MOVL     @(SP)+,R4                    ; Get address of IDB and remove
;                                               ; pointer from stack.
; MOVL     IDBSLOWNER(R4),R5            ; Get address of device owner's
;                                               ; UCB.
; MOVL     IDBSLCSR(R4),R4 ; Get address of device's CSR.

```

# AVA FRAME BUFFER I/O DRIVER

```

;      BICW2    <AVCSRMCSR0>,AVCSR(R4); SET CONTROL LINE 0 LOW
;      BBCC     UCBS$VINT,-             ; If device does not expect
;      UCBS$WSTS(R5),-                 ; interrupt, dismiss it.
;      UNSOLINTERRUPT
;
; This is a solicited interrupt. Save
; the contents of the device registers in the UCB. NOT NEEDED IN THIS DRIVER
;
; Restore control to the main driver.
;
RESTOREDRIVER:
;      MOVL     UCBS$LFR3(R5),R3 ; Restore driver's R3 (use a
;                               ; MOVQ to restore R3-R4).
;      JSB      @UCBS$LFP3(R5)    ; Call driver at interrupt
;                               ; wait address.
;
; Dismiss the interrupt.
;
UNSOLINTERRUPT:
;      POPR     M<R0,R1,R2,R3,R4,R5> ; Restore R0-R5
;      REI      ; Return from interrupt.
;      .SBTTL   AVCANCEL, Cancel I/O routine
;
; ++
; AVCANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;      This routine calls IOC$CANCELIO to set the cancel bit in the
;      UCB status word if:
;
;          the device is busy,
;          the IRP's process ID matches the cancel process ID,
;          the IRP channel matches the cancel channel.
;
;      If IOC$CANCELIO sets the cancel bit, then this driver routine
;      does device-dependent cancel I/O fixups.
;
; Inputs:
;
;      R2      - channel index number
;      R3      - address of the current IRP (I/O request packet)
;      R4      - address of the PCB (process control block) for the
;                process canceling I/O
;      R5      - address of the UCB (unit control block)
;      R8      - cancel reason code, one of:
;                CAN$CCANCEL      if called through $CANCEL or
;                                $DALLOC system service
;                CAN$CDASSGN     if called through $DASSGN system
;                                service
;                These reason codes are defined by the $CANDEF macro.
;
; Outputs:
;
;      The routine must preserve all registers except R0-R3.

```

# AVA FRAME BUFFER I/O DRIVER

```

;
;   The routine may set the UCBSMCANCEL bit in UCBSWSTS.
;
;--
AVCANCEL:
    JSB      GIOCSCANCELIO      ; Cancel an I/O operation
    BBC      UCBSVCANCEL,-      ; Set cancel bit if appropriate.
                                ; If the cancel bit is not set,
                                ; just return.
    UCBSWSTS(R5),10$
;
; Device-dependent cancel operations go next.
;
; Finally, the return.
;
10$:
    RSB      ; Return
    .SBTTL   AVREGDUMP, Device register dump routine
;+
; AVREGDUMP, Dumps the contents of device registers to a buffer
;
; Functional description:
;
;   Writes the number of device registers, and their current
;   contents into a diagnostic or error buffer.
;
; Inputs:
;
;   R0      - address of the output buffer
;   R4      - address of the CSR (controller status register)
;   R5      - address of the UCB (unit control block)
;
; Outputs:
;
;   The routine must preserve all registers except R1-R3.
;
;   The output buffer contains the current contents of the device
;   registers. R0 contains the address of the next empty longword in
;   the output buffer.
;
;--
AVREGDUMP:
    MOVZBL   AVNUMREGS,(R0)+      ; Dump device registers
                                ; Store device register count.
    MOVZWL   UCBSWAVBYTCNT(R5),-  ; Store BYTE count register.
                                ; (R0)+
    RSB      ; Return
    .SBTTL   AVEND, End of driver
;+
; Label that marks the end of the driver
;--
AVEND:
    .END      ; Last location in driver

```

## APPENDIX AJ

