```
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFFFFFFFFFFFFFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEEEEEEEEEEE     DDD         DDD  FFFFFFFFFFFF
EEEEEEEEEEEE     DDD         DDD  FFFFFFFFFFFF
EEEEEEEEEEEE     DDD         DDD  FFFFFFFFFFFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEE              DDD         DDD  FFF
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFF
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFF
EEEEEEEEEEEEEEE  DDDDDDDDDDDD     FFF
```

```
**FILE**ID**EDFGRF                                    L  3

EEEEEEEEEE  DDDDDDDD    FFFFFFFFFF    GGGGGGGG  RRRRRRRR    FFFFFFFFFF
EEEEEEEEEE  DDDDDDDD    FFFFFFFFFF    GGGGGGGG  RRRRRRRR    FFFFFFFFFF
EE          DD      DD  FF          GG          RR      RR  FF
EE          DD      DD  FF          GG          RR      RR  FF
EE          DD      DD  FF          GG          RR      RR  FF
EE          DD      DD  FF          GG          RR      RR  FF
EEEEEEEE    DD      DD  FFFFFFFF    GG          RRRRRRRR    FFFFFFFF
EEEEEEEE    DD      DD  FFFFFFFF    GG          RRRRRRRR    FFFFFFFF
EE          DD      DD  FF          GG  GGGGGG  RR  RR      FF
EE          DD      DD  FF          GG  GGGGGG  RR  RR      FF
EE          DD      DD  FF          GG      GG  RR    RR    FF               ....
EE          DD      DD  FF          GG      GG  RR    RR    FF               ....
EEEEEEEEEE  DDDDDDDD    FF            GGGGGG    RR      RR  FF               ....
EEEEEEEEEE  DDDDDDDD    FF            GGGGGG    RR      RR  FF               ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 %TITLE 'EDF$GRAPH plotting module'
0 MODULE EDF$GRAPH (
0                   IDENT = 'V04-000',
0                   ADDRESSING_MODE ( EXTERNAL = GENERAL ),
0                   ADDRESSING_MODE ( NONEXTERNAL = GENERAL ),
0                   OPTLEVEL=3
0                    ) =
1 BEGIN
1
1 !++
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1 !
1 ! FACILITY:              VAX/VMS EDF (EDIT/FDL) UTILITY
1 !
1 ! ABSTRACT:              This facility is used to create, modify, and optimize
1 !                        FDL specification files.
1 !
1 ! ENVIRONMENT:           NATIVE/USER MODE
1 !
1 ! AUTHOR:                Tamar Krichevsky
1 !
1 ! CREATION DATE:         31-Jul-1981
1 !
1 ! MODIFIED BY:
1 !
1 !                        V03-002 KFH0002         Ken Henderson   20 Jan 1983
1 !                                Fixed definition of xxx_INTENSITY
1 !                                constants.
1 !
1 !                        V03-001 KFH0001         Ken Henderson   22 Nov 1982
1 !                                Commented out references to
1 !                                EDF$_INTSWERR; added OPTLEVEL=3
1 !                                Add require of lib$:edfstruct
1 !
1 ! --
```

```
  1 !
  1 ! TABLE OF CONTNETS
  1 !
  1
  1 FORWARD ROUTINE
  1     EDF$GRAPH          : NOVALUE,       !Driver for the plot routines
  1     GET_GRAPH_INFO     : NOVALUE,       !Determines plot characteristics
  1     PLOT_LINE_GRAPH    : NOVALUE,       !Plots a simple graph (line graph)
  1     MOVE_LINE_GRAPH    : NOVALUE,       !Changes the shape of a simple graph
  1     PLOT_SURFACE_GRAPH: NOVALUE,        !Plots a tabular graph (surface plot)
  1     ERASE_PAGE         : NOVALUE,       !Erases a screen
  1     DRAW_Y_AXIS        : NOVALUE,       !Draws the y-axis line
  1     LABEL_Y_AXIS       : NOVALUE,       !Puts the y-axis descriptive labels out
  1     PUT_ROW_SEGMENT    : NOVALUE,       !Put a segment of a graph's row to the terminal
  1     MOVE_CURSOR_REGIS  : NOVALUE,       !Position the cursor (REGIS terminals)
  1     PUT_REGIS_TEXT     : NOVALUE,       !Build and put a REGIS text command
  1     SHADE_ROW_REGIS    : NOVALUE,       !Shade a segment of the graph -- VT125
  1     DRAW_BARS_REGIS    : NOVALUE,       !Draw the bars for the VT125 line graph
  1     DRAW_X_AXIS        : NOVALUE        !Format x-axis lablels
  1     ;
  1
  1
  1 !
  1 ! INCLUDE FILES
  1 !
  1
  1
  1 LIBRARY
  1     'SYS$LIBRARY:STARLET';
  1 REQUIRE
  1     'LIB$:EDFSTRUCT';
  1
  1 !
  1 ! MACROS
  1 !
  1
  1 MACRO
  1     DSC$L_U1 = 32, 0, 32, 0 %,  !Field in the array descriptor for the first
  1                                 !dimension's upper bound
  1     DSC$L_U2 = 40, 0, 32, 0 %,  !Field in the array descriptor for the second
  1                                 !dimension's upper bound
  1
  1
  1     !Translate the given value into ASCII.
M 1     TRANSLATE_VALUE ( CTRSTR_ADR, RESULT_LEN, DESC_ADR, PARAM_LIST) =
M 1         BEGIN
M 1         LOCAL  RTN_STATUS;        !Return status from $FAOL system service
M 1         !Translate, and if not successful, then signal the error.
M 1         IF NOT ( RTN_STATUS = $FAOL( CTRSTR = CTRSTR_ADR,
M 1                                      OUTLEN = RESULT_LEN,
M 1                                      OUTBUF = DESC_ADR,
M 1                                      PRMLST = PARAM_LIST ))
M 1         THEN
M 1             SIGNAL( .RTN_STATUS );
M 1         END
  1     %,
  1
```

```
    !Put the given text to regested location on the screen.
    PUT_TEXT( DESC_ADR, ROW, COLUMN ) =
        BEGIN
        LOCAL RTN_STATUS;          !Return status from RTL call
        IF NOT ( RTN_STATUS = LIB$PUT_SCREEN( DESC_ADR,
                                              ROW,
                                              COLUMN
                                            ))
        THEN
            SIGNAL( .RTN_STATUS );
        END
    %,


    !Put the given text to the terminal; let REGIS position it.
    PUT_REGIS( DESC_ADR ) =
        BEGIN
        LOCAL RTN_STATUS;          !Return status from RTL call
        IF NOT ( RTN_STATUS = LIB$PUT_OUTPUT( DESC_ADR ))
        THEN
            SIGNAL( .RTN_STATUS );
        END
    %


 ;
```

```
1 !
1 ! EQUATED SYMBOLS
1 !
1
1 LITERAL
1     TRUE                 = 1,
1     FALSE                = 0,
1     SUCCESS              = 1,
1     FAILURE              = 0,
1     ALL_BITS_SET         = 255,  !Unsigned -1, used for comparisons
1     FULLWORD             = 4,    !Number of bytes in a BLISS-32 fullword
1     QUAD                 = 8,    !Size of a quad word -- for declarations
1     NULL                 =       !Longword of null characters
1                            %CHAR(0,0,0,0)
1     ;
1
1 LITERAL
1 !     INVALID_GRAPH_CODE  = 100, !Signal arguement - graph code is off the wall
1 !     INVALID_LABEL       = 101, !Signal arguement - y-axis label is too long
1     FIRST_ROW            = 1,    !Position on screen of first row
1     FIRST_COLUMN         = 1,    !Position on screen of first column
1
1     NO_LAST_INDEX        = -1,   !LAST_INDEX value -- signals that graph is to
1                                  !be drawn with axes (as opposed to moved)
1     NO_VALUE             = -1,   !Signals that or y of the pixel address is to
1                                  !be left out (i.e. - [,y] or [x,] )
1
1     MAX_PAGE_WIDTH       = 132,  !Maximum number of columns on a page
1     SCREEN_BUFFER_SIZE   = 512,  !Size of buffer for formatting graphs
1     MAX_BUCKET_SIZE      = 32,   !Maximum number of block a bucket can contain
1     SURFACE_GRAPH_LEN    = 19,   !Maximum number of rows in a tabular graph
1     LINE_GRAPH_LEN       = 10,   !Maximum number of rows in a line graph
1     SEPARATOR_WIDTH      = 2,    !Number columns a plotted value occupies
1     HEADER_HIEGHT        = 3,    !Number of lines in the x-axis header
1
1 !     GRF$_LINE          =  0,   !Code for line  graph
1 !     GRF$_SRF_INCREASING =  1,    !Code for surface graphs with increasing values
1 !     GRF$_SRF_DECREASING =  2,  !Code for surface graphs with decreasing values
1     GRF$_LINE            =  EDF$C_LINE,  !Code for line  graph
1     GRF$_SRF_INCREASING  =  EDF$C_SRF_INCREASING,   !Code for surface graphs with increasing values
1     GRF$_SRF_DECREASING  =  EDF$C_SRF_DECREASING,        !Code for surface graphs with decreasing values
1
1     GRF$_REGIS_POS       =  0,   !Code for position command
1     GRF$_REGIS_VCTR      =  1,   !Code for vector command
1     GRF$_REGIS_SHADE     =  2,   !Code for shaded vector command
1
1     CHAR_WIDTH           =  9,   !Width, in pixels,  of a text-character cell
1     CHAR_HEIGHT          = 15,   !Height, in pixels, of a text_character cell
1     UNIT_WIDTH           = 18,   !Width, in pixels, of a column
1     UNIT_HEIGHT          = 18,   !Height, in pixels, of a row
1     RIGHT_SIDE_LOC       = 767,  !Right most pixel address
1     BOTTOM_LOC           = 479,  !Bottom most pixel address
1     AXIS_SHIFT           = 5,    !Number of pixels from origin axes are located
1
1 !     BACKGROUND_INTENSITY= '0', !Code for background intensity
1 !     LIGHT_INTENSITY     = '3', !Code for light color (green on color monitor)
1 !     MEDIUM_INTENSITY    = '2', !Code for medium color (yellow)
```

```
!    DARK_INTENSITY       = '1',  !code for dark color (red)
    BACKGROUND_INTENSITY= '0' + EDF$C_BACKGROUND_COLOR, !Code for background intensity
    LIGHT_INTENSITY     = '0' + EDF$C_LIGHT_GREEN,      !Code for light color (green on color monitor)
    MEDIUM_INTENSITY    = '0' + EDF$C_MEDIUM_YELLOW,    !Code for medium color (yellow)
    DARK_INTENSITY      = '0' + EDF$C_DARK_RED,         !code for dark color (red)

                                  !Location of graph's x-origin on a VT125
    X_ORIGIN            = RIGHT_SIDE_LOC - (UNIT_WIDTH * MAX_BUCKET_SIZE)
    ;

BIND
    REGIS_SET_UP = %ASCID %STRING(
                                  !Set terminal into REGIS mode
                  %CHAR(27),'Pp',
                                  !Set default screen characteristics
                  'S[0,0] (A[0,0] [767,479], I(d), S1)',
                                  !Set default writing characteristics
                  'W(V, I(w), F3, M1, N0, P1, P(M2), S0)'
                  ),

                                  !Change output color map to RED, YELLOW
                                  !and GREEN colors to signify good, fair
                                  !poor regions of the surface graph.OP
    COLOR_SET_RYG = %ASCID %STRING(
                  'S( M0 (L0)   (a      L0)',        ! background, black
                  '   M1 (L35)  (a h120 L30 s30)',   ! dark RED
                  '   M2 (L65)  (a h150 L70 s50)',   ! medium YELLOW
                  '   M3 (L100) (a h240 L35 s30))'   ! light GREEN
                 ),

                                  !Change output color mapping to a
                                  !more pleasing set of colors
    COLOR_SET_BLUE = %ASCID %STRING(
                  'S( M0 (L0)   (a      L0)',        ! background, black
                  '   M1 (L35)  (a h5   L30 s20)',   ! dark blue
                  '   M2 (L65)  (a h325 L60 s60)',   ! medium blue
                  '   M3 (L100) (a h245 L80 s10))'   ! light blue
                 ),

    REGIS_ON    =                 !Set terminal into REGIS mode
            %ASCID %STRING(  %CHAR(27),'Pp' ),

    REGIS_OFF   =                 !Turn off regis mode
            %ASCID %STRING( %CHAR(27), '\' )
    ;
```

```
!
! Structure declarations used for system defined structures to
! save typing.  These structures are byte sized.
!

STRUCTURE
    BBLOCK [O, P, S, E; N] =
        [N]
        (BBLOCK+O)<P,S,E>,

    BBLOCKVECTOR [I, O, P, S, E; N, BS] =
        [N*BS]
        ((BBLOCKVECTOR+I*BS)+O)<P,S,E>,

    TWO_DIM_ARRAY[ ROW, COL, DESC_ADR; S, E ] =
        [ 0 ]
            BEGIN
            LOCAL DESC_ADR_LCL : REF BBLOCK;

            DESC_ADR_LCL = DESC_ADR;
            (.DESC_ADR_LCL[ DSC$A_POINTER ] +
                ( ROW * .DESC_ADR_LCL[ DSC$L_M2 ] + COL ) * %UPVAL)<0, S, E>
            END
        ;
```

```
1 !
1 ! OWN STORAGE
1 !
1
1 OWN
1     !The following information is needed each time EDF$GRAPH is entered.
1     !Therefore, it is maintained across calls.
1     DEVICE_TYPE  : BYTE INITIAL( ALL_BITS_SET ),!Type of terminal for this session
1     DEVICE_FLAGS : BLOCK[ 1 ],                  !DEC supported terminal
1     Y_AXIS_LINE                                 !Column on which y-axis falls
1     ;
1
1 !
1 ! EXTERNAL REFERENCES
1 !
1 !EXTERNAL LITERAL
1 !    EDF$_INTSWERR                    !Message code for internal software
1 !    ;                                        !error.  This value is used when
1                                      !signalling a wrong graph code or
1                                      !peculiar y-axis label.
1
1
1 EXTERNAL ROUTINE
1     LIB$SCREEN_INFO,                 !Retrieve terminal characteristics.
1     LIB$SET_BUFFER,                  !Set buffer mode on.
1     LIB$ERASE_PAGE,                  !Erase the screen.
1     LIB$PUT_SCREEN,                  !Write output to screen.
1     LIB$PUT_BUFFER,                  !Set buffer mode off.
1     LIB$PUT_OUTPUT                   !Write a line of output to SYS$OUTPUT
1     ;
```

```
1 %SBTTL 'EDF$GRAPH main routine'
1 GLOBAL ROUTINE EDF$GRAPH( GRAPH_TYPE, XY_ARRAY_DESC, CURRENT_INDEX, LAST_INDEX,
1                           Y_HIGH,     Y_LOW,         Y_INCR,
1                           Y_LABEL_DESC,               SHADE_ARRAY_DESC
1                         ) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Determine the user's terminal type and characteristics.  Set up a
1 !       buffer to hold the graph as it is being created.  Plot the requested
1 !       graph.  (Line, Surface)  Put the graph to the user's terminal.
1 !
1 !       If a line graph is requested and the last index (LAST_INDEX
1 !       parameter) equals -1, then the plot will be drawn with the axes and
1 !       labels; if the previous index (LAST_INDEX) is 0 or greater, then
1 !       the graph will be "moved" -- only the inside of the graph will be
1 !       redrawn.  Should the user's terminal be something other than a
1 !       DIGITAL supported video terminal, hard copy or foreign terminal,
1 !       the graph will always be completely replotted.
1 !
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       GRAPH_TYPE      : one byte code which determines which type of graph
1 !                         will be plotted on this code.
1 !       XY_ARRAY_DESC   : address of the array descriptor which points to the
1 !                         array containing the data to be plotted.
1 !       CURRENT_INDEX   : the current index on the y axsis of the array which
1 !                         contains the data for the requested line  graph.
1 !       Y_LOW, Y_HIGH, Y_INCR : the low, high and increment values for the y
1 !                         axsis.
1 !       XY_ARRAY_DESC   : address of the array descriptor which points to the
1 !                         array containing the shading information for VT125
1 !                         surface plots.
1 !
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       DEVICE_FLAGS :  a longword of flags which are set or cleared by
1 !                       LIB$SCREEN_INFO depending on the terminal
1 !                       characteristics.
1 !       DEVICE_TYPE  :  also returned by LIB$SCREEN_INFO.  If all bits are set,
1 !                       then this is the first time EDF$GRAPH is being called.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None, unless an nonrecoverable condition is encountered.  EDF$GRAPH
1 !       will signal the condition for the caller to handle.
1 !
1 ! COMPLETION CODES:
1 !
```

```
1 !         None
1 !
1 ! SIDE EFFECTS:
1 !
1 !         None
1 !
1 !--
1
2 BEGIN
2
2     MAP
2         XY_ARRAY_DESC     : REF BBLOCK,  !Descriptor for the data array
2         SHADE_ARRAY_DESC  : REF BBLOCK   !Descriptor for the shading array
2         ;
2
2     LOCAL
2         OLD_BUFFER,                      !Address of previous screen buffer,
2                                          !if any existed -- otherwise, zero.
2         RTN_STATUS,                      !Status returned from external calls
2         NEW_SURFACE_GRAPH : BYTE,        !Set if graph will be completely rewritten
2         XY_ARRAY          :              !Array to hold the data to be plotted
2             REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
2         SHADE_ARRAY       :              !Array of shading information
2             REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
2
2
2         !This vector and its descriptor will be used to format the different
2         !graphs, before they are written to the user's terminal.
2         !
2         SCREEN_BUFFER     :              !Buffer for the output.
2             VECTOR[ SCREEN_BUFFER_SIZE, BYTE ],
2         SCREEN_BUFFER_DESC :             !Descriptor for the buffer
2             BBLOCK[ DSC$K_Z_BLN ]
2         ;
```

```
2        !Do we know what kind of terminal are we dealing with?
2        !
2        IF .DEVICE_TYPE EQLU ALL_BITS_SET
2        THEN
2
2            !This is the first time EDF$GRAPH has been called.  Determine the
2            !terminal type and check to make sure it is possible to write a
2            !plot to the terminal given its charateristics.
2            !
2            GET_GRAPH_INFO();
2
2
2        !Initialize the screen buffer's descriptor and then set buffer mode on
2        !
2        SCREEN_BUFFER_DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
2        SCREEN_BUFFER_DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
2        SCREEN_BUFFER_DESC[ DSC$W_LENGTH ]  = SCREEN_BUFFER_SIZE;
2        SCREEN_BUFFER_DESC[ DSC$A_POINTER ] = SCREEN_BUFFER;
2
2
2        !Prepare for putting graphs to the user's terminal.
2        !
2        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
2        THEN
2
2            !Turn REGIS mode on and set up default screen characteristics.
2            !
3            PUT_REGIS( REGIS_SET_UP )
3
2        ELSE
2
2            !Set up bufferring for hard copy and ANSI terminals
2            !
3            IF NOT ( RTN_STATUS = LIB$SET_BUFFER( SCREEN_BUFFER_DESC, OLD_BUFFER ))
2            THEN
2
2                !An error has occured while setting buffer mode.
2                !
2                SIGNAL( .RTN_STATUS );
2
2
2        !Which kind of graph was requested?
2        !
2        SELECTONE ..GRAPH_TYPE OF
2            SET
2
2            !Line graph
2            !
2            [ GRF$_LINE ] :
3                BEGIN
3                IF NOT .DEVICE_FLAGS[ SCR$V_REGIS ]
3                           AND
3                   ..LAST_INDEX NEQU NO_LAST_INDEX
3                           AND
3                    .DEVICE_FLAGS[ SCR$V_SCREEN ]
3                THEN
```

```
                    !A line graph was plotted during the last call and the
                    !terminal is a video;  just the shape of the "curve" needs to
                    !be changed.
                    !
                    MOVE_LINE_GRAPH( .XY_ARRAY_DESC, ..CURRENT_INDEX, ..LAST_INDEX)

                ELSE

                    !Either this is the first time for the graph or the user's
                    !terminal is hard copy.  The whole graph must be redrawn.
                    !
                    PLOT_LINE_GRAPH( .XY_ARRAY_DESC, ..CURRENT_INDEX,
                                     .Y_LABEL_DESC, .SHADE_ARRAY_DESC );
                END;

            !Surface graph
            !
            [ GRF$_SRF_INCREASING, GRF$_SRF_DECREASING ] :
                BEGIN
                IF ..LAST_INDEX NEQU NO_LAST_INDEX AND .DEVICE_FLAGS[ SCR$V_SCREEN ]
                THEN

                    !The last graph which was plotted was the same type of surface
                    !graph.  Rewrite the contents of the surface graph and leave
                    !the axes alone.
                    !
                    NEW_SURFACE_GRAPH = FALSE

                ELSE

                    !Write a whole new surface graph, icluding the axes.
                    !
                    NEW_SURFACE_GRAPH = TRUE;

                PLOT_SURFACE_GRAPH( ..GRAPH_TYPE, .XY_ARRAY_DESC,    ..Y_HIGH,
                                    ..Y_LOW,      ..Y_INCR,          .Y_LABEL_DESC,
                                    .NEW_SURFACE_GRAPH,     .SHADE_ARRAY_DESC
                                  );
                END;


            [ OTHERWISE ] :
!               SIGNAL( EDF$_INTSWERR, 1, INVALID_GRAPH_CODE );
                0;

            TES
        ;


        !Clean up before exitting.
        !
        IF (NOT .DEVICE_FLAGS[ SCR$V_REGIS ])
        THEN
            BEGIN

            !Put graph to the terminal.
            !
```

```
4          IF NOT (RTN_STATUS = LIB$PUT_BUFFER( OLD_BUFFER ))
3          THEN
3              SIGNAL( .RTN_STATUS);
3          END
2      ELSE
2
2          !Turn REGIS mode off
2          !
