VSV21 PROGRAMMER'S GUIDE ADDENDUM

AD-FV67B-T1

GETTING STARTED

A guide to getting started with VIVID is contained in the Programmer's Guide, Chapter 19.

CHAPTER 5 THE VIVID SUBROUTINE LIBRARY

INTRODUCTION

The display area is a 2Mb global section in MicroVMS. Access to the section is by array element. Each element is a one-word integer in the array.

NB. Under VMS, the user must use the switches /NOI4 and /NOOP when compiling application programs. This is because the parameters passed in VSL calls should be INTEGER*2 unless otherwise stated. Under VMS, all VSL functions return an INTEGER*4 value. ISTAT will work as either I*2 or I*4 except for VVRREP, as specified later in this addendum.

The subroutine library can be accessed from high level languages other than FORTRAN, such as 'C', PASCAL, BASIC etc. The important stipulation to remember, when writing such applications, is that all calls to the VSL functions must pass their parameters by reference rather than by value. This is a standard interface within FORTRAN programs. Again, it is advised to use the 'NO OPTIMISE' switch when compiling.

5.1 GENERAL FUNCTIONS

5.1.1 Initialize Display Processing - VVXINI

Under VMS, the first parameter DLEN should be defined as an INTEGER*4.

5.1.3 Assign VSV21 Device - VVXASS

The first parameter DEV should be passed as the device physical unit number,

```
eg. VMS     0 for VSA0:
            1 for VSB0:
            2 for VSC0:   etc.

    RSX     0 for VS0:
            1 for VS1:
            2 for VS2:    etc.
```

5.4.2 Get Report - VVRREP

Under VMS, the return status value (eg.ISTAT) MUST be defined as an INTEGER*4.

5,6.6 Text Functions

VVTDRP - DRAW_PACKED_CHARS (*)

There is an inherent difference in the writing of VSL applications in FORTRAN on VMS and RSX systems. It is to do with the way the operating systems handle strings. On RSX systems, a string buffer can be passed in a FORTRAN call simply by the name of the buffer containing the string, or by the defined string itself, and the address is used directly by the called function.

On VMS systems, the text string is passed by String Descriptor Block.

On RSX, an example of a call to this function would be:

```
CALL  VVTDRP ('A string',4)
```

that is, a string plus the number of WORDS used in the string.

On VMS, an example of a call to this function would be:

```
CALL  VVTDRP (%REF('A string'),4)
```

that is, the address of a string plus the number of words.

It will be seen that, under VMS, the use of the %REF qualifier ensures that the string address is passed to the VVTDRP function.

VSL EXAMPLES

An example of using VSL to produce a simple display is provided. This should especially aid new VSV21 users to access the VIVID instruction set by the use of the higher level calls within VSL. The program is written in FORTRAN 77.

Examples are provided for both RSX and VMS systems.

