USING GPLII

ASCUS TUTORIAL

April 1985

CALMA COMPANY, 501 SYCAMORE DRIVE, MILPITAS, CALIFORNIA 95035-7489

Proprietary and Legal Notices

TABLE OF CONTENTS

BITAND EXTERNAL DYADIC FUNCTION

Syntax:

Var := Array BITAND Array (CR)

Where Array is a scalar, vector or matrix.

Function:

The BITAND function returns the decimal value of two integers whose binary values are AND'ed together bit by bit. The maximum integer which may be entered for an argument is 32767. The minimum is -32768.

See also BITOR, BITXOR, AND, OR and XOR.

Program Information:

TYPEOF = Integer
SIZE = 1
LENGTH = 1
SHAPE = null

Example:

```
? 5 BITAND 6
4
?
```

Syntax:

Var := Array BITOR Array (CR)
Where Array is a scalar, vector or matrix.

Function:

The BITOR function returns the decimal value of two integers whose binary values are OR'ed together bit by bit. The maximum integer which may be entered for an argument is 32767. The minimum is -32768.

See also BITAND, BITXOR, AND, OR, AND XOR.

Program Information:

TYPEOF = Integer
SIZE = 1
LENGTH = 1
SHAPE = null

Example:

```
? 88 BITOR 22
94
?
```

Syntax:

Var := Array BITXOR Array (CR)

Where Array is a scalar, vector or matrix.

Function:

The BITXOR function returns the decimal value of two integers whose binary values are XOR'ed together bit by bit. The maximum integer which may be entered for an argument is 32767. The minimum is -32768.

See also BITAND and BITOR

Program Information:

TYPEOF = Integer
SIZE = 1
LENGTH = 1
SHAPE = null

Example:

```
? 8 BITXOR 32
40
?
```

Syntax:

Var := BOUNDARYAREA (n by 2 array) (CR)

Guidelines:

The first and last coordinates of the boundary matrix must be the same.

Function:

Assigns the variable the area of a boundary whose coordinates are specified in an n by 2 array. The area is returned in square user units.

PATHBOUNDARY may be used as an argument for BOUNDARYAREA, to find a path boundary area. For example:

```
AA := CEKEY (CE/CR)
AA (CR)
48367
BB := BOUNDARYAREA(PATHBOUNDARY AA)(CR)
BB (CR)
x
```

Where x is the path boundary area.

See also PATHBOUNDARY.

Example:

```
NILADIC PROCEDURE AREA
EXTERNAL CRD; CRDLIST; KEYNUM; INFO; BDAREA; PAREA

DO
CRD:=EXPINPUT"PLEASE INDICATE BOUNDARY OR PATH:  "
IF CRD <> "" THEN
  KEYNUM:=CEKEY CRD
  IF KEYNUM <> "" THEN
    INFO:=GETEL KEYNUM
    IF (INFO[1])[1] = 3 THEN
      CRDLIST:=INFO[6]
      GPLDRAW"WHITE";"SOLID";CRDLIST
      BDAREA:=BOUNDARYAREA CRDLIST
      "THE BOUNDARY AREA IS:  ";BDAREA
    ELIF (INFO[1])[1] = 4 THEN
      IF INFO[4] <> 0 THEN
        GPLDRAW"WHITE";"SOLID";(PATHBOUNDARY KEYNUM)
        PAREA:=BOUNDARYAREA (PATHBOUNDARY KEYNUM)
```

```
        "THE PATH AREA IS:   ";PAREA
      ELSE
        "PATH HAS NO WIDTH .... 0 AREA"
      ENDIF
    ENDIF
    ENDIF
  ENDIF
UNTIL CRD = ""
ENDDO

ENDSUB
```

```
? LOAD"AREA"
?
? AREA
PLEASE INDICATE BOUNDARY OR PATH: -5 31.
THE PATH AREA IS: 22.
PLEASE INDICATE BOUNDARY OR PATH: 0. 44.
THE BOUNDARY AREA IS: 342.
PLEASE INDICATE BOUNDARY OR PATH:
?
```

Syntax:

var := CECOUNT (CR)	Returns the number of coordinates in the current Item.
var := CECOUNT n (CR)	Returns the number of coordinates in Item n, where "n" is an integer number between 0 and 9.

Function:

This function returns the number of coordinates in an Item. If no coordinates are found in the item, the result will be 0.

Example:

```
 1  NILADIC PROCEDURE NUMPTS
 2  EXTERNAL CRD; PTS
 3
 4  CRD:=EXPINPUT"PLEASE INDICATE ELEMENT:   "
 5  IF CRD <> "" THEN
 6    GET CRD
 7    PTS:=CECOUNT
 8    "COORDINATE COUNT:   ";PTS
 9    PUT
10  ENDIF
11
12  ENDSUB
```

```
? NUMPTS
PLEASE INDICATE ELEMENT: 20. 14.
COORDINATE COUNT: 13
?
```

```
TYPE OF = INTEGER
SIZE    = 1
LENGTH  = 1
SHAPE   = 1
RANK    = 1
```

Syntax:

CFORMAT (vector or list) (CR)
or
CHARS := CFORMAT (vector or list) (CR)

Function:

Returns a character vector representing the elements of a numeric vector or list. CFORMAT can handle a list, FORMAT cannot. No (CR)'s are inserted into the character vector (differs from FORMAT).

If the argument is a list, the character vector to be output cannot exceed 512 characters, or the error message "FORMAT: Cannot handle this big list" appears. If the argument is a vector or matrix, the size of the output vector is only limited by the size of the GPLII work area.

The primary function of CFORMAT is to convert a numeric vector or list to a character vector.

Example:

```
MONADIC FUNCTION RESULT:=SUBMAT NUM
LOCAL RESULT; NUM

RESULT:=CFORMAT NUM
RESULT:=RESULT[IOTA (1, ((RESULT INDICESOF ".") - 1))]

ENDSUB
```

This monadic function uses the CFORMAT command to first return a character vector from the input value of the variable NUM, then subscript the decimal off the end of the character vector.

This character vector may then be concatenated onto the end of a structure name, as in the next program example:

```
NILADIC PROCEDURE BLDFONTLIB
EXTERNAL MONADIC FUNCTION SUBMAT
EXTERNAL N
:This program will create the necessary structures for creating
:text fonts, using the first structure "MYFONTO" which contains
:the defining box and measurement paths.


N:=31
DO
N:=N + 1
N:=SUBMAT N
CSTRUCT "MYFONTO";"MYFONT",N
N:=EXECUTE N
UNTIL N = 125
```

```
ENDD 0

ENDS UB
```

TYPE OF = CHAR or NULL
LENGTH = 1
SIZE = n CHAR (maximum of 512)
SHAPE = n CHAR (maximum of 512)
RANK = 1

Syntax:

Var := (nX2 array) CLOSESTTO (CE/CR)

Function:

Assigns to the variable the index locating the closest polygon coordinate to the specified coordinate. The coordinates of the polygon must be stored in an nX2 array. Each coordinate is compared to the second argument and the closest one is assigned to the variable.

It is suggested that the coordinates of a polygon be entered into the nX2 array in a counter-clockwise direction for consistency.

Program Information:

TYPEOF = Integer
SIZE = 1
LENGTH = 1
SHAPE = null

Example:

```
? 34 -34 234 12 -12 10 CLOSESTTO 4 7
3
?
```

or

```
? AA := 34 -34 234 12 -12 10
? AA CLOSESTTO 4 7
3
?
```

For these examples, 3 is the third coordinate in the array, i.e., (-12, 10).

Syntax:

COORDMARK (CR)
or
COORDMARK n (CR)

Function:

Marks and numbers the coordinates of an element in an Item. COORDMARK n (CR), where "n" is an integer number between 0 and 9, will mark the coordinates of the element in Item n.

The numbers assigned to each coordinate will represent the order in which the element was entered into the data base structure.

Example:

```
NILADIC PROCEDURE SHOWCRDS
EXTERNAL CRD

CRD:=EXPINPUT"PLEASE INDICATE ELEMENT:  "
IF CRD <> "" THEN
  GET CRD
  COORDMARK
  PUT
ENDIF

ENDSUB
```

This program will get the specified element into the current item, the mark and number the coordinates in the order they were entered.