2          PUT_REGIS( REGIS_OFF );
2
2      RETURN;
1  END;                                        ! End of routine EDF$GRAPH
```

```
                                                                      .TITLE  EDF$GRAPH EDF$GRAPH plotting module
                                                                      .IDENT  \V04-000\

                                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

2C 30 5B 41 28 20 5D 30 2C 30 5B 53 70 50 1B  00000 P.AAB:  .ASCII  <27>\PpS[0,0] (A[0,0] [767,479], I(d), \
49 20 2C 5D 39 37 34 2C 37 36 37 5B 20 5D 30  0000F
                              20 2C 29 64 28  0001E
46 20 2C 29 77 28 49 20 2C 56 28 57 29 31 53  00023         .ASCII  \S1)W(V, I(w), F3, M1, N0, P1, P(M2), S0)\
20 2C 31 50 20 2C 30 4E 20 2C 31 4D 20 2C 33  00032
               29 30 53 20 2C 29 32 4D 28 50  00041
                                          00  0004B         .ASCII  <0>
                                    010E004B  0004C P.AAA:  .LONG   17694795
                                   00000000'  00050         .ADDRESS P.AAB
61 28 20 20 20 29 30 6C 28 20 30 4D 20 28 53  00054 P.AAD:  .ASCII  \S( M0 (l0)   (a      l0)   M1 (l35)  (a \
20 31 4D 20 20 20 29 30 6C 20 20 20 20 20 20  00063
               20 61 28 20 20 29 35 33 6C 28  00072
20 20 29 30 33 73 20 30 33 6C 20 30 32 31 68  0007C         .ASCII  \h120 l30 s30)   M2 (l65)  (a h150 l70 s5\
68 20 61 28 20 20 29 35 36 6C 28 20 32 4D 20  0008B
               35 73 20 30 37 6C 20 30 35 31  0009A
20 29 30 30 31 6C 28 20 33 4D 20 20 20 29 30  000A4         .ASCII  \0)   M3 (l100) (a h240 l35 s30))\
30 33 73 20 35 33 6C 20 30 34 32 68 20 61 28  000B3
                                       29 29  000C2
                                    010E0070  000C4 P.AAC:  .LONG   17694832
                                   00000000'  000C8         .ADDRESS P.AAD
61 28 20 20 20 29 30 6C 28 20 30 4D 20 28 53  000CC P.AAF:  .ASCII  \S( M0 (l0)   (a      l0)   M1 (l35)  (a \
20 31 4D 20 20 20 29 30 6C 20 20 20 20 20 20  000DB
               20 61 28 20 20 29 35 33 6C 28  000EA
20 20 29 30 32 73 20 30 33 6C 20 20 20 35 68  000F4         .ASCII  \h5   l30 s20)   M2 (l65)  (a h325 l60 s6\
68 20 61 28 20 20 29 35 36 6C 28 20 32 4D 20  00103
               36 73 20 30 36 6C 20 35 32 33  00112
20 29 30 30 31 6C 28 20 33 4D 20 20 20 29 30  0011C         .ASCII  \0)   M3 (l100) (a h245 l80 s10))\
30 31 73 20 30 38 6C 20 35 34 32 68 20 61 28  0012B
                                       29 29  0013A
                                    010E0070  0013C P.AAE:  .LONG   17694832
                                   00000000'  00140         .ADDRESS P.AAF
                                 00 70 50 1B  00144 P.AAH:  .ASCII  <27>\Pp\<0>
                                    010E0003  00148 P.AAG:  .LONG   17694723
                                   00000000'  0014C         .ADDRESS P.AAH
                                 00 00 5C 1B  00150 P.AAJ:  .ASCII  <27><92><0><0>
                                    010E0002  00154 P.AAI:  .LONG   17694722
                                   00000000'  00158         .ADDRESS P.AAJ
```

```
                                                           .PSECT   $OWN$,NOEXE,2

                                      FF  00000 DEVICE_TYPE:
                                                           .BYTE    -1                                    ;
                                          00001            .BLKB    3
                                          00004 DEVICE_FLAGS:
                                                           .BLKB    4
                                          00008 Y_AXIS_LINE:
                                                           .BLKB    4

                                                REGIS_SET_UP=        P.AAA
                                                COLOR_SET_RYG=       P.AAC
                                                COLOR_SET_BLUE=      P.AAE
                                                REGIS_ON=            P.AAG
                                                REGIS_OFF=           P.AAI
                                                           .EXTRN   LIB$SCREEN_INFO
                                                           .EXTRN   LIB$SET_BUFFER, LIB$ERASE_PAGE
                                                           .EXTRN   LIB$PUT_SCREEN, LIB$PUT_BUFFER
                                                           .EXTRN   LIB$PUT_OUTPUT

                                                           .PSECT   $CODE$,NOWRT,2

                                    007C  00000            .ENTRY   EDF$GRAPH, Save R2,R3,R4,R5,R6        ; 0593
         56 00000000G          00    9E  00002            MOVAB    LIB$SIGNAL, R6                        ;
         55 00000000G          00    9E  00009            MOVAB    LIB$PUT_OUTPUT, R5                    ;
         54 00000000'          00    9E  00010            MOVAB    DEVICE_FLAGS, R4                      ;
         5E       FDF4         CE    9E  00017            MOVAB    -524(SP), SP                          ;
    FF   8F         FC         A4    91  0001C            CMPB     DEVICE_TYPE, #255                     ; 0686
                               07    12  00021            BNEQ     1$                                    ;
00000000V  00                  00    FB  00023            CALLS    #0, GET_GRAPH_INFO                    ; 0693
      04   AE     0200         8F    3C  0002A 1$:        MOVZWL   #512, SCREEN_BUFFER_DESC              ; 0700
      08   AE       0C         AE    9E  00030            MOVAB    SCREEN_BUFFER, SCREEN_BUFFER_DESC+4   ; 0701
 10        64                  02    E1  00035            BBC      #2, DEVICE_FLAGS, 2$                  ; 0706
            00000000'          00    9F  00039            PUSHAB   REGIS_SET_UP                          ; 0711
           65                  01    FB  0003F            CALLS    #1, LIB$PUT_OUTPUT                    ;
           1B                  50    E8  00042            BLBS     RTN_STATUS, 4$                        ;
                               50    DD  00045            PUSHL    RTN_STATUS                            ;
                               14    11  00047            BRB      3$                                    ;
                               5E    DD  00049 2$:        PUSHL    SP                                    ; 0717
                    08         AE    9F  0004B            PUSHAB   SCREEN_BUFFER_DESC                    ;
00000000G  00                  02    FB  0004E            CALLS    #2, LIB$SET_BUFFER                    ;
           53                  50    D0  00055            MOVL     R0, RTN_STATUS                        ;
           05                  53    E8  00058            BLBS     RTN_STATUS, 4$                        ;
                               53    DD  0005B            PUSHL    RTN_STATUS                            ; 0722
           66                  01    FB  0005D 3$:        CALLS    #1, LIB$SIGNAL                        ;
           52       04         BC    D0  00060 4$:        MOVL     @GRAPH_TYPE, R2                       ; 0727
                               36    12  00064            BNEQ     6$                                    ; 0732
 1F        64                  02    E0  00066            BBS      #2, DEVICE_FLAGS, 5$                  ; 0734
FFFFFFFF   8F       10         BC    D1  0006A            CMPL     @LAST_INDEX, #-1                      ; 0736
                               15    13  00072            BEQL     5$                                    ;
           12                  64    E9  00074            BLBC     DEVICE_FLAGS, 5$                      ; 0738
                    10         BC    DD  00077            PUSHL    @LAST_INDEX                           ; 0745
                    0C         BC    DD  0007A            PUSHL    @CURRENT_INDEX                        ;
                    08         AC    DD  0007D            PUSHL    XY_ARRAY_DESC                         ;
00000000V  00                  03    FB  00080            CALLS    #3, MOVE_LINE_GRAPH                   ;
                               4C    11  00087            BRB      9$                                    ;
           7E       20         AC    7D  00089 5$:        MOVQ     Y_LABEL_DESC, -(SP)                   ; 0753
```

```
                                   0C  BC  DD 0008D       PUSHL    @CURRENT_INDEX                        ; 0752
                                   08  AC  DD 00090       PUSHL    XY_ARRAY_DESC                          ;
              00000000V  00            04  FB 00093       CALLS    #4, PLOT_LINE_GRAPH                    ;
                                       39  11 0009A       BRB      9$                                     ; 0727
                                       37  15 0009C 6$:   BLEQ     9$                                     ; 0758
                         02            52  D1 0009E       CMPL     R2, #2                                 ;
                                       32  14 000A1       BGTR     9$                                     ;
              FFFFFFFF   8F        10  BC  D1 000A3       CMPL     @LAST_INDEX, #-1                       ; 0760
                                       07  13 000AB       BEQL     7$                                     ;
                         04            64  E9 000AD       BLBC     DEVICE_FLAGS, 7$                       ;
                                       50  94 000B0       CLRB     NEW_SURFACE_GRAPH                      ; 0767
                                       03  11 000B2       BRB      8$                                     ;
                         50            01  90 000B4 7$:   MOVB     #1, NEW_SURFACE_GRAPH                  ; 0773
                                   24  AC  DD 000B7 8$:   PUSHL    SHADE_ARRAY_DESC                       ; 0777
                         7E            50  9A 000BA       MOVZBL   NEW_SURFACE_GRAPH, -(SP)               ;
                                   20  AC  DD 000BD       PUSHL    Y_LABEL_DESC                           ; 0776
                                   1C  BC  DD 000C0       PUSHL    @Y_INCR                                ;
                                   18  BC  DD 000C3       PUSHL    @Y_LOW                                 ;
                                   14  BC  DD 000C6       PUSHL    @Y_HIGH                                ; 0775
                                   08  AC  DD 000C9       PUSHL    XY_ARRAY_DESC                          ;
                                       52  DD 000CC       PUSHL    R2                                     ;
              00000000V  00            08  FB 000CE       CALLS    #8, PLOT_SURFACE_GRAPH                 ;
          13             64            02  E0 000D5 9$:   BBS      #2, DEVICE_FLAGS, 10$                  ; 0792
                                       5E  DD 000D9       PUSHL    SP                                     ; 0798
              00000000G  00            01  FB 000DB       CALLS    #1, LIB$PUT_BUFFER                     ;
                         53            50  D0 000E2       MOVL     R0, RTN_STATUS                         ;
                         15            53  E8 000E5       BLBS     RTN_STATUS, 12$                        ;
                                       53  DD 000E8       PUSHL    RTN_STATUS                             ; 0800
                                       0E  11 000EA       BRB      11$                                    ;
                         00000000'     00  9F 000EC 10$:  PUSHAB   REGIS_OFF                              ; 0806
                         65            01  FB 000F2       CALLS    #1, LIB$PUT_OUTPUT                     ;
                         05            50  E8 000F5       BLBS     RTN_STATUS, 12$                        ;
                                       50  DD 000F8       PUSHL    RTN_STATUS                             ;
                         66            01  FB 000FA 11$:  CALLS    #1, LIB$SIGNAL                         ;
                                           04 000FD 12$:  RET                                             ; 0809

; Routine Size:  254 bytes,    Routine Base:  $CODE$ + 0000
```

```
1 %SBTTL 'Get the graph characteristics'
1 ROUTINE GET_GRAPH_INFO   : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Determine the terminal characteristics.  Determine the width of the
1 !       graph.  Set up any screen characteristics, if this is a terminal
1 !       which understands REGIS.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Y_AXIS_LINE   :  column which contains the y-axis line.  This value
1 !                        is only output for non-REGIS terminals.
1 !       DEVICE_FLAGS  :  a longword of flags which are set or cleared by
1 !                        LIB$SCREEN_INFO depending on the terminal
1 !                        characteristics.
1 !       DEVICE_TYPE   :  also returned by LIB$SCREEN_INFO.  If all bits are set,
1 !                        then this is the first time EDF$GRAPH is being called.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the user's terminla understands REGIS, the screen defaults may be
1 !       altered.
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2     SKINNY_LINE         = 80,
2     FAT_LINE            = 132,
2     WIDTH_BOUNDARY      = 100    !Minimum columns needed to format a wider graph
2     ;
2
2 LOCAL
2     RTN_STATUS,                  !Return status from external calls
2     LINE_WIDTH      : WORD,      !Width of page for this session
2     LINES_PER_PAGE  : WORD       !Number of lines per page on the terminal
2     ;
```

```
; 581          0867  2
```

```
2       !Determine the terminal type and characteristics.
2       !
3       IF NOT ( RTN_STATUS = LIB$SCREEN_INFO( DEVICE_FLAGS,  DEVICE_TYPE,
3                                              LINE_WIDTH,    LINES_PER_PAGE
3                                            ))
2       THEN
2
2           !An error has occured while obtaining the device information.
2           !
2           SIGNAL( .RTN_STATUS );
2
2
2       !If terminal understands REGIS, set up screen characteristics needed
2       !for plotting the graphs.  Otherwise, position the y-axis line.
2       !
2       IF .DEVICE_FLAGS[ SCR$V_REGIS ]
2       THEN
2           Y_AXIS_LINE  = X_ORIGIN - AXIS_SHIFT
2
2       ELSE
2
2           !Determine where the y-axis line should be positioned.
2           !
3           IF  (.LINE_WIDTH LEQ WIDTH_BOUNDARY
3                       OR
3               (NOT .DEVICE_FLAGS[ SCR$V_SCREEN ]))
2           THEN
2
2               !Have the right most column of the graph fall on the right most
2               !column of the page.  If the user's terminal is a hard copy (bit
2               !zero clear in DEVICE_FLAGS), then have the y-axis line fall to
2               !the left -- to reduce the number of characters being printed.
2               !
2               Y_AXIS_LINE = SKINNY_LINE -
2                             ((MAX_BUCKET_SIZE + 1) * SEPARATOR_WIDTH) + 1
2           ELSE
2
2               !If the line is wide enough, then center the graph across the line.
2               !
3               Y_AXIS_LINE = (FAT_LINE -
2                             ((MAX_BUCKET_SIZE + 1) * SEPARATOR_WIDTH)) / 2;
2
2       RETURN;
1 END;                                          ! End of routine GET_GRAPH_INFO



                                0004 00000 GET_GRAPH_INFO:
                                                 .WORD    Save R2                        ; 0811
        52 00000000'   00  9E 00002              MOVAB    DEVICE_FLAGS, R2               ;
        5E             08  C2 00009              SUBL2    #8, SP                         ;
                       5E  DD 0000C              PUSHL    SP                             ; 0870
               08      AE  9F 0000E              PUSHAB   LINE_WIDTH                     ;
               FC      A2  9F 00011              PUSHAB   DEVICE_TYPE                    ;
                       52  DD 00014              PUSHL    R2                             ;
```

```
        00000000G  00        04  FB 00016      CALLS   #4, LIB$SCREEN_INFO          ;
                   09        50  E8 0001D      BLBS    RTN_STATUS, 1$               ;
                             50  DD 00020      PUSHL   RTN_STATUS                   ; 0877
        00000000G  00        01  FB 00022      CALLS   #1, LIB$SIGNAL               ;
     06            62        02  E1 00029 1$:  BBC     #2, DEVICE_FLAGS, 2$         ; 0883
               04  A2    BA  8F  9A 0002D      MOVZBL  #186, Y_AXIS_LINE            ; 0885
                                 04 00032      RET                                  ;
             0064  8F    04  AE  B1 00033 2$:  CMPW    LINE_WIDTH, #100             ; 0891
                             03  1B 00039      BLEQU   3$                           ;
                   05        62  E8 0003B      BLBS    DEVICE_FLAGS, 4$             ; 0893
               04  A2        0F  D0 0003E 3$:  MOVL    #15, Y_AXIS_LINE             ; 0902
                                 04 00042      RET                                  ; 0901
               04  A2        21  D0 00043 4$:  MOVL    #33, Y_AXIS_LINE             ; 0908
                                 04 00047      RET                                  ; 0911

; Routine Size:  72 bytes,    Routine Base:  $CODE$ + 00FE
```

```
%SBTTL 'Plot a line graph'
ROUTINE PLOT_LINE_GRAPH( XY_ARRAY_DESC, CURRENT_INDEX, Y_LABEL_DESC,
                         SHADE_ARRAY_DESC ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       For each row in the graph:  write the y-axis label and mark
!       all x-values in the x,y pairs with the y-values which belong
!       in this row.  Draw and label the x-axis.
!
!
! FORMAL PARAMETERS:
!
!       XY_ARRAY_DESC    :  address of the descriptor for the data array
!       CURRENT_INDEX    :  index of the "row" (first dimension) requested
!                           for graphing
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       A line graph
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       Filling of SCREEN_BUFFER; adjusting the values in SCREEN_BUFFER_DESC.
!
!--

BEGIN

MAP
    XY_ARRAY_DESC : REF BBLOCK; !Descriptor for the data array


LOCAL
    XY_ARRAY              :      !Array to hold the data to be plotted
            REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
    RTN_STATUS,                  !Status returned from external calls
    DIM2,                        !Index for the array's second dimension
    ROW_LABEL,                   !Label for y-axis value
    ROW_POS,                     !Current row on graph
    COLUMN_POS,                  !Column postion for output
    SEPARATOR_CHAR,              !Dummy arguement to PUT_ROW_SEGMENT
    CURRENT_VALUE,               !Current xy-value to be plotted
```

```
; 685      0969  2      REPEAT_COUNT                     !Number of times current xy-value is in row
; 686      0970  2      ;
; 687      0971  2
```

```
2        !Erase the screen to give the graph a "clean slate".
2        !
2        ERASE_PAGE();
2
2
2        !Set up colors for the graph.
2        !
2        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
2        THEN
2            PUT_REGIS( COLOR_SET_BLUE );
2
2
2        !Draw and label the y-axis
2        !
2        DRAW_Y_AXIS( LINE_GRAPH_LEN, GRF$_LINE, NULL, NULL );
2        LABEL_Y_AXIS( .Y_LABEL_DESC, LINE_GRAPH_LEN );
2
2
2        !Create x-axis labels and put them to the screen
2        !
2        DRAW_X_AXIS( LINE_GRAPH_LEN );
2
2
2        !Inititalize those variables needed to create the graph's "curve".
2        !
2        DIM2           = 0;
2        SEPARATOR_CHAR = NULL;
2
2
2        !For each row (y-value) in the graph:
2        !
3        INCR ROW_COUNT FROM 0 TO (LINE_GRAPH_LEN - 1)
2        DO
3            BEGIN
3
3            !Determine which row is being written and what it's label is.
3            !
3            ROW_POS   = .ROW_COUNT + 1;
3            ROW_LABEL = LINE_GRAPH_LEN - .ROW_COUNT;
3
3
3            !If there are more elements for this row of the xy-array and they have
3            !the correct value, then print their value.
3            !
4            WHILE (.DIM2 LEQ .XY_ARRAY_DESC[ DSC$L_U2 ]
4                         AND
5                   (.XY_ARRAY[ .CURRENT_INDEX,.DIM2,.XY_ARRAY_DESC ] GEQ .ROW_LABEL
5                         OR
4                   .XY_ARRAY[ .CURRENT_INDEX,.DIM2,.XY_ARRAY_DESC ] LEQ 0))
3            DO
4                BEGIN
4
4                !Save the current xy-value for comparisons; determine the position
4                !in the row.
4                !
4                CURRENT_VALUE = .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ];
4                COLUMN_POS    = .DIM2;
```

```
                !Compare the current value to all subsequent values in the row
                !until a value which is not equal is encountered.  Keep a count
                !of the equal values, so the right number can be printed in the
                !graph.
                !
                DIM2 = .DIM2 + 1;
                WHILE .DIM2 LEQ .XY_ARRAY_DESC[ DSC$L_U2 ]
                            AND
                      (.CURRENT_VALUE EQL .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ]
                            OR
                      .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC] LEQ 0)
                DO
                    DIM2 = .DIM2 + 1;


                !Calculate the number of equal values and write them out.
                !
