CSNLIB: A GRAPHICS SUBROUTINE SYSTEM FOR SUPPORTING INTERACTIVE GRAPHICAL APPLICATIONS ON THE PDP-11.[1]

By

A. I. Karshmer

COINS Technical Report 77-12
October 1977

Production and Illustrations by

Joyce Rodriguez

[1] The research reported herein was supported by an Alfred P. Sloan Foundation grant to Michael A. Arbib for the development of a Center for Systems Neuroscience.

TABLE OF CONTENTS

		Page
I.	Introduction	1
II.	The GT-44	
	A. The Hardware	2
	B. The Software	3
III.	CSN Graphics Subroutines	6
	A. Creating Display and Subroutine Files	6
	B. Creating a Named Subroutine	7
	C. Starting and Stopping the Display Processor	9
	D. Blanking - Unblanking and Removing the Display File	9
	E. The Basic Graphics Operations	
	(1) The Vector or Line Mode	10
	(2) The Point Mode	12
	(3) The Text Mode	14
	F. Sample Programs Using the Basic Graphics Operations	15
	(1) A Simple Square	15
	(2) 'CAN YOU SEE ME NOW'	16
	(3) Diagonal Line of Points	17
	G. Connecting Subroutines to Display Files and Connecting Subroutines to Other Subroutines	17
	H. Disconnecting and Connecting Subroutine Linkage from Display and Subroutine Files	21
IV.	The Light Pen and How to Use It	23
V.	Light Pen Tracking	30
VI.	Saving and Restoring Display Files	33
VII.	Ending a Display File in the Middle	40
VIII.	Functions which Return Data Concerning a Display or Subroutine File	41
IX.	Blinking After the Fact	43
X.	A Basic Line Graphics Routine	45
XI.	Other CSN Utility Functions and Subroutines	48
	A. Functions	48
	(1) ISWTCH	48
	(2) JSIGN	50
	(3) IADDR	50
	(4) LOOK	51
	(5) ITTYIN	52
	B. Subroutines	
	(1) IZERO LZERO RZERO	53

Table of Contents (continued)

		Page
	(2) IPUT	53
	(3) TTOUT	54
	(4) SNOOZE	55
XII.	Compiling and Linking FORTRAN Programs Calling CSN Subroutines and Functions	56
	A. Compiling	56
	B. Linking	57
XIII.	Building the CSNLIB on Disk or DECtape	57
	A. Assembling the Macro Programs	57
	B. Compiling the FORTRAN Subroutine	61
	C. Building the CSNLIB using RT-11 Librarian	61
XIV.	Calling CSNLIB Routines from Assembly Language Programs	61
XV.	CSN Graphics Error Messages	66
XVI.	The Internal Structure of Display and Subroutine Files	68
	A. The HEADER Structure	69
	B. The INTERNAL Structure	72
XVII.	Graphics Subroutine Summary	77
XVIII.	Program Listings	83
	A. Directory of Module Names and Entry Points	84
	B. Source Listings	85
	BEGIN.MAC	85
	CLINK.MAC	86
	CHARS.MAC	86
	CLOSE.MAC	88
	DJSR.MAC	88
	DRWREL.MAC	89
	ECHOES.MAC	90
	END.MAC	91
	FILCHK.MAC	91
	GEXIT.MAC	92
	GLOBAL.MAC	93
	INIT.MAC	94
	LGRAPH.FOR	95
	LINKVT.MAC	96
	LONGV.MAC	96
	LPEN1.MAC	98
	MINMAX.MAC	99
	MOVETO.MAC	100
	NEXTPT.MAC	101
	ONOFF.MAC	101
	OPEN.MAC	103
	PICSAV.MAC	104
	POINT.MAC	105
	RELDOT.MAC	106
	ROOM.MAC	107
	SIGN.MAC	108

Table of Contents (continued)

Page

SAVE.MAC 108
SCAN.MAC 110
SCBUF.MAC 112
SLEEP.MAC 113
SPACE.MAC 113
SWAPS.MAC 114
SWITCH.MAC 114
TTYIN.MAC 115
VECTOR.MAC 116
VTSCN.MAC 117
WHAT.MAC 120
WRTEXT.MAC 120
ZERO.MAC 121

XIX. Programs for Communicating with the CYBER System 122

CSN SUBROUTINE/FUNCTION REFERENCE TABLE

SUBROUTINE	PRIMARY REF PAGE	OTHER REFS.
APNT	13	15, 17, 19, 31
BLNKDF	9	
CLOSSF	7	8, 9, 18, 19, 20, 22, 31, 36, 38, 39, 75
DRAW	10	16, 18, 20, 22, 27, 31, 35, 36, 37, 38, 39, 41, 43, 63, 65, 75
ECHO	55	
FLASH	44	45
IADDR	50	51
INITDF	6	6, 7, 9, 13, 15, 16, 17, 18, 19, 20, 22, 25, 26, 27, 29, 31, 33, 35, 37, 38, 41, 43, 44, 47, 48, 63, 65, 67
INITSF	6	6, 7, 8, 18, 19, 20, 22, 31, 33, 38, 73, 75, 76
IPUT	52	
ISPACE	41	
ISWTCH	48	49
ITTYIN	52	56
IZERO	53	
JSIGN	50	
LNGRPH	45	47, 48
LOOK	51	52, 54
LPEN	25	26, 27, 29, 44
LVECT	10	15, 19, 20, 22, 25, 26
LZERO	53	
MOVE	13	16, 17, 19, 21, 25, 26, 27, 28, 29, 31, 35, 36, 37, 38, 39, 41, 42, 43, 44, 63, 65, 67, 68, 73, 76
NEXT	41	42, 43
NOECHO	55	56
OFFSUB	21	22, 23
ONSUB	21	23
OPENSF	7	8, 18, 19, 20, 22, 31, 35, 38, 73, 75
RADAR	30	31
RECORD	36	37, 39
RELPNT	14	
REMOVE	10	
REPLAY	36	39, 40
RESET	39	40
RESTOR	34	36
RZERO	53	
SAVEDF	34	35, 36
SNOOZE	55	
START	9	15, 16, 17, 19, 22, 25, 26, 27, 28, 29, 30, 31, 35, 36, 37, 39, 40, 41, 43, 44, 47, 63, 65, 67, 68

RK2:CREF.DAT 05-OCT-76

SUBROUTINE	PRIMARY REF. PAGE	OTHER REFS.
STOP	9	40
SUBJMP	17	18, 19, 20, 21, 22, 31, 38, 39
TEXT	14	15, 16, 17, 27, 28, 29, 44
TTOUT	54	55
UNBLNK	9	10
VECT	12	
WRITE	14	

I. Introduction

The CSN Graphics subroutine package is a collection of macro-assembly language routines which supports the graphics display system on the DEC GT-40/42/44 series of graphics computers. By using the CSN graphics package, the FORTRAN or assembly language programmer is provided with the basic tools with which he or she can create and manipulate display and subroutine files which are the basis of interactive graphics.

It is assumed that the reader of this manual has a basic knowledge of the FORTRAN programming language as well as the RT-11 operating system as all examples are given in RT-11 FORTRAN. The later sections of the manual are included for the use of more advanced assembly language programmers and those users who will be installing the CSN Graphics System on their computers. The basic system needed to run the CSN Graphics System is:

PDP-11 CPU with 8K words of memory

VT-11 and VR17 or VR14 display

Any mass storage device

The RT-11 operating system

Finally, as this manual is designed to teach the user how to use the CSN subroutine system, it is strongly recommended that the user actually run the sample programs given in the manual - they have all been tested and run properly underversion 2B of RT-11 and version 1B of RT-11 FORTRAN.

All inquiries and comments should be sent to:

Arthur Karshmer
Center for Systems Neuroscience
Graduate Research Center
University of Massachusetts
Amherst, Massachusetts 01002

2

CSN GRAPHICS

II. The GT-44

A. The Hardware

The CSN GT-44 computer is a PDP-11/40 central processor with the following peripheral devices:

- 28K words of memory
- RK11 Disk Controller with 1 IMS DMØ6 Disk Drive and 1 RKØ5 Disk Drive
- LA30 Decwriter
- VT11 Graphics Processor
- VR17 Graphics Display Unit
- DL11-E Communications adapter
- KW11-L Line Clock
- KE11-E Multiply/Divide Unit
- KE11-F Floating Point Unit

with all the peripheral devices communicating with each other via the unibus.

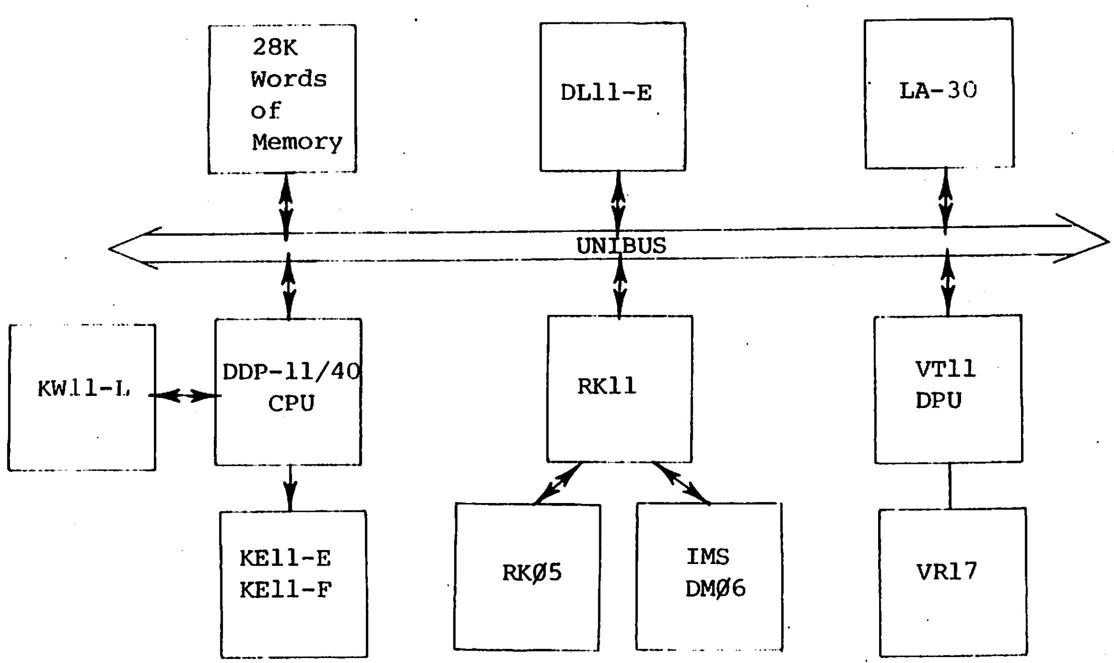


FIGURE 1

3

The PDP-11 unibus structure allows the 11/40 CPU and VT11 DPU to access the same memory locations, which permits the generation of dynamic graphics programs. More specifically, graphics programs are generated and executed as follows:

(1) The 11/40 CPU inserts graphics instructions into contiguous memory locations.

(2) The CPU loads the starting address of the graphics instructions into the VT11 DPU program counter and starts its execution.

(3) The VT11 DPU executes the instructions loaded into memory by the CPU.

B. The Software

The area of memory which is defined by the user and shared by the CPU and DPU is called the "DISPLAY File" and is the basic unit of graphic programming - i.e., all graphics routines manipulate the display file to effect changes to a graphics program. The most basic elements of a display file are:

(1) Primitive Display instructions, e.g., point mode, character mode, vector mode and display jump.

(2) A jump to the beginning of the display file to refresh the picture on the screen.

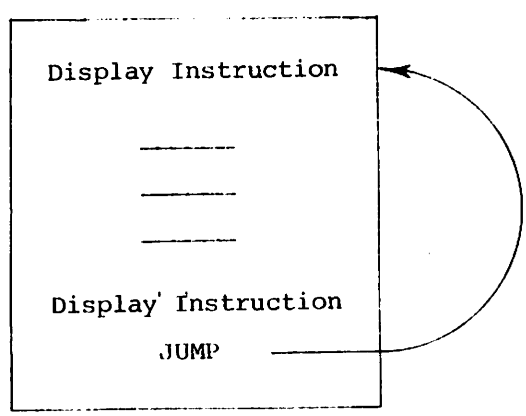


FIGURE 2

Display File

4

This structure allows the DPU to execute graphics instructions and then return to the top of the display file and execute the instructions again. Therefore, to add a display instruction to the display file, the jump instruction must be replaced by the new display instruction and a new jump instruction inserted into the display file, which forces the DPU to re-execute the display commands in the display file and therefore keep a steady picture on the screen.

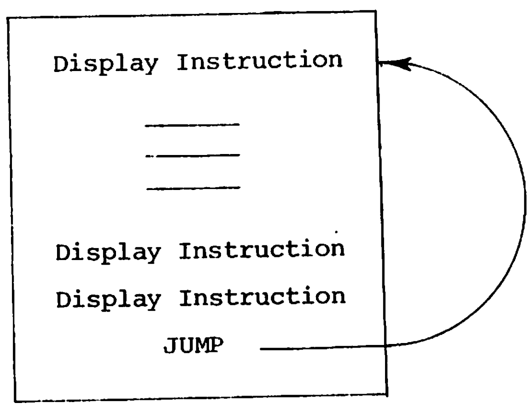


FIGURE 3

Updated Display File

In addition to display files, the CSN Graphics System allows the user to define any number of "SUBROUTINE Files" in which the user can define and use up to 256 named subroutines per subroutine file. For example, the user could define a subroutine which draws a box and then jumps to the subroutine whenever a box needs to be drawn on the screen.

The CSN Graphics System allows the programmer to define as many display and subroutine files as needed, with the only restrictions being:

(1) No more than two display files may be active at any given time.

(2) Each subroutine file can contain no more than 256 named subroutines.

5 5

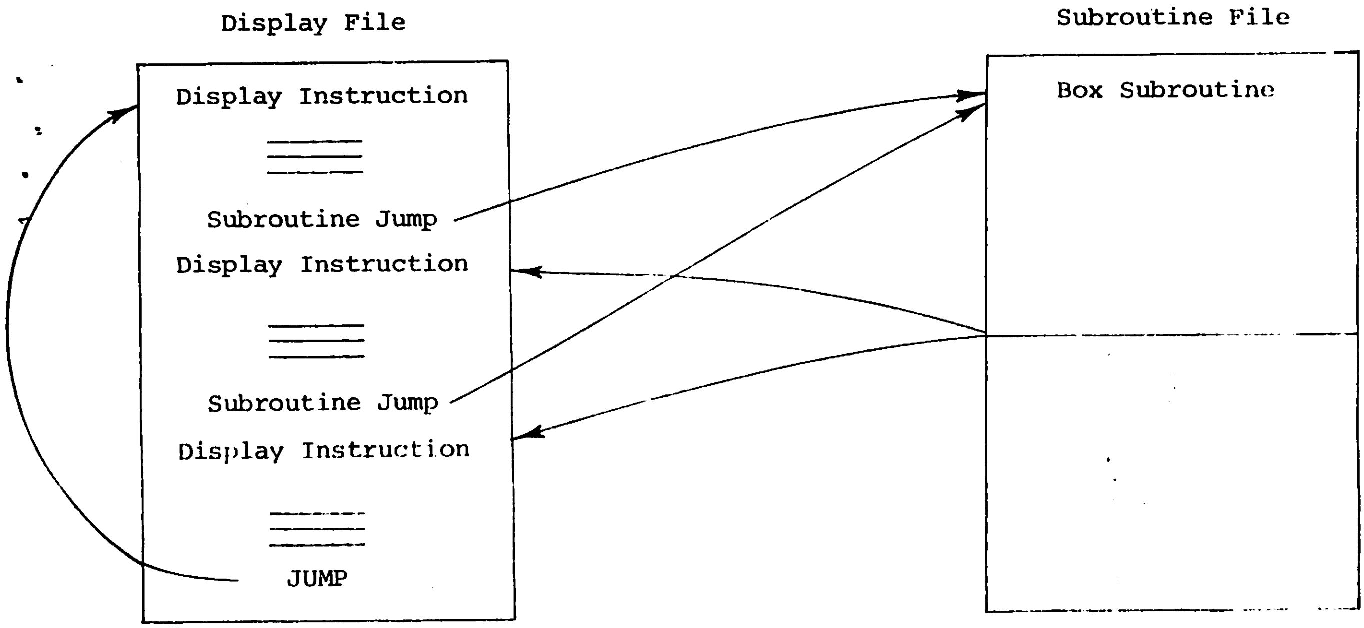


FIGURE 4

Display and Subroutine File Interaction

The number of subroutine calls and depth of subroutine nesting is a function of available memory, and there is no restriction against one subroutine calling other graphics subroutine either residing in its own subroutine file or other subroutine files.

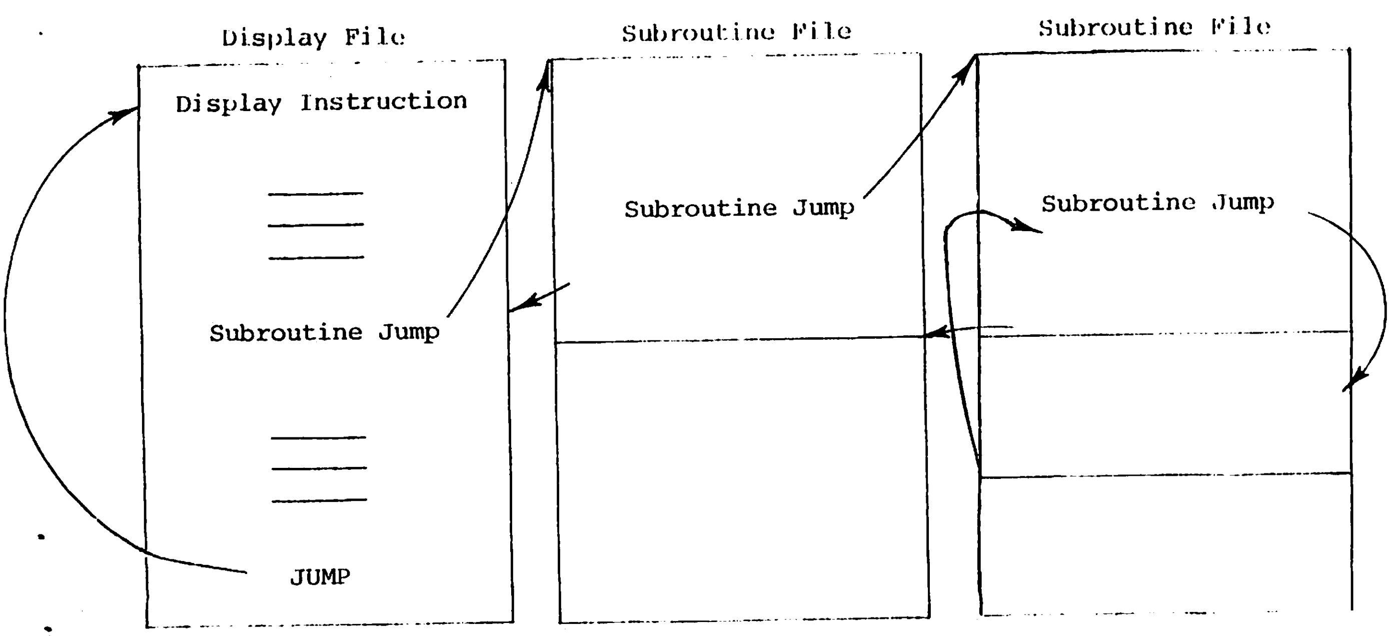


FIGURE 5

A More Complex Subroutine Structure

6

To aid the user in creating display and subroutine files, all CSN Graphics subroutines operate on both display and subroutine files - e.g., the same long vector subroutine call could be used in a display or subroutine file.

III. CSN Graphics Subroutines

The following descriptive material details the use of the CSN Graphics subroutines by FORTRAN programs on the GT-44 system. Calling these subroutines from assembly language programs is discussed in a later section of this paper.

A. Creating Display and Subroutine Files

Display and subroutine files are contiguous areas of memory defined in the FORTRAN program by means of a dimension, integer or common statement, and should be integer type. For example,

```
DIMENSION IFILE(1ØØØ), JFILE(5ØØ)
```

would define a 1000 word area of memory named IFILE and a 500 word area of memory known as JFILE. At this point the user must notify the graphics system what type of file each area is to be.

For example, if the user was to use IFILE as a display file and JFILE as a subroutine file, the following subroutine call would be used:

```
CALL INITDF(IFILE, 1ØØØ)
CALL INITSF(JFILE, 5ØØ)
```

INITDF creates a display file in memory, while INITSF creates a subroutine file. The two arguments passed to the subroutines are the file name and length in words.

A contiguous area of memory could be used to achieve the same end, as follows:

```
DIMENSION IFILE(15ØØ)
CALL INITDF(IFILE, 1ØØØ)
CALL INITSF(IFILE(1ØØ1), 5ØØ)
```

7

or more simply:

```
DIMENSION IFILE(15ØØ)
EQUIVALENCE (JFILE, IFILE(1ØØ1))
CALL INITDF(IFILE, 1ØØØ)
CALL INITSF(JFILE, 5ØØ)
```

It is strongly suggested that the last technique be used if the user wishes to use the save and restore routines available in the CSN Graphics System.

Once initialized, the display and subroutine files are available for further use. If the user were to attempt to operate on a display or subroutine file which was not initialized, the message

```
**FATAL ERROR** DISPLAY/SUB FILE NOT KNOWN
```

would be typed and program execution would be terminated.

B. Creating a Named Subroutine

To open a new named subroutine in a given subroutine file or to extend the scope of an existing named subroutine the CSN Graphics user simply uses the OPENSF subroutine. Once opened, the user can put graphics instructions in the subroutine using any CSN Graphics subroutine. During the creation of a named subroutine, the user can continue to operate on other subroutines or display files.

In order to close a subroutine definition, the user calls the CLOSSF subroutine. The following FORTRAN code demonstrates the opening and closing of a subroutine named 'A'.

```
DIMENSION JFILE(1ØØ)
CALL INITSF(JFILE, 1ØØ)
CALL OPENSF(JFILE, 'A')

CALL CLOSSF(JFILE)
```

8

After subroutine 'A' has been closed, the user can open another subroutine in JFILE, i.e. -

```
CALL OPENSF(JFILE, 'C')

      ≡

CALL CLOSSF(JFILE)
```

Further, subroutine 'A' could now be re-opened as follows in order to expand upon the definition of the display subroutine.

```
CALL OPENSF(JFILE, 'A')

      ≡

CALL CLOSSF(JFILE)
```

It would not be legal, however, to have two subroutine definitions open in the same subroutine file at a given time. Therefore, the following sequence would cause an error condition.

```
DIMENSION JFILE(5ØØ)
CALL INITSF(JFILE, 5ØØ)
CALL OPENSF(JFILE, 'A')

      ≡

CALL OPENSF(JFILE, 'C')

      ≡
```

The following sequence would be quite proper, however:

```
DIMENSION JFILE(5ØØ), KFILE(5ØØ)
CALL INITSF(JFILE, 5ØØ)
CALL INITSF(KFILE, 5ØØ)
CALL OPENSF(JFILE, 'A')

      ≡

CALL OPENSF(KFILE, 'C')

      ≡
```

```
CALL CLOSSF(JFILE)

    ═══════

CALL CLOSSF(KFILE)
```

C. Starting and Stopping the Display Processor

The CSN Graphics System allows the user to have up to two display files actively displayed at any time, with an unlimited number of subroutine files active. To initialize display processor execution, the user would use the following sequence of commands:

```
DIMENSION IFILE(5ØØ)
CALL INITDF(IFILE, 5ØØ)

    ═══════

CALL START(IFILE)
```

At this point, the DPU is executing instructions in IFILE. It is not necessary to turn off the DPU to manipulate the display file, as the CSN Graphics System automatically turns the DPU on and off when necessary. If the user attempts to START more than two display files, the CSN Graphics System issues a warning message, and execution continues.

To stop the DPU, the user simply issues a CALL STOP command - e.g.,

```
CALL STOP(IFILE)
```

D. Blanking - Unblanking and Removing the Display File

Once a display file has been made active via a CALL START subroutine call, it may be turned off and turned on in a more rapid manner by way of the BLNKDF and UNBLNK subroutine calls. By use of these calls the display processor simply bypasses the named display file rather than stopping the display processor - e.g.,

```
CALL BLNKDF(IFILE)
```

To reactivate the display file, the user simply executes the following command:

CALL UNBLNK(IFILE)

If the user wishes to remove a display file from the active list, the following statement is executed:

CALL REMOVE(IFILE)

Since the CALL REMOVE statement actually removes the display file from the active list, the user would now be free to insert a new display file into the active list by using another CALL START subroutine call.

E. The Basic Graphics Operations

(1) The Vector or Line Mode

While it is possible for the user to create line segments by plotting a series of dots using point mode operations, the GT-44 allows lines to be drawn in a much simpler fashion. For example, if the beam was currently located at coordinates X=500, Y=600 and the user wished to draw a line to point X=750, Y=800, she could simply call the long vector subroutine as follows:

CALL LVECT(DELTA-X, DELTA-Y, INTEN, BLINK, LINE, LPEN, FILE,
[, DISPLAY NAME])

or

CALL DRAW(DELTA-X, DELTA-Y, FILE[, INTEN, BLINK, LINE, LPEN,
[, DISPLAY NAME]])

The first two parameters in the call list are known as the DELTA-X and DELTA-Y values. They simply tell the DPU how many units to move on the X and Y axes before connecting the old and new points with a line.

The parameter called 'INTEN' tells the DPU how bright the line is to be with the range being run Ø to 7.

The 'BLINK' parameter lets the user specify whether or not the line is to blink on the screen.

Ø - NO BLINK

1 - BLINK

The parameter called 'LINE' allows the user to specify what type of line is to be drawn by the DPU. The line types are as follows:

Ø - Solid line ________________________

1 - Long Dashed line — — — — — —

2 - Short Dashed line ----------------

3 - Dot Dashed line -.-.-.-.-.-.-.-.-.

If the user wishes to make a line segment light pen sensitive, she specifies this fact in the 'LPEN' parameter.

Ø - not sensitive

1 - sensitive

The parameter called 'FILE' is the name of either the display file or subroutine file into which the line segment commands are to be stored.

The final parameter in the list is called 'DISPLAY NAME' and is optional. It is only used when the user specifies that the line segment is light pen sensitive. The display name then becomes a name specified by the user as a unique name for the specific displayed item. The subject of light pen sensitivity and display names will be discussed in detail in a later section.

In the case of the CALL DRAW subroutine call, the only required parameters are DELTA-X, DELTA-Y and FILE. The default for the remaining parameters are:

INTEN = 2

BLINK = Ø

LINE = Ø

LPEN = Ø

DISPLAY-NAME = N/A (Not required when there is no light pen sensitivity.)

If the user wishes to reset the default parameters in the draw routine, she simply

12

calls it with all parameters specified. From that point on, the user-supplied parameters become the default parameters until changed again.

Getting back to the original problem - to draw the line segment from X=500, Y=600 to X=750, Y=800 with:

```
INTENSITY = 4
BLINK     = YES
LINE      = DOT DASHED
LPEN      = NONE
FILE      = IFILE
DISPLAY-NAME = NONE
```

We would simply:

```
CALL MOVE(5ØØ, 6ØØ, IFILE)
CALL LVECT(25Ø, 2ØØ, 4, 1, 3, Ø, IFILE)
                    or
CALL DRAW(25Ø, 2ØØ, IFILE, 4, 1, 3, Ø)
```

It turns out that the GT-44 has still another form of vector it can draw - the short vector. The main use of the short vector is that it can help save display file space because the DELTA-X and DELTA-Y values can be combined in a single data word. The allowable range of numbers acceptable is only from -64 to +63. To use the short vector, the user would call

```
CALL VECT(DELTA-X, DELTA-Y, INTEN, BLINK, LINE, LPEN, FILE
 [, DISPLAY NAME])
```

If the user calls this routine with DELTA-X or DELTA-Y value out of range, the long vector routine is automatically called.

(2) The Point Mode

The VR17 screen is composed of a grid of over one million addressable and intensifiable points. The coordinate system of the screen is best described as

 13

having 1024 points on the X axis for each of the 1024 points on the Y axis. The four corners of the screen are numbered as follows:

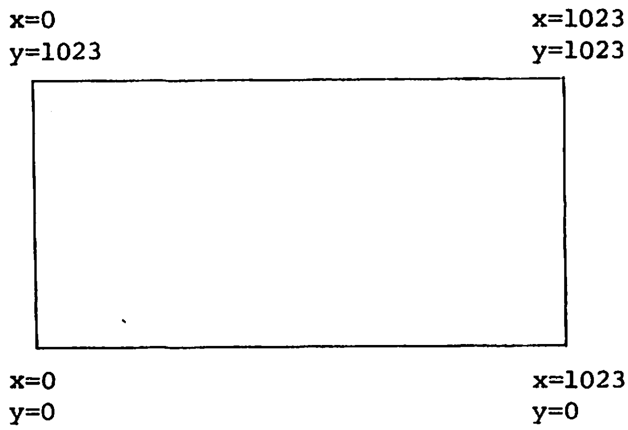


FIGURE 6

To move the electron gun to any point on the VR17 axis system, the user can use one of two commands:

```
CALL APNT(X, Y, INTEN, BLINK, LPEN, FILE[, DISPLAY-NAME])
```

(See section XVII for a list of parameter values and meanings.)

or

```
CALL MOVE(X, Y, FILE[, INTEN, BLINK, LPEN[, DISPLAY-NAME]])
```

For example, to position the beam, unintensified, at coordinate X=600, Y=500 the following two program sequences could be used:

```
DIMENSION IFILE(5ØØ)
CALL INITDF(IFILE, 5ØØ)
CALL APNT(600, 5ØØ, -1, Ø, Ø, IFILE)
```

or

```
DIMENSION IFILE(5ØØ)
CALL INITDF(IFILE, 5ØØ)
CALL MOVE(6ØØ, 5ØØ, IFILE)
```

The major difference between the two commands is that the APNT subroutine requires more parameters to be passed than does the MOVE subroutine which uses a set of pre-

14

defined default values. (The default values can be changed by specifying the optional arguments in the list.) The MOVE subroutine does require more words of memory than does the APNT subroutine, as it must save certain default parameters and then call the APNT subroutine. The starting default values for the MOVE subroutine are:

INTENSITY	--	NOT INTENSIFIED
BLINK	--	NO BLINK
LPEN	--	NO LPEN
DISPLAY-NAME	--	NONE

Both the APNT and MOVE subroutines position the beam at the absolute points specified by the user. It is possible, however, to move the beam relative to its current location without actually knowing its X and Y coordinates. To do this, the user would call a relative point routine:

```
CALL RELPNT(DELTA-X, DELTA-Y, INTEN, BLINK, LPEN, FILE
[, DISPLAY-NAME])
```

The range of the DELTA-X and DELTA-Y values is -64 to +63.

(3) The Text Mode

The GT-44 has the ability to write textual information on the screen through the use of a hardware character generator. To allow the user to exploit this feature, the CSN Graphics System has two subroutine calls which allow the user to insert textual data into a display or subroutine file. The two subroutine calls are:

```
CALL TEXT(INTEN, BLINK, LPEN, CASE, STRING, FILE[, DISPLAY-NAME])
```

or

```
CALL WRITE(STRING, FILE[, INTEN, BLINK, LPEN, CASE[, DISPLAY-NAME]])
```

All parameters listed are the same as previously described except for string and case.

15

The STRING parameter can be either a character string enclosed in single quotation marks or a variable name. In either case, the last character in the string must be a semicolon, e.g., 'HI THERE FOLKS;'. The semicolon will not appear on the screen, but must be included in the string. If not included, the results will be unpredictable with a strong possibility of crashing the entire system.

The CASE parameter allows the user to write in either upper or lower case characters.

Ø = upper case

1 = lower case

F. Sample Programs Using the Basic Graphics Operations

(1) The first program positions the unintensified beam at position 5ØØ, 5ØØ - then draws a box 200 units square. The box is made of solid lines, no blinking, intensity 3, and is not light pen sensitive. Finally, the program moves the unintensified beam to X=100, Y=200 and writes 'Hi there folks' in lower case, intensity 2, blinking letters with no light pen sensitivity.

```
DIMENSION IBUF(1ØØ) !**CREATE DISPLAY FILE
CALL INITDF(IBUF, 1ØØ) !**INITIALIZE DISPLAY FILE
CALL START(IBUF) !**TURN ON THE SCREEN
CALL APNT(5ØØ, 5ØØ, -1, Ø, Ø, IBUF) !**MOVE BEAM
CALL LVECT(1ØØ, Ø, 3, Ø, Ø, Ø, IBUF) !**DRAW LINES
CALL LVECT(Ø, 1ØØ, 3, Ø, Ø, Ø, IBUF)
CALL LVECT(-1ØØ, Ø, 3, Ø, Ø, Ø, IBUF)
CALL LVECT(Ø, -1ØØ, 3, Ø, Ø, Ø, IBUF)
CALL APNT(1ØØ, 2ØØ, -1, Ø, Ø, IBUF)
CALL TEXT(2, 1, Ø, 1, 'HI THERE FOLKS;', IBUF)
PAUSE !**TO HOLD PICTURE ON SCREEN
END
```

or

16

```
      DIMENSION IBUF(1ØØ) !**CREATE DISPLAY FILE
      CALL INITDF(IBUF, 1ØØ) !**INITIALIZE DISPLAY FILE
      CALL START(IBUF)
      CALL MOVE(5ØØ, 5ØØ, IBUF)
      CALL DRAW(1ØØ, Ø, IBUF, 3, Ø, Ø, Ø) !**DRAW LINE AND SET
      CALL DRAW(Ø, 1ØØ, IBUF)
      CALL DRAW(-1ØØ, Ø, IBUF)
      CALL DRAW(Ø, -1ØØ, IBUF)
      CALL MOVE(1ØØ, 2ØØ, IBUF)
      CALL TEXT(2, 1, Ø, 1, 'HI THERE FOLKS;', IBUF)
      PAUSE
      END
```

(2) The second sample program writes the message 'CAN YOU SEE ME NOW' in upper case letters at the seven different intensities at seven different places on the screen, the first occurrence of the message will be at position Ø, Ø with each of the next occurrences moving up and to the right by 30 screen units.

```
      INTEGER DFILE(3ØØ)
      CALL INITDF(DFILE, 3ØØ)
      CALL START(DFILE)
      DO 1Ø I = Ø, 6
      CALL APNT(I*3Ø, I*3Ø, -1, Ø, Ø, DFILE)
1Ø    CALL TEXT(I, Ø, Ø, Ø, 'CAN YOU SEE ME NOW;', DFILE)
      PAUSE
      END
```

or

```
      DIMENSION IFILE(3ØØ)
      CALL INITDF(IFILE, 3ØØ)
      CALL START(IFILE)
```

17

```
          DO 1Ø I = Ø, 6
          CALL MOVE(I*3Ø, I*3Ø, IFILE)
1Ø        CALL TEXT(I, Ø, Ø, Ø, 'CAN YOU SEE ME NOW;', IFILE)
          PAUSE
          END
```

(3) Our third and final example of using the basic CSN Graphics functions shows a program which will draw a line diagonally across the screen using intensified points. The intensity is 2, no blinking and no light pen sensitivity.

```
          DIMENSION IBUF(3ØØØ)
          CALL INITDF(IBUF, 3ØØØ)
          CALL START(IBUF)
          DO 1Ø I = Ø, 1Ø23
1Ø        CALL APNT(I, I, 2, Ø, Ø, IBUF)
          PAUSE
          END
```

or

We could do the same thing but only putting in a point every other time as follows:

```
          DIMENSION IBUF(3ØØØ)
          CALL INITDF(IBUF, 3ØØØ)
          CALL START(IBUF)
          DO 1Ø I = Ø, 1Ø23, 2
          CALL APNT(I, I, 2, Ø, Ø, IBUF)
          PAUSE
          END
```

G. Connecting Subroutines to Display Files and Connecting Subroutines to Other Subroutines

Once the user has created a named subroutine in a subroutine file, the next step is to connect that subroutine to a display or subroutine file. This is accomplished

18

by the following CSN Graphics call:

```
CALL SUBJMP(FILE, SUB-FILE, SUBROUTINE)
```

The FILE parameter is simply the name of the source display or subroutine file.