Syntax:

```
    DRAWARROW (CE1)(CE2) . . . (CE/CR)
or
    DRAWARROW(CR)
    Coordinate: (CE/CR)
        :
        :
    Coordinate: (CR)
```

Function:

This command draws a bright white arrow on the graphics CRT, superimposed over other graphics data. The arrow can be erased (along with all other bright white, temporary graphics) by REMOVE or REDRAW without affecting graphics that represent elements stored in the GDSII database. More selective erasure is available with ERASEARROW. The database itself is not affected by either drawing or erasing the arrow.

The arrow shaft may consist of any number of segments. DRAWARROW draws a line segment from each CE entered to the next CE. The arrowhead is located at the first CE, and is automatically rotated to match the orientation of the first segment of the arrow shaft. The size of the arrowhead is fixed.

DRAWARROW is useful in application such as error reporting and computer-aided training.

Example:

```
? DRAWARROW
  Coordinate: 0. 0.
  Coordinate: 10. 0.
  Coordinate: (CR)
?
? DRAWARROW 0. 0. 10. 0. (CR)
?
```

Syntax:

DRAWBOX "CORNER"; lower_left; upper_right; corner_radius; oversize(CR)

or

DRAWBOX "CENTER"; center; width, height; corner_radius; oversize(CR)

Function:

This command draws a bright white, temporary box on the graphics CRT, superimposed over other graphic data. The box can be erased (along with other temporary graphics) by REMOVE or REDRAW, without erasing graphic data that represents elements stored in the GDSII database. More selective erasure is possible using ERASEBOX. DRAWBOX adds no data to the GDSII database.

The box location and dimensions can be specified in terms of either:

1) lower left and upper right box coordinates ("CORNER" mode); or

2) box center, width, and height ("CENTER" mode).

The mode is determined by the first command argument, while location and dimension information is specified by the second and third arguments, which are given in user units.

The "corner_radius" argument, which is optional, determines whether the box will have square or rounded corners. If "corner_radius" is omitted or zero, square corners will be drawn; otherwise, the corners will be 90-degree arcs with an arc radius equal to "corner radius". Unlike the preceding arguments, "corner_radius" is expressed as a fraction of the width of the working area on the screen. For example, a value of 0.01 is equal to 1/100 of the width of the working area.

If "corner_radius" is larger than either half the box height or half the box width, the actual arc radius will be reduced to the smaller opf those two values. DRAWBOX will thus draw a circle of diameter equal to the box height if the command arguments specify a large "corner_radius" value and equal box height and width (see Examples).

The optional "oversize" argument, if present, causes the box to be oversized relative to the dimensions specified by the second and third arguments. Like "corner_radius", "oversize" is specified as a fraction of the width of the working area on the screen.

Examples:

? DRAWBOX "CENTER"; 10, 10; 4, 7

Draws a box centered at 10, 10, with a width of 4 user units and a height of 7 user units. The box will have square corners.

? DRAWBOX "CORNER"; 8, 6.5; 12, 13.5

Draws the same box as the preceding example, but specifies it in terms of its extent.

? DRAWBOX "CORNER"; 8, 6.5; 12, 13.5; 0.02

Draws a box similar to the preceding example, but with rounded corners.

? DRAWBOX "CENTER"; 10, 10; 6, 6; 1.0

Draws a circle of diameter 6, centered at 10, 10.

? DRAWBOX "CORNER"; 8, 6.5; 12, 13.5; 0.01; 0.01

Draws a box with rounded corners around the text. The box is oversized relative to the box extent.

Program example:

For a program example please see the GPLII program IDNODEPROP.GS. This program may be found with the command PROPVALUE in this section.

Syntax:

```
    DRAWCROSS (CE1)(CE2) . . . (CE/CR)
or
    DRAWCROSS(CR)
    Coordinate: (CE/CR)
        :
        :
    Coordinate: (CR)
```

Function:

This command draws bright white crosses, resembling GDSII Node elements, on the graphics CRT. A cross is drawn at each coordinate entered. Just as for Node elements, the size of the cross is determined by the value of MARKSIZE. However, no element is actually created in the GDSII database.

The crosses are superimposed over other graphic data on the screen. They can be erased by REMOVE, REDRAW, or ERASECROSS without affecting graphics that represent elements stored in the database. UNDO does not erase crosses drawn by DRAWCROSS.

DRAWCROSS is normally used in GPLII programs, when Node element high-lighting is required.

Program example:

For a program example please see the GPLII program IDNODEPROP.GS. This program is shown with the command PROPVALUE in this section.

Syntax:

DRAWPROMPT "prompt"(CR)
or
DRAWPROMPT "prompt";"R"(CR)
or
DRAWPROMPT "prompt";"L"(CR)
or
DRAWPROMPT "prompt";"B"(CR)

Function:

Displays a prompt, in bright white erasable text, on the graphics screen. Normally it is used within a GPLII program to generate messages at certain locations on the graphics screen, when the flexibility of the DRAWTEXT command is not required. The text font and character set are fixed, and are the same as for DRAWTEXT. The text runs horizontally, and has a fixed size. Justification is set automatically, according to location.

The location at which the text will be drawn is controlled by the second argument. There are three possible locations. If the second argument is omitted or is "R", the text appears at the right edge of the working area on the screen, halfway between top and bottom. An "L" argument causes the text to appear at the left edge of the screen, midway between top and bottom, while "B" causes it to appear at the bottom edge, centered between left and right edges.

The prompt will remain on the screen until erased by REMOVE, ERASEPROMPT, or any command which causes the screen to be redrawn. It is advantageous to use ERASEPROMPT when it is desired to erase the prompt without erasing other bright white graphics that might be on the screen.

Program example:

For a program example please refer to the GPLII program IDNODEPROP.GS. This program may be found with the command PROPVALUE in this section.

Syntax:

```
DRAWSEGS (CR1)(CE2) . . . (CE/CR)

or

DRAWSEGS(CR)
Coordinate: (CE/CR)
      :
      :
Coordinate: (CR)
```

Function:

This command draws bright white line segments, connecting the entered coordinates, on the graphics CRT. If only a single coordinate is entered, a diamond is drawn. No element is created in the GDSII database.

These line segments are superimposed over other graphic data on the screen. They can be erased by REMOVE or REDRAW without affecting graphics that represent elements stored in the database. Alternatively, more selective segment erasure can be accomplished by ERASESEGS. UNDO does not erase line segments drawn by DRAWSEGS.

DRAWSEGS is normally used in GPLII programs, when polygon highlighting or other temporary line graphics are required.

Example:

```
?
? DRAWSEGS
Coordinate: 0 0
Coordinate: 10 0
Coordinate: 10 10
Coordinate:
?
? DRAWSEGS 0 0 10 0 10 10
?
```

Syntax:

DRAWTEXT text_string; justification; orientation; size; origins; "BOX" (CR)

Function:

This command draws bright white text on the graphics CRT, superimposed over other graphics data. The text may be enclosed in a bright white box, if "BOX" option is used. No database elements are created, and the text may be erased by REMOVE or REDRAW without affecting graphics representing elements stored in the GDSII database. A specific text string may be erased by ERASETEXT, without affecting other superimposed graphics.

DRAWTEXT is normally used in GPLII programs, when erasable text is required. Typical applications are graphical error reporting and visual display of text stored in User Properties.

A preset text font is used. The following characters may be displayed:

(1) uppercase letters (A through Z);
(2) numerals (0 through 9);
(3) dollar sign, period, slash, hyphen, underbar, colon, question mark, comma, apostrophe, exclamation point, left/right parentheses, and left/right angle brackets.

The text string may also contain lowercase characters, but these will be translated to uppercase before display. Any non-displayable character is replaced by a space. The text string may contain up to 1000 characters. The user can enter multiline text string. Lines are terminated by a percent character, which is not displayed on the graphics CRT, but just serves as a line separator in the text_string argument.