                REPEAT_COUNT = .DIM2 - .COLUMN_POS;

                IF .DEVICE_FLAGS[ SCR$V_REGIS ]
                THEN
                    BEGIN
                    IF .CURRENT_VALUE GTR 0
                    THEN
                        DRAW_BARS_REGIS( .ROW_POS, .COLUMN_POS, .REPEAT_COUNT );
                    END

                ELSE
                    PUT_ROW_SEGMENT( .CURRENT_VALUE, .REPEAT_COUNT,   .COLUMN_POS,
                                     .ROW_POS, .SEPARATOR_CHAR, .CURRENT_INDEX,
                                     .SHADE_ARRAY_DESC);

                END;                                    !DIM2

            END;                                        !ROW_COUNT

    END;                                                ! End of routine PLOT_LINE_GRAPH
```

```
                                     0FFC 00000 PLOT_LINE_GRAPH:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0913
             5B 00000000'    00   9E 00002             MOVAB   DEVICE_FLAGS, R11                           ;
     00000000V   00              00   FB 00009             CALLS   #0, ERASE_PAGE                              ; 0974
   19            6B              02   E1 00010             BBC     #2, DEVICE_FLAGS, 1$                        ; 0979
                    00000000'    00   9F 00014             PUSHAB  COLOR_SET_BLUE                              ; 0981
     00000000G   00              01   FB 0001A             CALLS   #1, LIB$PUT_OUTPUT                          ;
                 09              50   E8 00021             BLBS    RTN_STATUS, 1$                              ;
                                 50   DD 00024             PUSHL   RTN_STATUS                                  ;
     00000000G   00              01   FB 00026             CALLS   #1, LIB$SIGNAL                              ;
                                 7E   7C 0002D 1$:         CLRQ    -(SP)                                       ; 0986
                 7E              0A   7D 0002F             MOVQ    #10, -(SP)                                  ;
     00000000V   00              04   FB 00032             CALLS   #4, DRAW_Y_AXIS                             ;
```

```
                                            0A  DD 00039           PUSHL   #10                                            ; 0987
                                      0C    AC  DD 0003B           PUSHL   Y_LABEL_DESC                                   ;
                 00000000V  00              02  FB 0003E           CALLS   #2, LABEL_Y_AXIS                               ;
                                            0A  DD 00045           PUSHL   #10                                            ; 0992
                 00000000V  00              01  FB 00047           CALLS   #1, DRAW_X_AXIS                                ;
                                            56  D4 0004E           CLRL    DIM2                                           ; 0997
                                            5A  D4 00050           CLRL    SEPARATOR_CHAR                                 ; 0998
                            55        04    AC  D0 00052           MOVL    XY_ARRAY_DESC, R5                              ; 1016
                                            52  D4 00056           CLRL    ROW_COUNT                                      ; 1027
                            53        01    A2  9E 00058 2$:       MOVAB   1(R2), ROW_POS                                 ; 1009
          54                0A              52  C3 0005C           SUBL3   ROW_COUNT, #10, ROW_LABEL                      ; 1010
                      28    A5              56  D1 00060 3$:       CMPL    DIM2, 40(R5)                                   ; 1016
                                            23  14 00064           BGTR    4$                                             ;
                            50              55  D0 00066           MOVL    R5, DESC_ADR_LCL                               ; 1018
          51          08    AC        18    A0  C5 00069           MULL3   24(DESC_ADR_LCL), CURRENT_INDEX, R1            ;
                            51              56  C0 0006F           ADDL2   DIM2, R1                                       ;
                            54       04 B041    D1 00072           CMPL    @4(DESC_ADR_LCL)[R1], ROW_LABEL                ;
                                            12  18 00077           BGEQ    5$                                             ;
                            50              55  D0 00079           MOVL    R5, DESC_ADR_LCL                               ; 1020
          51          08    AC        18    A0  C5 0007C           MULL3   24(DESC_ADR_LCL), CURRENT_INDEX, R1            ;
                            51              56  C0 00082           ADDL2   DIM2, R1                                       ;
                                     04 B041    D5 00085           TSTL    @4(DESC_ADR_LCL)[R1]                           ;
                                            74  14 00089 4$:       BGTR    10$                                            ;
                            51              55  D0 0008B 5$:       MOVL    R5, DESC_ADR_LCL                               ; 1027
          50          08    AC        18    A1  C5 0008E           MULL3   24(DESC_ADR_LCL), CURRENT_INDEX, R0            ;
                            50              56  C0 00094           ADDL2   DIM2, R0                                       ;
                            57       04 B140    D0 00097           MOVL    @4(DESC_ADR_LCL)[R0], CURRENT_VALUE            ;
                            58              56  D0 0009C           MOVL    DIM2, COLUMN_POS                               ; 1028
                                            56  D6 0009F 6$:       INCL    DIM2                                           ; 1036
                      28    A5              56  D1 000A1           CMPL    DIM2, 40(R5)                                   ; 1037
                                            25  14 000A5           BGTR    7$                                             ;
                            50              55  D0 000A7           MOVL    R5, DESC_ADR_LCL                               ; 1039
          51          08    AC        18    A0  C5 000AA           MULL3   24(DESC_ADR_LCL), CURRENT_INDEX, R1            ;
                            51              56  C0 000B0           ADDL2   DIM2, R1                                       ;
                      04 B041               57  D1 000B3           CMPL    CURRENT_VALUE, @4(DESC_ADR_LCL)[R1]            ;
                                            E5  13 000B8           BEQL    6$                                             ;
                            50              55  D0 000BA           MOVL    R5, DESC_ADR_LCL                               ; 1041
          51          08    AC        18    A0  C5 000BD           MULL3   24(DESC_ADR_LCL), CURRENT_INDEX, R1            ;
                            51              56  C0 000C3           ADDL2   DIM2, R1                                       ;
                                     04 B041    D5 000C6           TSTL    @4(DESC_ADR_LCL)[R1]                           ;
                                            D3  15 000CA           BLEQ    6$                                             ;
          59                56              58  C3 000CC 7$:       SUBL3   COLUMN_POS, DIM2, REPEAT_COUNT                 ; 1048
          11                6B              02  E1 000D0           BBC     #2, DEVICE_FLAGS, 8$                           ; 1050
                                            57  D5 000D4           TSTL    CURRENT_VALUE                                  ; 1053
                                            88  15 000D6           BLEQ    3$                                             ;
                                    0308    8F  BB 000D8           PUSHR   #^M<R3,R8,R9>                                  ; 1055
                 00000000V  00              03  FB 000DC           CALLS   #3, DRAW_BARS_REGIS                            ;
                                            17  11 000E3           BRB     9$                                             ; 1050
                                      10    AC  DD 000E5 8$:       PUSHL   SHADE_ARRAY_DESC                               ; 1061
                                      08    AC  DD 000E8           PUSHL   CURRENT_INDEX                                  ; 1060
                                    0408    8F  BB 000EB           PUSHR   #^M<R3,R10>                                    ;
                                            58  DD 000EF           PUSHL   COLUMN_POS                                     ; 1059
                                    0280    8F  BB 000F1           PUSHR   #^M<R7,R9>                                     ;
                 00000000V  00              07  FB 000F5           CALLS   #7, PUT_ROW_SEGMENT                            ;
                                          FF61  31 000FC 9$:       BRW     3$                                             ; 1016
     FF53        52                 01      09  F1 000FF 10$:      ACBL    #9, #1, ROW_COUNT, 2$                          ; 1003
                                                04 00105           RET                                                    ; 1067
```

```
; Routine Size:  262 bytes,    Routine Base:  $CODE$ + 0146
```

```
%SBTTL 'Move the curve on a line graph'
ROUTINE MOVE_LINE_GRAPH (XY_ARRAY_DESC, CURRENT_INDEX, LAST_INDEX ) :
NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       MOVE_LINE_GRAPH changes the existing curve of the graph.
!       It checks each value in the current row of th xy-array against
!       the appropriate value in the last row.  If there is a difference,
!       the old value is erased and the new value is written out at the
!       proper location.
!
! FORMAL PARAMETERS:
!
!       XY_ARRAY_DESC   :  Address of xy-array descriptor
!       CURRENT_INDEX   :  Index of current row to be plotted, in xy-array
!       LAST_INDEX      :  Index of last row to be plotted
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

BEGIN

MAP
    XY_ARRAY_DESC : REF BBLOCK; !Descriptor for the data array

LITERAL
    CTRSTR_LEN = 4,                 !Length of the control string
    LEN_FIELD  = 1                  !Position of length field in the control string
    ;

LOCAL
    XY_ARRAY            :           !Array to hold the data to be plotted
            REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
    ROW_POS,                        !Current row on graph
    COLUMN_POS,                     !Column postion for output
```

```
;  843    1125  2       BUFFER          :               !Buffer for building the output
;  844    1126  2           VECTOR[ MAX_PAGE_WIDTH, BYTE ],
;  845    1127  2       DESC            :               !Descriptor for BUFFER
;  846    1128  2           BBLOCK[ DSC$K_Z_BLN ]
;  847    1129  2       ;
;  848    1130  2
;  849    1131  2 BIND
;  850    1132  2       CTRSTR_DESC = %ASCID'!2UB'
;  851    1133  2       ;
```

```
!Intialize variables
!
(BUFFER)                = '    ';
DESC[ DSC$B_CLASS ]    = DSC$K_CLASS_Z;
DESC[ DSC$B_DTYPE ]    = DSC$K_DTYPE_Z;
DESC[ DSC$A_POINTER ] = BUFFER;


!For each value:
INCR DIM2 FROM 0 TO .XY_ARRAY_DESC[ DSC$L_U2 ]
DO

    !If the value at the current index is not equal to the value at the
    !old index, then the curve must be altered... erase the old value
    !and write the new one in the appropriate row.
    !
    IF .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ]
          NEQ
       .XY_ARRAY[ .LAST_INDEX, .DIM2, .XY_ARRAY_DESC ]
    THEN
        BEGIN

        !Erase the old value.
        !
        INCR CHAR_POS FROM 0 TO ( SEPARATOR_WIDTH - 1 ) DO
             BUFFER[ .CHAR_POS ] = ' ';
        DESC[ DSC$W_LENGTH ]     = SEPARATOR_WIDTH;


        !Determine which row the value to be erased is in.
        !
        IF .XY_ARRAY[ .LAST_INDEX, .DIM2, .XY_ARRAY_DESC ] GEQ LINE_GRAPH_LEN
                   OR
           .XY_ARRAY[ .LAST_INDEX, .DIM2, .XY_ARRAY_DESC ] LEQ 0
        THEN
            ROW_POS = FIRST_ROW
        ELSE
            ROW_POS = LINE_GRAPH_LEN -
                      .XY_ARRAY[ .LAST_INDEX, .DIM2, .XY_ARRAY_DESC ] + 1;


        !Which column does it reside in?
        !
        COLUMN_POS  = (.DIM2 + 1) * SEPARATOR_WIDTH + .Y_AXIS_LINE;


        !Write over the old value with blanks.
        !
        PUT_TEXT( DESC, ROW_POS, COLUMN_POS );


        !Translate the new value to be placed on the curve.
        !
        IF .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ] LEQ 0
        THEN

            BEGIN
```

```
;     4                      (BUFFER[ 0 ])         = '  ';
;     4                      DESC[ DSC$W_LENGTH ] = SEPARATOR_WIDTH;
;     4                      END
;     3
;     3                  ELSE
;   P 3                      TRANSLATE_VALUE( CTRSTR_DESC, DESC[ DSC$W_LENGTH ], DESC,
;   P 3                                       XY_ARRAY[ .CURRENT_INDEX, .DIM2,
;     3                                                 .XY_ARRAY_DESC ]);
;     3
;     3                  !Determine which row the value to be written should be in.
;     3                  !
;     3                  IF .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ] GEQ LINE_GRAPH_LEN
;     3                              OR
;     3                     .XY_ARRAY[ .CURRENT_INDEX, .DIM2, .XY_ARRAY_DESC ] LEQ 0
;     3                  THEN
;     3                      ROW_POS = FIRST_ROW
;     3                  ELSE
;     3                      ROW_POS = LINE_GRAPH_LEN -
;     3                            .XY_ARRAY[ .CURRENT_INDEX,.DIM2,.XY_ARRAY_DESC ] + 1;
;     3
;     3
;     3                  !Write over the old value with blanks.
;     3                  !
;     3                  PUT_TEXT( DESC, ROW_POS, COLUMN_POS );
;     2                  END;
;     1 END;                                         ! End of routine MOVE_LINE_GRAPH


                                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

                       42  55  32  21  0015C P.AAL:      .ASCII  \!2UB\                          ;
                             010E0004  00160 P.AAK:      .LONG   17694724                        ;
                             00000000' 00164             .ADDRESS P.AAL                          ;

                                                 CTRSTR_DESC=        P.AAK
                                                         .EXTRN  SYS$FAOL

                                                         .PSECT  $CODE$,NOWRT,2

                                     01FC 00000 MOVE_LINE_GRAPH:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8       ; 1069
                 58 00000000G   00   9E 00002            MOVAB   LIB$PUT_SCREEN, R8              ;
                 57 00000000G   00   9E 00009            MOVAB   LIB$SIGNAL, R7                  ;
                 5E      FF6C   CE   9E 00010            MOVAB   -148(SP), SP                    ;
           10    AE  20202020   8F   D0 00015            MOVL    #538976288, BUFFER              ; 1136
                           0A   AE   B4 0001D            CLRW    DESC+2                          ; 1138
           0C    AE        10   AE   9E 00020            MOVAB   BUFFER, DESC+4                  ; 1139
                 55        04   AC   7D 00025            MOVQ    XY_ARRAY_DESC, R5               ; 1143
                 54             01   CE 00029            MNEGL   #1, DIM2                        ; 1152
                              0121   31 0002C 1$:        BRW     13$                             ;
                 52             55   D0 0002F 2$:        MOVL    R5, DESC_ADR_LCL                ; 1150
      51         56        18   A2   C5 00032            MULL3   24(DESC_ADR_LCL), R6, R1        ;
                 51             54   C0 00037            ADDL2   DIM2, R1                        ;
                 50             55   D0 0003A            MOVL    R5, DESC_ADR_LCL                ; 1152
      53   0C    AC        18   A0   C5 0003D            MULL3   24(DESC_ADR_LCL), LAST_INDEX, R3 ;
```

```
                      53        54  C0 00043        ADDL2   DIM2, R3                                          ;
                 04 B043   04 B241  D1 00046        CMPL    @4(DESC_ADR_LCL)[R1], @4(DESC_ADR_LCL)[R3]        ;
                                DD  13 0004D        BEQL    1$                                                ;
                                50  D4 0004F        CLRL    CHAR_POS                                          ; 1158
                 10 AE40        20  90 00051 3$:    MOVB    #32, BUFFER[CHAR_POS]                             ; 1159
            F7        50        01  F3 00056        AOBLEQ  #1, CHAR_POS, 3$                                  ;
            08        AE        02  B0 0005A        MOVW    #2, DESC                                          ; 1160
                      50        55  D0 0005E        MOVL    R5, DESC_ADR_LCL                                  ; 1165
      51    0C        AC   18   A0  C5 00061        MULL3   24(DESC_ADR_LCL), LAST_INDEX, R1                  ;
                      51        54  C0 00067        ADDL2   DIM2, R1                                          ;
                      0A   04 B041  D1 0006A        CMPL    @4(DESC_ADR_LCL)[R1], #10                         ;
                                12  18 0006F        BGEQ    4$                                                ;
                      50        55  D0 00071        MOVL    R5, DESC_ADR_LCL                                  ; 1167
      51    0C        AC   18   A0  C5 00074        MULL3   24(DESC_ADR_LCL), LAST_INDEX, R1                  ;
                      51        54  C0 0007A        ADDL2   DIM2, R1                                          ;
                           04 B041  D5 0007D        TSTL    @4(DESC_ADR_LCL)[R1]                              ;
                                06  14 00081        BGTR    5$                                                ;
            04        AE        01  D0 00083 4$:    MOVL    #1, ROW_POS                                       ; 1169
                                13  11 00087        BRB     6$                                                ;
                      50        55  D0 00089 5$:    MOVL    R5, DESC_ADR_LCL                                  ; 1172
      51    0C        AC   18   A0  C5 0008C        MULL3   24(DESC_ADR_LCL), LAST_INDEX, R1                  ;
                      51        54  C0 00092        ADDL2   DIM2, R1                                          ;
04    AE              0B   04 B041  C3 00095        SUBL3   @4(DESC_ADR_LCL)[R1], #11, ROW_POS                ;
                      50 00000000'  00  D0 0009C 6$:    MOVL    Y_AXIS_LINE, R0                                   ; 1177
                      6E   02 A044  3E 000A3        MOVAW   2(R0)[DIM2], COLUMN_POS                           ;
                                5E  DD 000A8        PUSHL   SP                                                ; 1182
                      08        AE  9F 000AA        PUSHAB  ROW_POS                                           ;
                      10        AE  9F 000AD        PUSHAB  DESC                                              ;
                      68        03  FB 000B0        CALLS   #3, LIB$PUT_SCREEN                                ;
                      05        50  E8 000B3        BLBS    RTN_STATUS, 7$                                    ;
                                50  DD 000B6        PUSHL   RTN_STATUS                                        ;
                      67        01  FB 000B8        CALLS   #1, LIB$SIGNAL                                    ;
                      50        55  D0 000BB 7$:    MOVL    R5, DESC_ADR_LCL                                  ; 1187
      51              56   18   A0  C5 000BE        MULL3   24(DESC_ADR_LCL), R6, R1                          ;
                      51        54  C0 000C3        ADDL2   DIM2, R1                                          ;
                           04 B041  D5 000C6        TSTL    @4(DESC_ADR_LCL)[R1]                              ;
                                0C  14 000CA        BGTR    8$                                                ;
            10        AE   2020 8F  3C 000CC        MOVZWL  #8224, BUFFER                                     ; 1191
            08        AE        02  B0 000D2        MOVW    #2, DESC                                          ; 1192
                                2A  11 000D6        BRB     9$                                                ; 1187
                      50        55  D0 000D8 8$:    MOVL    R5, DESC_ADR_LCL                                  ; 1198
      51              56   18   A0  C5 000DB        MULL3   24(DESC_ADR_LCL), R6, R1                          ;
                      51        54  C0 000E0        ADDL2   DIM2, R1                                          ;
                           04 B041  DF 000E3        PUSHAL  @4(DESC_ADR_LCL)[R1]                              ;
                           0C   AE  9F 000E7        PUSHAB  DESC                                              ;
                           10   AE  9F 000EA        PUSHAB  DESC                                              ;
                      00000000' 00  9F 000ED        PUSHAB  CTRSTR_DESC                                       ;
         00000000G    00        04  FB 000F3        CALLS   #4, SYS$FAOL                                      ;
                      05        50  E8 000FA        BLBS    RTN_STATUS, 9$                                    ;
                                50  DD 000FD        PUSHL   RTN_STATUS                                        ;
                      67        01  FB 000FF        CALLS   #1, LIB$SIGNAL                                    ;
                      50        55  D0 00102 9$:    MOVL    R5, DESC_ADR_LCL                                  ; 1203
      51              56   18   A0  C5 00105        MULL3   24(DESC_ADR_LCL), R6, R1                          ;
                      51        54  C0 0010A        ADDL2   DIM2, R1                                          ;
                      0A   04 B041  D1 0010D        CMPL    @4(DESC_ADR_LCL)[R1], #10                         ;
                                11  18 00112        BGEQ    10$                                               ;
                      50        55  D0 00114        MOVL    R5, DESC_ADR_LCL                                  ; 1205
```

```
              51          56    18    A0  C5 00117        MULL3   24(DESC_ADR_LCL), R6, R1                       ;
                          51          54  C0 0011C        ADDL2   DIM2, R1                                       ;
                               04 B041  D5 0011F          TSTL    @4(DESC_ADR_LCL)[R1]                           ;
                          06          14 00123            BGTR    11$                                            ;
                    04    AE          01  D0 00125 10$:   MOVL    #1, ROW_POS                                    ; 1207
                          12          11 00129            BRB     12$                                            ;
                          50          55  D0 0012B 11$:   MOVL    R5, DESC_ADR_LCL                               ; 1210
              51          56    18    A0  C5 0012E        MULL3   24(DESC_ADR_LCL), R6, R1                       ;
                          51          54  C0 00133        ADDL2   DIM2, R1                                       ;
         04   AE          0B   04 B041  C3 00136          SUBL3   @4(DESC_ADR_LCL)[R1], #11, ROW_POS             ;
                                      5E  DD 0013D 12$:   PUSHL   SP                                             ; 1215
                                08    AE  9F 0013F        PUSHAB  ROW_POS                                        ;
                                10    AE  9F 00142        PUSHAB  DESC                                           ;
                          68          03  FB 00145        CALLS   #3, LIB$PUT_SCREEN                             ;
                          05          50  E8 00148        BLBS    RTN_STATUS, 13$                                ;
                                      50  DD 0014B        PUSHL   RTN_STATUS                                     ;
                          67          01  FB 0014D        CALLS   #1, LIB$SIGNAL                                 ;
      FED8                54          01    28    A5  F1 00150 13$:   ACBL    40(R5), #1, DIM2, 2$               ; 1150
                                      04 00157            RET                                                    ; 1217

; Routine Size:  344 bytes,    Routine Base:  $CODE$ + 024C
```

```
%SBTTL 'Create a surface graph'
ROUTINE PLOT_SURFACE_GRAPH( GRAPH_TYPE, XY_ARRAY_DESC, Y_HIGH, Y_LOW, Y_INCR,
                            Y_LABEL_DESC, NEW_SURFACE_GRAPH, SHADE_ARRAY_DESC )

    : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       PLOT_SURFACE_GRAPH produces a "table" from the xy-array.  For each
!       row in the array, each value in the row is printed.  If the terminal
!       understands REGIS, then shade the background.
!
! FORMAL PARAMETERS:
!
!       GRAPH_TYPE        :  type of table being produced (increasing or
!                            decreasing values
!       XY_ARRAY_DESC     :  address of xy-array descriptor
!       Y_HIGH            :  high value of y-axis numeric label
!       Y_LOW             :  low value of numeric label
!       Y_INCR            :  increment of numeric labels
!       NEW_SURFACE_GRAPH :   set if complete new graph will be written
!       SHADE_ARRAY_DESC  :  Descriptor for array containing shading info
!
! IMPLICIT INPUTS:
!
!       Y_AXIS_LINE  :  column which contains the y-axis line.  This value
!                       is considered for non-REGIS terminals only.
!       DEVICE_FLAGS :  a longword of flags which are set or cleared by
!                       LIB$SCREEN_INFO depending on the terminal
!                       characteristics.
!
! IMPLICIT OUTPUTS:
!
!       Y_AXIS_LINE  :  column which contains the y-axis line.  This value
!                       is only output for REGIS terminals.
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

BEGIN

MAP
    XY_ARRAY_DESC : REF BBLOCK; !Descriptor for the data array

```

```
2 LOCAL
2       XY_ARRAY          :                    !Array to hold the data to be plotted
2               REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
2       SEPARATOR_CHAR    : WORD,              !Character for visually separating bands of values
2       RTN_STATUS,                            !Status returned from external calls
2       MAX_ROWS,                              !Number of rows in the surface graph
2       DIM1,                                  !Index for first dimension of xy-array
2       DIM2,                                  !Index for second dimension of xy-array
2       ROW_LABEL,                             !Label for y-axis value
2       CURRENT_VALUE,                         !Current value in row, for comparisons
2       REPEAT_COUNT,                          !Number of values equal to current value
2       ROW_POS,                               !Current row on graph
2       COLUMN_POS                             !Column postion for output
2       ;
```

```
2       !Determine the number of rows in the graph.
2       !
3       IF SURFACE_GRAPH_LEN LEQ ((.Y_HIGH - .Y_LOW) / .Y_INCR)
2       THEN
2           MAX_ROWS = SURFACE_GRAPH_LEN - 1
2       ELSE
2           MAX_ROWS = ((.Y_HIGH - .Y_LOW) / .Y_INCR);
2
2
2       IF .DEVICE_FLAGS[ SCR$V_REGIS ]
2       THEN
3           BEGIN
3
3           !Choose separator character.  Set up colors for this graph.
3           !
3           SEPARATOR_CHAR = NULL;
3           PUT_REGIS( COLOR_SET_RYG );
3           END
2
2       ELSE
2
2           !Choose the appropriate separator character
2           !
2           IF .GRAPH_TYPE EQL GRF$_SRF_DECREASING
2           THEN
2               SEPARATOR_CHAR = '\'
2           ELSE
2               SEPARATOR_CHAR = '/';
2
2
2       IF .NEW_SURFACE_GRAPH
2       THEN
3           BEGIN
3
3           !Erase the screen to give the graph a "clean slate".
