ARL-FLIGHT-MECH-TM-446

AR-006-643

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

AERONAUTICAL RESEARCH LABORATORY

MELBOURNE, VICTORIA

Flight Mechanics Technical Memorandum 446

THE CONVERSION OF A FORTRAN DATA PLOTTING PROGRAM USING DI-3000 GRAPHICS TO OPERATION ON A MACINTOSH PERSONAL COMPUTER

by

D.M. Blunt

Approved for public release

SEPTEMBER 1991

AR-006-643

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
AERONAUTICAL RESEARCH LABORATORY

Flight Mechanics Technical Memorandum 446

THE CONVERSION OF A FORTRAN DATA PLOTTING PROGRAM USING DI-3000 GRAPHICS TO OPERATION ON A MACINTOSH PERSONAL COMPUTER

by

D.M. Blunt

SUMMARY

The planned retirement of the Elxsi 6400 mainframe computer at ARL has necessitated the conversion of the general purpose Fortran data plotting program TRANS to operation on a Macintosh IIfx personal computer, where it has been renamed MacTRANS. The conversion has included the limited use of regular Macintosh features such as pull down menus, and allowed the plots produced by MacTRANS to be edited in most commercial Macintosh graphics programs. For most cases, the execution speed of MacTRANS on the Macintosh IIfx has been found to be faster than that of TRANS on the Elxsi.

DSTO ARL MELBOURNE

POSTAL ADDRESS: Director, Aeronautical Research Laboratory
506 Lorimer Street, Fishermens Bend 3207
Victoria Australia

CONTENTS

1. INTRODUCTION

The imminent retirement of the Elxsi 6400 mainframe computer at ARL has necessitated the transferral of several programs to other computers. While many of these programs perform large amounts of numerical calculations, and thus are more suited to transferral to other mainframe computers, the programs which are graphics intensive are more suited to transferral to computers which specialise in the presentation and manipulation of graphics.

The Macintosh IIfx was selected as a possible alternative for these graphics intensive programs as, in addition to the graphical capabilities it shares with the other Macintosh personal computers, it possesses the fast processing speeds necessary for it to be a viable alternative to the Elxsi. However, the Macintosh computers use a completely different graphics system to the Elxsi, and thus these programs need to be significantly modified.

This document presents the conversion of the general purpose data plotting program TRANS (Refs 1-3), which uses the DI-3000 computer graphics package[1], from operation on the Elxsi to operation on a Macintosh IIfx, where it has been renamed MacTRANS. MacTRANS will also run, albeit with some speed penalty, on other Macintosh computers equipped with 68020 or 68030 processors, plus 68881 or 68882 maths coprocessors.

The conversion has involved the modification of the Elxsi Fortran code so that it would compile under Language Systems Fortran (Ref. 4) in the Macintosh Programmers Workshop (MPW) environment, the development of a partial emulation of the DI-3000 graphics package using Macintosh QuickDraw procedures, and the addition of an "event-loop" [2] to the program so that some of the regular features of the Macintosh could be used, e.g. pull-down menus and "clicking" between windows. However, the basic structure of the program has been left intact.

2. CONVERSION STRATEGY

The strategy adopted in the conversion of TRANS to MacTRANS was to

a) make the necessary changes to the Fortran code to remove all the Language System Fortran compile-time and run-time errors

b) fix any errors in the parts of the program that produce non-graphical output, and tailor this output so that it can be printed with the minimum of alteration on an Apple LaserWriter through MPW

c) develop the DI-3000 emulation to the extent of producing an image on the Macintosh monitor, and fix any errors in the graphics producing sections of the program

d) add colour to the graphics

e) produce the code necessary to save the graphics in PICT files [3]

---

1 DI3000 is a computer graphics package from Precision Visuals Inc., Colorado USA that plots graphics in two or three dimensions. It is used in combination with the Metafile System from the same company, and stores the DI3000 graphical information in a device independent file called a metafile, which can then be read by a metafile translator and plotted on various devices.

2 An event-loop is the fundamental part of a Macintosh application which examines the events that occur while a program is executing, e.g. mouse button 'clicks' and menu item selection, and passes control to the relevant sections of code to provide the appropriate actions.

3 A PICT file (defined in Macintosh Technical Note #27) is a data fork file with a 512-byte header, followed by a QuickDraw data structure within which drawing opcodes represent a graphic image.

f) add the event-loop

g) incorporate graphics printing into the event-loop

The code modifications and additions are dealt with in Sections 3 to 9, which give representative examples for each change made. A more comprehensive description of the modifications is given in Ref. 5. Wherever Macintosh Toolbox (intrinsic functions and procedures) or QuickDraw (subset of the Toolbox dealing with graphics procedures) calls are shown in the text, the reader is referred to Ref. 6 for information on the variables shown in the calls and the exact actions carried out by them. These calls also need compiler switches, include files, and special calling procedures which are dealt with in Appendix A (see also Ref. 4). Similarly, wherever DI-3000 calls are shown, the reader is referred to Ref. 7. TRANS commands are summarised in Appendix B.

3. NECESSARY CHANGES TO ELXSI FORTRAN

These are the changes that were needed to remove the compile-time and run-time errors, and to accommodate the differences between Language Systems Fortran and Elxsi Fortran.

3.1 Intrinsic Functions

The bell sounding and file renaming subroutines were rewritten to use the Macintosh Toolbox equivalents of the Elxsi intrinsic calls, as shown below.

a) Bell

```
Call SysBeep(Duration)          !Duration in 1/60 th's of a second
```

b) Rename

```
OSErr = GetVol(Int4(0),%Ref(VRefNum))                  !Get Volume Number
OSErr = Rename(OldFileName,VRefNum,NewFileName)        !Change File Name
OSErr = FlushVol(Int4(0),VRefNum)                      !Update Volume
```

3.2 Encode Statements

Problems were experienced with the ENCODE statements which stored character data in real variables, particularly where these variables were subsequently used as formats. In these cases the original ENCODE statements were changed to internal WRITE statements, and the real variables were changed to character variables. For example, in Subroutine PNTHDS, the code

```
102   FORMAT(3H ',,I1)
30    ENCODE(4,102,TTY(4))NC
```

was changed to

```
30    WRITE(TTY(16:16),'(I1)') NC
```

where "TTY" was changed from a real*4 array "TTY(8)" initialised by

```
      DATA     (TTY(J),J=1,8)/4H(2X,,4H'  T,4Hime ,4H ',5,
     1 4H('  ,4HBlk#,4H',I4,4H))  /
```

to a character*31 initialised by:

```
      DATA TTY/'(3X,''  Time  '',5('' Blk#'',I4)/)'/
```

It would appear that the ENCODE statement cannot be used to encode characters (or character variables) into real variables; it can only be used to encode characters into other character variables. However, it is possible to use an internal READ statement to read characters into real variables.

3.3 Formats

It was necessary to insert the additional edit descriptor "1X," to the start of all the write statement formats to perform the function of a carriage control character, to ensure that the first character in each write was printed.

3.4 Mismatched Variable Types

The Elxsi version of TRANS extensively used real variables to store character data. This was acceptable in Language Systems Fortran where data was read into real variables (with the exceptions noted in Section 3.2), but resulted in problems when these character-containing real variables were compared to other character or string variables. This mostly occurred when a response from a keyboard entry was compared to a character constant.

The problem was removed by assigning the character constants to real variables in data statements and then using the real variables in the comparison expressions. For example, in Subroutine TRAN2, the code

```
      TYPE 305
305   FORMAT(1X,'ARE SYMBOLS REQRD FOR PLOTS : '$)
      ACCEPT 201,SYMANS
201   FORMAT(A3)
      CALL CHKUPR(SYMANS,1)          !CONVERTS TO UPPER CASE
      IF(SYMANS.EQ.'Y') GO TO 325
```

in which SYMANS has a default type of REAL*4, was changed to

```
      TYPE 305
305   FORMAT(1X,'ARE SYMBOLS REQRD FOR PLOTS : '$)
      ACCEPT 201,SYMANS
201   FORMAT(A3)
      CALL CHKUPR(SYMANS,1)          !CONVERTS TO UPPER CASE
      IF(SYMANS.EQ.YES$) GO TO 325
```

where YES$ (a REAL*4) was assigned in the data statement:

```
      DATA YES$/'Y'/
```

Note: A similar problem occurred when characters were stored in integer variables as well. However, in this case, the character-containing integer variables could be directly compared with character constants (as in I.EQ.'Y'*), but not with character-containing real variables (as in* SYMANS.EQ.I*).*

3.5 Opening and Closing Files

In Elxsi Fortran code, the OPEN statement can be used a second time (i.e. after the file has already been opened) to reset the file pointer to the beginning of the file. In Language Systems Fortran, however, once a file is opened it must be explicitly closed before it can be opened again (otherwise a run-time error will result). This necessitated the inclusion of two CLOSE statements, one at the end of Subroutine BLKLST (CLOSE (20)), and the other before the call to Subroutine TRAN2 in the main program (CLOSE (1)).

3.6 Save Statement

Elxsi Fortran automatically saves all subprogram (subroutine or function) variables between calls to those subprograms, while Language Systems Fortran does not.

To overcome this problem, the "-saveall" compiler switch can be used at the compilation stage, which may significantly slow execution of the program, or the necessary variables can be saved individually in the appropriate subroutines and functions with the

SAVE statement, as was done in MacTRANS. The subroutines and variables involved are shown in Table 1.

TABLE 1
Subprogram Variables Put into SAVE Statements

Subprogram	Variables
BLKLST	N
BRKLIN	All variables
INBLK	IDATA, LNUMS, SAVE
STRIP	Y, P, UP, ND, XLA, YLA, NOPIPS
TRAN2	NOPEN, Y, RECTS, PICTURES, SCALING, JUST, NAME, OFFSET, WINS, MARK

4. ADDITIONAL CHANGES TO FORTRAN CODING

Additional changes were made to tailor TRANS to the Macintosh. These included:

Main Program — Removal of the default plot offset (used for a particular plotting device linked to the Elxsi).

Subroutines FCLOSE, FOPEN, RPT, and METRED — Changes to the code necessary for the new file-naming procedure (see Section 5.1).

Subroutine FRMOPN — Adjustment of the starting position of the plots to suit the LaserWriter.

Subroutines LPLOT, HEADNS, CONBLK, PNTHDS, and ZERO — Modifications to the format statements to fit the text to the window used by MacTRANS, and keep printer output within A4 paper dimensions.

Subroutine MOVE — Name change to MMOVE to remove a conflict with a reserved QuickDraw procedure name.

Subroutines NUMB, PLTHD, XAXIS, and YAXIS — Adjustments to the positioning of the text in the plots to suit the Times font used in MacTRANS.

5. DI-3000 GRAPHICS EMULATION

The partial implementation of the DI-3000 graphics routines using QuickDraw procedures involved emulating some of the DI-3000 routines and leaving the rest as dummy subroutines, as illustrated in Table 2.

TABLE 2
Skeleton DI-3000 Program Commands and Emulation Actions

Command	DI-3000 Action	QuickDraw Emulation
CALL JBEGIN	Initialise DI-3000	No action
CALL JDINIT (1)	Initialise device number 1	No action
CALL JDEVON (1)	Activate device number 1	No action
CALL JVSPAC (...)	Set the dimensions of the display device	No action
CALL JVPORT (...)	Set the viewport on the display device	No action
CALL JWINDO (...)	Set the window bordering the objects to be viewed	Set the size of the picture frame (@ 300 dpi) and window (to fit the screen), and set the scale factor
CALL JFRAME	Cause a new frame action	Open a new window of the size calculated in JWINDO
CALL JOPEN	Open a segment	Open a picture with the picture frame set in JWINDO
CALL JMOVE (...)	Move the pen to the global coordinate in the argument	Same as DI-3000
CALL JDRAW (...)	Draw to the global coordinate in the argument	Same as DI-3000
CALL JCLOSE	Close the segment	Close the picture and save it in a PICT file
CALL JPAUSE (1)	Wait for user response	Wait for mouse button click
CALL JDEVOF (1)	Turn off device number 1	No action
CALL JDEND (1)	Terminate device number 1	No action
CALL JEND	Terminate DI-3000	No action

Note: A DI-3000 program need not call JFRAME at the start of the first picture; however, in this emulation it is necessary.

