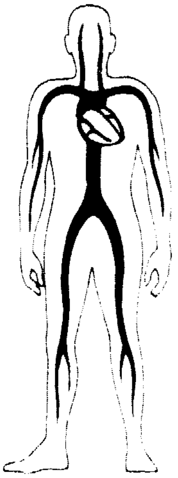
1. Marvin Bernhard SD

NASA-CR-167705



2. SPECIAL REPORT

on the

GRAPHICS AND DATA ACQUISITION

SOFTWARE PACKAGE

Prepared for the NASA/Johnson Space Center

Biomedical Research Laboratories

3. by William G. Crosier

6. December 1, 1981

5. Contract NAS 9-14880

4. TECHNOLOGY INCORPORATED
LIFE SCIENCES DIVISION
P. O. Box 58827
Houston, Texas 77058

(NASA-CR-167705) THE GRAPHICS AND DATA ACQUISITION SOFTWARE PACKAGE Special Report (Technology, Inc., Houston, Tex.) 77 p
HC A05/MF A01 CSCL 09B
N82-30967
Unclas
G3/61 30971

Technology Incorporated
Life Sciences Division

P.O. Box 58827
Houston, Texas 77058

Abstract

A new software package has been developed for use with micro and minicomputers, particularly Digital Equipment Corporation's LSI-11/PDP-11 series. The package has a number of Fortran-callable subroutines which perform a variety of frequently needed tasks for biomedical applications. All routines are well documented, flexible, easy to use and modify, and require minimal programmer knowledge of peripheral hardware. The package is also economical of memory and CPU time. A single subroutine call can perform any one of the following functions: (1) Plot an array of integer values from sampled A/D data; (2) Plot an array of Y values versus an array of X values; (3) Draw horizontal and/or vertical grid lines of selectable type; (4) Annotate grid lines with user units; (5) Get coordinates of user-controlled crosshairs from the terminal for interactive graphics; (6) Sample any analog channel with program selectable gain; (7) Wait a specified time interval; and (8) Perform random access I/O of one or more blocks of a sequential disk file. Several miscellaneous functions are also provided. These routines are modular and easily changed, and are especially applicable for uses in biomedical research laboratories such as NASA's where adaptability is important and software development time is limited. Complete source code listings, example main programs, and sample output are included.

APPROVAL SHEET

for the

GRAPHICS AND DATA ACQUISITION

SOFTWARE PACKAGE

Approved by: Millard F Reschke

Millard Reschke, Ph.D. (Systems Requirements)
NASA/JSC Neuroscience Laboratory

Edward C. Moseley

Edward C. Moseley, Ph.D.
NASA/JSC Medical Sciences Division

Phillip C Johnson

Phillip C. Johnson, M.D. (Branch Chief)
NASA/JSC Medical Research Branch

Joseph T. Baker

Joseph T. Baker (Section Supervisor)
Technology Incorporated

Harry L. Halbrecher

Harry Halbrecher (Project Manager)
Technology Incorporated

PERSONNEL

William G. Crosier - Software Design and Development, Documentation

Kay Elton - Word Processing for Documentation

TABLE OF CONTENTS

Introduction

Numerous experiments in biomedical research laboratories involve the collection and display of sampled analog data using a mini or microcomputer. Often, however, many experiments which initialy seem simple, turn out to involve a considerable investment in programming effort because of the difficulty in finding suitable graphics packages, or in using such devices as analog to digital (A/D) converters in a high level language, such as Fortran. Most data acquisition routines (except for those involving realtively low sampling rates) tend to be application specific, and are written in assembly language by individuals who must have a fairly intimate knowledge both of the particular A/D systems used and of the assembly language of the computer. In addition, biomedical data often seems particularly well suited for graphical displays, because of the complexity of physiological data and also because of the ease of interpretation of graphs by medical or other personnel without an extensive computer or statistical background. Graphics software packages are available from various sources, but most are not particularly well suited for use with microcomputers. Many of the commercially available packages are quite large and have features that are much more extensive than needed in many applications.

Some of the features are used rarely, if at all, but their presence still requires a rather large amount of main memory and disk space in the system. The architecture of some of these packages can be quite complicated, and with many of their routines rather poorly documented, it can be a formidable task for a programmer to remove the extra, unneeded features from such a package without interfering with the basic functions that are required. As a result, these large packages are often only suited for larger computer facilities, where large amounts of memory are available and where numerous users may make use of the myriad of features offered.

Another problem with some graphics packages is that they are not particularly easy to use with real time interactive programs. Although fine for offline plotting appliations, they may be too slow or too difficult to use in cases where real time data acquisition and control of an experiment are taking place. In addition, some packages are not sufficiently well documented, particularly with comments in the subroutines or with examples, to allow them to be used or modified easily by programmers without extensive knowledge of the hardware being used. Sometimes, it can take a naive programmer several weeks to write one fairly simple program, because of the time required to discover the pecularities of analog to digital converters and graphics terminals or plotters. The plethora of functions available in some packages also seems only to cause confusion among some users, since many of the functions are redundant and do not have to be used at all.

Because of these problems with commercially available software, a new data acquisition graphics package was developed for use at the NASA Johnson Space Center Life Sciences Laboratories. It was designed for use specifically with micro or minicomputers, particularly Digital Equipment Corporation's LSI-11 and PDP-11 series, using the RT-11 operating system. Desired features of the package include the following: (1) It should be relatively small in size so that it can be used easily with microcomputers having limited memory (56K bytes or less). (2) It should include the most frequently needed graphics, analog to digital conversion (A/D), and miscellaneous capabilities. However, extra functions which are not commonly needed or which can be performed with alternate methods, and which require too much memory, should not be included. (3) The package should be modular, flexible, easy to learn, use, and modify. Additionally, use of the package should require only a minimal knowledge of the peripheral hardware. The package should be adaptable to various types of Tektronix terminals or possibly to X-Y plotters. It should also be useable with Fortran main programs. (4) In order to meet the previous objective, the package should be written primarily in a high level language, and should be well commented and otherwise documented as well. (5) Finally, it also should be able to handle moderately fast analog to digital conversion and displays for real-time applications.

With the above objectives in mind, a software package was developed which has the following characteristics. First, it consists of a number of Fortran-callable subroutines which perform all necessary tasks required for interfacing with an A/D system or various types of Tektronix terminals. Only a few of these routines would need to be modified if a different type of terinal or X-Y plotter was used. Further, most of the subroutines themselves are written in Fortran and all are fully commented so that they can be easily modified by others, if necessary. In addition, a few other simple miscellaneous routines were written to perform other tasks used frequently in laboratory applications. A list of all of the routines in the package is shown in Table 1.

TABLE 1

SUBROUTINES IN THE SOFTWARE PACKAGE

Routine		Description
GRINIT	-	Initialize graphics parameters for other routines
MPLOT	-	Change position (move cursor), go to alphanumeric mode, or draw vector
COPY	-	Make hard copy of Tektronix terminal screen & erase it if desired
ERASE	-	Erase Tektronix screen without copying
CHRSIZ	-	Change character size
GINPUT	-	Switch to graphic input mode, display & get coordinates of user-controlled crosshairs
ARYPLT	-	Plot an integer array of Y values (A/D samples, etc), with straight lines connecting points, with variable scaling
XYPLOT	-	Plot an array of real Y values vs an array of real X values, with straight interconnecting lines, with variable scaling
GRID	-	Draw grid lines (selectable type) over desired area
ANOTAT	-	Label plot axes with user units at some or all grid lines
BELL	-	Ring terminal bell/beep (Variable duration & modulation control)
WAIT	-	Wait a desired period of time (1/60 sec resolution)
ISAMPA	-	Sample an A/D channel with selectable gain & channel number
DISKIO	-	Perform random access multiblock binary I/O to a sequential disk file

A few of these routines also use subroutines from the Fortran library in the system to access the system line frequency clock, to send characters to the terminal through the terminal handler, to perform double word integer arithmetic, and do disk I/O. These few routines in the package would have to be modified if a different computer or operating system was used. In addition, routines MPLOT, COPY, ERASE, and possibly GRINIT would have to be changed if a different terminal or plotter was used. However, such changes should be fairly straightforward once the characteristics of any particular terminal were determined.

The package will presently support the following hardware on an LSI-11 series microcomputer. First, analog to digital conversion can be performed. with LSI-11 compatible A/D systems manufactured by ADAC, Data Translation, or DEC, using different modes. Programmable gain and random channel selection are supported. A throughput rate of 4000 samples per second or more can be accomplished if little or no computation is performed between samples. Furthermore, all of the Tektronix 4000 series graphics terminals can be used with the package. Features of some of these terminals which are supported, but not required, include: (1) 12-bit addressing for X and Y coordinates, (2) variable character sizes and dot-dash line types (using the hardware built into the terminals), (3) graphic input mode for interactive applications (so that a program can determine the positions of user controlled crosshairs), and (4) write-through mode for non-stored (refreshed) displays. Additionally, several different types of X-Y plotters could be used with the package, although routine MPLOT would have to be modified to accommodate the requirements of the particular plotter used.

The remainder of this special report consists of the following. First, some simple example main programs are given in the next section in order to demonstrate the use of, and capabilities of, some of the routines in the package. Graphical outputs from the sample programs are also provided. Following that section, each subroutine in the package is discussed separately, with information on calling conventions, parameters (arguments) to be passed, and any restrictions or additional details concerning its operation. Source listings of each subroutine are also given in each section.

Additional examples and other programs are provided in the references below.

References

1. Crosier, William G.; Forrest, Larry J.; and Jones, Kenneth W. "A Microcomputer-Based Data Acquisition, Display, and Control System for Vestibulo-Spinal Hoffmann Reflex Experiments." Proceedings, 3rd Annual Conference of the IEEE Engineering in Medicine and Biology Society, Frontiers of Engineering in Health Care, B. A. Cohen, ed. September, 1981, pp. 71-75.

2. Crosier, William G. "A Simplified Data Acquisition and Graphics Software Package for Biomedical Research Applications with Small Computers." Proceedings, IEEE Frontiers of Computers in Medicine Conference, Robin B. Lake, Ed., September, 1981, pp. 84-86.

3. Crosier, William G. "Special Report: A General-Purpose Data Acquisition and Analysis System for Nystagmus and Related Data." Prepared for the NASA/JSC Neuroscience Research Laboratory, December 1981.

INSTALLATION

All of the subroutines in this package are callable by Fortran programs and are designed to be used with DEC's RT-11 operating system. For convenience, some or all of them may be placed in a library on the same or a different disk as the System Subroutine Library (SYSLIB).

For example, in order to create a library called LABLIB containing routines GRINIT, MPLOT, ARYPLT, and ISAMPA, do the following. First, put the source code for each routine in separate files on the default device DK:, each with the name of the subroutine: GRINIT.FOR, MPLOT.FOR, ARYPLT.FOR, and ISAMPA.MAC. Second, compile and assemble these routines with the RT-11 (Version 4) Fortran compiler (Version 02.1 or later) and Macro assembler (Version 4 or later):

```
FORT/WARN/LIST:DK:   (GRINIT,MPLOT,ARYPLT)
MACRO/CR/LIST:DK:    ISAMPA
```

Next, create the actual library file LABLIB.OBJ from the individual object files:

```
LIBR/CREATE  LABLIB
Files?  GRINIT,MPLOT,ARYPLT,ISAMPA
```

The same procedure may also be followed for more subroutines, but you should include only six or fewer file names on any command line. If you want to put routines from more than six files in the library, then include the "/PROMPT" option after the command "LIBR", then specify six or fewer file names per line. After the last line, type two slashes (//) to terminate the file name entry. Refer to the RT-11 System User's Guide for more information.

You will note, in the descriptions of several of the subroutines, that they in turn call other subroutines. For instance, MPLOT and WAIT are called by a number of other routines. Make sure that all of the subroutines which are needed (either directly or indirectly) are included in the library which you build. Otherwise, an undefined global error will occur when you attempt to link the programs. In addition, several DEC supplied routines from the System Subroutine Library are used, and these must be present in SYSLIB. Such routines include ITTOUR (called by MPLOT, COPY, ERASE, CHRSIZ, BELL, etc.) and GTIM and the Integer*4 routines (called by WAIT). Refer to the descriptions of the appropriate routines in this report for more information.

Sample Main Programs

This section contains several simple programs which demonstrate some of the capabilities of the subroutines in this package. By referring to the program listings and sample output which follow, and running these programs yourself, you should be able to learn how to use the subroutines to perform other similar functions.

The first example program is ADTEST. This uses only the routine ISAMPA from the subroutine package. In addition, ITTINR and IPOKE from the System Subroutine Library are also used. More information on these last two routines is in the DEC RT-11 Version 4 Programmer's Reference Manual.

Program ADTEST does the following. First, it asks the user for the types of analog-to-digital (A/D) and digital-to-analog (D/A) converters present in the system. Second, the program asks which analog input channel should be sampled, and the programmable gain code to use. After this, the program samples the specified channel with subroutine ISAMPA and sends the converted signal back out to the D/A converter(s) in the system where an oscilloscope can be used to monitor the D/A output. The sampling is repeated until the user strikes the Return key. The D/A output resembles a sampled (choppy) version of the input analog signal, with a scale factor dependent on the programmable gain, A/D range jumpers, and D/A range jumpers selected.