3           !
3           ERASE_PAGE();
3
3
3           !Draw and label the y-axis
3           !
3           DRAW_Y_AXIS( .MAX_ROWS+1, .GRAPH_TYPE, .Y_HIGH, .Y_INCR );
3           LABEL_Y_AXIS( .Y_LABEL_DESC, .MAX_ROWS+1 );
3
3
3           !Draw and label the x-axis
3           !
3           DRAW_X_AXIS( .MAX_ROWS+1 );
2           END;
2
2
2       !For each row:
2       !
2       INCR DIM1 FROM 0 TO  .MAX_ROWS
2       DO
3           BEGIN
3
```

```
        !Reinitialize values for this row. If terminal understands regis,
        !move the cursor to the beginning of the this row.  Shade the row
        !according to the information passed in the shade array.
        !
        ROW_POS = .MAX_ROWS - .DIM1 + 1;
        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
        THEN
            BEGIN
            MOVE_CURSOR_REGIS( GRF$_REGIS_POS, X_ORIGIN,
                               (.ROW_POS - 1) * UNIT_HEIGHT,
                               NO_VALUE, NO_VALUE );
            SHADE_ROW_REGIS( .DIM1, (.ROW_POS * UNIT_HEIGHT), .SHADE_ARRAY_DESC);
            MOVE_CURSOR_REGIS( GRF$_REGIS_POS, X_ORIGIN,
                               (.ROW_POS - 1) * UNIT_HEIGHT + 1,
                               NO_VALUE, NO_VALUE );
            END;



        !For each element in the row:
        !
        DIM2 = 0;
        WHILE .DIM2 LEQ .XY_ARRAY_DESC[ DSC$L_U2 ]
        DO
            BEGIN

            !Save the current xy-value for comparisons; determine the position
            !in the row.
            !
            CURRENT_VALUE = .XY_ARRAY[ .DIM1, .DIM2, .XY_ARRAY_DESC ];
            COLUMN_POS    = .DIM2;

            !Compare the current value to all subsequent values in the row
            !until a value which is not equal is encountered.  Keep a count
            !of the equal values, so the right number can be printed in the
            !graph.
            !
            DIM2 = .DIM2 + 1;
            WHILE .DIM2 LEQ .XY_ARRAY_DESC[ DSC$L_U2 ]
                           AND
                  (.CURRENT_VALUE EQL .XY_ARRAY[ .DIM1, .DIM2, .XY_ARRAY_DESC ]
                           OR
                   .XY_ARRAY[ .DIM1, .DIM2, .XY_ARRAY_DESC] LEQ 0 )
            DO
                DIM2 = .DIM2 + 1;

            !Calculate the number of equal values and write them out.
            !
            REPEAT_COUNT = .DIM2 - .COLUMN_POS;
            PUT_ROW_SEGMENT( .CURRENT_VALUE, .REPEAT_COUNT, .COLUMN_POS,
                             .ROW_POS,      .SEPARATOR_CHAR, .DIM1,
                             .SHADE_ARRAY_DESC);

            END;                                    !DIM2
        END;                                        !ROW_COUNT
```

```
; 1124      1403  2
; 1125      1404  2
; 1126      1405  1 END;                                ! End of routine PLOT_SURFACE_GRAPH


                                               0FFC 00000 PLOT_SURFACE_GRAPH:
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      ; 1219
   50        0C    AC           10   AC  C3 00002             SUBL3   Y_LOW, Y_HIGH, R0                         ; 1291
                   50           14   AC  C6 00008             DIVL2   Y_INCR, R0                                ;
                   13                50  D1 0000C             CMPL    R0, #19                                   ;
                                     05  19 0000F             BLSS    1$                                        ;
                   53                12  D0 00011             MOVL    #18, MAX_ROWS                             ; 1293
                                     03  11 00014             BRB     2$                                        ;
                   53                50  D0 00016 1$:         MOVL    R0, MAX_ROWS                              ; 1295
   1D 00000000'    00                02  E1 00019 2$:         BBC     #2, DEVICE_FLAGS, 3$                      ; 1298
                                     5B  B4 00021             CLRW    SEPARATOR_CHAR                            ; 1304
                         00000000'   00  9F 00023             PUSHAB  COLOR_SET_RYG                             ; 1305
      00000000G    00                01  FB 00029             CALLS   #1, LIB$PUT_OUTPUT                        ;
                   1A                50  E8 00030             BLBS    RTN_STATUS, 5$                            ;
                                     50  DD 00033             PUSHL   RTN_STATUS                                ;
      00000000G    00                01  FB 00035             CALLS   #1, LIB$SIGNAL                            ;
                                     0F  11 0003C             BRB     5$                                        ; 1298
                   02           04   AC  D1 0003E 3$:         CMPL    GRAPH_TYPE, #2                            ; 1312
                                     06  12 00042             BNEQ    4$                                        ;
                   5B           5C   8F  9B 00044             MOVZBW  #92, SEPARATOR_CHAR                       ; 1314
                                     03  11 00048             BRB     5$                                        ;
                   5B                2F  B0 0004A 4$:         MOVW    #47, SEPARATOR_CHAR                       ; 1316
                   31           1C   AC  E9 0004D 5$:         BLBC    NEW_SURFACE_GRAPH, 6$                     ; 1319
      00000000V    00                00  FB 00051             CALLS   #0, ERASE_PAGE                            ; 1325
                                14   AC  DD 00058             PUSHL   Y_INCR                                    ; 1330
                                0C   AC  DD 0005B             PUSHL   Y_HIGH                                    ;
                                04   AC  DD 0005E             PUSHL   GRAPH_TYPE                                ;
                                01   A3  9F 00061             PUSHAB  1(MAX_ROWS)                               ;
      00000000V    00                04  FB 00064             CALLS   #4, DRAW_Y_AXIS                           ;
                                01   A3  9F 0006B             PUSHAB  1(MAX_ROWS)                               ; 1331
                                18   AC  DD 0006E             PUSHL   Y_LABEL_DESC                              ;
      00000000V    00                02  FB 00071             CALLS   #2, LABEL_Y_AXIS                          ;
                                01   A3  9F 00078             PUSHAB  1(MAX_ROWS)                               ; 1336
      00000000V    00                01  FB 0007B             CALLS   #1, DRAW_X_AXIS                           ;
                   56           08   AC  D0 00082 6$:         MOVL    XY_ARRAY_DESC, R6                         ; 1368
                   52                01  CE 00086             MNEGL   #1, DIM1                                  ; 1398
                                   00B4  31 00089             BRW     12$                                       ;
   55              53                52  C3 0008C 7$:         SUBL3   DIM1, MAX_ROWS, R5                        ; 1350
                   54           01   A5  9E 00090             MOVAB   1(R5), ROW_POS                            ;
   42 00000000'    00                02  E1 00094             BBC     #2, DEVICE_FLAGS, 8$                      ; 1351
                   7E                01  CE 0009C             MNEGL   #1, -(SP)                                 ; 1354
                   7E                01  CE 0009F             MNEGL   #1, -(SP)                                 ;
                   55           FF   A4  9E 000A2             MOVAB   -1(R4), R5                                ; 1355
                   55                12  C4 000A6             MULL2   #18, R5                                   ;
                                     55  DD 000A9             PUSHL   R5                                        ;
                   7E           BF   8F  9A 000AB             MOVZBL  #191, -(SP)                               ; 1354
                                     7E  D4 000AF             CLRL    -(SP)                                     ;
      00000000V    00                05  FB 000B1             CALLS   #5, MOVE_CURSOR_REGIS                     ;
                                20   AC  DD 000B8             PUSHL   SHADE_ARRAY_DESC                          ; 1357
```

```
                7E          54         12  C5 000BB        MULL3   #18, ROW_POS, -(SP)                           ;
                                       52  DD 000BF        PUSHL   DIM1                                          ;
                00000000V   00         03  FB 000C1        CALLS   #3, SHADE_ROW_REGIS                           ;
                            7E         01  CE 000C8        MNEGL   #1, -(SP)                                     ; 1358
                            7E         01  CE 000CB        MNEGL   #1, -(SP)                                     ;
                                  01   A5  9F 000CE        PUSHAB  1(R5)                                         ; 1359
                            7E    BF   8F  9A 000D1        MOVZBL  #191, -(SP)                                   ; 1358
                                       7E  D4 000D5        CLRL    -(SP)                                         ;
                00000000V   00         05  FB 000D7        CALLS   #5, MOVE_CURSOR_REGIS                         ;
                                       57  D4 000DE  8$:   CLRL    DIM2                                          ; 1367
                       28   A6         57  D1 000E0  9$:   CMPL    DIM2, 40(R6)                                  ; 1368
                                       5A  14 000E4        BGTR    12$                                           ;
                            50         56  D0 000E6        MOVL    R6, DESC_ADR_LCL                              ; 1375
                51          52    18   A0  C5 000E9        MULL3   24(DESC_ADR_LCL), DIM1, R1                    ;
                            51         57  C0 000EE        ADDL2   DIM2, R1                                      ;
                            59  04 B041    D0 000F1        MOVL    @4(DESC_ADR_LCL)[R1], CURRENT_VALUE           ;
                            58         57  D0 000F6        MOVL    DIM2, COLUMN_POS                              ; 1376
                                       57  D6 000F9  10$:  INCL    DIM2                                          ; 1384
                       28   A6         57  D1 000FB        CMPL    DIM2, 40(R6)                                  ; 1385
                                       23  14 000FF        BGTR    11$                                           ;
                            50         56  D0 00101        MOVL    R6, DESC_ADR_LCL                              ; 1387
                51          52    18   A0  C5 00104        MULL3   24(DESC_ADR_LCL), DIM1, R1                    ;
                            51         57  C0 00109        ADDL2   DIM2, R1                                      ;
                       04 B041         59  D1 0010C        CMPL    CURRENT_VALUE, @4(DESC_ADR_LCL)[R1]           ;
                                       E6  13 00111        BEQL    10$                                           ;
                            50         56  D0 00113        MOVL    R6, DESC_ADR_LCL                              ; 1389
                51          52    18   A0  C5 00116        MULL3   24(DESC_ADR_LCL), DIM1, R1                    ;
                            51         57  C0 0011B        ADDL2   DIM2, R1                                      ;
                                04 B041    D5 0011E        TSTL    @4(DESC_ADR_LCL)[R1]                          ;
                                       D5  15 00122        BLEQ    10$                                           ;
                5A          57         58  C3 00124  11$:  SUBL3   COLUMN_POS, DIM2, REPEAT_COUNT                ; 1396
                                  20   AC  DD 00128        PUSHL   SHADE_ARRAY_DESC                              ; 1399
                                       52  DD 0012B        PUSHL   DIM1                                          ; 1398
                            7E         5B  3C 0012D        MOVZWL  SEPARATOR_CHAR, -(SP)                         ;
                                       54  DD 00130        PUSHL   ROW_POS                                       ;
                                       58  DD 00132        PUSHL   COLUMN_POS                                    ; 1397
                            7E         59  7D 00134        MOVQ    CURRENT_VALUE, -(SP)                          ;
                00000000V   00         07  FB 00137        CALLS   #7, PUT_ROW_SEGMENT                           ;
                                       A0  11 0013E        BRB     9$                                            ; 1368
      FF46      52          01         53  F1 00140  12$:  ACBL    MAX_ROWS, #1, DIM1, 7$                        ; 1342
                                           04 00146        RET                                                   ; 1405

; Routine Size:  327 bytes,    Routine Base:  $CODE$ + 03A4
```

```
1 %SBTTL 'Erase the screen or page'
1 ROUTINE ERASE_PAGE : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Starting in the first row and column erase the page.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2     RTN_STATUS,                     !Status returned from RTL call
2     ROW_POS,                        !Row to start erasing from
2     COLUMN_POS                      !Column to start erasing from
2     ;
```

```
; 1171          1448  2       IF .DEVICE_FLAGS[ SCR$V_REGIS ]
; 1172          1449  2       THEN
; 1173          1450  2
; 1174          1451  2           !Erase screen using REGIS command.
; 1175          1452  2           !
; 1176          1453  3           PUT_REGIS( %ASCID 'S(e)' )
; 1177          1454  3
; 1178          1455  2       ELSE
; 1179          1456  3           BEGIN
; 1180          1457  3
; 1181          1458  3           !Erase the screen to give the graph a "clean slate".
; 1182          1459  3           !
; 1183          1460  3           ROW_POS    = FIRST_ROW;
; 1184          1461  3           COLUMN_POS = FIRST_COLUMN;
; 1185          1462  4           IF NOT ( RTN_STATUS = LIB$ERASE_PAGE( ROW_POS, COLUMN_POS ))
; 1186          1463  3               THEN
; 1187          1464  3
; 1188          1465  3               !An error has occured while erasing the page.
; 1189          1466  3               !
; 1190          1467  3               SIGNAL( .RTN_STATUS );
; 1191          1468  3
; 1192          1469  2           END;
; 1193          1470  2
; 1194          1471  2       RETURN;
; 1195          1472  1   END;                                        ! End of routine ERASE_PAGE


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                   29  65  28  53  00168 P.AAN:    .ASCII  \S(e)\                                           ;
                         010E0004  0016C P.AAM:    .LONG   17694724                                         ;
                        00000000'  00170           .ADDRESS P.AAN                                           ;


                                                        .PSECT  $CODE$,NOWRT,2

                              0000 00000 ERASE_PAGE:
                                                        .WORD   Save nothing                                ; 1407
              5E                 08  C2 00002           SUBL2   #8, SP                                      ;
   0F 00000000'  00              02  E1 00005           BBC     #2, DEVICE_FLAGS, 1$                        ; 1448
                     00000000'   00  9F 0000D           PUSHAB  P.AAM                                       ; 1453
      00000000G  00              01  FB 00013           CALLS   #1, LIB$PUT_OUTPUT                          ;
                                 13  11 0001A           BRB     2$                                          ;
         04      AE              01  D0 0001C 1$:       MOVL    #1, ROW_POS                                 ; 1460
                 6E              01  D0 00020           MOVL    #1, COLUMN_POS                              ; 1461
                                 5E  DD 00023           PUSHL   SP                                          ; 1462
                          08     AE  9F 00025           PUSHAB  ROW_POS                                     ;
      00000000G  00              02  FB 00028           CALLS   #2, LIB$ERASE_PAGE                          ;
                 09              50  E8 0002F 2$:       BLBS    RTN_STATUS, 3$                              ;
                                 50  DD 00032           PUSHL   RTN_STATUS                                  ; 1467
      00000000G  00              01  FB 00034           CALLS   #1, LIB$SIGNAL                              ;
                                     04 0003B 3$:       RET                                                 ; 1472

; Routine Size:  60 bytes,    Routine Base:  $CODE$ + 04EB
```

```
; 1196          1473  1
```

```
%SBTTL 'Draw y-axis for surface graph'
ROUTINE DRAW_Y_AXIS ( GRAPH_LEN, GRAPH_TYPE, Y_HIGH, Y_INCR ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       DRAW_Y_AXIS draws the y-axis line and writes the numeric labels.
!
! FORMAL PARAMETERS:
!
!       GRAPH_LEN       :  number of rows in the graph
!       GRAPH_TYPE      :  type of graph being drawn
!       Y_HIGH          :  high value for y-axis numeric labels
!       Y_INCR          :  increment for y-axis numeric labels
!
! IMPLICIT INPUTS:
!
!       Y_AXIS_LINE  :  column which contains the y-axis line.
!       DEVICE_FLAGS :  a longword of flags which are set or cleared by
!                       LIB$SCREEN_INFO depending on the terminal
!                       characteristics.
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       Output to the terminal
!
!--

BEGIN

LOCAL
    ROW_POS,                            !Row where label is to be placed
    COLUMN_POS,                         !Column in which the label starts
    ROW_LABEL,                          !Numeric label for y-axis
    X_AXIS_LINE,                        !Location of x-axis line
    CTRSTR_DESC,                        !Address of control string
    BUFFER          :                   !Buffer for building the output
        VECTOR[ MAX_PAGE_WIDTH, BYTE ],
    DESC            :                   !Descriptor for BUFFER
        BBLOCK[ DSC$K_Z_BLN ]
    ;

BIND
    CTRSTR_DESC1 = %ASCID'!2UB',
```

```
; 1255        1531  2      CTRSTR_DESC2 = %ASCID'!UL'
; 1256        1532  2      ;
```

```
!Inititalize variables
!
(BUFFER)              = '    ';
DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
DESC[ DSC$A_POINTER ] = BUFFER;

IF .GRAPH_TYPE EQL GRF$_LINE
THEN
    CTRSTR_DESC = CTRSTR_DESC1
ELSE
    CTRSTR_DESC = CTRSTR_DESC2;


!If this is a regis terminal, draw the axis line
!
IF .DEVICE_FLAGS[ SCR$V_REGIS ]
THEN
    BEGIN

    !Position the cursor at the begining of the vector.
    !
    MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .Y_AXIS_LINE, FIRST_ROW,
                       NO_VALUE, NO_VALUE );

    !Draw a vector finishing at the the x-axis line.
    !
    X_AXIS_LINE = UNIT_HEIGHT * (.GRAPH_LEN ) + AXIS_SHIFT;
    MOVE_CURSOR_REGIS( GRF$_REGIS_VCTR, NO_VALUE, .X_AXIS_LINE,
                       NO_VALUE, NO_VALUE );
    END;


!For each row of the graph:
!
INCR ROW_COUNT FROM 0 TO .GRAPH_LEN - 1
DO
    BEGIN

    !Initialize variables for this row
    !
    DESC[ DSC$W_LENGTH ] = MAX_PAGE_WIDTH;
    ROW_POS = .ROW_COUNT + 1;

    IF .GRAPH_TYPE EQL GRF$_LINE
    THEN
        ROW_LABEL = .GRAPH_LEN - .ROW_COUNT
    ELSE
        ROW_LABEL = .Y_HIGH - (.Y_INCR * .ROW_COUNT);


    !If this is a line graph or an "even" row, put the numeric label along
    !the y-axis line.  This is for readability... too many numeric
    !labels make it hard to read the graph.
    !
    IF .GRAPH_TYPE EQL GRF$_LINE
           OR
```

```
        (.ROW_COUNT MOD 2) EQL 0
        THEN
            TRANSLATE_VALUE( .CTRSTR_DESC, DESC[ DSC$W_LENGTH ], DESC, ROW_LABEL )
        ELSE
            DESC[ DSC$W_LENGTH ] = 0;


        !If this is the first label for the line graph, an "*" is really
        !desired to denote that the numbers in this row are large. Replace
        !the numeric label with a "*".
        !
        IF .GRAPH_TYPE EQL GRF$_LINE AND .ROW_COUNT EQL 0
        THEN
            BEGIN
            (BUFFER)              = ' *';
            DESC[ DSC$W_LENGTH ] = SEPARATOR_WIDTH;
            END;


        !Determine column to start the string in and write it out.
        !
        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
        THEN
            BEGIN
            COLUMN_POS = .Y_AXIS_LINE - (.DESC[ DSC$W_LENGTH ]+1) * CHAR_WIDTH;
            MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS,
                               (.ROW_POS - 1) * UNIT_HEIGHT + 1,
                               NO_VALUE, NO_VALUE );
            PUT_REGIS_TEXT( DESC, LIGHT_INTENSITY );
            END

        ELSE
            BEGIN

            !Add the y-axis line delimiter to the output string.
            !