5.1 Metafiles and PICT Files

The DI-3000 graphics package allows several individual frames (pictures) to be saved in one metafile. In the MacTRANS emulation, the frames were replaced by QuickDraw pictures, and the metafile by several PICT files (each containing one QuickDraw picture). PICT files are described more fully in Section 7.

To accommodate the consequential increase in the number of picture files, the file-naming procedure was modified. The new procedure adopted gives each PICT file

produced from a single plot command (PLS or PLO) the same name it would be given in TRANS, but with a single character extension to identify it. These characters start with a lower case "a" and continue alphabetically. For example, a data file with the name "POLAR.DAT" would have the first group of PICT files (from the first plot command) named "POLAR.PLT.a", "POLAR.PLT.b", etc; the second group (from the second plot command) named "POLAR.PL0.a", "POLAR.PL0.b", etc; the third group (from the third plot command) named "POLAR.PL1.a", "POLAR.PL1.b", etc; and so on.

This modification was implemented in the new code associated with the DI-3000 graphics package emulation, but also required changes to several sections of the existing TRANS code. The sections of TRANS code involved were those that

- checked for existing metafiles produced from the current data file (Subroutine FOPEN)
- renamed the metafiles from "META.TEMP" to "<DataFileName>.PLT", "<DataFileName>.PL0", etc. (Subroutine FCLOSE)
- overlaid plots from separate metafiles into one metafile (Subroutines RPT and METRED)

5.2 Common Blocks

Many of the subroutines in the DI-3000 emulation required the use of common blocks to pass values between them. All of these common blocks were included in the variable declarations of Subroutine TRAN2 (the highest level subroutine linking together the subroutines from which the DI-3000 calls are made), and wherever necessary throughout the code (namely the DI-3000 emulation subroutines themselves and any other subroutines that needed access to the common block variables). The following blockdata section, showing all the common blocks and their functions, was used to set the initial default values for the common block variables.

```
CDB THESE DECLARATIONS RELATE TO MAC IMPLEMENTATION OF THE DI3000 CALLS
      BLOCKDATA DI3000
      RECORD /PICHANDLE/PIC(20)           ! HANDLES TO THE PICTURES
      RECORD /RECT/WINDOWRECT,HIRESRECT   ! BOUNDING RECTS FOR WINDOWS AND PICTURES
      COMMON /RECTS/WINDOWRECT,HIRESRECT
      COMMON /PICTURES/PIC
      REAL SCALE                          ! SCALE FACTOR
      COMMON /SCALING/SCALE
      INTEGER JHORIZ,JVERT                ! TEXT JUSTIFICATION INDICES
      COMMON /JUST/JHORIZ,JVERT
      DATA JHORIZ/1/,JVERT/1/             ! SET JUSTIFICATION TO BOTTOM LEFT
      STRING*255 FNAME                    ! DATA FILE NAME
      COMMON /NAME/FNAME
      INTEGER*2 HOFFSET,VOFFSET           ! GRAPHICS WINDOW OFFSET
      COMMON /OFFSET/HOFFSET,VOFFSET
      RECORD /CWINDOWPTR/WPTR(20)         ! POINTERS TO THE GRAPHICS WINDOWS
      INTEGER*4 WINNUM                    ! NUMBER OF GRAPHICS WINDOWS OPEN
      COMMON /WINS/WPTR,WINNUM
      DATA WINNUM/0/                      ! SET NUMBER OF WINDOWS OPEN TO ZERO
      INTEGER*4 MKSYM                     ! MARK SYMBOL INDEX
      COMMON/MARK/MKSYM
      DATA MKSYM/1/                       ! SET MARK SYMBOL TO A DOT
      LOGICAL RFLAG                       ! FLAG TO STOP DISPLAYING PICTURES
      COMMON/FLAG/RFLAG                   ! DURING REP COMMAND
      DATA RFLAG/.FALSE./                 ! SET FLAG TO FALSE, IE SHOW PICTURES
      SAVE /RECTS/,/PICTURES/,/SCALING/,/JUST/,/NAME/,/OFFSET/,/WINS/,/MARK/
      END
```

For the REPeat command, it was necessary to save some of the common blocks between calls to Subroutine TRAN2 (see Section 3.6). Some of the common blocks were also used to pass information about the windows and pictures opened by the DI-3000 emulation to the event-loop section of MacTRANS (see Section 8).

5.3 Dummy DI-3000 Subroutines

The DI-3000 routines listed below were replaced by dummy subroutines which return control straight back to calling code without performing any operations.

JBEGIN	JVSPAC	JDEND	JMREAD
JDINIT	JVPORT	JEND	JMINTR
JDEVON	JDEVOF	JMGET	

5.4 Emulated DI-3000 Subroutines

The following DI-3000 routines were emulated:

JCLOSE	JFRAME	JJUST	JPAUSE
JCMARK	JFSOPN	JMARK	JSIZE
JDRAW	JHSTRG	JMOVE	JWINDO
JFONT	JHTEXT	JOPEN	

A brief description of the code used to perform the actions required by these routines follows.

Note: Code dealing with the variable RFLAG is not strictly part of the emulation: it was included for the convenience of dealing with the REPeat command.

CALL JCLOSE

a) Close the currently open picture

```
CALL CLOSEPICTURE
```

b) If not in REPeat command, draw the picture

```
IF (.NOT. RFLAG) CALL DRAWPICTURE(PIC(WINNUM),WPTR(WINNUM).WP^.PORTRECT)
```

c) Get the name of the current window

```
CALL GETWTITLE(WPTR(WINNUM),%REF(FILENAME))
```

d) If not in REPeat command, save the picture in a PICT file (see Section 7)

```
IF (.NOT. RFLAG) CALL PICSAVE(PIC(WINNUM),FILENAME,ERR)
```

CALL JCMARK (CVALUE)

Set the mark symbol index

```
MKSYM = CVALUE
```

CALL JDRAW (X,Y)

Draw a line from the current pen position to (XX,YY) — calculated by multiplying X and Y by the scaling factor (300 dots per inch), rounding to the nearest integer, and changing the sign of YY (the QuickDraw positive Y direction is down)

```
XX = ININT(X * scale)
YY = -ININT(Y * scale)
CALL LINETO(XX,YY)
```

CALL JFONT (CVALUE)

Change font to "Times" — ignores argument

```
CALL TEXTFONT(TIMES)
```

CALL JFRAME

a) Increment the window number

```
WINNUM = WINNUM +1
```

b) Set the window title to the file name with the character extension

```
WINTITLE = FNAME//'.'//CHAR(96+WINNUM)
```

c) Open a new window

```
WPTR(WINNUM).WP = NEWCWINDOW(NIL,WINDOWRECT,WINTITLE,FLS,
1                            NOGROWDOCPROC,INFRONT,TRU,WINNUM)
```

d) Send the new window to the front

```
CALL SETPORT(WPTR(WINNUM).WP)
```

e) Calculate the window offset and move the window

```
HGLOB = (WINNUM - 1)*HOFFSET + WINDOWRECT.LEFT
VGLOB = (WINNUM - 1)*VOFFSET + WINDOWRECT.TOP
CALL MOVEWINDOW(WPTR(WINNUM).WP,HGLOB,VGLOB,FLS)
```

f) If not in the REPeat command, make the window visible

```
IF (.NOT. RFLAG) CALL SHOWWINDOW(WPTR(WINNUM).WP)
```

CALL JFSOPN (CODE,DSPDEV,LUN,FILNAM)

Set the file name — ignores the remaining arguments

```
FNAME = TRIM(FILNAM)
```

CALL JHSTRG (STRING)

a) Get the current pen location

```
CALL GETPEN(%REF(PT))
```

b) Calculate the length of the string

```
LENGTH = INT2(LEN(STRING))
PIXLEN = TEXTWIDTH(%LOC(STRING),INT2(0),LENGTH)
```

c) Get the font information and move the pen to the position needed for the text justification

```
CALL GETFONTINFO(%REF(FINFO))
SELECT CASE (JHORIZ)
  CASE (1)          !LEFT
  CASE (2)          !CENTRE
    CALL MOVE(-ININT(PIXLEN/2.0),INT2(0))
  CASE (3)          !RIGHT
    CALL MOVE(-PIXLEN,INT2(0))
END SELECT
SELECT CASE (JVERT)
  CASE (1)          !BOTTOM = BASELINE
  CASE (2)          !CENTRE = 1/2 ASCENT
    CALL MOVE(INT2(0),ININT(FINFO.ASCENT/2.0))
  CASE (3)          !TOP = ASCENT LINE
    CALL MOVE(INT2(0),FINFO.ASCENT)
END SELECT
```

d) Draw the text and return the pen to the original position

```
CALL DRAWTEXT(%LOC(STRING),INT2(0),LENGTH)
CALL MOVETO(PT.H,PT.V)   !REVERT TO ORIGINAL LOCATION
```

CALL JHTEXT (NCHARS,RSTR)

a) As for JHSTRG

b) Calculate the length of the string

```
PIXLEN = TEXTWIDTH(%LOC(RSTR),INT2(0),INT2(NCHARS))
```

c) As for JHSTRG

d) Draw the text and return the pen to the original position

```
CALL DRAWTEXT(%LOC(RSTR),INT2(0),INT2(NCHARS))
CALL MOVETO(PT.H,PT.V)   !REVERT TO ORIGINAL POSITION
```

CALL JJUST (CHORIZ,CVERT)

a) Set the text justification indices (arguments shown in Table 3)

```
JHORIZ = CHORIZ
JVERT = CVERT
```

TABLE 3
Justification Arguments for Subroutine JJUST

Argument	1	2	3
CHORIZ	Left	Centre	Right
CVERT	Bottom	Centre	Top

CALL JMARK (X,Y)

a) Move to (X,Y) using a DI-3000 call

```
CALL JMOVE(X,Y)
```

b) Set symbol size to 0.5% of the lessor of the width or height of the picture rectangle

```
      IF ((HIRESRECT.RIGHT - HIRESRECT.LEFT).GT.
     1    (HIRESRECT.BOTTOM - HIRESRECT.TOP)) THEN
        SYMSZ = ININT(0.005*FLOAT(HIRESRECT.BOTTOM - HIRESRECT.TOP))
      ELSE
        SYMSZ = ININT(0.005*FLOAT(HIRESRECT.RIGHT - HIRESRECT.LEFT))
      END IF
C MAKE SYMBOL SIZE AN EVEN NUMBER OF PIXELS
      IF (MOD(SYMSZ,2) .NE. 0) SYMSZ = SYMSZ + 1
```

c) Draw the mark

```
SELECT CASE (MKSYM)
  CASE(1)          ! .
    CALL GETPEN(%REF(PT))
    MRECT.TOP = PT.V - SYMSZ/2
    MRECT.BOTTOM = PT.V + SYMSZ/2 + 1
    MRECT.LEFT = PT.H - SYMSZ/2
    MRECT.RIGHT = PT.H + SYMSZ/2 + 1
    CALL PAINTOVAL(MRECT)
  CASE(2)          ! +
    CALL MOVE(INT2(0),INT2(-SYMSZ/2))
    CALL LINE(INT2(0),SYMSZ)
```

```
    CALL MOVE(INT2(-SYMSZ/2),INT2(-SYMSZ/2))
    CALL LINE(SYMSZ,INT2(0))
  CASE(3)          ! *
    CALL MOVE(INT2(0),INT2(-SYMSZ/2))
    CALL LINE(INT2(0),SYMSZ)
    CALL MOVE(ININT(-0.866*SYMSZ/2.0),INT2(-SYMSZ/4))
    CALL LINE(ININT(0.866*SYMSZ),INT2(-SYMSZ/2))
    CALL MOVE(INT2(0),INT2(SYMSZ/2))
    CALL LINE(ININT(-0.866*SYMSZ),INT2(-SYMSZ/2))
  CASE(4)          ! CIRCLE
    CALL GETPEN(%REF(PT))
    MRECT.TOP = PT.V - SYMSZ/2
    MRECT.BOTTOM = PT.V + SYMSZ/2 + 1
    MRECT.LEFT = PT.H - SYMSZ/2
    MRECT.RIGHT = PT.H + SYMSZ/2 + 1
    CALL FRAMEOVAL(MRECT)
  CASE(5)          ! X
    CALL MOVE(INT2(-SYMSZ/2),INT2(-SYMSZ/2))
    CALL LINE(SYMSZ,SYMSZ)
    CALL MOVE(INT2(0),INT2(-SYMSZ))
    CALL LINE(-SYMSZ,SYMSZ)
  CASE(6)          ! SQUARE
    CALL GETPEN(%REF(PT))
    MRECT.TOP = PT.V - SYMSZ/2
    MRECT.BOTTOM = PT.V + SYMSZ/2 + 1
    MRECT.LEFT = PT.H - SYMSZ/2
    MRECT.RIGHT = PT.H + SYMSZ/2 + 1
    CALL FRAMERECT(MRECT)
END SELECT
```

d) Return the pen to its original position

```
CALL JMOVE(X,Y)     !RETURN TO START POSITION
```