The SUB-FILE parameter is the name of the subroutine file in which the named subroutine resides.

The SUBROUTINE parameter is the name of the previously created and named graphics subroutine.

To illustrate the use of the SUBJMP command we shall write a program that first creates a subroutine named 1, which draws a 1ØØ unit square box. Next through the use of MOVE and SUBJMP calls we shall draw the box at fifty different places on the screen starting at position Ø, Ø - incrementing the X coordinate by 15 units and the Y position by 10 units.

The boxes will be:

```
INTENSITY = 2
BLINK     = NONE
LINE      = SOLID
LPEN      = NONE
```

```
DIMENSION IBUF(1ØØØ), JBUF(3Ø) !**CREATE DISPLAY AND SUBROUTINE FILES
CALL INITDF(IBUF, 1ØØØ)
CALL INITSF(JBUF, 3Ø) !**INITIALIZE SUBROUTINE FILE
CALL OPENSF(JBUF, 1) !**OPEN SUBROUTINE #1
CALL DRAW(1ØØ, Ø, JBUF) !**DRAW THE BOY
CALL DRAW(Ø, 1ØØ, JBUF)
CALL DRAW(-1ØØ, 0, JBUF)
CALL DRAW(Ø, -1ØØ, JBUF)
CALL CLOSSF(JBUF) !**CLOSE SUBROUTINE #1
```

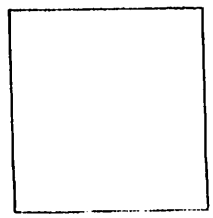
```
          DO 1ØØ I = 1, 5Ø
          CALL MOVE((I-1)*15, (I-1)*1Ø, IBUF) !**MOVE TO NEXT POINT
1ØØ       CALL SUBJMP(IBUF, JBUF, 1) !**JUMP TO BOX SUBROUTINE

          CALL START(IBUF) !**TURN ON DISPLAY
          PAUSE
          END
```

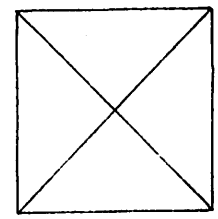
or

```
          DIMENSION IBUF(1ØØØ), JBUF(3Ø) !**CREATE DISPLAY AND SUBROUTINE FILES
          CALL INITDF(IBUF, 1ØØØ) !**INITIALIZE DISPLAY FILE
          CALL INITSF(JBUF, 3Ø) !**INITIALIZE SUBROUTINE FILE

          CALL OPENSF(JBUF, 1) !**OPEN SUBROUTINE #1
          CALL LVECT(1ØØ, Ø, 2, Ø, Ø, Ø, JBUF) !**DRAW BOX
          CALL LVECT(Ø, 1ØØ, 2, Ø, Ø, Ø, JBUF)
          CALL LVECT(-1ØØ, Ø, 2, Ø, Ø, Ø, JBUF)
          CALL LVECT(Ø, -1ØØ, 2, Ø, Ø, Ø, JBUF)
          CALL CLOSSF(JBUF) !**CLOSE SUBROUTINE #1
          DO 1ØØ I = 1, 5Ø
          CALL APNT((I-1)*5Ø, (I-1)*5Ø, -1, Ø, Ø, IBUF) !**MOVE BEAM
1ØØ       CALL SUBJMP(IBUF, JBUF, 1) !**JUMP TO SUBROUTINE #1

          CALL START(IBUF)
          PAUSE
          END
```

Our next example of using the SUBJMP subroutine is a bit more complex than the first, but will serve to point out the power of the graphic subroutine feature of the CSN Graphics System. In this example, we wish to be able to draw the following objects on the screen in several different places.

20





First, we will define a subroutine to draw the empty 1ØØ unit square box. Second, we will define another graphic subroutine which will access the first box routine and then connect the corners. We will then have two subroutines with the second being defined in terms of the first.

Finally, we will display the two objects on the screen at positions 1ØØ, 14Ø, and 5ØØ, 5ØØ respectively.

```
            DIMENSION IBUF(1ØØ), JBUF(1ØØ) !**CREATE FILES
            CALL INITDF(IBUF, 1ØØ)
            CALL INITSF(JBUF, 1ØØ)
C----------FIRST DEFINE THE EMPTY BOX
            CALL OPENSF(JBUF, 1)
            CALL DRAW(1ØØ, Ø, JBUF)
            CALL DRAW(Ø, 1ØØ, JBUF)
            CALL DRAW(-1ØØ, Ø, JBUF)
            CALL DRAW(Ø, -1ØØ, JBUF)
            CALL CLOSSF(JBUF)
C----------NOW WE CALL THE BOX SUBROUTINE AND PUT IN THE X
            CALL OPENSF(JBUF, 2)
            CALL SUBJMP(JBUF, JBUF, 1)
            CALL DRAW(1ØØ, 1ØØ, JBUF)
            CALL LVECT(-1ØØ, Ø, -1, Ø, Ø, Ø, JBUF) !**A RELATIVE MOVE
            CALL DRAW(1ØØ, -1ØØ, JBUF)
            CALL CLOSSF(JBUF)
```

21

```
C----------NOW WE PUT THE BOXES ON THE SCREEN
           CALL MOVE(1ØØ, 14Ø, IBUF)
           CALL SUBJMP(IBUF, JBUF, 1)
           CALL MOVE(5ØØ, 5ØØ, IBUF)
           CALL SUBJMP(IBUF, JBUF, 2)
           CALL START(IBUF)
           PAUSE
           END
```

H. Disconnecting and Connecting Subroutine Linking from Display and Subroutine Files

Once a link has been made between a display file and a named subroutine, or a subroutine and another named subroutine, the CSN Graphics user has the ability to selectively break and re-establish the linkage, thereby dynamically changing the display. The CSN Graphics call to accomplish this is:

CALL OFFSUB(FILE, SUBFILE, NAME, SUB-PICTURE1[, SUB-PICTURE2])

CALL ONSUB(FILE, SUBFILE, NAME, SUB-PICTURE1[, SUB-PICTURE2])

Where:

FILE is either a display or subroutine file in which the subroutine linkage resides.

SUBFILE NAME is the name of the target subroutine.

SUB-PICTURE1 is the name of the target subroutine.

SUB-PICTURE2 is the name of the source sub-picture if FILE is a subfile name.

For example, assume the user wanted to alternate the display of boxes and boxes with x's in the middle on the screen (the previous example shows this example in a static sense). In the current example, we will display the plain boxes on the left side of the screen, pause, then display the boxes with x's on the right side of the screen, pause, show the plain boxes, etc.

22

```
          DIMENSION IBUF(5ØØ), JBUF(1ØØ) !**CREATE FILES
          CALL INITDF(IBUF, 5ØØ) !**INITIALIZE FILES
          CALL INITSF(JBUF, 1ØØ)
C----------CREATE BOX SUBROUTINE
          CALL OPENSF(JBUF, 1)
          CALL DRAW(1ØØ, Ø, JBUF)
          CALL DRAW(Ø, 1ØØ, JBUF)
          CALL DRAW(-1ØØ, Ø, JBUF)
          CALL DRAW(Ø, -1ØØ, JBUF)
          CLASS CLOSSF(JBUF)
C----------CREATE BOX WITH X SUBROUTINE
          CALL OPENSF(JBUF, 2)
          CALL SUBJMP(JBUF, JBUF, 1)
          CALL DRAW(1ØØ, 1ØØ, JBUF)
          CALL LVECT(-1ØØ, Ø, -1, Ø, Ø, Ø, JBUF)
          CALL DRAW(1ØØ, -1ØØ, JBUF)
          CALL CLOSSF(JBUF)
C----------PUT EMPTY BOXES IN DISPLAY FILE
          DO 1ØØ I = Ø, 4ØØ, 1ØØ
          CALL MOVE(I, I, IBUF)
1ØØ       CALL SUBJMP(IBUF, JBUF, 1)
C----------PUT X BOXES IN DISPLAY FILE
          DO 2ØØ I = Ø, 4ØØ, 1ØØ
          CALL MOVE(I + 5ØØ, I, IBUF)
2ØØ       CALL SUBJMP(IBUF, JBUF, 2)
C----------DISCONNECT X BOXES
          CALL OFFSUB(IBUF, JBUF, 2)
C----------TURN ON THE DISPLAY
          CALL START(IBUF)
          PAUSE
```

```
C----------TURN OFF BOXES, TURN ON X BOXES
300         CALL OFFSUB(IBUF, JBUF, 1)
            CALL ONSUB(IBUF, JBUF, 2)
            PAUSE
C----------TURN OFF X BOXES, TURN ON BOXES
            CALL OFFSUB(IBUF, JBUF, 2)
            CALL ONSUB(IBUF, JBUF, 1)
            PAUSE
            GO TO 300
            END
```

IV. The Light Pen and How to Use It

One of the most powerful features of the CSN GT-44 Graphics System is the interactive light pen which can be used as an input device. By declaring certain areas of a display to be light pen sensitive, the user can interact with a graphics program.

At this point, a brief explanation of how the light pen operates will aid the user in effective use of this device.

As described earlier, the GT-44 system draws points, vectors and characters according to instructions which are contained in the display file. Through the use of its display program counter (DPC), the graphics processor executes the instructions in the display file in much the same way as does the central processor. What this means is that at any given time, the DPC knows exactly where and what it is drawing and can report this data to the executing program. As soon as light enters the photo diode on the top of the light pen, an interrupt is raised. In response to this interrupt, the light pen handler, which is software to handle light pen interrupts, examines the contents of the DPC and the instruction it is executing. If the user had specified light pen sensitivity for the current instruction through the use of the LPEN parameter, the light pen handler will return certain information

24

to the running program. If the interrupt occurred while executing a non-sensitive instruction, no action is taken.

As well as specifying light pen sensitivity, the CSN Graphics System allows the user to give each sensitive item a unique name (number) which can range from 0 to 32767. When a light pen sensitive item is hit by the light pen, the handler can return this pre-defined 'display name'.

Since the concept of "interrupts" is not applicable to FORTRAN programming, the CSN Graphics support system handles the interrupts and passes the relevant data back to the executing FORTRAN program. To do this job, it is necessary to allocate a seven-word buffer for use by the CSN light pen handler. This is accomplished through the use of the following subroutine call:

```
CALL LPEN(BUFF-NAME)
```

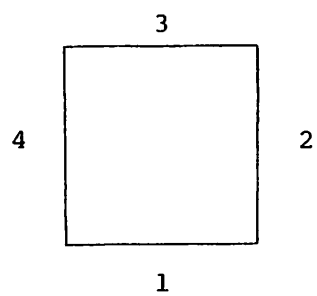
The seven-word buffer is used to pass the following data to the FORTRAN program.

Word #	Information
1	LIGHT PEN FLAG - 1 IF HIT
2	DISPLAY NAME
3	UNUSED
4	DISPLAY PROGRAM COUNTER
5	DISPLAY STATUS REGISTER
6	X POSITION OF HIT
7	Y POSITION OF HIT

While the call to the LPEN subroutine does set up the linkage between the light pen handler and the FORTRAN program, it does not actually enable the light pen. To enable the light pen, the user simply moves zero to word #1 of the seven-word buffer. When a light pen hit comes in, word #1 is then set to a 1 and all the light pen hits are disabled until word #1 is again cleared by the FORTRAN program. In this way, the CSN Graphics System protects all pertinent light pen data until the executing program re-enables the light pen handler.

The following examples should further help the CSN Graphics user understand the basic use of the light pen.

In our first example we will draw a square 200 raster or screen unit square with its lower lefthand corner located at position 5ØØ, 5ØØ on the screen. The four sides of the square will be light pen sensitive with display names of 1, 2, 3 and 4, respectively, starting with the bottom line as #1 and moving in a counterclockwise fashion. The lines will be intensity 2, solid and non-blinking.



As soon as any of the four sides is touched by the light pen, the program will print out the number of the side hit and the actual X and Y coordinates of the light pen hit. The program will then loop back and wait for another hit.

```
        DIMENSION IFILE(1ØØ), LPBUFF(7)
        CALL INITDF(IFILE, 1ØØ)
        CALL LPEN(LPBUFF) !**CREATE LPEN LINKAGE
        CALL MOVE(5ØØ, 5ØØ, IFILE)
        CALL LVECT(2ØØ, Ø, 2, Ø, Ø, 1, IFILE, 1) !**SIDE #1
        CALL LVECT(Ø, 2ØØ, 2, Ø, Ø, 1, IFILE, 2) !**SIDE #2
        CALL LVECT(-2ØØ, Ø, 2, Ø, Ø, 1, IFILE, 3) !**SIDE #3
        CALL LVECT(Ø, -2ØØ, 2, Ø, Ø, 1, IFILE, 4) !**SIDE #4
        CALL START(IFILE) !**TURN ON DISPLAY
1ØØ     LPBUFF(1) = Ø !**ENABLE LIGHT PEN
2ØØ     IF(LPBUFF(1).EQ.Ø) GO TO 2ØØ !**WAIT FOR HIT
        TYPE 3ØØ, LPBUFF(2), LPBUFF(6), LPBUFF(7)
```

 26

```
300       FORMAT(1X, 'SIDE=', I6, 'X=', I6, 'Y=', I6)
          GO TO 100 !**BACK FOR ANOTHER HIT
          END
```

In our second example of light pen sensitive programs, we will draw 3 squares which are 2ØØ raster units on each side. Each of the 3 squares will have a unique display name and will be drawn at intensity 3 with dot-dashed lines which blink. As soon as a square is touched by the light pen, the program will report which one was touched.

```
          DIMENSION IB(2ØØ), LPBUF(7)
          CALL INITDF(IB, 2ØØ)

          DO 1ØØ I = 1, 3
          CALL MOVE(I*2ØØ, I*2ØØ, IB)
          CALL LVECT(2ØØ, Ø, 3, 1, 3, 1, IB, I)
          CALL LVECT(Ø, 2ØØ, 3, 1, 3, 1, IB, I)
          CALL LVECT(-2ØØ, Ø, 3, 1, 3, 1, IB, I)
1ØØ       CALL LVECT(Ø, -2ØØ, 3, 1, 3, 1, IB, I)

          CALL START(IB)
          CALL LPEN(LPBUF)
2ØØ       LPBUF(1) = Ø
3ØØ       IF(LPBUF(1).EQ.Ø) GO TO 3ØØ
          TYPE 4ØØ, LPBUF(2)
4ØØ       FORMAT(1X, 'YOU TOUCHED BOX #', I3)
          GO TO 2ØØ
          END
```

or

```
          DIMENSION IB(2ØØ, LPBUF(7)
          CALL INITDF(IB, 2ØØ)
```

27

```
          DO 1ØØ I = 1, 3
          CALL MOVE(I*2ØØ, I*2ØØ, IB)
          CALL DRAW(2ØØ, Ø, IB, 3, 1, 3, 1, I)
          CALL DRAW(Ø, 2ØØ, IB)
          CALL DRAW(-2ØØ, Ø, IB)
1ØØ       CALL DRAW(Ø, -2ØØ, IB)

          CALL START(IB)
          CALL LPEN(LPBUF)
2ØØ       LPBUF(1) = Ø
3ØØ       IF(LPBUF(1).EQ.Ø) GO TO 3ØØ
          TYPE 4ØØ, LPBUF(2)
4ØØ       FORMAT(1X, 'YOU TOUCHED BOX #', I3)
          GO TO 2ØØ
          END
```

In our third and final example of using the light pen, before turning to tracking, we will put the digits from Ø to 9 on the screen with the #0 on the top and the number 9 on the bottom. The digits will be displayed at intensity 4 with no blink. At the top righthand corner of the screen we will put the word 'exit' in lower case. As each digit is hit with the light pen the program will report which number was touched. If the word 'exit' is touched, the program will exit to RT-11 monitor.

```
          DIMENSION IB(2ØØ), LPBUF(7) !**CREATE DISPLAY FILE AND L.P. BUFFER
          CALL INITDF(IB, 2ØØ)
          CALL LPEN(LPBUF) !**SET UP LIGHT PEN HANDLER
          CALL MOVE(5ØØ, 1ØØØ, IB)
          CALL TEXT(4, Ø, 1, Ø, 'Ø;', IB, Ø) !**PUT NUMBERS ON SCREEN
          CALL MOVE(5ØØ, 9ØØ, IB)
          CALL TEXT(4, Ø, 1, Ø, '1;', IB, 1)
          CALL MOVE(5ØØ, 8ØØ, IB)
```

28

```
      CALL TEXT(4, Ø, 1, Ø, '2;', IB, 2)
      CALL MOVE(5ØØ, 7ØØ, IB)
      CALL TEXT(4, Ø, Ø, 1, '3;', IB, 3)
      CALL MOVE(5ØØ, 6ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '4;', IB, 4)
      CALL MOVE(5ØØ, 5ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '5;', IB, 5)
      CALL MOVE(5ØØ, 4ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '6;', IB, 6)
      CALL MOVE(5ØØ, 3ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '7;', IB, 7)
      CALL MOVE(5ØØ, 2ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '8;', IB, 8)
      CALL MOVE(5ØØ, 1ØØ, IB)
      CALL TEXT(4, Ø, 1, Ø, '9;', IB, 9)
      CALL MOVE(9ØØ, 1ØØØ, IB)
      CALL TEXT(4, Ø, 1, 1, 'EXIT;', IB, 99)
      CALL START(IB) !**TURN ON DISPLAY
1ØØ   LPBUF(1) = Ø !**ENABLE LIGHT PEN
2ØØ   IF(LPBUF(0).EQ.Ø) GO TO 2ØØ !**WAIT FOR HIT
      IF(LPBUF(2).EQ.99) CALL EXIT
      TYPE 3ØØ, LPBUF(2)
3ØØ   FORMAT(1X, 'NUMBER', I3)
      GO TO 1ØØ
      END
```

or

```
      DIMENSION IB(2ØØ), LPBUF(7), NUMS(1Ø)
      DATA NUMS/ 'Ø;', '1;', '2;', '3;', '4;', '5;',
     1      '6;', '7;', '8;', '9;'/ !**SET UP TABLE OF NUMBERS
```

```
      CALL INITDF(IB, 2ØØ)
      DO 1ØØ I = 1, 1Ø !**INDEX THROUGH TABLE
      CALL MOVE(5ØØ, 1ØØØ - ((I - 1)*1ØØ), IB)
1ØØ   CALL TEXT(4, Ø, 1, Ø, NUMS(I), IB, I - 1) !**DISPLAY NUMBER

      CALL MOVE(9ØØ, 1ØØØ, IB)
      CALL TEXT(4, Ø, 1, 1, 'EXIT;', IB, 99)

      CALL LPEN(LPBUF) !**SET UP LIGHT PEN HANDLER
      CALL START(IB)
2ØØ   LPBUF(1) = Ø !**ENABLE LIGHT PEN
3ØØ   IF(LPBUF(1).EQ.Ø) GO TO 3ØØ !**WAIT FOR HIT
      IF(LPBUF(2).EQ.99) CALL EXIT
      TYPE 4ØØ, LPBUF(2)
4ØØ   FORMAT(1X, 'NUMBER', I3)
      GO TO 2ØØ
      END
```

or

```
      DIMENSION IB(2ØØ), LPBUF(7)
      LOGICAL*1 NUM(3), CLN
      DATA CLN/';'/
      CALL INITDF(IB, 2ØØ)
      DO 1ØØ I = Ø, 9
      CALL MOVE(5ØØ, 1ØØØ - (I*1ØØ), IB)
      ENCODE(2, 1Ø5, NUM) I
1Ø5   FORMAT(I2)
1ØØ   CALL TEXT(4, Ø, 1, Ø, NUM, IB, I)
      CALL MOVE(9ØØ, 1ØØØ, IB)
      CALL TEXT(4, Ø, 1, 1, 'EXIT;', IB, 99)
      CALL LPEN(LPBUF)
```

30

```
      CALL START(IB)
200   LPBUF(1) = Ø
300   IF(LPBUF(1).EQ.Ø) GO TO 300
      IF(LPBUF(2).EQ.99) CALL EXIT
      TYPE 400, LPBUF(2)
400   FORMAT(1X, 'NUMBER', I3)
      GO TO 200
      END
```

It should be pointed out that these three cases of the last example point out three different solutions to the same problem. The first case is an attempt to solve the problem in a 'brute force' manner while the next two cases use a more sophisticated approach. The third case in particular should be of interest to the more advanced FORTRAN programmer.

V. Light Pen Tracking

It is often convenient for the graphics programmer to have the ability to place objects on the screen by using the light pen. This general type of function is commonly known as "tracking" and is usually carried out through the use of a diamond shape tracking object which can be moved around the screen with the light pen. As soon as the user is satisfied with the position of the tracking object, he uses its last X and Y coordinates as the coordinates of the object to be drawn on the screen. The CSN Graphics System does not employ this traditional tracking system - instead we have developed a tracking routine which searches the screen to find the light pen and when it does, it reports the coordinates to the user. By calling the subroutine with the proper arguments, the user can specify exactly which portion of the screen is to be searched. The general order of the subroutine call is:

```
CALL RADAR(Y1, Y2, X1, X2, INC, DFILE, X, Y)
```

where

Y1 is the bottom position of the search area on the Y axis.

31

Y2 is the top position of the search area on the Y axis.

X1 is the lefthand position of the search area on the X axis.

X2 is the righthand position of the search area on the X axis.

INC is the number of lines to be skipped on the Y axis between iterations (should be set at 1).

DFILE is the name of the user's display file; X is the X coordinate of the light pen when found.

Y is the Y coordinate of the light pen when found.

When RADAR is called from a FORTRAN program, the bell on the DECwriter is sounded and the following message flashed on the bottom of the screen:

TYPE ANY KEY TO START SCAN

At this point, the user places the light pen on the desired point of the screen and then strikes any key on the DECwriter keyboard. The key when struck does not echo, but does start the scan process. If the light pen is not found, the bell is again sounded and the message is again flashed on the screen. This sequence is repeated until the light pen is found, at which time the proper data is returned to the calling program and execution continues. In the first sample program, the light pen is tracked 12 times in order to put 12 squares on the screen. The squares are 1ØØ units square, intensity 2, no blink and drawn with solid lines. The entire 1Ø23 x 1Ø23 screen locations will be scanned.

```
      DIMENSION IF(3ØØ), JF(5Ø)
      CALL INITDF(IF, 3ØØ)
      CALL INITSF(JF, 5Ø)
C
C----------CREATE SQUARE SUBROUTINE
C
```

 32

```
      CALL OPENSF(JF, 1)
      CALL DRAW(1ØØ, Ø, JF)
      CALL DRAW(Ø, 1ØØ, JF)
      CALL DRAW(-1ØØ, Ø, JF)
      CALL DRAW(Ø, -1ØØ, JF)
      CALL CLOSSF(JF)
      CALL START(IF)
C
C----------GET THE 12 LOCATIONS AND PUT UP BOXES
C
      DO 1ØØ, I = 1, 12
      CALL RADAR(Ø, 1Ø23, Ø, 1Ø23, 1, IF, IX, IY)
      CALL MOVE(IX, IY, JF)
1ØØ   CALL SUBJMP(IF, JF, 1)

      PAUSE
      END
```

In our second example, we will define a light pen sensitive square which is 5ØØ units square with its lower lefthand corner located at position X = 2ØØ, Y = 3ØØ. Each time the light pen is found in the square, an intensified point will be put on the screen.

```
      DIMENSION IB(1ØØØ)
      CALL INITDF(IB, 1ØØØ)
      CALL START(IB)

1ØØ   CALL RADAR(3ØØ, 8ØØ, 2ØØ, 7ØØ, 1, IB, IX, IY)
      CALL APNT(IX, IY, 2, Ø, Ø, IB)
      GO TO 1ØØ

      END
```

VI. Saving and Restoring Display Files

Once a display file has been created using the CSN Graphics System, the user has the ability to save a copy, on DECtape or disk and then at any later time to restore the display file to the same program or to any other program. In fact, the SAVED display file can contain any number of references to subroutines and other subroutine files, the only restriction being that the display and subroutine files be in contiguous memory locations. There are two ways to assure the contiguity of display and subroutine files.

The first method requires the creation of one very large display file with the user partitioning the display file into subroutine files. This can be done in a number of ways. For example:

```
DIMENSION IFILE(1ØØØ)
CALL INITDF(IFILE, 5ØØ) !**DISPLAY FILE
CALL INITSF(IFILE(5Ø1), 25Ø)
CALL INITSF(IFILE(751), 25Ø) !**SUBFILE
```

or

```
DIMENSION IFILE(1ØØØ)
EQUIVALENCE(JF1, IFILE(5Ø1)), (JF2, IFILE(751))
CALL INITSF(IFILE, 5ØØ) !**DISPLAY FILE
CALL INITSF(JF1, 25Ø) !**SUBFILE
CALL INITSF(JF2, 25Ø) !**SUBFILE
```

The next and probably simpler method of guaranteeing contiguity is that of dimensioning the display and subroutines in a contiguous fashion. For example:

```
DIMENSION IFILE(5ØØ), JF1(25Ø), JF2(25Ø)
CALL INITDF(IFILE, 5ØØ) !**DISPLAY FILE
CALL INITSF(JF1, 25Ø) !**SUBFILE
CALL INITSF(JF2, 25Ø) !**SUBFILE
```

34

is equivalent to the two methods described above in guaranteeing display file contiguity.

Now that we have covered the problem of contiguity, we can talk about the actual saving and restoring of display files.

There are two basic methods to save and restore display/subroutine files - (1) with individual user supplied names, and (2) with sequential names supplied by the CSN Graphics System.

If the user wishes to supply a unique file name to each save/restore operation, he may use the following subroutine calls:

```
CALL SAVEDF(FILE, NO-WORDS, COMMAND STRING)
CALL RESTOR(FILE, NO-WORDS, COMMAND STRING)
```

where

FILE is the name of the display file.

NO-WORDS is the total number of words in the contiguous display/subroutine file as described above.

COMMAND STRING is a standard RT-11 command string which includes

[DEVICE NAME]:

FILE NAME

[EXTENSION]

;

The DEVICE NAME is optional as the system will assume the system unit if none is specified.

The FILE NAME is a standard RT-11 file name.

EXTENSION is a standard RT-11 file extension - if not specified, the extension CSN is assumed.

35

The ; (semicolon) must be the last character of the command string. If it is omitted, unpredictable results will occur.

The following sample programs should aid the CSN Graphics user in understanding the use of the SAVEDF and RESTOR subroutines.

In the first example, we will create a sample display file consisting of 5Ø squares which are 1ØØ raster units square which start at location Ø, Ø and are incremented by 1Ø units on both the X and Y axes.

```
          DIMENSION IB(1ØØØ) !**CREATE DISPLAY FILE
          CALL INITDF(IB, 1ØØØ) !**INITIALIZE DISPLAY FILE
          CALL START(IB)
          DO 1ØØ I = 1, 5Ø !**DRAW 5Ø BOXES
          CALL MOVE((I-1)*1Ø, (I-1)*1Ø, IB) !**MOVE BEAM
          CALL DRAW(1ØØ, Ø, IB) !**DRAW A BOX
          CALL DRAW(Ø, 1ØØ, IB)
          CALL DRAW(-1ØØ, Ø, IB)
1ØØ       CALL DRAW(Ø, -1ØØ, IB)

          CALL SAVEDF(IB, 1ØØØ, 'RK1:TEST.CSN;') !**SAVE DISPLAY FILE
          END
```

In the next example, we will do essentially the same job as in the previous example, except that we will use a combination of a display and a subroutine file:

```
          DIMENSION IB(5ØØ), JB(5ØØ) !**CREATE FILES
          CALL INITDF(IB, 5ØØ) !**INITIALIZE DISPLAY FILE
          CALL INITSF(JB, 5ØØ) !**INITIALIZE SUBROUTINE FILE

          CALL OPENSF(JB, 1) !**CREATE BOX SUBROUTINE
          CALL DRAW(1ØØ, Ø, JB)
          CALL DRAW(Ø, 1ØØ, JB)
          CALL DRAW(-1ØØ, Ø, JB)
```

36

```
          CALL DRAW(Ø, -1ØØ, JB)
          CALL CLOSSF(JB) !**CLOSE BOX SUBROUTINE

          DO 1ØØ I = 1, 5Ø !**DRAW 5Ø BOXES
          CALL MOVE((I-1)*1Ø, (I-1)*1Ø, IB) !**MOVE BEAM
1ØØ       CALL SUBJMP(IB, JB, 1) !**JUMP TO BOX SUBROUTINE
          CALL SAVEDF(IB, 1ØØØ, 'RK1:TEST.CSN;') !**SAVE DISPLAY AND SUBROUTINE FILES
          END
```

An example of a program which will restore either of the display files saved in the previous examples is:

```
          DIMENSION IFILE(1ØØØ)
          CALL RESTOR(IFILE, 1ØØØ, 'RK1:TEST.CSN;')
          CALL START(IFILE)
          PAUSE
          END
```

In addition to saving and restoring a single display file, the CSN Graphics System allows the graphics programmer to save and restore an entire series of display files on a mass storage device. The only limit to the number of display files saved is an upper bound of 676 save operations or the capacity of the storage medium. For example, on an RKØ5 disk pack, a user could not save 676 2ØØØ word display files as the required storage is greater than the capacity of the disk.

To save or restore a series of display files, the user would simply call:

```
          CALL RECORD(DISPLAY FILE, NUMBER OF WORDS)
          CALL REPLAY(DISPLAY FILE, NUMBER OF WORDS)
```

The meaning of the parameters is the same as for the parameters in the

SAVEDF

and

RESTOR

37

subroutine calls, with the one exception that the user does not specify a command string which includes the device, file name and extension. The command string is not needed in the RECORD and REPLAY subroutines because they generate a series of file names and extensions for their own use. The generated file names run:

```
AA.CSN
AB.CSN
AC.CSN
  .
  .
  .
ZZ.CSN
```

and include a default to device RK1:. (Of course, the default device could be changed to RKØ:, DT1:, or any other mass storage device by simple re-assembling the source program.)

In our first example, we will generate 5Ø squares which are 1ØØ raster units square and run diagonally across the screen. The squares will be intensity 2, solid line with no blink, and will start at location Ø, Ø and move up and to the right by 15 raster units. We will save fifty copies of the display file with each successive copy of the display file containing one more square than the previous.

```
          DIMENSION IB(1ØØØ)
          CALL INITDF(IB, 1ØØØ)
          CALL START(IB)

          DO 1ØØ I = 1, 5Ø
          CALL MOVE((I-1)*15, (I-1)*15, IB) !**THE OLD SQUARE AGAIN
          CALL DRAW(1ØØ, Ø, IB)
          CALL DRAW(Ø, 1ØØ, IB)
          CALL DRAW(-1ØØ, Ø, IB)
          CALL DRAW(Ø, -1ØØ, IB)
1ØØ       CALL RECORD(IB, 1ØØØ) !**RECORD 5Ø ITERATIONS
```

```
          PAUSE
          END
```

In our next example, we will do functionally the same thing as in our last example, except that we will define the square in a graphic subroutine.

```
          DIMENSION IB(9ØØ), JB(1ØØ)
          CALL INITDF(IB, 9ØØ)
          CALL INITSF(JB, 1ØØ)

          CALL OPENSF(JB, 1) !**DEFINE THE SQUARE
          CALL DRAW(1ØØ, Ø, JB)
          CALL DRAW(Ø, 1ØØ, JB)
          CALL DRAW(-1ØØ, Ø, JB)
          CALL DRAW(Ø, -1ØØ, JB)
          CALL CLOSSF(JB)

          DO 1ØØ I = 1, 5Ø
          CALL MOVE((I-1)*15, (I-1)*15, IB)
          CALL SUBJMP(IB, JB, 1)
1ØØ       CALL RECORD(IB, 1ØØØ) !**RECORD THE DISPLAY AND SUBROUTINE FILES

          PAUSE
          END
```

or

```
          DIMENSION IB(1ØØØ)
          EQUIVALENCE(IB(9Ø1), JB)
          CALL INITDF(IB, 9ØØ)
          CALL INITSF(JB, 1ØØ)

          CALL OPENSF(JB, 1)
          CALL DRAW(1ØØ, Ø, JB)
          CALL DRAW(Ø, 1ØØ, JB)
```

39

```
          CALL DRAW(-1ØØ, Ø, JB)
          CALL DRAW(Ø, -1ØØ, JB)
          CALL CLOSSF(JB)

          DO 1ØØ I = 1, 5Ø
          CALL MOVE((I-1)*15, (I-1)*15, IB)
          CALL SUBJMP(IB, JB, 1)
1ØØ       CALL RECORD(IB, 1ØØØ)

          PAUSE
          END
```

Our final sample program shows how to restore a series of display files back from the disk. The program assumes that the display files are 1ØØØ words in length.

```
          DIMENSION IB(1ØØØ)

          DO 1ØØ I = 1, 5Ø
          CALL REPLAY(IB, 1ØØØ)
          CALL START(IB)
          PAUSE
1ØØ       CALL STOP(IB)

          PAUSE
          END
```

It turns out, however, that there is a system program resident on the system disk that will replay any series of display files without regard to their length. To use this program, the user simply types

```
          R REPLAY
```

There is one more subroutine associated with RECORD and REPLAY - it is

```
          CALL RESET
```

The function of the RESET subroutine is to reset the sequential naming code to AA.CSN

40

at any time a RECORD or REPLAY is being executed. Using the RESET routine we could continually replay our series of 5Ø display files in the following manner:

```
              DIMENSION IB(1ØØØ)

1Ø            DO 1ØØ J = 1, 5Ø !**REPLAY AA.CSN THRU BX.CSN
              CALL REPLAY(IB, 1ØØØ) !**GET A DISPLAY
              CALL START(IB) !**SHOW IT
              PAUSE
1ØØ           CALL STOP(IB) !**TURN OFF DISPLAY

              CALL RESET !**RESET TO AA.CSN
              GO TO 1Ø !**BACK FOR MORE
              END
```

VII. Ending a Display File in the Middle

There are times when it is desirable to have the ability to cut off a portion of a display file and also readjust all of the internal pointers to reflect this action. The CSN Graphics System allows the graphics programmer to do this through the following subroutine:

```
CALL DRETN(DISPLAY-FILE, POSITION)
```

where

DISPLAY-FILE is the display file name.

POSITION is the actual word of the display file which is to be the new end of file mark.

This routine can be very useful in cases where the user has invariant data in the beginning of a display file which does not change over time and more variable display data at the end. Rather than regenerating the entire display file after each iteration, the user can set an end of file mark at the end of the invariant portion of display code and regenerate the variable portion of code after each program iteration.

41

VIII. Functions which Return Data Concerning a Display or Subroutine File

The CSN Graphics System contains two function calls which return data about the size and status of a display or subroutine file. The two functions are:

```
ISPACE(FILE)
NEXT(FILE)
```

where

FILE is the name of a display or subroutine file.

Since the CSN Graphics System does not check for display or subroutine file overflow, it is important for the user to have the ability to make such a check. (The CSN Graphics System does not make the check because of the excessive amount of time needed to make such a check before each operation.)

For example, a user could check how much space was left in a display file before inserting a sizable amount of code into the file. As an example, we will write a program which places boxes on the screen until the display file is within 5Ø words of its upper limit.

```
          DIMENSION IB(3ØØ) !**CREATE DISPLAY FILE
          CALL INITDF(IB, 3ØØ)
          CALL START(IB)
          I = Ø
1ØØ       CALL MOVE(I*1Ø, I*1Ø, IB) !**MOVE THE BEAM
          CALL DRAW(1ØØ, Ø, IB) !**DRAW A BOX
          CALL DRAW(Ø, 1ØØ, IB)
          CALL DRAW(-1ØØ, Ø, IB)
          CALL DRAW(Ø, -1ØØ, IB)
          ISIZE = ISPACE(IB) !**GET NUMBER OF WORDS LEFT
          IF(ISIZE.LE.5Ø) GO TO 2ØØ !**CHECK FOR ENOUGH ROOM
          I = I + 1
          GO TO 1ØØ
```

42

```
200         TYPE 210, ISIZE
210         FORMAT(1X, 'THERE ARE ONLY', I3, 1X, 'WORDS LEFT')
            PAUSE
            END
```

At times it is very useful for the graphics programmer to effect changes to certain parts of the display file. For example, to move an object across the screen, the FORTRAN programmer need only change the X and Y coordinates of the object to make it move. To be able to do this, the programmer must be able to find which words of the display file are to be changed. The CSN Graphics System gives the programmer this ability through the use of the NEXT function, which returns the index of the next free word of the display file.