            BUFFER[ .DESC[ DSC$W_LENGTH ] ] = '|';
            DESC[ DSC$W_LENGTH ]            = .DESC[ DSC$W_LENGTH ] + 1;

            COLUMN_POS = .Y_AXIS_LINE - .DESC[ DSC$W_LENGTH ] + 1;
            PUT_TEXT( DESC, ROW_POS, COLUMN_POS );
            END;

        END;
  END;                                  ! End of routine DRAW_Y_AXIS


                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

        42  55  32  21   00174 P.AAP:   .ASCII   \!2UB\                  ;
              010E0004   00178 P.AAO:   .LONG    17694724                ;
             00000000'   0017C          .ADDRESS P.AAP                   ;
        00  4C  55  21   00180 P.AAR:   .ASCII   \!UL\<0>                ;
              010E0003   00184 P.AAQ:   .LONG    17694723                ;
             00000000'   00188          .ADDRESS P.AAR                   ;
```

```
                                              CTRSTR_DESC1=          P.AAO
                                              CTRSTR_DESC2=          P.AAQ

                                                         .PSECT  $CODE$,NOWRT,2

                                     01FC 00000 DRAW_Y_AXIS:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8                        ; 1475
                 58 00000000G  00  9E 00002              MOVAB   LIB$SIGNAL, R8                                   ;
                 57 00000000V  00  9E 00009              MOVAB   MOVE_CURSOR_REGIS, R7                            ;
                 56 00000000'  00  9E 00010              MOVAB   DEVICE_FLAGS, R6                                 ;
                 5E      FF68  CE  9E 00017              MOVAB   -152(SP), SP                                     ;
           14    AE 20202020   8F  D0 0001C              MOVL    #538976288, BUFFER                               ; 1535
                           0E  AE  B4 00024              CLRW    DESC+2                                           ; 1537
           10    AE        14  AE  9E 00027              MOVAB   BUFFER, DESC+4                                   ; 1538
                               55  D4 0002C              CLRL    R5                                               ; 1540
                       08      AC  D5 0002E              TSTL    GRAPH_TYPE                                       ;
                               0B  12 00031              BNEQ    1$                                               ;
                               55  D6 00033              INCL    R5                                               ;
                 54 00000000'  00  9E 00035              MOVAB   CTRSTR_DESC1, CTRSTR_DESC                        ; 1542
                               07  11 0003C              BRB     2$                                               ;
                 54 00000000'  00  9E 0003E 1$:          MOVAB   CTRSTR_DESC2, CTRSTR_DESC                        ; 1544
      28         66            02  E1 00045 2$:          BBC     #2, DEVICE_FLAGS, 3$                             ; 1549
                 7E            01  CE 00049              MNEGL   #1, -(SP)                                        ; 1555
                 7E            01  CE 0004C              MNEGL   #1, -(SP)                                        ;
                               01  DD 0004F              PUSHL   #1                                               ;
                       04      A6  DD 00051              PUSHL   Y_AXIS_LINE                                      ;
                               7E  D4 00054              CLRL    -(SP)                                            ;
                 67            05  FB 00056              CALLS   #5, MOVE_CURSOR_REGIS                            ;
      50         04  AC        12  C5 00059              MULL3   #18, GRAPH_LEN, R0                               ; 1560
                 50            05  C0 0005E              ADDL2   #5, X_AXIS_LINE                                  ;
                 7E            01  CE 00061              MNEGL   #1, -(SP)                                        ; 1561
                 7E            01  CE 00064              MNEGL   #1, -(SP)                                        ;
                               50  DD 00067              PUSHL   X_AXIS_LINE                                      ;
                 7E            01  CE 00069              MNEGL   #1, -(SP)                                        ;
                               01  DD 0006C              PUSHL   #1                                               ;
                 67            05  FB 0006E              CALLS   #5, MOVE_CURSOR_REGIS                            ;
                 53            01  CE 00071 3$:          MNEGL   #1, ROW_COUNT                                    ; 1568
                          00C9 31 00074              BRW     12$                                              ;
           0C    AE        84  8F  9B 00077 4$:          MOVZBW  #132, DESC                                       ; 1574
           08    AE        01  A3  9E 0007C              MOVAB   1(R3), ROW_POS                                   ; 1575
                 07            55  E9 00081              BLBC    R5, 5$                                           ; 1577
      6E         04  AC        53  C3 00084              SUBL3   ROW_COUNT, GRAPH_LEN, ROW_LABEL                  ; 1579
                               0A  11 00089              BRB     6$                                               ;
      50         10  AC        53  C5 0008B 5$:          MULL3   ROW_COUNT, Y_INCR, R0                            ; 1581
      6E         0C  AC        50  C3 00090              SUBL3   R0, Y_HIGH, ROW_LABEL                            ;
                 0E            55  E8 00095 6$:          BLBS    R5, 7$                                           ; 1588
 7E              00  53        01  7A 00098              EMUL    #1, ROW_COUNT, #0, -(SP)                         ; 1590
 50              50  8E        02  7B 0009D              EDIV    #2, (SP)+, R0, R0                                ;
                               50  D5 000A2              TSTL    R0                                               ;
                               1B  12 000A4              BNEQ    8$                                               ;
                               5E  DD 000A6 7$:          PUSHL   SP                                               ; 1592
                       10      AE  9F 000A8              PUSHAB  DESC                                             ;
                       14      AE  9F 000AB              PUSHAB  DESC                                             ;
                               54  DD 000AE              PUSHL   CTRSTR_DESC                                      ;
            00000000G  00      04  FB 000B0              CALLS   #4, SYS$FAOL                                     ;
                       0A      50  E8 000B7              BLBS    RTN_STATUS, 9$                                   ;
```

```
                                  50  DD 000BA        PUSHL   RTN_STATUS                          ;
               68                 01  FB 000BC        CALLS   #1, LIB$SIGNAL                      ;
                                  03  11 000BF        BRB     9$                                  ; 1588
                          0C      AE  B4 000C1 8$:    CLRW    DESC                                ; 1594
               0E                 55  E9 000C4 9$:    BLBC    R5, 10$                             ; 1601
                                  53  D5 000C7        TSTL    ROW_COUNT                           ;
                                  0A  12 000C9        BNEQ    10$                                 ;
          14   AE       2A20      8F  3C 000CB        MOVZWL  #10784, BUFFER                      ; 1604
          0C   AE                 02  B0 000D1        MOVW    #2, DESC                            ; 1605
               52         04      A6  D0 000D5 10$:   MOVL    Y_AXIS_LINE, R2                     ; 1614
               50         0C      AE  3C 000D9        MOVZWL  DESC, R0                            ;
     31        66                 02  E1 000DD        BBC     #2, DEVICE_FLAGS, 11$               ; 1611
     51        50                 09  C5 000E1        MULL3   #9, R0, R1                          ; 1614
     51        52                 51  C3 000E5        SUBL3   R1, R2, R1                          ;
          04   AE         F7      A1  9E 000E9        MOVAB   -9(R1), COLUMN_POS                  ;
               7E                 01  CE 000EE        MNEGL   #1, -(SP)                           ; 1615
               7E                 01  CE 000F1        MNEGL   #1, -(SP)                           ;
     51   10   AE                 12  C5 000F4        MULL3   #18, ROW_POS, R1                    ; 1616
                          EF      A1  9F 000F9        PUSHAB  -17(R1)                             ;
                          10      AE  DD 000FC        PUSHL   COLUMN_POS                          ; 1615
                                  7E  D4 000FF        CLRL    -(SP)                               ;
               67                 05  FB 00101        CALLS   #5, MOVE_CURSOR_REGIS               ;
                                  33  DD 00104        PUSHL   #51                                 ; 1618
                          10      AE  9F 00106        PUSHAB  DESC                                ;
     00000000V  00                02  FB 00109        CALLS   #2, PUT_REGIS_TEXT                  ;
                                  2E  11 00110        BRB     12$                                 ; 1611
          14 AE40         7C      8F  90 00112 11$:   MOVB    #124, BUFFER[R0]                    ; 1626
                          0C      AE  B6 00118        INCW    DESC                                ; 1627
               51         0C      AE  3C 0011B        MOVZWL  DESC, R1                            ; 1629
     51        52                 51  C3 0011F        SUBL3   R1, R2, R1                          ;
          04   AE         01      A1  9E 00123        MOVAB   1(R1), COLUMN_POS                   ;
                          04      AE  9F 00128        PUSHAB  COLUMN_POS                          ; 1630
                          0C      AE  9F 0012B        PUSHAB  ROW_POS                             ;
                          14      AE  9F 0012E        PUSHAB  DESC                                ;
     00000000G  00                03  FB 00131        CALLS   #3, LIB$PUT_SCREEN                  ;
                05                50  E8 00138        BLBS    RTN_STATUS, 12$                     ;
                                  50  DD 0013B        PUSHL   RTN_STATUS                          ;
               68                 01  FB 0013D        CALLS   #1, LIB$SIGNAL                      ;
     01        53         04      AC  F2 00140 12$:   AOBLSS  GRAPH_LEN, ROW_COUNT, 13$           ; 1568
                                  04  00145           RET                                         ; 1634
                            FF2E  31 00146 13$:       BRW     4$                                  ; 1568

; Routine Size:  329 bytes,    Routine Base:  $CODE$ + 0527
```

```
%SBTTL 'Label the y-axis for a surface graph'
ROUTINE LABEL_Y_AXIS( Y_LABEL_DESC, GRAPH_LEN ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Draw the y-axis and write the numeric labels.
!
! FORMAL PARAMETERS:
!
!       Y_LABEL_DESC  :  Address of descriptor for descriptive labels
!       GRAPH_LEN     :  The number of rows in the graph
!
! IMPLICIT INPUTS:
!
!       Y_AXIS_LINE  :  column which contains the y-axis line.
!       DEVICE_FLAGS :  a longword of flags which are set or cleared by
!                       LIB$SCREEN_INFO depending on the terminal
!                       characteristics.
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

BEGIN

LITERAL
    LABEL_WIDTH = 15
    ;

LOCAL
    ROW_POS,                            !Row where label is to be placed
    COLUMN_POS,                         !Column in which the label starts
    STRING_LEN,                         !Length of remaining string
    STRING_START,                       !Address of first character
    CURRENT_LEN,                        !Length of remaining string
    WORD_START,                         !Address of first character in word
    WORD_END,                           !Address of last character in word
    WORD_COUNT,                         !Number of words in label
    DESC            :                   !Descriptor for output buffer
        BBLOCK [ DSC$K_Z_BLN ]
    ;
```

```
2
2 MAP
2     Y_LABEL_DESC      :                    !Descriptor for y-axis labels
2         REF BBLOCK
2     ;
```

```
    !Initialize variables
    !
    DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
    DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
    STRING_LEN   = .Y_LABEL_DESC[ DSC$W_LENGTH ];
    CURRENT_LEN  = .STRING_LEN;
    STRING_START = .Y_LABEL_DESC[ DSC$A_POINTER ];
    WORD_START   = .STRING_START;
    WORD_END     = .STRING_START;
    WORD_COUNT   = 0;


    !Count the number of words in the label
    !
    UNTIL .CURRENT_LEN GTRU .STRING_LEN
    DO
        BEGIN

        !Find the start of the next word.
        !
        WORD_START  = CH$FIND_NOT_CH( .CURRENT_LEN, .WORD_END, ' ');
        CURRENT_LEN = .STRING_LEN - ( .WORD_START - .STRING_START );

        IF .WORD_START EQL 0
        THEN

            !There are no more words in the string. Finish processing.
            !
            EXITLOOP;

        !Find the end of the next word.
        !
        WORD_END    = CH$FIND_CH( .CURRENT_LEN, .WORD_START, ' ');
        CURRENT_LEN = .STRING_LEN - ( .WORD_END - .STRING_START );

        IF (.WORD_END - .WORD_START + 1) GTR LABEL_WIDTH
        THEN

            !Word is too longer than the maximum width allowed (LABEL_WIDTH)
            !
            SIGNAL( EDF$_INTSWERR, 1, INVALID_LABEL );

        !Count the new word.
        WORD_COUNT = .WORD_COUNT + 1;
        END;

      IF .WORD_COUNT GTR (.GRAPH_LEN / 2) OR .WORD_COUNT EQL 0
      THEN

        !There are more words than lines to write them on or there are none.
        !
        SIGNAL( EDF$_INTSWERR, 1, INVALID_LABEL );
```

```
!Put the label to the screen.  Determine in which row and column the
!first word of the label is to be positioned.
!
IF .DEVICE_FLAGS[ SCR$V_REGIS ]
THEN
    COLUMN_POS = .Y_AXIS_LINE - ( LABEL_WIDTH  * CHAR_WIDTH )
ELSE
    COLUMN_POS = .Y_AXIS_LINE - LABEL_WIDTH;

IF (.COLUMN_POS LSSU 1)
THEN
    COLUMN_POS = 1;


ROW_POS = (.GRAPH_LEN - .WORD_COUNT) / 2;
IF NOT ((.ROW_POS MOD 2) EQL 0)
THEN
    ROW_POS = .ROW_POS - 1;


!Extract next word from string and print it in the proper place.
!
CURRENT_LEN = .Y_LABEL_DESC[ DSC$W_LENGTH ];
WORD_START  = .STRING_START;
WORD_END    = .STRING_START;


DECR LOOP_INDEX FROM .WORD_COUNT TO 1 DO
    BEGIN

    !Find the start of the next word.
    !
    WORD_START  = CH$FIND_NOT_CH( .CURRENT_LEN, .WORD_END, ' ');
    CURRENT_LEN = .STRING_LEN - ( .WORD_START - .STRING_START );


    !Find the end of the next word.
    !
    WORD_END    = CH$FIND_CH( .CURRENT_LEN, .WORD_START, ' ');
    CURRENT_LEN = .STRING_LEN - ( .WORD_END - .STRING_START );

    IF .WORD_END EQL 0
    THEN
        WORD_END = .STRING_START + .Y_LABEL_DESC[ DSC$A_POINTER ];


    !Adjust Output buffer descriptor to point to the current word.
    !
    DESC[ DSC$W_LENGTH ]  = .WORD_END - .WORD_START;
    DESC[ DSC$A_POINTER ] = .WORD_START;


    !Write this part of the label on the graph and decide where next
    !word should be written.
    !
```

```
; 1538     1811  3            IF .DEVICE_FLAGS[ SCR$V_REGIS ]
; 1539     1812  3            THEN
; 1540     1813  4                BEGIN
; 1541     1814  4                MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS,
; 1542     1815  4                                   (.ROW_POS - 1) * UNIT_HEIGHT,
; 1543     1816  4                                   NO_VALUE, NO_VALUE );
; 1544     1817  4                PUT_REGIS_TEXT( DESC, LIGHT_INTENSITY );
; 1545     1818  4                END
; 1546     1819  4
; 1547     1820  3            ELSE
; 1548     1821  3                PUT_TEXT( DESC, ROW_POS, COLUMN_POS );
; 1549     1822  3
; 1550     1823  3
; 1551     1824  3            ROW_POS = .ROW_POS + 2;
; 1552     1825  2            END;                                  !Print loop
; 1553     1826  1 END;                                          ! End of routine LABEL_Y_AXIS


                                    03FC 00000 LABEL_Y_AXIS:
                                                      .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9                 ; 1636
                 59 00000000'   00  9E 00002          MOVAB    DEVICE_FLAGS, R9                             ;
                 5E             10  C2 00009          SUBL2    #16, SP                                      ;
                        0A      AE  B4 0000C          CLRW     DESC+2                                       ; 1700
                 54     04      AC  D0 0000F          MOVL     Y_LABEL_DESC, R4                             ; 1701
                 58             64  3C 00013          MOVZWL   (R4), STRING_LEN                             ;
                 55             58  D0 00016          MOVL     STRING_LEN, CURRENT_LEN                      ; 1702
                 52     04      A4  D0 00019          MOVL     4(R4), STRING_START                          ; 1703
                 57             52  D0 0001D          MOVL     STRING_START, WORD_START                     ; 1704
                 56             52  D0 00020          MOVL     STRING_START, WORD_END                       ; 1705
                                53  D4 00023          CLRL     WORD_COUNT                                   ; 1706
                 58             55  D1 00025 1$:      CMPL     CURRENT_LEN, STRING_LEN                      ; 1711
                                2E  1A 00028          BGTRU    4$                                           ;
          66     55             20  3B 0002A          SKPC     #32, CURRENT_LEN, (WORD_END)                 ; 1717
                                02  12 0002E          BNEQ     2$                                           ;
                                51  D4 00030          CLRL     R1                                           ;
                 57             51  D0 00032 2$:      MOVL     R1, WORD_START                               ;
          50     52             57  C3 00035          SUBL3    WORD_START, STRING_START, R0                 ; 1718
          55     50             58  C1 00039          ADDL3    STRING_LEN, R0, CURRENT_LEN                  ;
                                57  D5 0003D          TSTL     WORD_START                                   ; 1721
                                17  13 0003F          BEQL     4$                                           ;
          67     55             20  3A 00041          LOCC     #32, CURRENT_LEN, (WORD_START)               ; 1730
                                02  12 00045          BNEQ     3$                                           ;
                                51  D4 00047          CLRL     R1                                           ;
                 56             51  D0 00049 3$:      MOVL     R1, WORD_END                                 ;
          50     52             56  C3 0004C          SUBL3    WORD_END, STRING_START, R0                   ; 1731
          55     50             58  C1 00050          ADDL3    STRING_LEN, R0, CURRENT_LEN                  ;
                                53  D6 00054          INCL     WORD_COUNT                                   ; 1743
                                CD  11 00056          BRB      1$                                           ; 1711
                 50     04      A9  D0 00058 4$:      MOVL     Y_AXIS_LINE, R0                              ; 1760
          07     69             02  E1 0005C          BBC      #2, DEVICE_FLAGS, 5$                         ; 1758
                 6E     FF79    C0  9E 00060          MOVAB    -135(R0), COLUMN_POS                         ; 1760
                                04  11 00065          BRB      6$                                           ;
                 6E     F1      A0  9E 00067 5$:      MOVAB    -15(R0), COLUMN_POS                          ; 1762
                                03  12 0006B 6$:      BNEQ     7$                                           ; 1765
```

```
                               6E        01  D0 0006D        MOVL    #1, COLUMN_POS                                 ; 1767
              50         08    AC        53  C3 00070  7$:   SUBL3   WORD_COUNT, GRAPH_LEN, R0                      ; 1770
         04   AE               50        02  C7 00075        DIVL3   #2, R0, ROW_POS                                ;
  7E          00         04    AE        01  7A 0007A        EMUL    #1, ROW_POS, #0, -(SP)                         ; 1771
  50          50               8E        02  7B 00080        EDIV    #2, (SP)+, R0, R0                              ;
                                         50  D5 00085        TSTL    R0                                             ;
                                         03  13 00087        BEQL    8$                                             ;
                                    04   AE  D7 00089        DECL    ROW_POS                                        ; 1773
                               55        64  3C 0008C  8$:   MOVZWL  (R4), CURRENT_LEN                              ; 1778
                               57        52  D0 0008F        MOVL    STRING_START, WORD_START                       ; 1779
                               56        52  D0 00092        MOVL    STRING_START, WORD_END                         ; 1780
                                         53  D6 00095        INCL    LOOP_INDEX                                     ; 1783
                                       0084  31 00097        BRW     15$                                            ;
              66               55        20  3B 0009A  9$:   SKPC    #32, CURRENT_LEN, (WORD_END)                   ; 1788
                                         02  12 0009E        BNEQ    10$                                            ;
                                         51  D4 000A0        CLRL    R1                                             ;
                               57        51  D0 000A2  10$:  MOVL    R1, WORD_START                                 ;
              50               52        57  C3 000A5        SUBL3   WORD_START, STRING_START, R0                   ; 1789
              55               50        58  C1 000A9        ADDL3   STRING_LEN, R0, CURRENT_LEN                    ;
              67               55        20  3A 000AD        LOCC    #32, CURRENT_LEN, (WORD_START)                 ; 1794
                                         02  12 000B1        BNEQ    11$                                            ;
                                         51  D4 000B3        CLRL    R1                                             ;
                               56        51  D0 000B5  11$:  MOVL    R1, WORD_END                                   ;
              50               52        56  C3 000B8        SUBL3   WORD_END, STRING_START, R0                     ; 1795
              55               50        58  C1 000BC        ADDL3   STRING_LEN, R0, CURRENT_LEN                    ;
                                         56  D5 000C0        TSTL    WORD_END                                       ; 1797
                                         05  12 000C2        BNEQ    12$                                            ;
              56               52   04   A4  C1 000C4        ADDL3   4(R4), STRING_START, WORD_END                  ; 1799
         08   AE               56        57  A3 000C9  12$:  SUBW3   WORD_START, WORD_END, DESC                     ; 1804
                         0C    AE        57  D0 000CE        MOVL    WORD_START, DESC+4                             ; 1805
              29               69        02  E1 000D2        BBC     #2, DEVICE_FLAGS, 13$                          ; 1811
                               7E        01  CE 000D6        MNEGL   #1, -(SP)                                      ; 1814
                               7E        01  CE 000D9        MNEGL   #1, -(SP)                                      ;
              50         0C    AE        01  C3 000DC        SUBL3   #1, ROW_POS, R0                                ; 1815
              7E               50        12  C5 000E1        MULL3   #18, R0, -(SP)                                 ;
                                    0C   AE  DD 000E5        PUSHL   COLUMN_POS                                     ; 1814
                                         7E  D4 000E8        CLRL    -(SP)                                          ;
                   00000000V   00        05  FB 000EA        CALLS   #5, MOVE_CURSOR_REGIS                          ;
                                         33  DD 000F1        PUSHL   #51                                            ; 1817
                                    0C   AE  9F 000F3        PUSHAB  DESC                                           ;
                   00000000V   00        02  FB 000F6        CALLS   #2, PUT_REGIS_TEXT                             ;
                                         1B  11 000FD        BRB     14$                                            ; 1811
                                         5E  DD 000FF  13$:  PUSHL   SP                                             ; 1821
                                    08   AE  9F 00101        PUSHAB  ROW_POS                                        ;
                                    10   AE  9F 00104        PUSHAB  DESC                                           ;
                   00000000G   00        03  FB 00107        CALLS   #3, LIB$PUT_SCREEN                             ;
                               09        50  E8 0010E        BLBS    RTN_STATUS, 14$                                ;
                                         50  DD 00111        PUSHL   RTN_STATUS                                     ;
                   00000000G   00        01  FB 00113        CALLS   #1, LIB$SIGNAL                                 ;
                          04   AE        02  C0 0011A  14$:  ADDL2   #2, ROW_POS                                    ; 1824
                               01        53  F5 0011E  15$:  SOBGTR  LOOP_INDEX, 16$                                ; 1783
                                         04     00121        RET                                                    ; 1826
                                       FF75  31 00122  16$:  BRW     9$                                             ; 1783

; Routine Size:  293 bytes,    Routine Base:  $CODE$ + 0670
```

```
%SBTTL 'Write a portion of a graph'
ROUTINE PUT_ROW_SEGMENT( CURRENT_VALUE, REPEAT_COUNT,   COLUMN_POS,
                         ROW_POS,       SEPARATOR_CHAR, DIM1,
                         SHADE_ARRAY_DESC ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       PUT_ROW_SEGMENT places a requested number of repetions of the current
!       value in a buffer and put the buffer to the terminal in the proper
!       place.
!
! FORMAL PARAMETERS:
!
!       CURRENT_VALUE    : Current value to be printed in the row segment
!       REPEAT_COUNT     : Number of times the value is to be printed
!       COLUMN_POS       : Column in which row segment starts
!       ROW_POS          : Row in which row segment is to be written
!       SEPARATOR_CHAR   : Character used to delimit ''good'' section
!       DIM1             : Index into SHADE_ARRAY
!       SHADE_ARRAY_DESC: Descriptor for the array of shading information
!
! IMPLICIT INPUTS:
!
!       DEVICE_FLAGS :  a longword of flags which are set or cleared by
!                       LIB$SCREEN_INFO depending on the terminal
!                       characteristics.
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       Output to the terminal.
!
!--

BEGIN

LITERAL
    TWO_DIGITS = 10             !First two digit number
    ;

LOCAL
    SHADE_ARRAY         :       !Array containing shading values
        REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
    BUFFER              :       !Buffer for building the output
```

```
2          VECTOR[ MAX_PAGE_WIDTH, BYTE ],
2       DESC                :       !Descriptor for BUFFER
2          BBLOCK[ DSC$K_Z_BLN ],
2       REGION_FOUND        :  BYTE  !Flag -- set if we need to delimit "good" area
2       ;
2
2  BIND
2       CTRSTR_DESC = %ASCID'!2UB'
2       ;
```

```
!Inititalize variables
!