CALL JMOVE (X,Y)

Move the current pen position to (XX,YY) — calculated by multiplying X and Y by the scaling factor (300 dots per inch), rounding to the nearest integer, and changing the sign of YY (the QuickDraw positive Y direction is down)

```
XX = ININT(X * scale)
YY = -ININT(Y * scale)
CALL MOVETO(XX,YY)
```

CALL JOPEN

a) Open a new picture

```
PIC(WINNUM) = OPENPICTURE(HIRESRECT)
```

b) Set the origin to the bottom left corner of the picture frame

```
CALL SETORIGIN(INT2(0),-(HIRESRECT.BOTTOM - HIRESRECT.TOP))
```

CALL JPAUSE

If not in the REPeat command, wait for a button click

```
IF (RFLAG) RETURN
DO WHILE (.NOT. BUTTON)
END DO
```

CALL JSIZE (CXSIZE,CYSIZE)

Change the text size of the current font — ignores CXSIZE as the aspect ratio of the characters cannot be changed in QuickDraw

```
SIZE = ININT(CYSIZE * SCALE)
CALL TEXTSIZE(SIZE)
```

CALL JWINDO (UMIN,UMAX,VMIN,VMAX)

a) Set the picture frame rectangle

```
HIRESRECT.TOP = 0
HIRESRECT.LEFT = 0
HIRESRECT.RIGHT = ININT((UMAX - UMIN) * 300)
HIRESRECT.BOTTOM = ININT((VMAX - VMIN) * 300)
```

b) Set the graphics window rectangle

```
      QDG = JQDGLOBALS()        !GET QUICKDRAW GLOBALS
C     SET WINDOWRECT TO SCREEN DIMENSIONS
      WINDOWRECT = QDG^.SCREENBITS.BOUNDS
C     CALCULATE ASPECT RATIO OF PICTURE FRAME AND SCREEN
      RATIO = (UMAX - UMIN) / (VMAX - VMIN)
      SRATIO = FLOAT(WINDOWRECT.RIGHT - WINDOWRECT.LEFT - 8) /
     1           FLOAT(WINDOWRECT.BOTTOM - WINDOWRECT.TOP - 44)
C     IF ASPECT RATIO OF PICTURE > SCREEN THEN MAKE WINDOW SHALLOWER THAN
C     SCREEN AND OFFSET WINDOWS DOWN, ELSE MAKE WINDOW NARROWER THAN SCREEN
C     AND OFFSET WINDOWS TO THE RIGHT.
      IF (RATIO .GT. SRATIO) THEN
        WINDOWRECT.TOP = WINDOWRECT.TOP + 40
        WINDOWRECT.LEFT = WINDOWRECT.LEFT + 4
        WINDOWRECT.RIGHT = WINDOWRECT.RIGHT - 4
        WINDOWRECT.BOTTOM = ININT(FLOAT(WINDOWRECT.RIGHT -
     1        WINDOWRECT.LEFT) / RATIO) + WINDOWRECT.TOP
        HOFFSET = 0
        VOFFSET = 20
      ELSE
        WINDOWRECT.TOP = WINDOWRECT.TOP + 40
        WINDOWRECT.LEFT = WINDOWRECT.LEFT + 4
        WINDOWRECT.BOTTOM = WINDOWRECT.BOTTOM - 4
        WINDOWRECT.RIGHT = ININT(FLOAT(WINDOWRECT.BOTTOM -
     1        WINDOWRECT.TOP) * RATIO) + WINDOWRECT.LEFT
        HOFFSET = 20
        VOFFSET = 0
      END IF
```

b) Set the scaling factor to 300 dots per inch

```
SCALE = 300.0
```

6. COLOUR

To add colour, "CWindows" were used instead of normal windows. "CWindows" are based on "CGrafPorts" which allow full RGB colour control. The pen colour in a "CGrafPort" is changed by the QuickDraw call

```
CALL RGBFORECOLOR(COLOR)
```

where COLOR is a variable of type RGBCOLOR with a Language Systems Fortran structure:

```
STRUCTURE /RGBCOLOR/
     INTEGER*2          RED
     INTEGER*2          GREEN
     INTEGER*2          BLUE
END STRUCTURE
```

Red, green, and blue specify the amount of each colour component ranging from 0 to 65535 ($0000 – $FFFF).

Colours were assigned to the eight individual line types used in TRANS for data plotting by defining a new variable structure COLORARRAY (see Appendix A), as shown below, and using a one dimensional array COLOR of that type with eight elements to record the RGB colour values together with a text description of the colours for each line type.

```
STRUCTURE/COLORARRAY/
  CHARACTER*9 TEXT
  RECORD/RGBCOLOR/ RGB
END STRUCTURE
```

In addition, a separate variable of the same type was used to return the pen colour to black after the coloured lines were drawn (all borders, scales, and text were left black). The default colours were initialised in data statements, shown below, and passed to other subroutines through a common block, also shown below.

```
RECORD/COLORARRAY/COLOR(8),BLKCOL
COMMON/COL/COLOR,BLKCOL
DATA COLOR(1).TEXT/'BLACK    '/,COLOR(2).TEXT/'RED      '/,
1    COLOR(3).TEXT/'GREEN    '/,COLOR(4).TEXT/'BLUE     '/,
2    COLOR(5).TEXT/'CYAN     '/,COLOR(6).TEXT/'MAGENTA  '/,
3    COLOR(7).TEXT/'YELLOW   '/,COLOR(8).TEXT/'ORANGE   '/
DATA COLOR(1).RGB.RED/$0000/,COLOR(1).RGB.GREEN/$0000/,COLOR(1).RGB.BLUE/$0000/
DATA COLOR(2).RGB.RED/$FFFF/,COLOR(2).RGB.GREEN/$0000/,COLOR(2).RGB.BLUE/$0000/
DATA COLOR(3).RGB.RED/$0000/,COLOR(3).RGB.GREEN/$FFFF/,COLOR(3).RGB.BLUE/$0000/
DATA COLOR(4).RGB.RED/$0000/,COLOR(4).RGB.GREEN/$0000/,COLOR(4).RGB.BLUE/$FFFF/
DATA COLOR(5).RGB.RED/$0000/,COLOR(5).RGB.GREEN/$FFFF/,COLOR(5).RGB.BLUE/$FFFF/
DATA COLOR(6).RGB.RED/$FFFF/,COLOR(6).RGB.GREEN/$0000/,COLOR(6).RGB.BLUE/$FFFF/
DATA COLOR(7).RGB.RED/$FFFF/,COLOR(7).RGB.GREEN/$FFFF/,COLOR(7).RGB.BLUE/$0000/
DATA COLOR(8).RGB.RED/$FFFF/,COLOR(8).RGB.GREEN/$61A8/,COLOR(8).RGB.BLUE/$0000/
DATA BLKCOL.TEXT/'BLACK    '/
DATA BLKCOL.RGB.RED/$0000/,BLKCOL.RGB.GREEN/$0000/,BLKCOL.RGB.BLUE/$0000/
```

To change the default colours, code was inserted into the section of Subroutine TRAN2 dealing with the PLS command. This code worked on the principle of reading a line number and colour typed on the keyboard into a character variable as a response to a prompt, extracting the number to identify the line, and comparing the text description of the colour to those of the various colours shown above. For example:

```
      TYPE 333,(COLOR(I).TEXT,I=1,8)
333   FORMAT(1X,'THE CURRENT LINE COLOURS ARE:',/
     1       1X,'0 & 1    2        3        4    ',/
     2       1X,4A9,/
     3       1X,'5        6        7        8    ',/
     4       1X,4A9,/
     5       1X,'AVAILABLE COLOURS :'/
     6       1X,'BLACK, RED, GREEN, LTGREEN, BLUE, LTBLUE, CYAN,'/
     7       1X,'MAGENTA, PURPLE, YELLOW, ORANGE.'/
     5       1X,'DO YOU WISH TO CHANGE THE COLOURS? : ',$)
      ACCEPT 334,ANSWER
334   FORMAT(A1)
      IF ((ANSWER .NE. "Y").AND.(ANSWER .NE. "y")) GO TO 336
      TYPE *,'LINE  COLOUR'
339   CONTINUE
      READ(5,'(A)') COLSTR
      IF (COLSTR.EQ.'') GO TO 340
      COLSTR = ADJUSTL(COLSTR)                    !REMOVE LEADING SPACES
      DO I=1,LEN(TRIM(COLSTR))                    !FIND POSITION OF COMMA
        IF (COLSTR(I:I) .EQ. ',') GO TO 341
      END DO
```

```
341   READ(COLSTR(1:I-1),*) II                    !READ THE LINE NUMBER
      IF(II.EQ.0) II = 1
      IF(II.GT.8) THEN
        TYPE *,'ONLY LINE TYPES 1 TO 8 EXIST'
        GO TO 339
      END IF
      COLOR(II).TEXT = ADJUSTL(COLSTR(I+1:))      !CHANGE THE TEXT DESCRIPTION
      SELECT CASE (COLOR(II).TEXT)                !CHANGE THE COLOUR
        CASE ('BLACK     ')
          COLOR(II).RGB.RED   = $0000
          COLOR(II).RGB.GREEN = $0000
          COLOR(II).RGB.BLUE  = $0000
        CASE ('RED       ')
          COLOR(II).RGB.RED   = $FFFF
          COLOR(II).RGB.GREEN = $0000
          COLOR(II).RGB.BLUE  = $0000
        CASE ('GREEN     ')
          COLOR(II).RGB.RED   = $0000
          COLOR(II).RGB.GREEN = $FFFF
          COLOR(II).RGB.BLUE  = $0000
        CASE ('BLUE      ')
          COLOR(II).RGB.RED   = $0000
          COLOR(II).RGB.GREEN = $0000
          COLOR(II).RGB.BLUE  = $FFFF
        CASE ('CYAN      ')
          COLOR(II).RGB.RED   = $0000
          COLOR(II).RGB.GREEN = $FFFF
          COLOR(II).RGB.BLUE  = $FFFF
        CASE ('MAGENTA   ')
          COLOR(II).RGB.RED   = $FFFF
          COLOR(II).RGB.GREEN = $0000
          COLOR(II).RGB.BLUE  = $FFFF
        CASE ('YELLOW    ')
          COLOR(II).RGB.RED   = $FFFF
          COLOR(II).RGB.GREEN = $FFFF
          COLOR(II).RGB.BLUE  = $0000
        CASE ('ORANGE    ')
          COLOR(II).RGB.RED   = $FFFF
          COLOR(II).RGB.GREEN = $6000
          COLOR(II).RGB.BLUE  = $0000
        CASE ('PURPLE    ')
          COLOR(II).RGB.RED   = $80E8
          COLOR(II).RGB.GREEN = $0000
          COLOR(II).RGB.BLUE  = $FFFF
        CASE ('LTBLUE    ')
          COLOR(II).RGB.RED   = $0000
          COLOR(II).RGB.GREEN = $9C40
          COLOR(II).RGB.BLUE  = $FFFF
        CASE ('LTGREEN   ')
          COLOR(II).RGB.RED   = $86B0
          COLOR(II).RGB.GREEN = $FFFF
          COLOR(II).RGB.BLUE  = $0000
        CASE DEFAULT
          TYPE *,'NOT A VALID COLOUR'
      END SELECT
      GO TO 339
340   CONTINUE
336   CONTINUE
```

The sections of code where the pen colour was actually changed with calls to `RGBForeColor` are set out in Table 4.