Program ADTEST may also be used for diagnostic purposes, since it enables one to quickly check both the A/D and D/A converters plus certain CPU and memory functions, without being particularly difficult to use. In addition, it may be easily modified for other purposes. The program listing and sample output follows.

```
0001          PROGRAM ADTEST
       C
       C      PURPOSE: TEST A/D & D/A BOARD
       C
       C      METHOD: SAMPLE SELECTED A/D CHANNEL & TRANSFER SAMPLED
       C          VALUE TO BOTH D/A'S, AND REPEAT UNTIL
       C          INTERRUPTED BY USER.  USER MAY INPUT 10 HZ
       C          TRIANGLE WAVE FROM SIGNAL GENERATOR, OR ANY
       C          OTHER SIGNAL, AND D/A'S SHOULD FOLLOW THE INPUT.
       C          SCALING MAY BE DIFFERENT, HOWEVER, DEPENDING ON
       C          PROGRAMMBLE GAIN AND VOLTAGE RANGES SELECTED.
       C
       C      WRITTEN BY: WILLIAM G. CROSIER
       C      REVISED:    23 DEC. 1981
       C
       C      SUBROUTINES REQUIRED: ISAMPA
       C
       C
0002          INTEGER DAC1
       C
       C      SET UP DEVICE ADDRESSES
0003          DAC1 = "176760                ! D/A #1 DATA REGISTER
       C
0004          TYPE *,'A/D & D/A CONVERTER TEST PROGRAM -- VERSION 3'
0005          TYPE *,'ENTER A/D TYPE:      0 FOR ADAC, OR'
0006          TYPE *,'                     1 FOR DATA TRANSLATION'
0007          ACCEPT *,IADTYP
0008          IF (IADTYP.LT.0 .OR. IADTYP.GT.1) STOP 'ILLEGAL A/D TYPE'
       C
0010          TYPE *,'ENTER D/A TYPE:      0 FOR ADAC, OR'
0011          TYPE *,'                     1 FOR DATA TRANSLATION'
0012          ACCEPT *,IDATYP
0013          IF (IDATYP.LT.0 .OR. IDATYP.GT.1) STOP 'ILLEGAL D/A TYPE'
       C
0015          TYPE *,'ENTER NUMBER OF D/A CONVERTERS ON BOARD (1-4)'
0016          ACCEPT *, NDACS
0017          IF (NDACS.LT.1 .OR. NDACS.GT.4) STOP 'ILLEGAL NO. OF DACS'
       C
0019          TYPE *,'VERIFY THAT A/D IS JUMPERED FOR BIPOLAR,'
0020          TYPE *,'2''S COMPLEMENT OPERATION, WITH CSR ADDRESS'
0021          TYPE *,'OF 176770 OCTAL'
0022          TYPE *,'ALSO VERIFY THAT D/A IS JUMPERED FOR BIPOLAR'
0023          TYPE *,'OPERATION, WITH ADDR OF FIRST D/A OF 176760 OCTAL.'
0024          TYPE *.'ENTER -1 FOR INPUT CHANNEL NO. TO STOP.'
       C
       C      REQUEST A/D CONTROL PARAMETERS
       C
0025   5      TYPE 1005
0026   1005   FORMAT('0A/D INPUT CHANNEL NO. ?',$)
0027          ACCEPT *, ICHAN
0028          IF (ICHAN.LT.0 .OR. ICHAN.GT. 31) STOP
0030   20     TYPE *,'PROGRAMMABLE GAIN CODE:'
0031          IF (IADTYP .EQ. 0)  TYPE 1020
0033   1020   FORMAT ('   0 = GAIN OF 10' / '   1 = GAIN OF 5' /
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
              @                 '    2 = GAIN OF 2' / '    3 = GAIN OF 1')
0034            IF (IADTYP .EQ. 1)  TYPE 1030
0036    1030    FORMAT ('    0 = GAIN OF 1' / '    1 = GAIN OF 2' /
              @                 '    2 = GAIN OF 4' / '    3 = GAIN OF 8')
0037            TYPE *,'GAIN CODE DESIRED (0-3)?'
0038            ACCEPT *,IPGNCD
0039            IF (IPGNCD.LT.0 .OR. IPGNCD.GT.3)  GO TO 20
        C
0041            TYPE *,'HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D'
0042            TYPE *,'  CHANNEL NO. OR PROGRAMMABLE GAIN.'
        C
        C       DO A/D CONVERSION & OUTPUT DATA TO BOTH D/A'S
        C       UNTIL USER STRIKES RETURN KEY ON TERMINAL.
        C
0043    40      IF (ITTINR() .GE. 0) GO TO 5          !HAS RETURN KEY BEEN HIT?
        C
0045            IDATA = ISAMPA(ICHAN,IPGNCD,IADTYP) !SAMPLE A/D CHAN.
        C       IF DATA TRANSLATION D/A, CONVERT CODING TO OFFSET BINARY
0046            IF (IDATYP .EQ. 1)  IDATA=IDATA+2048
        C
0048            IDAC = DAC1                     !D/A CONV. ADDR.
0049            DO 50 K=1,NDACS                       !FOR EACH D/A,
0050            CALL IPOKE(IDAC,IDATA)                ! OUTPUT SAMPLE TO D/A
0051    50      IDAC = IDAC + 2                       !ADDR. OF NEXT D/A
0052            GO TO 40
        C
0053            END
```

```
FORTRAN IV       Storage Map for Program Unit ADTEST

Local Variables, .PSECT $DATA, Size = 000022 (     9. words)

Name    Type  Offset      Name    Type  Offset      Name    Type  Offset
DAC1    I*2   000000      IADTYP  I*2   000002      ICHAN   I*2   000010
IDAC    I*2   000016      IDATA   I*2   000014      IDATYP  I*2   000004
IPGNCD  I*2   000012      K       I*2   000020      NDACS   I*2   000006

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
IPOKE   I*2     ISAMPA  I*2     ITTINR  I*2
```

```
RUN ADTEST
A/D & D/A CONVERTER TEST PROGRAM -- VERSION 3
ENTER A/D TYPE: 0 FOR ADAC, OR
                1 FOR DATA TRANSLATION
0
ENTER D/A TYPE: 0 FOR ADAC, OR
                1 FOR DATA TRANSLATION
0
ENTER NUMBER OF D/A CONVERTERS ON BOARD (1-4)
2
VERIFY THAT A/D IS JUMPERED FOR BIPOLAR,
2'S COMPLEMENT OPERATION, WITH CSR ADDRESS
OF 176770 OCTAL
ALSO VERIFY THAT D/A IS JUMPERED FOR BIPOLAR
OPERATION, WITH ADDR OF FIRST D/A OF 176760 OCTAL
ENTER -1 FOR INPUT CHANNEL NO  TO STOP

A/D INPUT CHANNEL NO. ?0
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
3
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO. OR PROGRAMMABLE GAIN.
```

← Check A/D channel 0 with programmable gain of 1

```
A/D INPUT CHANNEL NO. ?0
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
2
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO. OR PROGRAMMABLE GAIN.
```

Check A/D channel 0 with prog. gain of 2

```
A/D INPUT CHANNEL NO. ?2
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
2
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO. OR PROGRAMMABLE GAIN.
```

Check A/D channel 2 with prog. gain of 2

```
A/D INPUT CHANNEL NO. ?5
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
0
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO. OR PROGRAMMABLE GAIN.
```

Check A/D channel 5 with prog. gain of 10

```
A/D INPUT CHANNEL NO. ?5
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
1
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO  OR PROGRAMMABLE GAIN.
```

Check A/D channel 5 with prog. gain of 5

```
A/D INPUT CHANNEL NO. ?5
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
2
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO  OR PROGRAMMABLE GAIN.
```

Check A/D channel 5 with prog. gain of 2

```
A/D INPUT CHANNEL NO. ?5
PROGRAMMABLE GAIN CODE:
  0 = GAIN OF 10
  1 = GAIN OF 5
  2 = GAIN OF 2
  3 = GAIN OF 1
GAIN CODE DESIRED (0-3)?
3
HIT RETURN KEY TO TERMINATE SAMPLING & CHANGE A/D
  CHANNEL NO. OR PROGRAMMABLE GAIN
```

Check A/D channel 5 with prog. gain of 1

```
A/D INPUT CHANNEL NO. ?-1
```

Enter -1 to stop.

```
STOP --
```

The second sample program is called DEMOGR. This program demonstrates the use of GRINIT, CHRSIZ, ERASE, and MPLOT to plot interesting patterns on the terminal screen. Subroutine ERASE also calls WAIT, and the subroutines also use several routines from the System Subroutine Library (such as ITTOUR, GTIM, and the Integer*4 functions).

Program operation proceeds as follows. First, graphics parameters are initialized by calling routine GRINIT. This routine asks about the type of terminal that is being used. Second, routine CHRSIZ is called to change the character size to #3 (next to the smallest) if a 4014-type terminal is being used. Third, the screen is erased, and the program asks the operator to supply four numbers which control the plot to be produced. The first value requested is N, which is the number of points. N should range between 10 and 30,000. The second value is the Shrinkage Factor (see below). This value should be between 0 and 2. The third number is the Angle Increment, and the fourth is the Line Type (normally 1, 97, 98, 99, 100, 104, or 112). The meaning of these parameters should be clear from the discussion following, and from observing the program's operation. The plot is drawn as follows. First, a point slightly to the right of the center of the terminal's screen is the middle of the plot. From this point, any other point can be defined by a vector with a particular radius and angle, using polar coordinates. The program selects a starting value for the radius, and a starting angle of 0 (relative to horizontal), so that the first point on the plot is to the right of the plot's center. The program then draws N lines, by changing the values for R and the angle each time. If the Shrinkage Factor is 0, then R is always the same (all points will lie on an imaginary circle). If the Shrinkage is 1, then the last point will be at the center and all other points will lie on an imaginary spiral. Other values for the Shrinkage may be used also. After plotting each point on the plot, the program changes the angle by the user-specified Angle Increment. If this value is very small, then a cirle or spiral will be drawn. If the value is larger, then a polygon or star may be drawn. The process will continue until N lines have been drawn, and then the user may try a different combination of parameters to produce a new plot. Note that after R and the angle are computed for each point, they are transformed back into rectangular (X and Y) coordinates before the call to MPLOT. The various line types are defined in the section on routine MPLOT.

Refer there for more information. However, the various dotted and dashed lines (types 97-104) can only be produced on a Tektronix 4014 terminal with the enhanced graphics option.

Although this program does not do anything particularly useful, it does demonstrate the use of several graphics subroutines. In particular, it is an example of how MPLOT may be used in a special plotting situation, with little extra programming required. It also shows how a relatively simple program can generate rather intricate plots. The program listing and sample output follow.

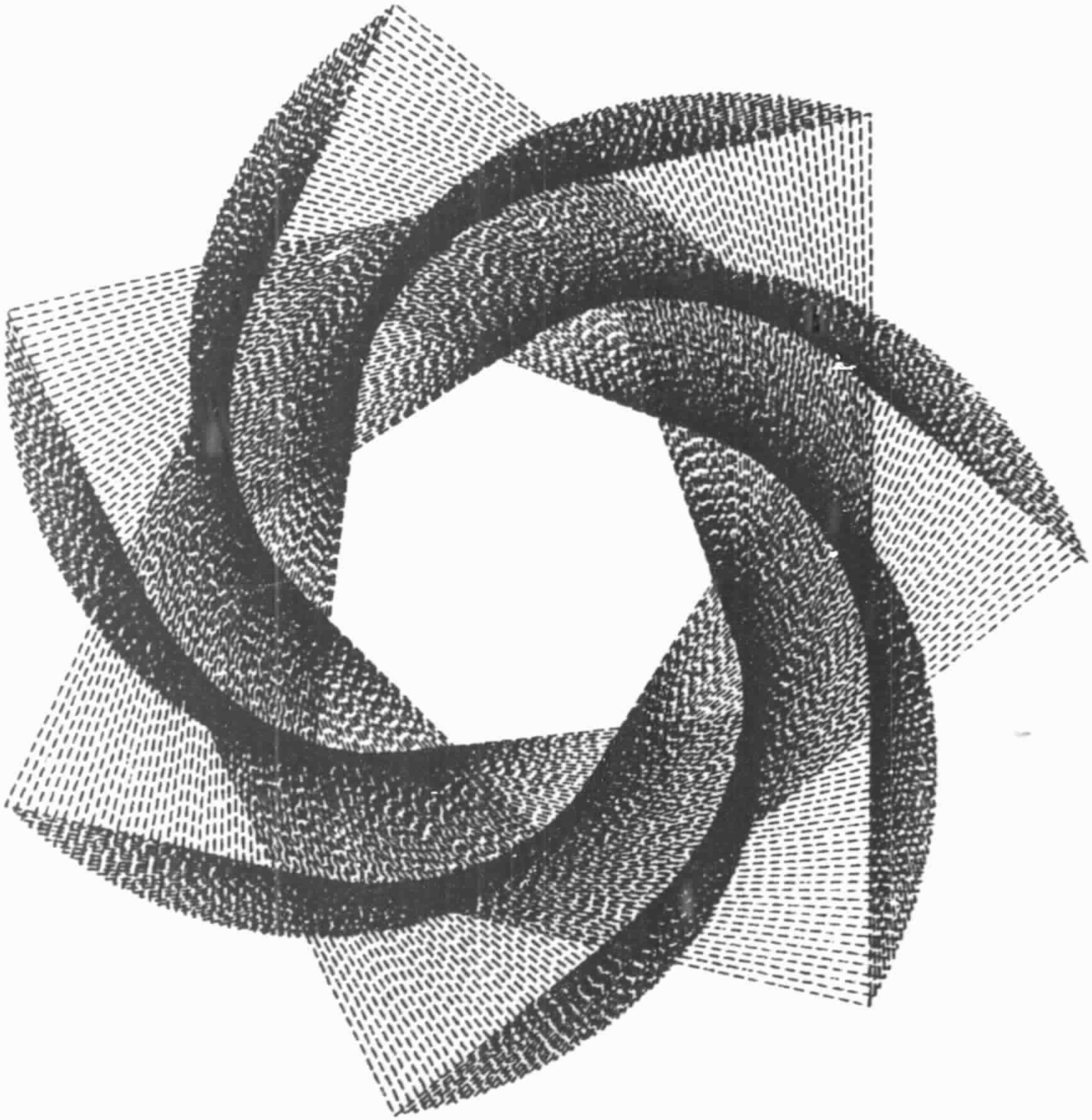
ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 15:40:44                 PAGE 001

0001            PROGRAM DEMOGR
          C
          C     DEMONSTRATION PROGRAM FOR GRAPHICS CAPABILITY
          C     OF TEKTRONIX TERMINALS
          C
          C     COMPILING/LINKING PROCEDURE:
          C         FORT/WARN DEMOGR/LIST
          C         LINK/MAP:DK:/LIB:SY:FPU/LIB:SY:WGCLIB DEMOGR
          C     NOTE:       /LIB:SY:FPU SHOULD NOT BE USED WHEN YOU WILL
          C         BE RUNNING THIS PROGRAM ON A REGULAR LSI-11
          C         (USE IT ONLY FOR LSI-11/23'S)
          C
0002            RO = 1500.
          C     DEFINE CENTER OF DISPLAY PATTERN (SLIGHTLY TO THE RIGHT
          C     OF THE ACTUAL CENTER OF THE SCREEN)
0003            IXCNTR = 2500
0004            IYCNTR = 1550
          C
          C     INITIALIZE GRAPHICS PARAMETERS
0005            CALL GRINIT(-1,0,0)
0006            CALL CHRSIZ(3)
0007            CALL ERASE
          C
0008      20    TYPE *,'Enter N (10-30000), SHRINKAGE (0-2),'
0009            TYPE *,'ANGLE INCREMENT (1-360), and LINE'
0010            TYPE *,'TYPE (1,97,98,99,100,104,112)'
0011            ACCEPT *,N,RFACTR,ANGINC,ITYPE
          C     CHANGE ANGLE INCREMENT FROM DEGREES TO RADIANS
0012            ANGINC = ANGINC * 3.141593 / 180.
          C     MOVE TO RIGHT SIDE (STARTING POINT)
0013            CALL MPLOT (IXCNTR+IFIX(RO),IYCNTR,0)
0014            DO 50 K=1,N
          C         DRAW LINE TO A POINT AT RADIUS SLIGHTLY LESS
          C         THAN PREVIOUS VALUE (DETERMINED BY SHRINKAGE)
          C         AND AT AN ANGLE OF ANGINC COUNTERCLOCKWISE
          C         FROM THE PREVIOUS POINT
0015                ANGLE = FLOAT(K) * ANGINC
0016                R = RO * (1. - RFACTR * FLOAT(K)/FLOAT(N))
0017                IX = R * COS(ANGLE) + IXCNTR
0018                IY = R * SIN(ANGLE) + IYCNTR
0019                CALL MPLOT(IX,IY,ITYPE)
0020      50    CONTINUE
0021            CALL MPLOT(0,300,-1)
0022            PAUSE 'Hit Return key'
0023            CALL ERASE
0024            GO TO 20
0025            END
```

```
FORTRAN IV      Storage Map for Program Unit DEMOGR

Local Variables, .PSECT $DATA, Size = 000050 (    20. words)

Name   Type  Offset     Name   Type  Offset     Name   Type  Offset
ANGINC R*4   000016     ANGLE  R*4   000026     ITYPE  I*2   000022
IX     I*2   000036     IXCNTR I*2   000004     IY     I*2   000040
IYCNTR I*2   000006     K      I*2   000024     N      I*2   000010
R      R*4   000032     RFACTR R*4   000012     RO     R*4   000000

Subroutines, Functions, Statement and Processor-Defined Functions:

Name   Type   Name   Type   Name   Type   Name   Type   Name   Type
CHRSIZ  R*4   COS     R*4   ERASE   R*4   FLOAT   R*4   GRINIT  R*4
IFIX    I*2   MPLOT   I*2   SIN     R*4
```

```
RUN DEMOGR
What is the model no. for the terminal you are using?  (Don't enter
  dash numbers.  Example: If you have a 4014-1, just enter 4014) ?4014
What is the model no. for the hard copy unit you are using?
  (If none is connected to your terminal, enter 0.) ?4631
Does your terminal have the enhanced graphics option?
  (Can it draw dotted and dashed lines?)  (Y=Yes) Y
```

```
Enter M (10-30000), SHRINKAGE (0-2),
ANGLE INCREMENT (1-360), and LINE
TYPE (1,97,98,99,100,104,112)
500,0.5,103,99
```