(BUFFER)                  = '    ';
DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
DESC[ DSC$A_POINTER ] = BUFFER;
DESC[ DSC$W_LENGTH  ] = SEPARATOR_WIDTH;
REGION_FOUND          = FALSE;


!If this is an invalid value, then use blanks to fill the row segment.
!Otherwise, translate the value into ASCII.
!
IF .CURRENT_VALUE LEQ 0
THEN
    (BUFFER) = '  '
ELSE
    TRANSLATE_VALUE( CTRSTR_DESC, DESC[ DSC$W_LENGTH ], DESC,
                     CURRENT_VALUE );


!If the start of this row segement is also the start of a "good" section,
!then flag it so that the end of the "good" section can be looked for.
!
IF .SEPARATOR_CHAR NEQ NULL
            AND
    .CURRENT_VALUE LSS TWO_DIGITS
            AND
    .SHADE_ARRAY[.DIM1, .COLUMN_POS, .SHADE_ARRAY_DESC]
    EQL ([IGHT_INTENSITY - '0')
THEN
    REGION_FOUND = TRUE;


!For each repition of the value:
!
INCR LOOP_COUNT FROM 1 TO (.REPEAT_COUNT - 1)
DO
    BEGIN

    !Copy the ASCII value into the next available position in the buffer.
    !
    CH$MOVE( SEPARATOR_WIDTH, BUFFER,
             (BUFFER + (.LOOP_COUNT * SEPARATOR_WIDTH)));
    DESC[ DSC$W_LENGTH ] = .DESC[ DSC$W_LENGTH ] + SEPARATOR_WIDTH;


    !If the end of the "good" area is encountered, put in a delimiter.
    !
    IF .REGION_FOUND
            AND
       .SHADE_ARRAY[.DIM1, (.COLUMN_POS + .LOOP_COUNT), .SHADE_ARRAY_DESC]
      NEQ (LIGHT_INTENSITY - '0')
    THEN
        BEGIN
        BUFFER[ .DESC[ DSC$W_LENGTH ] - 2 ] = .SEPARATOR_CHAR;
        REGION_FOUND = FALSE;
```

```
                END;

            END;

        !If this is the beginning or the end of the "good" area of the graph,
        !place a visual separator in the front of the buffer.  If separator is
        !more than one byte in length, this won't work -- it only moves one
        !byte into the buffer at the specified position.
        !
        IF .SEPARATOR_CHAR NEQ NULL
                AND
           .CURRENT_VALUE LSS TWO_DIGITS
                AND
           (.SHADE_ARRAY[.DIM1, .COLUMN_POS, .SHADE_ARRAY_DESC]
            EQL (LIGHT_INTENSITY - '0')
                OR
           .SHADE_ARRAY[.DIM1, .COLUMN_POS - 1, .SHADE_ARRAY_DESC]
            EQL (LIGHT_INTENSITY - '0'))
        THEN
            BUFFER[ 0 ] = .SEPARATOR_CHAR;


        !If the end of the "good" region has not been encountered, then assume
        !the region ends with the row segment. Put the separator at the end.
        !
        IF .REGION_FOUND
        THEN
            BEGIN
            BUFFER[ .DESC[DSC$W_LENGTH] ] = .SEPARATOR_CHAR;
            DESC[ DSC$W_LENGTH ] = .DESC[ DSC$W_LENGTH ] + 1;
            END;


        !Put the buffer out to the terminal.
        !
        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
        THEN
            PUT_REGIS_TEXT( DESC, BACKGROUND_INTENSITY )
        ELSE
            BEGIN
            COLUMN_POS = (.COLUMN_POS + 1) * SEPARATOR_WIDTH + .Y_AXIS_LINE;
            PUT_TEXT( DESC, ROW_POS, COLUMN_POS );
            END;

        RETURN;
    END;                                        ! End of routine PUT_ROW_SEGMENT


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                      42  55  32  21   0018C P.AAT:   .ASCII  \!2UB\
                           010E0004    00190 P.AAS:   .LONG   17694724
                           00000000'   00194          .ADDRESS P.AAT

                                         CTRSTR_DESC=        P.AAS
```

```
                                              .PSECT  $CODE$,NOWRT,2

                              00FC 00000 PUT_ROW_SEGMENT:
                                              .WORD   Save R2,R3,R4,R5,R6,R7                    : 1828
          57 00000000G   00   9E 00002        MOVAB   LIB$SIGNAL, R7                            :
          5E      FF78   CE   9E 00009        MOVAB   -136(SP), SP                              :
     04   AE  20202020   8F   D0 0000E        MOVL    #538976288, BUFFER                        : 1895
                         02   DD 00016        PUSHL   #2                                        : 1899
     04   AE        08   AE   9E 00018        MOVAB   BUFFER, DESC+4                            : 1898
                         55   94 0001D        CLRB    REGION_FOUND                              : 1900
                    04   AC   D5 0001F        TSTL    CURRENT_VALUE                             : 1906
                         08   14 00022        BGTR    1$                                        :
     08   AE      2020   8F   3C 00024        MOVZWL  #8224, BUFFER                             : 1908
                         1E   11 0002A        BRB     2$                                        :
                    04   AC   9F 0002C 1$:    PUSHAB  CURRENT_VALUE                             : 1911
                    04   AE   9F 0002F        PUSHAB  DESC                                      :
                    08   AE   9F 00032        PUSHAB  DESC                                      :
               00000000'  00   9F 00035        PUSHAB  CTRSTR_DESC                               :
     00000000G   00        04   FB 0003B        CALLS   #4, SYS$FAOL                              :
                 05        50   E8 00042        BLBS    RTN_STATUS, 2$                            :
                           50   DD 00045        PUSHL   RTN_STATUS                                :
                 67        01   FB 00047        CALLS   #1, LIB$SIGNAL                            :
                 54   14   AC   D0 0004A 2$:    MOVL    SEPARATOR_CHAR, R4                        : 1917
                           56   D4 0004E        CLRL    R6                                        :
                           54   D5 00050        TSTL    R4                                        :
                           20   13 00052        BEQL    3$                                        :
                           56   D6 00054        INCL    R6                                        :
           0A         04   AC   D1 00056        CMPL    CURRENT_VALUE, #10                        : 1919
                           18   18 0005A        BGEQ    3$                                        :
                 50   1C   AC   D0 0005C        MOVL    SHADE_ARRAY_DESC, DESC_ADR_LCL            : 1921
     51    18    AC   18   A0   C5 00060        MULL3   24(DESC_ADR_LCL), DIM1, R1                :
                 51   0C   AC   C0 00066        ADDL2   COLUMN_POS, R1                            :
                 03   04 B041   D1 0006A        CMPL    @4(DESC_ADR_LCL)[R1], #3                  : 1922
                           03   12 0006F        BNEQ    3$                                        :
           55              01   90 00071        MOVB    #1, REGION_FOUND                          : 1924
                           52   D4 00074 3$:    CLRL    LOOP_COUNT                                : 1929
                           2F   11 00076        BRB     5$                                        :
     08 AE42          08   AE   B0 00078 4$:    MOVW    BUFFER, BUFFER[LOOP_COUNT]                : 1936
                 6E        02   A0 0007E        ADDW2   #2, DESC                                  : 1937
                 23        55   E9 00081        BLBC    REGION_FOUND, 5$                          : 1942
                 50   1C   AC   D0 00084        MOVL    SHADE_ARRAY_DESC, DESC_ADR_LCL            : 1944
     51    18    AC   18   A0   C5 00088        MULL3   24(DESC_ADR_LCL), DIM1, R1                :
     53          52   0C   AC   C1 0008E        ADDL3   COLUMN_POS, LOOP_COUNT, R3                :
                 51        53   C0 00093        ADDL2   R3, R1                                    :
                 03   04 B041   D1 00096        CMPL    @4(DESC_ADR_LCL)[R1], #3                  : 1945
                           0A   13 0009B        BEQL    5$                                        :
                 50        6E   3C 0009D        MOVZWL  DESC, R0                                  : 1948
        06 AE40            54   90 000A0        MOVB    R4, BUFFER-2[R0]                          :
                           55   94 000A5        CLRB    REGION_FOUND                              : 1949
     CC          52   08   AC   F2 000A7 5$:    AOBLSS  REPEAT_COUNT, LOOP_COUNT, 4$              : 1929
                 38        56   E9 000AC        BLBC    R6, 7$                                    : 1960
                 0A   04   AC   D1 000AF        CMPL    CURRENT_VALUE, #10                        : 1962
                           32   18 000B3        BGEQ    7$                                        :
                 50   1C   AC   D0 000B5        MOVL    SHADE_ARRAY_DESC, DESC_ADR_LCL            : 1964
     51    18    AC   18   A0   C5 000B9        MULL3   24(DESC_ADR_LCL), DIM1, R1                :
```

```
                  51      0C    AC   C0 000BF          ADDL2   COLUMN_POS, R1                              ;
                  03      04 B041    D1 000C3          CMPL    @4(DESC_ADR_LCL)[R1], #3                     ; 1965
                                19   13 000C8          BEQL    6$                                           ;
                  50      1C    AC   D0 000CA          MOVL    SHADE_ARRAY_DESC, DESC_ADR_LCL               ; 1967
      51      18  AC      18    A0   C5 000CE          MULL3   24(DESC_ADR_LCL), DIM1, R1                   ;
                  51      0C    AC   C0 000D4          ADDL2   COLUMN_POS, R1                               ;
                  50      04 B041    DE 000D8          MOVAL   @4(DESC_ADR_LCL)[R1], R0                     ;
                  03      FC    A0   D1 000DD          CMPL    -4(R0), #3                                   ; 1968
                                04   12 000E1          BNEQ    7$                                           ;
              08  AE            54   90 000E3  6$:     MOVB    R4, BUFFER                                   ; 1970
                  0A            55   E9 000E7  7$:     BLBC    REGION_FOUND, 8$                             ; 1976
                  50            6E   3C 000EA          MOVZWL  DESC, R0                                     ; 1979
           08 AE40              54   90 000ED          MOVB    R4, BUFFER[R0]                               ;
                                6E   B6 000F2          INCW    DESC                                         ; 1980
   0D 00000000'   00            02   E1 000F4  8$:     BBC     #2, DEVICE_FLAGS, 9$                         ; 1986
                                30   DD 000FC          PUSHL   #48                                          ; 1988
                          04    AE   9F 000FE          PUSHAB  DESC                                         ;
      00000000V   00            02   FB 00101          CALLS   #2, PUT_REGIS_TEXT                           ;
                                     04 00108          RET                                                  ;
                  50      0C    AC   D0 00109  9$:     MOVL    COLUMN_POS, R0                               ; 1991
                  51 00000000'  00   D0 0010D          MOVL    Y_AXIS_LINE, R1                              ;
              0C  AC      02 A140    3E 00114          MOVAW   2(R1)[R0], COLUMN_POS                        ;
                          0C    AC   9F 0011A          PUSHAB  COLUMN_POS                                   ; 1992
                          10    AC   9F 0011D          PUSHAB  ROW_POS                                      ;
                          08    AE   9F 00120          PUSHAB  DESC                                         ;
      00000000G   00            03   FB 00123          CALLS   #3, LIB$PUT_SCREEN                           ;
                  05            50   E8 0012A          BLBS    RTN_STATUS, 10$                              ;
                                50   DD 0012D          PUSHL   RTN_STATUS                                   ;
                  67            01   FB 0012F          CALLS   #1, LIB$SIGNAL                               ;
                                     04 00132  10$:    RET                                                  ; 1996

; Routine Size:  307 bytes,    Routine Base:  $CODE$ + 0795
```

```
1 %SBTTL 'Position the cursor on a given pixel address'
1 ROUTINE MOVE_CURSOR_REGIS (COMMAND_TYPE, X_POS, Y_POS, INTENSITY, SHADE_LINE)
1         : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !
1 !       MOVE_CURSOR_REGIS builds REGIS commands to move the cursor to a
1 !       specified position, draw a vector from the current postion to a
1 !       another postion, or draw a vector with shading to a relative line.
1 !       Send the command to the terminal.
1 !
1 !       >>>NOTE: all pixel addresses are absolute, except the line to be
1 !                shaded which is relative to the current postion.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       COMMAND_TYPE : code to indicate if a postion or vector command is
1 !                      being built
1 !       X_POS        : x-value address of pixel
1 !       Y_POS        : y-value address of pixel
1 !       INTENSITY    : intensity of the shaded area
1 !       SHADE_LINE   : relative pixel address of the shade line
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2 BEGIN
2
2 BIND
2     POS_CMD_START   = UPLIT( 'P[' ),            !Command to move cursor
2     COMMAND_MID     = UPLIT( ',' ),             !X and Y value separator
2     COMMAND_END     = UPLIT( ']' ),             !Command terminator
2     VCTR_CMD_START  = UPLIT( 'V(W(I3))[][' ),   !Vector command start
2     SHADE_CMD_1     = UPLIT( 'V(W (S1 S[,' ),   !First part of shade command
2     SHADE_CMD_2     = UPLIT( '] I3))[][+' )     !Remainder of command start
```

```
2           ;
2
2
2   LITERAL
2       POS_CMD_START_LEN  = %CHARCOUNT( 'P[' ),
2       CMD_MID_LEN        = %CHARCOUNT( ',' ),
2       CMD_END_LEN        = %CHARCOUNT( ']' ),
2       VCTR_CMD_START_LEN = %CHARCOUNT( 'V(W(I3))[][' ),
2       SHADE_CMD_1_LEN    = %CHARCOUNT( 'V(W (S1 S[,' ),
2       SHADE_CMD_2_LEN    = %CHARCOUNT( '] I3))[][+' ),
2       INTENSITY_LOC      = 7                  !Distance to intensity indicator from end
2       ;
2
2
2   LOCAL
2       TEMP_DESC :                             !Record temporary changes in buffer here
2           BBLOCK[ DSC$K_Z_BLN ]
2           INITIAL ( 0, 0 ),
2       BUFFER    :                             !Buffer to build REGIS command
2           VECTOR[ MAX_PAGE_WIDTH, BYTE ],
2       DESC      :                             !Descriptor for the buffer
2           BBLOCK[ DSC$K_Z_BLN ]
2       ;
2
2   BIND
2       CTRSTR = %ASCID '!UL'
2       ;
```

```
	!Initialize descriptor
	!
	DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
	DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
	DESC[ DSC$A_POINTER ] = BUFFER;


	!Place start of command string in buffer
	!
	SELECTONE .COMMAND_TYPE OF
	    SET

	    !Positon cursor
	    !
	    [ GRF$_REGIS_POS ] :
		BEGIN
		CH$MOVE( POS_CMD_START_LEN, POS_CMD_START, BUFFER);
		DESC[ DSC$W_LENGTH ] = POS_CMD_START_LEN;
		END;

	    !Draw vector
	    !
	    [ GRF$_REGIS_VCTR ] :
		BEGIN
		CH$MOVE( VCTR_CMD_START_LEN, VCTR_CMD_START, BUFFER);
		DESC[ DSC$W_LENGTH ] = VCTR_CMD_START_LEN;
		END;

	    !Draw a shaded vector
	    !
	    [ GRF$_REGIS_SHADE ] :
		BEGIN
		CH$MOVE( SHADE_CMD_1_LEN, SHADE_CMD_1, BUFFER);
		DESC[ DSC$W_LENGTH ] = SHADE_CMD_1_LEN;


		!Translate shade line address into ASCII and place in command.
		!
		TEMP_DESC[ DSC$W_LENGTH ] = MAX_PAGE_WIDTH - .DESC[ DSC$W_LENGTH ];
		TEMP_DESC[ DSC$A_POINTER ] =
				.DESC[ DSC$A_POINTER ] + .DESC[ DSC$W_LENGTH ];
P		TRANSLATE_VALUE( CTRSTR, TEMP_DESC[ DSC$W_LENGTH ], TEMP_DESC,
				SHADE_LINE );
		DESC[ DSC$W_LENGTH ] =
				.DESC[ DSC$W_LENGTH ] + .TEMP_DESC[ DSC$W_LENGTH ];


		CH$MOVE( SHADE_CMD_2_LEN, SHADE_CMD_2,
			BUFFER[ .DESC[ DSC$W_LENGTH ]  ]);
		DESC[ DSC$W_LENGTH ] = .DESC[ DSC$W_LENGTH ] + SHADE_CMD_2_LEN;


		!Place desired intensity in command string.
		!
		BUFFER[ .DESC[ DSC$W_LENGTH ] - INTENSITY_LOC ] =
				.INTENSITY;
```

```
            END;
        TES;

    !If there is an x value, translate it and put it in the command string.
    !
    IF .X_POS NEQ NO_VALUE
    THEN
        BEGIN

        TEMP_DESC[ DSC$W_LENGTH ]  = MAX_PAGE_WIDTH - .DESC[ DSC$W_LENGTH ];
        TEMP_DESC[ DSC$A_POINTER ] =
                .DESC[ DSC$A_POINTER ] + .DESC[ DSC$W_LENGTH ];
        TRANSLATE_VALUE( CTRSTR, TEMP_DESC[ DSC$W_LENGTH ], TEMP_DESC, X_POS );
        DESC[ DSC$W_LENGTH ]        =
                .DESC[ DSC$W_LENGTH ] + .TEMP_DESC[ DSC$W_LENGTH ];
        END;

    !Put comma in to separate x and y values.  If separator is more than one
    !byte this will not work!!!
    !
    BUFFER[ .DESC[ DSC$W_LENGTH ] ] = .COMMAND_MID;
    DESC[ DSC$W_LENGTH ]            = .DESC[ DSC$W_LENGTH ] + CMD_MID_LEN;

    !If there is a y value, translate it and put it in the command string.
    !
    IF .Y_POS NEQ NO_VALUE
    THEN
        BEGIN

        TEMP_DESC[ DSC$W_LENGTH ]  = MAX_PAGE_WIDTH - .DESC[ DSC$W_LENGTH ];
        TEMP_DESC[ DSC$A_POINTER ] =
                .DESC[ DSC$A_POINTER ] + .DESC[ DSC$W_LENGTH ];
        TRANSLATE_VALUE( CTRSTR, TEMP_DESC[ DSC$W_LENGTH ], TEMP_DESC, Y_POS );
        DESC[ DSC$W_LENGTH ]        =
                .DESC[ DSC$W_LENGTH ] + .TEMP_DESC[ DSC$W_LENGTH ];
        END;

    !Finish command string.
    !
    BUFFER[ .DESC[ DSC$W_LENGTH ] ] = .COMMAND_END;
    DESC[ DSC$W_LENGTH ]            = .DESC[ DSC$W_LENGTH ] + CMD_END_LEN;

    !Write the command string.
    !