TABLE 4
Sections of Code Where Pen Colour is Changed

Section	Subroutine
Repeat Line Key	REP command section of TRAN2
Plot	STRIP
Overlay Line Key	GRID

7. PICT FILES

PICT files are the standard Macintosh files for transferring graphical data between programs. The PICT file format (see page 85, Vol 5, Ref. 6) consists of a 512 byte header followed by opcodes which contain QuickDraw picture information.

PICT files were used to store the plots generated by MacTRANS because they can be read by most commercial graphics software, thus removing the need for a graphics editor within MacTRANS.

In MacTRANS, two separate subroutines were written for saving pictures to (Subroutine PICSAVE), and retrieving pictures from (Subroutine LOADPIC), PICT files. The procedures for these subroutines are shown below.

7.1 Procedure Used to Save a Picture in a PICT File

a) Get the current volume number

```
OSERR = GETVOL(NIL, %REF(VREFNUM))
```

b) Create a new file

```
OSERR = CREATE(FILENAME, VREFNUM, CREATOR, FILETYPE)
```

c) Open the new file

```
OSERR = FSOPEN(FILENAME, VREFNUM, %REF(FILENO))
```

d) Set the file marker to the start of the file

```
OSERR = SETFPOS(FILENO, FSFROMSTART, INT4(0))
```

e) Write 128 integer*4 zeros to the 512 byte header — the header was not used to store any information

```
DO I = 1,128
  COUNT = 4
  OSERR = FSWRITE(FILENO, %REF(COUNT), %LOC(ZERO))
END DO
```

f) Get the size (in bytes) of the picture in memory

```
COUNT = GETHANDLESIZE(PICTURE)
```

g) Dump this number of bytes starting from the address of the picture into the PICT file

```
OSERR = FSWRITE(FILENO, %REF(COUNT), PICTURE.PICH^.PICP)
```

h) Close the file

```
OSERR = FSCLOSE(FILENO)
```

i) Flush the volume buffer

```
OSERR = FLUSHVOL(NIL, VREFNUM)
```

7.2 Procedure Used to Retrieve a Picture from a PICT File

a) Call the Macintosh Toolbox standard "Get File" dialog box

```
CALL SFGETFILE(WHERE, NIL, NIL, NUMTYPES, TYPELIST, NIL, %REF(REPLY))
```

b) If the dialog box succeeds in selecting a PICT file, set the volume to that where the file is located

```
IF (REPLY.GOOD) THEN
  FILENAME = REPLY.FNAME
  OSERR = SETVOL(NIL, REPLY.VREFNUM)
END IF
```

c) Open the file

```
OSERR = FSOPEN(REPLY.FNAME, REPLY.VREFNUM, %REF(FILENO))
```

d) Get the size of the file

```
OSERR = GETEOF(FILENO, %REF(COUNT))
```

e) Subtract 512 bytes from the size of the file

```
COUNT = COUNT - 512
```

f) Create a new handle for the picture

```
PICTURE.PICH = NEWHANDLE(COUNT)
```

g) Lock the picture handle

```
CALL HLOCK(PICTURE)
```

h) Set the file marker to the start of the picture information

```
OSERR = SETFPOS(FILENO, FSFROMSTART, INT4(512))
```

i) Read the picture information into memory

```
OSERR = FSREAD(FILENO, %REF(COUNT), PICTURE.PICH^.PICP)
```

j) Unlock the picture handle

```
CALL HUNLOCK(PICTURE)
```

k) Close the file

```
OSERR = FSCLOSE(FILENO)
```

l) Flush the volume buffer

```
OSERR = FLUSHVOL(NIL, REPLY.VREFNUM)
```

8. EVENT-LOOP

In a Macintosh application, an event-loop examines the events that occur while a program is executing, e.g. mouse button "clicks" and menu item selection, and calls the relevant subroutines to provide the appropriate actions.

In this conversion, full use of the Macintosh operating system through the event-loop was not made, as this would have needed a considerable amount of code rewriting. Instead, a limited event-loop with the features of moving windows, opening and closing PICT files, and printing was added to the end of the program so that it is only entered just before the execution of TRANS would normally finish.

This means that MacTRANS operates in almost exactly the same fashion as TRANS does, but at the end of the program, allows all the graphics windows produced during that run to be viewed on the screen at the same time, moved around, printed, etc. Plots created from previous runs can also be opened in the event loop, and for the cases when it is desired to use MacTRANS solely to view existing plots, a switch was added to the start of MacTRANS to enable the program to skip straight to the event-loop, by-passing the plotting prompts.

The Language Systems Fortran compiler also automatically adds a very limited sort of event-loop around the entire program. This provides an output window for the program input/output and, after the program finishes, provides menus which allow the user to first edit and print the text in the output window, and then rerun the program. This event-loop is not discussed here. For further information see Ref. 4.

The event-loop is handled by five subroutines: Loop, EventHandler, Do_Menu, SetUp, and Open_Win. These subroutines are based on code used in the example program "EventLoop" supplied with Language Systems Fortran. Loop contains the main loop which detects the events as they happen. EventHandler is called from Loop whenever an event is detected. It determines what type of event it is, and calls Do_Menu or Open_Win and/or Macintosh Toolbox procedures to perform the required actions. Do_Menu determines which menu item was selected and does, or calls other subroutines which do, all the actions required by the menu selections. Similarly, Open_Win does, or calls other subroutines which do, all the actions required by window events. It has several entry points to it which in essence makes it a composite of several subroutines, each dealing with a different window action (see Section 8.5). SetUp is the subroutine that creates all the menus for the event-loop.

Three menus are used in MacTRANS: an Apple menu for the About MacTRANS dialog box, and access to the desk accessories; a File menu for opening, closing, and printing the PICT files and associated graphics windows, and for quitting the program; and a Window menu for selecting the graphics windows, i.e. bringing them to the front. The Apple and File menus are defined in the resource fork of MacTRANS, along with the icons for MacTRANS and the PICT files, and the dialog box brought up when the About MacTRANS item is selected from the Apple menu. For details on how the resource fork of MacTRANS was added, see Section 10.

The variables dealing with the graphics windows are passed into the event-loop part of the program via common blocks. Specifically, these blocks are:

```
RECORD /RECT/ WINDOWRECT,HIRESRECT
COMMON /RECTS/WINDOWRECT,HIRESRECT
RECORD /CWINDOWPTR/ WPTR(20)
INTEGER*4 WINNUM
COMMON /WINS/ WPTR,WINNUM
RECORD /PICHANDLE/ PIC(20)
COMMON /PICTURES/ PIC
```

WindowRect and HiResRect are the rectangles bounding the graphics windows and picture frames respectively. These are used when PICT files are opened and displayed. Pointers to the graphics windows are in the array WPTR, and handles to the pictures in the array PIC (PIC(1) corresponds to WPTR(1), etc.). Winnum is an integer equal to the number of graphics windows.

A brief description of the code used in the event-loop subroutines to perform the actions required follows.

8.1 Subroutine Loop

a) Calls SetUp to create menus

```
CALL SETUP
```

b) Frees the memory used by SetUp

```
CALL UNLOADSEG(%LOC(SETUP
```

c) Sets the event mask and flags which indicate if Quit or Exit have been selected

```
EVENT_MASK = $FFFF      !ALL EVENTS
DONE = .FALSE.          !EXIT TO OUTPUT WIN.  W FLAG
FINDER = .FALSE.        !QUIT TO FINDER FLAG
```

d) Enters the event-loop

```
DO WHILE (DONE = .FALSE.)
    AN_EVENT = GETNEXTEVENT(EVENT_MASK,%REF(THEEVENT))
    IF (AN_EVENT) CALL EVENTHANDLER(THEEVENT,DONE,FINDER)
END DO
```

e) After leaving the event-loop, quits the Language System Fortran event-loop if Quit was selected

```
IF (FINDER) CALL DOMENU('FILE','QUIT')
```

8.2 Subroutine SetUp

a) Get the Apple and File menus from the resource fork of MacTRANS, add the desk accessories to the Apple menu, and create the Window menu

```
APPLEMENU.MENUH = GETMENU(APPLEID)
CALL INSERTMENU (APPLEMENU,ZERO)
CALL ADDRESMENU (APPLEMENU,'DRVR')
FILEMENU.MENUH = GETMENU(FILEID)
CALL INSERTMENU (FILEMENU,ZERO)
WINMENU.MENUH = NEWMENU(WINMENID,WINMENTIT)
CALL INSERTMENU (WINMENU,ZERO)
```

b) If no graphics windows are open, then disable the relevant menu items; if there are graphics windows open, then add their titles to the Window menu

```
IF (WINNUM .EQ. 0) THEN      !IF THERE ARE NO EXISTING WINDOWS
    CALL DISABLEITEM(FILEMENU,INT2(2))      !CLOSE
    CALL DISABLEITEM(FILEMENU,INT2(4))      !PRINT
    CALL DISABLEITEM(FILEMENU,INT2(5))      !PAGE SETUP
    CALL DISABLEITEM(WINMENU,INT2(0))       !WINDOW
    CALL HILITEMENU(ZERO)                   !THIS UN-HILIGHTS ALL MENUS
    CALL FLUSHEVENTS($FFFF,ZERO)            !START FRESH ON EVENTS
ELSE      !IF THERE ARE EXISTING WINDOWS ADD THEM TO THE WINDOW MENU
    DO J = 1,WINNUM
        CALL GETWTITLE(WPTR(J).WP,WINMENTIT)
        CALL APPENDMENU(WINMENU,WINMENTIT)
    END DO
END IF
```

c) Send the output window behind the graphics windows and draw the menu bar

```
CALL SENDBEHIND(OUTPUTWINDOW(),NIL)
CALL DRAWMENUBAR
```

