terak
CORPORATION

F4 LOW LEVEL
GRAPHICS RELEASE GUIDE

REV 1

TERAK P/N 60-0050-001

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

terak
CORPORATION

F4 LOW LEVEL
GRAPHICS RELEASE GUIDE

REV 1

TERAK P/N 60-0050-001

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

LOW LEVEL GRAPHICS INTRODUCTION

The LOW LEVEL graphics routines provide primitive FORTRAN IV 8510a graphics programming support. The user of this library is required to deal with the graphics display page as a dot array of 320 horizontal by 240 vertical dots. These routines support turning dots on and off, testing dot status, performing dot masks, and mapping display page characters.

A more sophisticated FORTRAN graphics support is provided with an ACM SIGGRAPH CORE-79 compatible graphics library. This library is described in another document, "F4 SIGGRAPH CORE-79 GRAPHICS Release Guide", TERAK publication number 60-0049-001.

RECOMMEMDED READING

To utilize the LOW LEVEL graphics library, the user should be familiar with the general content of the PDP-11 FORTRAN IV Language Reference Manual, the RT-11/RSTS/E FORTRAN IV User's Guide, and the Video Display and 24K Memory System (TERAK Pub No. 52-0002-001). Particular attention should be given to the information contained in Appendix A, FORTRAN DATA REPRESENTATION of the User's Guide and Section 2.6, ARRAYS, of the Language Reference Manual. This information provides the basis for understanding the FORTRAN implementation.

Other tools such as IPEEK and IPOKE are described in the "RT-11 Advanced Programmer's Guide", chapter 4.

DISPLAY PAGE ALLOCATION

The display space may be a LOGICAL*1 or LOGICAL*2 array, typically two dimensioned as:

```
LOGICAL*1 DISPLA(40,240)
```

Note that the first dimension of 40 times 8 bits per byte is 320, the dot width of the display page. This approach allows use of the FORTRAN array subscript calculation facility to address whole bytes, or 8 dot fields within the display array. These can be passed to subroutines as the base of a SUB-ARRAY to be manipulated by the routines in this low level package.

This approach has two difficulties. First, the array space will be built into the save image file, inflating its disk space. Second, FORTRAN will place arrays in the Root Section of the program. However, the Display Space must be above absolute 20000 (OCT). If the Display Space is below 20000, the actual display will "WRAP-AROUND" to locations below 160000.

To overcome this, the program must be linked such that the display array is loaded above 20000. This may require use of the linker's "/P" switch to adjust the bottom of the program, thus further inflating the disk space required for the program, and possibly wasting memory usage. This is the price paid for a simpler graphics programming technique.

FORTRAN programmers may use IPEEK and IPOKE to control the contents of the VCR (VIDEO CONTROL REGISTER, 177744), and the GAR (GRAPHIC ADDRESS REGISTER, 177740). For example, to completely blank the character display, and completely unblank the graphics display, write Octal 70 into the VCR as:

```
CALL IPOKE("70,"177744)
```

Data may be written into the character display independent of the blanking of the character or graphics zones, with two exceptions. First if a monitor with the GT (glass teletype) emulator is being used, any data written to the character display through the emulator (ITTOUR,IPRINT,WRITE,etc), then the lower character zone will always by unblanked. If this zone is blanked, and no data is written to the character display, it will remain unblanked. If a monitor with the TK (standard) emulator is being used, the character and graphics blanking set by the user program will be unaffected by character writing. This allows a character display to be created, then unblanked for an "instantaneous" effect.

Second, a control-L (Form Feed, Octal 14) character written to any emulator will clear the character display, blank the entire graphics display, and unblank the entire character display. The graphics display buffer and the contents of the GAR will not be affected.

See V3b Release Notes for more information on the Console Emulator.

SPECIAL DISPLAY PAGE CONSIDERATIONS

If the user decides to utilize the "free" memory area between the top of their program and the bottom of the RT-11 monitor as the graphics display page, then the function IGETSP can be used. (See RT-11 Advanced Programmer's Guide, chapter 4, for IGETSP details). The LOW LEVEL routines must be modified to use the variable argument in the calling sequence as an indirect pointer to the graphics display page. The LOW LEVEL graphics library has also been provided in source form to allow the user to make such modifications, as desired.

LOW LEVEL GRAPHICS OVERVIEW.

SETBIT and CLRBIT are used to set and clear individual dots in the display space. Note that the bit offset argument can be any value...its range is not restricted. This allows its working range to be 0 to 319 (DEC) to address the horizontal position of a dot. The routines work on a sub-array, so the second subscript of the integer argument addresses 1 to 240 (DEC) rows of dots in the display space.