```
PAUSE -- Hit Return key
```

The third sample program is GRTEST. This demonstrates a more realistic plotting situation, although dummy data was used for this example. The program uses GRINIT, CHRSIZ, GRID, ANOTAT, XYPLOT, MPLOT, COPY, and BELL. Several other routines are also called by these. Refer to the descriptions of each subroutine for more details.

The operation of this program should be fairly clear from the program listing and the sample output. However, a few points are worth noting. First, when you use ANOTAT, you do not have to label every grid line. Every other vertical grid line was labelled in this example by specifying 10 horizontal segments for GRID, and 5 for ANOTAT. Second, when drawing a single line (such as the "50% of max." line here), two calls to MPLOT are usually required. The first call, with the third argument set to 0, moves the current position to one end of the line (without drawing anything on the screen). The second call to MPLOT, with the third argument set to a positive number, draws the line to the coordinates of the specified end point. In addition, when labelling the plot axes or performing other alphanumeric I/O, you should first call MPLOT with the third argument set to -1, then use a formatted write (or type) statement, with a "+" in the first print position in order to disable carriage control on that line. Otherwise, the line may be printed at a location different from the one you specified.

Finally, the calls to routine BELL demonstrate how a program may provide some auditory feedback to the operator, perhaps to verify that certain data is acceptable or unacceptable (using different sounds), or to let a user who is away from the terminal know that some action is needed. The program listing and sample output follow.

ORIG
OF PO

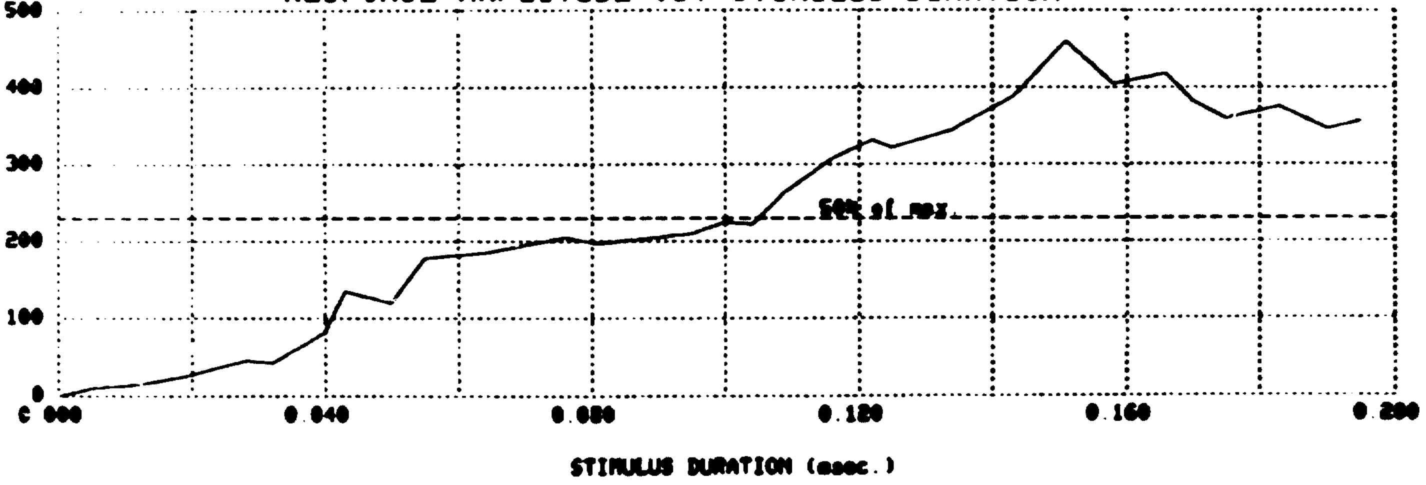
```
0001          PROGRAM GRTEST
      C
      C   DEMONSTRATE & TEST CERTAIN FEATURES OF TEKTRONIX
      C   GRAPHICS PACKAGE
      C
      C   COMPILING/LINKING PROCEDURE:
      C      FORT/WARN GRTEST/LIST
      C      LINK/MAP:DK:/LIB:SY:WGCLIB/LIB:SY:FPU GRTEST
      C
      C      NOTE: THE /LIB:SY:FPU SHOULD NOT BE USED UNLESS YOU
      C          WILL BE RUNNING THE PROGRAM ON AN LSI-11/23.
      C          DO NOT USE /LIB:SY:FPU WITH A REGULAR LSI-11.
      C
      C
0002          REAL X(30),Y(30)
      C
      C      DEFINE LIMITS OF PLOTTING AREA ON TERMINAL SCREEN
0003          DATA IL,IR,IB,IT / 400,3900,1000,2000 /
      C
      C      DUMMY X & Y VALUES TO PLOT
0004          DATA X / 0.,0.005,0.012,0.019,0.028,0.032,0.040,0.043,
             @            0.050,0.055,0.064,0.076,0.081,0.095,0.100,0.104,
             @            0.109,0.116,0.122,0.125,0.134,0.143,0.151,0.158,
             @            0.166,0.170,0.175,0.183,0.190,0.195 /
0005          DATA Y / 0.,10.,15.,25.,45.,42.,83.,135.,120.,178.,185.,
             @            205.,197.,210.,225.,222.,265.,308.,332.,322.,345.,
             @            387.,460.,405.,418.,382.,360.,375.,347.,357. /
      C
      C      INITIALIZE PARAMETERS FOR GRAPHICS ROUTINES (GET INFO.
      C      FROM USER)
0006          CALL GRINIT(-1,0,0)
      C
      C      SET CHARACTER SIZE TO #3 (NEXT TO SMALLEST)
0007          CALL CHRSIZ(3)
      C
      C      DRAW HORIZ. & VERT. GRID LINES
0008          CALL GRID (10,5,IL,IR,IB,IT,97)
      C
      C      LABEL (ANOTATE) GRID LINES WITH NUMERICAL USER UNITS
0009          CALL ANOTAT (5,5,IL,IR,IB,IT,0.,0.2,0.,500.)
      C
      C      PLOT THE DATA (CONNECT POINTS WITH STRAIGHT LINES)
0010          CALL XYPLOT (X,Y,30,IL,IR,IB,IT,0.,0.2,0.,500.,1)
      C
      C      GET MAX. Y VALUE & DRAW HORIZ. LINE ON PLOT
      C      AT LEVEL CORRESPONDING TO HALF THAT VALUE
0011          YMAX = 0.
0012          DO 50 K=1,30
0013              IF (YMAX .LT. Y(K))  YMAX = Y(K)
0015  50          CONTINUE
0016          IYHALF = ((0.5*YMAX) / 500.) * (IT-IB) + IB
      C      MOVE TO LEFT SIDE OF PLOT AT PROPER Y DISTANCE UP
0017          CALL MPLOT (IL,IYHALF,0)
      C      DRAW HORIZ. LINE TO RIGHT SIDE OF PLOT
```

```
0018            CALL MPLOT (IR,IYHALF,99)
          C     MOVE TO POSITION JUST ABOVE THE LINE WE DREW
0019            IX = (IL+IR) / 2 + 250
0020            CALL MPLOT (IX,IYHALF+10,-1)
          C     LABEL THE LINE
0021            TYPE 60
0022      60    FORMAT ('+50% of max.')
          C     NOTE THAT YOU MUST USE A + TO DISABLE CARRIAGE CONTROL
          C     IN THE FORMAT.  OTHERWISE, A LINE FEED WOULD BE SENT TO
          C     THE TERMINAL BEFORE TYPING THE LINE.
          C
          C     MOVE TO POSITION BELOW THE X-AXIS & LABEL IT
0023            CALL MPLOT (IL+1350,IB-200,-1)
0024            TYPE 70
0025      70    FORMAT ('+STIMULUS DURATION (msec.)')
          C
          C     CHANGE TO CHARACTER SIZE #2 (SECOND LARGEST)
0026            CALL CHRSIZ (2)
          C     MOVE TO POSITION ABOVE TOP OF PLOT & LABEL IT
0027            CALL MPLOT (IL+600,IT+30,-1)
0028            TYPE 80
0029      80    FORMAT ('+RESPONSE AMPLITUDE VS. STIMULUS DURATION')
          C
          C     MOVE TO BELOW BOTTOM OF PLOT
0030            CALL MPLOT(0,IB-300,-1)
          C
          C     MAKE A HARD COPY OF THE TERMINAL SCREEN
0031            CALL COPY (1)
          C
          C     MAKE SOME NOISES TO ALERT THE OPERATOR IN CASE
          C     HE/SHE WENT TO SLEEP.
0032            CALL BELL (20,8)
0033            CALL BELL (200,1)
0034            CALL BELL (5,60)
          C
0035            STOP
0036            END
```

```
FORTRAN IV      Storage Map for Program Unit GRTEST

Local Variables, .PSECT $DATA, Size = 000420 (   136. words)

Name    Type   Offset       Name    Type   Offset       Name    Type   Offset
IB      I*2    000364       IL      I*2    000360       IR      I*2    000362
IT      I*2    000366       IX      I*2    000402       IYHALF  I*2    000400
K       I*2    000376       YMAX    R*4    000372

Local and COMMON Arrays:

Name      Type     Section Offset   ------Size----- Dimensions
X       R*4      $DATA    000000   000170 (    60.) (30)
Y       R*4      $DATA    000170   000170 (    60.) (30)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
ANOTAT  R*4     BELL     R*4    CHRSIZ  R*4     COPY     R*4    GRID     R*4
GRINIT  R*4     MPLOT    I*2    XYPLOT  R*4
```

```
RUN GRTEST
What is the model no. for the terminal you are using?  (Don't enter
  dash numbers.  Example: If you have a 4014-1, just enter 4014) ?4014
What is the model no. for the hard copy unit you are using?
  (If none is connected to your terminal, enter 0.) ?4631
Does your terminal have the enhanced graphics option?
  (Can it draw dotted and dashed lines?)  (Y=Yes) Y
```

RESPONSE AMPLITUDE VS. STIMULUS DURATION



The fourth sample program is DISKRW. This routine demonstrates the use of subroutine DISKIO to read and write data to disk files. Refer to the program listing and sample output below. Note that this program is written specifically for use with the DEC RT-11 operating system, and is probably not adaptable to other systems. DISKIO calls a number of RT-11-specific subroutines, as indicated in its description, later in this document. DISKIO was written primarily to facilitate binary input and output with variable record length to random (not necessarily sequential) blocks in a disk file, using a Fortran main program. The normal Fortran I/O normally requires that direct access binary (unformatted) records all be of the same length. Records in sequential files may be of variable length, but are inconvenient and inefficient to access in a random-access fashion.

In general, the normal DEC Fortran I/O methods, using an OPEN statement followed by READ or WRITE statements, are preferrable if their restrictions are not a problem. Otherwise, you may use subroutine DISKIO. However, records which have been written with DISKIO may only be read using the same routine, and not with the usual Fortran I/O methods. Basically, DISKIO allows more efficient I/O than Fortran in many cases, both in terms of time and also storage space, but its files are not compatible with those produced by the usual Fortran I/O routines.

Operation of program DISKRW and DISKIO should be more clear after referring to the program listing, sample output, and output file dumps which follow.

```
FORTRAN IV      V02.1-1    Thu 01-Oct-81 15:12:58                 PAGE 001

0001            PROGRAM DISKRW
          C
          C PROGRAM TO VERIFY OPERATION & DEMONSTRATE USE OF SUBROUTINE DISKIO
          C FOR WRITING DATA TO & READING IT FROM DISK FILES.
          C SEE COMMENTS IN DISKIO FOR MORE INFO.
          C
          C WRITTEN BY:  WILLIAM G. CROSIER
          C DATE:           11 JUNE 1980
          C
          C COMPILING/LINKING SEQUENCE:
          C     FORT/NOSWAP DISKRW/LIST
          C     FORT DISKIO/LIST
          C     LINK/MAP:DK: DISKRW,DISKIO
          C
0002            INTEGER BUFR1(1024),BUFR2(1024),IERR
0003            BYTE FILNAM(12)
0004      10    DO 20 K=1,12
0005      20    FILNAM(K)=0
0006            TYPE 30
0007      30    FORMAT ('0ENTER COMPLETE FILE NAME FOR DISK I/O IN THE FORMAT',
              @' 'DEVFILNAMEXT','/' WHERE: 'DEV' IS THE 3-CHARACTER DEVICE CODE'/
              @8X,''FILNAM' IS THE 6-CHARACTER BASIC FILE NAME'/
              @8X,''EXT' IS THE 3-CHARACTER EXTENSION/FILE TYPE DESIRED (OPT.)'/
              @' EACH PORTION OF THE NAME SHOULD BE THE EXACT LENGTH SPECIFIED,'/
              @' WITH SPACES ADDED, IF NECESSARY, TO FORM THE PROPER LENGTH.'/
              @' DO NOT USE A COLON OR PERIOD TO SEPARATE PORTIONS OF THE FILE',
              @' NAME.'/ ' EXAMPLE:   DK A12345DAT' / ' FILE NAME ? ',$)
0008            ACCEPT 40, FILNAM
0009      40    FORMAT (12A1)
0010            TYPE *,'ENTER SIZE OF FILE (IN NO. OF 256-WORD BLOCKS)'
0011            ACCEPT *, NBLK
0012      45    TYPE *, 'ENTER NO. OF WORDS TO WRITE & READ (1-1024)'
0013            ACCEPT *, NWRDS
0014            TYPE *,'ENTER BLOCK NO. OF FILE WHERE I/O IS TO START'
0015            TYPE *,'(0=START AT 1ST BLOCK, 1=START AT 2ND, ETC.)'
0016            ACCEPT *, IBLK
0017            TYPE *,'LEAVE FILE OPEN AFTER I/O (1=YES) ? '
0018            ACCEPT *,LOPEN
0019            IWMODE = -1
0020            DO 50 K=1,NWRDS
0021            BUFR1(K)=K
0022      50    BUFR2(K)=0
0023            TYPE*,'NOW WRITING DATA TO DISK'
0024            CALL DISKIO(FILNAM,IWMODE,BUFR1,NWRDS,IBLK,NBLK,IERR)
0025            IF (IERR.NE.0) TYPE *,'ERROR CODE',IERR,'DURING WRITE'
          C FOR EXPLANATION OF ERROR CODES, SEE COMMENTS IN DISKIO
0027            IRMODE = 3
0028            IF (LOPEN .EQ. 1)  IRMODE = -3
0030            TYPE *,'NOW READING DATA FROM DISK'
0031            CALL DISKIO(FILNAM,IRMODE,BUFR2,NWRDS,IBLK,NDUMMY,IERR)
0032            IF (IERR.NE.0) TYPE *,'ERROR CODE',IERR,' DURING READ'
0034            TYPE 60, NWRDS,(BUFR2(K),K=1,NWRDS)
0035      60    FORMAT ('0DATA READ FROM DISK (SHOULD BE CONSECUTIVE INTEGERS'
              @/' FROM 1 THROUGH',I5,'):'/(10I7))
```

```
FORTRAN IV      V02.1-1    Thu 01-Oct-81 15:12:58                PAGE 002

0036         IF (LOPEN .EQ. 1)  GO TO 45
0038         TYPE *,'MORE (1=YES) ? '
0039         ACCEPT *, MORE
0040         IF (MORE .EQ. 1)  GO TO 10
0042         STOP
0043         END
```

```
FORTRAN IV      Storage Map for Program Unit DISKRW

Local Variables, .PSECT $DATA, Size = 010050 ( 2068. words)

Name    Type  Offset      Name    Type  Offset      Name    Type  Offset
IBLK    I*2   010034      IERR    I*2   010024      IRMODE  I*2   010042
IWMODE  I*2   010040      K       I*2   010026      LOPEN   I*2   010036
MORE    I*2   010046      NBLK    I*2   010030      NDUMMY  I*2   010044
NWRDS   I*2   010032

Local and COMMON Arrays:

Name      Type    Section Offset  ------Size----- Dimensions
BUFR1   I*2     $DATA   000000  004000 ( 1024.) (1024)
BUFR2   I*2     $DATA   004000  004000 ( 1024.) (1024)
FILNAM  L*1     $DATA   010000  000014 (    6.) (12)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
DISKIO   R*4
```

```
RUN DISKRW

ENTER COMPLETE FILE NAME FOR DISK I/O IN THE FORMAT 'DEVFILNAMEXT'.
WHERE  'DEV' IS THE 3-CHARACTER DEVICE CODE
       'FILNAM' IS THE 6-CHARACTER BASIC FILE NAME
       'EXT' IS THE 3-CHARACTER EXTENSION/FILE TYPE DESIRED (OPT.)
EACH PORTION OF THE NAME SHOULD BE THE EXACT LENGTH SPECIFIED.
WITH SPACES ADDED. IF NECESSARY. TO FORM THE PROPER LENGTH.
DO NOT USE A COLON OR PERIOD TO SEPARATE PORTIONS OF THE FILE NAME
EXAMPLE:   DK A12345DAT
FILE NAME ? DV1A00002DAT
ENTER SIZE OF FILE (IN NO. OF 256-WORD BLOCKS)
2
ENTER NO  OF WORDS TO WRITE & READ (1-1024)
32
ENTER BLOCK NO  OF FILE WHERE I/O IS TO START
(0=START AT 1ST BLOCK. 1=START AT 2ND. ETC.)
1
LEAVE FILE OPEN AFTER I/O (1=YES) ?
1
NOW WRITING DATA TO DISK
NOW READING DATA FROM DISK