8.3 Subroutine EventHandler

a) Get the QuickDraw global variables and set the exit flag to false

```
QDG = JQDGLOBALS()
DONE = .FALSE.
```

b) Determine the type of the event and perform, or call the appropriate subroutine to perform, the required actions

```
SELECT CASE (THEEVENT.WHAT)
  CASE (MOUSEDOWN)
    WINDOWPART = FINDWINDOW(THEEVENT.WHERE,%REF(WINP))
    SELECT CASE (WINDOWPART)
      CASE (INMENUBAR)                              !MENUBAR
        CALL DO_MENU(MENUSELECT(THEEVENT.WHERE),DONE,FINDER)
      CASE (INSYSWINDOW)                            !IN SYS WINDOW
        CALL SYSTEMCLICK(THEEVENT,WINP)
      CASE (INCONTENT)                              !CONTENT REGION
        IF (WINP.WP .NE. FRONTWINDOW) CALL SELECTWINDOW(WINP.WP)
        WREC = WINP.WP
        IF ((WREC^.WINDOWKIND .NE. USERKIND).OR.
1           (WINP.WP .EQ. OUTPUTWINDOW())) THEN
          CALL DISABLEITEM(FILEMENU,INT2(4))        !PRINT
          CALL DISABLEITEM(FILEMENU,INT2(5))        !PAGE SETUP
        ELSE
          CALL ENABLEITEM(FILEMENU,INT2(4))         !PRINT
          CALL ENABLEITEM(FILEMENU,INT2(5))         !PAGE SETUP
        END IF
        CALL DRAWMENUBAR
      CASE (INDRAG)                                 !DRAG REGION
        IF (WINP.WP .NE. OUTPUTWINDOW()) THEN
          IF (WINP.WP .NE. FRONTWINDOW) CALL SELECTWINDOW(WINP.WP)
          LIMITRECT = QDG^.SCREENBITS.BOUNDS
          CALL DRAGWINDOW(WINP,THEEVENT.WHERE,LIMITRECT)
        END IF
      CASE (INGROW)                                 !SIZE REGION
      CASE (INGOAWAY)                               !CLOSE BOX
        IF (WINP.WP .NE. OUTPUTWINDOW()) THEN
          CLOSEIT = TRACKGOAWAY(WINP,THEEVENT.WHERE)
          IF (CLOSEIT) CALL CLOSE_WIN(WINP)
        END IF
      CASE DEFAULT
        ! NO OTHER WINDOW PARTS TO DEAL WITH.
    END SELECT                                      !END OF MOUSEDOWN EVENT.
  CASE (MOUSEUP)                                    !MOUSE UP
    !THIS PROGRAM DOES NOTHING IN RESPONSE TO THIS EVENT.
  CASE ( KEYDOWN)                                   !KEY PRESS
    CHCODE = JIAND(THEEVENT.MESSAGE,CHARCODEMASK)
    CH = CHAR(CHCODE)
    CHCODE = JIAND(THEEVENT.MODIFIERS,CMDKEY)
    IF (CHCODE .NE. 0) CALL DO_MENU(MENUKEY(CH),DONE,FINDER)
  CASE (KEYUP)
    !THIS PROGRAM DOES NOTHING IN RESPONSE TO THIS EVENT.
  CASE (AUTOKEY)
    !THIS PROGRAM DOES NOTHING IN RESPONSE TO THIS EVENT.
  CASE (UPDATEEVT)
    CALL REDRAW_WIN(THEEVENT.MESSAGE)
  CASE (DISKEVT)
    !THIS PROGRAM DOES NOTHING IN RESPONSE TO THIS EVENT.
  CASE (ACTIVATEEVT)
    CALL SETPORT(THEEVENT.MESSAGE)
  CASE DEFAULT
END SELECT
```

8.4 Subroutine Do_Menu

a) Determine the menu and menu item from the first argument in the call to Do_Menu

```
PORTTEMP = 32767
IF (MENUCHOICE > 0) THEN
```

```
ITEM = JIAND(MENUCHOICE,PORTTEMP)     !EXTRACT THE MENU & ITEM NUMBERS
MENU = JISHFT(MENUCHOICE,-16)
```

b) Perform the appropriate actions based on the menu and menu item

```
IF (MENU = 1) THEN                    !APPLE MENU
  IF (ITEM = 1) THEN                  !ABOUT EVENTLOOP
C SINCE DPTR IS NEVER GOING TO BE DEREFERENCED, IT CAN
C JUST BE AN INTEGER*4
    DPTR = GETNEWDIALOG(INT2(13748),INT4(0),-1)  !DIALOG RES ID = 13748
    IF (DPTR .NE. 0) THEN
      CALL GETPORT(%REF(PORTTEMP))
      CALL SETPORT(DPTR)
      CALL DRAWDIALOG(DPTR)
      DO WHILE (.NOT. BUTTON)
      END DO
      CALL DISPOSDIALOG(DPTR)
      CALL SETPORT(PORTTEMP)
    END IF
  ELSE                                !ITEM IS A DESK ACCESSORY
    CALL GETITEM(APPLEMENU,ITEM,%REF(ITEMSTRING))
    CALL GETPORT(%REF(OLDPORT))
    TEMP = OPENDESKACC(ITEMSTRING)
    CALL SETPORT(OLDPORT)
  END IF
ELSE IF (MENU = 2) THEN               !FILE MENU
  IF (ITEM = 1) THEN                  !OPEN
    CALL OPEN_WIN
  ELSE IF (ITEM = 2) THEN             !CLOSE
    WINP.WP = FRONTWINDOW()
    CALL CLOSE_WIN(WINP)
  ELSE IF (ITEM = 4) THEN             !PRINT
    WINP.WP = FRONTWINDOW()
    WNUM = GETWREFCON(WINP.WP)
    CALL PRINTPIC(PIC(WNUM),HPRINT)
  ELSE IF (ITEM = 5) THEN             !PAGE SETUP
    CALL PAGESETUP(HPRINT)            !MODIFY PRINT RECORD
  ELSE IF (ITEM = 7) THEN             !EXIT TO OUTPUT WINDOW
    DONE = .TRUE.
    WINP.WP = FRONTWINDOW()
    CALL CLOSE_ALL_WIN
  ELSE IF (ITEM = 8) THEN             !QUIT TO FINDER
    DONE = .TRUE.
    FINDER = .TRUE.
    CALL CLOSE_ALL_WIN
  END IF
ELSE IF (MENU = WINMENID) THEN        !WINDOW MENU
  CALL SELECTWINDOW(WPTR(ITEM).WP)
END IF
END IF
```

c) Turn off the menu bar highlighting

```
CALL HILITEMENU(INT2(0))
```

8.5 Subroutine Open_Win

Subroutine Open_Win has several entry points which are dealt with separately below.

CALL OPEN_WIN

a) Set the true and false variables and increment the window number

```
FLS = .FALSE.
TRU = .TRUE.
WINNUM = WINNUM + 1
```

b) Check if too many windows will be open

```
IF (WINNUM > 20) THEN
  CALL ALERTBOX('YOU HAVE EXCEEDED THE MAXIMUM NUMBER OF WINDOWS.')
  WINNUM = 10
  GO TO 10     !RETURN
END IF
```

c) Load the new picture from the PICT file

```
CALL LOADPIC(PIC(WINNUM),WINTITLE,ERR)
```

d) Append the window title to the "Window" menu

```
CALL APPENDMENU(WINMENU,WINTITLE)
```

e) Calculate the window size

```
      QDG = JQDGLOBALS()
      WINDOWRECT = QDG^.SCREENBITS.BOUNDS
C     CALCULATE ASPECT RATIOS OF PICTURE FRAME AND SCREEN
      RATIO = FLOAT(PIC(WINNUM).PICH^.PICP^.PICFRAME.RIGHT -
     1              PIC(WINNUM).PICH^.PICP^.PICFRAME.LEFT)
     2      / FLOAT(PIC(WINNUM).PICH^.PICP^.PICFRAME.BOTTOM -
     3              PIC(WINNUM).PICH^.PICP^.PICFRAME.TOP)
      SRATIO = FLOAT(WINDOWRECT.RIGHT - WINDOWRECT.LEFT - 8) /
     1         FLOAT(WINDOWRECT.BOTTOM - WINDOWRECT.TOP - 44)
C     IF FRAME RATIO > SCREEN RATIO THEN MAKE WINDOW SHALLOWER THAN SCREEN AND
C     OFFSET DOWN OTHERWISE MAKE WINDOW NARROWER THAN SCREEN AND SET OFFSET TO
C     THE RIGHT
      IF (RATIO .GT. SRATIO) THEN
        WINDOWRECT.TOP = WINDOWRECT.TOP + 40
        WINDOWRECT.LEFT = WINDOWRECT.LEFT + 4
        WINDOWRECT.RIGHT = WINDOWRECT.RIGHT - 4
        WINDOWRECT.BOTTOM = ININT(FLOAT(WINDOWRECT.RIGHT - WINDOWRECT.LEFT) /
     1                            RATIO) + WINDOWRECT.TOP
        HOFFSET = 0
        VOFFSET = 20
      ELSE
        WINDOWRECT.TOP = WINDOWRECT.TOP + 40
        WINDOWRECT.LEFT = WINDOWRECT.LEFT + 4
        WINDOWRECT.BOTTOM = WINDOWRECT.BOTTOM - 4
        WINDOWRECT.RIGHT = ININT(FLOAT(WINDOWRECT.BOTTOM - WINDOWRECT.TOP) *
     1                           RATIO) + WINDOWRECT.LEFT
        HOFFSET = 20
        VOFFSET = 0
      END IF
```

f) Open the new window

```
      WPTR(WINNUM).WP = NEWWINDOW(NIL,WINDOWRECT,WINTITLE,FLS,NOGROWDOCPROC,
     1                            INFRONT,TRU,WINNUM)
```

g) If the new window was opened, calculate the offset, move the window, make it visible, make it the current window, and enable the relevant menu items; otherwise, send an error message and decrement the number of windows.

```
IF (WPTR(WINNUM).WP .NE. NIL) THEN
  HGLOB = (WINNUM - 1)*HOFFSET + WINDOWRECT.LEFT
  VGLOB = (WINNUM - 1)*VOFFSET + WINDOWRECT.TOP
```

```
        CALL MOVEWINDOW(WPTR(WINNUM).WP,HGLOB,VGLOB,FLS)
C     THE WINDOW WAS INITIALLY INVISIBLE SO IT COULD BE MOVED BEFORE BEING SHOWN
        CALL SHOWWINDOW(WPTR(WINNUM).WP)
        CALL SETPORT(WPTR(WINNUM).WP)
        IF (WINNUM = 1) THEN    ! IF THIS IS THE FIRST WINDOW ENABLE THE MENUS
          CALL ENABLEITEM(FILEMENU,INT2(2))        !CLOSE
          CALL ENABLEITEM(FILEMENU,INT2(4))        !PRINT
          CALL ENABLEITEM(FILEMENU,INT2(5))        !PAGE SETUP
          CALL ENABLEITEM(WINMENU,INT2(0))         !WINDOW
        END IF
        CALL DRAWMENUBAR
      ELSE
        CALL ALERTBOX('THE WINDOW COULD NOT BE OPENED.')
        WINNUM = WINNUM - 1
      END IF
```

CALL CLOSE_WIN(CLWIN)

a) If the window selected is the output window, do nothing

```
IF (CLWIN.WP .EQ. OUTPUTWINDOW()) RETURN
```

b) If the window selected was created by the program

```
IF (CLPEEK^.WINDOWKIND = USERKIND) THEN
```

then do (c), (d), (e), and (f); otherwise, do (g)

c) Move to the selected window, hide it, dispose of it, and release the memory used by the picture associated with it

```
CALL SETPORT(CLWIN.WP)
WNUM = GETWREFCON(CLWIN.WP)       !FIND OUT WHICH WINDOW #
CALL HIDEWINDOW(CLWIN.WP)
CALL DISPOSEWINDOW(CLWIN.WP)
CALL KILLPICTURE(PIC(WNUM))
```

d) If it is the last window, go to (e); otherwise, renumber the windows and pictures to fill the space

```
        IF (WNUM = WINNUM) GO TO 6      !IF IT'S THE LAST WINDOW
C       OTHERWISE MOVE THE HIGHER WINDOWS DOWN TO FILL THE SPACE
        DO 5 J=WNUM,(WINNUM-1)
          WPTR(J).WP = WPTR(J+1).WP
          PIC(J) = PIC(J+1)
          CALL SETWREFCON(WPTR(J).WP,J)
          CALL GETWTITLE(WPTR(J+1),%REF(WINTITLE))
          CALL SETWTITLE(WPTR(J).WP,WINTITLE)
          CALL SELECTWINDOW(WPTR(WINNUM).WP)
5       CONTINUE
```

e) Decrement the number of windows

```
6       WINNUM = WINNUM - 1
```

f) If no windows are open, then disable the relevant menu items; otherwise, remake the "Window" menu

```
        IF (WINNUM = 0) THEN        !IF NO WINDOWS OPEN DISABLE MENUS
          CALL DISABLEITEM(FILEMENU,INT2(2))      !CLOSE
          CALL DISABLEITEM(FILEMENU,INT2(4))      !PRINT VERT
          CALL DISABLEITEM(FILEMENU,INT2(5))      !PRINT HORIZ
          CALL DELETEMENU(WINMENID)
          CALL DISPOSEMENU(WINMENU)
          WINMENU.MENUH = NEWMENU(WINMENID,WINMENTIT)
          CALL INSERTMENU(WINMENU,INT2(0))
          CALL DISABLEITEM(WINMENU,INT2(0))       !TURN OFF WINDOW MENU
```

```
    CALL DRAWMENUBAR
  ELSE                              !RECONFIGURE WINDOW MENU
    CALL DELETEMENU(WINMENID)
    CALL DISPOSEMENU(WINMENU)
    WINMENU.MENUH = NEWMENU(WINMENID,WINMENTIT)
    DO J=1,WINNUM
      CALL GETWTITLE(WPTR(J).WP,WINTITLE)
      CALL APPENDMENU(WINMENU,WINTITLE)
    END DO
    CALL INSERTMENU(WINMENU,INT2(0))
    CALL DRAWMENUBAR
    CALL SETPORT(FRONTWINDOW)
  END IF
```

g) If the window selected was not created by the program, send it behind the other windows

```
  ELSE
    CALL SENDBEHIND(CLWIN,NIL)
  END IF
```