For the feature to be of value, however, the graphics programmer must have some knowledge of the structure of the basic graphics instructions. For example, to move an object around the screen, the X and Y coordinates which we set by a MOVE or APNT call must be changed - the basic point instruction when placed in a display file looks like:

```
POINT
X COORDINATE
Y COORDINATE
```

Therefore, to be able to access the X and Y coordinates the programmer must get the index of the two words following the POINT instruction which is located at the NEXT of the display file. Assume that the display file is named IBUF and the user will save the location of the X coordinate in a variable named IX and the Y coordinate in a variable named IY. The code to accomplish this would be

```
IX = NEXT(IBUF) + 1
IY = IX + 1
CALL MOVE(10, 10, IBUF)
```

43

The reason that we add 1 to the value returned by the NEXT function is because the index returned by the NEXT function is that of the POINT instruction, not the X coordinate. The Y coordinate, of course, is yet another word deeper in the display file - therefore, IY = IX + 1.

To carry this example to its logical conclusion, we will now write a program which draws a 1ØØ unit square at location Ø, Ø on the screen and then moves it up and across the screen to the right. Each iteration will move the square 1Ø units on each of the two axes, until the bottom left corner of the box is located at screen positions 9ØØ, 9ØØ.

```
          DIMENSION IB(1ØØ)
          CALL INITDF(IB, 1ØØ)
          CALL START(IB)
          IX = NEXT(IB) + 1
          IY = IX + 1
          CALL MOVE(Ø, Ø, IB)
          CALL DRAW(1ØØ, Ø, IB)
          CALL DRAW(Ø, 1ØØ, IB)
          CALL DRAW(-1ØØ, Ø, IB)
          CALL DRAW(Ø, -1ØØ, IB)

          DO 1ØØ I = Ø, 9ØØ, 1Ø
          IB(IX) = I
1ØØ       IB(IY) = I

          END
```

IX. Blinking After the Fact

One of the most effective techniques of drawing attention to an item on the screen is to make it blink. This, of course, can be done when an item is originally drawn

44

on the screen by setting the blink parameter to 1. This method does not, however, allow for dynamic changes to the blinking items on the screen. The CSN Graphics System allows the programmer to selectively flash graphic items through the use of the following subroutine:

```
CALL FLASH(FILE, POSITION, OFF/ON)
```

where

FILE is the display or subroutine file name.

POSITION is the index in the display or subroutine file of the start of the graphics instruction.

OFF/ON is a Ø to turn off blinking and a 1 to turn on blinking.

To illustrate the use of the flash subroutine, we will again use the light pen sample program which puts the digits Ø through 9 on the screen, but for our current needs, we will not type out the number touched by the light pen but rather make it flash.

```
        DIMENSION IB(2ØØ), LPBUF(7), LITES(1Ø)
        LOGICAL*1 NUM(3), LLN
        DATA LLN/';'/
        CALL INITDF(IB, 2ØØ)
        LITE ON = 1 !**LIGHT FLAG
        DO 1ØØ I = Ø, 9 !**PUT UP NUMBERS
        CALL MOVE(5ØØ, 1ØØØ - I*1ØØ, IB)
        ENCODE(2, 1Ø5, NUM)I
1Ø5     FORMAT(I2)
        LITES(I + 1) = NEXT(IB) + 1 !**MAKE LIST OF LIGHTED NUMBERS
1ØØ     CALL TEXT(4, Ø, 1, Ø, NUM, IB, I)
        CALL MOVE(9ØØ, 1ØØØ, IB)
        CALL TEXT(4, Ø, 1, 1, 'EXIT;', IB, 99)
        CALL LPEN(LPBUF) !**SET UP LIGHT PEN
        CALL START(IB) !**START DISPLAY
```

```
200         LPBUF(1) = 0 !**ENABLE LIGHT PEN
300         IF(LPBUF(1).EQ.0) GO TO 300 !**WAIT FOR HIT
            IF(LPBUF(2).EQ.99) CALL EXIT

            CALL FLASH(IB, LITES(LITE ON), 0) !**TURN OFF OLD NUM
            LITE ON = LPBUF(2) + 1 !**KEEP TRACK OF FLASHER
            CALL FLASH(IB, LITES(LITE ON), 1) !**TURN ON HIT NUM
            GO TO 200 !**BACK FOR MORE

            END
```

X. A Basic Line Graphing Routine

In addition to the basic graphics routines included in the CSN Graphics System, there is also a basic line graph routine which allows the programmer to produce elegant line graphs with a minimum of effort. The basic subroutine call is:

```
CALL LNGRPH(ARRAY, IMAX, JMAX, IUNIT, JSIZE, INTEN, BLINK, LINE-TYPE,
LINEX, LINEY, XPOS, YPOS, AXIS-INTEN, AXIS-BLINK, AXIS-LINE-TYPE,
FILE, FILE-SIZE, MESSAGE, Y-HIGH, Y-LOW, FLASH-POINT)
```

where

ARRAY is the name of a two-dimensional array of data.

IMAX is the upper bound on the first dimension of the array.

JMAX is the upper bound on the second dimension of the array.

IUNIT is the specific element of the first dimension to be graphed.

JSIZE is the number of elements of the second dimension to be graphed.

INTEN is the intensity of the data graphed.

46

BLINK is the blink bit associated with the graphed data - Ø = NO BLINK, 1 = BLINK

LINE-TYPE is the type of line to be used in the graph - Ø = Solid, 1 = long dash, 2 = short dash, 3 = dot dash.

LINEX is the length of the X axis.

LINEY is the height of the Y axis.

XPOS and YPOS are the X and Y coordinates of the intersection of the X and Y axes.

AXIS-INTEN is the intensity of the axis system - if set to a negative value, the axis system will be invisible.

AXIS-BLINK is the blink bit associated with the axis system.

AXIS-LINE-TYPE is the type of line used in drawing the axis system.

FILE is the name of a display or subroutine file.

FILE-SIZE is the size of the display or subroutine file.

MESSAGE is a character string ending with a semicolon which will be placed below the X axis.

Y-HIGH and Y-LOW are real variables or constants to be used by the graphing routine. If Y-HIGH and Y-LOW are both equal to Ø.ØØ, the LNGRPH subroutine will calculate the high and low values and scale the graph accordingly. If either Y-HIGH or Y-LOW are non-zero, the LNGRPH routine will use Y-HIGH and Y-LOW as the upper and lower bounds and scale the graph accordingly.

47

FLASH-POINT is the index of the line graphed which is returned by the LNGRPH subroutine. This value can be used by the FLASH routine to flash a line or series of graphed lines.

While the user supplies the X and Y positions of the axes intersection, it should be kept in mind that room should be left on the bottom and left side of the graph to allow room for the X/Y axis scaling and the user supplied message. The Y axis scaling requires 15Ø raster units to the left of the graph, while the X axis scaling and user supplied message requires a total of 1ØØ raster units below the X axis.

By setting the AXIS-INTEN parameter to a negative value, the programmer has the ability to plot several values in the same axis system. It should be kept in mind, however, that if more than one set of data is graphed on a single axis system, the user must supply a Y-HIGH and Y-LOW value to make the graph meaningful.

In our first sample program we will plot two exponential decay functions on two different axis systems with the Y-HIGH and Y-LOW values automatically computed. The axis system and data graph will both be intensity 2, solid line with no blink.

```
          DIMENSION X(2, 1ØØ), IB(5ØØ)
          CALL INITDF(IB, 5ØØ)
          DO 1ØØ I = 1, 1ØØ
          X(1, I) = 1.ØØ/FLOAT(I**2)
1ØØ       X(2, I) = 2.ØØ/FLOAT(I**2)
          CALL LNGRPH(X, 2, 1ØØ, 2, 1ØØ, 2, Ø, Ø, 5ØØ, 3ØØ, 5ØØ, 7ØØ, 2, Ø, Ø,
                     IB, 5ØØ, '2.ØØ/I**2;', Ø.ØØ, Ø.ØØ, IJ.)
          CALL LNGRPH(X, 2, 1ØØ, 1, 1ØØ, 2, Ø, Ø, 5ØØ, 3ØØ, 15Ø, 7ØØ, 2, Ø, Ø,
                     IB, 5ØØ, '1.ØØ/I**2;', Ø.ØØ, Ø.ØØ, IJ.)
          CALL START
          PAUSE
          END
```

In our next example, we will plot the same decay functions, but this time on the same axis system.

```
      DIMENSION X(2, 1ØØ), IB(5ØØ)
      CALL INITDF(IB, 5ØØ)

      DO 1ØØ I = 1, 1ØØ
      X(1, I) = 1.ØØ/FLOAT(I**2)
1ØØ   X(2, I) = 2.ØØ/FLOAT(I)

C----------FIND Y-HIGH AND Y-LOW
      YHI = -1Ø.ØØE6
      YLO = 1Ø.ØØE6

      DO 2ØØ I = 1, 2
      DO 2ØØ J = 1, 1ØØ
      IF(X(I, J).LT.YLO) YLO = X(I, J)
      IF(X(I, J).GT.YHI) YHI = X(I, J)
2ØØ   CONTINUE

      CALL LNGRPH(X, 2, 1ØØ, 1, 1ØØ, 2, Ø, Ø, 7ØØ, 4ØØ, 15Ø, 7ØØ, 2, Ø, Ø,
          IB, 5ØØ, 'EXPONENTIAL DECAY CURVES;', YHI, YLO, IJ)
      CALL LNGRPH(X, 2, 1ØØ, 2, 1ØØ, 2, Ø, Ø, 7ØØ, 4ØØ, 15Ø, 7ØØ, -1, Ø, Ø,
          IB, 5ØØ, ';', YHI, YLO, IJ)
      CALL START(IB)
      PAUSE
      END
```

XI. Other CSN Utility Functions and Subroutines

A. Functions

(1) ISWTCH([ARG])

where

ARG is an optional octal variable or constant.

The ISWTCH function allows the FORTRAN programmer to read the contents of the PDP-11 switch register. If called with no argument, the ISWITCH function returns the value set in the switch register - for example, the following program will prompt the user to set 1Ø numbers into the switch register and then return the numbers on the DEC-writer.

```
          DO 1ØØ I = 1, 1Ø
          TYPE 1Ø
1Ø        FORMAT(1X, 'SET A VALUE AND PRESS CR')
          PAUSE
          J = ISWTCH()
1ØØ       TYPE 2Ø, J
2Ø        FORMAT(1X, 'YOUR NUMBER IS', 1X, O6)
          CALL EXIT
          END
```

If the ISWTCH function is called with the optional octal argument, the value of the argument is compared to the current contents of the switch register. If the values match, a 1 is returned to the calling program, otherwise a zero is returned. In our next program, we will present 1Ø octal numbers to the user and request that he set the values into the switch register. The program then repeats whether or not the user matched the requested value.

```
          IVAL = "173Ø41
          DO 1ØØ I = 1, 1Ø
          TYPE 1Ø, IVAL + I
1Ø        FORMAT(1X, 'PLEASE ENTER', 1X, O6, 1X, 'AND PRESS CR')
          PAUSE
          IF(ISWTCH(IVAL + I).EQ.1)GO TO 2Ø
```

50

```
          TYPE 3Ø
3Ø        FORMAT(1X, 'SORRY - BUT NOT RIGHT')
          GO TO 1ØØ
2Ø        TYPE 4Ø
4Ø        FORMAT(1X, 'GOOD SHOW - YOU DID IT RIGHT')
1ØØ       CONTINUE
          CALL EXIT
          END
```

(2) JSIGN(INT-VAR)

where

INT-VAR is the name of an integer variable.

The JSIGN function returns any one of three values depending on the value of the integer variable argument.

VALUE OF ARGUMENT	VALUE RETURNED
Negative	-1
Zero	Ø
Positive	+1

(3) IADDR(VAR-NAME)

where

VAR-NAME is the name of any FORTRAN variable.

The IADDR function returns the absolute memory location of the variable named in the argument list. While this function is of little interest to the average FORTRAN programmer, it can be of great value to the advanced programmer as it allows him to examine the contents of key memory locations via the switch register once the absolute memory location is returned by way of the IADDR function.

To illustrate the use of the IADDR function, we will write a program to type out the address of three variables used in the program - I, J, X. The addresses will

51

be typed in octal so they will be of value to the user.

```
          K = IADDR(I)
          L = IADDR(J)
          M = IADDR(X)
          TYPE 1Ø, K, L, M
1Ø        FORMAT(3(IX, O6))
          CALL EXIT
          END
```

or

```
          TYPE 1Ø, IADDR(I), IADDR(J), IADDR(X)
1Ø        FORMAT(3(IX, O6))
          CALL EXIT
          END
```

(4) LOOK(OCT-ADDR, WORD-BYTE)

where

OCT-ADDR is an absolute memory address in octal.

WORD-BYTE is a Ø for word and a 1 for BYTE.

The LOOK function allows the CSN FORTRAN programmer to examine any memory location in the PDP-11 in either word or byte form. For example, if the FORTRAN program cannot execute when the output speed of the DECwriter is set to fast, we could use the following program to determine the operating environment. Keep in mind that the type speed key word is stored in memory location 56 octal. If the speed is set for fast typing, the value of this location will be zero.

```
          J = LOOK("56, Ø)
          IF(J.NE.Ø) GO TO 1Ø
          TYPE 2Ø
2Ø        FORMAT(1X, 'WRONG TYPE SPEED')
          CALL EXIT
```

52

```
10      ________
         ______
          ___
           .
           .
           .
        END
```

or

```
        IF(LOOK("56, 0).NE.0) GO TO 10
        TYPE 20
20      FORMAT(1X, 'WRONG TYPE SPEED')
        CALL EXIT
10      ________
         ______
          ___
           .
           .
           .
        END
```

(5) ITTYIN()

The ITTYIN function allows the FORTRAN program to accept input from the DECwriter keyboard, one character at a time, without stopping program execution.

There are two basic modes of execution for the function - in the normal mode no characters are available until the user types the carriage return, while in the special mode characters are available as soon as they are typed. To put the GT-44 system into special mode, the FORTRAN programmer must use the NOECHO subroutine which suppresses echoing on the keyboard and puts the system into special mode.

For an example of using the ITTYIN function, see the section describing the NOECHO and ECHO subroutines.

The ITTYIN function returns a negative value if no character is typed. When a character is entered, the ITTYIN function returns the ASCII code for the character typed.

B. Subroutines

(1) CALL IZERO(ARRAY-NAME, NO-ELEMENTS)

CALL RZERO(ARRAY-NAME, NO-ELEMENTS)

CALL LZERO(ARRAY-NAME, NO-ELEMENTS)

where

ARRAY-NAME is the name of the array to be zeroed.

NO-ELEMENTS is the total number of elements in the named array.

The IZERO, RZERO and LZERO subroutines are used as a convenient means of zeroing integer, real and logical arrays in FORTRAN. These subroutines are very useful for initializing and re-initializing FORTRAN arrays.

In the following example we will initialize three different arrays - integer, real and logical*1.

```
DIMENSION I(1ØØ), X(2, 2ØØ)
LOGICAL*1 P(2, 2, 4ØØ)
CALL IZERO(I, 1ØØ)
CALL RZERO(X, 4ØØ)
CALL LZERO(P, 16ØØ)
________
________
________
END
```

(2) CALL IPUT(ADDRESS, VALUE)

where

ADDRESS is an absolute memory address in octal.

54

VALUE is an octal value to be placed in the memory location specified by the address.

The IPUT subroutine allows the advanced FORTRAN programmer to change memory locations either within or outside the bounds of the executing program. This subroutine can be used to change memory locations associated with RT-11 operations. For example, we can re-write the program used to illustrate the LOOK function so that we can change the type speed to slow if it is set at fast.

```
J = LOOK("56, Ø)
IF(J.EQ.Ø) CALL IPUT("56, "5Ø15)
________
________
________
END
```

or

```
IF(LOOK("56, Ø).EQ.Ø) CALL IPUT("56, "5Ø15)
________
________
________
END
```

(3) CALL TTOUT(CHARACTER)

where

CHARACTER is a valid and printable ASCII character code.

The TTOUT subroutine sends a single character at a time to the DECwriter. It can be very useful in echoing characters which are received by the ITTYIN function in the special mode.

For our current example, we will type out the 26 letters of the alphabet using the TTOUT subroutine. All the letters will be on the same line and will be followed by a carriage return and a line feed.

55

```
          LETTER = "1ØØ
          DO 1ØØ I = 1, 26
1ØØ       CALL TTOUT(LETTER + I)
          CALL TTOUT("15)
          CALL TTOUT("12)
          CALL EXIT
          END
```

(4) CALL SNOOZE(TICKS)

where

TICKS is the number of clock ticks to be counted until control returns to the calling program. TICKS are expressed in 60ths of a second.

The SNOOZE subroutine is very useful for creating timed delays in program execution. The following sample program asks the user how many seconds to wait before typing a message to the console, types the message and asks the user for the number of seconds again.

```
1Ø        TYPE 2Ø
2Ø        FORMAT(1X, 'HOW MANY SECONDS')
          ACCEPT 3Ø, ISEC
3Ø        FORMAT(I6)
          ISEC = ISEC*6Ø
          CALL SNOOZE(ISEC)
          TYPE 4Ø
4Ø        FORMAT(1X, 'THE WAIT IS OVER')
          GO TO 1Ø
          END
```

(5) CALL NOECHO

CALL ECHO

The NOECHO and ECHO subroutines allow the FORTRAN programmer to change the

 56

input characteristics of the console input device.

When the NOECHO subroutine is called, characters entered into the DECwriter are not echoed back, and each character is immediately available to the FORTRAN program - even before the carriage return key is hit. If the user wants characters echoed in this mode he must echo them by using the TTOUT subroutine.

When the ECHO subroutine is called, the input mode is returned to its normal state.

In our first sample program, we will simply read characters from the keyboard as they are typed and echo them back until the letter Q is entered, the program will then type the word 'BYE' and exit to the monitor.

```
          CALL NOECHO !**TURN OFF ECHO
1ØØ       J = ITTYIN() !**TRY TO GET A CHARACTER
          IF(J.LT.Ø) GO TO 1ØØ !**IF NONE AVAILABLE - GO BACK
          IF(J.EQ."121) GO TO 1ØØØ !**IS IT A 'Q'
          CALL TTOUT(J) !**ECHO THE CHARACTER
          GO TO 1ØØ
1ØØØ      TYPE 1Ø1Ø
1Ø1Ø      FORMAT(1X, 'BYE')
          CALL ECHO
          CALL EXIT
          END
```

XII. Compiling and Linking FORTRAN Programs Calling CSN Graphics Subroutines and Functions

A. Compiling

CSN Graphic FORTRAN programs are compiled in the same way as non-graphic programs with one exception. It is strongly recommended that the 'U' switch be used in compiling graphics programs to insure that the user service routines are locked in memory at run time. For example, if our FORTRAN program were named TEST.FOR and re-

57

siding on RK1 we would enter the following command string to compile it putting the object program on RK1 and the listing on the teletype.

```
*RK1:TEST, TT:<RK1:TEST/U
```

The reason it is advisable to use the 'U' switch is that the data area of memory containing the display file could possibly be swapped out of memory when the user service routines are needed. If this were to happen, the DPU would attempt to execute the USR instructions which would most certainly cause the system to crash. If locking the USR into memory causes the program to become too large to be loaded, the user can solve the problem by being sure that the arrays used for the display and subroutine files are listed at the very end of the dimension statement insuring that they will not be loaded into the overlay area of memory.

B. Linking

Once compiled, the object module must be linked with the CSN library and the FORTRAN library before being run. Using the output of the FORTRAN compiler described above, we would enter the following command string to the RT-11 linker.

```
*RK1:TEST<RK1:TEST, RKØ:CSNLIB/F
```

Of course, if we were operating from a single disk or tape system, we would simply type

```
*TEST<TEST, CSNLIB/F
```

I. Building the CSNLIB on Disk or DECtape

Assuming that the CSNLIB source programs were delivered on DECtape or DECpak, the user should execute the following steps to create the CSNLIB library.

A. Assembling the Macro Programs

Most of the CSNLIB programs are written in assembly language and must be assembled before the CSNLIB can be built. They must be assembled with the special VTCSN.MAC which is supplied with the other source programs. The following listings assume that the source programs reside on disk unit RK1 and that the output of the macro assembler will also be stored on unit RK1, with no listing generated.

58

```
.R MACRO
*RK1:EXIT<RK1:VTCSN,EXIT
ERRORS DETECTED: 0
FREE CORE: 13610. WORDS

*RK1:FILCHK<RK1:VTCSN,FILCHK
ERRORS DETECTED: 0
FREE CORE: 13609. WORDS

*RK1:INIT<RK1:VTCSN,INIT
ERRORS DETECTED: 0
FREE CORE: 13542. WORDS

*RK1:SPACE<RK1:VTCSN,SPACE
ERRORS DETECTED: 0
FREE CORE: 13579. WORDS

*RK1:NEXTPT<RK1:VTCSN,NEXTPT
ERRORS DETECTED: 0
FREE CORE: 13635. WORDS

*RK1:BLINK<RK1:VTCSN,BLINK
ERRORS DETECTED: 0
FREE CORE: 13564. WORDS

*RK1:END<RK1:VTCSN,END
ERRORS DETECTED: 0
FREE CORE: 13554. WORDS

*RK1:CHARS<RK1:VTCSN,CHARS
ERRORS DETECTED: 0
FREE CORE: 13478. WORDS

*RK1:OPEN<RK1:VTCSN,OPEN
ERRORS DETECTED: 0
FREE CORE: 13543. WORDS

*RK1:DJSR<RK1:VTCSN,DJSR
ERRORS DETECTED: 0
FREE CORE: 13502. WORDS

*RK1:RELDOT<RK1:VTCSN,RELDOT
ERRORS DETECTED: 0
FREE CORE: 13532. WORDS

*RK1:VECTOR<RK1:VTCSN,VECTOR
ERRORS DETECTED: 0
FREE CORE: 13466. WORDS

*RK1:LONGV<RK1:VTCSN,LONGV
ERRORS DETECTED: 0
FREE CORE: 13458. WORDS

*RK1:ONOFF<RK1:VTCSN,ONOFF
ERRORS DETECTED: 0
FREE CORE: 13483. WORDS
```

59

```
*RK1:POINT<RK1:VTCSN,POINT
ERRORS DETECTED: 0
FREE CORE: 13458. WORDS

*RK1:CLOSE<RK1:VTCSN,CLOSE
ERRORS DETECTED: 0
FREE CORE: 13548. WORDS

*RK1:BEGIN<RK1:VTCSN,BEGIN
ERRORS DETECTED: 0
FREE CORE: 13514. WORDS

*RK1:ZERO<RK1:VTCSN,ZERO
ERRORS DETECTED: 0
FREE CORE: 13548. WORDS

*RK1:PUT<RK1:VTCSN,PUT
ERRORS DETECTED: 0
FREE CORE: 13576. WORDS

*RK1:SWITCH<RK1:VTCSN,SWITCH
ERRORS DETECTED: 0
FREE CORE: 13556. WORDS

*RK1:MOVETO<RK1:VTCSN,MOVETO
ERRORS DETECTED: 0
FREE CORE: 13540. WORDS

*RK1:DRWREL<RK1:VTCSN,DRWREL
ERRORS DETECTED: 0
FREE CORE: 13536. WORDS

*RK1:WRTEXT<RK1:VTCSN,WRTEXT
ERRORS DETECTED: 0
FREE CORE: 13540. WORDS

*RK1:SAVE<RK1:VTCSN,SAVE
ERRORS DETECTED: 0
FREE CORE: 12650. WORDS

*RK1:PICSAV<RK1:VTCSN,PICSAV
ERRORS DETECTED: 0
FREE CORE: 13520. WORDS

*RK1:GLOBAL<RK1:VTCSN,GLOBAL
ERRORS DETECTED: 0
FREE CORE: 13734. WORDS

*RK1:ECHO<RK1:VTCSN,ECHO
ERRORS DETECTED: 0
FREE CORE: 13602. WORDS

*RK1:SLEEP<RK1:VTCSN,SLEEP
ERRORS DETECTED: 0
FREE CORE: 13524. WORDS
```

60

```
*RK1:LPEN1<RK1:VTCSN,LPEN1
ERRORS DETECTED: 0
FREE CORE: 13560. WORDS

*RK1:TTY<RK1:VTCSN,TTY
ERRORS DETECTED: 0
FREE CORE: 13515. WORDS

*RK1:SCAN<RK1:SCAN
ERRORS DETECTED: 0
FREE CORE: 14476. WORDS
```

61

B. Compiling the FORTRAN Subroutine

The only FORTRAN subroutine in the CSNLIB is the LNGRPH routine and it can be compiled as follows:

```
.R FORTRA
*RK1:LNGRPH<RK1:LNGRPH
```

C. Building the CSNLIB Using the RT-11 Librarian

Once the source programs have been assembled and compiled, the following instructions should be executed to create the actual CSNLIB on unit RK1.

```
.R LIBR
*RK1:CSNLIB<RK1:VTLIB, ROOM, ACTIVE/C
*RK1:WHAT, GEXIT, FILCHK, INIT, SPACE/C
*RK1:NEXTPT, JSIGN, BLINK, END, CHARS/C
*RK1:OPEN, DJSR, RELDUT, VECTOR/C
*RK1:LONGV, ONOFF, POINT, CLOSE/C
*RK1:BEGIN, ZERO, PUT, SWITCH, LNGRPH/C
*RK1:MOVETO, DRWREL, WRTEXT, SAVE/C
*RK1:PICSAV, GLOBAL, ECHOES, SLEEP/C
*RK1:LPEN1, TTYIN, SCAN, DUMMY
```

XIV. Calling CSNLIB Routines from Assembly Language Programs

The advanced graphics programmer can also use the CSN Graphics System in conjunction with his assembly language programs. The basic linkage is accomplished using general register #5 as a pointer to the actual arguments passed. The actual structure is:

```
(R5) ---------->  # of Args
                Addr. of Arg #1
                Addr. of Arg #2
                Addr. of Arg #3
                       .
                       .
                       .
```

62

Addr. of Arg #n

Very simply, this means that register #5 points to a list containing the total number of arguments passed and the actual addresses of the arguments passed. As an example, let us look at a simple assembly language subroutine which will sum a list of n integers and place the total in the n + 1st argument. The routine could be called from a FORTRAN program as follows:

```
        CALL ADD(I, J, K, L, M, ITOT)
```

The following assembly language subroutine will sum the series of integers and return the sum in the users variable which in this case is called ITOT.

```
        .GLOBL           ADD
        RØ = %0          ;DEFINE REGISTERS
        R1 = %1
        R2 = %2
        R3 = %3
        R4 = %4
        R5 = %5
        SP = %6
        PC = %7
ADD:    MOV(R5) +, RØ    ;GET # OF ARGS
        DEC RØ           ;SUBTRACT 1
        CLR R1           ;CLEAR COUNTER
LOOP:   TST RØ           ;ARE WE DONE YET
        BEQ OUT          ;YUP
        ADD @(R5) +, R1  ;ADD IN A VALUE
        DEC RØ           ;SUBTRACT 1 FROM COUNT
        BR LOOP          ;BACK FOR MORE
OUT:    MOV R1, @(R5)    ;RETURN TOTAL
        RTS PC           ;THAT'S ALL FOLKS
        .END
```

63

The linkage system described above is the one used by the CSN Graphics System and the RT-11 FORTRAN Language - therefore, by using the linkage structure the assembly language programmer can take advantage of the CSN Graphics library.

In our next sample program, we will draw a one hundred-unit square, solid line, with no blink and no light pen sensitivity with its lower lefthand corner located at position 5ØØ, 5ØØ on the screen.

```
            .GLOBL INITDF, MOVE, DRAW, START
            RØ = %Ø             ;DEFINE REGISTERS
            R1 = %1
            R2 = %2
            R3 = %3
            R4 = %4
            R5 = %5
            SP = %6             ;STACK POINTER
            PC = %7             ;PROGRAM COUNTER
START:      MOV #2, ARGLST              ;SET UP FOR CALL INITDF
            MOV #FILE, ARGLST + 2       ;ADDRESS OF DISPLAY FILE
            MOV #SIZE, ARGLST + 4       ;SIZE OF DISPLAY FILE
            MOV #ARGLST, R5             ;SET UP POINTER
            JSR PC, INITDF              ;GO TO IT

            MOV #3, ARGLST              ;SET UP FOR CALL MOVE
            MOV #XPOS, ARGLST + 2       ;SET X POSITION
            MOV #XPOS, ARGLST + 4       ;SET Y POSITION
            MOV #FILE, ARGLST + 6       ;DISPLAY FILE NAME
            MOV #ARGLST, R5             ;POINTER
            JSR PC, MOVE                ;MOVE THE BEAM
```

64

```
MOV #3, ARGLST               ;SET UP FOR CALL DRAW
MOV #HUNDRD, ARGLST + 2      ;DELTA X
CLR ARGLST + 4               ;DELTA Y
MOV #FILE, ARGLST + 6        ;DISPLAY FILE
MOV #ARGLST, R5              ;POINTER
JSR PC, DRAW                 ;DRAW A LINE

MOV #3, ARGLST               ;SET UP FOR CALL DRAW
CLR ARGLST + 2               ;DELTA X
MOV #HUNDRD, ARGLST + 4      ;DELTA Y
MOV #FILE, ARGLST + 6        ;DISPLAY FILE
MOV #ARGLST, R5              ;POINTER
JSR PC, DRAW

NEG HUNDRD                   ;CHANGE TO -1ØØ

MOV #3, ARGLST               ;SET UP FOR CALL DRAW
MOV #HUNDRD, ARGLST + 2      ;DELTA X
CLR ARGLST + 4               ;DELTA Y
MOV #FILE, ARGLST + 6        ;DISPLAY FILE
MOV #ARGLST, R5              ;POINTER
JSR PC, DRAW

MOV #3, ARGLST               ;SET UP FOR CALL DRAW
CLR ARGLST + 2               ;DELTA X
MOV #HUNDRD, ARGLST + 4      ;DELTA Y
MOV #FILE, ARGLST + 6        ;DISPLAY FILE
MOV #ARGLST, R5              ;POINTER
JSR PC, DRAW

MOV #1, ARGLST               ;SET UP FOR CALL START
```

65

```
          MOV #FILE, ARGLST + 2   ;DISPLAY FILE

          JSR PC, START           ;TURN ON THE PICTURE

LOOP:     BR LOOP                 ;WAIT FOR CONTROL C

FILE:     .BLKW 1ØØØ              ;DISPLAY FILE

SIZE:     .WORD 1ØØØ              ;SIZE OF DISPLAY FILE

XPOS:     .WORD 5ØØ               ;X AND Y COORDINATES

ARGLST:   .BLKW 3                 ;ARGUMENT LIST

HUNDRD:   .WORD 1ØØ.              ;DELTA X/Y

          .END START
```

The equivalent FORTRAN program would be:

```
          INTEGER FILE(1ØØØ)
          CALL INITDF(FILE, 1ØØØ)
          CALL MOVE(5ØØ, 5ØØ, FILE)
          CALL DRAW(1ØØ, Ø, FILE)
          CALL DRAW(Ø, 1ØØ, FILE)
          CALL DRAW(-1ØØ, Ø, FILE)
          CALL DRAW(Ø, -1ØØ, FILE)
          CALL START(FILE)
1Ø        GO TO 1Ø
          END
```

While the FORTRAN version of our sample case seems much shorter and simpler, the assembly language version will, in fact, produce a much shorter and more efficient object program than will the FORTRAN version.

The sample assembly language program could have been more efficient by omitting three of the

```
          MOV #3, ARGLST
```

66

and three of the

```
MOV #FILE, ARGLST + 6
```

statements. They were included in the example strictly for clarity.

If we assume that the sample assembly language program was named

```
BOX.MAC
```

and resides on unit RK1, the user could follow the following instructions to assemble, link and run the program from unit RK1.

```
.R MACRO
*RK1:BOX = RK1:BOX

ERRORS DETECTED:Ø
FREE STORAGE XXXXX WORDS.

*^C

.R LINK
*RK1:BOX = RK1:BOX, RKØ:CSNLIB
*^C

.RUN RK1:BOX
```

XV. CSN Graphics Error Messages

The CSN Graphics System combines the error handling facilities of the FORTRAN object time system with the specific error messages generated by the CSN System itself.

Once an error has been encountered by the CSN Graphics System, the following error message is generated by the FORTRAN error handling routine

```
?ERRØ NON-FORTRAN ERROR CALL
IN ROUTINE "XXXXXX" LINE YYYY
```

where

XXXXXX is the name of the FORTRAN routine which caused the error condition to arise.

67

YYYY is the line number in the named routine which caused the error condition.

The line that precedes the FORTRAN error message is the one generated by the CSN Graphics System, which takes the following form:

```
***ERROR***TEXT
```

where

TEXT is an explanatory diagnostic message.

The following sample programs will create error conditions to show the error message generated by the CSN Graphics System. It should be noted that all CSN error messages are considered FATAL and will cause program termination.

In our first example, we will pass too few parameters to a CSN Graphics subroutine.

```
DIMENSION IBUF(1ØØØ)
CALL INITDF(IBUF, 1ØØØ)
CALL START(IBUF)
CALL MOVE(1ØØ, 2ØØ)
________
________
________
END
```

This program would produce the following error message:

```
***FATAL ERROR***WRONG # OF ARGUMENTS PASSED
?ERRØ NON-FORTRAN ERROR CALL
IN ROUTINE "MAIN" LINE 4
```

In our next example, we will create an error condition from a FORTRAN subroutine to see how the error condition is noted by the CSN Graphics System.

```
DIMENSION IB(1ØØØ)
CALL INITDF(IB, 1ØØØ)
```

68

```
          CALL START(IB)
          ________
          ________
          ________
          END

          SUBROUTINE PICTUR(IARRY, ISIZE)
          DIMENSION IARRY(ISIZE), NARRY(5ØØ)
          DO 1ØØ I = 1, 1ØØ
          L = I**2
          M = L/3
1ØØ       CALL MOVE(5ØØ, 5ØØ, NARRY)
          ________
          ________
          ________
          RETURN
          END
```

This program would cause the following error message to be printed on the console.

```
***FATAL ERROR***DISPLAY/SUB FILE NOT KNOWN
?ERRØ NON-FORTRAN ERROR CALL
IN ROUTINE "PICTUR" LINE 6
```

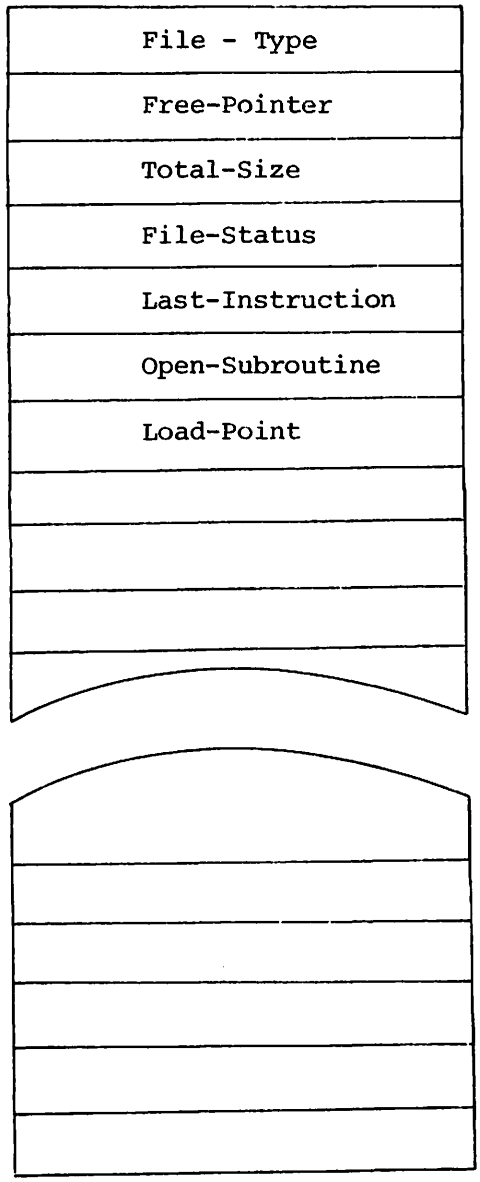
XVI. The Internal Structure of Display and Subroutine Files

The heart of any graphics system is the data structure used by the object time display subroutines; therefore, the design of the data structure used is of great importance. The CSN System uses the same basic data structure for both display and subroutine files with the addition of another data structure within subroutine definitions. For the current discussion, we will call the command data structure the 'Header' and the specific structure used in subroutine definitions the 'internal' data structure.