```
.TITLE ODDRIVER - VAX/VMS ON LINE DIGITIZER TAPE CONTROLLER DRIVER (DR11-C)
.IDENT 'V03-001'
```

✿✿

**FACILITY:**

```

; VAX/VMS On Line Digitizer Tape controller driver (DR11-C)

```

**ABSTRACT:**

```

; This module contains the driver:

```

```

:      Tables
:      Controller and unit initialization routines
:      The FDT routine
:      The start I/O routine
:      The interrupt service routine
:      The cancel I/O routine
:      The device register dump routine

```

; AUTHOR:

S. Richard F. Sims Aug. 23, 1982

; REVISION HISTORY:

1-1

```

.SBTTL  External and local symbol definitions

```

External symbols

```

SCANDEF      ; Cancel reason codes
SCR8DEF      ; Channel request block
$DCDEF       ; Device classes and types
$DDBDEF      ; Device data block
$DEVDEF      ; Device characteristics
$IDBDEF      ; Interrupt data block
$IODEF       ; I/O function codes
$IPLDEF      ; Hardware IPL definitions
$IIRPDEF     ; I/O request packet

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

SSSDEF          ; System status codes
SUCBDEF         ; Unit control block
SVECDEF        ; Interrupt vector block
SJIBDEF        ; JOB INFO BLOCK OFFSET DEFS
SPCBDEF        ; PROCESS CONTROL BLOCK OFFSET DEFS
;
; Local symbols
;
;
; Argument list (AP) offsets for device-dependent QIO parameters
;
P1      = 0      ; First QIO parameter
P2      = 4      ; Second QIO parameter
P3      = 8      ; Third QIO parameter
P4      = 12     ; Fourth QIO parameter
P5      = 16     ; Fifth QIO parameter
P6      = 20     ; Sixth QIO parameter
;
; Other constants
;
ODDEFBUFSIZ     = 1      ; Default buffer size
ODTIMEOUTSEC    = 10     ; 10 second device timeout
ODNUMREGS       = 3      ; Device has 3 registers
BUFOVRHD        = 12     ; SYSTEM BUFFER OVERHEAD FOR BUFFERED I/O
;
; Definitions that follow the standard UCB fields
;
SDEFINI UCB      ; Start of UCB definitions
.=UCBSKLENGTH   ; Position at end of UCB
SDEF UCBSWODCSR   ; Device's CSR register
      .BLKW 1
SDEF UCBSWODBYTCNT ; Device's BYTE count register
      .BLKW 1
SDEF UCBSWODOUTBUF ; DEVICE OUTBUF REGISTER
      .BLKW 1
SDEF UCBSKODUCBLEN ; Length of extended UCB
      .BLKW 1
;
; Bit positions for device-dependent status field in UCB
;
SVIELD UCBCSR,0,<- ; Device status
      <BITZERO,,M>,- ; First bit
      <BITONE,,M>,- ; Second bit
      >
SDEFEND UCB       ; End of UCB definitions
;
; Device register offsets from CSR address
;
SDEFINI OD       ; Start of status definitions
SDEF ODCSR       ; Control/status
      .BLKW 1
;
; Bit positions for device control/status register
;
VIELD ODCSR,0,<- ; Control/status register