Program name for RSX: VSLRSX.FTN

Program name for VMS: VSLVMS.FOR

```
        PROGRAM VSLVMS

C A simple example to illustrate VSL and its ease of use.
C Program sets up VSV21 processing and creates a segment to output
C a header message and sequence of colored rectangles.
C Full error handling is included.
C NB.This example runs on a MicroVAX II under MicroVMS.
C
C To compile and link the program :-
C
C       FOR/NOOP/NOI4 program name
C       LINK program name,SYS$LIBRARY:VSLLIB/LIB
C
C
        IMPLICIT INTEGER*4 (V)

        INTEGER*4       DRAWPIC
        INTEGER*2       ISTAT,LUN


C Initialize VIVID processing.
C Set up the display area to be 4096 bytes long,
C maximum segments to be 10.

        ISTAT = VVXINI (4096, 10)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVXINI  ', ISTAT)  ! Check status


C Assign VSV21 device for VSL processing.
C First, set up a logical unit number for all subsequent device access.

        LUN = 1

C Device physical unit 0 will assign to device VSA0:
C                      1 will assign to device VSB0:
C                      2 will assign to device VSC0:  etc.
C Third parameter 1024 sets up a report segment of that size,
C default segment ID Hex 2001.

        ISTAT = VVXASS (0, LUN, 1024)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVXASS  ', ISTAT)  ! Check status


C Create a VIVID instruction segment of length 1000 bytes

        ISTAT = VVMCRS (LUN, '201'X, 1000)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVMCRS1 ', ISTAT)  ! Check status

C Call subroutine to build up the display segment and output the picture
C Supply logical unit number and segment ID.
```

```
      ISTAT = DRAWPIC (LUN, '201'X)
      IF (ISTAT .NE. 1)  CALL SRERR ('DRAWPIC ', ISTAT)  ! Check status

C Release VSV21 device from VSL processing.
      ISTAT = VVXREL (LUN)
      IF (ISTAT .NE. 1)  CALL SRERR ('VVXREL  ', ISTAT)  ! Check status


C Release VSV21 processor and free the VSV21 buffers.
      ISTAT = VVXEND ()
      IF (ISTAT .NE. 1)  CALL SRERR ('VVXREL  ', ISTAT)  ! Check status
      STOP
      END

C Set up title "THIS IS VSV21" and draw colored rectangles
      INTEGER*4 FUNCTION  DRAWPIC (LUN, SEGID)
      INTEGER*2       LUN,SEGID,ISTAT,RECX,RECY,I
      CALL  VVBBGN (SEGID)              ! start of segment
      CALL  VVCINI (-1)                 ! initialise all
      CALL  VVACLS ()                   ! clear screen
      CALL  VVGMOD (1,0)                ! drawing mode - foreground/background
      CALL  VVGFCL (14)                 ! foreground color yellow
      CALL  VVDMOV (36,420)             ! move abs
      CALL  VVTMAG (1,3,3)              ! cell mag 3x
      CALL  VVBPMD (1)                  ! set param mode for arrays
      CALL  VVTDRP (%REF('T H I S  I S  V S V 2 1 '),12)
      CALL  VVBPMD (0)                  ! reset param mode
      CALL  VVCINI (-1)                 ! initialise all
      CALL  VVDMOV (176,128)            ! move abs
      RECX = 448                        ! init rectangle x-val
      RECY = 368                        ! init rectangle y-val

C 2 loops to produce 30 colored rectangles
      DO 100 I=1,15,1
          CALL  VVGFCL (I)              ! foreground colors 1-15
          CALL  VVBMOD (1)              ! instruction mode - relative
          CALL  VVDMOV (4,4)            ! move rel
          CALL  VVBMOD (0)              ! instruction mode - absolute
          CALL  VVDREC (RECX,RECY)      ! rectangle absolute
          RECX = RECX - 4               ! rectangles in by 4
          RECY = RECY - 4
```

```
100     CONTINUE

        DO 200 I=1,15,1
            CALL  VVGFCL (I)              ! foreground colors 1-15
            CALL  VVBMOD (1)              ! instruction mode - relative
            CALL  VVDMOV (4,4)            ! move rel
            CALL  VVBMOD (0)              ! instruction mode - absolute
            CALL  VVDREC (RECX,RECY)      ! rectangle absolute
            RECX = RECX - 4               ! rectangles in by 4
            RECY = RECY - 4
200     CONTINUE

        CALL  VVBEND ()                   ! end of segment

C Now execute the segment

        CALL  VVEEXE (LUN,SEGID,32000)  ! execute segment
        CALL  VVRSTA (LUN, DRAWPIC, IDUM) ! get execute status

        RETURN
        END

C Error handling.
C Routine prints function and error code.


        SUBROUTINE  SRERR (ISRNAM, ISTAT)       ! Handle errors

        CHARACTER*8         ISRNAM
        INTEGER*2            ISTAT

1000    FORMAT  (/, ' **** Failed: routine ', a8,
        1         '   Status = ', I7, ' ****', /)

        WRITE (5, 1000) ISRNAM, ISTAT
        STOP
        END


        PROGRAM VSLRSX

C A simple example to illustrate VSL and its ease of use.
C Program sets up VSV21 processing and creates a segment to output