69

A. The 'Header' Structure

The first seven words of either a display or subroutine file are used by the CSN Graphics System to store and manipulate information concerning the particular file. A representation of this data structure follows:



where:

FILE-TYPE is either ';;' for a display file or ';$' for a subroutine file.

FREE-POINTER is a pointer to the next free word in either the display or subroutine file. This pointer is used each time a new instruction is

70

SORRY! There is no Page 70!

71

placed in a file and updated accordingly.

TOTAL-SIZE is the actual upper memory location of the display or subroutine file.

FILE-STATUS is the words which keep track of the following information for display and subroutine files.

> In display files the file-status word tells whether the particular display file is actually being executed. This information is critical because instructions cannot be inserted into an active display file. The CSN Graphics routine first checks this word before updating a display file. If the file is active, the DPU is first stopped, the new instructions added, and thus the DPU is restarted.
>
> In subroutine files, the file-status word is used to signify whether or not a subroutine definition is currently open, because only one subroutine can be defined at a time in a given subroutine file.

LAST-INSTRUCTION is the actual last graphic instruction inserted in a display file. This particular element of the data structure is of great importance when a number of similar instructions are inserted in a file because it allows the CSN Graphics System to optimize the insertion of graphics instructions.

OPEN-SUBROUTINE. This word in the 'Header' data structure is not used in the case of display files. In subroutine files, however, it is used as a pointer to the 'Internal' data structure used in subroutine definitions.

LOAD-POINT. The load-point word is used to keep track of the absolute memory location of the beginning of the display or subroutine file. This data is crucial in relocating display and subroutine files through

72

the use of the SAVEDF and RESTOR subroutines.

The appropriate values are placed in the 'Header' data structure by the INITDF and INITSF subroutines. The data within the 'Header' structure is manipulated by all of the CSN Graphics routines, but should never be altered by the FORTRAN or Assembly language graphics programmer.

B. The 'Internal' Structure

As well as the 'Header' data structure, each subroutine, defined by the user, has a four-word data control segment - at present, only the first two words are used.

S-File

;$
Free-Pointer
Size
Open/Closed
Old-Instruction
Open-Subroutine
Load Point
; Name
End-Pointer
Unused
Unused

73

where:

;-NAME is the character ';' concentrated with the user-supplied subroutine name.

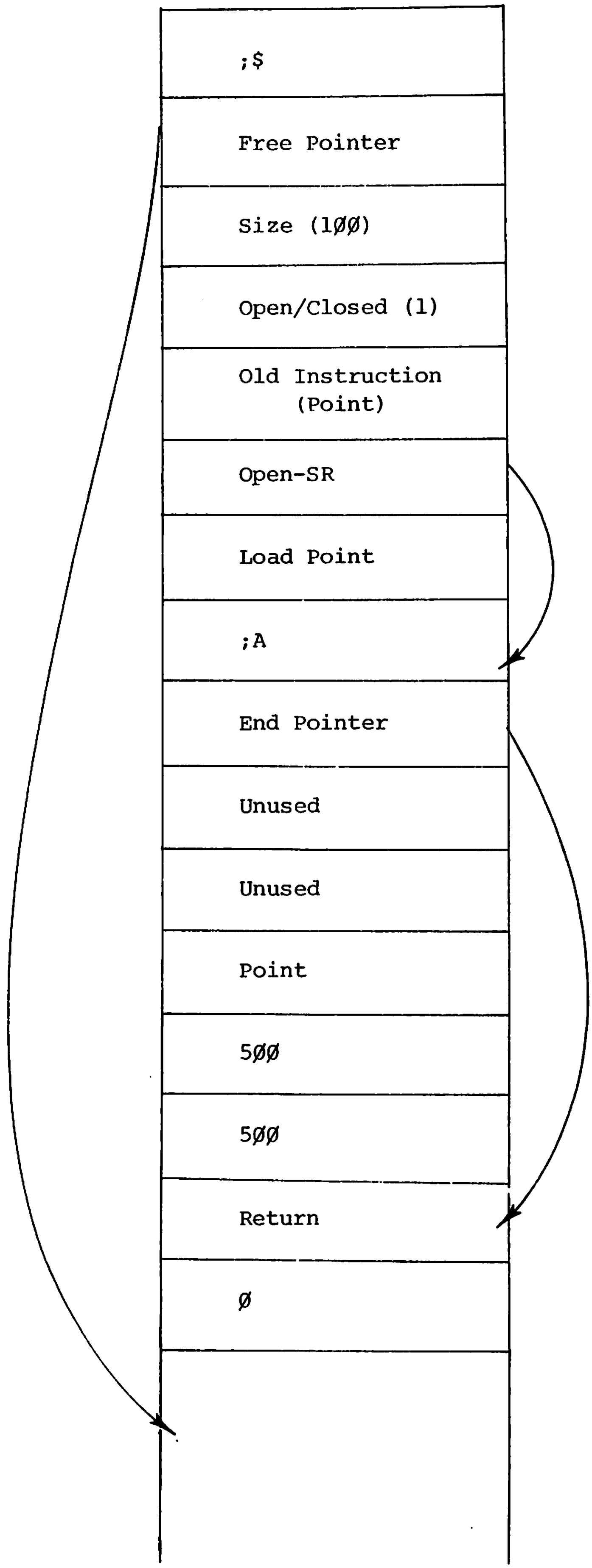
NOTE - All display files, subroutine files and named subroutines are composed of the semicolon character with some other character. Also note that the semicolon character is used as a delimeter for character strings and, therefore, cannot be present in display or subroutine files. In this way, the system guarantees that no spurious headings can appear in a display or subroutine file - unless the user creates such an entry directly in a FORTRAN or Assembly language statement.

END-POINTER is a pointer to the LAST word used by the particular user defined subroutine. This pointer is used by the CSN Graphics System to re-open previously closed user defined subroutines.

To illustrate how this system works, we will now look at several 'snapshots' of a subroutine file header and internal data structure after defining subroutine 'A', subroutine 'B' and then re-opening subroutine 'A'.

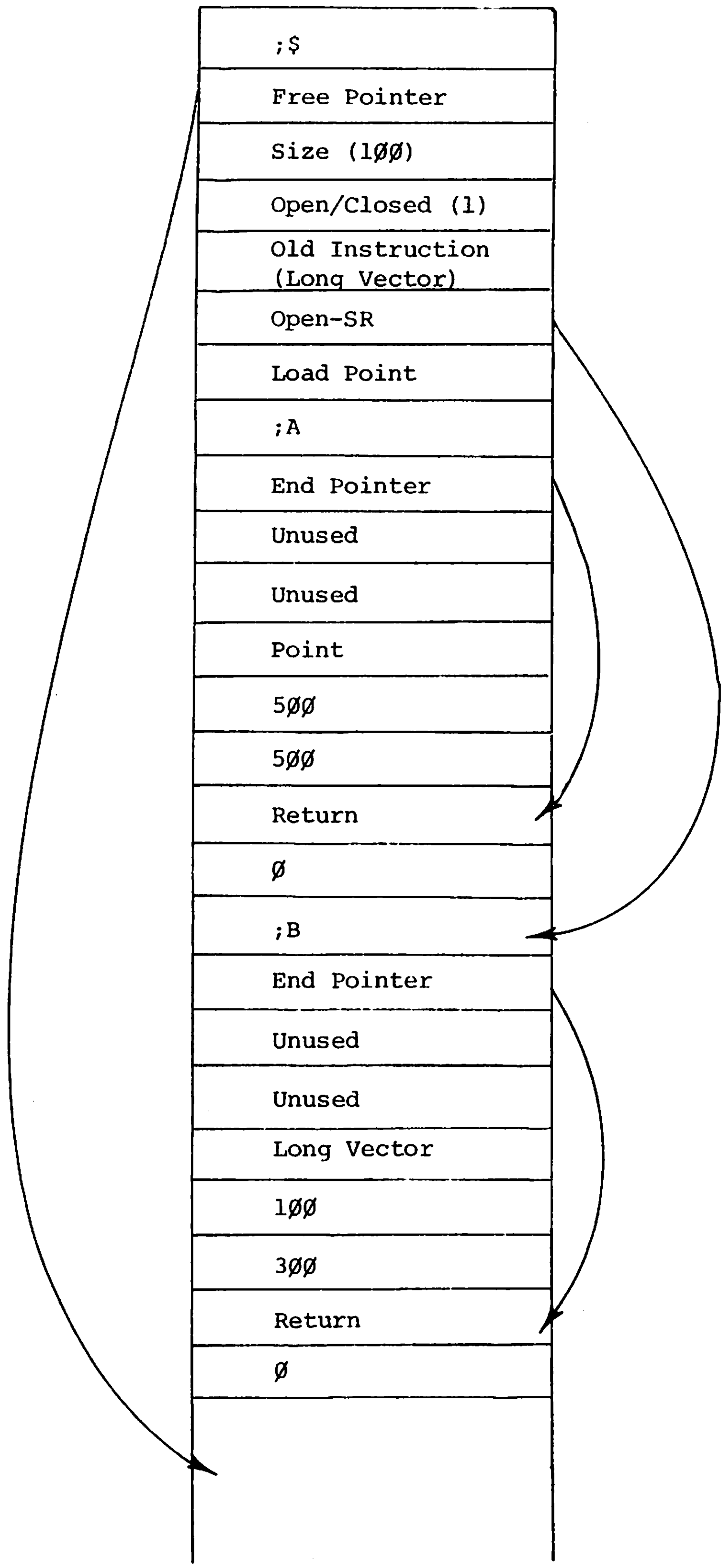
```
DIMENSION JFILE(1ØØ)
CALL INITSF(JFILE, 1ØØ)

CALL OPENSF(JFILE, 'A') !**OPEN A
CALL MOVE(5ØØ, 5ØØ, JFILE) !**PUT IN MOVE
CALL CLOSSF(JFILE)
```

74


;$
Free Pointer
Size (1ØØ)
Open/Closed (1)
Old Instruction (Point)
Open-SR
Load Point
;A
End Pointer
Unused
Unused
Point
5ØØ
5ØØ
Return
Ø

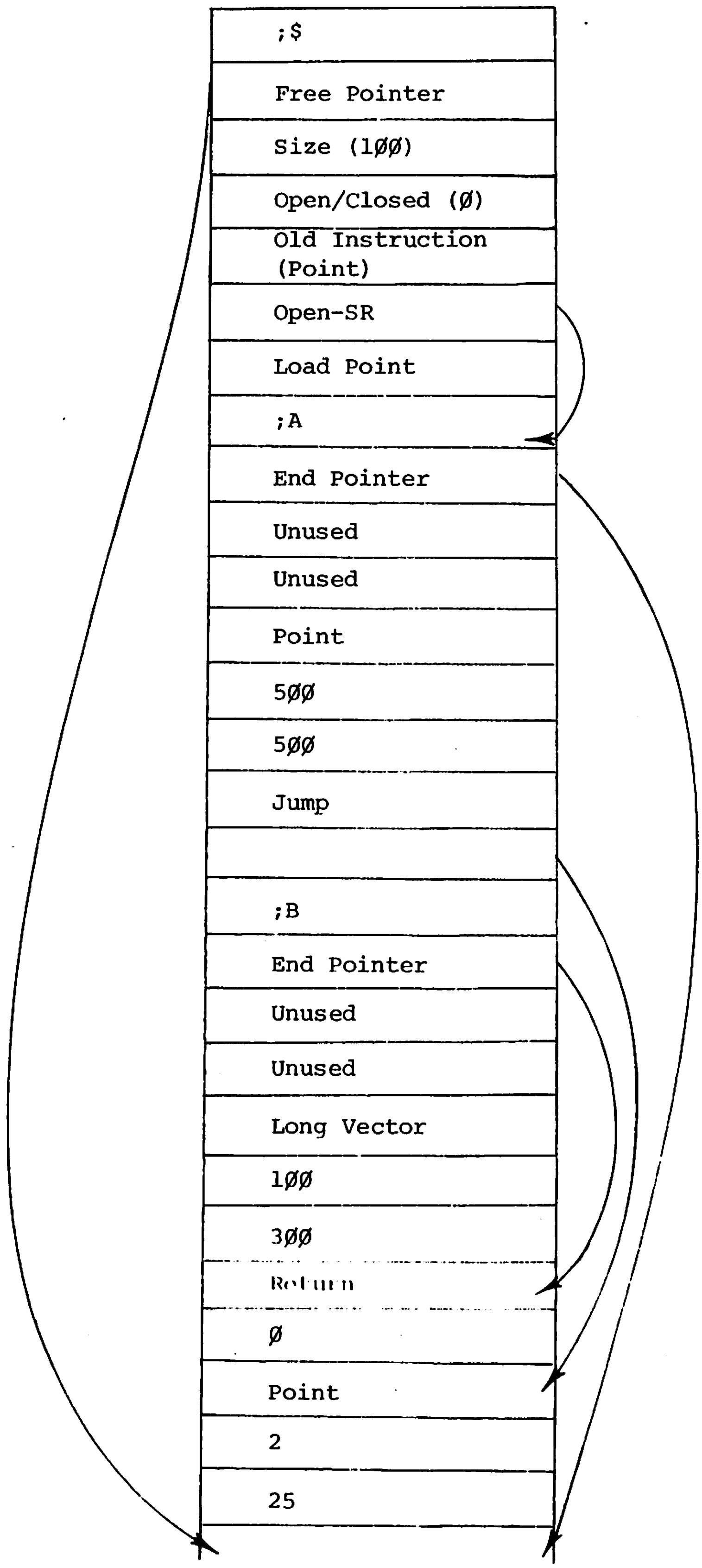
75

```
CALL OPENSF(JFILE, 'B') !**OPEN 'B'

CALL DRAW(1ØØ, 3ØØ, JFILE) !**INSERT VECTOR

CALL CLOSSR(JFILE) !**CLOSE SUBROUTINE
```



76

```
CALL OPENSF(JFILE, 'A') !**RE-OPEN 'A'

CALL MOVE(2, 25, JFILE) !**INSERT POINT
```



77

XVII. Graphics Subroutine Summary

CSN GRAPHICS LIBRARY FORMATS

TERM	MEANING	VALUES
DF-NAME	DISPLAY FILE NAME	N/A
SF-NAME	SUBPICTURE FILE NAME	N/A
FILE	DF-NAME OR SF-NAME	N/A
S-PICTURE	NAMED SUB-PICTURE	0<=NAME<=255
X	X POSITION	0<=X<=1023
Y	Y POSITION	0<=Y<=1023
D-X	DELTA X	0<=D-X<=1023
D-Y	DELTA Y	0<=D-Y<=1023
INTEN	INTENSITY OF DISPLAY	0<=INTEN<=7
BLINK	BLINK	0=NO, 1=YES
LINE	TYPE OF LINE TO BE DRAWN	0=SOLID 1=LONG DASH 2=SHORT DASH 3=DOT DASH
LPEN	LIGHT PEN SENSITIVITY	0=NO, 1=YES
CASE	UPPER/LOWER CASE	0=UPPER, 1=LOWER
STRING	ASCII CHARACTER STRING FOLLOWED BY ';'	1<=L(STRING)<=256
D-NAME	NAME ASSOCIATED WITH LIGHT PEN SENSITIVE ITEM	0<=D-NAME<=16K
BUFF-ADDR	BUFFER ADDRESS FOR LIGHT PEN DATA - 7 WORDS	N/A
INCR	INCREMENT FOR X/Y GRAPH	
ARG	ARGUMENT FOR ISWTCH FCN	-32K<=ARG<=32K
COMMAND-STR	COMMAND STRING FOR SAVEDF & RESTOR ROUTINES IN THE FORM 'DEV:NAME.EXT;'	
NO-ELMTS	NUMBER OF ELEMENTS	
HIDE-KEY	DISPLAY HIDDEN LINES	0=YES, 1=NO
KMAX	NUMBER OF ELEMENTS/3RD DIM	
JMAX	NUMBER OF ELEMENTS/2ND DIM	
IMAX	NUMBER OF ELEMENTS/1ST DIM	
KSIZE	NUM ELEMENTS USED/3RD DIM	
JSIZE	NUM ELEMENTS USED/2ND DIM	
ISIZE	NUM ELEMENTS USED/1ST DIM	
NUM-VARS	NUMBER OF VARIABLES IN LIST	0 < NUM-VARS < 11
VAR-LIST	LIST OF VARIABLE NAMES	

```
#-ARGS              CSN GRAPHICS SUBROUTINE CALLS
======              =============================

[1]      CALL START(DF-NAME)

[1]      CALL STOP(DF-NAME)

[1]      CALL BLNKDF(DF-NAME)

[1]      CALL UNBLNK(DF-NAME)

[1]      CALL REMOVE(DF-NAME)

[0]      CALL BYE

[1]      CALL CLOSSF(SF-NAME)

[2]      CALL OPENSF(SF-NAME,S-PICTURE)

[2]      CALL INITDF(DF-NAME,FILE-SIZE)

[2]      CALL INITSF(SF-NAME,FILE-SIZE)

[7/8]    CALL VECT(D-X,D-Y,INTEN,BLINK,LINE,LPEN,FILE [,D-NAME])

[7/8]    CALL LVECT(D-X,D-Y,INTEN,BLINK,LINE,LPEN,FILE [,D-NAME])

[3]      CALL SUBJMP(FILE,SF-NAME,S-PICTURE)

[6/7]    CALL TEXT(INTEN,BLINK,LPEN,CASE,STRING,FILE [,D-NAME])

[6/7]    CALL RELPNT(D-X,D-Y,INTEN,BLINK,LPEN,FILE [,D-NAME])

[6/7]    CALL APNT(X,Y,INTEN,BLINK,LPEN,FILE [,D-NAME])

[3]      CALL SAVEDF(DF-NAME,# OF WORDS,COMMAND-STR)

[3]      CALL RESTOR(DF-NAME,# OF WORDS,COMMAND-STR)

[1]      CALL LPEN(BUFF-ADDR)

[2]      CALL DRETN(DF-NAME,POSITION)

[21]     CALL LNGRPH(ARRAY,IMAX,JMAX,IUNIT,JSIZE,
              INTEN,BLINK,LINE,IAXINT,LINEX,LINEY,
              X,Y,IAXINT,IAXBLK,IAXLINFILE,FILE-SIZE,MESAG,
              YHIGH,YLOW,FLASH-POINT)

[3/4]    CALL ONSUB(FILE,SF-NAME,S-PICTURE [,S-PICTURE])

[3/4]    CALL OFFSUB(FILE,SF-NAME,S-PICTURE [,S-PICTURE])

[3]      CALL FLASH(FILE,POSITION,OFF/ON)

[2]      CALL RECORD(FILE,SIZE)

[2]      CALL REPLAY(FILE,SIZE)
```

```
[0]     CALL RESET
[3/7/8] CALL DRAW(D-X,D-Y,FILE [,INTEN,BLINK,LINE,LPEN [,DNAME]])
[3/6/7] CALL MOVE(X,Y,FILE [,INTEN,BLINK,LPEN [,DNAME]])
[3/6/7] CALL WRITE(STRING,FILE [,INTEN,BLINK,LPEN,CASE [,DNAME]])
[9]     CALL RADAR(Y-BOTTOM,Y-TOP,X-LEFT,X-RIGHT,INC,D-FILE,X,Y)
```

#-ARGS	CSN GRAPHICS LIB FUNCTION CALLS
[1]	ISPACE(FILE)
[0/1]	ISWTCH([ARGS])
[1]	NEXT(FILE)
[1]	IADDR(VAR-NAME)
[2]	LOOK(OCT-ADDR,WORD/BYTE)
[2]	JSIGN(INTEGER-VARIABLE)
[2]	CALL IPUT(ADDRESS,VALUE)
[1]	CALL TTOUT(CHARACTER)
[0]	ITTYIN()

#-ARGS	CSN GRAPHICS UTILITY SUBROUTINES
[2]	CALL IZERO(ARRAY-NAME,NO-ELMTS)
[2]	CALL LZERO(ARRAY-NAME,NO-ELMTS)
[2]	CALL RZERO(ARRAY-NAME,NO-ELMTS)
[1]	CALL SNOOZE(TICKS)

83

XVIII. Program Listings

DIRECTORY OF MODULE NAMES AND ENTRY POINTS

MODULE	ENTRY/CSECT	ENTRY/CSECT	ENTRY/CSECT
BEGIN.MAC	START	STOP	BLNKDF
	UNBLNK	REMOVE	
BLINK.MAC	FLASH		
CHARS.MAC	TEXT		
CLOSE.MAC	CLOSSF		
DJSR.MAC	SUBJMP		
DRWREL.MAC	DRAW		
ECHOES.MAC	ECHO	NOECHO	
END.MAC	DRETN		
FILCHK.MAC (1)	FILCK$		
GEXIT.MAC	BYE		
GLOBAL.MAC (1)	CSNERR		
INIT.MAC	INITDF	INITSF	
LGRAPH.FOR	LNGRPH		
LINKVT.MAC	LINK		
LONGV.MAC	LVECT		
LPEN1.MAC	LPEN		
MINMAX.MAC	LISBIG	LISMAL	LISMAX
	LISMIN		
MOVETO.MAC	MOVE		
NEXTPT.MAC	NEXT		
ONOFF.MAC	OFFSUB	ONSUB	
OPEN.MAC	OPENSF		
PICSAV.MAC	RECORD	REPLAY	RESET
POINT.MAC	APNT		
RELDOT.MAC	RELPNT		
ROOM.MAC (1)	ROOM$		
SIGN.MAC	JSIGN		
SAVE.MAC	RESTOR	SAVEDF	
SCAN.MAC	RADAR		
SCBUF.MAC	SCROL		
SLEEP.MAC	ITICKS	SNOOZE	
SPACE.MAC	ISPACE		
SWAPS.MAC	NOSWAP	SWAP	
SWITCH.MAC	ISWTCH	IADDR	LOOP
TTYIN.MAC	ITTYIN	ITOUT	
VECTOR.MAC	VECT		
VTCSN.MAC (2)			
WHAT.MAC	WHAT$		
WRTEXT.MAC	WRITE		
ZERO.MAC	IZERO	LZERO	RZERO

NOTES:

(1) ROUTINES USED BY CSN GRAPHICS SYSTEM
(2) MACRO DEFINITIONS USED FOR ASSEMBLING CSN GRAPHICS SYSTEM PROGRAMS

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO START AND STOP DISPLAY
        ;       FILES CREATED BY FORTRAN PROGRAMS
        ;
        ;       CALL START(NAME)
        ;       CALL STOP(NAME)
        ;       CALL BLNKDF(NAME)
        ;       CALL UNBLNK(NAME)
        ;       CALL REMOVE(NAME)
        ;
        ;       IF NO DISPLAY FILE NAME IS GIVEN,
        ;       -ACTVD$- IS ASSUMED
        ;
        .TITLE  BEGIN
        .GLOBL  START,STOP,ACTVD$,ER11$,NMDSP$
        .GLOBL  BLNKDF,UNBLNK,CSNERR,REMOVE
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
START:  TST     (R5)+           ;DID WE GET A D-FILE NAME
        BEQ     NGARG           ;BAD- NO FILE NAME
        MOV     (R5)+,R0        ;GET USER NAME
        TST     NMDSP$          ;TOO MANY DISPLAYS?
        BLE     TOOMNY
        CLR     ONOFF$(R0)              ;SET ON BIT
        ADD     #FIRST$,R0
        .INSRT  R0              ;GET IT ON
        DEC     NMDSP$
        RTS     PC


STOP:   TST     (R5)+           ;DID WE GET THWE NAME
        BEQ     NGARG
        MOV     (R5)+,R0        ;GET USER NAME
        INC     ONOFF$(R0)              ;SET OFF BIT
        .CLEAR                  ;TURN IT OFF
        MOV     #2,NMDSP$       ;RESET # OF DISPLAY FILES
        RTS     PC              ;THAT DOES IT FOLKS

BLNKDF: TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG           ;NOPE
        MOV     (R5)+,R0        ;ADDR OF DISP FILE
        INC     ONOFF$(R0)      ;SET OFF BIT
        ADD     #FIRST$,R0
        .BLANK  R0              ;TURN IT OFF
        RTS     PC
```



```
UNBLNK: TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG           ;NOPE
        MOV     (R5)+,R0
        CLR     ONOFF$(R0)      ;SET ON BIT
        ADD     #FIRST$,R0
        .RESTR  R0              ;TURN IT BACK ON
        RTS     PC

REMOVE: TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG
        MOV     (R5)+,R0        ;GET FILE NAME
        INC     ONOFF$(R0)      ;SET OFF BIT
        ADD     #FIRST$,R0
        INC     NMDSP$          ;RESET # OF DFILES ALOWED
        .REMOV  R0              ;GET RID OF IT
        RTS     PC

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

TOOMNY: .PRINT  #ER11$
        RTS     PC

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO TURN BLINK ON OR OFF
        ;       FROM FORTRAN
        ;
        ;       CALL FLASH(FILE,POSITION,ON-OFF)
        ;
        ;       WHERE:
        ;
        ;               1 = ON
        ;               0 = OFF
        ;
        .TITLE  BLINK
        .GLOBL  FLASH,CSNERR,WHAT$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
FLASH:  CMP     #3,(R5)+        ;ENOUGH ARGS
        BNE     NGARG           ;NOPE
        MOV     (R5)+,R0        ;ADDRESS OF FILE
        JSR     PC,WHAT$        ;CHECK IT OUT
        TST     R1              ;WHAT WAS IT
        BEQ     NGNAME          ;IT WAS NOT GOOD NAME
        MOV     @(R5)+,R1       ;GET POSITION
        ASL     R1              ;CONVERT TO BYTE COUNT
        ADD     R1,R0           ;GET ACTUAL ADDRESS
        TST     @(R5)+          ;ON OR OFF?
        BEQ     OFF
        BIS     #BLKON,(R0)     ;TURN ON THE BLINK BITS
        RTS     PC
OFF:    BIC     #BLKON,(R0)     ;TURN OFF THE BLINK BITS
        RTS     PC

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

NGNAME: MOV     #4,R1           ;BAD NAME
        JMP     CSNERR

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO INSERT A TEXT STRING
        ;       INTO EITHER A DISPLAY OR SUB FILE
        ;
        ;
        ;       CALL TEXT(INTEN,BLINK,LPEN,CASE,STRING,NAME,[DNAME])
        ;
        ;
        .TITLE  CHARS
        .GLOBL  TEXT,INTEN$,BLINK$,CASE$,LPEN$,WHAT$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
TEXT:   CMP     #6,(R5)+        ;ENOUGH ARGS
        BHI     NGARG           ;NOPE
        MOV     #CHAR,R1        ;SET UP CHAR INSTRUCTION
        MOV     @(R5)+,R2       ;GET INTENSITY
        ASL     R2
        BIS     INTEN$(R2),R1
        MOV     @(R5)+,R2       ;GET BLINK
        ASL     R2
        BIS     BLINK$(R2),R1
        MOV     @(R5)+,R2       ;GET LPEN
        ASL     R2
        BIS     LPEN$(R2),R1
        MOV     @(R5)+,R2       ;GET CASE
        ASL     R2
        MOV     R1,-(SP)        ;SAVE CHAR INSTR
        MOV     (R5)+,R3        ;GET STRING ADDRESS
        MOV     (R5)+,R0        ;GET FILE NAME
        JSR     PC,WHAT$        ;SEE WHAT WE GOT
        TST     R1              ;IS IT KOSHER
        BLT     DFILE           ;IT IS A DISPLAY FILE
        BEQ     NGNAME          ;IT IS PURE POOP
        BGT     SFILE           ;IT IS A SUB FILE
DFILE:  MOV     FREE$(R0),R1    ;GET FREE PTR
        TST     ONOFF$(R0)      ;ARE THE LITES ON
        BNE     1$
        .STOP
1$:     TST     @-10(R5)        ;IS LP ON
        BEQ     10$             ;NOPE
        MOV     #DNAME,(R1)+
        MOV     @(R5),(R1)+     ;GET USER DNAME
10$:    MOV     (SP),(R1)+      ;SET CHAR WORD
        MOV     (SP)+,OLDX$(R0) ;SET OLD INSTR
        JSR     PC,STACK        ;STUFF IN THE BYTES
        TST     ONOFF$(R0)
```


98

```
        BNE     2$
        .START
2$:     RTS     PC

SFILE:  MOV     FREE$(R0),R1     ;GET FREE PTR
        MOV     R1,R4            ;SAVE IT FOR LATER
        TST     OPEN$(R0)        ;IS THE DEF OPEN
        BNE     NGCLSD           ;BOO - HISS
        TST     @-10(R5)         ;IS LP ON
        BEQ     5$               ;NOPE
        MOV     #DNAME,(R1)+
        MOV     @(R5),(R1)+      ;GET USER DNAME
5$:     MOV     (SP),(R1)+       ;SET CHAR WORD
        MOV     (SP)+,OLDX$(R0)
        JSR     PC,STACK         ;CRAM THOSE BYTES
        CMP     @OPNSR$(R0),R4   ;IS IT AN EXTENSION TO OLD S-FILE
        BNE     1$               ;YOU BET YOUR PSW
        TST     -(R1)
        MOV     R1,@OPNSR$(R0)   ;SET NEW END PTR
        RTS     PC

1$:     .STOP                    ;STOP THE DISPLAY BECAUSE
                                 ;WE DON'T KNOW IF THIS ROUTINE
                                 ;IS CURRENTLY ALIVE OR NOT
        MOV     OPNSR$(R0),R3    ;GET ADDR OF END PTR
        MOV     #DJMP,@(R3)      ;SET IN DJMP
        ADD     #2,(R3)
        MOV     R5,@(R3)         ;SET IN JUMP ADDRESS
        TST     -(R1)            ;ADJUST FREE PTR
        MOV     R1,(R3)          ;RE SET END PTR
        .START
        RTS     PC               ;BYE - BYE

STACK.  MOVB    (R3)+,LETR       ;GET A CHARACTER
        CMPB    #';,LETR         ;IS IT END OF STRING
        BEQ     OUT
        CMPB    LETR,#100        ;CHECK IF ELEGIBLE FOR CASE WORK
        BLOS    1$
        CMPB    LETR,#132
        BHI     1$
        ADD     CASE$(R2),LETR   ;ADD IN CASE FACTOR
1$:     MOVB    LETR,(R1)+       ;SET CHAR IN PLACE
        BR      STACK            ;BACK FOR MORE

LETR:   .WORD   0

OUT:    BIT     #1,R1            ;DID WE LAND ON BYTE BOUNDRY
        BEQ     FINI             ;NOPE
        CLRB    (R1)+            ;EVEN THE COUNT WITH NULL BYTE
FINI:   MOV     R1,FREE$(R0)     ;SET FREE PTR
        MOV     #DRET,(R1)+
        CLR     (R1)
        RTS     PC

NGARG:  MOV     #1,R1            ;NOT ENOUGH ARGS
        JMP     CSNERR
```



```
NGNAME: MOV     #4,R1            ;BAD NAME
        JMP     CSNERR

NGCLSD: MOV     #2,R1            ;BAD OPEN/CLOSE
        JMP     CSNERR

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO CLOSE A SUB ROUTINE
        ;       DEFINITION FROM FORTRAN -
        ;
        ;       CALL CLOSSF(NAME)
        ;
        ;       WHERE:
        ;
        ;               NAME IS IS THE NAME OF A SUB FILE,
        ;               OR IF NULL - THE SYSTEM WILL ASSUME
        ;               THE ACTIVE SUBFILE NAME -ACTVS$-
        ;
        .TITLE  CLOSE
        .GLOBL  CLOSSF,ACTVS$,FILCK$,ROOM$,CSNERR,ER2$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
CLOSSF: TST     (R5)+           ;DO WEHAVE A S-FILE NAME
        BNE     1$              ;YES AND USE IT
        MOV     ACTVS$,R0       ;NOPE - USE ACTVS$
        BR      2$
1$:     MOV     (R5)+,R0        ;USE GIVEN S-F NAME
2$:     MOV     #1,R1           ;ASK IF IT IS A S-F
        JSR     PC,FILCK$
        TST     R1              ;WELL- WAS IT AN S-F
        BEQ     10$             ;NO GOOD
        TST     OPEN$(R0)               ;ARE ANY ROUTINE DEFS OPEN
        BNE     20$             ;IF CLOSED - WARN SO
        MOV     #2,R1           ;SEE IF THERE IS ROOM ENOUGH
                                ;FOR TWO MORE INSTRUCTIONS
        JSR     PC,ROOM$
        MOV     #1,OPEN$(R0)    ;CLOSE  THE OPEN/CLOSED BIT
        MOV     #DRET,@FREE$(R0)        ;PUT IN DRET INSTR
        ADD     #2,FREE$(R0)
        CLR     @FREE$(R0)
        ADD     #2,FREE$(R0)
        MOV     FREE$(R0),@OPNSR$(R0)
        CLR     OLDX$(R0)       ;CLEAR OLD INSTR WORD
        RTS     PC
10$:    MOV     #4,R1           ;NOT AN S-F
        JMP     CSNERR

20$:    .PRINT  ER2$            ;ALREADY CLOSED
        RTS     PC
        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       PICTURE FILE FROM THE MAIN DISPLAY FILE
        ;
        ;       CALL SUBJMP(FILE,SFILE,ROUTINE NAME)
        ;
        ;       NO ACTIVE NAMES ARE ASSUMED
        ;       THE LINKAGE MAY BE FROM
        ;
        ;       DISPLAY FILE TO SUB-FILE
        ;               OR
        ;       SUB-FILE TO SUB-FILE
        ;
        .TITLE  DJSR
        .GLOBL  SUBJMP,FILCK$,ACTVD$,ACTVS$
        .GLOBL  WHAT$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
SUBJMP: CMP     (R5)+,#3        ;ENOUGH ARGS?
        BNE     NGARG           ;NOPE
        MOV     (R5)+,R0        ;GET FIRST NAME
        JSR     PC,WHAT$        ;FIND OUT WHAT IT IS
        TST     R1              ;WAS IT OK
        BEQ     NGKIND          ;WHAT CRAP
        BGT     20$             ;IT IS A SUB-FILE
        MOV     R0,R3           ;IT'S A D-FILE- GET AT IT
        CLR     OLDX$(R0)
        MOV     (R5)+,R4        ;GET ROUTINE NAME
13$:    MOVB    @(R5)+,50$      ;BUILD ROUTINE NAME
        SWAB    50$
        MOVB    #';,50$
        MOV     50$,R2
        MOV     #-1,R1          ;SET UP FOR NAME SEARCH
        MOV     R4,R0
        JSR     PC,FILCK$
        TST     R1              ;WAS IT OK?
        BEQ     NGKIND          ;NOPE
        TST     ONOFF$(R3)              ;ARE THE LITES ON
        BNE     41$
        .STOP                   ;TURN THEM OFF
41$:    MOV     FREE$(R3),R1    ;GET FREE PTR INTO DFILE
        MOV     #DJSR,(R1)+     ;SET UP SUB JUMP
        MOV     R1,(R1)         ;SET UP FOR .+4
        ADD     #4,(R1)+
        MOV     R2,(R1)+        ;SUB ROUTINE ADDRESS
        MOV     #DRET,(R1)+
```