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

        <CSR0,,M>,-      ; COMMAND BIT 0
        <CSR1,,M>,-      ; COMMAND BIT 1
        <,3>,-           ; THREE UNUSED BITS
        <IEB,,M>,-       ; ENABLE REQUEST B INTERRUPTS
        <IEA,,M>,-       ; Enable REQUEST A interrupts
        <REQA,,M>,-      ; UNDER CONTROL OF USER DEVICE
;                          ; NORMALLY USED FOR READY INDICATIONS
        <,7>,-           ; SEVEN Disregarded bits
        <REQB,,M>,-      ; UNDER CONTROL OF USER DEVICE
;                          ; NORMALLY USED FOR ERROR CONDITIONS
;
;
SDEF    >
SDEF    ODOUTBUF          ; OUTPUT BUFFER WORD
;
SDEF    ODINBUF           ; INPUT BUFFER WORD
        .BLKW 1
SDEFEND OD                ; End of device register
;                          ; definitions.
        .SBTTL Standard tables
;
; Driver prologue table
;
DPTAB -                    ; DPT-creation macro
        END=ODEND,-        ; End of driver label
        ADAPTER=UBA,-      ; Adapter type
        UCBSIZE=<UCBSKODUCBLEN>,- ; Length of UCB
        NAME=ODDRIVER      ; Driver name
DPTSTORE INIT              ; Start of load
;                          ; initialization table
DPTSTORE UCB,UCBSBFIPL,B,8 ; Device fork IPL
DPTSTORE UCB,UCBSBDIPL,B,21 ; Device interrupt IPL=21=BR5
DPTSTORE UCB,UCBSLDEVCHAR,L,<- ; Device characteristics
        DEVSMIDVI-        ; input device
        DEVSMAVLI-
        DEVSMODV>        ; output device
DPTSTORE UCB,UCBSBDEVCLASS,B,DCSSCOM ; Device class?
DPTSTORE UCB,UCBSBDEVTYPE,B,DTSDR11C ; DEVICE TYPE
DPTSTORE UCB,UCBSWDEVBUFSIZ,W,- ; Default buffer size
        ODDEFBUFSIZ
DPTSTORE REINIT            ; Start of reload
;                          ; initialization table
DPTSTORE DDB,DBBSLDDT,D,ODSDDT ; Address of DDT
DPTSTORE CRB,CRBSLINTD+4,D,- ; Address of interrupt
        ODINTERRUPT        ; service routine REQ A
DPTSTORE CRB,CRBSLINTD2+4,D,- ; REQ B INTERRUPT ROUTINE
        ODINTERRUPT
DPTSTORE CRB,-             ; Address of controller
        CRBSLINTD+VECSLINITIAL,- ; initialization routine
        D,ODCONTROLINIT
DPTSTORE CRB,-             ; Address of device
        CRBSLINTD+VECSLUNITINIT,- ; unit initialization
        D,ODUNITINIT        ; routine
DPTSTORE END               ; End of initialization
;                          ; tables
;
; Driver dispatch table

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

;
; ODTAB - DDT-creation macro
;         DEVNAM=OD,-      ; Name of device
;         START=ODSTART,-  ; Start I/O routine
;         FUNCTB=ODFUNCTABLE,- ; FDT address
;         CANCEL=ODCANCEL,- ; Cancel I/O routine
;         REGDMP=ODREGDUMP ; Register dump routine
;
; Function decision table
;
; ODFUNCTABLE:
;         FUNCTAB ,-      ; FDT for driver
;         <READVBLK,-      ; Valid I/O functions
;         READLBLK,-      ; Read virtual
;         READPBLK,-      ; Read logical
;         WRITEVBLK,-     ; Read physical
;         WRITELBLK,-     ; Write virtual
;         WRITEPBLK>      ; Write logical
;         FUNCTAB ,-      ; Write physical
;         <READVBLK,-     ; Buffered functions
;         READLBLK,-      ; Read virtual
;         READPBLK,-      ; Read logical
;         WRITEVBLK,-     ; Read physical
;         WRITELBLK,-     ; Write virtual
;         WRITEPBLK>      ; Write logical
;         FUNCTAB ODWRITEDR11CFDT,- ; Write physical
;         <WRITEVBLK,-     ; Write virtual
;         WRITELBLK,-     ; Write logical
;         WRITEPBLK>      ; Write physical
;         FUNCTAB ODREADDR11CFDT,- ; Read virtual
;         <READVBLK,-      ; Read logical
;         READLBLK,-      ; Read physical
;         READPBLK>      ; SET ALL BITS FOR THE
;         .LONG -1        ; FDT CATCH ALL ERROR ROUTINE
;         .LONG -1
;         .ADDRESS OOPS
;         .SBTTL ODCONTROLINIT, Controller initialization routine
;
; ++
; ODCONTROLINIT, Readies controller for I/O operations
;
; Functional description:
;
;     The operating system calls this routine in 3 places:
;
;         at system startup
;         during driver loading and reloading
;         during recovery from a power failure
;
; Inputs:
;
;     R4 - address of the CSR (controller status register)
;     R5 - address of the IDB (interrupt data block)
;     R6 - address of the DDB (device data block)
;     R8 - address of the CRB (channel request block)
;