DATA READ FROM DISK (SHOULD BE CONSECUTIVE INTEGERS
FROM 1 THROUGH   32):
      1        2        3        4        5        6        7        8        9       10
     11       12       13       14       15       16       17       18       19       20
     21       22       23       24       25       26       27       28       29       30
     31       32
ENTER NO  OF WORDS TO WRITE & READ (1-1024)
48
ENTER BLOCK NO  OF FILE WHERE I/O IS TO START
(0=START AT 1ST BLOCK. 1=START AT 2ND. ETC.)
0
LEAVE FILE OPEN AFTER I/O (1=YES) ?
0
NOW WRITING DATA TO DISK
NOW READING DATA FROM DISK

DATA READ FROM DISK (SHOULD BE CONSECUTIVE INTEGERS
FROM 1 THROUGH   48):
      1        2        3        4        5        6        7        8        9       10
     11       12       13       14       15       16       17       18       19       20
     21       22       23       24       25       26       27       28       29       30
     31       32       33       34       35       36       37       38       39       40
     41       42       43       44       45       46       47       48
MORE (1=YES) ?
0

STOP --

.DUMP/NOASCII DY1:A00002.DAT
```

```
DY1:A00002.DAT/N
BLOCK NUMBER  00000
000/ 000001 000002 000003 000004 000005 000006 000007 000010
020/ 000011 000012 000013 000014 000015 000016 000017 000020
040/ 000021 000022 000023 000024 000025 000026 000027 000030
060/ 000031 000032 000033 000034 000035 000036 000037 000040
100/ 000041 000042 000043 000044 000045 000046 000047 000050
120/ 000051 000052 000053 000054 000055 000056 000057 000060
140/ 000000 000000 000000 000000 000000 000000 000000 000000
160/ 000000 000000 000000 000000 000000 000000 000000 000000
200/ 000000 000000 000000 000000 000000 000000 000000 000000
220/ 000000 000000 000000 000000 000000 000000 000000 000000
240/ 000000 000000 000000 000000 000000 000000 000000 000000
260/ 000000 000000 000000 000000 000000 000000 000000 000000
300/ 000000 000000 000000 000000 000000 000000 000000 000000
320/ 000000 000000 000000 000000 000000 000000 000000 000000
340/ 000000 000000 000000 000000 000000 000000 000000 000000
360/ 000000 000000 000000 000000 000000 000000 000000 000000
400/ 000000 000000 000000 000000 000000 000000 000000 000000
420/ 000000 000000 000000 000000 000000 000000 000000 000000
440/ 000000 000000 000000 000000 000000 000000 000000 000000
460/ 000000 000000 000000 000000 000000 000000 000000 000000
500/ 000000 000000 000000 000000 000000 000000 000000 000000
520/ 000000 000000 000000 000000 000000 000000 000000 000000
540/ 000000 000000 000000 000000 000000 000000 000000 000000
560/ 000000 000000 000000 000000 000000 000000 000000 000000
600/ 000000 000000 000000 000000 000000 000000 000000 000000
620/ 000000 000000 000000 000000 000000 000000 000000 000000
640/ 000000 000000 000000 000000 000000 000000 000000 000000
660/ 000000 000000 000000 000000 000000 000000 000000 000000
700/ 000000 000000 000000 000000 000000 000000 000000 000000
720/ 000000 000000 000000 000000 000000 000000 000000 000000
740/ 000000 000000 000000 000000 000000 000000 000000 000000
760/ 000000 000000 000000 000000 000000 000000 000000 000000
```

```
BLOCK NUMBER  00001
000/ 000001 000002 000003 000004 000005 000006 000007 000010
020/ 000011 000012 000013 000014 000015 000016 000017 000020
040/ 000021 000022 000023 000024 000025 000026 000027 000030
060/ 000031 000032 000033 000034 000035 000036 000037 000040
100/ 000000 000000 000000 000000 000000 000000 000000 000000
120/ 000000 000000 000000 000000 000000 000000 000000 000000
140/ 000000 000000 000000 000000 000000 000000 000000 000000
160/ 000000 000000 000000 000000 000000 000000 000000 000000
200/ 000000 000000 000000 000000 000000 000000 000000 000000
220/ 000000 000000 000000 000000 000000 000000 000000 000000
240/ 000000 000000 000000 000000 000000 000000 000000 000000
260/ 000000 000000 000000 000000 000000 000000 000000 000000
300/ 000000 000000 000000 000000 000000 000000 000000 000000
320/ 000000 000000 000000 000000 000000 000000 000000 000000
340/ 000000 000000 000000 000000 000000 000000 000000 000000
360/ 000000 000000 000000 000000 000000 000000 000000 000000
400/ 000000 000000 000000 000000 000000 000000 000000 000000
420/ 000000 000000 000000 000000 000000 000000 000000 000000
440/ 000000 000000 000000 000000 000000 000000 000000 000000
460/ 000000 000000 000000 000000 000000 000000 000000 000000
500/ 000000 000000 000000 000000 000000 000000 000000 000000
520/ 000000 000000 000000 000000 000000 000000 000000 000000
540/ 000000 000000 000000 000000 000000 000000 000000 000000
560/ 000000 000000 000000 000000 000000 000000 000000 000000
600/ 000000 000000 000000 000000 000000 000000 000000 000000
620/ 000000 000000 000000 000000 000000 000000 000000 000000
640/ 000000 000000 000000 000000 000000 000000 000000 000000
660/ 000000 000000 000000 000000 000000 000000 000000 000000
700/ 000000 000000 000000 000000 000000 000000 000000 000000
720/ 000000 000000 000000 000000 000000 000000 000000 000000
740/ 000000 000000 000000 000000 000000 000000 000000 000000
760/ 000000 000000 000000 000000 000000 000000 000000 000000
```

GRAPHICS

SUBROUTINES

Subroutine GRINIT

This routine initializes the various parameters used by other graphics routines, and should be called before any of the other graphics routines. It is not needed if only the data acquisition routines ISAMPA, DAQISR, DAQ or miscellaneous routines BELL, WAIT, or DISKIO are used.

Information on the parameters or arguments to be passed to GRIWIT appears in the program listing below. When writing a particular program, if you will always be using it with the same type of terminal and hard copy unit, then you may specify their model numbers in the argument list so that GRINIT may define its internal parameters. However, if you may use different types of terminals, then you should set the first argument to -1 so that GRINIT will query the user at execution time about the type of terminal and hard copy unit to be used. The program listing follows.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:21:48                PAGE 001

0001          SUBROUTINE GRINIT(ITERM,IHDCP,IENH)
      C
      C WRITTEN BY:       WILLIAM G. CROSIER
      C REVISED:          28 SEPT. 1981
      C
      C PURPOSE:          INITIALIZE PARAMETERS DEFINING TERMINAL
      C           CHARACTERISTICS FOR THE VARIOUS GRAPHICS ROUTINES
      C ARGUMENTS:
      C       ITERM = MODEL NUMBER OF TERMINAL BEING USED.
      C           IF =-1, THEN USER WILL BE QUERIED DURING PROGRAM
      C           EXECUTION ABOUT TERMINAL & HARD COPY UNIT.
      C       IHDCP = MODEL NUMBER OF HARD COPY UNIT BEING USED
      C           SET IHDCP=0 IF NO HARD COPY UNIT IS BEING USED
      C       IENH = FLAG FOR ENHANCED GRAPHICS OPTION FOR TERMINAL.
      C           SET IENH=1 IF YOU HAVE A 4014/4015 WITH THIS OPTION.
      C           OTHERWISE, SET IENH=0
      C
      C       NOTE: IHDCP & IENH ARE BOTH IGNORED IF ITERM=-1.  IN THIS CASE,
      C           THEN THE USER WILL BE ASKED TO SUPPLY THE APPROPRIATE
      C           INFO. FOR THE TERMINAL WHEN THIS ROUTINE IS CALLED.
      C
      C
0002          LOGICAL*1 ANSWER
0003          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
             @                TIMERA,TIMHDC
      C
0004          ITERM1 = ITERM
0005          IHDCP1 = IHDCP
0006          IENH1 = IENH
0007          IF (ITERM1 .GT. 0)  GO TO 60
      C
      C QUERY USER ABOUT TYPE OF TERMINAL & HARD COPY UNIT BEING USED
0009          TYPE 20
0010   20     FORMAT (' What is the model no. for the terminal you are ',
             @                'using?  (Don''t enter' / '   dash numbers.  Example  ',
             @                ' If you have a 4014-1, just enter 4014) ?',$)
0011          ACCEPT *,ITERM1
0012          TYPE 30
0013   30     FORMAT (' What is the model no. for the hard copy unit',
             @                ' you are using?' / '   (If none is connected to',
             @                ' your terminal, enter 0.) ?',$)
0014          ACCEPT *, IHDCP1
0015          IF (ITERM1.NE.4014 .AND. ITERM1.NE.4015)  GO TO 60
0017          TYPE 40
0018   40     FORMAT (' Does your terminal have the enhanced graphics',
             @                ' option?' / '   (Can it draw dotted and dashed',
             @                ' lines?)  (Y=Yes) ',$)
0019          ACCEPT 50, ANSWER
0020   50     FORMAT (A1)
0021          IENH1 = 0
0022          IF (ANSWER .EQ. 'Y')  IENH1 = 1
      C
0024   60     MCHRSZ = 1
      C IF USING A 4014/4015 TERMINAL, MULTIPLE CHARACTER SIZES ARE AVAIL.
```

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:21:48                PAGE 002

0025          IF (ITERM1.NE.4014 .AND. ITERM1.NE.4015)  MCHRSZ = 0
      C
      C SET TIME TO ALLOW FOR ERASING SCREEN = 1.5 SEC.
0027          TIMERA = 1.5
      C SET DEFAULT HARD COPY TIME = 20 SEC.
0028          TIMHDC = 20.0
0029          IF (IHDCP1 .LE. 0)  TIMHDC=0          !NO HARD COPY UNIT AVAIL.
0031          IF (IHDCP1 .EQ. 4631)  TIMHDC=10.0  !10 SEC FOR MODEL 4631
0033          IENHAN = IENH1
0034          RETURN
0035          END
      C
```

```
FORTRAN IV      Storage Map for Program Unit GRINIT

Local Variables, .PSECT $DATA, Size = 000016 (     7. words)

Name    Type   Offset       Name    Type   Offset       Name    Type   Offset
ANSWER  L*1    000006       IENH    I*2 @ 000004       IENH1   I*2    000014
IHDCP   I*2 @ 000002       IHDCP1  I*2    000012       ITERM   I*2 @ 000000
ITERM1  I*2    000010

COMMON Block /GRFCOM/, Size = 000022 (     9. words)

Name    Type   Offset       Name    Type   Offset       Name    Type   Offset
MCHRSZ  I*2    000000       LSIZE   I*2    000002       IWIDTH  I*2    000004
IHIGHT  I*2    000006       IENHAN  I*2    000010       TIMERA  R*4    000012
TIMHDC  R*4    000016
```

Subroutine MPLOT

This is the basic line-drawing subroutine. It is used by all of the other graphics routines which draw lines or move the cursor, and it may be called directly by the user's program also.

The arguments or parameters for this routine are described in the program listing below. Refer there for more information. In addition, there are further details, concerning the various line types available in a 4014 with enhanced graphics, in the Tektronix manuals for the 4014 terminal. If you are not using a 4014 or 4015, then GRINIT will cause all requests for line types other than 1 (normal solid line) to be responded to as if line type 1 was specified. In this way, you do not need to worry about the presence of the optional enhanced graphics hardware in your terminal when you write the software, because if the hardware is not present then solid lines will be drawn regardless of the line type you specify with parameter IPEN.

Besides drawing a line, MPLOT can also be used to move the cursor immediately before plotting by setting parameter IPEN to 0. Also, it can be used to move the cursor before typing alphanumeric data by setting IPEN to -1. Refer to the third sample main program (GRTEST) in an earlier section of this report for examples of this. The second sample pgoram (DEMOGR) may also be helpful when using MPLOT.

If you are not using a 4014/4015 terminal with enhanced graphics, then you will have only 1024 point horizontal resolution horizontally and 780 point resolution vertically, but the same coordinates will still refer to the same point on the screen. Thus, an X coordinate of 2048 and a Y coordinate of 1560 will always refer to the center of the screen, regardless of the type of Tektronix terminal you are using.

If you want to modify this routine for use with another type of terminal or with an X-Y plotter, then several parts of the subroutine will need to be changed. First, Tektronix requires that the X and Y coordinates be split up and sent in a particular sequence. This sequence, under the heading "Draw Vector" in the program listing, would probably need to be changed for a diffe·ent terminal or plotter. Second, many plotters require a delay after transmitting the coordinates for a new point so that the pen has time to move. For this purpose, you can use a call to subroutine WAIT. If an analog X-Y recorder was to be used rather than a digital plotter, then the X and Y coordinates would need to be sent to the plotter via two IPOKE calls (refer to the System Subroutine Library routines) to a pair of digital-to-analog converters. The outputs of these may have to be slowed down with a pair of matched R-C networks in order to avoid too-rapid changes in the pen position, if the recorder so requires. Using this routine with a digital plotter should be much simpler, however. Many digital plotters also have built-in character generators for drawing the standard ASCII character set also, while analog X-Y recorders do not have this capability.

Note that when using a line type of 112 (for the write-thru mode), the lines which are drawn are faint, and do not store on the CRT screen. Normally, when using this line type you should refresh the display by repeating the plotting of the appropriate lines. Ideally, the lines should be redrawn at least 50 times per second in order to prevent flickering and to make the display easier to see. However, slower refresh rates may be necessary if a large number of lines have to be redrawn with each repetition. In addition, static displays (with line types other than 112) may be combined with dynamic displays (line Type 112) at the same time. That way you only have to refresh the lines whose positions change.

One problem when using MPLOT (and any routines which call it) has appeared in some systems when using the RT-11 Version 4 Foreground/Background monitor. The problem appears to be in the RT-11 terminal handler (TT.SYS) or in the System Subroutine Library routine ITTOUR, since the difficulty has never appeared with the RT-11 Single Job monitor. If your plots do not come our correctly but appear garbled, try running your program under the Single Job monitor (RT11SJ) rather than the Foreground/Background one (RT11FB).

MPLOT requires the following routine from the System Subroutine Library:

ITTOUR

A listing of MPLOT follows.