CALL CLOSE_ALL_WIN

a) Close all the graphics windows

```
  DO 15 J = 1,WINNUM
    CALL SETPORT(WPTR(J).WP)
    CALL HIDEWINDOW(WPTR(J).WP)
    CALL DISPOSEWINDOW(WPTR(J).WP)
15  CONTINUE
```

b) Set the number of windows to zero

```
  WINNUM = 0
```

c) Disable the relevant menu items

```
  CALL DISABLEITEM(FILEMENU,INT2(2))      !CLOSE
  CALL DISABLEITEM(FILEMENU,INT2(4))      !PRINT
  CALL DISABLEITEM(FILEMENU,INT2(5))      !PAGE SETUP
  CALL DRAWMENUBAR
```

CALL REDRAW_WIN(CLWIN)

a) Get the current window

```
  CALL GETPORT(%REF(OLDPORT))             !REMEMBER CURRENT PORT
```

b) Move to the window to be redrawn

```
  CALL SETPORT(CLWIN)
```

c) Erase and redraw the window

```
  CALL BEGINUPDATE(CLWIN)
  IF (CLWIN.WP .NE. OUTPUTWINDOW()) THEN
    TEMPRECT = CLWIN.WP^.PORTRECT
    CALL ERASERECT(TEMPRECT)
    WNUM = GETWREFCON(CLWIN)              !GET THE PICTURE NUMBER
    CALL DRAWPICTURE(PIC(WNUM),TEMPRECT)  !REDRAW THE PICTURE
  ELSE
       CALL F_DRAWOUTPWINDOW
       CALL DRAWCONTROLS(CLWIN)
       CALL DRAWGROWICON(CLWIN)
  END IF
  CALL ENDUPDATE(CLWIN)
```

d) Move back to the original window

```
CALL SETPORT(OLDPORT)
```

9. PRINTING ON THE APPLE LASERWRITER

In this conversion, the usual printing procedures were adopted, i.e. the selection of Page Setup and/or Print from the File menu. These menu items are handled by three subroutines: PrintPic, PageSetUp, and CheckPrintHandle.

CheckPrintHandle is called by both PageSetUp and PrintPic to check that the print record (HPRINT) passed to the subroutines is valid. If it is not, a new print record is created and the default print settings from the printer resource file are assigned to it.

PageSetUp allows the default print setting to be changed by calling the standard page set-up dialog box.

PrintPic prints the picture with the print record settings passed to it. It sets the LaserWriter resolution to 300 dpi, adjusts the print rectangle to the same size as the picture frame, positions the print rectangle to allow for adequate margins on the page (which depend on whether the page is printed vertically or horizontally), and calls the standard print dialog box to set the number of copies, pages, etc.

A brief description of the code used in the printing subroutines to perform these actions follows.

9.1 Subroutine CheckPrintHandle

If the print record is non-existent, create a new one and assign the default print settings to it; otherwise, check the record to ensure it is correct

```
IF (HPRINT.TH .EQ. NIL) THEN          !IF HPRINT NON-EXISTENT
  HPRINT.TH = NEWHANDLE(INT4(120))    !CREATE A NEW HANDLE FOR HPRINT
  CALL PRINTDEFAULT(HPRINT)           !SET THE DEFAULT VALUES FOR THE PRINTER
ELSE                                  !VALIDATE EXISTING PRINT RECORD
  IF (PRVALIDATE(HPRINT)) THEN        !CHECK RECORD AND ADJUST IF NECESSARY
  ELSE                                !NO ADJUSTMENT NECESSARY
  END IF
END IF
```

9.2 Subroutine PageSetUp

a) Open the print manager

```
CALL PROPEN
```

b) Check the print handle

```
CALL CHECKPRINTHANDLE(HPRINT)
```

c) Call the standard page set-up dialog box

```
IF (PRSTLDIALOG(HPRINT)) THEN
END IF
```

d) Close the print manager

```
CALL PRCLOSE
```

9.3 Subroutine PrintPic

a) As for PageSetUp

b) As for PageSetUp

c) Set the LaserWriter resolution to 300 dots per inch

```
SPRDATA.IOPCODE = 5
SPRDATA.HPRINT = HPRINT
SPRDATA.IXRSL = 300
SPRDATA.IYRSL = 300
CALL PRGENERAL(%LOC(SPRDATA))
```

d) Set the print rectangle - depending on the orientation

```
      PRINTRECT = HPRINT.TH^.TP^.PRINFO.RPAGE      !SET PRINT RECT TO DEFAULT PAGE
C     OFFSET PRINTRECT TO ALLOW FOR MARGIN - DIFFERENT
C     AMOUNT FOR VERTICAL OR HORIZONTAL PRINTING
      IF (HPRINT.TH^.TP^.PRINFO.RPAGE.RIGHT .GT.
     1    HPRINT.TH^.TP^.PRINFO.RPAGE.BOTTOM) THEN
C       RECALCULATE PRINTRECT BOTTOM AND RIGHT SO THAT
C       PRINTRECT IS SCALED TO FIT HORIZONTAL FORMAT
        PRINTRECT.BOTTOM = PRINTRECT.BOTTOM - 200
        PRINTRECT.RIGHT = PRINTRECT.LEFT +
     1                    INT(FLOAT(PICTURE.PICH^.PICP^.PICFRAME.RIGHT -
     2                              PICTURE.PICH^.PICP^.PICFRAME.LEFT) /
     3                        FLOAT(PICTURE.PICH^.PICP^.PICFRAME.BOTTOM -
     4                              PICTURE.PICH^.PICP^.PICFRAME.TOP) *
     5                        FLOAT(PRINTRECT.BOTTOM - PRINTRECT.TOP))
        CALL OFFSETRECT(%REF(PRINTRECT),INT2(0),INT2(200))     !HORIZ PRINT OFFSET
      ELSE
C       RECALCULATE PRINTRECT BOTTOM AND RIGHT SO THAT
C       PRINTRECT IS SAME SIZE AS PICTURE FRAME
        PRINTRECT.BOTTOM = PRINTRECT.TOP + PICTURE.PICH^.PICP^.PICFRAME.BOTTOM -
     1                     PICTURE.PICH^.PICP^.PICFRAME.TOP
        PRINTRECT.RIGHT  = PRINTRECT.LEFT + PICTURE.PICH^.PICP^.PICFRAME.RIGHT -
     1                     PICTURE.PICH^.PICP^.PICFRAME.LEFT
        CALL OFFSETRECT(%REF(PRINTRECT),INT2(75),INT2(75))     !VERT PRINT OFFSET
      END IF
```

e) *Print the picture*

```
      IF (PRJOBDIALOG(HPRINT)) THEN                !CALLS STANDARD PRINT DIALOG
        CALL GETPORT(%REF(OLDPORT))                !GET THE OLD PORT
        PRINTPORT.TP = PROPENDOC(HPRINT,NIL,NIL)   !OPEN THE PRINTING PORT
        IF (PRERROR .EQ. NOERR) THEN               !IF NO ERROR
          CALL SETPORT(PRINTPORT.TP)               !MAKE PRINT PORT CURRENT PORT
          IF (HPRINT.TH^.TP^.PRJOB.BJDOCLOOP .EQ. BDRAFTLOOP) THEN
            NCOPIES = HPRINT.TH^.TP^.PRJOB.ICOPIES
          ELSE
            NCOPIES = 1
          END IF
          DO 100 COPIES = 1,NCOPIES
            CALL PROPENPAGE(PRINTPORT.TP,NIL)
            CALL DRAWPICTURE(PICTURE,PRINTRECT)    !DRAW PICTURE IN PRINT PORT
            CALL PRCLOSEPAGE(PRINTPORT.TP)
100       CONTINUE
        END IF
        CALL PRCLOSEDOC(PRINTPORT.TP)              !CLOSE PRINTING PORT
        IF ((HPRINT.TH^.TP^.PRJOB.BJDOCLOOP .EQ. BSPOOLLOOP) .AND.
     1      (PRERROR .EQ. NOERR)) THEN
          CALL PRPICFILE(HPRINT,NIL,NIL,NIL,%REF(PRSTAT))    !SPOOL DOCUMENT
        END IF
        IF (PRERROR .NE. NOERR) CALL ALERTBOX('Print error.')
        CALL SETPORT(OLDPORT)                      !RETURN TO OLD PORT
      END IF
```

f) Close the print manager

```
CALL PRCLOSE                                       !CLOSE PRINT MANAGER
```

10. RESOURCE FORK OF MACTRANS

The resource fork of MacTRANS was added using ResEdit Version 2.0b2, which displays the various resources in their full graphical form. The resources were first saved into a file named "TRANS.rsrc" and then this file was copied into the resource fork of MacTRANS at the compilation stage (see Appendix E of Ref. 4).

10.1 Icons

The following steps were taken to add the icons for MacTRANS and the PICT files created in MacTRANS (see Figure 1).

APPL icon PICT icon

Fig. 1 Application and PICT file icons

a) Create New Resource was selected from the Resource menu, and BNDL was selected from the list brought up on the screen. This brought up two more windows, one of which was titled BNDL ID.

b) "TRAN" was typed for the Signature (the Signature being the same as the Creator given to MacTRANS when it was complied) and Create New File Type was selected from the Resource menu twice. This brought up two sets of blank icons.

c) Under Type in the BNDL ID window, "APPL" and "PICT" were entered for the appropriate icons.

d) The blank icons were double-clicked to bring up dialog boxes which prompt for the icons to be used. In both cases New was selected to bring up a window in which the icon was drawn, and these icons were then dragged into the Mask boxes to provide the highlight masks.

10.2 Menus

The Apple and File menus (see Figure 2) were created with the following steps.

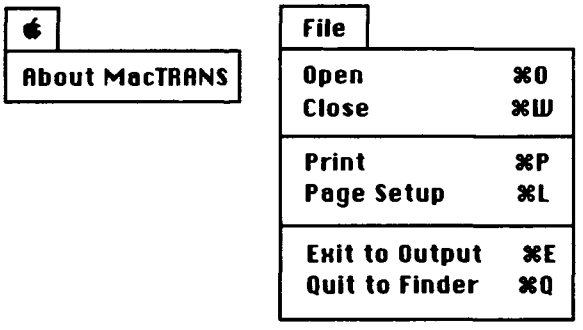

Fig. 2 Apple and File Menus

a) Create New Resource was selected from the Resource menu, and MENU was selected from the list brought up on the screen. This brought up a new MENU ID window.

b) The menu items (with optional command key equivalents) and separator lines were then entered.