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

; Outputs:
;
;   The routine must preserve all registers except R0-R3.
;
;--
ODCONTROLINIT:                ; Initialize controller
    RSB                        ; Return
    .SBTTL  ODUNITINIT, Unit initialization routine
;
;++
; ODUNITINIT, Readies unit for I/O operations
;
; Functional description:
;
;   The operating system calls this routine after calling the
;   controller initialization routine:
;
;       at system startup
;       during driver loading
;       during recovery from a power failure
;
; Inputs:
;
;   R4      - address of the CSR (controller status register)
;   R5      - address of the UCB (unit control block)
;
; Outputs:
;
;   The routine must preserve all registers except R0-R3.
;
;--
ODUNITINIT:                    ; Initialize unit
    BISW      UCBS$MONLINE, -   ; Set unit online
    RSB      UCBS$WSTS(R5)      ; Return
    .SBTTL  ODFDTRoutine, ON LINE DIGITIZER DR11-C FDT routine
;
;++
; ODFDTRoutine, ON LINE DIGITIZER DR11-C FDT routine
;
; Functional description:
;
;   SET UP FOR BUFFERED IO ON THIS DR11-C
;
; Inputs:
;
;   R0-R2   - scratch registers
;   R3      - address of the IRP (I/O request packet)
;   R4      - address of the PCB (process control block)
;   R5      - address of the UCB (unit control block)
;   R6      - address of the CCB (channel control block)
;   R7      - bit number of the I/O function code
;   R8      - address of the FDT table entry for this routine
;   R9-R11  - scratch registers
;   AP      - address of the 1st function dependent QIO parameter
;
; Outputs:

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

;
; The routine must preserve all registers except R0-R2, and
; R9-R11.
;
;--
;
; CATCH ALL FDT ERROR ROUTINE
;
OOPS:
    MOVL    SSSILLIOFUNC,R0      ; ILLEGAL I/O FUNCTION SPECIFIED
    JSB     GEXESABORTIO        ; SO LET'S ABORT
ODWRITEDR11CFDT:
    MOVQ    P1(AP),R0           ; WRITE FDT routine
                                ; MOVE BUFFER ADDRESS IN R0
                                ; AND BUFFER SIZE IN R1
                                ; IF BUFFER SIZE <=0 WE HAVE PROBLEMS
                                ; OTHERWISE LETS GET ON WITH IT
    TSTL    R1
    BGTR    1$
    MOVL    SSSIVBUFLN,R0      ; MOVE THE ERROR STATUS INTO R0
    JMP     GEXESFINISHIO      ; BAD BUFFER SIZE
1$:
    JSB     GEXESWRITECHK      ; ABORTS AND DOESN'T COME BACK IF IT
                                ; CAN'T WRITE TO BUFFER
                                ; *****ONLY FOR ERROR CHECKING****
                                ; *****ONLY FOR ERROR CHECKING****
    MOVL    SSSNOACNT,R0
    JMP     GEXESFINISHIO
    PUSHR   M<R2,R3>           ; SAVE R2 AND R3 FROM BUFFERQUOTA
    JSB     GEXESBUFFERQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
    POPR    M<R2,R3>           ; RESTORE R2 AND R3
    BLBS    R0,10$
11$:
    JMP     GEXESABORTIO      ; IF ERROR WE EXCEEDED QUOTA
10$:
    PUSHR   M<R3>
    MOVL    R1,R9
    ADDL2   12,R1
    JSB     GEXESALLOCBUF     ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
    POPR    M<R3>             ; SAVE IRP ADDRESS FROM ALLOCBUF
    BLBC    R0,11$
    ADDL3   R2, 12,(R2)       ; SAVE BUFFER SIZE IN R9 JUST FOR GRINS
                                ; SAVE BUFFER SIZE TO CHARGE PROCESS
                                ; ALLOCATE SOME NON-PAGED POOL FOR THIS
    MOVL    R2,IRPSLVAPTE(R3) ; RESTORE IRP ADDRESS TO R3
    MOVQ    P1(AP),4(R2)      ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
                                ; INIT FIRST LONGWORD OF BUFFER WITH
                                ; ADDRESS OF DATA AREA
                                ; PUT ADDRESS OF SYSTEM BUFFER
                                ; INIT SECOND LONGWORD WITH USER BUFFER
                                ; ADDRESS
    MOVL    PCBSLJIB(R4),R0 ; JET JIB ADDRESS
    SUBL    R9,JIBSLBYTCNT(R0) ; CHARGE PROCESS FOR BUFFER SPACE USED
    PUSHR   M<R1,R2,R3,R4,R5> ; SAVE ALL THESE FOR THE MOVC
    MOVC3   P2(AP),04(R2),0(R2) ; MOVE USER BUFFER INTO SYSTEM BUFFER
    POPR    M<R1,R2,R3,R4,R5> ; RESTORE THESE NOW AFTER MOVC
    MOVW    R9,IRPSWBOFF(R3) ; NUMBER OF BYTES CHARGED AGAINST
                                ; USER'S PROCESS QUOTA
                                ; NOW GO QUEUE I/O REQUEST PACKET
    JMP     GEXESQIODRVPKT
ODREADDR11CFDT:
    SUBW2   IOSREADLBLK-IOSREADPBLK,- ; READ FDT routine
                                ; SET I/O FUNCTION CODE IN IRP
    MOVQ    P1(AP),R0           ; MOVE BUFFER ADDRESS IN R0
                                ; AND BUFFER SIZE IN R1
                                ; IF BUFFER SIZE <=0 WE HAVE PROBLEMS
                                ; OTHERWISE LETS GET ON WITH IT
    TSTL    R1
    BGTR    51$
    JMP     GEXESFINISHIO      ; BAD BUFFER SIZE
51$:
    JSB     GEXESREADCHK      ; ABORTS AND DOESN'T COME BACK IF IT

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

PUSHR    M<R0,R3>                ; CAN'T WRITE TO BUFFER
ADDL2    12,R1                    ; SAVE R0 AND R3 FROM BUFRQUOTA
JSB      R0,510$                  ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
BLBS     R0,510$                  ; IF ERROR WE EXCEEDED QUOTA
511$:    JMP      GEXESABORTIO      ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
510$:    JSB      GEXESALLOCBUF     ; ALLOCATE SOME NON-PAGED POOL FOR THIS
BLBC     R0,511$                  ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
POPR     M<R0,R3>                ; RESTORE R0 AND R3
MOVL     R2,IRPSLSVAPTE(R3)       ; PUT ADDRESS OF SYSTEM BUFFER
MOVW     R1,IRPSWBOFF(R3)         ; BYTE QUOTA CHARGED
PUSHL    R0
MOVL     PCBSLJIB(R4),R0          ; JET JIB ADDRESS
SUBL     R1,JIBSLBYTCNT(R0)       ; CHARGE PROCESS FOR BUFFER SPACE USED
POPL     R0
MOVAB    12(R2),(R2)+             ; SAVE DATA AREA ADDRESS
MOVL     R0,(R2)                  ; SAVE USER BUFFER ADDRESS
JMP      GEXESQODRVPKT           ; NOW GO QUEUE I/O REQUEST PACKET
.SBTTL   ODDSTART, Start I/O routine

```