The following examples show some methods for turning on the 33'rd dot in the 97'th row. No particular way is recommended above any other. The choice of method depends upon the preference of the user and the application. The following are examples only and do not imply that there are not other methods for accomplishing the same result.

Method # 1

```
LOGICAL*1       DISPLA(40,240)
LOGICAL*2       COL,ROW
LOGICAL*2       A,O
.
.
.
COL = 33
ROW = 97
A = 1 + COL/8
O = MOD(COL,8)
CALL SETBIT (DISPLA(A,ROW),O)
```

Method # 2

```
LOGICAL*1       DISPLA(40,240)
LOGICAL*2       COL,ROW
.
.
.
COL = 33
ROW = 97
CALL SETBIT (DISPLA(1,1), COL*ROW-1)
```

NOTE: This method is effective only if COL*ROW is less than the maximum integer value 65535. Values greater than this limit cause integer arithmetic overflow. Utilizaton of this method should be limited to applications using no more than two zones.

Method # 3

```
LOGICAL*1       DISPLA(40,240),ROW97(40)
LOGICAL*2       COL,ROW
EQUIVALENCE     (ROW97(1),DISPLA(1,97)
.
.
.
COL = 33
ROW = 97
CALL SETBIT (ROW97,COL-1)
```

TSTBIT returns the state of an individual dot in the display space. Its use is similar to SETBIT or CLRBIT, with the addition of a third argument which returns a zero or non zero value.

HIDE draws a vertical erasing line from the addressed dot to the bottom of the display space. This routine is used in the LOW LEVEL graphics demonstration "LOWDEM" to erase hidden lines rapidly. This is much faster than the same function accomplished by calling CLRBIT from within a FORTRAN level loop.

GRFPNT is a subroutine to print a literal string into the Graphic Display Space using 8 by 10 dot blocks for each character. This is done by receiving a sub-array pointer from the mainline code, and mapping an equivalenced array onto the sequence of dot blocks as required. The location argument is passed as: DISPLA(COLUMN, ROW) where column is a number from 1 to 40, addressing one of the 40 multiple of 8 dot columns where characters can be placed by GRFPNT. GRFPNT scans the literal string or logical*1 array, and directs the array pointer sequentially to the right. CAUTION: The R.H. side of the Display Space will not wrap around cleanly, and exceeding the Display space is not checked for.

GRFASC is called by GRFPNT to do the transfer of each character block. The dot pattern is read from the writeable character generator. Thus, if the character set has been modified, GRFPNT/GRFASC will use the modified patterns.

ANDFOR, ORFOR, and XORFOR are simple routines to allow the masking, overlaying, and toggling of dots in separate Display Spaces.

LOW LEVEL GRAPHICS CALLING SEQUENCES

** SETBIT **

Turns on (Sets) a dot at a specified offset from the LSB of variable A.

CALL SEQUENCE...

```
CALL SETBIT(A,O)
```

ARGUMENTS...

A	Destination or source variable should be a logical*1, integer*2 variable or array element.
O	Offset in bits from LSB of A. May be greater than 7 to address a bit array. Range is not checked! Should be a integer*2 variable only. Note that Offset =0 -> LSB of A.

** CLRBIT **

Turns off (clears) a dot at a specified offset from the LSB of variable A.

CALL SEQUENCE...

CALL CLRBIT(A,O)

ARGUMENTS...

A	Destination or source variable should be a logical*1, integer*2 variable or array element.
O	Offset in bits from LSB of A. May be greater than 7 to address a bit array. Range is not checked! Should be a integer*2 variable only. Note that Offset =Ø -> LSB of A.

** TSTBIT **

Determines the status of a dot at a specified offset from the LSB of variable A.

CALL SEQUENCE...

CALL TSTBIT(A,O,L)

ARGUMENTS...

A	Destination or source variable Should be a logical*1, integer*2 variable or array element.
O	Offset in bits from LSB of A. May be greater than 7 to address a bit array. Range is not checked! Should be a integer*2 variable only. Note that Offset =Ø -> LSB of A.
L	Logical*2 variable to receive logic of tested bit.

** HIDE **

Subroutine to draw a vertical erasing line. It clears a dot in variable A offset O and every 320th dot thereafter, N times

CALL SEQUENCE...

```
CALL HIDE(A,O,N)
```

ARGUMENTS...

A	Destination or source variable should be a logical*1, integer*2 variable or array element.
O	Offset in bits from LSB of A. May be greater than 7 to address a bit array. Range is not checked! Should be a integer*2 variable only. Note that Offset =0 -> LSB of A.
N	Count of number of bits to zap must be integer*2.