The first five arguments must be supplied. The sixth argument is optional. The arguments are as follows:

text_string — The string of characters to be displayed.

justification — Justification of text relative to each origin. Acceptable values are:

" "	Left justification
"L"	Left justification
"C"	Center justification
"R"	Right justification
"DC"	Dead-center justification

For left, center, or right justification, text position is slightly offset from the actual origin, so that if a diamond were placed at the origin, text would not overlap the diamond. Dead-center justification centers the text precisely over the origin. If the text has multiple lines, however, text can overlap the diamond in any of the above justifications.

orientation	Specifies one of three directions for text: " " Horizontal "H" Horizontal "U" Upward "D" Downward
size	Numerical argument specifying text size as a fraction of the width of the working area on the screen. (The "corner radius" and "oversize" arguments of DRAWBOX are specified in a similar way.) For reference, a value of .01 sets the character height to 0.0155 of the working area width. If size=" ", a default value of .007 is used.
origins	A vector of CEs. Text string is displayed at each origin supplied. Maximum of 20 origins is allowed.
"BOX"	An optional argument, if present, causes a bright white box to be drawn around the text. The box has rounded corners, and is oversized relative to the text extents.

Examples:

? DRAWTEXT "THIS IS A%TEXT STRING!"; "DC"; "H"; 0.01; 10,10; "BOX"

Draws the two lines of the text string "THIS IS A TEXT STRING!" horizontally on the graphics CRT, dead-centered at 10, 10. The first line is "THIS IS A", the second line is "TEXT STRING!". The character height is 0.155 of the working area width on the screen. The origin is just between the two lines of text. The box is drawn around the text.

Program example:

For a program example please refer to the GPLII program, IDNODEPROP.GS. This program may be found with the command PROPVALUE in this section.

Syntax:

var := vector DROP array

Function:

DROP is a dyadic function defined analagously to TAKE except that the argument "vector" specifies the elements of "array" that will NOT appear in the result.

Example:

```
? AA := 1 2 3 4 5 6 7 8
? 2 DROP AA
 3. 4. 5. 6. 7. 8.
? #3 DROP AA
1. 2. 3. 4. 5.
? BB := 3 4 RESHAPE (IOTA 12)
? BB
1 2 3 4
5 6 7 8
9 10 11 12
? #1 #1 DROP BB
1 2 3
5 6 7
?
```

Syntax:

```
Var := C DROUND D (CR)
Z
```

Where C is an integer, D is a whole number and z is the value of c rounded to the decimal place location specified by D.

Function:

Assigns the variable the value of the first argument, rounded to the decimal place location specified by the second argument. The second argument must be a whole number. When the second argument is 1, it designates that the first argument is to be rounded off one place to the right of the decimal point. When it is 2, the first variable is rounded two places to the right of the decimal point; when it is 3, the first variable is rounded three places to the right of the decimal point, etc.

Similarly, if the second argument is 0, the variable is assigned the value of the first argument rounded to the one's decimal place. If the second argument is -1, the variable value is the first argument rounded to the ten's decimal place, and if the value of the first argument is -2, the variable value is the first argument, rounded to the hundred's decimal place, etc.

See also ROUND.

Program Information:

TYPEOF = Real
SIZE = Same as SIZE of first argument
LENGTH = 1
SHAPE = Same as SHAPE of first argument

Examples:

```
? 3 DROUND 3407837.8395017
3407837.840
?
```

or

```
? -3 DROUND 3407837.8395017
3408000
?
```

Syntax:

ERASEARROW (CE)(CE) . . . (CE/CR)

Function:

This command erases a bright white, temporary arrow from the graphics CRT. The arrow is assumed to have been drawn previously by the DRAWARROW command, using the same coordinates. ERASEARROW operates in a manner identical to DRAWARROW, except that it has only a monadic version and erases instead of draws. ERASEARROW does not erase graphics representing elements stored in the GDSII database, and does not delete data from the database.

A temporary, bright white arrow can also be erased by REMOVE or REDRAW, but ERASEARROW has the advantage of erasing only the arrow specified by the given coordinates.

ERASEARROW would generally be used in a GPLII program. Typical applications include error reporting and on-line tutorials.

Syntax:

ERASEBOX "CORNER"; lower_left; upper_right; corner_radius; oversize(CR)

or

ERASEBOX "CENTER"; center; width,height; corner_radius; oversize(CR)

Function:

This command erases a bright white, temporary box from the graphics CRT. The box is assumed to have been drawn previously by DRAWBOX, using the identical command arguments. ERASEBOX operates in a manner identical to DRAWBOX, except that it erases instead of draws. ERASEBOX does not erase graphics representing elements stored in the GDSII database, and does not delete data from the database.

A temporary, bright white box can also be erased by REMOVE or REDRAW, but ERASEBOX has the advantage of erasing only the box specified by the given arguments.

ERASEBOX would generally be used in a GPLII program. Typical applications include error reporting and on-line tutorials.

See DRAWBOX for further information on ERASEBOX arguments.

Syntax:

ERASECROSS (CE1)(CE2) . . . (CE/CR)

Function:

At each CE on the command line, ERASECROSS erases a bright white cross (X) from the graphics CRT. Underlying graphics (e.g., Node elements) are not affected if they represent data stored in the GDSII database. The size of the erased cross is set by MARKSIZE.

ERASECROSS is normally used for selective erasure of crosses drawn by the DRAWCROSS command.

Syntax:

ERASEPROMPT "prompt"(CR)
or
ERASEPROMPT "prompt"; "R"(CR)
or
ERASEPROMPT "prompt"; "L"(CR)
or
ERASEPROMPT "prompt"; "B"(CR)

Function:

ERASEPROMPT is the prompt-erasure counterpart of DRAWPROMPT. It operates in a manner identical to DRAWPROMPT, except that it erases rather than draws.

See the DRAWPROMPT helpfile for further information.

Syntax:

ERASESEGS (CE1)(CE2) . . . (CE/CR)

Function:

This command erases bright white line segments, previously drawn by DRAWSEGS, from the graphics CRT. If only a single coordinate is specified, a diamond is erased. No element is deleted from the GDSII database. ERASESEGS provides more selective erasure than either REMOVE or REDRAW, in the sense that it erases only the segments connecting the specified coordinates, and does not affect other temporary bright white graphics that may be on the screen.

ERASESEGS operates exactly the same as DRAWSEGS, except that it has only a monadic form, and erases rather than draws lines.

ERASESEGS is normally used in GPLII programs, when polygon highlighting or other temporary line graphics are required.

Syntax:

ERASETEXT text_string; justification; orientation; size; origins; "BOX"(CR)

Function:

This command erases a temporary bright white text string, previously drawn by DRAWTEXT, from the graphics CRT. It operates exactly the same as DRAWTEXT, except that it erases instead of draws.

No database elements are deleted. Unlike REMOVE or REDRAW, ERASETEXT erases selectively; temporary graphics other than the specified text string are not affected.

ERASETEXT is normally used in GPLII programs, in applications where temporary text is required. Typical application are graphical error reporting, prompting on the graphics CRT, and visual display of text stored in User Properties.

A preset text font is used. The following characters may be erased:

(1) uppercase letters (A through Z);
(2) numerals (0 through 9);
(3) dollar sign, period, slash, hyphen, underbar, and colon, question mark, comma, apostrophe, exclamation point, and left/right parentheses.

The text string may also contain lowercase characters, but these will be translated to uppercase before erasure. Any non-displayable character is replaced by a space. The text string may contain up to 1000 characters. The user can enter multiline text string. Lines are terminated by a percent character, which is not displayed on the graphics CRT, but just serves as a line separator in the text_string argument.