```
        CLR     (R1)
        SUB     #2,R1           ;RESET FREE PTR
        MOV     R1,FREE$(R3)
        TST     ONOFF$(R3)      ;LITES ON
        BNE     42$
        .START                  ;TURN THEM ON IF OFF
42$:    RTS     PC
;
;
20$:    MOV     R0,R3           ;SAVE S-FILE NAME
        MOV     (R5)+,R0        ;S-FILE 2 NAME
        MOVB    @(R5)+,50$      ;GET ROUTINE NAME
        SWAB    50$
        MOVB    #';,50$
        MOV     50$,R2          ;GET SET FOR NAME SEARCH
        MOV     #-1,R1
        JSR     PC,FILCK$
        TST     R1              ;WAS IT OK
        BEQ     NGKIND          ;NOPE
        TST     OPEN$(R3)       ;IS DEF OPEN
        BNE     NGCLSD
        MOV     FREE$(R3),R4    ;GET PTR INTO S-F-1
        MOV     R4,51$          ;SAVE FREE PTR
        MOV     #DJSR,(R4)+     ;SET UP JUMP
        MOV     R4,(R4)         ;SET UP .+4 INSTR
        ADD     #4,(R4)+
        MOV     R2,(R4)+        ;S-F-2 ADDRESS
        MOV     #DRET,(R4)+
        CLR     (R4)
        SUB     #2,R4
        MOV     R4,FREE$(R3)    ;RESET FREE PTR IN S-F-1
        CMP     51$,@OPNSR$(R3) ;IS IT EXTENDED S-FILE
        BEQ     22$             ;NO
        .STOP
        MOV     @OPNSR$(R3),R2  ;GET LAST ENTRY POINTER
        MOV     #DJMP,(R2)+     ;SET UP JUMP AROUND GARBAGE
        SUB     #6,R4           ;GET NEW JMP ADDR
        MOV     R4,(R2)         ;SET ADDR IN PLACE
        .START
22$:    MOV     FREE$(R3),@OPNSR$(R3)
        RTS     PC
;
;
51$:    .WORD   0
50$:    .WORD   0
NGKIND: MOV     #4,R1           ;BAD FILE NAME
        JMP     CSNERR

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

NGCLSD: MOV     #2,R1           ;OPEN/CLOSED
        JMP     CSNERR

        .END
```



```
;
;
;       C 1975,1976
;
;       ARTHUR I. KARSHMER
;       CENTER FOR SYSTEMS NEUROSCIENCE
;       GRADUATE RESEARCH CENTER
;       UNIVERSITY OF MASSACHUSETTS
;       AMHERST, MA.  01002
;
;
;
;
;       ROUTINE TO DRAW A LONG VECTOR FROM
;       A FORTRAN PROGRAM
;
;       CALL DRAW(D-X,D-Y,FILE [,INTEN,BLINK,LINE,LPEN [,
;               DNAME]])
;
;       TO CHANGE PARAMETERS, THE ITEMS IN '[]' SHOULD
;       BE SPECIFIED
;
;
        .TITLE  DRWREL
        .GLOBL  DRAW,LVECT,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
DRAW:   MOV     (R5)+,R0        ;GET NUMBER OF ARGS
        CMP     R0,#3           ;ENOUGH?
        BLT     NGARG           ;NOPE
        MOV     @(R5)+,DELTAX   ;GET DELTA X
        MOV     @(R5)+,DELTAY   ;GET DELTA Y
        MOV     (R5)+,FILE      ;GET FILE NAME
        CMP     #3,R0           ;DID WE GET NEW PARAMS
        BEQ     NOPE            ;NOPE
        MOV     @(R5)+,INTEN    ;GET INTENSITY
        MOV     @(R5)+,BLINK    ;GET BLINK
        MOV     @(R5)+,LINE     ;GET LINE TYPE
        MOV     @(R5)+,LPEN
        BLE     NOPE
        MOV     @(R5)+,DNAME    ;GET DISPLAY NAME

NOPE:   MOV     #LIST,R5
        JMP     LVECT           ;GO DRAW

NGARG:  MOV     #1,R1           ;# OF ARGS
        JMP     CSNERR

LIST:   .WORD   8.              ;# OF ARGUMENTS
        .WORD   DELTAX          ;ADDR OF D-X
        .WORD   DELTAY          ;ADDR OF D-Y
        .WORD   INTEN           ;ADDR OF INTENSITY
        .WORD   BLINK           ;ADDR OF BLINK
        .WORD   LINE            ;ADDR OF LINE TYPE
        .WORD   LPEN            ;ADDR OF LPEN COMMAND
FILE:   .WORD   0               ;ADDR OF DISPLAY FILE
        .WORD   DNAME           ;ADDR OF DISPLAY NAME
```

```
DELTAX: .WORD   0               ;VALUE OF DELTA X
DELTAY: .WORD   0               ;VALUE OF DELTA Y
INTEN:  .WORD   2               ;DEFAULT VALUE OF INTEN
BLINK:  .WORD   0               ;DEFAULT VALUE OF NO-BLINK
LPEN:   .WORD   0               ;DEFAULT VALUE OF NO-LPEN
DNAME:  .WORD   0               ;VALUE OF DISPLAY NAME
LINE:   .WORD   0               ;DEFAULT LINE TYPE- SOLID

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.   01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO TURN OF ECHOING ON TERMINAL
        ;       FROM A FORTRAN PROGRAM AND THEN TO TURN IT
        ;       BACK ON AGAIN
        ;
        ;       CALL NOECHO
        ;       CALL ECHO
        ;
        ;
        .TITLE  ECHOES
        .GLOBL  NOECHO,ECHO
        .MCALL  ..V2..,.REGDEF,.TTYOUT
        ..V2..
        .REGDEF
        JSW=44                          ;DEFINE JOB STATUS WORD
NOECHO: BIS     #10100,@#JSW            ;SET SPECIAL MODE BITS
        RTS     PC

ECHO:   BIC     #10100,@#JSW            ;RESTORE JSW
        MOV     #15,R0                  ;ISSUE CARRIAGE RETURN
        .TTYOUT
        RTS     PC

        .END
```

```
                ;
                ;
                ;       C 1975,1976
                ;
                ;       ARTHUR I. KARSHMER
                ;       CENTER FOR SYSTEMS NEUROSCIENCE
                ;       GRADUATE RESEARCH CENTER
                ;       UNIVERSITY OF MASSACHUSETTS
                ;       AMHERST, MA.  01002
                ;
                ;
                ;
                ;       ROUTINE TO KILL THE TAIL END OF
                ;       AND GIVEN DISPLAY FILE
                ;
                ;       CALL DRETN(FILE,POSITION)
                ;
                .TITLE  END
                .GLOBL  DRETN,WHAT$,CSNERR
                .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
                ..V2
                .REGDEF
        DRETN:  CMP     (R5)+,#2        ;ENOUGH ARGS
                BNE     NGARG
                MOV     (R5)+,R0        ;GET FILE NAME
                JSR     PC,WHAT$        ;CHECK IT OUT
                TST     R1              ;SEE WHAT IT WAS
                BGE     NGNAME          ;NOT A D-FILE
                CLR     OLDX$(R0)       ;CLEAR OLD INSTRUCTION WORD
                MOV     @(R5)+,R2       ;GET CUT OFF ADDRESS
                ASL     R2              ;MAKE INTO WORD COUNT
                ADD     R0,R2
                MOV     R2,FREE$(R0)    ;RE-SET FREE POINTER
                .STOP
                MOV     #DRET,(R2)+     ;SET RETURN WORD
                CLR     (R2)
                .START
                RTS     PC              ;GO BACK

        NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
                JMP     CSNERR

        NGNAME: MOV     #4,R1           ;BAD NAME
                JMP     CSNERR

                .END
```



```
                ;
                ;
                ;       C 1975,1976
                ;
                ;       ARTHUR I. KARSHMER
                ;       CENTER FOR SYSTEMS NEUROSCIENCE
                ;       GRADUATE RESEARCH CENTER
                ;       UNIVERSITY OF MASSACHUSETTS
                ;       AMHERST, MA.  01002
                ;
                ;
                ;
                ;       UTILITY ROUTINE USED BY FORTRAN GRAPHICS ROUTINES
                ;
                ;       JSR     PC,FILCK$
                ;
                ;       ON INPUT
                ;               (R0)= FILE NAME/ADDRESS
                ;               (R1)= -1 IF NAME SEARCH
                ;                      0 IF DISPLAY FILE SEARCH
                ;                      1 IF SUB FILE CHECK
                ;               (R2)= SUB FILE NAME
                ;
                ;       ON OUTPUT
                ;               (R0)= FILE NAME/ADDRESS
                ;               (R1)= 0 IF CHECK FAILED
                ;                     1 IF CHECK WAS OK
                ;               (R2)= ADDRESS REQUESTED
                ;
                .TITLE  FILCHK
                .GLOBL  FILCK$
                .MCALL  ..V2..,.REGDEF
                ..V2..
                .REGDEF
        FILCK$: TST     R1              ;WHAT TYPE OF CHECK IS IT
                BLT     SEARCH          ;IT IS A S-F NAME SEARCH
                BGT     SUBFIL          ;IT IS A SUB FILE CHECK
                                        ;IT IS A DISPLAY FILE CHECK
                CMP     #";;,(R0)       ;IS IT A DISPLAY FILE?
                BNE     1$              ;NOPE
                MOV     #1,R1           ;ITS OK - REPORT IT
                RTS     PC              ;RETURN WITH ANSWER
        1$:     CLR     R1              ;REPORT THE FAILURE
                RTS     PC
        ;
        ;
        SUBFIL: CMP     #",$,(R0)       ;IS IT A SUB-FILE
                BNE     1$              ;NOPE
                MOV     #1,R1           ;ITS OK - REPORT IT
                RTS     PC
        1$:     CLR     R1              ;REPORT THE FAILURE
                RTS     PC
        ;
        ;
        SEARCH: MOV     #1,R1           ;CHECK IF SUB-FILE
                JSR     PC,FILCK$       ;RECURSIVE CALL
```

```
        TST     R1              ;WAS IT OK
        BEQ     NOPE
        MOV     R0,R1
        ADD     #FIRST$,R1      ;WHERE TO START SEARCH
        MOV     SIZE$(R0),X             ;HIGH LIMIT
1$:     CMP     (R1)+,R2        ;IS IT THE S-R NAME REQUESTED
        BEQ     YES
        CMP     R1,X            ;HAVE WE HIT THE TOP
        BLT     1$              ;NOT QUITE YET
NOPE:   CLR     R1              ;REPORT THE BAD NEWS
        RTS     PC
YES:    MOV     R1,R2           ;REPORT ADRESS
        ADD     #6,R2
        MOV     #1,R1           ;REPORT GOOD NEWS
        RTS     PC              ;GO HOME
X:      .WORD   0
        .END
```



```
        ;
        ;       ROUTINE TO DO A HARD EXIT FROM A GRAPHICS
        ;       PROGRAM - IE- UNLINK THE SCROLLER AND RESET
        ;       THE MONITOR
        ;
        ;       CALL BYE
        ;
        .TITLE  GEXIT
        .GLOBL  BYE
        .MCALL  ..V2..,.REGDEF,.EXIT
        ..V2..
        .REGDEF
BYE:    .UNLNK          ;UNLINK THE SCROLLER
        CLR     R0      ;SET UP FOR HARD EXIT
        .EXIT

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       GLOBAL SYMBOLS USED BY FORTRAN GRAPHICS
        ;       ROUTINES
        ;
        ;       AS WELL AS THE GLOBAL ERROR HANDLER
        ;
        .TITLE  GLOBAL
        .GLOBL  ACTVS$,ACTVD$,INTEN$,LNTYP$,BLINK$,CASE$
        .GLOBL  ER1$,ER2$,ER3$,ER4$,ER5$,LPEN$
        .GLOBL  ER6$,ER7$,ER8$,ER9$,ER10$,ER11$
        .GLOBL  NMDSP$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT
        ..V2..
        .REGDEF


NMDSP$: .WORD   2                       ;NUM OF DISPLAY FILES
INTEN$: .WORD   INT0,INT1,INT2,INT3
        .WORD   INT4,INT5,INT6,INT7
LNTYP$: .WORD   LINE0,LINE1,LINE2,LINE3
BLINK$: .WORD   BLKOFF,BLKON
LPEN$:  .WORD   LPOFF,LPON
CASE$:  .WORD   0,40
ACTVD$: .WORD   0
ACTVS$: .WORD   0
ER1$:   .ASCIZ  \WRONG # OF ARGUMENTS PASSED\
        .EVEN
ER2$:   .ASCIZ  \SUB FILE ALREADY OPEN/CLOSED\
        .EVEN
ER3$:   .ASCIZ  \DISPLAY/SUB-FILE OVERFLOW\
        .EVEN
ER4$:   .ASCIZ  \DISPLAY/SUB FILE NOT KNOWN\
        .EVEN
ER5$:   .ASCIZ  \OPEN SUBROUTINE DEF IN FILE\
        .EVEN
ER6$:   .ASCIZ  \BAD COMMAND IN SAVE/RESTORE\
        .EVEN
ER7$:   .ASCIZ  \BAD DEVICE IN SAVE/RESTORE\
        .EVEN
ER8$:   .ASCIZ  \**SYSTEM ERROR**\
        .EVEN
ER9$:   .ASCIZ  \FILE NOT FOUND IN SAVE/RESTORE\
        .EVEN
ER10$:  .ASCIZ  <15><12>\**WARNING** X/Y VALUE OUT OF RANGE IN RELPNT\
        .EVEN
ER11$:  .ASCIZ  <15><12>\**WARNING** TOO MANY DISPLAY FILES -STARTED-\
        .EVEN
```



```
ERRORS: .WORD   ER1$,ER2$,ER3$,ER4$,ER5$,ER6$
        .WORD   ER7$,ER8$,ER9$,ER10$,ER11$
HEADER: .ASCII  <15><12>/***FATAL ERROR*** /<200>
        .EVEN

CSNERR: .PRINT  #HEADER
        DEC     R1                      ;BRING COUNT TO 0 BASE
        ASL     R1                      ;MAKE BYTE COUNT
        MOV     ERRORS(R1),R0           ;GET ADDR OF MESS
        .PRINT                          ;LET IT FLY
        .UNLNK                          ;UNLINK SCROLLER
        TRAP    200                     ;LET FORTRAN DO IT NOW

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO INITIALIZE EITHER A
        ;       DISPLAY OR SUB ROUTINE FILE
        ;
        ;       CALLED FROM FORTRAN -
        ;
        ;               CALL INITSF(NAME,SIZE)
        ;               CALL INITDF(NAME,SIZE)
        ;
        ;       WHERE:
        ;               NAME IS THE NAME OF AN INTEGER ARRAY
        ;               DIMENSIONED IN THE FORTRAN PROGRAM
        ;       &
        ;
        ;               SIZE IS THE NUMBER OF WORDS SPECIFIED
        ;               IN THE DIMENSION STATEMENT
        ;
        .TITLE  INIT
        .GLOBL  INITSF,INITDF,CSNERR,ACTVS$,ACTVD$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF

INITSF: CMP     (R5)+,#2                ;# OF ARGS OK
        BNE     NG
        MOV     (R5),ACTVS$             ;SET ACTIVE FILE NAME
        MOV     (R5)+,R1
        MOV     #";$,HEAD$(R1)          ;SET HEADER NAME
        MOV     #FIRST$,FREE$(R1)       ;FIRST FREE WORD
        ADD     ACTVS$,FREE$(R1)        ;ADDR OF FREE PTR
        MOV     ACTVS$,SIZE$(R1)
        MOV     @(R5)+,R0
        MOV     R0,R2
        ASL     R2                      ;MAKE IT BYTE COUNT
        ADD     R2,SIZE$(R1)            ;HIGH LIMIT OF FILE
        MOV     #1,OPEN$(R1)            ;SET CLOSED BIT
        CLR     OLDX$(R1)               ;CLEAR OLD X POSITION
        CLR     OLDY$(R1)               ;CLAER OLD Y POSITION
        MOV     R1,LDPNT$(R1)           ;SAVE INITIAL LOAD ADDRESS
        JSR     PC,CLEAR
        RTS     PC                      ;ALL DONE WITH S-FILE

INITDF: CMP     #2,(R5)+                ;ENOUGH ARGS
        BNE     NG
        MOV     (R5),ACTVD$             ;SET ACTIVE NAME/ADDR
        MOV     (R5)+,R1
```



```
        MOV     #";;,HEAD$(R1)          ;SET HEADER WORD
        MOV     #FIRST$,FREE$(R1)       ;SET FREE PTR
        ADD     ACTVD$,FREE$(R1)        ;ADD IN BASE OFFSET
        MOV     ACTVD$,SIZE$(R1)        ;SET HIGH LIMIT
        MOV     @(R5)+,R0
        MOV     R0,R2
        ASL     R2                      ;MAKE IT BYTE COUNT
        ADD     R2,SIZE$(R1)            ;ADD IN SIZE
        MOV     #1,ONOFF$(R1)           ;SET ON/OFF BIT
        CLR     OLDX$(R1)
        CLR     OLDY$(R1)
        MOV     R1,LDPNT$(R1)           ;SAVE INITIAL LOAD POINT
        JSR     PC,CLEAR
        MOV     #DRET,FIRST$(R1)
        .LNKRT
        RTS     PC

NG:     MOV     #1,R1                   ;WRONG # OF ARGS
        JMP     CSNERR

CLEAR:  SUB     #FIRST$,R0
        ADD     #6,R0
        ADD     #FIRST$,R1
        MOV     R1,R2
1$:     CLR     (R2)+
        DEC     R0
        BGT     1$
        RTS     PC

        .END
```

```
      SUBROUTINE LNGRPH(ARRAY,IMAX,JMAX,IUNIT,JSIZE,
     1       INTEN,IBLNK,LINE,LINEX,LINEY,IX,IY,
     2       IAXINT,IAXBLK,IAXLIN,
     3       NAME,ISIZE,MESAG,YHI,YLO,IFLASH)
      DIMENSION ARRAY(IMAX,JMAX),NAME(ISIZE),LH(11),LL(11),
     1       LB(11)
      INTEGER SEMI
C
C---->PUT SEMI-COLONS ON END OF STRINGS FOR TEXT ROUTINE
C
      DATA SEMI/';;'/
      DO 1 I=1,11
      LL(I)=SEMI
      LB(I)=SEMI
1     LH(I)=SEMI
C
C---->SOME CONSTANTS USED IN SUBROUTINE
C
      ISCALE  =     150     !**X POS OF SCALE NUMBERS
      IDOWN   =     20      !**OFF SET FROM TOP OF Y AXIS
C
C---->SEE IF X LENGTH IS PROPPER MULTIPLE OF # OF POINTS
C
      LINX=LINEX
      INCRX=FLOAT(LINEX)/FLOAT(JSIZE)+0.50
      LINX=(JSIZE-1)*INCRX
C
C---->DRAW GRAPH BOX
C
      IF(IAXINT.LT.0)GO TO 500         !**NO AXIS DATA TO PLOT
      CALL APNT(IX,IY,-1,0,0,NAME)
      CALL LVECT(LINX,0,IAXINT,IAXBLK,IAXLIN,0,NAME)
      CALL APNT(IX,IY,-1,0,0,NAME)
      CALL LVECT(0,LINEY,IAXINT,IAXBLK,IAXLIN,0,NAME)
      CALL APNT(IX-ISCALE,IY,-1,0,0,NAME)
C
C---->COMPUTE HIGHS AND LOWS
C
      XHI=YHI
      XLO=YLO
      IF(XHI.NE.0.00.OR.XLO.NE.0.00)GO TO 150
      XHI=1.00E-10
      XLO=1.00E+10
      DO 100 I=1,JSIZE
      X=ARRAY(IUNIT,I)
      IF (X.GT.XHI)XHI=X
      IF (X.LT.XLO)XLO=X
100   CONTINUE
C
C---->CONVERT THE HIGH AND LOW VALUES INTO ASCII STRINGS
C
150   ENCODE(10,200,LH)XHI
      ENCODE(10,200,LL)XLO
200   FORMAT(F10.5)
      ENCODE(4,300,LB)JSIZE
300   FORMAT(I4)
C
C---->PUT SCALE NUM ON THE SCREEN
```



```
C
      CALL TEXT(IAXINT,IAXBLK,0,0,LL,NAME)
      CALL APNT(IX-ISCALE,IY+LINEY-IDOWN,-1,0,0,NAME)
      CALL TEXT(IAXINT,IAXBLK,0,0,LH,NAME)
      CALL APNT(IX,IY-2*IDOWN,-1,0,0,NAME)
      CALL TEXT(IAXINT,IAXBLK,0,0,'0;',NAME)
      CALL APNT(IX+LINX-2*IDOWN,IY-2*IDOWN,-1,0,0,NAME)
      CALL TEXT(IAXINT,IAXBLK,0,0,LB,NAME)
C
C---->PUT USER'S MESSAGE ON SCREEN
C
      CALL APNT(IX,IY-4*IDOWN,-1,0,0,NAME)
      CALL TEXT(IAXINT,IAXBLK,0,0,MESAG,NAME)
C
C---->COMPUTE RANGE AND PLOT ZERO LINE IF ANY
C
      RANGE=FLOAT(LINEY)/(XHI-XLO)
      IF(XLO.EQ.0.00) GO TO 500
      IF(XLO.GE.0.00001)GO TO 500
      IF(XHI.LE.0.00001)GO TO 500      !**NO ZERO LINE
      MID=ABS(XLO)*RANGE               !**ZERO LINE POSITION
      CALL APNT(IX,IY+MID,-1,0,0,NAME)
      CALL LVECT(LINX,0,INTEN,0,1,0,NAME)
C
C---->REPOSITION BEAM AND PLOT LINES
C
500   CALL APNT(IX,IY,-1,0,0,NAME)
      CALL LVECT(0,INT((ARRAY(IUNIT,1)-XLO)*RANGE),
     1       -1,IBLNK,LINE,0,NAME)
C
C---->MASH IN THE REST OF THE VALUES
C
      IFLASH=NEXT(NAME)-1
      DO 600 I=2,JSIZE
      NEXTPT=(ARRAY(IUNIT,I)-XLO)*RANGE -
     1       (ARRAY(IUNIT,I-1)-XLO)*RANGE
600   CALL LVECT(INCRX,NEXTPT,INTEN,IBLNK,LINE,0,NAME)
      RETURN
      END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       FUNCTION TO RETURN THE STATUS OF THE
        ;       RT-11 GRAPHICS SCROLLER TO A CALLING
        ;       PROGRAM. CALL FROM FORTRAN AS:
        ;
        ;       LINK()
        ;
        ;       THE FUNCTION RETURNS:
        ;
        ;               1       IF      GT IS ON
        ;               0       IF      GT IS OFF
        ;
        .TITLE  LINKVT
        .GLOBL  LINK
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF

LINK:   .LNKRT                          ;LINK TO THE SCROLLER
        MOV     @#54,R1                 ;GET BOTTOM ADDRESS
        MOV     300(R1),R0              ;ADD IN OFFSET
        BIC     #177377,R0              ;STRIP AWAY BITS
        SWAB    R0                      ;PUT BIT 8 IN BIT 0
        RTS     PC

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO DRAW LONG VECTORS
        ;       IN EITHER DISPLAY OR SUB FILES
        ;       FROM FORTRAN VIA-
        ;
        ;       CALL LVECT(X,Y,INTEN,BLINK,LINE,LPEN,FILE,[DNAME])
        ;
        ;       WHERE:
        ;
        ;               X & Y ARE THE DELTA X & Y
        ;               INTEN IS THE LINE INTENSITY
        ;               BLINK IS 0 FOR N BLINK & 1 FOR BLINK
        ;               LINE IS THE LINE TYPE
        ;               LPEN IS 0 FOR OFF, 1 FOR ON
        ;               FILE IS THE NAME OF A DISPLAY OR
        ;                       SUB FILE
        ;
        .TITLE  LONGV
        .GLOBL  LVECT,WHAT$,CSNERR
        .GLOBL  INTEN$,BLINK$,LPEN$,LNTYP$,BITS$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
LVECT:  CMP     #7,(R5)+                ;ENOUGH ARGS
        BLE     1$
        JMP     NGARG
1$:     MOV     @(R5),-(SP)             ;SAVE X
        MOV     @(R5)+,R0
        BGE     PLUSX
        NEG     R0                      ;R0 IS NEG SO REVERSE SIGN &
        BIS     #MINUS,R0               ;SET MINUS BITS
PLUSX:  TST     @2(R5)                  ;SHOULD WE INTENSIFY
        BLT     1$                      ;IF NEG THEN NO
        BIS     #INTX,R0                ;SET INTENSITY BITS
1$:     MOV     @(R5),-(SP)             ;SAVE Y POSITION
        MOV     @(R5)+,R1
        BGE     PLUSY
        NEG     R1                      ;REVERSE THE SIGN
        BIS     #MINUS,R1
PLUSY:  MOV     #LONGV,R2               ;GET LONGV INSTR
        JSR     PC,BITS$
        MOV     R1,-(SP)                ;SAVE Y WORD
        MOV     R0,-(SP)                ;SAVE X WORD
        MOV     R2,-(SP)                ;SAVE LONGV WORD
        MOV     R2,THIS                 ;SAVE INSTR FOR TEST LATER
        MOV     (R5)+,R0                ;GET FILE NAME
        JSR     PC,WHAT$                ;WHAT IS IT
```

```
        TST     R1
        BLT     DFILE           ;A D-FILE
        BEQ     NGNAME          ;IT AIN'T NOTIN
        BGT     SFILE
DFILE:  MOV     FREE$(R0),R1    ;GET FREE POINTER
        TST     ONOFF$(R0)      ;ARE LITES ON
        BNE     1$              ;NOPE
         .STOP
1$:     TST     @-4(R5)         ;LP ON?
        BEQ     10$             ;NOPE
        MOV     #INAME,(R1)+
        MOV     @(R5),(R1)+
10$:    JSR     PC,PUSH
        TST     ONOFF$(R0)
        BNE     2$
         .START
2$:     TST     (SP)+           ;POP STACK
        TST     (SP)+           ;POP STACK
        MOV     THIS,OLDX$(R0)  ;SAVE INSTR FOR NEXT PASS
        RTS     PC              ;BYE - BYE

PUSH:   MOV     (SP)+,R5
        CMP     THIS,OLDX$(R0)  ;DO WE NEED COMMAND WORD
        BEQ     1$              ;NO- SAME INSTR AS LAST
        MOV     (SP)+,(R1)+     ;SET LONGV
2$.     MOV     (SP)+,(R1)+     ;SET X WORD
        MOV     (SP)+,(R1)+     ;SET Y WORD
        MOV     #DRET,(R1)+
        CLR     (R1)
        TST     -(R1)
        MOV     R1,FREE$(R0)    ;RESET FREE PTR
        MOV     R5,-(SP)
        RTS     PC

1$:     TST     (SP)+           ;POP LONGV INSTR
        BR      2$

BITS$:  MOV     @(R5)+,R3       ;GET INTENSITY
        TST     R3              ;SHOULD WE INTENSIFY
        BLT     1$              ;NO - IF LT 0
        ASL     R3
        BIS     INTEN$(R3),R2
1$:     MOV     @(R5)+,R3       ;GET BLINK
        ASL     R3
        BIS     BLINK$(R3),R2
        MOV     @(R5)+,R3       ;GET LINE TYPE
        ASL     R3
        BIS     LNTYP$(R3),R2
        MOV     @(R5)+,R3       ;GET LPEN
        ASL     R3
        BIS     LPEN$(R3),R2
        PTS     PC

SFILE:  MOV     FREE$(R0),R1    ;GET FREE PTR
        MOV     R1,R4           ;SAVE IT FOR LATER USE
        TST     OPEN$(R0)       ;IS A DEF OPEN?
        BNE     NGCLSD          ;BOO
```



```
        TST     @-4(R5)         ;IS LP ON
        BEQ     5$              ;NOPE
        MOV     #DNAME,(R1)+
        MOV     @(R5),(R1)+     ;GET USER DNAME
5$:     JSR     PC,PUSH
        MOV     @OPNSR$(R0),R1  ;GET END PTR
        CMP     R1,R4           ;IS IT ADDITION TO OLD FILE
        BEQ     10$             ;YUP
        .STOP
        MOV     #DJMP,(R1)+     ;SET UP DJMP
        MOV     R4,(R1)
        .START
10$:    MOV     FREE$(R0),@OPNSR$(R0)   ;SET UP NEW END PTR
        MOV     OPNSR$(R0),R1   ;ADDR OF SUB ROUTINE
        TST     (SP)+
        TST     (SP)+           ;ADD 4 TO SP (FOR YOU NOVICES)
        MOV     THIS,OLDX$(R0)  ;SAVE LAST INSTRUCTION
        RTS     PC

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

NGNAME: MOV     #4,R1           ;BAD NAME
        JMP     CSNERR

NGCLSD: MOV     #2,R1           ;OPEN/CLOSED
        JMP     CSNERR

THIS:   .WORD   0

        .END
```

CENTER FOR SYSTEMS NEUROSCIENCE - U. OF MASS./AMHERST 01002

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO SET UP AND USE A LIGHT
        ;       PEN HANDLER FROM FORTRAN
        ;
        ;       CALL LPEN(LP-BUFF)
        ;
        ;       WHERE:
        ;               LP-BUFF IS A 7 WORD BUFFER
        ;
        ;       WORD    CONTENTS
        ;       ====    ========
        ;
        ;       -1-     BUFFER FLAG
        ;       -2-     DISPLAY NAME
        ;       -3-     UNUSED
        ;       -4-     DISPLAY PROG COUNTER
        ;       -5-     DISPLAY STATUS REGISTER
        ;       -6-     X COORDINATE
        ;       -7-     Y COORDINATE
        ;
        ;
        .TITLE  LPEN1
        .GLOBL  LPEN,CSNERR,$NR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
LPEN:   TST     (R5)+               ;ENOUGH ARGS
        BEQ     NGARG               ;NOPE
        MOV     (R5)+,LPBUF         ;ADDRESS OF LP-BUFF
        MOV     #LPINT,@#LPVECT     ;SET UP INTERUPT VECTOR
        MOV     #200,@#LPVECT+2     ;SET UP PRIORITY
        RTS     PC                  ;GO BACK

LPINT:  TST     @LPBUF              ;WERE INTERUPTS ENABLED
        BNE     RETURN              ;NO - SO GO BACK
        MOV.    R0,-(SP)            ;SAVE R0
        MOV     LPBUF,R0            ;GET ADDRESS OF 7 WORD BUFFER
        MOV     #1,(R0)+            ;RESET INTERUPT FLAG
        MOV     $NR,(R0)+           ;GET USER DNAME FORM VTBASE
        CLR     (R0)+               ;CLR UNUSED WORD
        MOV     @#DPC,(R0)+         ;GET DPC
        MOV     @#DSR,(R0)+         ;GET DSR
        MOV     @#XSR,(R0)          ;GET X POSITION
        BIC     #176000,(R0)+       ;STRIP OFF INCR BITS
        MOV     @#YSR,(R0)          ;GET Y POSITION
        BIC     #176000,(R0)+       ;STRIP OFF INCR BITS
        MOV     (SP)+,R0            ;RETORE R0
```



```
RETURN: MOV     #1,@#DPC            ;RESTART GRAPHICS
        RTI                         ;RETURN FROM INTERUPT

NGARG:  MOV     #1,R1               ;PRINT ERROR

        JMP     CSNERR

LPBUF:  .WORD   0

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       FUNCTION TO RETURN THE LIST POSITION
        ;       OF THE SMALLEST/LARGEST INTEGER IN A LIST OF
        ;       INTEGERS
        ;
        ;       IF CALLED WITH LESS THAN 2 ARGS, 0 IS RETURNED
        ;
        ;       CALLED FROM FORTRAN AS:
        ;
        ;       LISMIN(ITEM-1,ITEM-2,...,ITEM-N)
        ;       LISLRG(ITEM-1,ITEM-2,...,ITEM-N)
        ;
        ;       IN CASE OF A TIE, THE FIRST OCCURANCE IN THE
        ;       LIST IS REPORTED
        ;
        ;       IF THE ROUTINE IS CALLED AS A SUBROUTINE, BOTH
        ;       THE POSITION AND THE MIN/MAX VALUE WILL BE RETURNED
        ;
        ;       CALL LISMAL(POS,MIN-VAL,ITEM-1,ITEM-2,...,ITEM-N)
        ;       CALL LISBIG(POS,MAX-VAL,ITEM-1,ITEM-2,...,ITEM-N)
        ;
        .TITLE  MINMAX
        .GLOBL  LISMIN,LISMAL,LISMAX,LISBIG
        .MCALL  ..V2..,REGDEF
        ..V2..
        .REGDEF
LISBIG: BIS     #100000,VALUE+2         ;CHANGE SIGN TO NEG
        MOV     #003401,INSTR           ;MODIFY SOME CODE
        CMP     (R5),#4                 ;ENOUGH ARGS
        BGE     OK                      ;YUP
        BR      NG                      ;BOO

LISMAX: CMP     (R5),#2                 ;ENOUGH ARGS
        BLT     BAD                     ;NOPE
        BIS     #100000,VALUE+2         ;CHANGE SIGN TO NEG
        MOV     #003401,INSTR           ;MODIFY SOME CODE
        BR      CONT                    ;GO TO IT

LISMAL: BIC     #100000,VALUE+2         ;CHANGE SIGN TO POS
        MOV     #002001,INSTR           ;MODIFY CODE TO BGE
        CMP     (R5),#4                 ;ENOUGH ARGS
        BGE     OK                      ;YUP
NG:     CLR     @2(R5)                  ;SET ERROR FLAG
        RTS     PC                      ;RETURN

OK:     MOV     #MORE,-(SP)             ;MODIFY LATER JUMP
        MOV     (R5)+,R1                ;GET # OF ARGS
```



```
        SUB     #2,R1                   ;SUBTRACT TWO
        TST     (R5)+                   ;BUMP POINTER
        TST     (R5)+
        BR      NEXT                    ;USE OTHER PART OF ROUTINE

LISMIN: CMP     (R5),#2                 ;ENOUGH ARGS?
        BLT     BAD                     ;NO GOOD
        BIC     #100000,VALUE+2         ;CHANGE SIGN TO POS
        MOV     #002001,INSTR           ;MODIFY CODE TO BGE
CONT:   MOV     (R5)+,R1                ;SAVE NUM OF ARGS
        MOV     #FINI,-(SP)
NEXT:   ASL     R1                      ;MULT BY 2
        ADD     (R5),R1                 ;ADD IN BASE ADDRESS
        MOV     (R5),R3                 ;GET BASE ADDR
VALUE:  MOV     #32767.,R2              ;HI VAL FOR TEST

LOOP:   CMP     R3,R1                   ;DONE WITH LOOP?
        BEQ     OUT                     ;YUP
        CMP     (R3),R2                 ;IS NUM SMALLER/LARGER THAN LAS
INSTR:  BGE     NO                      ;NO
        MOV     (R3),R2                 ;SAVE NEW LOW VALUE
NO:     TST     (R3)+                   ;ADD OFFSET
        BR      LOOP

OUT:    MOV     (R5),R3                 ;GET BASE ADDR

PLACE:  CMP     R3,R1                   ;DONE YET
        BEQ     DONE
        CMP     R2,(R3)                 ;IS THIS THE LOW GUY
        BEQ     DONE                    ;YES IT IS
        TST     (R3)+                   ;ADD OFFSET
        BR      PLACE                   ;BACK FOR MORE

DONE:   SUB     (R5),R3                 ;SUBTRACT BASE ADDR
        MOV     R3,R0                   ;GET POSITION COUNT
        TST     (R0)+                   ;INC BY TWO
        ASR     R0                      ;DIVIDE BY TWO
        JMP     @(SP)+                  ;GO TO PROPER END ROUTINE
FINI:   RTS     PC

BAD:    CLR     R0                      ;SET ERROR FLAG
        RTS     PC                      ;RETURN

MORE:   MOV     R0,@-4(R5)              ;RETURN POS NUMBER
        MOV     R2,@-2(R5)              ;RETURN LOW VALUE
        RTS     PC

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO MOVE THE BEAM FROM A FORTRAN
        ;       PROGRAM BY SPECIFYING ONLY THE X AND
        ;       Y COORDINATES AND THE DISPLAY FILE NAME
        ;
        ;       CALL MOVE(X,Y,FILE [,INTEN,BLINK,LPEN,DNAME])
        ;
        ;       IF THE PARAMETERS IN '[]' ARE SPECIFIED, THE
        ;       DEFAULT PARAMETERS WILL BE RESET.
        ;
        ;
        .TITLE  MOVETO
        .GLOBL  MOVE,APNT,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
MOVE:   MOV     (R5)+,R0        ;GET NUMBER OF ARGS
        CMP     R0,#3           ;LOWER BOUND NUMBER
        BLT     NGARG
        MOV     @(R5)+,X        ;GET X POS
        MOV     @(R5)+,Y        ;GET Y POS
        MOV     (R5)+,FILE      ;GET FILE NAME
        CMP     #3,R0           ;DO WE CHANGE PARAMETERS
        BEQ     NOPE            ;NO - IF EQUAL
        MOV     @(R5)+,INTEN    ;GET INTENSITY
        MOV     @(R5)+,BLINK    ;GET BLINK
        MOV     @(R5)+,LPEN     ;GET LPEN
        BLE     NOPE            ;DON'T GET DNAME
        MOV     @(R5)+,DNAME    ;GET DISPLAY NAME

NOPE:   MOV     #LIST,R5
        JMP     APNT

NGARG:  MOV     #1,R1           ;# OF ARGS
        JMP     CSNERR

LIST:   .WORD   7               ;# OF ARGS
        .WORD   X               ;ADDR OF X VALUE
        .WORD   Y               ;ADDR OF Y VALUE
        .WORD   INTEN           ;ADDR OF INTEN VALUE
        .WORD   BLINK           ;ADDR OF BLINK VALUE
        .WORD   LPEN            ;ADDR OF LPEN VALUE
FILE:   .WORD   0               ;ADDR OF FILE NAME
        .WORD   DNAME           ;ADDR OF DISPLAY NAME

X:      .WORD   0
```



```
Y:      .WORD   0
INTEN:  .WORD   -1              ;NO INTEN AS DEFAULT
BLINK:  .WORD   0               ;NO BLINK AS DEFAULT
LPEN:   .WORD   0               ;NO LPEN AS DEFAULT
DNAME:  .WORD   0               ;NO DNAME AS DEFAULT

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO RETURN THE SUBSCRIPT OF THE
        ;       NEXT FREE WORD IN EITHER A DISPLAY
        ;       OR SUB FILE
        ;
        ;       NEXT(FILE NAME)
        ;
        ;       IF THE FILE NAME IS NOT VALID,
        ;       A NEGATIVE RESULT IS RETURNED
        ;
        .TITLE  NEXTPT
        .GLOBL  NEXT,WHAT$,CSNERR
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF
NEXT:   TST     (R5)+
        BEQ     NGARG           ;NOT ENOUGH ARGS
        MOV     (R5)+,R0        ;GET FILE NAME/ADDRESS
        JSR     PC,WHAT$        ;SEE WHAT WE GOT
        TST     R1
        BEQ     NGNAME          ;WOW WHAT JUNK
        MOV     R0,R1
        MOV     FREE$(R1),R0    ;GET NEXT FRRE PTR
        SUB     R1,R0           ;SUBTRACT BASE ADDRESS
        ASR     R0
        INC     R0
        RTS     PC

NGNAME: MOV     #4,R1           ;BAD NAME
        JMP     CSNERR

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

        RTS     PC
        .END
```