** AND **

Places the logical "AND" of word at A onto the word at B.

CALL SEQUENCE...

```
CALL AND(A,B)
```

ARGUMENTS...

A,B	Destination or source variable must be a logical*2, integer*2 variable or array element.

** OR **

Places the logical "OR" of the word at A onto the word at B.

CALL SEQUENCE...

```
CALL OR(A,B)
```

ARGUMENTS...

A,B	Destination or source variable must be a logical*2, integer*2 variable or array element.

** XOR **

Places the logical "XOR" of the word at A onto the word at B.

CALL SEQUENCE...

```
CALL XOR(A,B)
```

ARGUMENTS...

A,B	Destination or source variable must be a logical*2, integer*2 variable or array element.

** GRFASC **

Subroutine to extract an ASCII dot pattern from the 8510a writeable character generator, and transfer it into a graphics display.

CALL SEQUENCE...

```
CALL GRFASC(A,B)
```

ARGUMENTS...

A	Address of byte of row in graphics display array to receive top row of-character---the upper left corner of the 8 by 10 zone to receive the dot pattern.
B	Address of byte containing the character code of the pattern required.

** GRFPNT **

Subroutine to map a character string onto the graphics display page. The source character string MUST be terminated with an octal zero (0) character.

CALL SEQUENCE...

```
CALL GRFPNT(A,B)
```

ARGUMENTS...

A	Address of byte of row in graphics display array to receive top row of character string.
B	Source character string terminated with an octal zero character. May be a quoted literal string or a string vector.

LOW LEVEL GRAPHICS LIBRARY

The dot graphics library is supplied to the user on distribution diskette, "TERAK FORTRAN IV GRAPHICS LIBRARY", TERAK part number (61-0009-003). The distribution file is a concatenated object file named "LOWGRF.OBJ". The LOW LEVEL graphics library has also been included in the "READY TO USE" system library provided on the distribution diskette TERAK part number 61-0009-005.

If a different library configuration other than that distributed by TERAK is developed by the user, care must be taken in the order of library linking. Refer to section 6 of the "RT-11 FORTRAN IV V2.1 RELEASE GUIDE", TERAK part number 60-0048-001 for a detailed discussion.

LOW LEVEL GRAPHICS DEMONSTRATION

A demonstrarion program utilizing the LOW LEVEL graphics library is contained on the distribution diskette "FORTRAN IV GRAPHICS DEMOS", TERAK part number 61-0009-006. The program is "LOWDEM" and exists as FORTRAN source and RT-11 executable module. In addition a documentation file "LOWDEM.DOC" exists which contatins the exact RT-11 command sequence used to create the ready to run demonstration. A listing of the program file and documentation file is also included in the next section of this document. These files provide an example of how to utilize the LOW LEVEL graphics library.

The demonstration program draws a sine mountain (Z = SIN(X+Y)/(X+Y) on the graphics display. When the image on the screen has been completed, the program will wait, echoing anything typed. Typing a <return> allows the program to exit back to the RT-11 monitor.