The first five arguments must be supplied. The sixth argument is optional. The arguments are as follows:

text_string — The string of characters to be erased.

justification — Justification of text relative to each origin. Acceptable values are:

" "	Left justification
"L"	Left justification
"C"	Center justification
"R"	Right justification
"DC"	Dead-center justification

For left, center, or right justification, text position is slightly offset from the actual origin, so that if a diamond were placed at the origin, text would not overlap the diamond. Dead-center justification centers the text precisely over the origin.

orientation	Specifies one of three directions for text: " " Horizontal "H" Horizontal "U" Upward "D" Downward
size	Numerical argument specifying text size as a fraction of the width of the working area on the screen. (The "corner_radius" and "oversize" arguments of DRAWBOX are specified in a similar way.) For reference, a value of .01 sets the character height to 0.0155 of the working area width. If size=" ", a default value of .007 is used.
origins	A vector of CEs. Text string is displayed at each origin supplied. Maximum of 20 origins is allowed.
"BOX"	An optional argument, if present, causes a bright white box to be drawn around the text. The box has rounded corners, and is oversized relative to the text extents.

Examples:

? DRAWTEXT "THIS IS A%TEXT STRING!"; "DC"; "H"; 0.01; 10,10; "BOX"

Draws the two lines of the text string "THIS IS A TEXT STRING!" horizontally on the graphics CRT, dead-centered at 10, 10. The first line is "THIS IS A", the second line is "TEXT STRING!". The character height is 0.155 of the working area width on the screen. The origin is just between the two lines of text. The box is drawn around the text.

Program example:

For a program example please refer to the GPLII program, IDNODEPROP.GS. This program may be found with the command PROPVALUE in this section.

Niladic Syntax:

var := GEDINTEGER(CR)

Function:

Assigns the value of property number 126 in the current Item to the variable as a real integer. Any alphanumerics in the value will be represented by zeros.

Returns a null if the Item has no value for property number 126.

GEDSTRING returns the value of property number 127.

Program Information:

TYPEOF = Integer or Null
SIZE = 1 (0 if null)
LENGTH = 1
SHAPE = 1 (0 if null)

Example:

```
? PROP126 := GEDINTEGER(CR)
?
```
Assigns the value of property number 126 to PROP126.

Niladic Syntax:

var := GEDPLEX(CR)

Function:

Assigns the plex number in the current Item to the variable as an Integer2.

Returns a null if the element in the Item has no plex number.

Program Information:

TYPEOF = Integer2 or Null
SIZE = 1 (0 if null)
LENGTH = 1
SHAPE = 1 (0 if null)

Example:

```
? GPX := GEDPLEX(CR)
?
```
Assigns the current plex number to GPX.

Niladic Syntax:

var := GEDSTRING(CR)

Function:

Assigns the value of property number 127 in the current Item to the variable as a character string.

Returns a null if the Item has no value for property number 127.

GEDINTEGER returns the value of property number 126.

Program Information:

TYPEOF = Char or Null
SIZE = n Char (0 if Null)
LENGTH = 1
SHAPE = n Char (0 if Null)

Example:

```
? PROP127 := GEDSTRING(CR)
?
```
Assigns the value of property number 127 to PROP127.

Syntax:

```
? GPLDRAW (CR)
    Color (WHITE): color (CR)
    Line Type (SOLID): line type (CR)
    Ce(s):
    Ce(s):
    .
    .
    Ce(s): (CR)
```

or

```
? GPLIIDRAW "color" ; "line type" ; coordinate array or vector (CR)
```

Function:

This command allows the user to draw any shape in bright white lines on the CRT. The data is temporary and may be removed with the command REDRAW. The command REMOVE does not remove the graphics. The temporary lines will be superimposed over other graphic data.

This command may be used both niladically and monadically. The arguments include the color (i.e., RED, GREEN, CYAN) in which to draw, the line type (i.e., SOLID, BROKEN, DASHED) in which the graphics will appear, and the coordinates of the graphics to be entered. The default color is WHITE and the default line type is SOLID.

The coordinates may be either a vector or a matrix.

Example:

```
? GPLIIDRAW
Color (WHITE):
Line type (SOLID):
Ce(s): 0 0 -9. 20.
Ce(s): 0. 21.
Ce(s): 0. 31.
Ce(s): -7. 35.
Ce(s): -11. 39.5
Ce(s): -15.5 31.5
Ce(s): -12.5 22.5
Ce(s):
?
? GPLDRAW"WHITE";"SOLID"; 0 0 10 0 10 10 0 10 0 0
?
CRDS:=5 2 RESHAPE 0 0 10 0 10 10 0 10 0 0
? CRDS
```

```
 0.       0.
10.       0.
10.      10.
 0.      10.
 0.       0.
? GPLDRAW "WHITE";"BROKEN"; CRDS
?
? GPLDRAW " ";" "; CRDS
```

In this last example, since the first two arguments were null (" ") the command will create graphics in WHITE and SOLID.

Program example:

```
NILADIC PROCEDURE AREA
EXTERNAL CRD; CRDLIST; KEYNUM; INFO; BDAREA; PAREA

DO
CRD:=EXPINPUT"PLEASE INDICATE BOUNDARY OR PATH:   "
IF CRD <> "" THEN
  KEYNUM:=CEKEY CRD
  IF KEYNUM <> "" THEN
    INFO:=GETEL KEYNUM
    IF (INFO[1])[1] = 3 THEN
      CRDLIST:=INFO[6]
      GPLDRAW"WHITE";"SOLID";CRDLIST
      BDAREA:=BOUNDARYAREA CRDLIST
      "THE BOUNDARY AREA IS:   ";BDAREA
    ELIF (INFO[1])[1] = 4 THEN
      IF INFO[4] <> 0 THEN
        GPLDRAW"WHITE";"SOLID";(PATHBOUNDARY KEYNUM)
        PAREA:=BOUNDARYAREA (PATHBOUNDARY KEYNUM)
        "THE PATH AREA IS:   ";PAREA
      ELSE
        "PATH HAS NO WIDTH .... 0 AREA"
      ENDIF
    ENDIF
    ENDIF
  ENDIF
UNTIL CRD = ""
ENDDO

ENDSUB
```

Syntax:

var := Array INDICESOF Array (CR)
or
var := Vector INDICESOF Vector (CR)

or

var := (nX2 array) INDICESOF Array (CR)
or
var := (nX2 array) INDICESOF (nX2 array) (CR)

Function:

INDICESOF finds all occurrences of elements in the right argument which exist in the left argument. It assigns the placement of each occurrence to the variable. This function is particularly useful for finding more than one placement of an X or Y coordinate. It is also very useful to find a particular world within a text string.

The function is similar to the INDEXOF GPLII operator which finds only one occurrence of the right argument in the left argument. INDEXOF will also not find a specified word within a text string, but the occurrence of each letter within the particular word.

Example:

```
DYADIC FUNCTION RESULT:=SUB LOCATE WHOLE
LOCAL RESULT; SUB; WHOLE

RESULT:=SUB INDICESOF WHOLE
IF RESULT <> "" THEN
  "LOCATED IN POSISTION(S):   ",(CFORMAT RESULT)
ELSE
  "NOT LOCATED"
ENDIF

ENDSUB

? LOAD"LOCATE"
? ANS:=23 24 15 -1 23 18 34 LOCATE 23
LOCATED IN POSITION(S):  1 5
?
? ANS:="THIS IS A TEST STRING OF TEXT" LOCATE "TEST"
LOCATED IN POSITION(S): 11
? ANS
  11
?
```

```
? VAR1:="THIS IS A TEXT STRING"
? VAR2:="IS"
?
? ANS:=VAR1 LOCATE VAR2
LOCATED IN POSITION(S): 3 6
? ANS
 3 6
?

? GET 0 0
? CRDS:=COORDS
? CRDS
    0.       0.
   10.       0.
   10.      10.
    0.      10.
    0.       0.
?
? NUM:=CRDS LOCATE 0 0
LOCATED IN POSITION(S): 1 9
? NUM
1 9
?
```

TYPEOF = Integer
LENGTH = 1
RANK = 1
SIZE = Equal to the number of times a searched item is found. If none are found = 0.
SHAPE = If a searched item is found, the shape = SIZE. If no searched items are found, SHAPE = 0.