101

101


```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO TURN SUBROUTINES ON OR
        ;       OFF FROM A FORTRAN PROGRAM
        ;
        ;       CALL ONSUB(FILE,S-FILE,S-F-NAME [,FILE-NAME])
        ;       CALL OFFSUB(FILE,S-FILE,S-F-NAME [,FILE-NAME])
        ;
        ;
        ;       IF JUMP IS FROM A DISPLAY FILE, THE
        ;       FOURTH PARAMETER MAY BE OMITTED.
        ;
        ;
        .TITLE  ONOFF 13-AUG-75
        .GLOBL  ONSUB,OFFSUB,FILCK$,WHAT$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
ONSUB:  MOV     #2,R4           ;KEY FOR DJMP SEARCH
        BR      NEXT
OFFSUB: CLR     R4              ;KEY FOR DJSR SEARCH
NEXT:   CMP     (R5)+,#3        ;ENOUGH ARGS
        BLT     NGARG           ;BOO
        MOV     (R5)+,R0        ;GET FIRST ARG- FILE NAME
        JSR     PC,WHAT$        ;WHAT TYPE OF STRUCTURE IS IT
        TST     R1
        BEQ     NGKIND          ;WOW WHAT SHIT
        BGT     SFILE           ;IT IS A SUB-FILE
        MOV     R0,R3           ;SAVR THE NAME FOR A WHILE
        JSR     PC,GETNAM       ;GET NEXT NAME
        MOV     R3,R1
        ADD     #FIRST$,R1

LOOP:   CMP     #173400,(R1)    ;COLUD IT BE A DRET 0?
        BNE     1$              ;NOPE
        TST     2(R1)           ;COULD IT BE A DRET 0
        BEQ     BYE             ;IT IS- SO SAY BYE BYE
1$:     CMP     FLAG(R4),(R1)   ;IS IT A DJMP OR DJSR
        BEQ     YUP             ;YUP
        TST     (R1)+           ;INCR POINTER
        BR      LOOP

YUP:    CMP     R2,4(R1)        ;IS IT THE RIGHT JUMP
        BNE     NOPE            ;NOPE
        MOV     FLAG1(R4),(R1)  ;CHANGE THE INSTRUCTION
NOPE:   TST     (R1)+
        BR      LOOP
```


101

102


```
BYE:    RTS     PC              ;CLOSE UP SHOP AND GO HOME

SFILE:  MOV     @4(R5),N        ;GET NAME OF SUB PICTURE
        MOVB    N,NAME
        SWAB    NAME            ;SWITCH THE BYTES
        MOVB    #';,NAME                ;COMPLETE THE NAME
        MOV     NAME,R2
        MOV     #-1,R1          ;SET UP FOR NAME SEARCH
        JSR     PC,FILCK$       ;GO LOOK IT UP
        TST     R1              ;SO WHAT HAPPENED?
        BEQ     NGKIND          ;WE FOUND IT DOESN'T EXIST
        MOV     R2,R3           ;SAVE ADDR FOR LATER USE
        JSR     PC,GETNAM       ;GET SECOND NAME SET

LOOP1:  CMP     #160000,(R3)    ;CHECK FOR CODE EXPANSION
        BEQ     JUMP            ;FOLLOW POINTERS
LOOP2:  CMP     #173400,(R3)    ;CHECK FOR DRET 0
        BNE     1$              ;NOT YET
        TST     2(R3)           ;LOOK AHEAD FOR DRET 0
        BEQ     BYE             ;WE FOUND THE END OF SUBROUTINE
1$:     CMP     FLAG(R4),(R3)   ;IS IT A JUMP OR DJSR
        BEQ     YES             ;YES
        TST     (R3)+           ;INCR POINTER
        BR      LOOP1

JUMP:   MOV     2(R3),R0        ;SEE IF IT IS REAL OR PHONEY
        SUB     #6,R0           ;CHECK IF IT IS .+4 INSTR
        CMP     R3,R0           ;IF EQUAL THEN .+4 INSTR
        BEQ     LOOP2
        MOV     2(R3),R0        ;GET POINTER TO EXPANDED CODE
        MOV     R0,R3
        BR      LOOP1           ;KEEP LOOKING

YES:    CMP     R2,4(R3)        ;IS THIS THE RIGHT ONE
        BNE     1$
        MOV     FLAG1(R4),(R3)  ;CHANGE THE INSTRUCTION
1$:     TST     (R3)+
        BR      LOOP1


GETNAM: MOV     (R5)+,R0        ;GET NAME OF FILE
        MOV     @(R5)+,N
        MOVB    N,NAME
        SWAB    NAME
        MOVB    #';,NAME
        MOV     #-1,R1          ;CHECK OUT THE NAME
        MOV     NAME,R2
        JSR     PC,FILCK$
        TST     R1              ;WAS IT KOSHER
        BEQ     NGKIND
        RTS     PC

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR
```


102

102


```
NGKIND: MOV     #4,R2                   ;BAD NAME
        JMP     CSNERR

FLAG:   DJSR,DJMP
FLAG1:  DJMP,DJSR
N:      .WORD   0
NAME:   .WORD   0

        .END
```


102

103


```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO NAME AND OPEN A SUB-FILE
        ;       FROM A FORTRAN PROGRAM-
        ;
        ;       CALL OPENSF(FILE NAME,ROUTINE NAME)
        ;
        ;       WHERE:
        ;
        ;               FILE NAME IS THE NAME OF A
        ;               DISPLAY OR SUB-FILE
        ;
        ;       &
        ;
        ;               ROUTINE IS A SINGLE CHARACTER
        ;               SUB ROUTINE NAME NOT ALREADY
        ;               USED IN THAT FILE
        ;
        .TITLE  OPEN
        .GLOBL  OPENSF,FILCK$,ACTVS$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
OPENSF: CMP     #2,(R5)+        ;WAS A NAME GIVEN
        BEQ     1$
        MOV     ACTVS$,R0
        BR      2$              ;NAME GIVEN
1$:     MOV     (R5)+,R0
2$:     MOVB    @(R5)+,7$               ;CREATE NAME RECORD
        SWAB    7$
        MOVB    #';,7$  ;GET USER SUPPLIED NAME
        MOV     7$,R2           ;CONCATENATE
        MOV     #1,R1           ;ASK IF ADDR IS S-F
        JSR     PC,FILCK$       ;GO GET THE ANSWER
        TST     R1              ;IS IT A S-F
        BEQ     21$             ;NOPE
        MOV     R0,R1
        TST     OPEN$(R1)       ;ARE FILES OPEN
        BEQ     22$             ;OH YEAH
        MOV     #-1,R1          ;SEE IF IT ALREADY EXISTS
        JSR     PC,FILCK$
        TST     R1              ;IS IT NEW
        BEQ     15$             ;YOU BET- SET UP HEADER
        SUB     #6,R2
        MOV     R2,OPNSR$(R0)   ;IT'S NOT NEW SO SIMPLY
                                ;PUT IT'S POINTER IN ACTIVE
                                ;FILE SLOT
        SUB     #4,@OPNSR$(R0)
```


103


```
        CLR     OPEN$(R0)       ;SET OPEN BIT
        RTS     PC
;
;
15$:    MOV     R2,@FREE$(R0)   ;SET ITS NAME IN PLACE
        ADD     #2,FREE$(R0)    ;INCR POINTER
        CLR     @FREE$(R0)      ;CLEAR END PTR
        MOV     FREE$(R0),OPNSR$(R0)    ;MAKE ITS POINTER ACTIVE
        ADD     #6,FREE$(R0)    ;INCR FREE PTR
        MOV     FREE$(R0),@OPNSR$(R0)
        CLR     OPEN$(R0)       ;SET OPEN BIT
        RTS     PC
7$:     .WORD   0
21$:    MOV     #4,R1           ;NOT A SUB-FILE
        JMP     CSNERR

22$:    MOV     #5,R1           ;SUBROUTINE DEF OPEN
        JMP     CSNERR

        .END
```


103

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO SAVE A SERIES OF PICTURES FROM
        ;       A RUNNING FORTRAN PROGRAM. THE MAXIMUM NUMBER
        ;       WHICH CAN BE SAVED IS 676.
        ;
        ;       THE FILE NAMES WILL RUN IN SEQUENCE
        ;       FROM AA.CSN TO ZZ.CSN
        ;
        ;       CALLED AS:
        ;
        ;       CALL RECORD(FILE-NAME,NUMBER-OF-ELEMENTS)
        ;       CALL REPLAY(FILE-NAME,NUMBER-OF-ELEMENTS)
        ;
        ;       TO STOP WRITING AND BEGIN READING IN
        ;       THE MIDDLE OF A RUN, THE USER SHOULD
        ;       CALL:
        ;
        ;       CALL RESET
        ;
        ;
        .TITLE  PICSAV
        .GLOBL  RECORD,REPLAY,RESET,SAVEDF,RESTOR,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
REPLAY: MOV     #RESTOR,-(SP)   ;MODIFY SOME CODE FOR LATER
        BR      NEXT            ;GET TO WORK

RECORD: MOV     #SAVEDF,-(SP)   ;MODIFY CODE FOR RECORD ENTRY

NEXT:   CMP     #2,(R5)+        ;ENOUGH ARGS
        BNE     NGARG           ;NOPE
        MOV     (R5)+,NAME      ;GET FILE NAME
        MOV     @(R5)+,SIZE     ;GET FILE SIZE
        CMP     SIZE,#256.      ;IS IT ONE BLOCK LONG
        BGE     YES             ;YUP IT IS
        MOV     #256.,SIZE      ;SET IT RIGHT

YES:    CMPB    FILE+1,#132     ;IS IT LETTER Z YET
        BEQ     RESX            ;YES- SO RESET ALPHABET
        INCB    FILE+1          ;GET NEXT LETTER
        MOV     #LIST,R5        ;SET UP LINKAGE
        JMP     @(SP)+          ;GO TO IT

RESX:   INCB    FILE            ;BUMP FILE NAME
        MOVB    #100,FILE+1     ;RESET SECOND LETTER
```



```
        BR      YES             ;BACK FOR MORE

RESET:  MOVB    #101,FILE       ;RESET FILE NAME COUNTER
        MOVB    #100,FILE+1
        CLR     FLAG            ;CLEAR ENTRY FLAG
        RTS     PC

NGARG:  MOV     #1,R1           ;# OF ARGS
        JMP     CSNERR

LIST:   .WORD   3               ;NUMBER OF ARGUMENTS
NAME:   .WORD   0               ;ADDRESS OF FILE
        .WORD   SIZE            ;ADDRESS OF SIZE
        .WORD   HEAD            ;ADDRESS OF FILE NAME

FLAG:   .WORD   0               ;ENTRY FLAG
SIZE:   .WORD   0               ;SIZE OF DISPLAY FILE
HEAD:   .ASCII  /RK1:/          ;DEVICE NAME
FILE:   .BYTE   101,100,73
        .EVEN

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO CREATE A POINT (INTENSIFIED
        ;       OR UNINTENSIFIED) IN A DISPLAY FILE
        ;       FROM FORTRAN
        ;
        ;       IF THE POINT IS NOT TO BE INTENSIFIED,
        ;       THE USER SHOULD SPECIFY A NEGATIVE
        ;       NUMBER FOR INTENSITY
        ;
        ;       CALL APNT(X,Y,INTEN,BLINK,LPEN,NAME,[DNAME])
        ;
        .TITLE  POINT
        .GLOBL  APNT,ACTVD$,INTEN$,BLINK$,FILCK$
        .GLOBL  CSNERR,WHAT$,LPEN$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
APNT:   CMP     #6,(R5)+        ;DID USER SUPPLY ENOUGH ARGS
        BLE     1$              ;YUP
        JMP     NGARG           ;NOPE
1$:     MOV     10.(R5),R0      ;USER NAME
2$:     CLR     R1              ;CHECK IF VALID NAME
        JSR     PC,WHAT$        ;SEE IF IT IS D-F OR S-F
        TST     R1              ;O.K.???
        BNE     25$             ;YUP
        JMP     NG              ;BAD NAME
25$:    BGT     20$             ;IT IS AN S-FILE
        MOV     #1,R4           ;SET RETURN FLAG TO D-FILE
        TST     ONOFF$(R0)      ;ARE THE LITES ON
        BNE     3$
        .STOP                   ;FLICK OFF THE SWITCH
3$:     MOV     FREE$(R0),R1    ;GET FREE PTR
        MOV     R1,19$          ;SAVE FREE PTR
        MOV     R1,R3
        TST     @8.(R5)         ;IS LP ON
        BEQ.    30$             ;NOPE
        MOV     #DNAME,(R1)+
        MOV     @12.(R5),(R1)+  ;GET USER DNAME
        ADD     #4,R3           ;ADJUST POINT INSTR PTR FOR LPEN
30$:    TST     (R1)+
        MOV     @(R5),-(SP)     ;SAVE X FOR LATER USE
        MOV     @(R5)+,(R1)+    ;X POSITION
        TST     @2(R5)          ;SHOULD WE INTENSIFY
        BLT     31$             ;NOPE
        BIS     #INTK,-2(R1)
31$:    MOV     @(R5),-(SP)     ;SAVE Y FOR LATER USE
        MOV     @(R5)+,(R1)+    ,Y POSITION
```



```
        MOV     #POINT,THIS     ;SET POINT INSTR
        MOV     @(R5)+,R2       ;GET INTEN
        BLT     5$              ;DO NOT INTENSIFY
        ASL     R2
        BIS     INTEN$(R2),THIS ;SET THE BITS
5$:     MOV     @(R5)+,R2       ;GET BLINK
        ASL     R2
        BIS     BLINK$(R2),THIS ;SET THE BITS
        MOV     @(R5)+,R2       ;GET LPEN
        ASL     R2
        BIS     LPEN$(R2),THIS
        CMP     THIS,OLDX$(R0)  ;NEW INSTR
        BNE     15$             ;PUT IN HEADER
        MOV     -4(R1),(R3)+    ;SHIFT X POS UP
        MOV     -2(R1),(R3)+    ;SHIFT Y POS UP
        MOV     R3,R1
        BR      16$
15$:    MOV     THIS,(R3)
16$:    MOV     #DRET,(R1)+
        CLR     (R1)
        SUB     #2,R1           ;SET FREE PTR
        MOV     R1,FREE$(R0)
        TST     R4              ;WAS IT A S-FILE
        BEQ     44$
        TST     ONOFF$(R0)      ;LITES ON?
        BNE     4$
        .START
4$:     TST     (SP)+           ;POP STACK
        TST     (SP)+           ;POP STACK
        MOV     THIS,OLDX$(R0)  ;SAVE INSTR
        RTS     PC
;
;
19$:    .WORD   0
20$:    TST     OPEN$(R0)       ;IS THERE AN OPEN DEF
        BNE     NGCLSD          ;NOPE
        CLR     R4              ;SET RETURN FLAG TO S-FILE
        BR      3$
;
;
44$:    CMP     @OPNSR$(R0),19$ ;IS IT ADDITION TO OLD FILE
        BNE     50$             ;YOU BET YOUR PSW IT IS
        MOV     R1,@OPNSR$(R0)  ;RESET END PTR
        BR      FIVSIX
;
50$:    .STOP
        MOV     @OPNSR$(R0),R3  ;GET ADDR OF END PTR
        MOV     #DJMP,(R3)      ;SET JUMP AROUND OTHER STUFF
        ADD     #2,R3
        MOV     19$,(R3)        .JUMP ADDR
        MOV     R1,@OPNSR$(R0)  ;SET NEW END PTR
        .START
FIVSIX: MOV     OPNSR$(R0),R3   ;GET ADDR OF ROUTINE
        ADD     #4,R3           ;ADDR OF OLD Y
        MOV     (SP)+,(R3)      ;SAVE CURRENT Y
        MOV     (SP)+,-(R3)     ;SAVE CURRENT X
        MOV     THIS,OLDX$(R0)  ;SAVE OLD INSTR
        RTS     PC
```

```
NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

NGCLSD: MOV     #2,R1           ;OPEN/CLOSED
        JMP     CSNERR

NG:     MOV     #4,R1           ;BAD NAME
        JMP     CSNERR

THIS:   .WORD   0
        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO CREATE A POINT, INTENSIFIED
        ;       OR UNINTENSIFIED, RELATIVE TO THE
        ;       CURRENT BEAM POSITION.
        ;
        ;       THE RANGE OF THE DELTA-X AND DELTA-Y
        ;       IS:     -64<RANGE<64.
        ;       IF A NUMBER IS OUT OF RANGE, AN ERROR
        ;       MESSAGE IS PRINTED AND 0,0 IS ASSUMED.
        ;
        ;       CALL RELPNT(X,Y,INTEN,BLINK,LPEN,FILE,[DNAME])
        ;
        ;       IF THE INTEN PARAMETER IS NEGATIVE, THE
        ;       POINT IS NOT INTENSIFIED.
        ;
        .TITLE  RELDOT
        .GLOBL  RELPNT,RDOT$,ER10$,INTEN$,BLINK$
        .GLOBL  LPEN$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
RELPNT: CMP     @2(R5),#63.     ;ARE X AND Y IN RANGE
        BGT     OUTRNG
        CMP     @2(R5),#-63.
        BLT     OUTRNG
        CMP     @4(R5),#63.
        BGT     OUTRNG
        CMP     @4(R5),#-63.
        BLT     OUTRNG
        BR      OK              ;THE #'S ARE IN RANGE

OUTRNG: .PRINT  #ER10$
        CLR     @2(R5)          ;SET DELTA-X = 0
        CLR     @4(R5)          ;SET DELTA-Y = 0

OK:     CMP     #6,(R5)+        ;ENOUGH ARGS
        BHI     NGARG
        CLR     R4              ;SET UP TO CREATE INSTR
        MOV     @(R5),-(SP)     ;SAVE DELTA-X
        MOV     @(R5)+,R0       ;GET IT TO WORK ON
        BGE     PLUSX           ;BR IF POSITIVE
        NEG     R0              ;MAKE IT POSITIVE
        BIS     #20000,R4       ;SET NEG BITS
PLUSX:  TST     @2(R5)          ;SHOULD WE INTENSIFY
        BLT     NEXT            ;NOPE
        BIS     #40000,R4       ;SET INTX BITS
NEXT:   ASL     R0              ;SHIFT VALUE 7 PLACES LEFT
```

```
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     R0,R4           ;SET THE BITS FOR DELTA-X
        MOV     @(R5),-(SP)     ;SAVE DELTA-Y
        MOV     @(R5)+,R0       ;GET DELTA Y
        BGE     PLUSY           ;IT IS POSITIVE
        NEG     R0              ;MAKE IT POSITIVE
        BIS     #100,R4         ;SET - BITS FOR DELTA-Y
PLUSY:  BIS     R0,R4           ;SET DELTA-Y
        MOV     #RELATV,R2      ;SET UP CONTROL WORD
        MOV     @(R5)+,R3       ;GET INTENSITY
        BLT     NOINT           ;DO NOT INTENSIFY
        ASL     R3
        BIS     INTEN$(R3),R2   ;SET INTEN BITS
NOINT:  MOV     @(R5)+,R3       ;GET BLINK BITS
        ASL     R3
        BIS     BLINK$(R3),R2   ;SET BLINK BITS
        MOV     @(R5)+,R3       ;GET LPEN
        ASL     R3
        BIS     LPEN$(R3),R2    ;SET LPEN BITS
        MOV     R4,-(SP)        ;SAVE DATA WORD
        MOV     R2,-(SP)        ;SAVE CONTROL WORD
        JMP     RDOT$           ;LET SOME ONE ELSE FINISH

NOARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO CHECK IF THERE IS ENOUGH ROOM
        ;       TO ADD A GIVEN # OF INSTRUCTIONS TO A
        ;       DISPLAY OR SUB-FILE
        ;
        ;       CALLED BY OTHER CSN ROUTINES AS:
        ;
        ;               JSR     PC,ROOM$
        ;
        ;       WHERE:
        ;
        ;               (R0)=NAME OF FILE
        ;               (R1)=# OF INSTRUCTIONS TO BE ADDED
        ;
        ;       IF O.K. THEN RETURN - ELSE PRINT ERROR
        ;       AND EXIT
        ;
        .TITLE  ROOM
        .GLOBL  ROOM$,CSNERR
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF
ROOM$:  MOV     FREE$(R0),R2    ;GET FREE POINTER
        MOV     SIZE$(R0),R3    ;GET DFILE SIZE
        ADD     R1,R3           ;ADD INSTRS TO BE ADDED
        CMP     R2,R3           ;WIIL THEY FIT
        BGT     TOOBIG          ;NO GOOD
        RTS     PC              ;ALL'S WELL - GO HOME

TOOBIG: MOV     #3,R1           ;SET UP ERROR MESSAGE
        JMP     CSNERR

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO RETURN THE SIGN OF AN INTEGER
        ;       VARIABLE OR CONSTATN FROM A FORTRAN PROGRAM
        ;       CALLED AS:
        ;
        ;       INTVAR = JSIGN(IVAR)
        ;
        ;
        .TITLE  SIGN
        .GLOBL  JSIGN,CSNERR
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF
JSIGN:  TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG           ;BAD NEWS
        CLR     R0              ;ZERO VALUE

        TST     @(R5)           ;TEST INTEGER VALUE
        BGT     PLUS            ;POSITIVE VALUE
        BLT     MINUS           ;NEG VALUE
        RTS     PC

PLUS:   INC     R0              ;SET FOR POS
        RTS     PC

MINUS:  DEC     R0              ;SET FOR NEG
        RTS     PC

NGARG:  MOV     #1,R1           ;SET UP FOR ERROR RETURN
        JMP     CSNERR

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINES TO SAVE AND RESTORE DISPLAY
        ;       AND SUB FILES FROM FORTRAN PROGRAMS
        ;
        ;       CALL SAVEDF(NAME,SIZE,'COMMAND STRING')
        ;       CALL RESTOR(NAME,SIZE,'COMMAND STRING')
        ;
        ;       WHERE:
        ;
        ;               COMMAND STRING IS OF THE FORM
        ;               XXX:YYYYYY.ZZZ
        ;
        ;               XXX = A LEGAL RT-11 DEVICE
        ;               YYYYYY = A USER'S FILE NAME
        ;               ZZZ = AN OPTIONAL EXTENSION
        ;               (IF ZZZ IS NOT GIVEN, CSN IS ASSUMED)
        ;
        .TITLE  SAVE
        .GLOBL  SAVEDF,RESTOR,WHAT$,CSNERR
        .MCALL  ..V2..,.REGDEF,.CSISPC,.READW,.WRITW
        .MCALL  .FETCH,.DELETE,.ENTER,.LOOKUP,.CLOSE
        .MCALL  .PRINT,.EXIT
        ..V2..
        .REGDEF
SAVEDF: MOV     #NEXT1,-(SP)    ;MODIFY SOME CODE - BOO, HISS
ENTER:  CMP     #3,(R5)+        ;ENOUGH ARGS?
        BEQ     1$
        JMP     NGARG
1$:     MOV     (R5)+,R0        ;GET NAME OF D FILE
        MOV     R0,R4           ;PROTECT IT TILL LATER
        JSR     PC,WHAT$        ;CHECK IT OUT
        TST     R1              ;WAS IT KOSHER
        BNE     2$
        JMP     NGNAME          ;BOO ON YOU THIS TIME
2$:     MOV     @(R5)+,R1       ;GET WORD COUNT
        MOV     R1,OLDY$(R4)    ;SAVE THE SIZE FOR RESTORE
        MOV     #BUFF,R2        ;SET UP FOR CSISPC
        MOV     (R5)+,R3        ;ADDR OF USER STRING
LOOP:   MOVB    (R3)+,(R2)+     ;GET A BYTE OF COMMAND
        CMPB    #';,-1(R2)      ;IS THAT ALL
        BEQ     OUT             ;YES
        BR      LOOP            ;NO - GET BAK TO WORK
OUT:    MOVB    #'=,-1(R2)      ;NOW TO FAKE OUT THE CSI
        CLRB    (R2)            ;PUT IN ZERO BYTE
        MOV     SP,X            ;SAVE STACK POINTER
        .CSISPC #OUTSPC,#DEFLT,#BUFF    ;LET IT FLY
        BCC     OK              ;IT WORKED OK
```

```
        MOV     X,SP                    ;RESTORE STACK POINTER
        TSTB    @#52                    ;SEE WHAT WENT WRONG
        BNE     1$
        MOV     #6,R1
        BR      2$
1$:     MOV     #7,R1
2$:     JMP     CSNERR

OK:     MOV     X,SP                    ;RESTORE STACK POINTER
        .FETCH  #DBUF,#DKNAM            ;GET THE DISK HANDLER
        BCC     NEXT                    ;IF OK GO ON
NG:     MOV     #8.,R1                  ;IT WASN'T TOO GOOD
        JMP     CSNERR

NEXT:   JMP     @(SP)+                  ;THIS INSTR MAY BE CHANGED
NEXT1:  .DELETE #EMTARG,#10.,#OUTSPC    ;DELETE OLD FILE
        BCC     GO                      ;ALL WENT WELL
        TSTB    @#52                    ;LET'S SEE WHAT WENT WRONG
        BNE     GO                      ;STILL OK
        BR      NG                      ;NOT SO GOOD

GO:     .ENTER  #EMTARG,#10.,#OUTSPC,#0 ;OPEN A NEW FILE
        BCS     NG                      ;IT DIDN'T WORK
        CLR     R5                      ;CLEAR BLOCK COUNTER
        MOV     #256.,R3                ;INITIAL BLOCK SIZE
        MOV     R1,X                    ;GET USER COUNT
LOOP1:  .WRITW  #EMTARG,#10.,R4,R3,R5
        BCS     NG                      ;NOT SO GOOD
        ADD     #512.,R4                ;INCR D-FILE PTR
        INC     R5                      ;INC BLOCK COUNT
        SUB     #256.,X                 ;DEC TOTAL WORD COUNT
        CMP     #256.,X
        BLT     LOOP1                   ;NOT DONE YET
        TST     R1                      ;IS THIS THE END
        BEQ     FINI
        MOV     X,R3                    ;SMALLER BLOCK SIZE
        CLR     R1                      ;SET UP END FLAG
        BR      LOOP1                   ;ONE MORE TIME

FINI:   .CLOSE  #10.                    ;CLOSE THE FILE
        RTS     PC                      ;GO HOME

RESTOR: MOV     #REST,-(SP)             ;MODIFY SOME CODE
        JMP     ENTER                   ;USE THE OTHER CODE

REST:   .LOOKUP #EMTARG,#10.,#OUTSPC    ;DID WE GET A GOOD NAME
        BCC     ONWARD                  ;YUP
        TSTB    @#52                    ;LET'S SEE WHY
        BEQ     NG
        MOV     #9.,R1
        JMP     CSNERR
ONWARD: MOV     #256.,R3                ;BLOCK SIZE
        MOV     R1,X                    ;USER WORD COUNT
        CLR     R5                      ;BLOCK COUNT
        MOV     R4,-(SP)                ;SAVE START ADDR
        MOV     #1,R2
READ:   .READW  #EMTARG,#10.,R4,R3,R5   ;READ A BLOCK
        BCC     1$                      ;IT DIDN'T WORK
```



```
        TSTB    @#52                    ;END OF FILE
        BEQ     REDOUT                  ;CALL IT QUITS
        JMP     NG
1$:     ADD     #512.,R4                ;INCR POINTER
        INC     R5                      ;INCR BLOCK COUNT
        SUB     #256.,X                 ;HOW CLOSE TO THE END
        CMP     #256.,X
        BLT     READ
        TST     R2
        BEQ     REDOUT
        MOV     X,R3
        CLR     R2
        BR      READ
REDOUT: MOV     (SP)+,R2                ;RESTORE LOAD ADDR
        MOV     LDPNT$(R2),R0           ;GET OLD LOAD POINT
        MOV     R2,LDPNT$(R2)           ;SET CURRENT LOAD POINT
        MOV     R2,R3                   ;GET LOAD PNT
        SUB     R0,R3                   ;GET DIFFERENCE

FIXLOP: CMPB    #';,(R2)                ;IS IT A CONTROL HEADER
        BEQ     SEMI                    ;YES
        CMP     #DJSR,(R2)              ;IS IT A SUB JUMP
        BEQ     YDJSR
        CMP     #DJMP,(R2)              ;IS IT A D JUMP
        BEQ     YDJMP
        TST     (R2)+                   ;INCR POINTER
        SOB     R1,FIXLOP               ;DEC USER COUNT

FIXOUT: .CLOSE  #10.
        RTS     PC

YDJSR:  TST     (R2)+                   ;INCR POINTER
        TST     (R2)                    ;IS IT A ZERO
        BEQ     FIXLOP
        ADD     R3,(R2)+                ;ADD DIF TO .+4 WORD
        ADD     R3,(R2)+                ;ADD DIFF TO ADDR
        BR      FIXLOP

YDJMP:  TST     (R2)+                   ;INCR POINTER
        ADD     R3,(R2)+                ;ADD DIFF TO ADDR
        BR      FIXLOP

SEMI:   CMP     #";;,(R2)               ;IS IT DISPLAY FILE HEADER
        BNE     1$
        ADD     R3,FREE$(R2)            ;FIX FREE PTR
        ADD     R3,SIZE$(R2)
        TST     (R2)+                   ;INC PTR
        BR      FIXLOP
1$:     CMP     #";$,(R2)               ;IS IT A SUB FILE HEADER
        BNE     2$
        ADD     R3,FREE$(R2)            ;FIX POINTERS
        ADD     R3,SIZE$(R2)
        ADD     R2,OPNSR$(R2)
;       SUB     #4,@OPNSR$(R2)
        TST     (R2)+
        BR      FIXLOP
2$:     CMP     -4(R2),#DRET            ;IS IT REALY SUB-PIC
        BEQ     10$
```

```
        CMP     -4(R2),#DJMP
        BEQ     10$
        CMP     #";$,-14.(R2)
        BEQ     10$
        BR      12$
10$:    ADD     R3,2(R2)                ;FIX END PTR
12$:    TST     (R2)+
        BR      FIXLOP

NGARG:  MOV     #1,R1                   ;# OF ARGS
        JMP     CSNERR

NGNAME: MOV     #4,R1                   ;BAD NAME
        JMP     CSNERR

DKNAM:  .RAD50  /DK /
OUTSPC: .BLKW   20.
EMTARG: .BLKW   10
        .BLKW   12.
X:      .WORD   0
DEFLT:  .RAD50  /CSN/
        .RAD50  /CSN/
        .RAD50  /CSN/
        .RAD50  /CSN/
BUFF:   .BLKW   15.
DBUF:   .BLKW   400

        .END
```



```
;
;
;       C 1975,1976
;
;       ARTHUR I. KARSHMER
;       CENTER FOR SYSTEMS NEUROSCIENCE
;       GRADUATE RESEARCH CENTER
;       UNIVERSITY OF MASSACHUSETTS
;       AMHERST, MA.  01002
;
;
;
;
;       ROUTINE TO FIND LIGHT PEN ON A DARK SCREEN
;       CALLED FROM FORTRAN AS
;
;       CALL RADAR(Y-BOTTOM,Y-TOP,X-LEFT,X-RIGTH,INC,
;               D-FILE,X,Y,T-DLAY)
;
;       WHERE:
;               Y-BOTTOM IS Y START POSITION
;               Y-TOP    IS Y END POSITION
;               X-LEFT   IS LEFT END OF SCAN LINE
;               X-RIGHT  IS RIGHT END OF SCAN LINE
;                           LENGTH=XRIGHT - X-LEFT
;               INC      IS LINE COUNT INCREMENT
;               D-FILE   IS USER DISPLAY FILE NAME
;               X        IS RETURN X POSITION OF HIT
;               Y        IS RETURN Y POSITION OF HIT
;               T-DLAY   IS TIME DELAY IN TICKS
;
        .TITLE  SCAN
        .GLOBL  RADAR,MOVE,LVECT,NEXT,LPEN,NOECHO,ECHO,DRETN,CSNERR
        .GLOBL  ITTYIN,SNOOZE,TEXT
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
RADAR:  CMP     #8.,(R5)+               ;ENOUGH ARGS
        BLE     OK                      ;ITS GOOD
        JMP     NGARG                   ;ITS BAD
OK:     MOV     #1,LIST                 ;SET UP FOR SUBR CALL
        MOV     10.(R5),LIST+2          ;D-FILE ADDR
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #LIST,R5                ;ARG POINTER
        JSR     PC,NEXT                 ;GET NEXT POINTER
        MOV     RFIVE,R5                ;RESTORE POINTER
        DEC     R0                      ;DATA FOR DRETN CALL LATER
        MOV     R0,ENDPTR               ;SAVE IT

RUN:    MOV     #3,LIST                 ;SET UP FOR MOVE
        MOV     #XY,LIST+2              ;X COORD
        MOV     #XY+2,LIST+4            ;Y COORD
        MOV     10.(R5),LIST+6          ;DFILE
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #LIST,R5
        JSR     PC,MOVE                 ;MOVE THE BEAM
        MOV     RFIVE,R5                ;RESTORE THE POINTER
```