LOW LEVEL DEMONSTRATION LISTINGS

```
      REAL*4 Z(69,69), OLDFUN, LATRL, OLDLAT, FUNC
      INTEGER*2 DSPLY1(20,240)
      INTEGER*2 DSPLY2(20,240)
      DATA PI/3.1415926/
      DATA ALPHA/.3000/     !RIGHT AXIS ANGLE ABOVE LEVEL
      DATA GAMMA/.6900/     !LEFT AXIS ANGLE ABOVE LEVEL
      DATA XGAIN/2.5/
      DATA YGAIN/2.5/
      DATA ZGAIN/1.1/
      DATA XORG/150./
      DATA ZORG/200./
1     CSA = XGAIN*COS(ALPHA)
      CSG = YGAIN*COS(GAMMA)
      SNA = XGAIN*SIN(ALPHA)
      SNG = YGAIN*SIN(GAMMA)
C
C     GRIND OUT FUNCTION BEFORE STARTING DISPLAY
C     CALL ITTOUR("14)          !CLEAR THE SCREEN
      DO 10, IX = 1, 69
      CALL ITTOUR('.')          !TELL HIM WE'RE WORKING
      DO 10, IY = 1, 69
C
      X = (IX-35)*7.*PI/69.     !X RANGES +- 7 PI
C
      Y = (IY-35)*7.*PI/69.     !Y RANGES +- 7 PI

      ZZ = 1.0
      IF (X) 5,6,5
5     ZZ = SIN(X)/X
6     IF (Y) 7,8,7
7     ZZ = ZZ*SIN(Y)/Y
8     Z(IX,IY)=100.*ZZ*ZGAIN
C
10    CONTINUE
15    DO 20 J=1,240
20    DSPLY1(I,J) = 0
      CALL IPOKE("177740,IADDR(DSPLY(1,1)))          !SET DISPLAY POINTER
      CALL IPOKE("177744,"70)                        !TURN ON DISPLAY
C
C
C
      DO 50 IX = 69,1,-2        !BACK TO FRONT
      OLDLAT = 0                !FIRST TIME FLAG
      K = 0
      DO 50 IY = 1,69           !CONTOURS IN Y DIRECTION
C
```

```
30      FUNC = ZORG - (Z(IX,IY)+IX*SNA+IY*SNG)
          IF (FUNC .GT. 240.) FUNC = 240.
          IF (FUNC .LT. 1.) FUNC = 1.
          LATRL = XORG + (IX*CSA) - (IY*CSG)
          IF (LATRL .GT. 320.) LATRL = 320.
          IF (LATRL .LT. 1.) LATRL = 1.
          IF (OLDLAT .NE. 0.) GOTO 31   !THIS IS FOR FIRST TIME
          OLDLAT = LATRL
          OLDFUN = FUNC
C
C       COMPUTE DISTANCE BETWEEN NEW POINT AND OLD
C
31        X    = (LATRL - OLDLAT)**2
          Y = (FUNC - OLDFUN)**2
          Y = SQRT(X+Y)
          IF (Y .EQ. 0.) GOTO 35        !COVERS ERRORS AND FIRST TIME
          Y = (LATRL-OLDLAT)/Y  !NOTE...Y = NEGATIVE INCREMENT
          X = OLDLAT
32        X = X + Y      !DECREASE X
          I = X +.5      !ROUND TO WHOLE CONTOUR STEP
          ZZ = FUNC + (OLDFUN-FUNC)*(X-LATRL)/(OLDLAT-LATRL)
          J = ZZ + .5
          IF (K .EQ. I) GOTO 33
          CALL HIDE(DSPLY1(1,J+1),I,240-J)
33        CALL SETBIT(DSPLY1(1,J),I)
          K = 1
          IF (X .GT. LATRL) GOTO 32
35        OLDLAT = LATRL
          OLDFUN = FUNC
50        CONTINUE
115       DO 120 I=1,20
          DO 120 J=1,240
120       DSPLY2(I,J) = 0
          CALL IPOKE("177740,%ADDR(DSPLY2(1,1)))        !SET DISPLAY POINTER
          CALL IPOKE("177744,"70)                       !TURN ON DISPLAY
C
C
C
          DO 150 IY = 69,1,-2   !BACK TO FRONT
          OLDLAT = 0            !FIRST TIME FLAG
          K = 0
          DO 150 IX = 1,69      !NOW CONTOURS IN X DIRECTION
C
```

```
130     FUNC = ZORG - (Z(IX,IY)+IX*SNA+IY*SNG)
        IF (FUNC .GT. 240.) FUNC = 240.
        IF (FUNC .LT. 1.) FUNC = 1.
        LATRL = XORG + (IX*CSA) - (IY*CSG)
        IF (LATRL .GT. 320.) LATRL = 320.
        IF (LATRL .LT. 1.) LATRL = 1.
        IF (OLDLAT .NE. 0.) GOTO 131  !THIS IS FOR FIRST TIME
        OLDLAT = LATRL
        OLDFUN = FUNC
C
C     COMPUTE DISTANCE BETWEEN NEW POINT AND OLD
C
131     X = (LATRL - OLDLAT)**2
        Y = (FUNC - OLDFUN)**2
        Y = SQRT(X+Y)
        IF (Y .EQ. 0.) GOTO 135        !COVERS ERRORS AND FIRST TIME
        Y = (LATRL-OLDLAT)/Y  !NOTE...Y = POSITIVE INCREMENT
        X = OLDLAT
132     X = X + Y      !INCREASE X
        I = X +.5      !ROUND TO WHOLE CONTOUR STEP
        ZZ = FUNC + (OLDFUN-FUNC)*(X-LATRL)/(OLDLAT-LATRL)
        J = ZZ + .5
        IF (K .EQ. I) GOTO 133
        CALL HIDE(DSPLY2(1,J+1),I,240-J)
133     CALL SETBIT(DSPLY2(1,J),I)
        K = I