```
0001          SUBROUTINE MPLOT(IX,IY,IPEN)
        C-----------------------------------------------------------------
        C
        C  AUTHOR:         WILLIAM G. CROSIER      TECHNOLOGY INCORPORATED
        C  REVISED         25 SEPT. 1981
        C  BASED ON ROUTINE WRITTEN BY CHUCK MANN
        C
        C  PURPOSE: TO DRAW LIGHT OR DARK VECTORS FROM THE CURSOR'S
        C    (PEN'S) PRESENT LOCATION TO THE COORDINATES PASSED.
        C
        C  ARGUMENTS:
        C     IX = X COORDINATE IN TEKTRONIX SCREEN UNITS (0 TO 4095)
        C     IY = Y COORDINATE IN TEKTRONIX SCREEN UNITS (0 TO 3120)
        C     IPEN CONTROLS PLOTTING AS FOLLOWS:
        C         IPEN = 0 MOVES POSITION TO (IX,IY) WITHOUT
        C                 DRAWING A LINE (DARK VECTOR)
        C         IPEN < 0 MOVES POSITION & CHANGES TO ALPHA MODE
        C         IPEN > 0 DRAWS A VISIBLE LINE TO (IX,IY)
        C                 IF IPEN = 1, THEN NORMAL SOLID LINE
        C                 IF IPEN = 97, THEN DOTTED LINE (ON 4014)
        C                         = 98, THEN DOT-DASH LINE
        C                         = 99, THEN SHORT DASH LINE
        C                         = 100, THEN LONG DASH LINE
        C                         = 104, THEN DEFOCUSED Z-AXIS
        C                                (SLIGHTLY WIDER LINE)
        C                         = 112, THEN WRITE-THRU (NON-STORE)
        C
        C                 (THE VARIOUS DOTTED & DASHED LINES
        C                 WILL ONLY BE PRODUCED ON A 4014
        C                 TERMINAL WITH ENHANCED GRAPHICS OPTION.)
        C
        C--------------------------------------------------------------------
        C
0002          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
             @                TIMERA,TIMHDC
        C
0003          DATA LSIZE,IWIDTH,IHIGHT,IENHAN,TIMERA,TIMHDC
             @                / 1,56,88,0,10.0,1.5 /
0004          DATA MINX,MAXX,MINY,MAXY /0,4095,0,3120/
0005          DATA IPEN1 /0/
        C
0006          IX1 = IX
0007          IY1 = IY
0008          IF (IX .LT. MINX) IX1=MINX
0010          IF (IX .GT. MAXX) IX1=MAXX
0012          IF (IY .LT. MINY) IY1=MINY
0014          IF (IY .GT. MAXY) IY1=MAXY
0016          IF (IPEN.GT.0) GO TO 10
        C
        C     DRAW DARK VECTOR (MOVE POSITION)
0018    100   IF (ITTOUR("35).NE.0) GO TO 100            !SEND GS CHAR.
0020    10    IF (IPEN.LE.0 .OR. IENHAN.EQ.0 .OR. IPEN.EQ.IPEN1)
             @                GO TO 30
        C
```

```
      C      CHANGE TO SELECTED PLOTTING MODE IF USING TERMINAL WITH
      C      ENHANCED GRAPHICS
0022         LINTYP = IPEN
0023         IF (IPEN .LT. 96)  LINTYP = 96                  !SOLID LINE?
0025  20     IF (ITTOUR(27).NE.0) GO TO 20                   !SEND ESC
0027  25     IF (ITTOUR(LINTYP).NE.0) GO TO 25   !SET LINE TYPE
0029  30     CONTINUE
0030         IPEN1 = IPEN                                    !RESET PREV. VALUE
      C
      C      DRAW VECTOR
      C
      C      SEND HIGH ORDER Y BYTE
0031         ICH='40+IY1/128
0032  105    IF (ITTOUR(ICH).NE.0) GO TO 105
      C      SEND EXTRA BYTE (2 LSB'S OF X & Y)
0034         ICH='140 + ((IY1.AND.'3)*4) + (IX1.AND.'3)
0035  108    IF (ITTOUR(ICH) .NE. 0)  GO TO 108
      C      SEND LOW ORDER Y BYTE
0037         ICH='140+((IY1/4).AND.'37)
0038  110    IF (ITTOUR(ICH).NE.0) GO TO 110
      C      SEND HIGH ORDER X BYTE
0040         ICH='40+IX1/128
0041  115    IF(ITTOUR(ICH).NE.0) GO TO 115
      C      SEND LOW ORDER X BYTE
0043         ICH='100+((IX1/4).AND.'37)
0044  120    IF(ITTOUR(ICH).NE.0) GO TO 120
0046         IF (IPEN .GE. 0)  GO TO 900
      C
      C      CHANGE TO ALPHA MODE IF IPEN IS LESS THAN 0
0048  200    IF (ITTOUR('37) .NE. 0)  GO TO 200
0050  900    RETURN
0051         END
      C

FORTRAN IV      Storage Map for Program Unit MPLOT

Local Variables, .PSECT $DATA, Size = 000030 (    12. words)

Name    Type    Offset       Name    Type    Offset       Name    Type    Offset
ICH     I*2     000026       IPEN    I*2 @ 000004         IPEN1   I*2     000016
IX      I*2 @ 000000         IX1     I*2     000020       IY      I*2 @ 000002
IY1     I*2     000022       LINTYP  I*2     000024       MAXX    I*2     000010
MAXY    I*2     000014       MINX    I*2     000006       MINY    I*2     000012


COMMON Block /GRFCOM/, Size = 000022 (     9. words)

Name    Type    Offset       Name    Type    Offset       Name    Type    Offset
MCHRSZ  I*2     000000       LSIZE   I*2     000002       IWIDTH  I*2     000004
IHIGHT  I*2     000006       IENHAN  I*2     000010       TIMERA  R*4     000012
TIMHDC  R*4     000016

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
ITTOUR   I*2
```

Subroutine COPY

This short routine causes a hard copy unit (if one is present) to make a paper copy of whatever appears on the Tektronix terminal screen. After waiting for a fixed length of time, or until the user strikes the Return key (depending on the parameter IFLAG), the routine returns to the calling program. More details are given in the program listing.

COPY requires the following subroutines:

ERASE
WAIT
ITTOUR (from the System Subroutine Library)

A listing of routine COPY follows.

ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV      V02.1-1    Wed 23-Dec-P  10:22:12                PAGE 001

0001          SUBROUTINE COPY(IFLAG)
        C-----------------------------------------------------------
        C
        C     PURPOSE: TO MAKE A HARD COPY OF THE TEKTRONIX SCREEN.
        C         IF: IFLAG=0, WAIT FOR USER TO TYPE <CR>, THEN
        C                      ERASE SCREEN.
        C                  =1, RETURN AFTER MAKING COPY (DON'T
        C                      WAIT FOR RESPONSE, DON'T ERASE)
        C                  =2, DON'T WAIT FOR RESPONSE AFTER MAKING
        C                         COPY, BUT ERASE SCREEN.
        C
        C-----------------------------------------------------------------
        C
0002          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
             @            TIMERA,TIMHDC
        C
        C  MAKE HARD COPY OF THE SCREEN IF HARD COPY UNIT IS AVAIL.
0003          IF (TIMHDC .LE. 0)  GO TO 50
0005    230   IF (ITTOUR(27).NE.0) GO TO 230                !SEND ESC
0007    235   IF (ITTOUR(23).NE.0) GO TO 235                !SEND ETB
        C
        C  WAIT FOR SCREEN TO COPY
0009          CALL WAIT (TIMHDC,0)                          !WAIT TIMHDC SEC.
0010    50    IF (IFLAG.EQ.1) RETURN
0012          IF(IFLAG.EQ.2) GO TO 200
        C
        C  WAIT FOR USER TO RESPOND
0014          PAUSE 'HIT RETURN KEY TO CONTINUE'
        C
        C  CLEAR THE SCREEN
0015    200   CALL ERASE
0016          RETURN
0017          END
        C


FORTRAN IV      Storage Map for Program Unit COPY

Local Variables, .PSECT $DATA, Size = 000002 (     1. words)

Name    Type  Offset       Name    Type  Offset       Name    Type  Offset
IFLAG   I*2 @ 000000


COMMON Block /GRFCOM/, Size = 000022 (     9. words)

Name    Type  Offset       Name    Type  Offset       Name    Type  Offset
MCHRSZ  I*2   000000       LSIZE   I*2   000002       IWIDTH  I*2   000004
IHIGHT  I*2   000006       IENHAN  I*2   000010       TIMERA  R*4   000012
TIMHDC  R*4   000016

Subroutines, Functions, Statement and Processor-Defined Functions:

Name     Type   Name     Type   Name     Type   Name     Type   Name     Type
ERASE    R*4    ITTOUR   I*2    WAIT     R*4
```

Subroutine ERASE

This simple routine is used to erase or clear the Tektronix terminal screen. It sends an escape followed by a form feed character, waits a short time for the erase process to finish, then returns to the calling program.

ERASE requires the following subroutines:

WAIT
ITTOUR (from the System Subroutine Library)

A listing of routine ERASE follows.

ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV       V02.1-1    Wed 23-Dec-81 10:22:21                 PAGE 001

0001            SUBROUTINE ERASE
        C
        C       ERASE (CLEAR) THE SCREEN ON THE TEKTRONIX TERMINAL
        C
0002            COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
              @            TIMERA,TIMHDC
        C
0003    10      IF (ITTOUR(27) .NE. 0) GO TO 10                 !SEND ESC
0005    20      IF (ITTOUR(12) .NE. 0) GO TO 20                 !SEND FF
0007            CALL WAIT(TIMERA,0)                      !WAIT TIMERA SEC.
        C
0008            RETURN
0009            END
        C
```

```
FORTRAN IV       Storage Map for Program Unit ERASE

COMMON Block /GRFCOM/, Size = 000022 (     9. words)

Name    Type   Offset       Name    Type   Offset       Name    Type   Offset
MCHRSZ  I*2    000000       LSIZE   I*2    000002       IWIDTH  I*2    000004
IHIGHT  I*2    000006       IENHAN  I*2    000010       TIMERA  R*4    000012
TIMHDC  R*4    000016

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
ITTOUR   I*2    WAIT     R*4
```

Subroutine CHRSIZ

This subroutine can be used to set the character size for those Tektronix terminals with multiple sizes available (the 4014/4015). This routine should be called before the other graphics routines which print alphanumeric characters, such as ANOTAT (or XYPLOT with ICODE equal to 0). It can also be used before regular type statements in order to change to smaller or larger characters for page headings, etc. Once CHRSIZ is called, the new character size stays in effect until CHRSIZ is called again with a different size specified, even if the screen is erased or if you change between alpha and graph modes. The parameter/argument ISIZE sets the character size as described in the program listing.

CHRSIZ requires the following subroutines from the System Subroutine Library:

ITTCUR

A listing of CHRSIZ follows.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:22:30                PAGE 001

0001          SUBROUTINE CHRSIZ(ISIZE)
      C
      C       ROUTINE TO SET CHARACTER SIZE ON TEKTRONIX 4014 SCREEN
      C
      C       ARGUMENT ISIZE CONTROLS CHAR. SIZE AS FOLLOWS:
      C           ISIZE = 1 SELECTS LARGEST CHARACTERS
      C           ISIZE = 2 SELECTS MEDIUM-LARGE CHAR.
      C           ISIZE = 3 SELECTS MEDIUM-SMALL CHAR.
      C           ISIZE = 4 SELECTS SMALLEST CHAR.
      C
0002          LOGICAL*1 ICODE(4)
0003          INTEGER ISIZE,IW(4),IH(4)
0004          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
             @                TIMERA,TIMHDC
      C
0005          DATA MCHRSZ / 1 /
0006          DATA ICODE / '8','9',':',';' /
0007          DATA IW / 56,51,34,31 /,IH / 88,82,53,48 /
      C
0008          IF (ISIZE.LT.1 .OR. ISIZE.GT.4 .OR. MCHRSZ.EQ.0)  GO TO 99
0010    20    IF (ITTOUR(27) .NE. 0) GO TO 20              !SEND ESCAPE
0012    30    IF (ITTOUR(ICODE(ISIZE)) .NE. 0)  GO TO 30  !SEND CHAR.
0014          IWIDTH = IW(ISIZE)                      !WIDTH OF A CHAR.
0015          IHIGHT = IH(ISIZE)                      !HEIGHT OF A CHAR.
0016          LSIZE = ISIZE
      C
0017    99    RETURN
0018          END
      C
```

```
FORTRAN IV      Storage Map for Program Unit CHRSIZ

Local Variables, .PSECT $DATA, Size = 000026 (    11. words)

Name    Type  Offset      Name    Type  Offset      Name    Type  Offset
ISIZE   I*2 @ 000000


COMMON Block /GRFCOM/, Size = 000022 (     9. words)

Name    Type  Offset      Name    Type  Offset      Name    Type  Offset
MCHRSZ  I*2   000000      LSIZE   I*2   000002      IWIDTH  I*2   000004
IHIGHT  I*2   000006      IENHAN  I*2   000010      TIMERA  R*4   000012
TIMHDC  R*4   000016

Local and COMMON Arrays:

Name      Type    Section Offset   ------Size----- Dimensions
ICODE    L*1      $DATA   000002   000004 (     2.) (4)
IH       I*2      $DATA   000016   000010 (     4.) (4)
IW       I*2      $DATA   000006   000010 (     4.) (4)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
ITTOUR   I*2
```

Subroutine GINPUT

This routine is used with those terminals such as the 4010 and 4014, having user-controlled crosshairs for interactive graphics input from the user. Each time this routine is called, the terminal switches to Graphics Input mode and displays the user-controlled crosshairs. The position of these two lines (one horizontal and one vertical) may be changed by the operator by turning a pair of thumbwheels by the keyboard. The crosshair lines are dim and do not store on the terminal screen. By using this routine, the program can allow the user to select a particular point on the display (perhaps one previously plotted with ARYPLT or XYPLOT) and can then easily get the coordinates of that point from the user. The subroutine does this by waiting until the user has positioned the crosshairs as desired. Then he or she can strike a single key on the keyboard and then the Return key. When this is done, the terminal automatically transmits the first character that was struck along with the X and Y coordinates of the crosshairs to the computer. Subroutine GINPUT then returns the ASCII equivalent of the character and the crosshair coordinates to the calling program through the argument list.

Each time GINPUT is called, a single character and the coordinates of one point are sent to the program. The character struck does not appear on the screen. (The terminal "bypass" circuitry keeps this from happening.) The character transmitted can be used as a code for the program to instruct it what function should be performed next, or it can be used for any other purpose by the calling program. Although both the X and Y coordinates are always passed to the program, frequently only one or the other is needed. In that case the user can be instructed to ignore the crosshair line that is not needed (for example, the horizontal one), and use only the other one to pick out the feature of interest on the screen. This can speed up the program's operation by cutting in half the amount of time the operator must spend in positioning the crosshairs. This is especially useful when one is selecting a number of points from a plot of sampled analog data (such as may be produced by ARYPLT) versus time, for instance, since only the vertical crosshair need be used to point out the times of interest. The program can then look up the Y coordinate from the sampled data point in memory which corresponds to that time or X coordinate.

Although the graphical output routines in this package can plot points with 4096 point resolution if a 4014/4015 terminal with the enhanced graphics option is used, the Tektronix terminals are all limited to 1024 point resolution for graphical input. Therefore, in detailed plots, it may be impossible to always select the exact point of interest with the GINPUT routine. In general, your program should take the coordinates returned by GINPUT and find the point which you plotted that is closest to the returned coordinates. Even then, it may be impossible to always select a single given point from among several very closely spaced points. Usually, however, the error in selecting among closely spaced points is not noticeable.

Routine GINPUT requires the following subroutine from the System Subroutine Library:

ITTOUR

The listing for GINPUT follows.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:22:49                 PAGE 001

0001          SUBROUTINE GINPUT(ICHAR,IX,IY)
        C
        C     SWITCH TO GRAPHIC INPUT MODE,DISPLAY USER-CONTROLLED
        C     CROSSHAIRS.  WAIT FOR USER TO TYPE IN A CHARACTER,
        C    & RETURN THAT CHARACTER (ICHAR), AND THE X AND Y
        C     COORDINATES OF THE CROSSHAIRS TO THE CALLING
        C     PROGRAM.
        C
0002          INTEGER ICHAR,HIGHX,LOWX,HIGHY,LOWY
        C
        C     SWITCH TO GRAPHIC INPUT MODE, DISPLAY CROSSHAIR
        C
0003    20    IF (ITTOUR(27) .NE. 0) GO TO 20               !SEND ESCAPE
0005    30    IF (ITTOUR(26) .NE. 0) GO TO 30               !SEND SUB
        C
        C     WAIT UNTIL USER HITS A TERMINAL KEY, THEN GET THAT
        C     CHARACTER AND THE CROSSHAIR ADDRESS
        C
0007          ACCEPT 50, ICHAR,HIGHX,LOWX,HIGHY,LOWY
0008    50    FORMAT (5A1)
0009          IX = 4 * ( ((HIGHX.AND.'37) * 32) + (LOWX.AND.'37) )
0010          IY = 4 * ( ((HIGHY.AND.'37) * 32) + (LOWY.AND.'37) )
0011          RETURN
0012          END
        C
```

```
FORTRAN IV      Storage Map for Program Unit GINPUT

Local Variables, .PSECT $DATA, Size = 000016 (     7. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
HIGHX   I*2     000006      HIGHY   I*2     000012      ICHAR   I*2 @ 000000
IX      I*2 @   000002      IY      I*2 @   000004      LOWX    I*2     000010
LOWY    I*2     000014

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
ITTOUR   I*2
```

Subroutine ARYPLT

This routine can be used to plot an integer array of Y values with equal increments in the X (horizontal) direction. It is particularly well suited for plotting sampled data such as from an analog-to-digital converter, with the samples equally spaced in time. All points are connected with straight solid lines. For integer data, ARYPLT provides a significant savings in memory usage over XYPLOT, at least if a large number of points are plotted. This is so because real arrays in PDP-11 Fortran require two words to store each element, while integer arrays only require one word per element. In addition, a separate array of X coordinates must be passed to XYPLOT, but is not needed for ARYPLT since ARYPLT generates the X coordinates automatically.

The arguments or parameters are described briefly in the program listing. In addition, you should note that all of the arguments except for YSCALE are of integer type. If IYOFST is set to 1560, then positive values in IARRAY will be plotted above the middle of the screen and negative values will be plotted below the middle. IYOFST can be set larger or smaller in order to move the plot up or down, respectively. YSCALE can be set to a value of 1.0 for many applications, if the elements of IARRAY are within a range of about -1000 to +1000, but YSCALE can be changed to adjust the vertical scale factor of the plot.

Several calls to ARYPLT can be made without erasing the screen, with different values for IYOFST and/or LEFT and RIGHT, in order to plot several curves on the screen together.

Routine ARYPLT requires the following subroutine, in addition to those from the Fortran library:

MPLOT

A listing of ARYPLT follows.