10.3 Dialog Box

The dialog box brought up when the About MacTRANS item in the Apple menu is selected (see Figure 3) was created using the following steps.

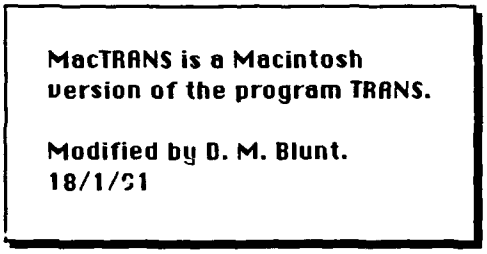

Fig. 3 "About MacTRANS" Dialog Box

a) Create New Resource was selected from the Resource menu, and DLOG was selected from the list brought up on the screen. This brought up a DLOG ID window in which the size and position of the dialog box was set.

b) Create New Resource was selected from the Resource menu, and DITL was selected from the list brought up on the screen. This brought up a DITL ID window.

c) Create New Dialog Item was selected from the resource menu.

d) The text for the dialog box was then typed in and the Static Text button was selected.

11. HOW TO RUN MACTRANS

MacTRANS is launched by double clicking on its icon. This starts the program and bring up the output window. The first prompt is:

```
GO STRAIGHT TO EVENT LOOP?
```

Answer No (a <CR> defaults to No) to continue on with the normal TRANS commands, or Yes to go to the event-loop part of the program where existing PICT files can be opened and printed.

11.1 Data File

The data file is entered in the normal way. Due to the extra extension involved with the PICT files, however, it can only be up to nine characters long (excluding the .DAT extension). Also note that the data file must be in Macintosh binary format (i.e. created on a Macintosh or converted to Macintosh binary format), which is different to the binary format on the Elxsi.

11.2 Commands

The existing TRANS commands (see Appendix B) have not been altered and work exactly the same as before, with the following exceptions:

i. The **SPAce** command no longer needs to be called to remove the space from the left hand side of the plot as this is now the default. The space can still be inserted if required.

ii. The prompt for graphics output to the screen can be ignored as graphics will be sent to the screen no matter what is answered.

iii. When graphics are sent to the screen, the program will wait for a mouse click to allow the user to inspect the result.

iv. The **EXIt** command transfers control of the program to the event-loop, where the plots can be printed, rather than exiting the program.

Note: Graphics output seen on the screen is reduced in size to fit the whole plot on the screen. Text and lines may therefore look somewhat distorted. Also, if graphics fail to appear on the screen, MacTRANS may have run out of memory. This can be overcome by increasing the memory allotted to MacTRANS under Multifinder.

11.3 Colour

Each individual line-type, of which there are eight, is assigned its own colour which can be changed at the time that the line-type is selected. The line colours can be changed by answering Yes (a <CR> defaults to No) to the prompt

```
DO YOU WISH TO CHANGE THE COLOURS? :
```

and entering the line number, a comma, and the new colour (spaces are ignored) after the prompt

```
LINE  COLOUR
```

e.g.

```
2,GREEN
```

A carriage return entered by itself will continue execution of the program when all the changes have been made.

*Note: When the **PLO** or **REP** commands are used with the default (0) line-type specified, the first block will be plotted with line type 1, the next with line-type 2, etc. The colours of these lines will be as set previously. When a line type of 1 is specified, all the blocks will be plotted with solid lines but with the colours still incrementing (as if 0 had been specified). Specifying line-types greater than 1 will plot all the blocks with the same line-type and colour.*

11.4 The Event-loop

The **EXIt** command moves control of the program from the output window to the event-loop. This brings all pictures created from the current session of MacTRANS to the front in the reverse order to which they were created, i.e. with the last picture on top. These pictures can be manipulated in the usual Macintosh fashion; however, they cannot be edited. If editing is necessary, it is easily accomplished in one of the commercial graphics programs such as Canvas or MacDraw II.

Printing a picture is accomplished by selecting the window in which it is displayed (bringing it to the front) and selecting Print from the File menu. This will print the picture

at 100% real size with the default vertical (portrait) page set-up. A vertical page set-up is only suitable for plots with a short time axis and no legend, i.e. those created with the **PLS** command. The horizontal (landscape) page set-up is more suitable for plots with long time axes, or plots where a legend is printed, but scales the plot down to approximately 70% real size to achieve this. Table 5 sets out the maximum axes sizes for printing on an A4 sheet of paper using the LaserWriter.

TABLE 5
Maximum Plot Sizes in Inches

Command	X	Y
PLS (vertical)	5	8
PLS (horizontal)	14	8
PLO / REP (horizontal)	11	8

Selecting the Quit to Finder item from the File menu will quit the program completely and return control to the Finder. The Exit to Output item in the File menu, however, returns control of the program back to the output window.

The output window has its own set of pull down menus (created by the Language System Fortran compiler) with commands that will save or print the contents of the window, and also a command to rerun the program.

11.5 MacTRANS Example

The following example uses a data file created by the performance estimation program Polar2 [4], estimating the performance of an Iroquois helicopter.

```
[TRANS version date 12-FEB-91]

GO STRAIGHT TO EVENT LOOP?
I/P FILENAME = POLAR2
Polar2
Iroquois Performance

I/P FILE RECORDED ON 14-JAN-91   AT 14:04:38

INTEGN INT = 0.0000E+00; RUN CPU TIME =   0.93 SEC.

TIME FROM 0.0000E+00 TO 1.0000E+02 IN STEPS OF 2.0000E+00

*PLO
IS GRAPHICS OUTPUT TO SCREEN REQUIRED :
[PLS/O Output, for this run, going to DSK:POLAR2.PL2      ]

OVERLAY PLOTS :

BLKS
1,2,3,4,34
```

---

4 "Polar2" uses velocity as the independent variable, thus references to time are actually references to velocity in knots.

```
BLKS TO DETERMINE Y SCALE
1

TO SPECIFY NO. OF X UNITS/INCH, TYPE 0 FOR X

LENGTH OF AXES IN INCHES; X, Y = 5,5
ARE SYMBOLS REQRD FOR PLOTS :
LINE KEY (0 GIVES DEFAULT) =
THE CURRENT LINE COLOURS ARE:
0 & 1    2        3        4
BLACK    RED      GREEN    BLUE
5        6        7        8
CYAN     MAGENTA  YELLOW   ORANGE
AVAILABLE COLOURS :
BLACK, RED, GREEN, LTGREEN, BLUE, LTBLUE, CYAN,
MAGENTA, PURPLE, YELLOW, ORANGE.
DO YOU WISH TO CHANGE THE COLOURS? :
*GOE
** RUNNING **
*EXI

STOP
```

Figure 4 shows the graphics from this example as they would appear on the screen, and Figure 5 shows the graphics as they would appear when printed on the LaserWriter when a horizontal page set-up has been selected (this is necessary to include the legend which would otherwise be clipped off).

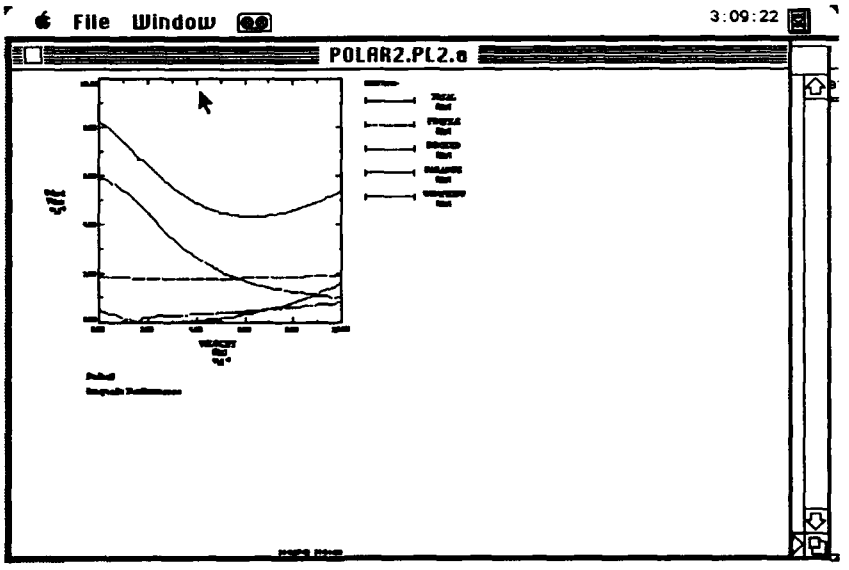

Fig. 4 MacTRANS Graphics Window for the Polar2 Example

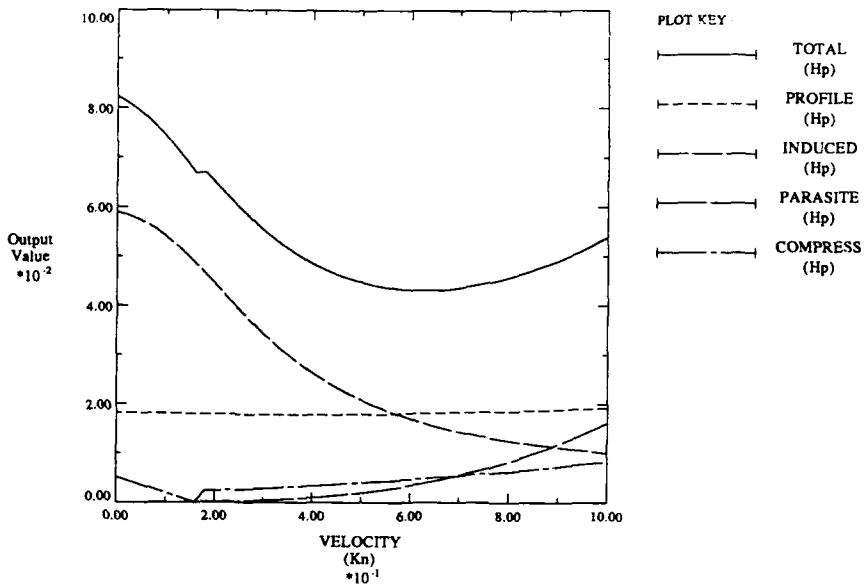

Fig. 5 Graphics for the Polar2 Example as Printed on the LaserWriter

11.6 Execution Time Comparison

To compare the execution times of MacTRANS with TRANS, the same data file was processed identically by both programs. TRANS was run on the Elxsi at four different loading levels, for which the number of time-sharing users provides a guide, and MacTRANS was run once on a Macintosh IIfx.

The data file was a recording of 34 channels, two of which were sampled at 40 Hz and the rest at 10 Hz, for a duration of 134.85 seconds. The processing consisted of using the PLS command to plot all the non-zero channels, a total of 29, on the screen in four separate plots, three with eight channels, and one with five channels. The results are set out in Table 6.

It can be seen that the Macintosh IIfx was faster than the Elxsi for all cases. As it is normal for TRANS to be run on the Elxsi when there are at least 21 time-sharing users, MacTRANS running on the Macintosh IIfx provides significantly improved performance.

TABLE 6
TRANS and MacTRANS Execution Time Comparison

Task	TRANS (Elxsi)		MacTRANS (Mac IIfx)
	Number of Time-sharing Users	Time (min:sec)	Time (min:sec)
Real time taken for initial read of data file	21 29 31 35	0:25 0:30 0:25 1:20	0:38
Real time taken for a PLS plot of all variables (4 pages of plots)†	21 29 31 35	3:30 3:50 4:50 8:00	3:05
Total real time	21 29 31 35	3:55 4:00 5:15 9:20	3:43
Total CPU time	21 29 31 35	1:42 1:42 1:46 1:47	n.a.

† TRANS and MacTRANS pause between pages so that the user can view the results. This is the time taken with no pauses.

12. CONCLUDING REMARKS

The conversion of TRANS to MacTRANS was undertaken using a strategy which required as few changes as were necessary to make the program operational under the Macintosh environment. While this was successful, MacTRANS does not take full advantage of all the Macintosh operating system features, and therefore there is scope for improvement in this area.