Syntax:

Var := (CE) INSIDEBOUNDARY (n by 2 array) (CR)

or

Var := (CE) (CE) (CE) . . . (CE) INSIDEBOUNDARY (n by 2 array) (CR)

Guidelines:

1. An array of coordinates n by 2 must be assigned as the right argument.
2. The entered coordinates for the right argument must form a legal boundary shape, with 200 or less vertices.

Function:

Assigns the variable a value of 1 if the coordinates(s) specified in the left argument are within the inside area of the boundary specified by the right argument. Assigns the variable with a value of 0 if the coordinate(s) specified in the left argument are outside of the boundary specified by the right argument. Assigns the variable with a value of -1 if the coordinate(s) specified in the left argument are exactly on a line segment of a Boundary.

Program Information:

TYPEOF = Integer
SIZE = Size of left argument divided by 2
LENGTH = 1
SHAPE = Same as SIZE

Example:

```
? 22. 27. INSIDEBOUNDARY AA
0
?
```

or

```
? 11. 28. 9. 22. 15. 18. INSIDEBOUNDARY AA
0 1 1
?
```

Where AA is an n by 2 array.

Program example:

```
DYADIC FUNCTION RESULT:=CRD INSIDE CRDLIST
LOCAL RESULT; CRD; CRDLIST

RESULT:=CRD INSIDEBOUNDARY CRDLIST
IF RESULT = 1 THEN
  "COORDINATE INSIDE BOUNDARY"
ELSE
  "COORDINATE <BRON>NOT <BROFF>INSIDE BOUNDARY"
ENDIF

ENDSUB

? LOAD"INSIDE"
?
? GET 0. 44.
? ITEM
You Are Editing Item #9
It is a Boundary (9 Points So Far)
layer 35, Datatype 1
You Are In Orthint Mode (Horizontal-First)
Number :  6
Value  : VDD
? CRDS:=GEDCOORDS
?
? CRDS
 33. 44.
  0. 44.
  0. 50.
 46. 50.
 46. 44.
 39. 44.
 39. 33.
 33. 33.
 33. 44.
?
? PNT:= 15.5 47. INSIDE CRDS
COORDINATE INSIDE BOUNDARY
? PNT
 1
?
? PNT:= 3. 47.
? ANS:=PNT INSIDE CRDLIST
COORDINATE INSIDE BOUNDARY
?
? PNT:= -5. 47.
? ANS:=PNT INSIDE CRDLIST
COORDINATE NOT INSIDE BOUNDARY
?
```

This next program will locate and identify any "contacts" which are not enclosed by "metal".

```
NILADIC PROCEDURE ENCLOSEDBY
EXTERNAL MONADIC PROCEDURE PLEXIT
LOCAL RESULT; CONTACT; METAL
EXTERNAL N; CRDLIST; INFO; KEYNUM; KEYS
EXTERNAL PLEXHEAD;ANS;KEY_NUMS;CONT_CRDS;MET_INFO;MET_CRDS
EXTERNAL X;CHECK;CONT_LAY;LAY

20 RESHAPE "<CR>"

VIEW
PLEXMODE 0
LOAD"PLEXIT"
KEY_NUMS:=""
CONT_LAY:=EXPINPUT"PLEASE ENTER CONTACT LAYER NUMBER:   "
BEGIN:
IF CONT_LAY <> "" THEN
  KEYS:=MSELECT 3;CONT_LAY
  IF KEYS <> "" THEN
     IF (SIZE KEYS) >= 1 THEN
       N:=0
       DO
       N:=N + 1
       KEYNUM:=KEYS[N]
       INFO:=GETEL KEYNUM
       IF (SIZE INFO[1]) = 4 THEN
         PLEXHEAD:=(INFO[1])[4]
       ELSE
         ""
         "(((ERROR))) CONTACT IS NOT PLEXED."
         ANS:=TEXTINPUT"DO YOU WISH TO ESTABLISH A PLEX (Y):   "
         IF (ANS = "Y") OR (ANS = "") THEN
           PLEXIT KEYNUM
           GOTO BEGIN
            ELSE
            GOTO FINI
         ENDIF
       ENDIF
         CONT_CRDS:=INFO[6]
         MET_INFO:=GETEL PLEXHEAD
         MET_CRDS:=MET_INFO[6]
         CHECK:=CONT_CRDS INSIDEBOUNDARY MET_CRDS
         IF (0 IN CHECK) OR (-1 IN CHECK) THEN
           KEY_NUMS:=KEY_NUMS , KEYNUM
         ENDIF


     UNTIL N = (SIZE KEYS)
     ENDDO

     ELSE
      "NO CONTACTS FOUND IN THIS STRUCTURE"
  ENDIF
ENDIF

ENDIF
IF KEY_NUMS <> "" THEN
  IDCLEAR
  N:=0
  DO
  N:=N + 1
  IDADD KEY_NUMS[N]
```

```
  UNTIL N = (SIZE KEY_NUMS)
  ENDDO
10 RESHAPE "<CR>"
"ERROR CONTACTS FOUND ... PLEASE SEE DISPLAY"
"THEY HAVE BEEN IDENTIFIED ... USE IDCLEAR TO CLEAR"
SETVIEW (DATAEXTENT IDKEYS)
ENDIF

PLEXMODE 1

FINI:
ENDSUB
```

This next program is a subroutine that is used by the GPLII program ENCLOSEDBY. If ENCLOSEDBY finds "contacts" that have not been plexed to the surrounding "metal", it prompts the user to indicate the proper "metal" boundary for enclosure.

```
MONADIC PROCEDURE PLEXIT KEY_NUM
LOCAL KEY_NUM
EXTERNAL MET_LAY;ANS;MET_KEY;WIN;NEW_WIN;MET_CRD;SL
EXTERNAL CONT_LAY



SL:=SLAYER
MET_LAY:=EXPINPUT"PLEASE ENTER METAL LAYER NUMBER:"
IF MET_LAY <> "" THEN
  SLAYER (MET_LAY, CONT_LAY)
ENDIF

ID KEY_NUM
WIN:=(DATAEXTENT IDKEYS)
SETVIEW WIN
ZOOM .5

FIND_METAL:
MET_CRD:=EXPINPUT"PLEASE INDICATE METAL TO PLEX TO (CE):  "
IF MET_CRD <> "" THEN
  MET_KEY:=CEKEY MET_CRD
  GET MET_KEY
  ANS:=TEXTINPUT"IS THIS THE CORRECT METAL (Y):  "
  IF (ANS = "Y") OR (ANS = "") THEN
    PLEX MET_KEY
    PUT
    IDCLEAR
    GET KEY_NUM
    PLEX MET_KEY
    PUT
  ELSE
    PUT
    GOTO FIND_METAL
  ENDIF
ENDIF
```

```
SLAYER SL
IDCLEAR
VIEW

ENDSUB
```

INTERRUPTMODE EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION

Syntax:

INTERRUPTMODE 1 or 0

Function:

This GPLII command toggles the interrupt capability for the GDSII workstation. INTERRUPTMODE 0 will disable the interrupt key. INTERRUPTMODE 1 will enable the interrupt key. This command may only be used within a GPLII program.

Example:

```
NILADIC PROCEDURE PASSWORD
LOCAL PASS

INTERRUPTMODE 0

DO
PASS:=TEXTINPUT" ENTER PASSWORD:   "
UNTIL PASS = "SOME PASSWORD"
ENDDO

INTERRUPTMODE 1

ENDSUB
```

In this example, the user is prompted to enter a password. The interrupt capability has been disabled to disallow the user to interrupt the program until the proper password has been entered.

Syntax:

KEYMARK key number (CR)
or
KEYMARK vector of key numbers (CR)

Function:

Puts a mark at the first point or origin specified key number. The mark may be removed using the REMOVE command or redrawing the screen.