```
            SUBROUTINE ARYPLT(IARRAY,N,IYOFST,YSCALE,LEFT,RIGHT)
      C
      C     ROUTINE TO PLOT AN INTEGER ARRAY OF Y VALUES
      C     IN 'IARRAY' ON THE TEKTRONIX TERMINAL.
      C
      C     THE X-COORDINATE IS AUTOMATICALLY GENERATED BY THIS
      C     ROUTINE SO THAT 'N' POINTS ARE PLOTTED WITH A CONSTANT
      C     INCREMENT IN X FROM X='LEFT' TO X='RIGHT'.
      C
      C     ARGUMENTS:
      C         IARRAY = (INTEGER) ARRAY OF Y VALUES TO BE PLOTTED
      C         N = NO. OF VALUES OF IARRAY TO USE
      C                 (STARTING WITH IARRAY(1))
      C         IYOFST = (INTEGER) OFFSET (IN TEK.UNITS) TO BE ADDED
      C                  TO EACH Y VALUE AFTER MULTIPLYING BY YSCALE
      C         YSCALE = (REAL) SCALE FACTOR BY WHICH TO MULTIPLY
      C                  EACH Y VALUE.
      C         LEFT = (INTEGER) X-COORDINATE FOR LEFT SIDE OF PLOT
      C         RIGHT = (INTEGER) X-COORDINATE FOR RIGHT SIDE OF PLOT
      C         BOTH LEFT & RIGHT ARE IN TEKTRONIX SCREEN UNITS
      C
      C     FOR IYOFST=0, YSCALE=1.0, LEFT=0, & RIGHT=4095, THE
      C     VALUES IN IARRAY WILL BE PLOTTED ACROSS THE ENTIRE WIDTH
      C     OF THE SCREEN, WITH VALUES OF 0 IN IARRAY PLOTTED AT
      C     THE EXTREME BOTTOM & VALUES OF 3070 PLOTTED AT THE TOP
      C     OF THE SCREEN.  USUALLY IT IS BEST TO AVOID PLOTTING ALL
      C     THE WAY TO THE EDGES OF THE SCREEN, ESPECIALLY IF HARD
      C     COPIES ARE DESIRED.
      C
      C     AUTHOR:               WILLIAM G. CROSIER
      C     DATE:                 JULY 1980
      C     REVISED:     DEC. 1980
      C
            INTEGER IARRAY(N),IYOFST,LEFT,RIGHT,MIN,MAX,IX,IY
            REAL YSCALE,DELTAX
            DELTAX = FLOAT(RIGHT-LEFT)/FLOAT(N-1)       !X INCREMENT
            IX = LEFT
            IY = IFIX(FLOAT(IARRAY(1))*YSCALE) + IYOFST
      C     MOVE TO POSITION OF FIRST POINT
            CALL MPLOT(IX,IY,0)
      C     DRAW LINES BETWEEN EACH OF THE N POINTS
            DO 20 I=2,N
            IX = LEFT + DELTAX*FLOAT(I-1)
            IY = IFIX(FLOAT(IARRAY(I))*YSCALE) + IYOFST
            CALL MPLOT(IX,IY,1)
      20    CONTINUE
            RETURN
            END
      C
```

```
FORTRAN IV      Storage Map for Program Unit ARYPLT

Local Variables, .PSECT $DATA, Size = 000052 (    21. words)

Name    Type    Offset       Name    Type    Offset       Name    Type    Offset
DELTAX  R*4     000026       I       I*2     000032       IX      I*2     000022
IY      I*2     000024       IYOFST  I*2 @ 000004         LEFT    I*2 @ 000010
MAX     I*2     000020       MIN     I*2     000016       N       I*2 @ 000002
RIGHT   I*2 @ 000012         YSCALE  R*4 @ 000006

Local and COMMON Arrays:

Name      Type    Section Offset    ------Size----- Dimensions
IARRAY I*2     @ $DATA    000000     ****  (   **   ) (N)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
FLOAT    R*4    IFIX     I*2    MPLOT    I*2
```

Subroutine XYPLOT

This is a general purpose plotting routine for plotting a real array of Y values versus a real array of X values. If the parameter ICODE is set to 0, then only the points themselves are plotted (with asterisks "*"), and if ICODE is some other value, then the points are connected with straight lines. More information on the arguments/parameters passed to XYPLOT is given in the program listing. Note that N, L, R, B, T, and ICODE are all of integer type, but that all other arguments are real.

An example of the use of XYPLOT, along with GRID and ANOTAT, is given in the third sample program (GRTEST) in another section of this report. Please refer there for more information on a typical use of XYPLOT.

Routine XYPLOT uses the following subroutine, in addition to those from the Fortran Library:

MPLOT

The listing for XYPLOT follows.

ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:08                PAGE 001

0001          SUBROUTINE XYPLOT(X,Y,N,L,R,B,T,XMIN,XMAX,YMIN,YMAX,ICODE)
      C
      C ROUTINE TO PLOT AN ARRAY OF Y VALUES VERSUS AN ARRAY OF X VALUES
      C ON A TEKTRONIX TERMINAL.  THE POINTS ARE CONNECTED WITH STRAIGHT
      C LINES.
      C
      C ARGUMENTS:
      C       X = ARRAY OF X-COORDINATE VALUES TO PLOT (REAL)
      C       Y = ARRAY OF Y-COORDINATES (REAL)
      C       N = NUMBER OF PAIRS OF X-Y VALUES TO PLOT (INTEGER)
      C       L = LEFT BOUNDARY OF PLOTTING AREA ON SCREEN (INTEGER)
      C       R = RIGHT PLOT BOUNDARY (INTEGER)
      C       B = BOTTOM PLOT BOUNDARY (INTEGER)
      C       T = TOP PLOT BOUNDARY (INTEGER)
      C       XMIN = X-VALUE CORRESPONDING TO LEFT SIDE OF PLOTTING AREA(REAL)
      C       XMAX = X-VALUE CORRESPONDING TO RIGHT SIDE OF PLOTTING AREA(REAL)
      C       YMIN = Y-VALUE CORRESPONDING TO BOTTOM SIDE OF PLOTTING AREA(REA
      C       YMAX = Y-VALUE CORRESPONDING TO TOP SIDE OF PLOTTING AREA(REAL)
      C       ICODE = CODE FOR CONTROLLING TYPE OF LINES TO DRAW BETWEEN
      C           POINTS (INTEGER).  IF ICODE IS POS THEN LINES WILL BE
      C           CLIPPED IF THEY WOULD EXTEND PAST THE PLOT BOUNDARIES.
      C           IF ICODE IS 0 OR NEG THEN THE LINES MAY EXTEND PAST THE
      C           BOUNDARIES.  THE ABSOLUTE VALUE OF ICODE DETERMINES THE
      C           TYPE OF LINES TO DRAW (SEE ROUTINE MPLOT).
      C           IF ICODE IS 0, THEN THE POINTS ARE PLOTTED WITH ASTERISKS
      C           (*), BUT NO LINES ARE DRAWN BETWEEN THEM.
      C           EXAMPLES:
      C             ICODE=0 WILL PLOT ASTERISKS WITH NO CONECTING LINES.
      C             ICODE=1 WILL DRAW NORMAL SOLID LINES BETWEEN POINTS
      C               & WILL CLIP LINES TO POINTS OUTSIDE PLOT BOUNDARY.
      C             ICODE=-97 WILL DRAW DOTTED LINES BETWEEN POINTS, &
      C               WILL ALLOW THE LINES TO EXTEND OUTSIDE THE
      C               DESIGNATED PLOT BOUNDARIES.
      C
      C       NOTE: ARGUMENTS L,R,B, & T ARE ALL OF INTEGER TYPE AND ARE
      C       IN TEKTRONIX SCREEN UNITS. L&R MUST BE BETWEEN 0 AND 4095.
      C       B&T MUST BE BETWEEN 0 AND 3120. THESE PARAMETERS DETERMINE
      C       THE PORTION OF THE SCREEN TO BE USED FOR A PLOT.
      C       ARGUMENTS XMIN,XMAX,YMIN, & YMAX ARE ALL REAL AND ARE IN USER
      C       UNITS (SAME AS IN ROUTINE ANOTAT).  THEY MAY BE IN ANY RANGE
      C       DESIRED.
      C
      C AUTHOR: WILLIAM G. CROSIER
      C DATE:   FEB. 1981
      C
0002          INTEGER N,L,R,B,T,ICODE,IX,IY,IXOFST,IYOFST
0003          REAL X(N),Y(N), XMIN,XMAX,YMIN,YMAX,XSCALE,YSCALE
      C
0004          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,TIMERA,
             @             TIMHDC
      C
      C       CALCULATE SCALE FACTORS TO CONVERT X & Y VALUES
      C       FROM USER UNITS INTO TEKTRONIX SCREEN UNITS
0005          XSCALE = FLOAT(R-L) / (XMAX-XMIN)
```

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:08                PAGE 002

0006          IXOFST = L - XMIN*XSCALE
0007          YSCALE = FLOAT(T-B) / (YMAX-YMIN)
0008          IYOFST = B - YMIN*YSCALE
      C
      C       IF ICODE IS NOT 0, DRAW LINES BETWEEN EACH OF THE N POINTS
0009          DO 50 I=1,N
0010            IX = IFIX(FLOAT(X(I)) * XSCALE) + IXOFST
0011            IY = IFIX(FLOAT(Y(I)) * YSCALE) + IYOFST
0012            IF (ICODE .LE. 0)  GO TO 30
      C           PREVENT PLOT FROM EXTENDING PAST DESIRED BOUNDARIES
0014              IF (IX .LT. L)  IX=L
0016              IF (IX .GT. R)  IX=R
0018              IF (IY .LT. B)  IY=B
0020              IF (IY .GT. T)  IY=T
0022  30        CONTINUE
      C IF ICODE=0, PLOT POINTS ONLY WITHOUT CONNECTING LINES
0023            IF (ICODE .EQ. 0)  GO TO 40
0025            ITYPE = 0
0026            IF (I.GT.1)  ITYPE = IABS(ICODE)
0028            CALL MPLOT(IX,IY,ITYPE)
0029          GO TO 50
0030  40      CALL MPLOT(IX-IWIDTH/2,IY-IHIGHT/3,-1)
0031          TYPE 45
0032  45      FORMAT ('+*')
0033  50      CONTINUE
0034          RETURN
0035          END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV      Stora_e Map for Program Unit XYPLOT

Local Variables, .PSECT $DATA, Size = 000100 (    32. words)

Name    Type    Offset        Name    Type    Offset        Name    Type    Offset
B       I*2 @ 000012          I       I*2     000054        ICODE   I*2 @ 000026
ITYPE   I*2     000056        IX      I*2     000034        IXOFST  I*2     000040
IY      I*2     000036        IYOFST  I*2     000042        L       I*2 @ 000006
N       I*2 @ 000004          R       I*2 @ 000010          T       I*2 @ 000014
XMAX    R*4 @ 000020          XMIN    R*4 @ 000016          XSCALE  R*4     000044
YMAX    R*4 @ 000024          YMIN    R*4 @ 000022          YSCALE  R*4     000050


COMMON Block /GPFCOM/, Size = 000022 (     9. words)

Name    Type    Offset        Name    Type    Offset        Name    Type    Offset
MCHRSZ  I*2     000000        LSIZE   I*2     000002        IWIDTH  I*2     000004
IHIGHT  I*2     000006        IENHAN  I*2     000010        TIMERA  R*4     000012
TIMHDC  R*4     000016

Local and COMMON Arrays:

Name       Type     Section Offset   ------Size----- Dimensions
X       R*4     @ $DATA    000000    ****   (   **   ) (N)
Y       R*4     @ $DATA    000002    ****   (   **   ) (N)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
FLOAT    R*4    IABS     I*2    IFIX     I*2    MPLOT    I*2
```

Subroutine GRID

This subroutine is used to draw horizontal and/or vertical grid lines over a specified portion of the terminal screen. All of the arguments/parameters are of integer type, and their functions should be clear from reading the program listing below and the third example program (GRTEST) in another section of this report. The values for arguments LEFT, RIGHT, BOTTOM, and TOP should normally be the same in the calls to XYPLOT (or ARYPLT), GRID, and ANOTAT. If NXDIV is set equal to 0, then no vertical grid lines will be drawn. Similarly, if NYDIV is set equal to 0, then no horizontal grid lines will be drawn. ICODE specifies the line type, as described in the section on routine MPLOT.

Routine GRID uses the following subroutine, in addition to those from the Fortran Library:

MPLOT

The program listing for GRID follows.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:21              PAGE 001

0001          SUBROUTINE GRID(NXDIV,NYDIV,LEFT,RIGHT,BOTTOM,TOP,ICODE)
      C       THIS ROUTINE DRAWS GRID LINES OVER A DESIRED PORTION
      C       OF THE SCREEN ON A TEKTRONIX TERMINAI .
      C
      C       ARGUMENTS(ALL ARE INTEGERS):
      C           NXDIV=# OF X-AXIS DIVISIONS = # OF VERT. LINES - 1
      C           NYDIV=# OF Y-AXIS DIVISIONS = # OF HORIZ. LINES - 1
      C           LEFT = LEFT BOUNDARY OF PLOTTING AREA IN TEK.
      C                   SCREEN UNITS (MIN. X VALUE)
      C           RIGHT = RIGHT BOUNDARY (MAX. X)
      C           BOTTOM = LOWER BOUNDARY (MIN. Y)
      C           TOP = UPPER BOUNDARY (MAX Y)
      C           ICODE = DESIGNATES TYPE OF LINES TO DRAW
      C                   (SEE ROUTINE MPLOT OR TEKTRONIX MANUAL)
      C                   EX.:
      C                   ICODE = 1 -- DRAW SOLID LINES
      C                   ICODE = 97 -- DRAW DOTTED LINES
      C
      C       AUTHOR: WILLIAM G. CROSIER
      C       DATE:   9 DEC. 1980
      C
0002          INTEGER NXDIV,NYDIV,LEFT,RIGHT,BOTTOM,TOP,ICODE,IX,IY
0003          REAL DELTA
      C
      C       DRAW HORIZ. LINES
      C
0004          IF (NYDIV .LE. 0) GO TO 80
0006          CALL MPLOT (LEFT,BOTTOM,0)
0007          CALL MPLOT (RIGHT,BOTTOM,ICODE)
0008          DELTA=FLOAT (TOP-BOTTOM)/FLOAT(NYDIV)
0009          DO 50 K=1,NYDIV
0010          IY = BOTTOM + IFIX (DELTA * FLOAT (K))
0011          CALL MPLOT (LEFT,IY,0)
0012     50   CALL MPLOT (RIGHT,IY,ICODE)
      C
      C       DRAW VERTICAL LINES
      C
0013     80   IF (NXDIV .LE. 0) GO TO 999
0015          CALL MPLOT(LEFT,BOTTOM,0)
0016          CALL MPLOT(LEFT,TOP,ICODE)
0017          DELTA = FLOAT(RIGHT-LEFT) / FLOAT(NXDIV)
0018          DO 100 K=1,NXDIV
0019          IX = LEFT + IFIX(DELTA*FLOAT(K))
0020          CALL MPLOT(IX,BOTTOM,0)
0021    100   CALL MPLOT(IX,TOP,ICODE)
0022    999   RETURN
0023          END
      C
```

```
FORTRAN IV      Storage Map for Program Unit GRID

Local Variables, .PSECT $DATA, Size = 000044 (    18. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
BOTTOM I*2 @ 000010      DELTA  R*4   000022      ICODE  I*2 @ 000014
IX     I*2   000016      IY     I*2   000020      K      I*2   000026
LEFT   I*2 @ 000004      NXDIV  I*2 @ 000000      NYDIV  I*2 @ 000002
RIGHT  I*2 @ 000006      TOP    I*2 @ 000012

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
FLOAT    R*4    IFIX     I*2    MPLOT    I*2
```

Subroutine ANOTAT

This subroutine is used to anotate (label) the horizontal and/or vertical plot axes with numerical user units at some or all grid lines (previously produced by routine GRID). Most of the arguments or parameters for this routine are the same as for GRID and XYPLOT, and you may refer to the listings of those routines and of ANOTAT itself (given below) for more information. In addition, the third sample main program (GRTEST) in another section of this report gives an example of how these routines can be used. You may note that the values for NXDIV and NYDIV may be different in the call to ANOTAT from what they were in the call to GRID. This can be done if you want to label only every second, fifth, etc., grid line with ANOTAT.