    PUT_REGIS( DESC );

    RETURN;
END;                                    ! End of routine MOVE_CURSOR_REGIS
```

```
                                                               .PSECT  $PLIT$,NOWRT,NOEXE,2

                            00  00  5B  50  00198 P.AAU:       .ASCII  \P[\<0><0>                                  ;
                            00  00  00  2C  0019C P.AAV:       .ASCII  \,\<0><0><0>                                ;
                            00  00  00  5D  001A0 P.AAW:       .ASCII  \]\<0><0><0>                                ;
00  5B  5D  5B  29  29  33  49  28  57  28  56  001A4 P.AAX:   .ASCII  \V(W(I3))[][\<0>                            ;
00  2C  5B  53  20  31  53  28  20  57  28  56  001B0 P.AAY:   .ASCII  \V(W (S1 S[,\<0>                            ;
00  00  2B  5B  5D  5B  29  29  33  49  20  5D  001BC P.AAZ:   .ASCII  \] I3))[][+\<0><0>                          ;
                            00  4C  55  21  001C8 P.ABB:       .ASCII  \!UL\<0>                                    ;
                               010E0003  001CC P.ABA:          .LONG   17694723                                    ;
                              00000000'  001D0                 .ADDRESS P.ABB                                      ;

                                                       POS_CMD_START=          P.AAU
                                                       COMMAND_MID=            P.AAV
                                                       COMMAND_END=            P.AAW
                                                       VCTR_CMD_START=         P.AAX
                                                       SHADE_CMD_1=            P.AAY
                                                       SHADE_CMD_2=            P.AAZ
                                                       CTRSTR=                 P.ABA

                                                               .PSECT  $CODE$,NOWRT,2

                                         01FC 00000 MOVE_CURSOR_REGIS:
                                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8               ; 1998
                58 00000000G   00  9E 00002                    MOVAB   SYS$FAOL, R8                            ;
                57 00000000G   00  9E 00009                    MOVAB   LIB$SIGNAL, R7                          ;
                56 00000000'   00  9E 00010                    MOVAB   CTRSTR, R6                              ;
                5E       FF6C  CE  9E 00017                    MOVAB   -148(SP), SP                            ;
                           F8  AD  7C 0001C                    CLRQ    TEMP_DESC                               ; 2045
                           02  AE  B4 0001F                    CLRW    DESC+2                                  ; 2085
          04  AE           08  AE  9E 00022                    MOVAB   BUFFER, DESC+4                          ; 2086
              50           04  AC  D0 00027                    MOVL    COMMAND_TYPE, R0                        ; 2091
                               0A  12 0002B                    BNEQ    1$                                      ; 2096
          08  AE      CC   A6  B0 0002D                        MOVW    POS_CMD_START, BUFFER                   ; 2098
              6E           02  B0 00032                        MOVW    #2, DESC                                ; 2099
                               5E  11 00035                    BRB     4$                                      ; 2091
              01           50  D1 00037 1$:                    CMPL    R0, #1                                  ; 2104
                               0B  12 0003A                    BNEQ    2$                                      ;
    08  AE       D8  A6        0C  28 0003C                    MOVC3   #12, VCTR_CMD_START, BUFFER             ; 2106
                 6E            0C  B0 00042                    MOVW    #12, DESC                               ; 2107
                               4E  11 00045                    BRB     4$                                      ; 2091
              02           50  D1 00047 2$:                    CMPL    R0, #2                                  ; 2112
                               49  12 0004A                    BNEQ    4$                                      ;
    08  AE       E4  A6        0B  28 0004C                    MOVC3   #11, SHADE_CMD_1, BUFFER                ; 2114
                 6E            0B  B0 00052                    MOVW    #11, DESC                               ; 2115
    F8  AD     0084  8F        6E  A3 00055                    SUBW3   DESC, #132, TEMP_DESC                   ; 2120
                 50            6E  3C 0005C                    MOVZWL  DESC, R0                                ; 2122
          FC  AD         04 BE40  9E 0005F                     MOVAB   @DESC+4[R0], TEMP_DESC+4                ;
                           14  AC  9F 00065                    PUSHAB  SHADE_LINE                              ; 2124
                           F8  AD  9F 00068                    PUSHAB  TEMP_DESC                               ;
                           F8  AD  9F 0006B                    PUSHAB  TEMP_DESC                               ;
                               56  DD 0006E                    PUSHL   R6                                      ;
              68               04  FB 00070                    CALLS   #4, SYS$FAOL                            ;
              05               50  E8 00073                    BLBS    RTN_STATUS, 3$                          ;
                               50  DD 00076                    PUSHL   RTN_STATUS                              ;
```

```
                      67          01 FB 00078      CALLS   #1, LIB$SIGNAL                      ;
                      6E      F8  AD A0 0007B 3$:  ADDW2   TEMP_DESC, DESC                     ; 2126
                      50          6E 3C 0007F      MOVZWL  DESC, R0                            ; 2130
      08 AE40         F0      A6  0A 28 00082      MOVC3   #10, SHADE_CMD_2, BUFFER[R0]        ;
                      6E          0A A0 00089      ADDW2   #10, DESC                           ; 2131
                      50          6E 3C 0008C      MOVZWL  DESC, R0                            ; 2136
            01 AE40   10      AC  90 0008F         MOVB    INTENSITY, BUFFER-7[R0]             ; 2137
            FFFFFFFF  8F  08  AC  D1 00095 4$:     CMPL    X_POS, #-1                          ; 2144
                          2A  13 0009D             BEQL    6$                                  ;
      F8  AD   0084   8F      6E  A3 0009F         SUBW3   DESC, #132, TEMP_DESC               ; 2148
                      50          6E 3C 000A6      MOVZWL  DESC, R0                            ; 2150
              FC  AD      04 BE40 9E 000A9         MOVAB   @DESC+4[R0], TEMP_DESC+4            ;
                          08  AC  9F 000AF         PUSHAB  X_POS                               ; 2151
                          F8  AD  9F 000B2         PUSHAB  TEMP_DESC                           ;
                          F8  AD  9F 000B5         PUSHAB  TEMP_DESC                           ;
                              56  DD 000B8         PUSHL   R6                                  ;
                      68      04  FB 000BA         CALLS   #4, SYS$FAOL                        ;
                      05      50  E8 000BD         BLBS    RTN_STATUS, 5$                      ;
                              50  DD 000C0         PUSHL   RTN_STATUS                          ;
                      67      01  FB 000C2         CALLS   #1, LIB$SIGNAL                      ;
                      6E  F8  AD  A0 000C5 5$:     ADDW2   TEMP_DESC, DESC                     ; 2153
                      50      6E  3C 000C9 6$:     MOVZWL  DESC, R0                            ; 2160
            08 AE40   D0      A6  90 000CC         MOVB    COMMAND_MID, BUFFER[R0]             ;
                              6E  B6 000D2         INCW    DESC                                ; 2161
            FFFFFFFF  8F  0C  AC  D1 000D4         CMPL    Y_POS, #-1                          ; 2166
                              2A  13 000DC         BEQL    8$                                  ;
      F8  AD   0084   8F      6E  A3 000DE         SUBW3   DESC, #132, TEMP_DESC               ; 2170
                      50      6E  3C 000E5         MOVZWL  DESC, R0                            ; 2172
              FC  AD      04 BE40 9E 000E8         MOVAB   @DESC+4[R0], TEMP_DESC+4            ;
                          0C  AC  9F 000EE         PUSHAB  Y_POS                               ; 2173
                          F8  AD  9F 000F1         PUSHAB  TEMP_DESC                           ;
                          F8  AD  9F 000F4         PUSHAB  TEMP_DESC                           ;
                              56  DD 000F7         PUSHL   R6                                  ;
                      68      04  FB 000F9         CALLS   #4, SYS$FAOL                        ;
                      05      50  E8 000FC         BLBS    RTN_STATUS, 7$                      ;
                              50  DD 000FF         PUSHL   RTN_STATUS                          ;
                      67      01  FB 00101         CALLS   #1, LIB$SIGNAL                      ;
                      6E  F8  AD  A0 00104 7$:     ADDW2   TEMP_DESC, DESC                     ; 2175
                      50      6E  3C 00108 8$:     MOVZWL  DESC, R0                            ; 2181
            08 AE40   D4      A6  90 0010B         MOVB    COMMAND_END, BUFFER[R0]             ;
                              6E  B6 00111         INCW    DESC                                ; 2182
                              5E  DD 00113         PUSHL   SP                                  ; 2187
            00000000G 00      01  FB 00115         CALLS   #1, LIB$PUT_OUTPUT                  ;
                      03      50  E8 0011C         BLBS    RTN_STATUS, 9$                      ;
                              50  DD 0011F         PUSHL   RTN_STATUS                          ;
                      67      01  FB 00121         CALLS   #1, LIB$SIGNAL                      ;
                              04 00124 9$:         RET                                         ; 2192

; Routine Size:  293 bytes,    Routine Base:  $CODE$ + 08C8

; 1924        2193  1
```

```
1 %SBTTL 'Put REGIS text to the terminal'
1 ROUTINE PUT_REGIS_TEXT ( DESC, INTENSITY ) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Build a REGIS text command and put it out.  This procedure assumes the
1 !       cursor is already at the desired position.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DESC      :  address of descriptor for test string to be put out
1 !       INTENSITY :  intensity at which text is to be written
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2 BEGIN
2
2 MAP
2     DESC : REF BBLOCK                     !Descriptor for text to be output
2     ;
2
2 BIND
2     COMMAND_START = UPLIT( %STRING( 'T( W(i3))', %CHAR(39) )),
2     COMMAND_END   = UPLIT( %STRING( %CHAR(39)) )
2     ;
2
2 LITERAL
2     CMD_START_LEN = %CHARCOUNT( %STRING( 'T( W(i3))', %CHAR(39) )),
2     CMD_END_LEN   = %CHARCOUNT( %STRING (%CHAR(39)) ),
2     INTENSITY_LOC = 4
2     ;
2
2 LOCAL
2     CMD_BUFFER :                          !Buffer to build REGIS command
2         VECTOR[ MAX_PAGE_WIDTH, BYTE ],
```

```
	CMD_DESC		:				!Descriptor for the buffer
	    BBLOCK[ DSC$K_Z_BLN ]
	;
```

```
        !Initialize.
        !
        CMD_DESC[ DSC$B_CLASS ]   = DSC$K_CLASS_Z;
        CMD_DESC[ DSC$B_DTYPE ]   = DSC$K_DTYPE_Z;
        CMD_DESC[ DSC$A_POINTER ] = CMD_BUFFER;


        !Build start of command string.
        !
        CH$MOVE( CMD_START_LEN, COMMAND_START, CMD_BUFFER );
        CMD_DESC[ DSC$W_LENGTH ]  = CMD_START_LEN;
        CMD_BUFFER[ .CMD_DESC[DSC$W_LENGTH] - INTENSITY_LOC ] = .INTENSITY;


        !Copy text to be written into command buffer.
        !
        CH$MOVE( .DESC[ DSC$W_LENGTH ], .DESC[ DSC$A_POINTER ],
                 (CMD_BUFFER + .CMD_DESC[ DSC$W_LENGTH ] ));
        CMD_DESC[ DSC$W_LENGTH ] =
                 .CMD_DESC[ DSC$W_LENGTH ] + .DESC[ DSC$W_LENGTH ];


        !Put command terminator at end of string. WARNING!!! If more than one
        !character is used as a terminator this won't work.  This only places one
        !byte in the buffer.
        !
        CMD_BUFFER[ .CMD_DESC[ DSC$W_LENGTH ] ] = .COMMAND_END;
        CMD_DESC[ DSC$W_LENGTH ] = .CMD_DESC[ DSC$W_LENGTH ] + CMD_END_LEN;


        !Write the command string out.
        !
        PUT_REGIS( CMD_DESC );


        RETURN;
    END;                                        ! End of routine PUT_REGIS_TEXT


                                               .PSECT  $PLIT$,NOWRT,NOEXE,2

 00 00 27 29 29 33 69 28 57 20 28 54 001D4 P.ABC:  .ASCII  \T( W(i3))'\<0><0>                    ;
                         00 00 00 27 001E0 P.ABD:  .ASCII  \'\<0><0><0>                           ;

                                     COMMAND_START=        P.ABC
                                     COMMAND_END=          P.ABD

                                               .PSECT  $CODE$,NOWRT,2

                               007C 00000 PUT_REGIS_TEXT:
                                               .WORD   Save R2,R3,R4,R5,R6                      ; 2195
               5E      FF74  CE  9E 00002      MOVAB   -140(SP), SP                             ;
                         02  AE  B4 00007      CLRW    CMD_DESC+2                               ; 2258
                   04  AE  08  AE  9E 0000A    MOVAB   CMD_BUFFER, CMD_DESC+4                   ; 2259
      08   AE 00000000'  00      0A  28 0000F  MOVC3   #10, COMMAND_START, CMD_BUFFER           ; 2264
```

```
                          6E          0A  B0 00018          MOVW    #10, CMD_DESC                             ; 2265
                          50          6E  3C 0001B          MOVZWL  CMD_DESC, R0                              ; 2266
                     04 AE40      08  AC  90 0001E          MOVB    INTENSITY, CMD_BUFFER-4[R0]               ;
                          56      04  AC  D0 00024          MOVL    DESC, R6                                  ; 2271
        08 AE40      04   B6          66  28 00028          MOVC3   (R6), @4(R6), CMD_BUFFER[R0]              ; 2272
                          6E          66  A0 0002F          ADDW2   (R6), CMD_DESC                            ; 2274
                          50          6E  3C 00032          MOVZWL  CMD_DESC, R0                              ; 2281
                     08 AE40 00000000' 00  90 00035          MOVB    COMMAND_END, CMD_BUFFER[R0]               ;
                                      6E  B6 0003E          INCW    CMD_DESC                                  ; 2282
                                      5E  DD 00040          PUSHL   SP                                        ; 2287
              00000000G  00           01  FB 00042          CALLS   #1, LIB$PUT_OUTPUT                        ;
                         09           50  E8 00049          BLBS    RTN_STATUS, 1$                            ;
                                      50  DD 0004C          PUSHL   RTN_STATUS                                ;
              00000000G  00           01  FB 0004E          CALLS   #1, LIB$SIGNAL                            ;
                                          04 00055 1$:      RET                                               ; 2291

; Routine Size:  86 bytes,    Routine Base:  $CODE$ + 09ED
```

```
%SBTTL 'Shade a row of the graph'
ROUTINE SHADE_ROW_REGIS ( DIM1, SHADE_LINE, SHADE_ARRAY_DESC ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Determine the length of the vector to be drawn, where the shading
!       line is to be located and the intesity of the shaded area.
!
! FORMAL PARAMETERS:
!
!       DIM1            : index for the current row of the shade array
!       SHADE_LINE      : realtive location of the shade line
!       SHADE_ARRAY_DESC: descriptor for the array of shading information
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

BEGIN

MAP
    SHADE_ARRAY_DESC : REF BBLOCK           !Descriptor for the shading array
    ;

LOCAL
    SHADE_ARRAY     :                       !Array of intesities for shading
        REF TWO_DIM_ARRAY[ LONG, UNSIGNED ],
    DIM2,                                   !Index for second dimension of xy-array
    CURRENT_VALUE,                          !Current value in row, for comparisons
    REPEAT_COUNT,                           !Number of values equal to current value
    ROW_POS,                                !Current row on graph
    COLUMN_POS,                             !Column postion for output
    VECTOR_LEN                              !Location of x-origin when working with a VT125
    ;

```

```
; 2083     2348 2          !For each element in the row:
; 2084     2349 2          !
; 2085     2350 2          DIM2 = 0;
; 2086     2351 2          WHILE .DIM2 LEQ .SHADE_ARRAY_DESC[ DSC$L_U2 ]
; 2087     2352 2          DO
; 2088     2353 3              BEGIN
; 2089     2354 3
; 2090     2355 3              !Save the current xy-value for comparisons; determine the position
; 2091     2356 3              !in the row.
; 2092     2357 3              !
; 2093     2358 3              CURRENT_VALUE = .SHADE_ARRAY[ .DIM1, .DIM2, .SHADE_ARRAY_DESC ];
; 2094     2359 3              COLUMN_POS        = .DIM2;
; 2095     2360 3
; 2096     2361 3
; 2097     2362 3              !Compare the current value to all subsequent values in the row
; 2098     2363 3              !until a value which is not equal is encountered.  Keep a count
; 2099     2364 3              !of the equal values, so the right number can be printed in the
; 2100     2365 3              !graph.
; 2101     2366 3              !
; 2102     2367 3              DIM2 = .DIM2 + 1;
; 2103     2368 3              WHILE .DIM2 LEQ .SHADE_ARRAY_DESC[ DSC$L_U2 ]
; 2104     2369 3                      AND
; 2105     2370 3                  .CURRENT_VALUE EQL .SHADE_ARRAY[ .DIM1, .DIM2, .SHADE_ARRAY_DESC ]
; 2106     2371 3              DO
; 2107     2372 3                  DIM2 = .DIM2 + 1;
; 2108     2373 3
; 2109     2374 3
; 2110     2375 3              !Calculate the number of equal values and draw a shade line of the
; 2111     2376 3              !appropriate length (number-of-equal-values * unit-width).
; 2112     2377 3              !
; 2113     2378 3              REPEAT_COUNT = .DIM2 - .COLUMN_POS;
; 2114     2379 3              VECTOR_LEN   = .REPEAT_COUNT * UNIT_WIDTH;
; 2115     2380 3              MOVE_CURSOR_REGIS( GRF$_REGIS_SHADE, .VECTOR_LEN, NO_VALUE,
; 2116     2381 3                                 .CURRENT_VALUE + '0', .SHADE_LINE );
; 2117     2382 3
; 2118     2383 2          END;                            !DIM2
; 2119     2384 1  END;                                            ! End of routine SHADE_ROW_REGIS


                                          00FC 00000 SHADE_ROW_REGIS:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7                         ; 2293
                                          53 D4 00002     CLRL    DIM2                                           ; 2350
                    52        0C          AC D0 00004     MOVL    SHADE_ARRAY_DESC, R2                           ; 2351
               28   A2                    53 D1 00008 1$: CMPL    DIM2, 40(R2)                                   ;
                                          4B 14 0000C     BGTR    4$                                             ;
                    51                    52 D0 0000E     MOVL    R2, DESC_ADR_LCL                               ; 2358
          50        04    AC      18      A1 C5 00011     MULL3   24(DESC_ADR_LCL), DIM1, R0                     ;
                    50                    53 C0 00017     ADDL2   DIM2, R0                                       ;
                    54               04 B140 D0 0001A     MOVL    @4(DESC_ADR_LCL)[R0], CURRENT_VALUE            ;
                    56                    53 D0 0001F     MOVL    DIM2, COLUMN_POS                               ; 2359
                                          53 D6 00022 2$: INCL    DIM2                                           ; 2367
               28   A2                    53 D1 00024     CMPL    DIM2, 40(R2)                                   ; 2368
                                          13 14 00028     BGTR    3$                                             ;
                    50                    52 D0 0002A     MOVL    R2, DESC_ADR_LCL                               ; 2370
```

```
       51      04   AC     18  A0  C5 0002D        MULL3   24(DESC_ADR_LCL), DIM1, R1               ;
                    51         53  C0 00033        ADDL2   DIM2, R1                                 ;
                04 B041        54  D1 00036        CMPL    CURRENT_VALUE, @4(DESC_ADR_LCL)[R1]      ;
                               E5  13 0003B        BEQL    2$                                       ;
       57           53         56  C3 0003D 3$:    SUBL3   COLUMN_POS, DIM2, REPEAT_COUNT           ; 2378
       55           57         12  C5 00041        MULL3   #18, REPEAT_COUNT, VECTOR_LEN            ; 2379
                           08  AC  DD 00045        PUSHL   SHADE_LINE                               ; 2381
                           30  A4  9F 00048        PUSHAB  48(CURRENT_VALUE)                        ;
                    7E         01  CE 0004B        MNEGL   #1, -(SP)                                ; 2380
                               55  DD 0004E        PUSHL   VECTOR_LEN                               ;
                               02  DD 00050        PUSHL   #2                                       ;
              FE2E  CF         05  FB 00052        CALLS   #5, MOVE_CURSOR_REGIS                    ;
                               AF  11 00057        BRB     1$                                       ; 2351
                                   04 00059 4$:    RET                                              ; 2384

; Routine Size:  90 bytes,    Routine Base:  $CODE$ + 0A43
```

```
1 %SBTTL 'Draw a portion of the bar graph'
1 ROUTINE DRAW_BARS_REGIS ( Y_VALUE, X_VALUE, REPEAT_COUNT ) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Draw the bars for the current row segment of the REGIS line graph.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       Y_VALUE      :  The row in which contains the top of the bar.
1 !       X_VALUE      :  The first column of this row segment
1 !       REPEAT_COUNT :  The number of bars to be drawn
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2     BAR_GAP = 2                     !Number of ixels between bars in hte graph
2     ;
2
2 LOCAL
2     ROW_POS,                        !Y value for a pixel address
2     COLUMN_POS,                     !X value for a pixel address
2     SHADE_LINE                      !Line to which bar is shaded
2     ;
```

```
2       !Initialize values.
2       !
2       ROW_POS    = ( .Y_VALUE - 1 ) * UNIT_HEIGHT;
2       COLUMN_POS = ( .X_VALUE * UNIT_WIDTH ) + X_ORIGIN + 1 ;
2       SHADE_LINE = LINE_GRAPH_LEN * UNIT_HEIGHT;
2
2
2       !Move the cursor to the starting postion.
2       !
2       MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, .ROW_POS,
2                          NO_VALUE, NO_VALUE );
2
2
2       !For each bar in the line graph:
2       !
2       DECR LOOP_COUNT FROM (.REPEAT_COUNT - 1) TO 0
2       DO
3           BEGIN
3
3           !Draw a vector; shade it to the bottom of the graph.
3           !
3           MOVE_CURSOR_REGIS( GRF$_REGIS_SHADE, (UNIT_WIDTH - BAR_GAP), NO_VALUE,
3                              DARK_INTENSITY, .SHADE_LINE );
3
3
3           !Move the cursor over, to create spaces between the bars.
3           !
3           COLUMN_POS = .COLUMN_POS + UNIT_WIDTH;
3           MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, NO_VALUE,
3                              NO_VALUE, NO_VALUE );
3
2           END;
2
1   END;                                    ! End of routine DRAW_BARS_REGIS


                                003C 00000 DRAW_BARS_REGIS:
                                                  .WORD   Save R2,R3,R4,R5                    ; 2386
                  55    FE25  CF  9E 00002        MOVAB   MOVE_CURSOR_REGIS, R5
        50    04  AC          01  C3 00007        SUBL3   #1, Y_VALUE, R0                     ; 2435
                  50          12  C4 0000C        MULL2   #18, ROW_POS
        52    08  AC          12  C5 0000F        MULL3   #18, X_VALUE, R2                    ; 2436
                  52    00C0  C2  9E 00014        MOVAB   192(R2), COLUMN_POS
                  54      B4  8F  9A 00019        MOVZBL  #180, SHADE_LINE                    ; 2437
                  7E          01  CE 0001D        MNEGL   #1, -(SP)                           ; 2442
                  7E          01  CE 00020        MNEGL   #1, -(SP)                           ;
                              50  DD 00023        PUSHL   ROW_POS                             ;
                              52  DD 00025        PUSHL   COLUMN_POS                          ;
                              7E  D4 00027        CLRL    -(SP)                               ;
                  65          05  FB 00029        CALLS   #5, MOVE_CURSOR_REGIS               ;
                  53      0C  AC  D0 0002C        MOVL    REPEAT_COUNT, LOOP_COUNT            ; 2455
                              21  11 00030        BRB     2$                                  ;
                              54  DD 00032 1$:    PUSHL   SHADE_LINE                          ;
                              31  DD 00034        PUSHL   #49                                 ; 2454
```

```
                 7E      01  CE 00036         MNEGL   #1, -(SP)                   ;
                         10  DD 00039         PUSHL   #16                         ;
                         02  DD 0003B         PUSHL   #2                          ;
                 65      05  FB 0003D         CALLS   #5, MOVE_CURSOR_REGIS       ;
                 52      12  C0 00040         ADDL2   #18, COLUMN_POS             ; 2460
                 7E      01  CE 00043         MNEGL   #1, -(SP)                   ; 2461
                 7E      01  CE 00046         MNEGL   #1, -(SP)                   ;
                 7E      01  CE 00049         MNEGL   #1, -(SP)                   ;
                         52  DD 0004C         PUSHL   COLUMN_POS                  ;
                         7E  D4 0004E         CLRL    -(SP)                       ;
                 65      05  FB 00050         CALLS   #5, MOVE_CURSOR_REGIS       ;
                 DC      53  F4 00053 2$:     SOBGEQ  LOOP_COUNT, 1$              ; 2448
                             04 00056         RET                                 ; 2466

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 0A9D


; 2204          2467  1
```

```
%SBTTL 'Format the x-axis headers'
ROUTINE DRAW_X_AXIS ( GRAPH_LEN ) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Draw the x-axis line and then write the x-axis numeric and
!       descriptive labels.  These are the same for all graphs.
!
! FORMAL PARAMETERS:
!
!       GRAPH_LEN       :  the number of rows in a the graph
!
! IMPLICIT INPUTS:
!
!       Y_AXIS_LINE  :  column which contains the y-axis line.
!       DEVICE_FLAGS :  a longword of flags which are set or cleared by
!                       LIB$SCREEN_INFO depending on the terminal
!                       characteristics.