Example:

```
NILADIC PROCEDURE FIRSTPNT
EXTERNAL CRD; KEY_NUM

CRD:=EXPINPUT"PLEASE INDICATE ELEMENT:   "
IF CRD <> "" THEN
  KEY_NUM:=@KEY CRD
  KEYMARK KEY_NUM
ENDIF

ENDSUB
```

This program will mark the element that is found closest to the coordinate entered.

```
NILADIC PROCEDURE WINMARK
EXTERNAL KEY_NUMS; LL_CRD; UR_CRD

LL_CRD:=EXPINPUT"PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   "
UR_CRD:=EXPINPUT"PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   "

KEY_NUMS:=MSELECT "";"";"";"";"";LL_CRD,UR_CRD
IF KEY_NUMS <> "" THEN
  KEYMARK KEY_NUMS
ENDIF

ENDSUB
```

This program will find all the elements that are viewable and selectable within the specified window and mark either their first point or their origin.

Syntax:

var := LINEINTERSECT CE1 CE2 CE3 CE4 . . . CEn CEm (CR)

Where n and m are X and Y coordinates, respectively.

or

var :=LINEINTERSECT Matrix1, Matrix2, Matrix3, Matrix4

Function:

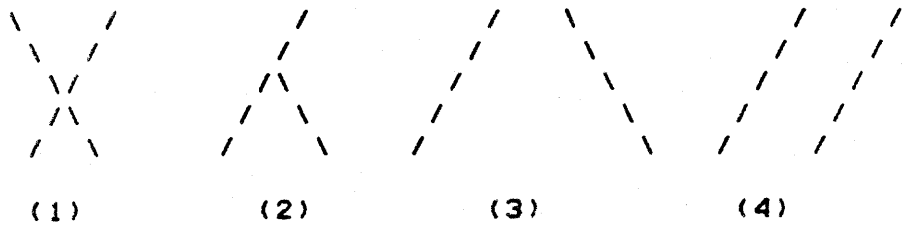
This function returns a list of one vector and one scalar or one matrix and one scalar, stating whether or not two or more line segments intersect. If they intersect, it also states:

1) The point of intersection
2) Whether or not the point(s) of intersection occur on the line segments themselves

The point of intersection is returned even if that point does not occur on the line segments themselves. Intersection points for more than one pair of line segments may be determined concurrently. In this case, matrices built from a list of line segment coordinates may be used as LINEINTERSECT arguments. These matrices may be created using either RESHAPE or matrix concatenation. Four nX2 matrices or one nX8 matrix may be used, where n is an integer from 1 to 32767, representing the number of rows in the matrix.

If the lines are parallel, i.e., they do not intersect, the "intersection coordinate" listed appears as 0 0, but has no real value.

The scalar returned by LINEINTERSECT is a flag, (1, -1 or 0), stating whether or not the two line segments intersect, or would intersect if extended. The flag is 1 if the line segments intersect at a point common to both of them, (as in figures (1) and (2)0, and -1 if the lines WOULD intersect if extended (figure 3). The flag is 0 if the line segments are parallel (figure 4).



Program Information:

TYPEOF = LIST
SIZE = null
LENGTH = 2
SHAPE = null

Example:

```
? LINEINTERSECT 2 3 -5 104 32 32 10 59 (CR)
-2.9857353664535 74.9379388588294 -1
?
```

Program example:

```
;PROGRAM NAME:          AUTODONUT
;DATE:                  2/7/85
;CREATED BY:            ERIC ANDERSON
;RELEASE:               4.1.1
;REVISION:              A
;SUBROUTINES:           OPENSTRUCT
;PURPOSE:               TO CREATE DONUT W/O USING REDO

NILADIC PROCEDURE AUTODONUT

EXTERNAL YCRS;XCRS;INT
EXTERNAL CRD; KEY_NUM; INFO; CRDS; PTS
EXTERNAL LL; UR; LR; UL; RECT
EXTERNAL RECT_SEG; NEW_CRDS; N; BD_SEG; CROSS
EXTERNAL CNT; BEG_CRDS; END_CRDS
EXTERNAL PNTS;ANS;PNT;LAY;DT

;EXTERNAL NILADIC PROCEDURE OPENSTRUCT

;LOAD"OPENSTRUCT"
;OPENSTRUCT
""

DO
END_CRDS:=""
BEG_CRDS:=""
NEW_CRDS:=""
CRD:=EXPINPUT"INDICATE BOUNDRY W/CE:   "
IF CRD <> "" THEN
KEY_NUM:= CEKEY CRD

INFO:= GETEL KEY_NUM
CRDS:= INFO [6]
PTS:= ( SHAPE CRDS ) [1] - 1
NLOOP:
LL:= EXPINPUT "LOWER LEFT:   "
UR:= EXPINPUT "UPPER RIGHT:   "

DRAWBOX "CORNER"; LL;UR
DRAWPROMPT "OK? (Y):";"L"
ANS:=TEXTINPUT"RECTANGLE OKAY? (Y):   "
IF (ANS <> "") AND (ANS <> "Y") THEN
ERASEPROMPT"OK? (Y):";"L"
ERASEBOX"CORNER";LL;UR
GOTO NLOOP
ENDIF
```

```
ERASEPROMPT"OK? (Y):";"L"
LR:= ( UR [1], LL [2] )
UL:= ( LL [1], UR [2] )
RECT_SEG:= ( LL,LR )
RECT:=LL,UL,UR,LR


NEW_CRDS:= ""

N:= 0

DO

N:= N+1
BD_SEG:= ( CRDS [N; ], CRDS [(N+1); ])
CROSS:= LINEINTERSECT BD_SEG, RECT_SEG
INT:=,(CROSS[1])
ANS:=,(CROSS[2])
UNTIL ANS = #1
ENDDO

IF (ANS = #1) AND (N > 1) THEN
PNT:=N,2
BEG_CRDS:=,(PNT TAKE CRDS),INT
IF (LL[1] < INT[1]) THEN
  NEW_CRDS:=LL,UL,UR,LR,INT
  PNTS:= -(PTS - (N - 1)),2
  END_CRDS:=,(PNTS TAKE CRDS)
  NEW_CRDS:=BEG_CRDS,NEW_CRDS,END_CRDS
ELIF (LL[1] > INT[1]) THEN
  NEW_CRDS:=LR,UR,UL,LL,INT
  PNTS:= -(PTS - (N - 1)),2
  END_CRDS:=,(PNTS TAKE CRDS)
  NEW_CRDS:=BEG_CRDS,NEW_CRDS,END_CRDS
ENDIF
ENDIF


DATADELETE"Y";KEY_NUM
LAY:=INFO[2]
DT:=INFO[3]
BOUNDARY
LAYER LAY
DATATYPE DT
ORTH INT
CE NEW_CRDS
PUT

ENDIF

UNTIL (CRD = "")
ENDDO

ENDSUB
```

Syntax:

MASKFREE maskfile (CR)

or

MASKFREE maskfile; maskfile; . . . (CR)

Guidelines:

1. This command should be issued as soon as a mask file is no longer needed in a GPLII DRC program.

2. Because of disk space considerations, a GPLII DRC program should use the fewest mask files possible at one time.

Function:

Deletes the specified mask file(s) from the disk.

Restrictions:

MASKFREE can only be executed in the background as part of a GPLII DRC program.

Example:

```
NILADIC PROCEDURE DRCTEST
:
MASK FILE UN_LAY1
MASK FILE NEWUN_LAY1
MASK FILE UN_LAY2
MASK FILE NEWUN_LAY2
MASK FILE UN_LAY3
MASK FILE NEWUN_LAY3
UN_LAY1:=INPUTMASK 1;(IOTA 0 63)
NEWUN_LAY1:=UNDERSIZE .75;UN_LAY1
OUTPUTMASK NEWUN_LAY1;11;"JOBLOG"; 0;200
:
MASKFREE UN_LAY1
MASKFREE NEWUN_LAY1
:
UN_LAY2:=INPUTMASK 2;(IOTA 0 63)
NEWUN_LAY2:=UNDERSIZE 1.;UN_LAY2
OUTPUTMASK NEWUN_LAY2;12;"JOBLOG"; 0;200
:
MASKFREE UN_LAY2
MASKFREE NEWUN_LAY2
:
UN_LAY3:=INPUTMASK 3;(IOTA 0 63)
NEWUN_LAY3:=UNDERSIZE .375;UN_LAY3
OUTPUTMASK NEWUN_LAY3;13;"JOBLOG"; 0;200
```

```
:
MASKFREE UN_LAY3
MASKFREE NEWUN_LAY3
:
:
ENDSUB
```

Monadic Syntax:

newmaskfile := MASKRESTORE "filename" (CR)

Function:

Assigns the specified CDOS disk file name (created by MASKSAVE) to the new mask file. MASKRESTORE can only be used once with a given disk file.