Be sure to call routine CHRSIZ at least once in your program before calling ANOTAT. This is necessary so that ANOTAT will know what the current character size is, so that it can position the numerical units for the axes properly. In addition, make sure that arguments L and B are large enough so that ANOTAT will not attempt to type the numerical units to the left of, or below, the allowable plotting area. This may especially be a problem if one of the two larger character sizes are being used. If L or B are too small, then the numbers may be typed over the plot axes and be difficult to read.

ANOTAT requires the following subroutine, in addition to those from the Fortran Library:

MPLOT

A listing of ANOTAT follows.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:32                PAGE 001

0001          SUBROUTINE ANOTAT(NXDIV,NYDIV,L,R,B,T,XMIN,XMAX,YMIN,YMAX)
        C
        C ROUTINE TO ANOTATE PLOT AXES WITH USER UNITS AT GRID LINES.
        C IF ROUTINE GRID IS ALSO USED, THE FIRST 6 ARGUMENTS SHOULD BE
        C IDENTICAL TO THOSE USED IN GRID.  IN THAT CASE, THE GRID LABELS
        C WILL BE TYPED JUST OUTSIDE THE PLOTTING AREA.
        C
        C ARGUMENTS:
        C     FIRST 6--SAME AS FOR GRID (ALL INTEGER):
        C          NXDIV=# X-AXIS DIVISIONS = # VERT. LINES - 1
        C          NYDIV=# Y-AXIS DIVISIONS = # HORIZ LINES - 1
        C          L = LEFT BOUNDARY OF PLOT IN TEKTRONIX SCREEN UNITS
        C          R = RIGHT BOUNDARY
        C          B = BOTTOM BOUNDARY
        C          T = TOP BOUNDARY
        C     XMIN = MIN. X VALUE TO BE TYPED AT BOTTOM LEFT (REAL)
        C     XMAX = MAX. X VALUE TO BE TYPED AT BOTTOM RIGHT (REAL)
        C     YMIN = MIN. Y VALUE TO BE TYPED AT BOTTOM LEFT (REAL)
        C     YMAX = MAX. Y VALUE TO BE TYPED AT TOP LEFT (REAL)
        C
        C IF NXDIV IS GREATER THAN 0, THEN NXDIV+1 NUMBERS (STARTING WITH
        C XMIN & ENDING WITH XMAX) ARE TYPED BELOW VERT. GRID LINES (DRAWN
        C SEPARATELY WITH ROUTINE GRID) WITH AN F6.D FORMAT JUST BELOW PLOT.
        C IF NYDIV IS GREATER THAN 0, THEN NYDIV+1 NUMBERS (STARTING WITH
        C YMIN & ENDING WITH YMAX) ARE TYPED TO THE LEFT OF HORIZ. GRID
        C LINES (DRAWN WITH GRID) WITH AN F6.D FORMAT.
        C
        C AN F6.0 FORMAT WILL BE USED UNLESS THE MIN. X OR Y VALUE IS
        C GREATER THAN -9.99 AND THE MAX VALUE IS LESS THAN 99.9, IN
        C WHICH CASE AN F6.3 FORMAT WILL BE USED INSTEAD.
        C
        C AUTHOR: WILLIAM G. CROSIER
        C DATE:          16 DEC. 1980
        C
0002          INTEGER NXDIV,NYDIV,L,R,B,T,IX,IY
0003          REAL XMIN,XMAX,YMIN,YMAX,DELTA,DELTA1,X,Y,FMT(3)
0004          COMMON /GRFCOM/ MCHRSZ,LSIZE,IWIDTH,IHIGHT,IENHAN,
             @            TIMERA,TIMHDC
        C
0005          DATA FMT /4H('+' ,4H,F6. ,4H0)   /
        C
        C     LABEL Y (VERTICAL) AXIS
        C
0006          IF (NYDIV .LE. 0) GO TO 80
0008          FMT(3) = '0)  '
0009          IX = L - 6*IWIDTH
0010          DELTA = FLOAT(T-B) / FLOAT(NYDIV)
0011          DELTA1 = (YMAX-YMIN) / FLOAT(NYDIV)
0012          DO 50 K=1,NYDIV+1
0013          IY = B + IFIX(DELTA*FLOAT(K-1)) - IFIX(0.2*IHIGHT)
        C     MOVE TO DESIRED POSITION TO LEFT OF AXIS, ALPHA MODE
0014          CALL MPLOT(IX,IY,-1)
0015          Y = YMIN + DELTA1 * FLOAT(K-1)
0016          IF(YMIN.GT.-9.99 .AND. YMAX.GT.-9.99 .AND. YMIN.LT.99.9
               @               .AND. YMAX.LT.99.9) FMT(3)='3)  '
0018           TYPE FMT, Y
0019     50    CONTINUE
         C
         C     LABEL X (HORIZ.) AXIS
         C
0020     80    IF (NXDIV .LE. 0) GO TO 999
0022           FMT(3) = '0)  '
0023           IY = B - 1.2*IHIGHT
0024           DELTA = FLOAT(R-L) / FLOAT(NXDIV)
0025           DELTA1 = (XMAX-XMIN) / FLOAT(NXDIV)
0026           DO 100 K=1,NXDIV+1
0027           IX = L + IFIX(DELTA*FLOAT(K-1)) - 4*IWIDTH
         C     MOVE TO DESIRED POSITION BELOW AXIS,ALPHA MODE
0028           CALL MPLOT(IX,IY,-1)
0029           X = XMIN + DELTA1 * FLOAT(K-1)
0030           IF(XMIN.GT.-9.99 .AND. XMAX.GT.-9.99 .AND. XMIN.LT.99.9
               @               .AND. XMAX.LT.99.9) FMT(3)='3)  '
0032           TYPE FMT, X
0033    100    CONTINUE
         C
0034    999    RETURN
0035           END
         C


FORTRAN IV       Storage Map for Program Unit ANOTAT

Local Variables, .PSECT $DATA, Size = 000132 (    45. words)

Name    Type  Offset     Name    Type  Offset     Name    Type  Offset
B       I*2 @ 000010     DELTA   R*4   000044     DELTA1  R*4   000050
IX      I*2   000040     IY      I*2   000042     K       I*2   000064
L       I*2 @ 000004     NXDIV   I*2 @ 000000     NYDIV   I*2 @ 000002
R       I*2 @ 000006     T       I*2 @ 000012     X       R*4   000054
XMAX    R*4 @ 000016     XMIN    R*4 @ 000014     Y       R*4   000060
YMAX    R*4 @ 000022     YMIN    R*4 @ 000020


COMMON Block /GRFCOM/, Size = 000022 (    9. words)

Name    Type  Offset     Name    Type  Offset     Name    Type  Offset
MCHRSZ  I*2   000000     LSIZE   I*2   000002     IWIDTH  I*2   000004
IHIGHT  I*2   000006     IENHAN  I*2   000010     TIMERA  R*4   000012
TIMHDC  R*4   000016

Local and COMMON Arrays:

Name      Type    Section Offset  ------Size----- Dimensions
FMT       R*4     $DATA   000024  000014 (    6.) (3)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
FLOAT   R*4     IFIX    I*2     MPLOT   I*2
```

DATA ACQUISITION
AND MISCELLANEOUS
SUBROUTINES

Subroutine BELL

This subroutine is used to make the terminal beep or ring its bell. The sound produced depends on the particular terminal. The duration of the sound, as well as its modulation, can be controlled with the two parameters or arguments NUMBER and IDELAY. Examples of the use of this routine are given in the third sample main program (GRTEST), in another section of this report.

Routine BELL requires the following subroutines, in addition to those from the Fortran Library:

WAIT
ITTOUR (from the System Subroutine Library)

The listing for subroutine BELL follows.

ORIGINAL PAGE IS
OF POOR QUALITY

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:22:39                  PAGE 001

0001          SUBROUTINE BELL(NUMBER,IDELAY)
      C
      C       RING TERMINAL BELL/BEEP WITH VARIABLE DURATION
      C      & MODULATION CONTROL
      C
      C       NUMBER = NO. OF BELL CHARACTERS TO TRANSMIT
      C           (CONTROLS DURATION)
      C       IDELAY = NO. OF 1/60 SEC INCREMENTS TO
      C           WAIT BETWEEN BELLS (CONTROLS MODULATION
      C           & PERCEIVED FREQUENCY).
      C
      C       IDELAY CAN BE 0 FOR A CONTINUOUS TONE, WITH DURATION
      C       CONTROLLED BY NUMBER, OR IDELAY CAN BE A POSITIVE
      C       INTEGER TO PRODUCE A BUZZING SOUND OR DISCRETE BEEPS.
      C
      C       NOTE: SOUND IS DEPENDENT ON THE TERMINAL & ON ITS
      C       BAUD RATE SETTING.
      C
0002          DO 100 K=1,NUMBER
0003            IF (IDELAY .LT. 1)  GO TO 40
0005              DELAY = FLOAT(IDELAY) / 60.0
0006              CALL WAIT(DELAY,0)
0007  40        IF (ITTOUR(7) .NE. 0) GO TO 40     !SEND BELL
0009  100     CONTINUE
0010          RETURN
0011          END
      C
```

```
FORTRAN IV      Storage Map for Program Unit BELL

Local Variables, .PSECT $DATA, Size = 000012 (     5. words)

Name    Type   Offset        Name    Type   Offset       Name     Type   Offset
DELAY   R*4    000006        IDELAY  I*2 @ 000002       K        I*2    000004
NUMBER  I*2 @  000000

Subroutines, Functions, Statement and Processor-Defined Functions:

Name     Type    Name     Type    Name     Type    Name     Type    Name     Type
FLOAT     R*4    ITTOUR   I*2     WAIT      R*4
```

Subroutine WAIT

This routine uses the RT-11 system line frequency clock to time a waiting period. The user simply passes the routine a real value in argument/parameter SEC which specifies the duration, in seconds, of the waiting period. The normal functions of the line time clock are not affected. When the waiting period has elapsed, then control is returned to the calling program. If you want to be able to terminate the wait prematurely (in less than "SEC" seconds), then an interrupt service routine can be used to set the argument IABORT equal to a non-zero value. You may want to do this in a real-time experiment control program, if something happened during a programmed wait or if you pressed a button on a control panel, for example. If you do not need to prematurely terminate the waiting period, then set IABORT equal to 0. An example of the use of this routine can be seen in the listing for subroutine ERASE. This routine calls WAIT after sending the command to erase the terminal screen, so that the terminal will have enough time (normally TIMERA is 1.5 seconds) to completely clear the screen.

Subroutine WAIT may also be used to time the periods between data acquisition samples, if the sampling rate if fairly slow (60 HZ or slower). The accuracy and resolution of the programmed wait is 1/60 = 0.017 second, since that is the time between cycles of the line frequency. There should be no cumulative time error between waiting periods, however, as long as not too many CPU operations are performed, so that the long term accuracy of the times measured should be quite good. As an example of using WAIT to control data acquisition, the following could be used to get 1000 samples of signals on analog channel 3 and 5 with the samples taken every 1/30 = 0.033 second:

```
      INTEGER IDATA(1000),IDATB(1000)
      IPGNCD = 2          !PROGRAMMABLE GAIN CODE
      DO 10 K=1,1000      !DO 1000 TIMES
         IDATA(K) = ISAMPA(3,IPGNCD,0)    !SAMPLE CHANNEL 3
         IDATB(K) = ISAMPA(5,IPGNCD,0)    !SAMPLE CHANNEL 5
         CALL WAIT(0.033,0)
   10    CONTINUE
```

At the end of 33 seconds, 1000 samples would be collected from Channel 3 and stored in array IDATA, while 1000 samples from Channel 5 would be in array IDATB. If numerous computations such as averaging of many samples are performed between waiting periods, then the period may actually be longer than desired, since the computations make take more than 1/60 of a second.

If you have a 50 HZ line frequency system clock, rather than the 60 HZ usually used in the United States, be sure to change this line in the program:

```
from:  TICKS = SEC*60. + 0.5
  to:  TICKS = SEC*50. + 0.5
```

Do this only if you have a 50 HZ line frequency clock.

Routine WAIT requires the following subroutines from the System Subroutine Library:

GTIM	(Time of Day in clock ticks past midnight)
JADD	(Integer*4 addition)
JAFIX	(Real*4 to Integer*4 conversion)
JCMP	(Integer*4 compare)
JJCVT	(Interchange halves of Integer*4 variable)

A listing of WAIT follows.

```
FORTRAN IV      V02.1-1     Wed 23-Dec-81 10:23:45                PAGE 001

0001          SUBROUTINE WAIT(SEC,IABORT)
        C
        C     LINE TIME CLOCK WAIT ROUTINE
        C     WRITTEN BY: WILLIAM G. CROSIER
        C     SEC = NUMBER OF SECONDS (REAL, NOT INTEGER) TO WAIT
        C     USES LINE TIME CLOCK FOR TIMING CONTROL.
        C     THE J--- SUBROUTINES USED HERE PERFORM INTEGER*4 ARITHMETIC.
        C     RESOLUTION & ACCURACY = APPROX. 0.017 = 1/60 SECOND
        C     THIS ROUTINE DOES NOT AFFECT NORMAL FUNCTIONS OF LTC.
        C     IF PARAMETER IABORT BECOMES NON-ZERO DURING
        C     THE WAITING PERIOD (IF SET BY AN INTERRUPT ROUTINE),
        C     THEN THE WAIT IS IMMEDIATELY TERMINATED.
        C     IF THIS FEATURE IS NOT NEEDED, USE A VALUE OF 0 FOR IABORT
        C
0002          INTEGER*4 ITIM1,ITIM2,IDELTA
0003          CALL GTIM(ITIM1)                !STORE CURRENT TIME IN ITIME
0004          CALL JJCVT(ITIM1)               !INTERCHANGE WORDS
0005          TICKS = SEC*60. + 0.5                     !CONVERT SEC TO CLOCK TICKS
0006          CALL JAFIX(TICKS,IDELTA)        !CONVERT TO INTEGER
0007          CALL JADD(ITIM1,IDELTA,ITIM1)             !CALCULATE STOP TIME
0008    10    CALL GTIM(ITIM2)                !GET CURRENT TIME OF DAY
0009          CALL JJCVT(ITIM2)               !INTERCHANGE WORDS
0010          IF(IABORT .NE. 0) GO TO 99      !CHECK FOR ABORT
0012          IF (JCMP(ITIM2,ITIM1) .LT. 0)  GO TO 10
0014    99    RETURN
0015          END
        C
```

```
FORTRAN IV      Storage Map for Program Unit WAIT

Local Variables, .PSECT $DATA, Size = 000024 (    10. words)

Name    Type   Offset       Name    Type   Offset       Name    Type   Offset
IABORT  I*2 @ 000002        IDELTA  I*4    000014       ITIM1   I*4    000004
ITIM2   I*4    000010       SEC     R*4 @ 000000        TICKS   R*4    000020

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
GTIM    R*4     JADD    I*2     JAFIX   I*2     JCMP    I*2     JJCVT   I*2
```

Subroutine ISAMPA

This is a Fortran-callable subroutine, written in MACRO Assembly Language, for sampling an analog signal with an analog-to-digital (A/D) converter. It was written in MACRO so that it can execute as quickly as possible, but still be usable with FORTRAN programs. With this routine, sampling rates of several hundred samples per second can easily be achieved, even with some computations performed between samples. For accurate control of the time intervals between samples, you may use routine WAIT if the sampling rate is 60 HZ or slower. Otherwise, you should use a programmable clock/timer such as the KW-11P. If ISAMPA is called by a MACRO interrupt service routine for the KW-11P, make sure that the routine uses the normal PDP-11 Fortran calling conventions for passing arguments, etc.