C a header message and sequence of colored rectangles.
C Full error handling is included.
C NB.This example runs on a Micro PDP-11 or PDP-11/73 under RSX.
C
C To compile and build the program :-
C
C       F77 vslrsx=vslrsx
C       TKB vslrsx,vslrsx/-sp=vslrsx
C       TKB lb:[1,1]vivlib/lb
C       TKB lb:[1,1]f4pots/lb
C       TKB /
C       TKB vsect=vv21da:160000:20000
```

```
C       TKB wndws=1
C       TKB maxbuf=512
C       TKB //
C
C       It is advised to store the VSL library VIVLIB.OLB in
C       system UIC [1,1] as specified in the TKB example.
C
C
        IMPLICIT INTEGER*2 (V)

        INTEGER*2       ISTAT,LUN

C Initialize VIVID processing.
C Set up the display area to be 4096 bytes long,
C maximum segments to be 10.

        ISTAT = VVXINI (4096, 10)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVXINI  ', ISTAT)  ! Check status


C Assign VSV21 device for VSL processing.
C First, set up a logical unit number for all subsequent device access.

        LUN = 1

C Device physical unit 0 will assign to device VS0:
C                      1 will assign to device VS1:
C                      2 will assign to device VS2:  etc.
C Third parameter 1024 sets up a report segment of that size,
C default segment ID Hex 2001.

        ISTAT = VVXASS (0, LUN, 1024)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVXASS  ', ISTAT)  ! Check status


C Create a VIVID instruction segment of length 1000 bytes

        ISTAT = VVMCRS (LUN, '201'X, 1000)

        IF (ISTAT .NE. 1)  CALL SRERR ('VVMCRS1 ', ISTAT)  ! Check status

C Call subroutine to build up the display segment and output the picture
C Supply logical unit number and segment ID.

        ISTAT = IDRAW (LUN, '201'X)

        IF (ISTAT .NE. 1)  CALL SRERR ('IDRAW  ', ISTAT)  ! Check status


C Release VSV21 device from VSL processing.

        ISTAT = VVXREL (LUN)
```

```
        IF (ISTAT .NE. 1)  CALL SRERR ('VVXREL  ', ISTAT)  ! Check status


C Release VSV21 processor and free the VSV21 buffers.

        ISTAT = VVXEND ()

        IF (ISTAT .NE. 1)  CALL SRERR ('VVXREL  ', ISTAT)  ! Check status

        STOP

        END


C Set up title "THIS IS VSV21" and draw colored rectangles

        FUNCTION  IDRAW (LUN, SEGID)

        IMPLICIT INTEGER*2 (V)

        INTEGER*2       LUN,SEGID,ISTAT,RECX,RECY,I

        CALL  VVBBGN (SEGID)            ! start of segment
        CALL  VVCINI (-1)               ! initialise all
        CALL  VVACLS ()                 ! clear screen
        CALL  VVGMOD (1,0)              ! drawing mode - foreground/background
        CALL  VVGFCL (14)               ! foreground color yellow
        CALL  VVDMOV (36,420)           ! move abs
        CALL  VVTMAG (1,3,3)            ! cell mag 3x
        CALL  VVBPMD (1)                ! set param mode for arrays
        CALL  VVTDRP ('T H I S  I S  V S V 2 1 ',12)
        CALL  VVBPMD (0)                ! reset param mode
        CALL  VVCINI (-1)               ! initialise all
        CALL  VVDMOV (176,128)          ! move abs
        RECX = 448                      ! init rectangle x-val
        RECY = 368                      ! init rectangle y-val

C 2 loops to produce 30 colored rectangles

        DO 100 I=1,15,1
            CALL  VVGFCL (I)            ! foreground colors 1-15
            CALL  VVBMOD (1)            ! instruction mode - relative
            CALL  VVDMOV (4,4)          ! move rel
            CALL  VVBMOD (0)            ! instruction mode - absolute
            CALL  VVDREC (RECX,RECY)    ! rectangle absolute
            RECX = RECX - 4             ! rectangles in by 4
            RECY = RECY - 4
100     CONTINUE

        DO 200 I=1,15,1
            CALL  VVGFCL (I)            ! foreground colors 1-15
            CALL  VVBMOD (1)            ! instruction mode - relative
            CALL  VVDMOV (4,4)          ! move rel
            CALL  VVBMOD (0)            ! instruction mode - absolute
```

```
            CALL  VVDREC (RECX,RECY)     ! rectangle absolute
            RECX = RECX - 4              ! rectangles in by 4
            RECY = RECY - 4
200         CONTINUE

            CALL  VVBEND ()              ! end of segment

C Now execute the segment

            CALL  VVEEXE (LUN,SEGID,32000)  ! execute segment
            CALL  VVRSTA (LUN, IDRAW, IDUM) ! get execute status

            RETURN
            END

C Error handling.
C Routine prints function and error code.


            SUBROUTINE  SRERR (ISRNAM, ISTAT)       ! Handle errors

            CHARACTER*8         ISRNAM
            INTEGER*2            ISTAT

1000        FORMAT  (/, ' **** Failed: routine ', a8,
            1         '   Status = ', I7, ' ****', /)

            WRITE (5, 1000) ISRNAM, ISTAT
            STOP
            END
```