```

;--
; ODDSTART - Start a transmit, receive data from or to drill-c
;
; Functional description:
;
;     START A READ OR WRITE TO ON LINE DIGITIZER DR11-C
;
; Inputs:
;
;     R3      - address of the IRP (I/O request packet)
;     R5      - address of the UCB (unit control block)
;
; Outputs:
;
;     R0      - 1st longword of I/O status: contains status code and
;               number of bytes transferred
;     R1      - 2nd longword of I/O status: device-dependent
;
;     The routine must preserve all registers except R0-R2 and R4.
;
;--

```

```

ODDSTART:
ADDL2     BUFOVRHD,UCBSLSVAPTE(R5) ; Process an I/O packet
REQPCHAN  ; SKIP SYS BUF HEADER
MOVW      UCBSWBCNT(R5),UCBSWODBYTCNT(R5) ; PUTS CSR ADDRESS IN R4
;               ; MOVE BYTE COUNT TO
;               ; NEW UCB FIELD
CLRW      UCBSWBCNT(R5)            ; CLEAR UCB BYTE COUNT
CLRW      ODCSR(R4)                ; CLEAR CSR
EXTZV     IRPSVFCD, IRPSSFCD,-
;         IRPSWFUNC(R3),R2
;
CMPL      IOSREADPBLK,R2
CMPZV     IRPSVFCD, IRPSSFCD,-
;         IRPSWFUNC(R3), IOSREADPBLK
BEQL      READD                    ; WANT TO GO READ DR11-C
WRITE:    BRB      READD

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

DSBINT  UCBS$BDIPL(R5)          ; DISABLE INTERRUPTS
MOVW    @UCBS$LSVAPTE(R5),ODOUTBUF(R4); PUT DATA INTO DEVICE OUTPUT REGISTER
BISW2   <ODCSRMIEA>,- ;+ODCSRMCSR0
        ODCSR(R4)              ;ENABLE DEVICE TO INTERRUPT
SETIPL  IPL$POWER                ; CHECK FOR POWER FAIL
BBCC    UCBS$VPOWER,-
        UCBS$WSTS(R5),-
        WAITWRITE

ENBINT
RELCHAN
MOVZWL  SSS$POWERFAIL,R0
REQCOM

WAITWRITE:
WFIKPC  ODTIMEOUT, ODTIMEOUTSEC
INCL    UCBS$WBCNT(R5)          ; INCREMENT NUMBER OF WORDS TRANSFERED
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
MOVW    ODOUTBUF(R4),UCBS$WODOUTBUF(R5); DEVICE OUTPUT REGISTER
MOVW    ODINBUF(R4),UCBS$WODCSR(R5) ; PUT THE INBUF IN IOSB STATUS WORD
IOFORK
INCL    UCBS$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
INCL    UCBS$LSVAPTE(R5)
DECW    UCBS$WODBYTCNT(R5)      ; DECREMENT BYTE COUNT TO SEE IF DONE
DECW    UCBS$WODBYTCNT(R5)      ; DECREMENT BYTE COUNT TO SEE IF DONE
BGTR    WRITE
FINISH: RELCHAN

INSV    UCBS$WBCNT(R5), 16,-      ; Load number of bytes trans-
        16,R0                    ; ferred into high word of R0.
MOVW    SSS$NORMAL,R0            ; Load a success code into R0.
INSV    UCBS$WODOUTBUF(R5), 16,- ;LOAD OUTBUF IN IOSB(4)
        16,R1
;
INSV    0, 16, 16,R1             ; CLEAR UPPER WORD. IOSB(4)
MOVW    UCBS$WODCSR(R5),R1 ; PUT THE INBUF IN IOSB STATUS WORD
;
; Call I/O postprocessing.
;
COMPLETEIO:
REQCOM                                ; Driver processing is finished.
; Complete I/O.
;
; READ LOOP
;
READD:  MOVL    @IRP$LSVAPTE(R3),UCBS$LSVAPTE(R5)      ; GET BUFFER ADDRESS
READ:
GO:     DSBINT  UCBS$BDIPL(R5)          ; DISABLE INTERRUPTS
        BISW2   <ODCSRMIEB+ODCSRMCSR0>,- ;ODCSRMIEA+
        ODCSR(R4)              ;ENABLE DEVICE TO INTERRUPT
SETIPL  IPL$POWER                ; CHECK FOR POWER FAIL
BBCC    UCBS$VPOWER,-
        UCBS$WSTS(R5),-
        WAITREAD