If accurate control of the sampling rate is not a requirement, but you need to sample a large number of values in a certain time period and average them in order to reduce noise effects, you may use the following procedure. First, get the current time of day (in seconds past midnight) with the RT-11 system routine SECNDS, or wait until an appropriate external event occurs. Second, call ISAMPA, convert the returned sampled value to real or double precision, and add it to a real or double precision variable used as an accumulator. Repeat the sampling and accumulating until either you have enough samples, or until enough time has elapsed. (Use the SECNDS subroutine again.) Finally, divide by the number of samples collected. A real variable (rather than integer) should be used if you are using a 12-bit A/D converter and are adding together more than 16 samples, because a 16-bit integer accumulator can be overflowed by adding more than 16 12-bit values together if each of them are near full scale. Generally, no error message will occur if this happens, since PDP-11 Fortran does not check for an overflow on an integer add operation. Similarly, you should use a double precision accumulator if you add together more than about 2000 samples, because you can drop bits when doing so with Real*4 arithmetic.

For examples of how ISAMPA may be used, refer to the discussions for subroutine WAIT and for the first sample main program (ADTEST), in another section of this report.

If you are using a DEC ADV-11A A/D converter, then you must mask out the four most significant bits, since DEC uses them for other purposes. In addition, DEC's ADV-11A converters can only be used with an offset binary format, so that a value of 4000 (octal) or 2048 (decimal) must be subtracted from the sampled value in order to convert it to the normal two's complement coding. The following will mask out the 4 MSB's and convert the value to 2's complement:

```
I=("7777 .AND. ISAMPA(ICHAN,0,1)) -"4000
```

This is necessary only with DEC A/D boards. Note also, in the above example, that a value of 0 must be used for the second argument (IPGNCD), since the DEC boards do not have programmable gain.

ISAMPA requires no subroutines.

A listing of ISAMPA follows.

```
                                             .TITLE  ISAMPA
                                      ;  INTEGER FUNCTION ISAMPA
                                      ;  PURPOSE:  SAMPLE ANALOG CHANNEL  ICHAN USING PROGRAMMABLE
                                      ;          GAIN CODE IPGNCD ON ADAC, DATA TRANSLATION, OR DEC
                                      ;          ANALOG INTERFACE BOARD.
                                      ;
                                      ;  USE: I = ISAMPA(ICHAN,IPGNCD,IADTYP)
                                      ;          WHERE ICHAN CAN BE FROM 0 TO 31 (15 ON SOME BOARDS)
                                      ;          AND   IPGNCD IS 0,1,2, OR 3
                                      ;                FOR DATA TRANSLATION A/D'S, 0 GIVES LOWEST
                                      ;                (UNITY) GAIN, AND 3 GIVES HIGHEST GAIN (8 OR
                                      ;                10, DEPENDING ON BOARD USED).
                                      ;                FOR ADAC A/D'S, 0 GIVES HIGHEST GAIN (8 OR 10),
                                      ;                AND 3 GIVES LOWEST GAIN (UNITY).
                                      ;                IF THE BOARD YOU ARE USING DOES NOT HAVE
                                      ;                PROGRAMMABLE GAIN (DEC ONES DO NOT),
                                      ;                THEN YOU SHOULD SPECIFY 0 FOR IPGNCD.
                                      ;          AND   IADTYP IS 0 OR 1
                                      ;                0 IS USED FOR ADAC A/D BOARDS, &
                                      ;                1 IS USED FOR DATA TRANSLATION OR DEC.
                                      ;
                                      ;          VALUE RETURNED BY THE FUNCTION (IN R0) IS THE INTEGER
                                      ;          NUMBER OF COUNTS FROM THE A/D CONVERTER (-2048 FOR NEG.
                                      ;          FULL SCALE; +2047 FOR POS. FULL SCALE).  ONLY THE LOW
                                      ;          ORDER 12 BITS ARE VALID IF A DEC A/D BOARD IS USED.
                                      ;
                                      ;          NOTE:   THE A/D BOARD CSR ADDRESS SHOULD BE SET
                                      ;                  TO 176770 (OCTAL)
                                      ;
                                      ;  WRITTEN BY:           WILLIAM G. CROSIER
                                      ;  DATE:                 2 MAY 1981
                                      ;
                                             .GLOBL  ISAMPA
        176770                               ADCSR=176770              ;A/D CONVERTER CSR ADDRESS
        176772                               ADDATA=ADCSR+2            ;A/D CONV DATA BUFFER ADDR.
                                      ;
000000  017500  000004                ISAMPA: MOV     @4(R5),R0        ;LOAD GAIN CODE IN R0
000004  006300                               ASL     R0               ;SHIFT LEFT 2 BITS
000006  006300                               ASL     R0
000010  012701  176770                       MOV     #ADCSR,R1        ;STORE A/D CSR ADDR. IN R1
000014  005775  000006                       TST     @6(R5)           ;DETERMINE A/D TYPE
000020  001006                               BNE     DTDEC            ;SKIP IF NOT TYPE 0
000022  006300                               ASL     R0               ;ADAC BOARD, SO SHIFT 1 MORE BIT
000024  110011                               MOVB    R0,(R1)          ;PUT GAIN CODE IN A/D CSR
000026  117561  000002  000001               MOVB    @2(R5),1(R1)     ;LOAD CHAN # & START CONVERSION
000034  000405                               BR      GETDAT           ;GO GET SAMPLED VALUE
000036                                DTDEC:                           ;DATA TRANSLATION OR DEC BOARD
000036  117561  000002  000001               MOVB    @2(R5),1(R1)     ;PUT CHAN. # .N A/D CSR HI BYTE
000044  110011                               MOVB    R0,(R1)          ;PUT GAIN CODE IN A/D CSR
000046  005211                               INC     (R1)             ;SET BIT 0 (START CONVERSION)
000050  012700  176772                GETDAT: MOV     #ADDATA,R0       ;PUT A/D DATA BUFR ADDR IN R0
000054  105711                        LOOP:   TSTB    (R1)             ;A/D CONVERSION DONE?
000056  100376                               BPL     LOOP             ;WAIT TILL FINISHED
000060  011000                               MOV     (R0),R0          ;PUT A/D SAMPLED VALUE IN R0
000062  000207                               RTS     PC               ;RETURN
        000001                               .END
```

```
ISAMPA  MACRO V04.00  22-DEC-81 09:00:23 PAGE 1-1
SYMBOL TABLE

ADCSR = 176770          DTDEC    000036R          GETDAT  000050R          ISAMPA  000000RG          LOOP    000054R
ADDATA= 176772

. ABS.  000000     000
        000064     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  64 PAGES
DK:ISAMPA,DK:ISAMPA=DK:ISAMPA/C
```

```
ISAMPA  MACRO V04.00  22-DEC-81 09:00:23 PAGE S-1
CROSS REFERENCE TABLE (CREF V04.00 )

ADCSR     1-34#     1-35      1-40
ADDATA    1-35#     1-51
DTDEC     1-42      1-47#
GETDAT    1-46      1-51#
ISAMPA    1-33      1-37#
LOOP      1-52#     1-53
```

Subroutine DISKIO

This routine is used to read or write binary unformatted data, in a random fashion, to a sequential disk file. Variable length records are supported (in multiple of 256 integer words), and the routine is somewhat more economical of memory and CPU time than the normal Fortran disk I/O, since the transfer takes place directly from the arrays in the user's program, rather than through intermediate buffers. Any binary data can be transferred, regardless of whether the calling program treats it as logical, integer, real, or string data. The principal restrictions on the data is that it must all be in contiguous memory locations, and that only multiples of 256 words should be transferred normally through each call to DISKIO. Generally, the data should all be placed in a COMMON block in order to force the compiler to place it all in contiguous locations, unless it is in a single array.

The arguments/parameters for this routine are discussed in the program lising below. The file name passed in argument FILNAM can be any valid RT-11 file name. Note, however, that it must be exactly 12 characters (bytes) long, with no colons or periods within it to separate the device name or file type/suffix. Trailing spaces are allowed, however, at the end of each portion of the file name in order to make the device name identifier exactly 3 characters long and the main part of the file name exactly 6 characters long. A null or zero byte should follow the 12 character file name (as in the MACRO assembler ASCIZ construction).

Data written by this routine can only be read back with the same routine, and not by Fortran READ statements. In addition, this subroutine was designed to work only with the RT-11 operating system.

For more information and examples of the use of DISKIO, refer to the fourth sample main program, DISKRW, in another section of this document.

Refer to the program listing which follows for more information.

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:55                  PAGE 001

0001            SUBROUTINE DISKIO(FILNAM,MODE,BUFFER,NWRDS,IBLK,NBLK,IERR)
      C
      C         PURPOSE:     READ OR WRITE BINARY DATA TO A DISK FILE
      C         WRITTEN BY: WILLIAM G. CROSIER
      C         DATE:                11 JUNE 1980
      C
      C         ARGUMENTS:
      C
      C         FILNAME=ARRAY CONTAINING ASCII FILE NAME (12 CHAR.)
      C           (FILNAM IS IGNORED WHEN MODE WAS NEG. ON THE LAST
      C           CALL TO DISKIO.)
      C
      C             EX:      DX1TEST07DAT
      C                      DK MYFILE
      C                      DK FILE7
      C
      C         MODE=+1 OR -1 TO CREATE A NEW FILE & WRITE OUT DATA TO IT
      C              (+1 OR -1 WILL CAUSE ANY FILE WITH THE SAME NAME WHICH
      C               PREVIOUSLY EXISTED TO BE DELETED WHEN THE NEW
      C               FILE IS CLOSED)
      C             =2 OR -2 TO MODIFY AN EXISTING FILE (OVERWRITE ALL OR PART)
      C             =3 OR -3 TO READ DATA FROM AN EXISTING FILE
      C         IF MODE IS POSITIVE, THE FILE IS CLOSED AFTER THE I/O.
      C         IF MODE IS NEGATIVE, THE FILE IS NOT CLOSED, SO THAT THE NEXT
      C         CALL TO DISKIO WILL NOT REQUIRE RE-OPENING THE FILE.
      C        (THE NEXT CALL WILL ALSO IGNORE FILNAM SINCE THE
      C         PREVIOUSLY SPECIFIED NAME WILL BE USED AGAIN.)
      C         IF AN ERROR OCCURS WHEN MODE IS NEG., THE NEXT I/O
      C         MAY NOT BE VALID SINCE THE FILE MAY NOT BE OPENED
      C         PROPERLY. TO AVOID THIS PROBLEM, DO A READ OPERATION
      C         WITH MODE=3 TO CLOSE THE FILE IF AN ERROR OCCURS
      C         WHEN MODE IS NEG.
      C         NOTE: ALL FILES ARE UNCONDITIONALLY CLOSED WHEN
      C         THE PROGRAM TERMINATES, REGARDLESS OF WHETHER MODE
      C         WAS POS. OR NEG. ON THE LAST CALL.
      C
      C         BUFFER=AREA IN MEMORY WHERE DATA IS TO BE TRANSFERRED TO/FROM
      C
      C         NWRDS=NO. OF INTEGER WORDS TO READ FROM OR WRITTEN INTO BUFFER
      C                (SHOULD BE A MULTIPLE OF 256)
      C
      C         IBLK=STARTING BLOCK NO. IN FILE WHERE DATA TRANSFER IS TO OCCUR
      C
      C         NBLK=NO. OF 256-WORD INTEGER BLOCKS TO ALLOCATE FOR A NEW FILE
      C               (NBLK IS IGNORED EXCEPT WHEN MODE=1 OR -1)
      C
      C         IERR=ERROR CODE RETURNED BY DISKIO
      C             =0 MEANS NO ERRORS OCCURRED
      C             =1 MEANS QUEUE ELEMENT FAILURE OCCURRED
      C             =2 MEANS NO I/O CHANNEL WAS AVAILABLE
      C             =3 MEANS HANDLER FOR SPECIFIED DEVICE CAN'T BE LOADED
      C             =4 MEANS FILE ALLOCATION FAILED WHEN CREATING FILE
      C             =5 MEANS A DATA OUTPUT ERROR OCCURRED
      C             =6 MEANS A FILE LOOKUP FAILURE (COULD NOT FIND FILE)
```

```
FORTRAN IV      V02.1-1    Wed 23-Dec-81 10:23:55                PAGE 002

      C          =7 MEANS A DATA INPUT ERROR OCCURRED
      C
      C     THIS ROUTINE TAKES CARE OF OPENING & CLOSING THE FILE FOR
      C     EACH DATA TRANSFER, SETTING QUEUE ELEMENTS APPROPRIATELY,
      C     GETTING AN I/O CHANNEL, FETCHING THE DEVICE HANDLER, CREATING
      C     THE FILE ENTRY, & DOING THE ACTUAL DATA I/O
      C
0002        BYTE FILNAM(12)
0003        INTEGER MODE,BUFFER,NWRDS,IBLK,NBLK,IFILE(4),FLAG,IERR,ICHAN,
           @IPMODE
0004        COMMON /DISCOM/ IPMODE
0005        DATA IPMODE /0/
0006        IERR=0
0007        IF (IPMODE.LT.0) GO TO 60    !FILE LEFT OPEN?
      C CONVERT FILE NAME TO RADIX-50
0009        CALL IRAD50(12,FILNAM,IFILE)
0010        IF (IPMODE .NE. 0) GO TO 20
      C FIRST TIME ROUTINE HAS BEEN CALLED, SO SET QUEUE ELEMENTS
0012        IF (IQSET(2) .EQ. 0) GO TO 20
0014        IERR=1                                 !ERROR-QUEUE ELEMENT FAILURE
0015        GO TO 999
0016  20    ICHAN=IGETC(IDUMMY)        !GET AN I/O CHANNEL
0017        IF (ICHAN .GE. 0) GO TO 30
0019        IERR=2                                 !ERR-NO CHAN. AVAIL.
0020        GO TO 99
0021  30    IF (IFETCH(IFILE(1)) .EQ. 0) GO TO 40       !FETCH DEVICE HANDLE
0023        IERR=3                                 !ERR-CANNOT LOAD HANDLER
0024        GO TO 90
0025  40    IF (IABS(MODE) .GT. 1) GO TO 50
      C CREATE NEW FILE ENTRY
0027        IF (IENTER(ICHAN,IFILE,NBLK).GE.0) GO TO 60
0029        IERR=4                                 !ERR-FILE ALLOCATION FAILED
0030        GO TO 90
0031  60    IF(IABS(MODE) .EQ. 3) GO TO 70
      C WRITE OUT DATA FROM BUFFER
0033        IF (IWRITW(NWRDS,BUFFER,IBLK,ICHAN).GE.0) GO TO 90
0035        IERR=5                      !ERR-DATA OUTPUT
0036        GO TO 90
      C FIND EXISTING FILE
0037  50    IF (LOOKUP(ICHAN,IFILE) .GE. 0) GO TO 60
0039        IERR=6                      !ERR IN FILE LOOKUP
0040        GO TO 90
      C READ DATA INTO BUFFER
0041  70    IF(IREADW(NWRDS,BUFFER,IBLK,ICHAN).GE.0) GO TO 90
0043        IERR=7                                 !ERR IN READING DATA
0044  90    IF (MODE .LT. 0) GO TO 99   !LEAVE CHAN. OPEN?
0046        CALL CLOSEC(ICHAN)          !CLOSE THE I/O CHANNEL
0047        CALL IFREEC(ICHAN)          !& FREE IT
0048  99    IPMODE=MODE
0049  999   RETURN
0050        END
```

```
FORTRAN IV      Storage Map for Program Unit DISKIO

Local Variables, .PSECT $DATA, Size = 000034 (    14. words)

Name    Type    Offset       Name    Type    Offset       Name    Type    Offset
BUFFER  I*2 @  000004       FLAG    I*2     000026       IBLK    I*2 @  000010
ICHAN   I*2     000030       IDUMMY  I*2     000032       IERR    I*2 @  000014
MODE    I*2 @  000002       NBLK    I*2 @  000012       NWRDS   I*2 @  000006


COMMON Block /DISCOM/, Size = 000002 (     1. words)

Name    Type    Offset       Name    Type    Offset       Name    Type    Offset
IPMODE  I*2     000000

Local and COMMON Arrays:.

Name      Type    Section Offset    ------Size----- Dimensions
FILNAM L*1     @ $DATA    000000   000014 (     6.) (12)
IFILE  I*2       $DATA    000016   000010 (     4.) (4)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name     Type    Name     Type    Name     Type    Name     Type    Name     Type
CLOSEC   R*4     IABS      I*2    IENTER    I*2    IFETCH    I*2    IFREEC    I*2
ICETC    I*2     IQSET     I*2    IRAD50    I*2    IREADW    I*2    IWRITW    I*2
LOOKUP   I*2
```

☆ U.S. GOVERNMENT PRINTING OFFICE: 1982—[illegible]-011/140