```
        MOV     #7.,LIST                ;ARGS FOR TEXT CALL
        MOV     #ONE,LIST+2             ;INTENSITY VALUE
        MOV     #ONE,LIST+4             ;BLINK
        MOV     #ZERO,LIST+6            ;NO LIGHT PEN
        MOV     #ZERO,LIST+8.           ;UPPER CASE
        MOV     #MESS2,LIST+10.         ;THE STRING
        MOV     10.(R5),LIST+12.        ;DFILE NAME
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #LIST,R5
        JSR     PC,TEXT                 ;PUT UP TEXT
        MOV     RFIVE,R5                ;RESTORE POINTER

        MOV     #1,LIST                 ;SET UP FOR NEXT CALL
        MOV     10.(R5),LIST+2          ;DFILE NAME
        MOV     R5,RFIVE
        MOV     #LIST,R5                ;ARG POINTER
        JSR     PC,NEXT                 ;GET NEXT WORD
        MOV     RFIVE,R5                ;RESTORE POINTER
        ADD     #1,R0                   ;Y COORD LOCATION
        MOV     R0,YPOS                 ;SAVE IT ALSO
        ASL     YPOS                    ;CHANGE TO BYTE COUNT

        MOV     #3,LIST                 ;SET UP FOR MOVE CALL
        MOV     4(R5),LIST+2            ;X COORD ADDR
        MOV     (R5),LIST+4             ;Y COORD ADDR
        MOV     10.(R5),LIST+6          ;D-FILE ADDR
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #LIST,R5                ;ARG POINTER
        JSR     PC,MOVE                 ;GO TO IT
        MOV     RFIVE,R5                ;RESTORE POINTER

TYPEIT: .PRINT  #MESS                   ;PRINT PROMPT MESSAGE

        MOV     #1,LIST                 ;SET UP FOR LPEN CALL
        MOV     #LIST+2,LIST+2          ;SET UP LPEN BUFFER ADDRESS
        MOV     #LIST,R5                ;ARG POINTER
        JSR     PC,LPEN
        MOV     RFIVE,R5

        JSR     PC,NOECHO               ;TURN OFFF ECHO

WAIT:   JSR     PC,ITTYIN               ;GET A CHAR
        TST     R0                      ;WAS THERE ONE
        BLT     WAIT                    ;GO BACK AND WAIT

        MOV     #8.,LIST                ;SET UP FOR LVECT CALL
        MOV     @6(R5),LEN              ;GET RIGHT END OF LINE
        SUB     @4(R5),LEN              ;SUB LEFT & GET LENGTH
        MOV     #LEN,LIST+2             ;SET DELTA-X
        MOV     #ZERO,LIST+4            ;SET DELTA-Y
        MOV     #ONE,LIST+6             ;INTEN
        MOV     #ZERO,LIST+8.           ;BLINK
        MOV     #ZERO,LIST+10.          ;LINE TYPE
        MOV     #ONE,LIST+12.           ;LIGHT PEN
        MOV     10.(R5),LIST+14.        ;D-FILE ADDR
        MOV     #DNAME,LIST+16.         ;DISPLAY NAME
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #LIST,R5                ;ARGUMENT POINTER
```



```
        JSR     PC,LVECT                ;DRAW THE LINE
        MOV     RFIVE,R5                ;RESTORE POINTER

        MOV     @(R5),R1                ;GET START Y POS
        MOV     @2(R5),R2               ;GET END Y POS
        MOV     @8.(R5),R3              ;GET INCREMENT
        MOV     10.(R5),R4              ;GET FILE ADDR
        ADD     YPOS,R4                 ;ADD IN OFFSET
        CLR     LIST+2                  ;ENABLE LIGHT PEN

LOOP:   CMP     R1,R2                   ;ARE WE DONE YET
        BGT     OUT                     ;YUP
        MOV     R1,(R4)                 ;UPDATE Y-POS WORD IN D-FILE

        MOV     R1,-(SP)                ;SAVE R1
        MOV     R5,RFIVE                ;SAVE POINTER
        MOV     #TLIST,R5               ;SET ARG POINTER
        CMP     #9.,-2(R5)              ;TIME DELAY GIVEN
        BNE     SLEEP                   ;NOPE
        MOV     @16.(R5),TLIST+2        ;GET USER TIME
        BR      NAP
SLEEP:  MOV     #2,TLIST+2              ;DEFAULT TIME - 2 TICKS
NAP:    JSR     PC,SNOOZE               ;TAKE A NAP
        MOV     RFIVE,R5                ;RESTORE POINTER
        MOV     (SP)+,R1                ;RESTORE R1

        TST     LIST+2                  ;LPEN HIT?
        BEQ     CONT                    ;NOPE
        CMP     #32767.,LIST+4          ;IS IT RIGHT NAME
        BEQ     HIT                     ;WE GO HIT
        CLR     LIST+2                  ;ENABLE LPEN
CONT:   ADD     R3,R1                   ;INCR COUNTER
        BR      LOOP                    ;BACK TO IT

OUT:    MOV     #2,LIST                 ;SET UP FOR DRETN
        MOV     10.(R5),LIST+2          ;D-FILE NAME
        MOV     #ENDPTR,LIST+4          ;END OF FILE WORD
        MOV     R5,RFIVE
        MOV     #LIST,R5
        JSR     PC,DRETN
        MOV     RFIVE,R5
        JMP     RUN

HIT:    MOV     LIST+12.,@12.(R5)       ;RETURN X COORD
        MOV     LIST+14.,@14.(R5)       ;RETURN Y COORD
        MOV     #2,LIST                 ;SET UP FOR DRETN
        MOV     10.(R5),LIST+2          ;D-FILE NAME
        MOV     #ENDPTR,LIST+4          ;END OF FILE WORD
        MOV     #LIST,R5
        JSR     PC,DRETN

        JSR     PC,ECHO

        RTS     PC                      ;ITS ALL OVER

NGARG:  MOV     #1,R1                   ;PRINT ERROR MESS
        JMP     CSNERR                          ;EXIT TO MONITR
```

```
        LIST:   .BLKW   9.
        ENDPTR: .WORD   0
        YPOS:   .WORD   0
        LEN.    .WORD   0
        ZERO:   .WORD   0
        ONE:    .WORD   1
        AFIVE:  .WORD   0
        DNAME:  .WORD   32767.
        TLIST:  .WORD   1,2
        XY:     .WORD   375.,0
        MESS:   .BYTE   7,7,200
                .EVEN
        MESS2:  .ASCII  /TYPE ANY KEY TO START SCAN;/
                .EVEN

                .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO ALLOW THE FORTRAN USER TO ADJUST
        ;       THE SIZE AND POSITION OF THE MONITOR SCROL
        ;       BUFFER FROM A FORTRAN PROGRAM. THIS ROUTINE
        ;       IS CALLED AS:
        ;
        ;       CALL SCROL(LINE-COUNT,INTENSITY,YPOS)
        ;
        ;       WHERE:
        ;
        ;               LINE-COUNT IS THE NUMBER OF LINES
        ;                       TO BE DISPLAYED ON THE SCREEN
        ;               INTENSITY IS THE INTENSITY OF THE SCROL
        ;                       BUFFER (1-8)
        ;               YPOS IS THE TOP POSITION OF THE SCROL
        ;                       BUFFER.
        .TITLE  SCBUF
        .GLOBL  SCROL,CSNERR
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF

SCROL:  CMP     #3,(R5)+                ;ENOUGH ARGS?
        BNE     NGARG                   ;BAD
        MOV     @(R5)+,R0               ;GET LINE-COUNT
        MOV     @(R5)+,R1               ;GET INTENSITY
        MOVB    R0,SCRBUF               ;MOVE INTO SCRBUF
        MOVB    R1,SCRBUF+1
        MOV     @(R5)+,YPOS             ;GET Y POSITION
        .SCROL  #SCRBUF                 ;LET IT RIP
        RTS     PC                      ;GO HOME

NGARG:  MOV     #1,R1                   ;SET UP FOR ERROR MESSAGE
        JMP     CSNERR                  ;LET IT FLY

SCRBUF: .WORD   0
YPOS:   .WORD   0

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO SUSPEND EXECUTION OF A JOB
        ;       FOR A CERTAIN NUMBER OF CLOCK TICKS.
        ;       EACH TICK IS 1/60 TH. OF A SECOND.
        ;       THE MAIXIMUM SLEEP TIME IS ABOUT
        ;       9 MINUTES OR 32767 TICKS.
        ;
        ;       CALL SNOOZE(TICKS)
        ;
        ;
        ;       TO GET JUST THE LOW ORDER TICKS
        ;
        ;       ITICKS()
        ;
        .TITLE  SLEEP
        .GLOBL  SNOOZE,ITICKS,CSNERR
        .MCALL  ..V2..,.REGDEF,.GTIM,.PRINT,.EXIT
        ..V2..
        .REGDEF
SNOOZE: CMP     #1,(R5)+                ;ENOUGH ARGS?
        BNE     NGARG                   ;BOO
        .GTIM   #LIST,#TIME             ;GET START TIME
        MOV     TIME+2,R1               ;GET LOW ORDER TICKS
        SUB     R1,TIME+2

LOOP:   CMP     TIME+2,@(R5)            ;CHECK FOR DONE YET
        BGE     OUT                     ;WERE DONE
        .GTIM   #LIST,#TIME             ;GET TIME
        SUB     R1,TIME+2
        BR      LOOP                    ;CHECK IT AGAIN

OUT:    RTS     PC                      ;TIMES UP

ITICKS: .GTIM   #LIST,#TIME             ;GET TIME BITS
        MOV     TIME,R0
        ADD     TIME+2,R0               ;GET LOW ORDER TICKS
        RTS     PC                      ;RETURN TO CALLER

NGARG:  MOV     #1,R1                   ;# OF ARGS
        JMP     CSNERR

LIST:   .BLKW   2
TIME:   .WORD   0,0

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO RETURN THE NUMBER OF
        ;       FREE WORDS IN EITHER A DISPLAY
        ;       OR SUB FILE TO THE FORTRAN USER
        ;
        ;       CALL ISPACE(NAME)
        ;
        ;       WHERE:
        ;
        ;               NAME IS EITHER A DISPLAY OR
        ;               SUB FILE NAME
        ;
        ;       IF NAME GIVEN IS NOT VALID, A NEGATIVE
        ;       INTEGER IS RETURNED
        ;
        .TITLE  SPACE
        .GLOBL  ISPACE,WHAT$,FILCK$,ACTVD$,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT
        ..V2..
        .REGDEF
ISPACE: TST     (R5)+                   ;DID WE GET A NAME
        BEQ     NGARG
1$:     MOV     (R5)+,R0                ;USER GIVEN NAME
2$:     JSR     PC,WHAT$                ;WHAT IS IT
        TST     R1                      ;WAS IT VALID
        BEQ     3$                      ;BOO
        MOV     SIZE$(R0),R1            ;GET UPPER BOUND
        SUB     FREE$(R0),R1            ;SUBTRACT CURENT PTR
        MOV     R1,R0                   ;FUNCTION RETURN
        ASR     R0
        RTS     PC
3$:     MOV     #4,R1                   ;REPORT BAD NAME
        JMP     CSNERR

NGARG:  MOV     #1,R1                   ;NOT ENOUGH ARGS
        JMP     CSNERR

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO SET USR NOSWAP AND THEN TO
        ;       R SET USR SWAPPING FROM FORTRAN
        ;
        ;
        ;       CALL NOSWAP
        ;       CALL SWAP
        ;
        ;
        .TITLE  SWAPS
        .GLOBL  NOSWAP,SWAP
        .MCALL  ..V2..,.REGDEF,.LOCK,.UNLOCK
        ..V2..
        .REGDEF
NOSWAP: .LOCK                           ;LOCK THE USR
        RTS     PC                      ;GO BACK

SWAP:   .UNLOCK                         ;UNLOCK THE USR
        RTS     PC                      ;BYE - BYE

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO READ THE CONTENTS OF THE
        ;       SWITCH REGISTER AND EITHER RETURN
        ;       THE CONTENTS OR COMPARE THEM AGAINST
        ;       A VALUE PASSED FROM THE FORTRAN PROGRAM
        ;
        ;       ISWTCH(ARG)
        ;       ISWTCH()
        ;
        ;       IF THE ISWTCH(ARG) FORM IS USED,
        ;       THE VALUE RETURNED WILL BE
        ;
        ;       1       IF THERE WAS A MATCH
        ;       0       IF THERE WASN'T A MATCH
        ;
        ;
        ;       ROUTINES TO RETURN THE ABSOULUTE
        ;       MEMORY ADDRESS OF A VARIABLE AND
        ;       ALSO TO LOOK AT ITS CONTENTS
        ;
        ;       IADDR(VAR-NAME)
        ;       LOOK(MEMORY ADDR,WORD/BYTE)
        ;
        ;       WHERE:
        ;               0=BYTE
        ;               NOT 0 = WORD
        .TITLE  SWITCH
        .GLOBL  ISWTCH,IADDR,LOOK,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
        SWREG=177570
ISWTCH: CMP     (R5)+,#1        ;DID WE GET ANY ARGS
        BNE     NOPE            ;NO JUST TRANSFER CONTENTS
        CMP     @(R5)+,@#SWREG  ;DOES IT MATCH
        BNE     BAD
        MOV     #1,R0           ;IT MATCHES
        RTS     PC
BAD:    CLR     R0
        RTS     PC
NOPE:   MOV     @#SWREG,R0      ;RETURN THE CONTENTS
        RTS     PC

IADDR:  TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG
        MOV     (R5)+,R0        ;GET ADDRESS
        RTS     PC
```

```
LOOK:   TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG           ;NOPE
        CLR     R0
        MOV     @(R5)+,R1       ;GET ADDRESS TO BE LOOKED AT
        BIT     #1,R1           ;IS IT BYTE ADDRESS
        BNE     BYTE            ;YUP - GET THE BYTE
        TST     @(R5)+          ;IS IT EVEN WORD BYTE REQUEST
        BEQ     BYTE
        MOV     (R1),R0         ;LOOK AT IT
        RTS     PC
BYTE:   MOVB    (R1),R0         ;MOVE THAT BYTE
        RTS     PC

NGARG:  MOV     #1,R1           ;# OF ARGS
        JMP     CSNERR

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO INPUT CHARACTERS FROM FORTRAN
        ;       EITHER ONE AT A TIME OR LINE AT A TIME
        ;
        ;       ITTYIN()
        ;
        ;       RETURNS WITH R0=-2 IF NO CHARACTERS
        ;       WERE AVAILABLE
        ;
        ;       ELSE THE CHAR IS RETURNED
        ;
        ;       ROUTINE TO TYPE A CHARACTER AT A TIME
        ;       TO THE CONSOLE
        ;       CALL TTOUT(CHAR)
        ;
        .TITLE  TTYIN ROUTINE
        .GLOBL  ITTYIN,TTOUT,CSNERR
        .MCALL  ..V2..,.REGDEF
        .MCALL  .TTINR,.TTYOUT,.PRINT,.EXIT
        ..V2..
        .REGDEF
ITTYIN: TST     (R5)
        .TTINR                  ;ANY CHARS
        BCS     NONE            ;NOPE
        BIC     #177600,R0      ;STRIP OFF HIGH ORDER BITS
        RTS     PC              ;RETURN

NONE:   MOV     #-2,R0          ;SET FLAG
        RTS     PC              ;AND RETURN


TTOUT:  TST     (R5)+           ;ENOUGH ARGS
        BEQ     NGARG           ;BOO
        .TTYOUT @(R5)+          ;SHIP OUT THE CHAR
        RTS     PC              ;GO BACK

NGARG:  MOV     #1,R1           ;# OF ARGS
        JMP     CSNERR

        .END
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE TO DRAW A VECTOR IN A GIVEN
        ;       DISPLAY OR SUBROUTINE FILE FROM A
        ;       FORTRAN PROGRAM.
        ;
        ;       IF THE DELTA X AND DELTA Y VALUES
        ;       ARE IN THE PROPER RANGE, -65<VAL<65,
        ;       A SHORT VECTOR IS DRAWN, OTHERWISE
        ;       A LONG VECTOR IS DRAWN.
        ;
        ;       CALL VECT(X,Y,INTEN,BLINK,LINE,LPEN,FILE,[DNAME])
        ;
        .TITLE  VECTOR
        .GLOBL  VECT,LVECT,WHAT$,CSNERR,INTEN$
        .GLOBL  BLINK$,LPEN$,LNTYP$,BITS$,RDOT$
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
VECT:   CMP     @2(R5),#63.     ;ACTUALLY DRAW A SHORT VECTOR
        BGT     OUTRNG
        CMP     @2(R5),#-63.
        BLT     OUTRNG
        CMP     @4(R5),#63.
        BGT     OUTRNG
        CMP     @4(R5),#-63.
        BLT     OUTRNG
        BR      OUT

OUTRNG: JSR     PC,LVECT        ;GO DRAW A LONG VECTOR
        RTS     PC              ;GO BACK

OUT:    CMP     #7,(R5)+        ;ENOUGH ARGS
        BHI     NGARG
        CLR     R4
        MOV     @(R5),-(SP)     ;SAVE DELTA X
        MOV     @(R5)+,R0       ;GET DELTA X
        BGE     PLUSX           ;IT IS POSITIVE
        NEG     R0              ;IT'S NEG SO CHANGE THE SIGN
        BIS     #20000,R4       ;SET - BIT
PLUSX:  BIS     #40000,R4       ;SET INTX BIT
        ASL     R0              ;SHIFT LEFT 7 PLACES
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
        ASL     R0
```



```
        BIS     R0,R4           ;SET DELTA X
        MOV     @(R5),-(SP)     ;SAVE DELTA Y
        MOV     @(R5)+,R0       ;GET DELTA Y
        BGE     PLUSY
        NEG     R0              ;REVERSE THE SIGN
        BIS     #100,R4         ;SET - BIT
PLUSY:  BIS     R0,R4           ;SET DELTA Y
        MOV     #SHORTV,R2      ;SET UP SHORTV INSTR
        JSR     PC,BITS$        ;GO SET OTHER BITS
        MOV     R4,-(SP)        ;SAVE DATA WORD
        MOV     R2,-(SP)        ;SAVE SHORTV WORD
RDOT$:  MOV     (R5)+,R0        ;GET FILE NAME
        JSR     PC,WHAT$        ;CHECK IT OUT
        TST     R1
        BLT     DFILE
        BEQ     NGNAME
        BR      SFILE
DFILE:  MOV     FREE$(R0),R1    ;GET FREE PTR
        TST     ONOFF$(R0)      ;ARE THE LITES ON
        BNE     1$
        .STOP
1$:     TST     @-4(R5)         ;IS LPEN ON?
        BEQ     10$             ;NOPE
        MOV     #DNAME,(R1)+    ;SET UP DNAME INSTR
        MOV     @(R5),(R1)+     ;GET USER DNAME
10$:    JSR     PC,PUSH         ;MOVE INTRS INTO PLACE
        TST     ONOFF$(R0)
        BNE     2$
        .START
2$:     TST     (SP)+           ;POP STACK
        TST     (SP)+
        RTS     PC

PUSH:   MOV     (SP)+,R5        ;SAVE RETURN ADDR
        MOV     (SP),(R1)+      ;SET IN SHORTV/RELPNT INSTR
        MOV     (SP)+,OLDX$(R0) ;SET OLD INSTR WORD IN STRUCTURE
        MOV     (SP)+,(R1)+     ;SET DATA WORD
        MOV     #DRET,(R1)
        CLR     2(R1)
        MOV     R1,FREE$(R0)    ;RESET FREE PTR
        MOV     R5,-(SP)
        RTS     PC

SFILE:  MOV     FREE$(R0),R1    ;GET FREE PTR
        MOV     R1,R4           ;SAVE IT FOR LATER USE
        TST     OPEN$(R0)       ;IS A DEF OPEN
        BNE     NGCLSD
        TST     @-4(R5)         ;IS LP ON?
        BEQ     5$              ;NOPE
        MOV     #DNAME,(R1)+
        MOV     @(R5),(R1)+     ;GET USER DNAME
5$:     JSR     PC,PUSH         ;LAY IN THE INSTRS
        MOV     @OPNSR$(R0),R1  ;GET END PTR
        CMP     R1,R4
        BEQ     10$
        .STOP
        MOV     #DJMP,(R1)+     ;JUMP OVER OTHER JUNK
        MOV     R4,(R1)         ;WHERE TO JUMP
```

```
        .START                          ;FLIP THAT SWITCH
10$:    MOV     FREE$(R0),@OPNSR$(R0)   ;SET NEW END PTR
        MOV     OPNSR$(R0),R1   ;GET OLD X & Y
        ADD     #4,R1
        ADD     (SP)+,(R1)+     ;SAVE DELTA Y
        ADD     (SP)+,-(R1)     ;SAVE DELTA X
        RTS     PC

NGCLSD: MOV     #2,R1           ;OPEN/CLOSED
        JMP     CSNERR

NGNAME: MOV     #4,R1           ;BAD NAME
        JMP     CSNERR

NGARG:  MOV     #1,R1           ;NOT ENOUGH ARGS
        JMP     CSNERR

        .END
```



```
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        .NLIST
        .TITLE  VTMAC
;
; VTMAC
;
; LIBRARY OF MACRO CALLS AND MNEMONIC DEFINITIONS
; FOR THE VT11 DEVICE SUPPORT PACKAGE
;
; DEC-11-OVTMA-C
;
; 24 OCTOBER 73
;  4 MAY 76  <CSN>
;
; VTMAC IS A LIBRARY OF MACRO CALLS WHICH PROVIDE SUPPORT
; OF THE VT11 DISPLAY PROCESSOR. THE MACROS PRODUCE CALLS
; TO THE VT11 DEVICE SUPPORT PACKAGE, USING GLOBAL REFER-
; ENCES.
;
; SPECIAL ADDITIONS HAVE BEEN MADE TO THE VTMAC LIBRARY
; FOR USE WITH THE CSN FORTRAN GRAPHICS SUPPORT SYSTEM
;
; MACRO TO GENERATE A MACRO WITH ZERO ARGUMENTS.
;
```

```
        .MACRO MAC0 NAME,CALL
        .MACRO NAME
        .GLOBL  CALL
        JSR     %^O7,CALL
        .ENDM
        .ENDM
;
; MACRO TO GENERATE A MACRO WITH ONE ARGUMENT
;
        .MACRO MAC1 NAME,CALL
        .MACRO NAME ARG
         IF NB,ARG
        MOV     ARG,%^O0
        .ENDC
        .GLOBL  CALL
        JSR     %^O7,CALL
        .ENDM
        .ENDM
;
; MACRO TO GENERATE A MACRO WITH TWO OPTIONAL ARGUMENTS
;
        .MACRO MAC2 NAME,CALL
        .MACRO NAME ARG1,ARG2
        .GLOBL  CALL
        .IF NB,ARG1
        MOV     ARG1,%^O0
        .ENDC
        .IF NB,ARG2
        MOV     ARG2,-(%^O6)
        .IFF
        CLR     -(%^O6)
        .NARG   T
        .IF EQ,T
        CLR     %^O0
        .ENDC
        .ENDC
        JSR     %^O7,CALL
        .ENDM
        .ENDM
```



```
;
; MACRO LIBRARY FOR VT11:
;
        MAC0    <.CLEAR>,<$VINIT>
        MAC0    <.STOP>,<$VSTOP>
        MAC0    <.START>,<$VSTRT>
        MAC0    <.SYNC>,<$SYNC>
        MAC0    <.NOSYN>,<$NOSYN>
        MAC1    <.STAT>,<$VSTPM>
        MAC1    <.NAME>,<$NAME>
        MAC1    <.INSRT>,<$VNSRT>
        MAC1    <.REMOV>,<$VRMOV>
        MAC1    <.BLANK>,<$VBLNK>
        MAC1    <.RESTR>,<$VRSTR>
        MAC1    <.STAT>,<$VSTPM>
        MAC1    <.LPEN>,<$VLPEN>
        MAC1    <.SCROL>,<$VSCRL>
        MAC2    <.TRACK>,<$VTRAK>
        MAC0    <.LNKRT>,<$VRTLK>
        MAC0    <.UNLNK>,<$VUNLK>
```

```
;
; MNEMONIC DEFINITIONS FOR THE VT11 DISPLAY PROCESSOR
;
DPC=172000      ;DISPLAY PROG COUNTER
DSR=172002      ;DISPLAY STAT REG
XSR=172004      ;X STAT REG
YSR=172006      ;Y STAT REG
DJMP=160000     ;DISPLAY JUMP
DNOP=164000     ;DISPLAY NOP
DJSR=173400     ;DISPLAY SUBROUTINE CALL
DRET=173400     ;DISPLAY SUBROUTINE RETURN
DNAME=173520    ;SET NAME REGISTER
DSTAT=173420    ;RETURN STATUS DATA
DHALT=173500    ;STOP DISPLAY AND RETURN STATUS DATA
STVECT=320      ;STOP VECTOR
LPVECT=324      ;LIGHT PEN VECTOR
TOVECT=330      ;SHIFT/TIME OUT VECTOR
.
CHAR=100000     ;CHARACTER MODE
SHORTV=104000    SHORT VECTOR MODE
SHIFT=200       ;LEFT BYTE SHIFT COUNT


LONG =110000    ;LONG VECTOR MODE

POINT=114000    ;POINT MODE
GRAPHX=120000   ;GRAPH X MODE
GRAPHY=124000   ;GRAPH Y MODE
RELATV=130000   ;RELATIVE VECTOR MODE
.
INT0=2000       ;INTENSITY 0
INT1=2200
INT2=2400
INT3=2600
INT4=3000
INT5=3200
INT6=3400
INT7=3600       ;INTENSITY 7
;
LPOFF=100       ;LIGHT PEN OFF
LPON=140        ;LIGHT PEN ON
BLKOFF=20       ;BLINK OFF
BLKON=30        ;BLINK ON
LINE0=4         ;SOLID LINE
LINE1=5         ;LONG DASH
LINE2=6         ;SHORT DASH
LINE3=7         ;DOT DASH
;
STATSA=170000   ;LOAD STATUS REG A
LPLITE=200      ;INTENSIFY ON LPEN HIT
LPDARK=300      ;DON'T INTENSIFY
ITAL0=40        ;ITALICS OFF
ITAL1=60        ;ITALICS ON
SYNC=4          ;POWER LINE SYNC
.
STATSB=174000   ;LOAD STATUS REG B
INCF=100        ;GRAPH PLOT INCREMENT
INTV=40000      ;INTENSIFY VECTOR OR POINT
```



```
MAXX=1777       ;MAXIMUM X INCR.  - LONGV
MAXY=1377       ;MAXIMUM Y INCR.  - LONGV
MINUS=20000     ;MINUS X OR Y INCREMENT
MINUSX=20000
MINUSY=20000
MAXSX=17600     ;MAXIMUM X INCR   - SHORTV
MAXSY=77        ;MAXIMUM Y INCR.  - SHORTV
MISVX=20000     ;NEGATIVE X INCR. - SHORTV
MISVY=100       ;NEGATIVE Y INCR. - SHORTV
;
;       SPECIAL DEFINITIONS FOR CSN GRAPHICS
;
HEAD$=0         ;LIST HEAD
FREE$=2         ;FREE POINTER POS.
SIZE$=4         ;SIZE WORD
OPEN$=6         ;OPEN/CLOSED FLAG
ONOFF$=6        ;ON/OFF FLAG
OPNSR$=10.      ;CURRENTLY ACVIVE SUBROUTINE
OLDX$=8.        ;OLD INSTR
OLDY$=10.
LDPNT$=12.      ;INITIAL LOAD POINT
FIRST$=14.      ;FIRST USABLE WORD
        .LIST
```

```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;       ROUTINE USED BY OTHER CSNLIB ROUTINES TO
        ;       DETERMINE WHAT TYPE OF DATA STRUCTURE
        ;       IS BEING USED - A DISPLAY OR A SUBROUTINE
        ;       FILE.
        ;
        ;       ON INPUT:
        ;
        ;               (R0)=FILE NAME/ADDRESS
        ;
        ;       ON OUTPUT:
        ;
        ;               (R1)=   -1 IF A DISPLAY FILE
        ;                        0 IF NEITHER DISPLAY OR SUB FILE
        ;                        1 IF SUB FILE
        ;
        .TITLE  WHAT
        .GLOBL  WHAT$,FILCK$
        .MCALL  ..V2..,.REGDEF
        ..V2..
        .REGDEF
WHAT$:  MOV     #1,R1           ;SET UP FOR S-FILE TEST
        JSR     PC,FILCK$       ;CHECK IT OUT
        TST     R1              ;WAS IT OK?
        BEQ     DFTST           ;NOT SUB-FILE KEEP CHECKING
        RTS     PC              ;IT IS SUB-FILE - GO BACK

DFTST:  CLR     R1              ;SET UP FOR D-FILE TEST
        JSR     PC,FILCK$       ;CHECK IT OUT
        TST     R1              ;IS IT A D-FILE
        BEQ     BADNAM          ;IT'S NO GOOD
        MOV     #-1,R1          ;IT IS A D-FILE
BADNAM: RTS     PC              ;THAT'S ALL FOLKS

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO WRITE A TEXT STRING FROM
        ;       A FORTRAN PROGRAM. CALLED AS
        ;
        ;       CALL WRITE(STRING,FILE [,INTEN,BLINK,LPEN,
        ;                  CASE [,DNAME]])
        ;
        .TITLE  WRTEXT
        .GLOBL  WRITE,CSNERR,TEXT
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
WRITE:  MOV     (R5)+,R0        ;SAVE ARG COUNT
        CMP     R0,#2           ;ENOUGH ARGS?
        BLT     NGARG           ;NOPE
        MOV     (R5)+,STRING    ;MOVE STRING ADDR
        MOV     (R5)+,FILE      ;MOVE FILE ADDR
        CMP     R0,#2           ;CHANGE PARAMS?
        BEQ     NOPE            ;NOPE
        MOV     @(R5)+,INTEN    ;GET INTENSITY
        MOV     @(R5)+,BLINK    ;GET BLINK
        MOV     @(R5)+,LPEN     ;GET LPEN
        MOV     @(R5)+,CASE     ;GET CASE
        TST     LPEN            ;ANY DNAME?
        BLE     NOPE            ;NOPE
        MOV     @(R5)+,DNAME    ;GET DISPLAY NAME

NOPE:   MOV     #LIST,R5        ;ADDR OF ARG LIST
        JMP     TEXT            ;GO TO IT

NGARG:  MOV     #1,R1
        JMP     CSNERR

LIST:   .WORD   7               ;# OF ARGS PASSED
        .WORD   INTEN           ;ADDR OF INTENSITY
        .WORD   BLINK           ;ADDR OF BLINK
        .WORD   LPEN            ;ADDR OF LPEN
        .WORD   CASE            ;ADDR OF CASE
STRING: .WORD   0               ;ADDR OF CHAR STRING
FILE:   .WORD   0               ;ADDR OF FILE
        .WORD   DNAME           ;ADDR OF DISPLAY NAME

INTEN:  .WORD   2               ;DEFAULT INTEN VALUE
BLINK:  .WORD   0               ;DEFAULT NO-BLINK
LPEN:   .WORD   0               ;DEFAULT NO-LPEN
CASE:   .WORD   0               ;DEFAULT UPPER CASE
```

```
DNAME:  .WORD   0               ;DEFAULT NO-DNAME

        .END
```



```
        ;
        ;
        ;       C 1975,1976
        ;
        ;       ARTHUR I. KARSHMER
        ;       CENTER FOR SYSTEMS NEUROSCIENCE
        ;       GRADUATE RESEARCH CENTER
        ;       UNIVERSITY OF MASSACHUSETTS
        ;       AMHERST, MA.  01002
        ;
        ;
        ;
        ;
        ;       ROUTINE TO ZERO OUT ARRAYS FROM FORTRAN
        ;       PROGRAMS- CALLED AS
        ;
        ;       CALL RZERO(REAL-ARRAY,LENGTH)
        ;       CALL IZERO(INTEGER-ARRAY,LENGTH)
        ;       CALL LZERO(LOGICAL-ARRAY,LENGTH)
        ;
        ;
        .TITLE  ZERO
        .GLOBL  RZERO,IZERO,LZERO,CSNERR
        .MCALL  ..V2..,.REGDEF,.PRINT,.EXIT
        ..V2..
        .REGDEF
RZERO:  MOV     #MUL4,-(SP)     ;ADDRESS OF MULT BY 4 ROUTINE
        BR      NEXT            ;GO TO IT

IZERO:  MOV     #MUL2,-(SP)     ;ADDRESS OF MULT BY 2 ROUTINE
        BR      NEXT

LZERO:  MOV     #MUL0,-(SP)     ;NO MULTIPLY

NEXT:   CMP     #2,(R5)+        ;DO WE HAVE ENOUGH ARGS
        BNE     NGARG           ;NO GOOD
        MOV     (R5)+,R0        ;GET ADDRESS OF ARRAY
        MOV     @(R5)+,R1       ;GET UNIT COUNT
        JMP     @(SP)+          ;JUMP TO PROPER ROUTINE

MUL4:   ASL     R1              ;MULTIPLY BY 2
MUL2:   ASL     R1              ;AND ONE MORE TIME
MUL0:   ADD     R0,R1           ;ADD BASE TO OFFSET FOR COUNTER


        DEC     R1              ;SET FOR ZERO BASE
LOOP:   CMP     R0,R1           ;ARE WE DONE YET
        BGT     OUT             ;YES IF R0>R1
        CLRB    (R0)+           ;ZERO A BYTE
        BR      LOOP

OUT:    RTS     PC              ;RETURN HOME

NGARG:  TST     (SP)+           ;POP OFF INDIRECT JUMP LOCATION
        MOV     #1,R1           ;PRINT ERROR MESSAGE
        JMP     CSNERR

        .END
```

122

XIX. Programs for Communicating with the CYBER System

By Stuart P. Sims

```
        .TITLE  DL-11 INPUT HANDLER V02-01 UPDATE 27
;
;
;
;
;-------NOTE----SYSTEM NAME FOR THIS HANDLER IS DI.SYS
;
;
;
;
;     ADAPTED (MOSTLY STOLEN) FROM DEC'S PAPERTAPE READER
;HANDLER.  TO KEEP IN LINE WITH STANDARD PROGRAMMING
;CONVENTIONS I ASSUME NO RESPONSIBILTY FOR WHATEVER MAY
;HAPPEN AS A RESULT OF USING THIS HANDLER.
;
;          U S E  A T  Y O U R  O W N  R I S K ! !
;
;
;
;
;
;
;DL-11 INPUT CONTROL REGISTER DEFINITIONS
;SIMILAR TO TKS AND TKB
;--------------------------------------
        DISR=175610             ;STATUS & CONTROL REGISTER
        DIDB=175612             ;DATA BUFFER REGISTER
        DIVEC=300               ;DL-11E RECIEVER INTERRUPT VECTOR
;*************************************
;
;
;CONSTANTS FOR MONITER COMMUNTICATION
;-----------------------------------
        HDERR=1                 ;MASK FOR HARD ERROR
        MONLOW=54               ;POINTER TO BEGINNING OF RMON
        OFFSET=270              ;OFFSET TO POINTER TO QUE MANAGER
        PS=177776               ;PROCESSOR STATUS WORD
        PR4=200                 ;PRIOIRTY 4
        PR7=340                 ;PRIORITY 7
        DINTN=140               ;INTERRUPT MASK
;*************************************
;
;
        .MCALL  .REGDEF
        .REGDEF                 ;DEFINE REGISTERS
;
```

```
;**********************************
;       THIS SECTION IS THE INTERFACE WITH THE RT-11
;'SET' PROCESSOR.  WHEN THE 'SET DI LF' COMMAND IS GIVEN
;RT-11 CALLS THE HANDLER TO MEMORY AND EXECUTES THE CODE
;STARTING AT OPLF.
;                  ***W A R N I N G***
;DO NOT ATTEMPT TO OPTIMIZE THIS CODE UNLESS
;YOU KNOW EXACTLY WHAT YOU ARE DOING.  I DON'T REALLY
;UNDERSTAND THE SET PROCESSOR MYSELF BUT I DO KNOW THAT
;THIS CODE WORKS!
        .ASECT
        .=400
        .WORD   1               ;FOR NO LF SET LFFALG TO 1
        .RAD50  /LF    /
        .WORD   <OPLF-400>/2+100000
        0
OPLF:   MOV     #0,R3           ;FOR LF SET LFFLAG TO 0
        MOV     R3,LFFLAG
        RTS     PC              ;RETURN TO WHERE-EVER
;
;
```

```
;*****************************************************
;
;
        .CSECT  DLIN
;
;               ---L O A D   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       BEGINNING OF RESIDENT HANDLER.
;       WHEN THE HANDLER IS 'FETCHED' THE VECTOR ADDRESS
;       AND INITIAL INTERRUPT PRIORITY ARE TAKEN FROM
;       THIS SECTION.  (INITIAL PRIORITY IS 7 AND IS LOWERED
;       TO PRIORITY 4 AFTER THE FIRST INTERRUPT)
LOADPT: .WORD   DIVEC           ;ADDR OF INTERRUPT VECTOR
        .WORD   DINT-.          ;OFFSET TO INTERRUPT ENTRY
        .WORD   PR7             ;PRIORITY 7
DILQE:  .WORD   0               ;POINTER TO LAST QUE ENTRY
DICQE:  .WORD   0               ;POINTER TO CURRENT QUE ENTRY
;
;
;               ---E N T R Y   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       INITIAL ENTRY POINT
;       RT-11 QUE MANAGER JUMPS TO THIS SECTION OF CODE
;       AFTER SETTING UP POINTERS TO THE QUE ENTRIES IN
;       'DILQE' AND 'DICQE'.  THE INTERRUPTS ARE TURNED ON
;       HERE AND THE WORD COUNT CHANGED TO A CHARACTER COUNT
;       (IE. CHAR COUNT=WORD COUNT*2.).  A REQUEST FOR ZERO
;       CHARACTERS IS TREATED AS A SEEK AND DOES NOTHING
;       EXCEPT VERIFY THAT THE HANDLER IS IN CORE.
DI:     MOV     DICQE,R4        ;POINTER TO Q ENTRY INTO R4
        ASL     6(R4)           ;WORD COUNT TO BYTE COUNT
        BCS     DIERR           ;A WRITE REQUEST IS ILLEGAL
        BEQ     SEEK            ;REQUEST FOE 0 BYTES IS A SEEK
        MOV     #DINTN,@#DISR   ;ENABLE INT. GET A CHAR.
        RTS     PC              ;IT'S BEEN A LONG LONELY SUMMER
;
;
;               ---A B O R T   E N T R Y   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       ENTRY POINT FOR ABORTING OPERATION IN PROGRESS.
;       USED WHEN CNTRL C IS STRUCK AND WHENEVER CALLING
;       PROGRAM IS ABORTED BY THE SYSTEM.
        BR      DIABRT          ;ABORTION ENTRY FOR F/B
;
;
;               ---I N T E R R U P T   S E R V I C E---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
DINT:   MOV     @#MONLOW,-(SP)  ;INTO SYSTEM STATE
        JSR     R5,@(SP)+       ;RETURN AT LEVEL 4
        .WORD   ^C<200>&340     ;NO SCREW UPS HERE!
        MOV     DICQE,R4        ;R4 POINTS TO QUE ENTRY
        ADD     #4,R4           ;POINT R4 TO BUFFER ADDR
        TST     @#DISR          ;ANY ERRORS??
        BMI     DIEO3           ;YES--TREAT AS END OF FILE!
        MOVB    @#DIDB,@(R4)    ;PUT CHAR IN BUFFER
        BEQ     2$              ;WAS A NULL SO FORGET IT.
        BICB    #200,@(R4)      ;STRIP PARITY BIT
        TST     LFFLAG          ;CHECK OPTION VARIABLE
```

```
        BNE     1$              ;NO LF CHECK!
        CMPB    @(R4),#12       ;IS IT A LF?
        BEQ     DIEO2           ;YES--COMPLETE...!
1$:     INC     (R4)+           ;INC BUFFER POINTER
        DEC     @R4             ;DECREASE BYTE COUNT
        BEQ     DIDONE          ;IF ZERO; WE'RE DONE
2$:     RTS     PC              ;SO LONG FOR NOW

;               ---E N D   O F   F I L E---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       EOF IS DETECTED WHEN THE CARRIER STATUS BIT IN THE DISR
;       CHANGES STATE.  THE REST OF THE BUFFER IS CLEARED AND
;       THE LAST CHARACTER RECIEVED SHOULD BE VALID.
DIEO3:  BIS     #20000,@-6(R4)  ;SET EOF IN CHANNEL STATUS WORD
        BR      DIEOF

;               ---B U F F E R   C L E A R---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       REST OF BUFFER IS CLEARED WHENEVER AN OPERATION COMPLETES
;       WITHOUT COMPLETELY FILLING THE CHARACTER COUNT REQUESTED.
;       IE.  WHENEVER EOF OCCURS OR, LINE FEED OCCURS AND THE LINE
;       FEED FLAG IS CLEAR.
DIEO1:  INC     (R4)            ;DONT KILL ANY CHARS!
DIEOF:  CLRB    @(R4)           ;CLEAR REMAINDER OF BUFFER
DIEO2:  DEC     2(R4)           ;DECREMENT BYTE COUNT
        BNE     DIEO1           ;MORE?