ENBINT
RELCHAN

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

MOVZWL    $$$POWERFAIL,R0
REQCOM
WAITREAD:
WFIKPCH ODTIMEOUT, ODTIMEOUTSEC
MOVW     ODINBUF(R4),UCB$SLVAPTE(R5); READ INPUT DATA REGISTER
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
INCL     UCB$WBCNT(R5)          ; INCREMENT NUMBER OF WORDS TRANSFERED
MOVW     ODINBUF(R4),UCB$WODCSR(R5) ; PUT THE INBUF IN IOSB STATUS WORD
MOVW     ODOUTBUF(R4),UCB$WODOUTBUF(R5); DEVICE OUTPUT REGISTER
IOFORK
INCL     UCB$SLVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
INCL     UCB$SLVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
DECW     UCB$WODBYTCNT(R5) ; DECREMENT BYTE COUNT TO SEE IF DONE
DECW     UCB$WODBYTCNT(R5) ; DECREMENT BYTE COUNT TO SEE IF DONE
BGTR     READ
BRW      FINISH
;
; Device timeout handling. Return an error status code.
;
ODTIMEOUT:
MOVW     0,UCB$SLVAPTE(R5) ; Timeout handling
BICW2    <ODCSRMC$R0>,ODCSR(R4); NO DATA PUT OUT
SETIPL   UCB$BFIP(L(R5) ; Lower to driver fork IPL
MOVZWL   SS$TIMEOUT,R0 ; Return error status.
;
MOVL     63,R1 ; .. STATUS TESTING..
MOVW     UCB$WODCSR(R5),R1 ; PUT THE CSR IN IOSB STATUS WORD
BRB      COMPLETEIO ; Call I/O postprocessing.
.SBTTL   ODINTERRUPT, Interrupt service routine
;
;++
; ODINTERRUPT, Analyzes interrupts, processes solicited interrupts
;
; Functional description:
;
; The sample code assumes either
;
; that the driver is for a single-unit controller, and
; that the unit initialization code has stored the
; address of the UCB in the IDB; or
;
; that the driver's start I/O routine acquired the
; controller's channel with a REQCHANL macro call, and
; then invoked the WFIKPCH macro to keep the channel
; while waiting for an interrupt.
;
; Inputs:
;
; 0(SP) - pointer to the address of the IDB (interrupt data
;        block)
; 4(SP) - saved R0
; 8(SP) - saved R1
; 12(SP) - saved R2
; 16(SP) - saved R3

```