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! COMPLETION CODES:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

BEGIN

LITERAL
    INCR_VAL   = 5,                 !The increment between x_value labels
    TWO_VALUES = 2                  !For formatting last x-value label
    ;

LOCAL
    RTN_STATUS,                     !Status returned from external calls
    ROW_POS,                        !Row postion for output
    COLUMN_POS,                     !Column postion for output
    ROW_INCR,                       !Distance to next row
    COLUMN_INCR,                    !Distance to next column
    Y_ORIGIN,                       !Left most column in graph
    CHARACTER_WIDTH,                !Width of character, for calculations
    PLOT_WIDTH,                     !The width of the current graph
    BUFFER          :               !Buffer for building the output
        VECTOR[ MAX_PAGE_WIDTH, BYTE ],
```

```
2       DESC                    :                 !Descriptor for BUFFER
2           BBLOCK[ DSC$K_Z_BLN ]
2       ;
2
2   BIND
2                                                 !Header for the x-axis
2       X_AXIS_HEADER   = %ASCID'Bucket Size (number of blocks)',
2                                                 !Numeric label for the x-axis
2       X_AXIS_LABEL_N = PLIT ( %ASCID' 1',
2                               %ASCID' 5',
2                               %ASCID'10',
2                               %ASCID'15',
2                               %ASCID'20',
2                               %ASCID'25',
2                               %ASCID'30',
2                               %ASCID'32'
2                             ) : VECTOR,
2                                                 !Formatting for the x-axis line
2       AXIS_LINE       = PLIT ( %ASCID'+- + - - - + - - - - + - - - - + - - - - + - - - - + - - - - - + - +'
2                             ) : VECTOR;
```

```
	!Initialize values for distance between rows and columns.  Draw x-axis
	!line.
	!
	IF .DEVICE_FLAGS[ SCR$V_REGIS ]
	THEN
	    BEGIN


		!Initialize.
		!
		ROW_INCR     = 3;
		COLUMN_INCR = 0;
		Y_ORIGIN        = .Y_AXIS_LINE + AXIS_SHIFT;
		CHARACTER_WIDTH = CHAR_WIDTH;


		!Position x-axis line. Draw it.
		!
		ROW_POS     = UNIT_HEIGHT * ( .GRAPH_LEN ) + AXIS_SHIFT;
		COLUMN_POS = .Y_AXIS_LINE;


		MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, .ROW_POS );
		COLUMN_POS = .Y_ORIGIN + ( MAX_BUCKET_SIZE * UNIT_WIDTH );
		MOVE_CURSOR_REGIS( GRF$_REGIS_VCTR, .COLUMN_POS, NO_VALUE );
		END

	ELSE
	    BEGIN

		!Initialize.
		!
		ROW_INCR     = 1;
		COLUMN_INCR = SEPARATOR_WIDTH;
		Y_ORIGIN        = .Y_AXIS_LINE;
		CHARACTER_WIDTH = 1;


		!Position x-axis line. Draw it.
		!
		ROW_POS     = .GRAPH_LEN + 1;
		COLUMN_POS = .Y_ORIGIN;
		PUT_TEXT( .AXIS_LINE, ROW_POS, COLUMN_POS );
		END;


	!Move cursor to the beginning of the next row, put out first numeric label.
	!
	ROW_POS          = .ROW_POS + .ROW_INCR;
	COLUMN_POS       = .Y_ORIGIN + .COLUMN_INCR;


	IF .DEVICE_FLAGS[ SCR$V_REGIS ]
	THEN
	    BEGIN
		MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, .ROW_POS );
		PUT_REGIS_TEXT( .X_AXIS_LABEL_N[ 0 ], LIGHT_INTENSITY );
```

```
                ROW_INCR    =  CHAR_HEIGHT + SEPARATOR_WIDTH;
                COLUMN_INCR =  UNIT_WIDTH;
                END

        ELSE
            PUT_TEXT ( .X_AXIS_LABEL_N[ 0 ], ROW_POS, COLUMN_POS );


        !For each subsequent numeric label: positon cursor and write the label.
        !
        INCR DESC_PTR FROM 1  TO (.(X_AXIS_LABEL_N - FULLWORD) - TWO_VALUES )
        DO

            BEGIN
            COLUMN_POS = .Y_ORIGIN + ( .COLUMN_INCR * INCR_VAL * .DESC_PTR );
            IF .DEVICE_FLAGS[ SCR$V_REGIS ]
            THEN
                BEGIN
                COLUMN_POS = .COLUMN_POS - UNIT_WIDTH;
                MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, NO_VALUE);
                PUT_REGIS_TEXT( .X_AXIS_LABEL_N[ .DESC_PTR ], LIGHT_INTENSITY );
                END

            ELSE
                PUT_TEXT( .X_AXIS_LABEL_N[ .DESC_PTR ], ROW_POS, COLUMN_POS );
            END;


        !Position and write the last numeric label.
        !
        COLUMN_POS = .COLUMN_POS + .COLUMN_INCR * TWO_VALUES;
        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
        THEN
            BEGIN
            MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, NO_VALUE );
            PUT_REGIS_TEXT( .X_AXIS_LABEL_N[ .(X_AXIS_LABEL_N - FULLWORD) - 1 ],
                            LIGHT_INTENSITY );
            END

        ELSE
            PUT_TEXT( .X_AXIS_LABEL_N[ .( X_AXIS_LABEL_N - FULLWORD ) - 1 ],
                      ROW_POS, COLUMN_POS);


        !Put the descriptive labels out.
        !
        ROW_POS = .ROW_POS + .ROW_INCR;
        PLOT_WIDTH  = MAX_BUCKET_SIZE * .COLUMN_INCR;
        COLUMN_POS = (.PLOT_WIDTH - (.X_AXIS_HEADER<0,8> * .CHARACTER_WIDTH) )/2
                     + .Y_ORIGIN;


        IF .DEVICE_FLAGS[ SCR$V_REGIS ]
        THEN
            BEGIN
            MOVE_CURSOR_REGIS( GRF$_REGIS_POS, .COLUMN_POS, .ROW_POS );
            PUT_REGIS_TEXT( X_AXIS_HEADER, LIGHT_INTENSITY );
```

```
      3         END
      3
      2     ELSE
      2         PUT_TEXT( X_AXIS_HEADER, ROW_POS, COLUMN_POS );
      2
      1 END;                                        ! End of routine DRAW_X_AXIS


                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

75 6E 28 20 65 7A 69 53 20 74 65 6B 63 75 42  001E4 P.ABF:  .ASCII  \Bucket Size (number of blocks)\<0><0>   ;
29 73 6B 63 6F 6C 62 20 66 6F 20 72 65 62 6D  001F3                                                      ;
                                        00 00  00202                                                      ;
                                    010E001E  00204 P.ABE:  .LONG   17694750                              ;
                                   00000000'  00208         .ADDRESS P.ABF                                ;
                               00 00 31 20    0020C P.ABI:  .ASCII  \ 1\<0><0>                            ;
                                    010E0002  00210 P.ABH:  .LONG   17694722                              ;
                                   00000000'  00214         .ADDRESS P.ABI                                ;
                               00 00 35 20    00218 P.ABK:  .ASCII  \ 5\<0><0>                            ;
                                    010E0002  0021C P.ABJ:  .LONG   17694722                              ;
                                   00000000'  00220         .ADDRESS P.ABK                                ;
                               00 00 30 31    00224 P.ABM:  .ASCII  \10\<0><0>                            ;
                                    010E0002  00228 P.ABL:  .LONG   17694722                              ;
                                   00000000'  0022C         .ADDRESS P.ABM                                ;
                               00 00 35 31    00230 P.ABO:  .ASCII  \15\<0><0>                            ;
                                    010E0002  00234 P.ABN:  .LONG   17694722                              ;
                                   00000000'  00238         .ADDRESS P.ABO                                ;
                               00 00 30 32    0023C P.ABQ:  .ASCII  \20\<0><0>                            ;
                                    010E0002  00240 P.ABP:  .LONG   17694722                              ;
                                   00000000'  00244         .ADDRESS P.ABQ                                ;
                               00 00 35 32    00248 P.ABS:  .ASCII  \25\<0><0>                            ;
                                    010E0002  0024C P.ABR:  .LONG   17694722                              ;
                                   00000000'  00250         .ADDRESS P.ABS                                ;
                               00 00 30 33    00254 P.ABU:  .ASCII  \30\<0><0>                            ;
                                    010E0002  00258 P.ABT:  .LONG   17694722                              ;
                                   00000000'  0025C         .ADDRESS P.ABU                                ;
                               00 00 32 33    00260 P.ABW:  .ASCII  \32\<0><0>                            ;
                                    010E0002  00264 P.ABV:  .LONG   17694722                              ;
                                   00000000'  00268         .ADDRESS P.ABW                                ;
                                    00000008  0026C         .LONG   8                                     ;
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00270 P.ABG:  .ADDRESS P.ABH, P.ABJ, P.ABL, P.ABN, P.ABP, -  ;
                                        00000000' 00000000'  00288                  P.ABR, P.ABT, P.ABV                    ;
20 2D 20 2B 20 2D 20 2D 20 2D 20 2B 20 2D 2B  00290 P.ABZ:  .ASCII  \+- + - - - + - - - - + - - - - + - - - -\  ;
2D 20 2D 20 2D 20 2D 20 2B 20 2D 20 2D 20 2D  0029F                                                      ;
               2D 20 2D 20 2D 20 2D 20 2B 20  002AE                                                      ;
20 2D 20 2B 20 2D 20 2D 20 2D 20 2D 20 2B 20  002B8         .ASCII  \ + - - - - + - - - - + - - +\<0><0>  ;
      00 00 2B 20 2D 20 2B 20 2D 20 2D 20 2D  20 2D  002C7                                               ;
                                    010E0042  002D4 P.ABY:  .LONG   17694786                              ;
                                   00000000'  002D8         .ADDRESS P.ABZ                                ;
                                    00000001  002DC         .LONG   1                                     ;
                                   00000000'  002E0 P.ABX:  .ADDRESS P.ABY                                ;

                                                    X_AXIS_HEADER=      P.ABE
                                                    X_AXIS_LABEL_N=     P.ABG
                                                    AXIS_LINE=          P.ABX
```

```
                                                         .PSECT  $CODE$,NOWRT,2

                                     OFFC 00000 DRAW_X_AXIS:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 2469
                5B 00000000'   00  9E 00002             MOVAB   DEVICE_FLAGS, R11                             ;
                5A     FDC7    CF  9E 00009             MOVAB   MOVE_CURSOR_REGIS, R10                        ;
                59 00000000'   00  9E 0000E             MOVAB   X_AXIS_LABEL_N-4, R9                          ;
                5E     FF6C    CE  9E 00015             MOVAB   -148(SP), SP                                  ;
                50         04  AB  D0 0001A             MOVL    Y_AXIS_LINE, R0                               ; 2557
        36      6B             02  E1 0001E             BBC     #2, DEVICE_FLAGS, 1$                          ; 2548
                57             03  D0 00022             MOVL    #3, ROW_INCR                                  ; 2555
                52                 D4 00025             CLRL    COLUMN_INCR                                   ; 2556
                53         05  A0  9E 00027             MOVAB   5(R0), Y_ORIGIN                               ; 2557
                58             09  D0 0002B             MOVL    #9, CHARACTER_WIDTH                           ; 2558
        51      04  AC         12  C5 0002E             MULL3   #18, GRAPH_LEN, R1                            ; 2563
                04  AE     05  A1  9E 00033             MOVAB   5(R1), ROW_POS                                ;
                6E             50  D0 00038             MOVL    R0, COLUMN_POS                                ; 2564
                           04  AE  DD 0003B             PUSHL   ROW_POS                                       ; 2567
                           04  AE  DD 0003E             PUSHL   COLUMN_POS                                    ;
                               7E  D4 00041             CLRL    -(SP)                                         ;
                6A             03  FB 00043             CALLS   #3, MOVE_CURSOR_REGIS                         ;
                6E       0240  C3  9E 00046             MOVAB   576(R3), COLUMN_POS                           ; 2568
                7E             01  CE 0004B             MNEGL   #1, -(SP)                                     ; 2569
                           04  AE  DD 0004E             PUSHL   COLUMN_POS                                    ;
                               01  DD 00051             PUSHL   #1                                            ;
                6A             03  FB 00053             CALLS   #3, MOVE_CURSOR_REGIS                         ;
                               30  11 00056             BRB     2$                                            ; 2548
                57             01  D0 00058 1$:         MOVL    #1, ROW_INCR                                  ; 2577
                52             02  D0 0005B             MOVL    #2, COLUMN_INCR                               ; 2578
                53             50  D0 0005E             MOVL    R0, Y_ORIGIN                                  ; 2579
                58             01  D0 00061             MOVL    #1, CHARACTER_WIDTH                           ; 2580
   04   AE      04  AC         01  C1 00064             ADDL3   #1, GRAPH_LEN, ROW_POS                        ; 2585
                6E             53  D0 0006A             MOVL    Y_ORIGIN, COLUMN_POS                          ; 2586
                               5E  DD 0006D             PUSHL   SP                                            ; 2587
                           08  AE  9F 0006F             PUSHAB  ROW_POS                                       ;
                           74  A9  DD 00072             PUSHL   AXIS_LINE                                     ;
           00000000G   00      03  FB 00075             CALLS   #3, LIB$PUT_SCREEN                            ;
                       09      50  E8 0007C             BLBS    RTN_STATUS, 2$                                ;
                               50  DD 0007F             PUSHL   RTN_STATUS                                    ;
           00000000G   00      01  FB 00081             CALLS   #1, LIB$SIGNAL                                ;
                04  AE         57  C0 00088 2$:         ADDL2   ROW_INCR, ROW_POS                             ; 2593
        6E      53             52  C1 0008C             ADDL3   COLUMN_INCR, Y_ORIGIN, COLUMN_POS             ; 2594
        1D      6B             02  E1 00090             BBC     #2, DEVICE_FLAGS, 3$                          ; 2597
                           04  AE  DD 00094             PUSHL   ROW_POS                                       ; 2600
                           04  AE  DD 00097             PUSHL   COLUMN_POS                                    ;
                               7E  D4 0009A             CLRL    -(SP)                                         ;
                6A             03  FB 0009C             CALLS   #3, MOVE_CURSOR_REGIS                         ;
                               33  DD 0009F             PUSHL   #51                                           ; 2601
                           04  A9  DD 000A1             PUSHL   X_AXIS_LABEL_N                                ;
                 0125  CA      02  FB 000A4             CALLS   #2, PUT_REGIS_TEXT                            ;
                       57      11  D0 000A9             MOVL    #17, ROW_INCR                                 ; 2602
                       52      12  D0 000AC             MOVL    #18, COLUMN_INCR                              ; 2603
                               1B  11 000AF             BRB     4$                                            ; 2597
                               5E  DD 000B1 3$:         PUSHL   SP                                            ; 2607
                           08  AE  9F 000B3             PUSHAB  ROW_POS                                       ;
                           04  A9  DD 000B6             PUSHL   X_AXIS_LABEL_N                                ;
```

```
           00000000G  00          03  FB 000B9        CALLS   #3, LIB$PUT_SCREEN                    ;
                      09          50  E8 000C0        BLBS    RTN_STATUS, 4$                        ;
                                  50  DD 000C3        PUSHL   RTN_STATUS                            ;
           00000000G  00          01  FB 000C5        CALLS   #1, LIB$SIGNAL                        ;
      56              69          02  C3 000CC 4$:    SUBL3   #2, X_AXIS_LABEL_N-4, R6              ; 2612
                                  54  D4 000D0        CLRL    DESC_PTR                              ; 2616
                                  47  11 000D2        BRB     7$                                    ;
      50              52          54  C5 000D4 5$:    MULL3   DESC_PTR, COLUMN_INCR, R0             ;
                      50          05  C4 000D8        MULL2   #5, R0                                ;
      6E              50          53  C1 000DB        ADDL3   Y_ORIGIN, R0, COLUMN_POS              ;
                      55     04 A944  DE 000DF        MOVAL   X_AXIS_LABEL_N[DESC_PTR], R5          ; 2622
      19              6B          02  E1 000E4        BBC     #2, DEVICE_FLAGS, 6$                  ; 2617
                      6E          12  C2 000E8        SUBL2   #18, COLUMN_POS                       ; 2620
                      7E          01  CE 000EB        MNEGL   #1, -(SP)                             ; 2621
                             04   AE  DD 000EE        PUSHL   COLUMN_POS                            ;
                                  7E  D4 000F1        CLRL    -(SP)                                 ;
                      6A          03  FB 000F3        CALLS   #3, MOVE_CURSOR_REGIS                 ;
                                  33  DD 000F6        PUSHL   #51                                   ; 2622
                                  65  DD 000F8        PUSHL   (R5)                                  ;
                0125  CA          02  FB 000FA        CALLS   #2, PUT_REGIS_TEXT                    ;
                                  1A  11 000FF        BRB     7$                                    ; 2617
                                  5E  DD 00101 6$:    PUSHL   SP                                    ; 2626
                             08   AE  9F 00103        PUSHAB  ROW_POS                               ;
                                  65  DD 00106        PUSHL   (R5)                                  ;
           00000000G  00          03  FB 00108        CALLS   #3, LIB$PUT_SCREEN                    ;
                      09          50  E8 0010F        BLBS    RTN_STATUS, 7$                        ;
                                  50  DD 00112        PUSHL   RTN_STATUS                            ;
           00000000G  00          01  FB 00114        CALLS   #1, LIB$SIGNAL                        ;
      B5              54          56  F3 0011B 7$:    AOBLEQ  R6, DESC_PTR, 5$                      ; 2612
                                9E42  3F 0011F        PUSHAW  @COLUMN_POS[COLUMN_INCR]              ; 2632
      1A              6B          02  E1 00122        BBC     #2, DEVICE_FLAGS, 8$                  ; 2633
                      7E          01  CE 00126        MNEGL   #1, -(SP)                             ; 2636
                             04   AE  DD 00129        PUSHL   COLUMN_POS                            ;
                                  7E  D4 0012C        CLRL    -(SP)                                 ;
                      6A          03  FB 0012E        CALLS   #3, MOVE_CURSOR_REGIS                 ;
                                  33  DD 00131        PUSHL   #51                                   ; 2637
                      50          69  D0 00133        MOVL    X_AXIS_LABEL_N-4, R0                  ;
                                6940  DD 00136        PUSHL   X_AXIS_LABEL_N-4[R0]                  ;
                0125  CA          02  FB 00139        CALLS   #2, PUT_REGIS_TEXT                    ;
                                  1E  11 0013E        BRB     9$                                    ; 2633
                                  5E  DD 00140 8$:    PUSHL   SP                                    ; 2643
                             08   AE  9F 00142        PUSHAB  ROW_POS                               ;
                      50          69  D0 00145        MOVL    X_AXIS_LABEL_N-4, R0                  ;
                                6940  DD 00148        PUSHL   X_AXIS_LABEL_N-4[R0]                  ;
           00000000G  00          03  FB 0014B        CALLS   #3, LIB$PUT_SCREEN                    ;
                      09          50  E8 00152        BLBS    RTN_STATUS, 9$                        ;
                                  50  DD 00155        PUSHL   RTN_STATUS                            ;
           00000000G  00          01  FB 00157        CALLS   #1, LIB$SIGNAL                        ;
                  04  AE          57  C0 0015E 9$:    ADDL2   ROW_INCR, ROW_POS                     ; 2648
      50              52          05  78 00162        ASHL    #5, COLUMN_INCR, PLOT_WIDTH'          ; 2649
                      51     98   A9  9A 00166        MOVZBL  X_AXIS_HEADER, R1                     ; 2650
                      51          58  C4 0016A        MULL2   CHARACTER_WIDTH, R1                   ;
                      50          51  C2 0016D        SUBL2   R1, R0                                ;
                      50          02  C6 00170        DIVL2   #2, R0                                ;
      6E              50          53  C1 00173        ADDL3   Y_ORIGIN, R0, COLUMN_POS              ; 2651
      16              6B          02  E1 00177        BBC     #2, DEVICE_FLAGS, 10$                 ; 2654
                             04   AE  DD 0017B        PUSHL   ROW_POS                               ; 2657
```

```
                                   04      AE  DD 0017E        PUSHL   COLUMN_POS                      ;
                                           7E  D4 00181        CLRL    -(SP)                           ;
                       6A                  03  FB 00183        CALLS   #3, MOVE_CURSOR_REGIS           ;
                                           33  DD 00186        PUSHL   #51                             ; 2658
                                   98      A9  9F 00188        PUSHAB  X_AXIS_HEADER                   ;
                 0125  CA                  02  FB 0018B        CALLS   #2, PUT_REGIS_TEXT              ;
                                               04 00190        RET                                     ; 2654
                                           5E  DD 00191 10$:   PUSHL   SP                              ; 2662
                                   08      AE  9F 00193        PUSHAB  ROW_POS                         ;
                                   98      A9  9F 00196        PUSHAB  X_AXIS_HEADER                   ;
            00000000G  00                  03  FB 00199        CALLS   #3, LIB$PUT_SCREEN              ;
                       09                  50  E8 001A0        BLBS    RTN_STATUS, 11$                 ;
                                           50  DD 001A3        PUSHL   RTN_STATUS                      ;
            00000000G  00                  01  FB 001A5        CALLS   #1, LIB$SIGNAL                  ;
                                               04 001AC 11$:   RET                                     ; 2664

; Routine Size:  429 bytes,    Routine Base:  $CODE$ + 0AF4


; 2404          2665  1
; 2405          2666  1 END
; 2406          2667  0 ELUDOM



                                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  $PLIT$                         740  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                           12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        3233  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                       Library Statistics
;
;                                       -------- Symbols --------       Pages       Processing
;       File                            Total   Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776       13         0          581        00:01.0




;                               COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:EDFGRF/OBJ=OBJ$:EDFGRF MSRC$:EDFGRF/UPDATE=(ENH$:EDFGRF)

; Size:         3233 code + 752 data bytes
```

```
; Run Time:          01:00.4
; Elapsed Time:      02:03.6
; Lines/CPU Min:      2651
; Lexemes/CPU-Min:  17403
; Memory Used:  213 pages
; Compilation Complete
```

0127 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

EDFSDLMSG
LIS

EDFMAIN
LIS

EDFSHOW
LIS

EDFGRF
LIS

EDFMSG
LIS