The first step in implementing these improvements would be to make the event-loop encompass the entire operation of the program, which would include making all the keyboard entered commands associated with TRANS (PLS, PLO, SCA, etc.) menu items. Other improvements that could be made include using a dialog box to select the data file, making MacTRANS launch and display a PICT file simply by double-clicking on that file, allowing the pictures to be edited in MacTRANS itself, and adding a zoom-in/zoom-out facility to the graphics windows.

It has been shown that MacTRANS, running on a Macintosh IIfx, offers substantial gains in operational speed over TRANS, running on the Elxsi, for the majority of cases. MacTRANS also has advantages in the graphics manipulation area, with the ability to edit plots produced by it in graphics programs such as Canvas or MacDraw II.

REFERENCES

1. Nankivell, P.G. and Gilbert, N.E., "A General Purpose Output Program for Use in Simulation," *ARL Aero Note 367*, December 1976.
2. Gilbert, N.E. and Nankivell, P.G., "Further Information for Users of the Simulation Language CSMP-10(ARL)," *ARL Aero Note 375*, May 1978.
3. Arney, A.M., "Using the General Purpose Output Program 'TRANS'," *ARL Group Report ABS-RW 88/4 (unpublished)*, July 1988.
4. *Language Systems Fortran 2.0 Reference Manual*, Language Systems Corporation, Herndon, VA, January 1990.
5. Blunt D.M., "MacTRANS: The Macintosh Conversion of TRANS," *ARL Group Report RW 91/3 (unpublished)*, June 1991.
6. *Inside Macintosh*, Addison-Wesley, Reading, MA, 1985 - 1988.
7. *DI-3000 User's Guide*, Precision Visuals, Inc., Boulder, Colorado, September 1984.

APPENDIX A

Compiler Switches, Include Files, and Procedures Necessary for Calling the Macintosh Toolbox

This appendix briefly details the requirements for calling the Macintosh Toolbox procedures. For more detailed information, see Ref. 4.

A.1 Compiler Switches and Include Files

The Language Systems Fortran compiler needs to know the trap addresses of the Toolbox calls, the number of arguments each requires, and what type of result each returns. This information is contained in the "Inlines.f" file which is accessed by adding the compiler switch

```
!!MP Inlines.f
```

to the start of any Fortran source code file using Toolbox calls.

Toolbox calls also use various Pascal records as arguments. As Language Systems Fortran cannot use these records, it uses its own records which are defined in structures in "FInclude" files. In order to declare variables using these structures, the appropriate "FInclude" files must be included in each subroutine using Toolbox calls, or a global include compiler switch must be used at the start of each file containing those subroutines. For example, a file containing a subroutine using calls to the window manager could include the file "Windows.f" in the subroutine itself, or use a compiler switch at the start of the file. The compiler switch is more useful, as it negates the need to place include statements throughout the program, and one switch can access as many of the "FInclude" files as necessary. Language Systems Fortran provides a file named "Toolbox.finc" which, when used in the switch

```
!!G Toolbox.finc
```

gives access to the most frequently used "FInclude" files.

"Toolbox.finc" is a file created from a dummy Fortran program that includes the most frequently used "FInclude" files. However, it does not contain the "FInclude" files "Fonts.f" or "Printing.f" needed for the calls to the font manager and print manager in the DI-3000 JFONT routine emulation and printing subroutines. Also, the "FInclude" file "QuickDraw.f", which is included in the "Toolbox.finc" file, does not have the structure definition for "CWindows" needed for the Toolbox calls dealing with these windows, nor the definition for the "ColorArray" structure used in the implementation of colour in MacTRANS.

To remove these problems, an extended dummy Fortran program, "ExToolbox", was created by adding the extra "FInclude" files and supplementary declarations to the dummy Fortran program used to create the "Toolbox.finc" file, and then executing the "BuildFTNGlobals" command to create an "ExToolbox.finc" file from it. The "ExToolbox" dummy program is shown below. Note, however, that the "BuildFTNGlobals" command did not work on "ExToolbox" until the EXTERNAL statements in the "SystemSubs.f" file were removed. The statements removed were identical to those in the "Globals.f" file and this suggests that they were not needed in the "SystemSubs.f" file. The "MPW:Interfaces:FIncludes:Read Me" file has further information on the "Globals.f" file.

```
      PROGRAM EXTOOLBOX
!!SETC USINGINCLUDES = .FALSE.
!!I WINDOWS.F
!!I MENUS.F
!!I DIALOGS.F
!!I ERRORS.F
!!I EVENTS.F
!!I PACKAGES.F
!!I QUICKDRAW.F
!!I TEXTEDIT.F
!!I FONTS.F
!!I PRINTING.F

      INTEGER*4 NIL
      PARAMETER(NIL = 0)

      STRUCTURE/COLORARRAY/
        CHARACTER*9 TEXT
        RECORD/RGBCOLOR/ RGB
      END STRUCTURE

      STRUCTURE/CWINDOWPTR/
          POINTER/CGRAFPORT/ WP
      END STRUCTURE

      END
```

A.3 Calling Procedures

The Macintosh Toolbox calls are Pascal functions and procedures, so the same procedure is adopted as for a Pascal subprogram called from Language Systems Fortran. The most important features of this procedure are:

a) It is necessary to ensure that variables passed in the arguments are of the proper data type. For example, a Toolbox call may expect to see an standard Pascal integer which is two bytes long, whereas the standard Fortran integer is four bytes, and should be converted by using the intrinsic function INT2().

b) Language Systems Fortran can pass arguments as values (`%Val()`) or by reference (`%Ref()`). When the "!!MP Inlines.f" compiler switch is used, Language Systems Fortran defaults to sending arguments to Pascal subprograms as values. In this case, `%Val()` is optional and most arguments can be passed as they would be in the call to a Fortran subroutine, but `%Ref()` must be used when an argument is VARed. Also, when an argument is a pointer to a variable, `%Loc()` can be used to give the memory location of the variable.

APPENDIX B

TRANS Commands

Table B.1 gives a summary of the TRANS commands. Care should be taken when using lower case responses to TRANS prompts, as parts of TRANS may not have been modified to accept them.

TABLE B1
Summary of TRANS Commands

Command	Description
BARs	Specifies bars on DI-3000 plot curves
EXIt	Exits program
GOE	Processes the input file to obtain output for the commands PRColumn, PRPlot, PLStrip and PLOverlay
LABel	Enables modification and addition to labelling information
LISt	Enables a list of blocks to be made or modified
PLOverlay	Specifies "overlay" plotting
PLStrip	Specifies "strip" plotting
PRColumn	Specifies tabular output in the form of printed columns
PRKonstant	Specifies printout of all constant blocks
PRPlot	Specifies graphical output on a line printer or monitor
REPeat	Repeats all previous commands for another input data file
RUN	Equivalent to the commands (in order) PRColumn, PRPlot, PLStrip and GOE; if any form of output is not required, a carriage return is typed in place of block numbers
SCAle	Enables the user to specify scale limits for the graphical output
SPAce	Enables the space at the left side of a plot to be removed
TIMe	Redefines time (independent variable) parameters
XBLock	Enables the user to specify any output block to be used as the independent variable

DISTRIBUTION

AUSTRALIA

Department of Defence

Defence Central

Chief Defence Scientist)
AS, Science Corporate Management)shared copy
FAS Science Policy)
Director, Departmental Publications
Counsellor, Defence Science, London (Doc Data Sheet Only)
Counsellor, Defence Science, Washington (Doc Data Sheet Only)
Scientific Adviser, Defence Central
OIC TRS, Defence Central Library
Document Exchange Centre, DSTIC (8 copies)
Defence Intelligence Organisation
Librarian H Block, Victoria Barracks, Melb (Doc Data Sheet Only)

Aeronautical Research Laboratory

Director
Library
Chief - Flight Mechanics and Propulsion Division
Head - Flight Mechanics Branch
Branch File - Flight Mechanics
Head - Rotary Wing Group (7 copies)
Author: D.M. Blunt (2 copies)
M.I. Cooper
N.T. Goldsmith

Navy Office

Navy Scientific Adviser (3 copies Doc Data Sheet)

Army Office

Scientific Adviser - Army (Doc Data Sheet Only)

Air Force Office

Air Force Scientific Adviser (Doc Data Sheet Only)

SPARES (10 COPIES)

TOTAL (39 COPIES)

AL 149 DEPARTMENT OF DEFENCE

DOCUMENT CONTROL DATA

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING

1a. AR NUMBER	1b. ESTABLISHMENT NUMBER	2. DOCUMENT DATE	3. TASK NUMBER
AR-006-643	ARL-FLIGHT-MECH-TM-446	SEPTEMBER 1991	DST 89/068

4. TITLE
THE CONVERSION OF A FORTRAN DATA PLOTTING PROGRAM USING DI-3000 GRAPHICS TO OPERATION ON A MACINTOSH PERSONAL COMPUTER

5. SECURITY CLASSIFICATION (PLACE APPROPRIATE CLASSIFICATION IN BOX(S) IE. SECRET (S), CONF. (C) RESTRICTED (R), UNCLASSIFIED (U)).

DOCUMENT	TITLE	ABSTRACT
U	U	U

6. NO. PAGES
37

7. NO. REFS.
7

8. AUTHOR(S)
D.M. BLUNT

9. DOWNGRADING/DELIMITING INSTRUCTIONS
Not applicable

10. CORPORATE AUTHOR AND ADDRESS
AERONAUTICAL RESEARCH LABORATORY
506 LORIMER STREET
FISHERMENS BEND VIC 3207

11. OFFICE/POSITION RESPONSIBLE FOR:
SPONSOR DSTO
SECURITY -
DOWNGRADING -
APPROVAL CFPD

12. SECONDARY DISTRIBUTION (OF THIS DOCUMENT)
Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DSTIC, ADMINISTRATIVE SERVICES BRANCH, DEPARTMENT OF DEFENCE, ANZAC PARK WEST OFFICES, ACT 2601

13a. THIS DOCUMENT MAY BE ANNOUNCED IN CATALOGUES AND AWARENESS SERVICES AVAILABLE TO . . . .
No limitations

13b. CITATION FOR OTHER PURPOSES (IE. CASUAL ANNOUNCEMENT) MAY BE [X] UNRESTRICTED OR [ ] AS FOR 13a.

14. DESCRIPTORS
Computer graphics
Apple Machintosh computers

15. DISCAT SUBJECT CATEGORIES
1205
1206

16. ABSTRACT
The planned retirement of the Elxsi 6400 mainframe computer at ARL has necessitated the conversion of the general purpose Fortran data plotting program TRANS to operation on a Macintosh IIfx personal computer, where it has been renamed MacTRANS. The conversion has included the limited use of regular Macintosh features such as pull down menus, and allowed the plots produced by MacTRANS to be edited in most commercial Macintosh graphics programs. For most cases, the execution speed of MacTRANS on the Macintosh IIfx has been found to be faster than that of TRANS on the Elxsi.

PAGE CLASSIFICATION
UNCLASSIFIED
PRIVACY MARKING

THIS PAGE IS TO BE USED TO RECORD INFORMATION WHICH IS REQUIRED BY THE ESTABLISHMENT FOR ITS OWN USE BUT WHICH WILL NOT BE ADDED TO THE DISTIS DATA UNLESS SPECIFICALLY REQUESTED.

16. ABSTRACT (CONT).

17. IMPRINT

AERONAUTICAL RESEARCH LABORATORY, MELBOURNE

18. DOCUMENT SERIES AND NUMBER	19. COST CODE	20. TYPE OF REPORT AND PERIOD COVERED
Flight Mechanics Technical Memorandum 446	51 510I	

21. COMPUTER PROGRAMS USED

22. ESTABLISHMENT FILE REF.(S)

23. ADDITIONAL INFORMATION (AS REQUIRED)