        IF (X .LT. LATRL) GOTO 132
135     OLDLAT = LATRL
        OLDFUN = FUNC
15      CONTINUE
200     DO 300 I=1,20
        DO 300 J=1,240
300     CALL OR(DSPLY1(I,J),DSPLY2(I,J))
C       STAY HERE UNTIL USER STRIKES ANY KEY THEN EXIT TO MONITOR
        CALL IPOKE("177744,"77)         !UNBLANK CHAR DISPLA AND GRAR DIAPLA
310     CALL GETSTR(5,I,1,K)
C       TURN OFF THE GRAPHICS SPACE AND TURN ON THE CHARACTER SPACE
        CALL IPOKE("177744,"07)
        END
```

```
!*****************************************************************************
  ********************    NOTES DESCRIBING THE CREATION    **********************
!*********************  PROCEDURE USED FOR "READY TO RUN"  **********************
!*********************       LOW LEVEL GRAPHICS DEMO       **********************
!*****************************************************************************
!
!LOWDEM.DOC
!CREATED 9-28-79          UPDATED 9-28-79
!
!THE PURPOSE OF THE FILE IS TO DOCUMENT THE EXACT RT-11 V3B COMMAND
!     SEQUENCE USED TO CREATE THE LOW LEVEL GRAPHICS DEMO CONTAINED
!     ON THE DISTRIBUTION DISKETTE, TERAK PART NUMBER 61-0009-006.
!
!THIS COMMAND SEQUENCE MAKES USE OF THE "READY TO USE" FORTRAN COMPILER ON
!     DISTRIBUTION DISKETTE, TERAK PART NUMBER 61-0009-004
!THIS COMMAND SEQUENCE MAKES USE OF THE "READY TO USE" SYSTEM LIBRARY ON
!     DISTRIBUTION DISKETTE, TERAK PART NUMBER 61-0009-005
!
!-----------------------------------------------------------------------------
!
!** STEP #1 **
!PLACE THE "FORTRAN IV GRAPHICS DEMO" DISKETTE,
!     TERAK PART NUMBER 61-0009-006,
!     INTO QX1:
!
!-----------------------------------------------------------------------------
!
!** STEP #2 **
!PLACE THE "READY TO USE" FORTRAN IV COMPILER DISKETTE,
!     TERAK PART NUMBER 61-0009-004,
!     INTO QX0:
!
!-----------------------------------------------------------------------------
!
!** STEP #3 **
!BOOT THE COMPILER DISKETTE, QX0:.
!
!-----------------------------------------------------------------------------
!
!** STEP #4 **
!COMPILE THE FORTRAN SOURCE FOR THE LOW LEVEL GRAPHICS DEMO
R FORTRA
QX1:LOWDEM,QX1:LOWDEM=QX1:LOWDEM
!
!-----------------------------------------------------------------------------
!
!** STEP #5 **
!REMOVE THE COMPILER DISKETTE IN QX0:
!
!-----------------------------------------------------------------------------
!
!** STEP #6 **
!PLACE THE "READY TO USE" SYSTEM LIBRARY DISKETTE,
!     TERAK PART NUMBER 61-0009-005,
!     INTO QX0:
!
!-----------------------------------------------------------------------------
```

```
!
!** STEP #7 **
!BOOT THE SYSTEM LIBRARY DISKETTE, QX0:
!
!-------------------------------------------------------------------------------
!
!** STEP #8 **
!LINK THE COMPILED FORTRAN OBJECT WITH THE SYSTEM LIBRARY
R LINK
QX1:LOWDEM,QX1:LOWDEM=QX1:LOWDEM
!
!-------------------------------------------------------------------------------
!
!THE COMPILATION AND LINKING OF THE LOW LEVEL GRAPHICS DEMO IS COMPLETE
!     THE FOLLOWING FILES SHOULD NOW EXIST ON QX1:
!          (1)"LOWDEM.LST" , COMPILER LISTING
!          (2)"LOWDEM.OBJ" , COMPILER OBJECT
!          (3)"LOWDEM.MAP" , LINKER MAP
!          (4)"LOWDEM.SAV" , RT-11 EXECUTABLE MODULE
!
!THE DEMONSTRATION CAN BE EXECUTED BY THE FOLLOWING RT-11 COMMAND SEQUENCE
SET USR SWAP
RUN QX1:LOWDEM
!
!*******************************************************************************
!**********************    NOTES DESCRIBING THE CREATION    **********************
!**********************  PROCEDURE USED FOR "READY TO USE"  **********************
!**********************       LOW LEVEL GRAPHICS DEMO       **********************
!*******************************************************************************
```