;       ---O P E R A T I O N   C O M P L E T E---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       RETURNS CONTROL TO RT-11 QUE MANAGER
SEEK:
DIABRT:
DIDONE: BIC     #DINTN,@#DISR   ;TURN OFF DL11 READER INTERRUPT
                                ;IN CASE WE GET AN ERROR LATER
        MOV     PC,R4
        ADD     #DICQE-.,R4     ;GET ADDR OF CURRENT QUE ENTRY
        MOV     @#MONLOW,R5
        JMP     @OFFSET(R5)     ;TO MONITER COMPLETION>>>>

;               ---H A R D   E R R O R---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       HARD ERROR BIT IS SET IN THE $CSW WHENEVER A NON-
;       RECOVERABLE HARDWARE ERROR IS DETECTED.
;       IE.  WHEN SOME TYPE OF ERROR OCCURS THAT WOULD INDICATE
;       THAT THE CHARACTER JUST RECIEVED IS INVALID.  THE
;       INVALID CHARACTER IS NOT PASSED TO THE CALLING PROGRAM
;       AND THE REST OF THE BUFFER IS NOT CLEARED.
;       FRAMING AND PARITY ERRORS CAUSE THIS CONDITION.
;       OVERRUN ERRORS ARE NOT CHECKED!!!
DIHERR: MOV     #DICQE,R4       ;PUT CORRECT ADDRESS IN R4
                                ;NOW IS OK TO SET ERROR FLAG
DIERR:  BIS     #HDERR,@-(R4)   ;SET HARD ERROR BIT
        BR      DIDONE          ;AND COMPLETE OPERATION
```

```
LFFLAG: .WORD   0          ;FLAG FOR SET PROCESSOR
$OTPTR: .WORD   0          ;USED BY RT-11 MONITER

DISIZE=.-LOADPT

.END
```

```
        .TITLE  DLOUT   V02-01  UPDATE 6
;
;           RT-11 DL-11 OUTPUT HANDLER
;BASED ON DEC'S PAPER PUNCH HANDLER V02-01 5/1/74
;DEC AND MYSELF (AS USUAL) CLAIM NO RESPONSIBILTY
;FOR WHATEVER MAY HAPPEN UPON USE OF THIS HANDLER
;
;
;                GOOD LUCK!
;
;
;------NOTE----THE SYSTEM NAME FOR THIS HANDLER IS DO.SYS
;
;
;
;
;
;
;DL-11 OUTPUT CONTROL REGISTER DEFINITIONS
;SIMILAR TO TPS AND TPB
;---------------------------------------
        DOSR=175614             ;STATUS & CONTROL REGISTER
        DODB=175616             ;DATA BUFFER REGISTER
        DOVEC=304               ;DL-11E TRANSMITTER INTERRUPT VECTOR
;***************************************
;
;
;
; CONSTANTS FOR MONITER COMMUNICATION
;---------------------------------------
        HDERR=1                 ;MASK FOR HARD ERROR
        MONLOW=54               ;POINTER TO BEGINNING OF RMON
        OFFSET=270              ;OFFSET TO POINTER TO QUE MANAGER
        PS=177776               ;PROCESSOR STATUS WORD
        PR4=200                 ;PRIORITY 4
        PR7=340                 ;PRIORITY 7
;***************************************
;
;
        .MCALL  .REGDEF
        .REGDEF                 ;DEFINE REGISTERS
;
;
;***************************************
```

```
;
        .CSECT  DLOUTA
;
;               ---L O A D   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       BEGINNING OF RESIDENT HANDLER.
;       WHEN THE HANDLER IS 'FETCHED' THE VECTOR ADDRESS
;       AND INITIAL INTERRUPT PRIORITY ARE TAKEN FROM
;       THIS SECTION.  (INITIAL PRIORITY IS 7 AND IS LOWERED
;       TO PRIORITY 4 AFTER THE FIRST INTERRUPT)
LOADPT: .WORD   DOVEC           ;ADDR OF INT VECTOR
        .WORD   DOINT-          ;OFFSET TO INT SERVICE
        .WORD   PR7             ;PRIORITY 7
DOLQE:  .WORD   0               ;POINTER TO LAST QUE ENTRY
DOCQE:  .WORD   0               ;POINTER TO CURRENT QUE ENTRY
;
;
;               ---E N T R Y   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       INITIAL ENTRY POINT
;       RT-11 QUE MANAGER JUMPS TO THIS SECTION OF CODE
;       AFTER SETTING UP POINTERS TO THE QUE ENTRIES IN
;       'DOLQE' AND 'DOCQE'.  THE INTERRUPTS ARE TURNED ON
;       HERE AND THE WORD COUNT CHANGED TO A CHARACTER COUNT
;       (IE. CHAR COUNT=WORD COUNT*2.).  A REQUEST TO WRITE
;       ZERO CHARACTERS WILL RESULT IN THE FIRST INTERRUPT
;       BEING PROCESSED AND NO CHARACTERS TRANSMITTED.
DO:     MOV     DOCQE,R4        ;R3 POINTS TO CURRENT Q ENTRY
        ASL     6(R4)           ;WORD CONT TO BYTE COUNT
        BCC     DOERR           ;A READ REQUEST IS ILLEGAL
        BIS     #100,@#DOSR     ;CAUSES INT. STARTING TRANSFER
        RTS     PC              ;BYE BYE, SO LONG, GOODBYE
;
;
;
;               ---A B O R T   E N T R Y   P O I N T---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       ENTRY POINT FOR ABORTING OPERATION IN PROGRESS.
;       USED WHEN CNTRL C IS STRUCK AND WHENEVER CALLING
;       PROGRAM IS ABORTED BY THE SYSTEM.
        BR      DODONE          ;ABORT BY STOPPING
;
;
;               ---I N T E R R U P T   S E R V I C E---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
DOINT:  MOV     @#MONLOW,-(SP)  ;DROP PRIORITY TO LEVEL 4
        JSR     R5,@(SP)+
        .WORD   PR4             ;DONT LET ANYBODY SCREW US UP!
        MOV     DOCQE,R4        ;POINTER TO CURRENT Q ENTRY
        ADD     #6,R4           ;POINT R4 TO BYTE COUNT
        TST     @R4             ;ANY MORE CHARS TO OUTPUT?
        BEQ     DODONE          ;NO-TRANSFER DONE!
        INC     @R4             ;INCREMENT BYTE COUNT (IT'S NEGATIVE)
        TSTB    @-(R4)          ;CHECK FOR NULL CHAR
                                ;END OF BUFFER??
        BEQ     DODONE          ;YES--SO ALL DONE EARLY!!
        MOVB    @(R4),@#DODB    ;SEND THE DAMN CHARACTER
        INC     @R4             ;BUMP POINTER
```

```
        RTS     PC              ;EL BYE BYES
;
;
;
;               ---H A R D   E R R O R---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
DOERR:  BIS     #HDERR,@-(R4)   ;SET HARD ERROR BIT
;
;
;       ---O P E R A T I O N   C O M P L E T E---
;\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
;       RETURNS CONTROL TO RT-11 QUE MANAGER
DODONE: BIC     #100,@#DOSR     ;TURN OFF INT. ENABLE
        MOV     PC,R4
        ADD     #DOCQE-.,R4     ;ADDR OF NEXT Q ENTRY POINTER
        MOV     @#MONLOW,R5
        JMP     @OFFSET(R5)     ;JMP TO QUE MANAGER
;
;
$INPTR: .WORD   0               ;POINTS TO COMMON ENTRY CODE
DOSIZE=.-LOADPT
;
.END
```

```
C       S Q R T . F O R
C
C       THIS PROGRAM DEMONSTRATES VARIOUS TECHNIQUES
C       THAT MAY BE USED TO COMMUNICATE WITH THE CYBER-74
C       VIA FORTRAN ON THE CSN GT-44.
C
C       THIS PROGRAM IS CURRENTLY COMPATIBLE WITH KRONOS 2.1.1
C       LEVEL 397 UNDER TELEX.  THE PROGRAM SHOULD BE COMPATIBLE
C       WITH THE NETWORK OPERATING SYSTEM (NOS) SCHEDULED TO COME
C       UP IN JUNE 1976 BUT THERE ARE NO GUARENTEES.
C
C

        LOGICAL*1 LINE(101),ERR,CNTRLC(2)

        DATA    LINE/101*0/,ERR/0/,CNTRLC/"3,"3/

        CALL    ASSIGN(10,'DI:',3,'RDO','NC',2) !INPUT DEVICE=UNIT 10
        CALL    ASSIGN(11,'DO:',3,'NEW','CC',1) !OUTPUT DEVICE=UNIT 11

1       WRITE   (11,109)CNTRLC                  !SEND A CONTROL C
        REWIND  11                              !THE HARD WAY
        CALL    GETSTR(10,LINE,100)             !GET TWO LINES
        CALL    GETSTR(10,LINE,100)             !AND IGNORE THEM
        CALL    SNOOZE(70)                      !SNOOZE A WHILE IN CASE
                                                !TELEX ADDS ANOTHER LINE
        CALL    LZERO(LINE,101)                 !ZERO ARRAY FOR NEXT TIME


5       CALL    LZERO(LINE,101)                 !ZERO OUT LINE ARRAY
        WRITE   (11,101)                        !SEND A STRING
        REWIND  11                              !FORCE OUTPUT NOW
        CALL    GETSTR(10,LINE,100,ERR)         !GET ANSWER
        CALL    SCOMP('READY.',LINE,IVAL)       !COMPARE ANSWER WITH 'READY.'
        IF      (IVAL)GOTO 5                    !IF IT FLUNKED TRY AGAIN

        WRITE   (11,102)                        !SEND RNH COMMAND
        REWIND  11                              !FORCE EMPTY BUFFER
7       CALL    LZERO(LINE,101)                 !ZERO LINE ARRAY
        CALL    GETSTR(10,LINE,100,ERR)         !GET A LINE FROM TELEX
        CALL    SCOMP('READY.',LINE,IVAL)       !CHECK FOR QUESTION MARK
        IF      (IVAL.NE.0)GOTO 7               !IF OKAY THEN CONTINUE
                                                !OTHERWISE TRY AGAIN

10      TYPE    108                             !ASK FOR NUMBER
        ACCEPT  104,X                           !GET A NUMBER
        WRITE   (11,105),X                      !SEND IT DOWNSTAIRS
        REWIND  11                              !FORCE EMPTY BUFFER
        CALL    LZERO(LINE,101)                 !ZERO ARRAY FOR READ & PRINT
        CALL    GETSTR(10,LINE,100,ERR)         !GET THE ANSWER
        TYPE    107,(LINE(I),I=1,80)            !TYPE IT
        GOTO    10                              !DO IT AGAIN
```

132

```
101     FORMAT(1X,'OLD,DSQRT')
102     FORMAT(1X,'RNH')
103     FORMAT(1X,'**ERROR** PROGRAM FAILED TO RUN')
104     FORMAT(F15.5)
105     FORMAT(1X,F15.5)
106     FORMAT(80A1)
107     FORMAT(1X,80A1)
108     FORMAT('$TYPE A REAL NUMBER: ')
109     FORMAT(1H$,2A1)
        END
```

133

```
READY

LIST

 76/04/21.  10.37.41.
PROGRAM   DSQRT

1CDSQRT
10      PROGRAM CSNSQRT(INPUT,OUTPUT)
11C
12C    ************* D S Q R T ************
14C    FORTRAN PROGRAM TO INTERACT WITH THE CSN GT-44
15C    ************* D S Q R T ************
16C
20      PRINT,*READY.*
25    1 READ,X
30      X=DSQRT(X)
40      PRINT 5,X
50      GOTO 1
55C
60    5 FORMAT(1X,D35.28)
999     END
READY.
```

DL-11E DEVICE HANDLERS

D O . S Y S (OUTPUT HANDLER)

\/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/

THE DO HANDLER IS A STANDARD RT-11 DEVICE HANDLER TO BE USED AS WOULD BE ANY OTHER DEVICE. ONE CONSIDERATION SHOULD KEPT IN MIND---THE DL-11 IS A BYTE ORIENTED TRANSFER DEVICE AND RT-11 DEVICE HANDLERS ARE WORD ORIENTED. THEREFORE YOU MUST BE CAREFUL WHEN ATTEMPTING TO TRANSFER AN ODD NUMBER OF CHARACTERS. APPENDING A NULL BYTE TO MAKE THE COUNT EVEN IS SUFFICIENT.

****** D O . S Y S E R R O R S ******

THE ONLY ERROR POSSIBLE WITH THE DO HANDLER IS ATTEMPTING TO READ FROM IT. THIS WILL RETURN A HARD ERROR TO THE CALLING PROGRAM.

BECAUSE THE HANDLER IS COMPLETELY OUTPUT ORIENTED IT WILL NOT DETECT LOSS OF THE CARRIER SIGNAL OR ANY OTHER TYPE OF ERROR. THIS MEANS THAT THE HANDLER WILL VERY HAPPILY TRANSMIT INFORMATION ALL DAY TO A DEAD TELEPHONE LINE.

D I . S Y S (INPUT HANDLER)

THE DI HANDLER IS SOMEWHAT MORE COMPLICATED. IT RECIEVES CHARACTERS IN MUCH THE SAME MANNER AS ANY OTHER HANDLER, BUT THE SPECIAL CONSIDERATIONS OF ASCII COMMUNICATION NESCESSITATED SEVERAL ADDITIONAL FEATURES. THE DI HANDLER CONTAINS AN INTERFACE WITH THE RT-11 "SET" PROCESSOR. THERE ARE TWO VALID COMMANDS FROM RT-11 MONITER MODE.

SET DI NO LF
&
SET DI LF

THESE COMMAND PERMANTLY (UNTIL THE OPPOSITE COMMAND IS GIVEN) MODIFY THE DI HANDLER. THE COPY OF THE HANDLER ON THE DISK IS ACTUALLY CHANGED.

THE "NO LF" COMMAND SPECIFIES THAT THE HANDLER SHOULD FILL THE WORD COUNT EXACTLY.

THE "LF" COMMAND TELLS THE HANDLER TO READ AS MANY WORDS AS SPECIFIED OR UNTIL A LINE FEED CHARACTER IS RECIEVED, WHICHEVER COMES FIRST. THIS ENABLES PROCESSING OF INFORMATION LINE BY LINE RATHER THAN HAVING TO KNOW AHEAD OF TIME EXACTLY HOW MANY CHARACTERS TO EXPECT.

NOTE THAT THE NORMAL MODE FOR THE HANDLER IS.
SET DI LF

****** D I . S Y S E R R O R S ******

IN IT'S CURRENT IMPLEMENTATION THE DI HANDLER WILL RETURN BOTH HARD AND SOFT ERRORS.

1) HARD ERRORS

A) ATTEMPTING WRITE INFORMATION TO THE DI HANDLER.

B) FRAMING ERRORS. CAUSED WHEN THE SERIAL INFORMATION PASSED TO THE "UART" IS BADLY DISTORTED OR WHEN NO VALID STOP BITS EXIST IN THE SERIAL DATA.

C) PARITY ERRORS. CAUSED WHEN THE PARITY OF THE RECEIVED CHARACTER DOES NOT AGREE WITH THE EXPECTED PARITY.

ALL HARD ERRORS RETURN THE HARD ERROR BIT IN THE CHANNEL STATUS WORD SET AND WILL PREVENT ANY FURTHER ACTIVITY ON THE PARTICULAR CHANNEL (OR FORTRAN UNIT) INVOLVED. THE CHANNEL MUST BE REINITIALIZED OR IN FORTRAN THE "ERR=NNN" CLAUSE MUST BE USED.

2) SOFT ERRORS

A) LOSS OF CARRIER. CAUSED WHEN THE CARRIER STATUS BIT IS NOT SET (IE. THE CARRIER SIGNAL HAS FAILED). THE READ IN PROGRESS IS ABORTED AND AN END OF FILE IS RETURNED TO THE CALLING PROGRAM. IN FORTRAN USE THE "END=NNN" CLAUSE TO DETECT THIS CONDITION

THERE CURRENTLY EXIST TWO PROGRAMS WHICH ARE DESIGNED TO INTERFACE THE PDP-11 WITH THE CDC-6600. THE FIRST PROGRAM, NAMED "DLN" IS A REENTRANT PROGRAM WHICH MAY BE RUN IN FOREGROUND OR BACKGROUND. DLM CAUSES THE GT-44 TO ACT EXACTLY LIKE A TELTYPE CONNECTED TO THE TIMESHARING SYSTEM OVER NORMAL PHONE LINES. "DLN" HAS SEVERAL FEATURES WHICH MAY BE OF GREAT HELP WHEN WRITING PROGRAMS ON TELEX.

1) TELEX IDLING FEATURE.

IF THERE IS NO ACTIVITY ON THE PHONE LINE FOR FOUR MINUTES "DLN" WILL AUTOMATICALLY SEND A CARRIAGE RETURN IN ORDER TO PREVENT TELEX FROM 'TIMING OUT'. THIS ENABLES THE USR OF THE GT-44 TO GO OUT AND HAVE A CUP OF COFFEE WITHOUT FEAR OF TELEX DECIDING HE IS TOO SLOW AND LOGGING HIM OFF.

THIS FEATURE IS CONTROLLED BY THE CONTROL R KEY ON THE CONSOLE DEVICE. TWO MODES ARE POSSIBLE. **IDLE ON** AND **IDLE OFF**. TYPING CONTROL R WILL PUT THE USER INTO THE OPPOSITE MODE.

NOTE THAT THIS FEATURE IS NOT AVAILBLE WITH THE SINGLE JOB RT-11 MONITER.

NORMAL MODE IS **IDLE ON**.

2) BELL FEATURE.

WHEN THE BELL FEATURE IS ON THE BELL WILL BE RUNG AT THE END OF EACH LINE SENT TO TELEX. THE BELL DOES NOT RING WHEN YOU TYPE A CARRIAGE RETURN ON THE CONSOLE BUT RATHER IT RINGS WHEN THE ENTIRE LINE HAS BEEN SENT TO TELEX.

THE BELL FEATURE IS TURNED ON AND OFF WITH CONTROL G. NORMAL MODE IS BELL OFF.

THE "FILES" INTERFACE PROGRAM IS DESIGNED TO TRANSFER FILES BETWEEN THE TWO MACHINES. IT IS FAIRLY OBVIOUS IN USE. YOU PROBABLY WILL WANT TO RUN THIS PROGRAM IN THE FOREGROUND SO YOU DON'T HAVE TO SIT AROUND AND WAIT FOR THE THING TO FINISH. IF SO BE SURE THAT ALL NESCESSARY HANDLERS ARE LOADED, INCLUDING DI AND DO. NOTE THAT NO NULLS ARE SENT TO TELEX DURING TRANSMISSION AND ALL NULLS RECIEVED ARE DELETED FROM THE FINAL STORED COPY.

"FILES" WILL ALSO WORK WITH A COMMAND FILE WHICH HAS PREVIOUSLY BEEN STORED. THE FORMAT OF THE COMMAND FILES IS AS FOLLOWS:

1ST LINE---<SEND> OR <RECIEVE> DEPENDING ON WHAT YOU WANT TO DO!
2ND LINE---<OLD FILE NAME> THE PRESENTLY EXISTING FILE
3RD LINE---<NEW FILE NAME> THE NEW FILE TO BE CREATED

FOR EXAMPLE:

SEND

```
OCCSUB.FOR
OCCSUB1
------------

------------
REC
OCCSUB1
OCCSUB.FOR
------------
```

A FOREGROUND RUN MIGHT LOOK LIKE THIS

```
------------
.FRUN FILES
F>
DO YOU HAVE A COMMAND FILE?YES
COMMAND FILE NAME:*CMD.DAT

B>
.
------------
```

DON'T FORGET TO TYPE A CNTRL F BEFORE YOU TRY AND ANSWER THE QUESTIONS. I WILL TRY TO ANSWER ANY QUESTIONS ON FOREGROUND/BACKGROUND OPERATION THAT MAY ARISE.

PROGRAMMING HINTS

PROBABLY THE MOST IMPORTANT THING TO REMEMBER ABOUT COMMINCATIONS WITH ANOTHER COMPUTER SYSTEM (TELEX IS USED IN THESE EXAMPLES) IS THAT IT WILL NEVER DO EXACTLY WHAT YOU EXPECT IT TO. IF YOUR PROGRAM EXPECTS TWO CARRIAGE RETURNS TELEX WILL SUPPLY THREE... OR ONE, AND SOMETIMES TWO.

THEREFORE THE CARDINAL RULE IN THIS GAME IS KNOW EXACTLY WHAT YOUR TELEX PROGRAM WILL DO UNDER ALL CIRCUMSTANCES AND EXACTLY WHAT YOUR GT-44 PROGRAM EXPECTS TO RECIEVE. THE EXAMPLE MENTIONED ABOVE HAS BEEN THE MOST COMMON SOURCE OF ERROR TO DATE. IF YOU EXPECT TO RECIEVE TWO LINES OF INFORMATION AND TELEX ONLY SENDS ONE-- YOUR GT-44 PROGRAM WILL PROBABLY SIT AROUND ALL AFTERNOON WAITING FOR THE SECOND LINE

BELOW IS AN SHORT SECTION OF FORTRAN CODE WHICH FETCHES THE DO AND DI HANDLERS FROM THE DISK AND ASSIGNS THEM TO FORTRAN LOGICAL UNIT NUMBERS 10 AND 11 RESPECTIVLY.

```
        INTEGER DONAME(4),DINAME(4)
        DATA    DONAME/2RDO,0,0,0/,DINAME/2RDI,0,0,0/

C       FETCH DO AND DI HANDLERS FROM THE DISK
C       NOTE THAT THESE FETCHES ARE NOT USUALLY NESCESSARY
        IF      (IFETCH(DONAME).NE.0) STOP 'FATAL FETCH ERROR ON DO'
        IF      (IFETCH(DINAME).NE.0) STOP 'FATAL FETCH ERROR ON DI'

C       ASSIGN #10 TO OUTPUT TO TELEX AND #11 TO INPUT FROM TELEX
        CALL ASSIGN(10,'DO:',3,'NEW','CC',1)     !SEE SECTION B-1 OF
        CALL ASSIGN(11,'DI:',3,'RDO','NC',2)     !FTN OTS MANUAL
NOTE----CARRAIGE CONTROL IS SPECIFIED FOR DO HANDLER----
    ----DI HANDLER IS SPECIFIED READ ONLY AND DOUBLE BUFFERED----
```

THE SECTION OF CODE BELOW SENDS THE FOLLOWING CHARACTERS TO THE DL-11E AND FROM THERE TO MA BELL

<   24<CR&LF>>

```
        LINE=24
        WRITE   (10,80) LINE                !SEND LINE TO TELEX
        REWIND  10
80      FORMAT(1X,I5)
```

NOTE THE REWIND STATEMENT DIRECTLY AFTER THE WRITE. THIS SENDS THE CHARACTERS IMMEDIATLY WITHOUT WAITING FOR THE FORTRAN OUTPUT BUFFER TO BECOME FULL. OTHERWISE FORTRAN WOULD NOT ACTUALLY WRITE UNTIL 512 (10) CHARACTERS WERE IN THE BUFFER

THE NEXT SEGMENT OF CODE READS A FIVE CHARACTER INTEGER

NUMBER FROM THE TELEPHONE LINE.

```
        READ    (11,90,END=100,ERR=110) LINE
        REWIND  11

100     TYPE    101
        STOP

110     TYPE    111
        STOP

90      FORMAT(I5)
101     FORMAT(1X,'END OF FILE---PROBABLY LOST CARRIER')
111     FORMAT(1X,'HARD ERROR----PROBABLY NOISE ON PHONE LINE')
```

```
******** BELOW IS A SHORT FORTRAN PROGRAM DESIGNED TO
         INTERFACE WITH A PROGRAM ON TELEX (SEE BELOW).
         NOTE EXAMPLES IN THE USE OF "GETSTR" AND "SCOMP"
         THESE TWO ROUTINES IN THE FORTRAN "SYSLIB" PROVIDE
         AN EASY WAY TO CHECK FOR ANY ARBITRARY STRING


C        S Q R T . F O R
C
C        THIS PROGRAM DEMONSTRATES VARIOUS TECHNIQUES
C        THAT MAY BE USED TO COMMUNICATE WITH THE CYBER-74
C        VIA FORTRAN ON THE CSN GT-44.
C
C        THIS PROGRAM IS CURRENTLY COMPATIBLE WITH KRONOS 2.1.1
C        LEVEL 397 UNDER TELEX.  THE PROGRAM SHOULD BE COMPATIBLE
C        WITH THE NETWORK OPERATING SYSTEM (NOS) SCHEDULED TO COME
C        UP IN JUNE 1976 BUT THERE ARE NO GUARENTEES
C
C

         LOGICAL*1 LINE(101),ERR,CNTRLC(2)

         DATA     LINE/101*0/,ERR/0/,CNTRLC/"3,"3/

         CALL     ASSIGN(10,'DI:',3,'RDO','NC',2) !INPUT DEVICE=UNIT 10
         CALL     ASSIGN(11,'DO:',3,'NEW','CC',1) !OUTPUT DEVICE=UNIT 11

1        WRITE    (11,109)CNTRLC                  !SEND A CONTROL C
         REWIND   11                              !THE HARD WAY
         READ     (10,106,END=35)LINE             !GET ONE LINE
         CALL     SNOOZE(70)                      !SNOOZE A WHILE IN CASE
                                                  !TELEX ADDS ANOTHER LINE
         CALL     LZERO(LINE,101)                 !ZERO ARRAY FOR NEXT TIME


5        CALL     LZERO(LINE,101)                 !ZERO OUT LINE ARRAY
         WRITE    (11,101)                        !SEND A STRING
         REWIND   11                              !FORCE OUTPUT NOW
         CALL     GETSTR(10,LINE,100,ERR)         !GET ANSWER
         CALL     SCOMP('READY.',LINE,IVAL)       !COMPARE ANSWER WITH 'READY.'
         IF       (IVAL)GOTO 5                    !IF IT FLUNKED TRY AGAIN

         WRITE    (11,102)                        !SEND RNH COMMAND
         REWIND   11                              !FORCE EMPTY BUFFER
7        CALL     LZERO(LINE,101)                 !ZERO LINE ARRAY
         CALL     GETSTR(10,LINE,100,ERR)         !GET A LINE FROM TELEX
         CALL     SCOMP('READY.',LINE,IVAL)       !CHECK FOR QUESTION MARK
         IF       (IVAL.NE.0)GOTO 7               !IF OKAY THEN CONTINUE
                                                  !OTHERWISE TRY AGAIN

10       TYPE     108                             !ASK FOR NUMBER
         ACCEPT   104,X                           !GET A NUMBER
         WRITE    (11,105),X                      !SEND IT DOWNSTAIRS
```

```
        REWIND  11                              !FORCE EMPTY BUFFER
        CALL    LZERO(LINE,101)                 !ZERO ARRAY FOR READ & PRINT
        CALL    GETSTR(10,LINE,100,ERR)         !GET THE ANSWER
        TYPE    107,(LINE(I),I=1,80)            !TYPE IT
        GOTO    10                              !DO IT AGAIN

35      TYPE    110
        CALL    EXIT


101     FORMAT(1X,'OLD,DSQRT')
102     FORMAT(1X,'RNH')
103     FORMAT(1X,'**ERROR** PROGRAM FAILED TO RUN')
104     FORMAT(F15.5)
105     FORMAT(1X,F15.5)
106     FORMAT(80A1)
107     FORMAT(1X,80A1)
108     FORMAT('$TYPE A REAL NUMBER: ')
109     FORMAT(1H$,2A1)
110     FORMAT(1X,'**ERROR** NO CARRIER PRESENT')
        END
```

******** BELOW IS THE PROGRAM RUN ON TELEX WHICH INTERFACES WITH "SQRT.FOR". THE MOST EFFICIENT WAY TO HAVE TELEX DO COMPUTATIONS FOR YOU IS SEND ALL THE DATA TO THE TELEX PROGRAM AT ONCE AND LET IT CHUNK AWAY AT IT FOR AS LONG AS IT TAKES. EVERY TIME A TELEX PROGRAM MAKES AN I/O REQUEST IT IS ROLLED OUT AND WILL NOT BE ROLLED IN AGAIN UNTIL THAT REQUEST IS COMPLETLY SATISFIED. THERFORE IT IS TO YOUR ADVANTAGE TO DO AS LITTLE I/O AS POSSIBLE! PLEASE NOTE THAT THIS PROGRAM IS DESIGNED TO RUN UNDER KRONOS 2.1 USING THE "FORTRAN" SUBSYSTEM. IT WILL NOT RUN UNDER NOS.

```
1CDSQRT
      PROGRAM CSNSQRT(INPUT,OUTPUT)
C
C    ************* D S Q R T ************
C    FORTRAN PROGRAM TO INTERACT WITH THE CSN GT-44
C    ************* D S Q R T ************
C
      PRINT,*READY.*
    1 READ,X
      X=DSQRT(X)
      PRINT 5,X
      GOTO 1
C
    5 FORMAT(1X,D35.28)
      END
```

DL-11E DESCRIPTION

THE DL-11E IS AN ASYNCHRONOUS LINE INTERFACE DESIGNED TO ASSEMBLE OR DISASSEMBLE THE SERIAL INFORMATION REQUIRED BY A COMMUNICATIONS DEVICE (COUPLER) FOR PARALLEL TRANSFER OF INFORMATION TO (OR FROM) THE PDP-11 UNIBUS.

IN SIMPLE LANGUAGE IT IS THE INTERFACE BETWEEN THE PDP-11 AND A TELEPHONE LINE TO THE OUTSIDE WORLD. IN PRACTICE IT MAY BE TREATED AS A CLOSE DUPLICATE TO DL-11A WHICH INTERFACES WITH A PAPER TAPE READER AND PUNCH.

COMMUNICATION WITH THE DL-11E AND THE PDP-11 IS THROUGH FOUR ADDRESSES ON THE UNIBUS: THE INPUT STATUS REGISTER, THE INPUT BUFFER, THE OUTPUT STATUS REGISTER AND THE OUTPUT BUFFER.

175610	=	INPUT STATUS REGISTER
175612	=	INPUT CHARACTER BUFFER
175614	=	OUTPUT STATUS REGISTER
175616	=	OUTPUT CHARACTER BUFFER

FOR INFORMATION ON MEANINGS OF PARTICULAR BITS IN EACH OF THESE REGISTERS SEE CHAPTER 4 OF THE "DL-11 ASYNCHRONOUS LINE INTERFACE MANUAL"

MORE INFORMATION CAN BE OBTAINED BY CONTACTING ME AT THE CSN LAB OR AT HOME (549-1387). PLEASE REFER ANY PROBLEMS OR SUGGESTIONS TO ME SO I CAN DO SOMETHING ABOUT THEM.

GOOD LUCK
STUART P. SIMS