MASKRESTORE can only be executed in the background as part of a GPLII DRC program.

Program Information:

TYPEOFF = Mask file
SIZE = Variable

Syntax:

MASKSAVE maskfile; "filename"(CR)

Guidelines:

1. The CDOS file name can contain a maximum of 10 characters plus a two character extension.

2. MASKSAVE is useful for check pointing and limiting the amount of time required by individual DRC runs.

Function:

Saves the specified mask file for another DRC run by creating a backup in a separate CDOS disk file with the file name specified. MASKSAVE does not delete the mask file from the GPLII DRC program.

See MASKFREE and MASKRESTORE.

Restrictions:

MASKSAVE can only be executed in the background as part of a GPLII DRC program.

Syntax:

```
var := MINMAX n1 n2 n3 n4 n5 . . . nn (CR)
Minimum Maximum
?
```

Where n is a scalar, vector or matrix

Function:

This GPLII function returns the minimum and maximum values present for a given set of scalars, vectors or matrices.

Example:

```
? MINMAX -28 29.3 -108 72 3682 21 (CR)
-108 3682
?
```

Syntax:

OUTPUTCHANGE library; structure(CR)

Guidelines:

1. The designated library must exist.

2. If the designated structure does not exist, the DRC program will create it.

Function:

Specifies the library and structure which will contain the results of the DRC run. OUTPUTCHANGE overrides the library and structure information previously specified with JOBCREATE or another OUTPUTCHANGE in the GPLII DRC program.

OUTPUTCHANGE enables the results of different DRC commands in a single GPLII DRC program to be routed to different structures and/or libraries.

Restrictions:

OUTPUTCHANGE can only be executed in the background as part of a GPLII DRC program.

Example:

```
OUTPUTCHANGE IC5;IC5ERR <CR>
CONTACTAREA := CHECKAREA "LT";16;CONTACT<CR>
OUTPUTCHANGE IC5;IC5PGDATA <CR>
NEWG := OVERSIZE 5; GATE <CR>
LESSP := UNDERSIZE 3; POLY <CR>
OUTPUTMASK NEWG;0 <CR>
OUTPUTMASK LESSP; 1 <CR>
```

The first OUTPUTCHANGE command in the GPLII DRC program fragment specifies that the results of the CHECKAREA function should be placed in structure IC5ERR in library IC5. Then the second OUTPUTCHANGE command specifies that the results of the OVERSIZE and UNDERSIZE functions should be placed in structure IC5PGDATA, also in library IC5.

Program example:

```
:PROGRAM NAME: ROUNDIT.GS
:CREATED BY: CHRIS MARTIN
:DATE: 3-20-85

:THIS PROGRAM IS A RECURSIVE DRC PROGRAM FOR ROUNDMASK
NILADIC PROCEDURE ROUNDIT
MASK FILE OLD_MASK; RMASK
LOCAL N;  CELL_NAME; STRUCT_OUT
EXTERNAL NN


OPENLIB"GDSII:CHRISLIB.DB"
"5"
CELL_NAME:=STRUCLIST"-"      :Obtaining structure names

N:=0
DO
N:=N + 1
STRUCT_OUT:=CELL_NAME[N]
OUTPUTCHANGE "GDSII:ERRCHRIS.DB";STRUCT_OUT        :Enter output library
                                                   :name inside quotes
OUTPUTVIEW (,DATAEXTENT)        :Set window to include all data

LEVEL 0
SKIND"BDPCPB"
OSTRUCT CELL_NAME[N]
         NN:=0
         DO
         NN:=NN + 1
         OLD_MASK:=INPUTMASK NN; IOTA 0 63
         RMASK:=ROUNDMASK .005; OLD_MASK
         OUTPUTMASK RMASK; NN
         MASKFREE RMASK
         MASKFREE OLD_MASK
         UNTIL NN = 9
         ENDDO
UNTIL N = (LENGTH CELL_NAME)
ENDDO

ENDSUB         :ROUNDIT
```

Syntax:

OUTPUTVIEW (CE) (CE/CR)

or

OUTPUTVIEW (CR)

Function:

Defines a window in the structure from which INPUTMASK will extract data for the DRC run. If no coordinates are entered, a window enclosing all of the data in the open structure will be defined.

Perhaps its most useful purpose is to describe the window for a foreground plot, using the lot option FGEXP.

By entering OUTPUTVIEW monadically prior to JOBCREATE, the command causes the jobcreate to use the specified window as the default for the job. This allows the user to plot a section of the current view window rather than the entire view window.

It is advised to VIEW the current open structure after using the OUTPUTVIEW command before editing continues.

Example:

```
?
? OUTPUTVIEW  -5.5 -9.  36. 29.
? JOBCREATE
Job Type (XCLI): OVERSA8242
Priority Class (B):
Jobname (OVERSA8242):
Plot Options (PLTR): FGEXP
Exploder Output File:
Plot Window
  CE1 = ( -5.5 , -9. )
  CE2 = ( 36. , 29. )
Scale Factor (27350.): 2000

Plot Size Will be        3.2677 By     2.9921 Inches
Layers are  9 25
Datatypes are  0 1
Assign fillcodes to layers
     L9 (1):
     L25 (2):
Run, Save, or Abort (RUN):
```

```
?
? OUTPUTVIEW
? AA:=VIEWWINDOW
? ;AA
 -39.446 -30.166
  40.446  33.166
?
? JOBCREATE
Job Type (XCLI): OVERSA8242
Priority Class (B):
Jobname (OVERSA8242):
Plot Options (PLTR): FGEXP
Exploder Output File:
Plot Window
  CE1 = ( -39.446 , -30.166 )
  CE2 = ( 40.446 , 33.166 )
Scale Factor (16410.):

Plot Size Will be      51.6153 By   40.9165 Inches
Layers are  9 25
Datatypes are  0 1
Assign fillcodes to layers
    L9 (1):
    L25 (2):
Run, Save, or Abort (RUN):
?
```

Syntax:

```
AA := CEKEY (CE/CR)
              AA (CR)
              Path_Key_Number
              BB := PATHBOUNDARY AA (CR)
              BB (CR)
```

Where AA is a path key number and BB (CR) returns an nX2 array containing the coordinates for the path boundary.

Guidelines:

1) GEDMODE must be OFF.
2) The coordinates of the path centerline, whose path boundary coordinates are to be returned, must be stored in an n by 2 array.
3) Path boundaries are dividied into separate matrices if they exceed more than 200 coordinates each.

Function:

Given a set of coordinates, this function returns a coordinate matrix for a path boundary only, excluding the coordinates for the path centerline. (See GETEL to obtain path centerline coordinates.)

If more than 200 coordinates for a path are used, multiple coordinate matrices of 200 or less coordinates each are returned in a list. The LENGTH of PATHBOUNDARY is the number of matrices into which a path boundary is divided. This number must be used to calculate the area of a path boundary.

PATHBOUNDARY may be used as an argument for the BOUNDARYAREA function, to calculate the area of a path element. (See BOUNDARYAREA.)

Example:

```
? AA := CEKEY 15. 25.
? AA
48367
? BB := PATHBOUNDARY AA
? BB
x
```

Where AA is an nX2 array of path centerline coordinates identified by a path key number, BB is the variable and x is the returned matrix or set of matrices assigned to BB.

NOTE: For GPLII program example, see BOUNDARYAREA.

Syntax:

PROPVALUE n (CR)
or
PROPVALUE n; key number (CR)

Function:

Returns the Property Value of Property Attribute number 'n'. PROPVALUE n (CR) returns the value of Property Attribute n in the current Item. PROPVALUE n; key number (CR) returns the value of property Attribute n in the database element whose key number corresponds to 'key number'. A null is returned if no Property Value is found.