```

;
; 28(SP) - saved R4
; 24(SP) - saved R5
; 20(SP) - saved PSL (program status longword)
; 16(SP) - saved PC
;
; The IOB contains the CSR address and the UCB address.
;
; Outputs:
;
; The routine must preserve all registers except R0-R5.
;
;--
ODINTERRUPT:                                ; Service device interrupt
    MOVL    @(SP)+,R4                        ; Get address of IOB and remove
                                           ; pointer from stack.
    MOVL    IDBSLOWNER(R4),R5                ; Get address of device owner's
                                           ; UCB.
    MOVL    IDBSLCSR(R4),R4                  ; Get address of device's CSR.
    BICW2    <ODCSRMCSPR0>,ODCSR(R4); SET CONTROL LINE 0 LOW
    BBCC     UCBSVINT,-                      ; If device does not expect
    UCBSWSTS(R5),-                          ; interrupt, dismiss it.
    UNSOLINTERRUPT
;
; This is a solicited interrupt. Save
; the contents of the device registers in the UCB. NOT NEEDED IN THIS DRIVER
;
; Restore control to the main driver.
;
RESTOREDRIVER:                                ; Jump to main driver code.
    MOVL    UCBSLFR3(R5),R3                  ; Restore driver's R3 (use a
                                           ; MOVQ to restore R3-R4).
    JSB     @UCBSLFPC(R5)                    ; Call driver at interrupt
                                           ; wait address.
;
; Dismiss the interrupt.
;
UNSOLINTERRUPT:                                ; Dismiss unsolicited interrupt.
    POPR     M<R0,R1,R2,R3,R4,R5>           ; Restore R0-R5
    REI      1                               ; Return from interrupt.
    SBTTL    ODCANCEL, Cancel I/O routine
;
; ODCANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
; This routine calls IOCSCANCELIO to set the cancel bit in the
; UCB status word if:
;
;     the device is busy,
;     the IRP's process ID matches the cancel process ID,
;     the IRP channel matches the cancel channel.
;
; If IOCSCANCELIO sets the cancel bit, then this driver routine
; does device-dependent cancel I/O fixups.

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

;
; Inputs:
;
;   R2      - channel index number
;   R3      - address of the current IRP (I/O request packet)
;   R4      - address of the PCB (process control block) for the
;             process canceling I/O
;   R5      - address of the UCB (unit control block)
;   R8      - cancel reason code, one of:
;             CANSCCANCEL      if called through $CANCEL or
;                               $DALLOC system service
;             CANS$DASSGN     if called through $DASSGN system
;                               service
;             These reason codes are defined by the $CANDEF macro.
;
; Outputs:
;
;   The routine must preserve all registers except R8-R3.
;
;   The routine may set the UCB$MCANCEL bit in UCB$WSTS.
;
;--
ODCANCEL:      JSB      GIO$SCANCELIO      ; Cancel an I/O operation
               BBC      UCB$VCANCEL,-      ; Set cancel bit if appropriate.
               UCB$WSTS(R5),10$           ; If the cancel bit is not set,
               ; just return.
;
; Device-dependent cancel operations go next.
;
; Finally, the return.
;
10$:
   RSB          ; Return
   .SBTTL  ODREGDUMP, Device register dump routine
;+
; ODREGDUMP, Dumps the contents of device registers to a buffer
;
; Functional description:
;
;   Writes the number of device registers, and their current
;   contents into a diagnostic or error buffer.
;
; Inputs:
;
;   R8      - address of the output buffer
;   R4      - address of the CSR (controller status register)
;   R5      - address of the UCB (unit control block)
;
; Outputs:
;
;   The routine must preserve all registers except R1-R3.
;
;   The output buffer contains the current contents of the device
;   registers. R8 contains the address of the next empty longword in

```

# ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```

;      the output buffer.
;
;--
ODREGDUMP:
MOVZBL   ODNUMREGS,(R0)+      ; Dump device registers
MOVZWL   UCBSWODBYTCNT(R5),-  ; Store device register count.
              (R0)+          ; Store BYTE count register.

RSB      ; Return
.SBTTL   ODEND, End of driver
;++
; Label that marks the end of the driver
;--
ODEND:
.END      ; Last location in driver

```

DISTRIBUTION

|                                           | <u>No. Copies</u> |
|-------------------------------------------|-------------------|
| US Army Materiel System Analysis Activity |                   |
| ATTN: DRXSY-MP                            | 1                 |
| Aberdeen Proving Ground, MD 21005         |                   |
| DRSMI-R, Dr. McCorkle                     | 1                 |
| Dr. Rhoades                               | 1                 |
| -RE, Mr. Lindberg                         | 1                 |
| -RES                                      | 15                |
| -RPT, Record                              | 1                 |
| -RPR, Reference                           | 15                |
| -LP, Mr. Voigt                            | 1                 |