When PROPVALUE is used as a monadic function, the value of Property Value n is returned to the variable as a character vector.

A Property Value may be assigned to a variable regardless of whether GEDMODE is on or off. A null vector is returned if no Property Value is found.

Example:

1. ? PROPVALUE 1(CR)
   R16
   ?
   Property Attribute 1 in the current Item has a value of R16.

2. ? PROPVALUE 1; 7583(CR)
   V16
   ?
   The database element whose key number is 7583 contains the value R16 for Property Attribute 1.

1. ? PROPVAL := PROPVALUE 1(CR)
   ?
   Assigns the value of property Attribute 1 in the current Item to PROPVAL.

2. ? PROPVAL :=PROPVALUE 1; 7583(CR)
   ?
   Assigns the value of Property Attribute 1 from the element whose database key number is 7583 to PROPVAL.

TYPEOF = Char or null
LENGTH = 1
SIZE = n Char (0 if null)
SHAPE = n Char (0 if null)

The following GPLII program will identify any user properties assigned to specified nodes. It will display the user property values both on the QTY and on the display screen.

```
NILADIC PROCEDURE IDNODEPROP
EXTERNAL LIB_NAME;VKINDS;SKINDS;VLAYERS;SLAYERS;VDTYPES
EXTERNAL NODE_LOCATION;CRD;LX;LY;UX;UY;LL_CRD;UR_CRD
EXTERNAL NODE_KEYS;KEY_INFO;KEY_CRD;U_STRING;U_INTEGER;NODE_KEY;NUM
EXTERNAL OPEN;STR_NAME;INFO;VNTYPES;WIN;OPTS
EXTERNAL SZ_NUM;N;X;PROPS;PROP
EXTERNAL SETV;TXT_CRD;XPT;YPT;SHOW_PROPS;SNTYPES;XX;SH_PROP


:PROGRAM NAME:  IDNODEPROP
:CREATED BY:    CHRIS MARTIN
:DATE:          NOVEMBER 27, 1984


START_PROGRAM:
PLEXMODE 0
""
""
SETV:=SETVIEW
TXT_CRD:=,SETV
XPT:=ABS (TXT_CRD[1] - TXT_CRD[3])
YPT:=ABS (TXT_CRD[2] - TXT_CRD[4])
XPT:=FLOOR (XPT * .04)
YPT:=FLOOR (YPT * .04)
TXT_CRD:=(TXT_CRD[1] + XPT),(TXT_CRD[4] - YPT)
SHOW_PROPS:=""

LIB_NAME:=OPENLIB
IF LIB_NAME <> "" THEN
  LIB_NAME:=LIB_NAME[1]
ENDIF

IF LIB_NAME = "" THEN
DO
LIB_NAME:=TEXTINPUT"LIBRARY NAME:   "
UNTIL LIB_NAME <> ""
ENDDO
OPENLIB LIB_NAME
ENDIF

OPTS:=OPTIONS
OPTIONS"SV"
VKINDS:=VKIND
SKINDS:=SKIND
VLAYERS:=VLAYER
SLAYERS:=SLAYER
VDTYPES:=VDTYPE
VNTYPES:=VNTYPE
```

```
SNTYPES:=SNTYPE
SNTYPE
VNTYPE
VKIND
SKIND
VLAYER
SLAYER
VDTYPE


!=========PROGRAM BEGIN=========================

"<BRON>PLEXMODE IS NOW OFF<BROFF>"
DO
REMOVE
""
""
       OPEN:=OSTRUCT
       IF OPEN = "" THEN
         DO
         STR_NAME:=TEXTINPUT"ENTER STRUCTURE NAME:   "
         UNTIL STR_NAME <> ""
         ENDDO
         OSTRUCT STR_NAME
        ENDIF

    DO
    ""
    NODE_LOCATION:=EXPINPUT"INDICATE ELEMENT WITH CE:   "
ERASETEXT SHOW_PROPS;"";"";"";TXT_CRD
       IF NODE_LOCATION <> "" THEN
         CRD:=NODE_LOCATION
         LX:=CRD[1] - 2.5
         LY:=CRD[2] - 2.5
         LL_CRD:=LX,LY

         UX:=CRD[1] + 2.5
         UY:=CRD[2] + 2.5
         UR_CRD:=UX,UY

         DRAWBOX "CORNER";LL_CRD;UR_CRD

          WIN:=LL_CRD,UR_CRD
         NODE_KEYS:=MSELECT 6;"";"";"";"";WIN
           IF NODE_KEYS <> "" THEN
             IF (SIZE NODE_KEYS) > 1 THEN
               X:=0
               DO
               X:=X + 1
               NODE_KEY:=NODE_KEYS[X]
               KEY_INFO:=GETEL NODE_KEY
               KEY_CRD:=KEY_INFO[6]

               DRAWCROSS KEY_CRD


SHOW_PROPS:=" "
XX:=8
DO
XX:=XX + 1
SH_PROP:=KEY_INFO[XX]
```

```
IF SH_PROP <> "" THEN
  IF (TYPEOF SH_PROP) <> "CHAR" THEN
    SH_PROP:=CFORMAT SH_PROP
  ENDIF
SHOW_PROPS:=SHOW_PROPS,"  ",SH_PROP
ENDIF
UNTIL XX = 11
ENDDO
!=========================================================

DRAWTEXT SHOW_PROPS;"L";"";"";TXT_CRD


               IF (LENGTH KEY_INFO) >= 11 THEN
                DRAWPROMPT"USER PROPERTY FOUND";"B"
                IF (KEY_INFO[9]) <> "" THEN
                  DRAWPROMPT"(USTRING)";"L"
                  U_STRING:=PROPVALUE 127;NODE_KEY
                  ""
                  "<BRON><TAB>USTRING PROPERTY VALUE ===>  ",U_STRING,"<BROFF>"
                ENDIF

                IF (KEY_INFO[10]) <> "" THEN
                  DRAWPROMPT"(UINTEGER)";"R"
                  U_INTEGER:=PROPVALUE 126;NODE_KEY
                  ""
                  "<BRON><TAB>UINTEGER PROPERTY VALUE ===>  ",U_INTEGER,"<BROFF>"
                ENDIF

                IF (KEY_INFO[11]) <> "" THEN
                  DRAWPROMPT"(USER DEFINED)";"R"
                  NUM:=(KEY_INFO[11]) INDICESOF "["
                   SZ_NUM:=SIZE NUM
                   IF (SZ_NUM) > 1 THEN
                     PROPS:="<TAB>"
                     N:=0
                     DO
                     N:=N + 1
                     INFO:=KEY_INFO[11]
                     PROP:="<TAB>",INFO[IOTA ((NUM[N]), ((NUM[(N + 1)]) - 1))]
                     PROPS:=PROPS,"<CR>",PROP
                     UNTIL N = (SZ_NUM - 1)
                     ENDDO
                     PROPS:=PROPS,"<CR><TAB>",(INFO[IOTA ((NUM[(N + 1)]),^
                     (SIZE INFO))])
                   ELSE
                     PROPS:="<TAB>",KEY_INFO[11]
                   ENDIF
                 ""
                  "<BRON><TAB>USER DEFINED PROPERTY AS FOLLOWS:<BROFF>"
                  ""
                  "<BRON>",PROPS,"<BROFF>"
                 ELSE
                  "NO USER PROPERTIES FOUND"
                 ENDIF



           ELSE
            DRAWPROMPT"NO USER PROPERTIES FOUND";"B"
           ENDIF
```

```
        ERASECROSS KEY_CRD
          UNTIL X = (SIZE NODE_KEYS)
          ENDDO

        ELSE
                KEY_INFO:=GETEL NODE_KEYS
                KEY_CRD:=KEY_INFO[6]

                DRAWCROSS KEY_CRD
SHOW_PROPS:=" "
XX:=8
DO
XX:=XX + 1
IF (LENGTH KEY_INFO) >= XX THEN
SH_PROP:=KEY_INFO[XX]
IF SH_PROP <> "" THEN
  IF ((TYPEOF SH_PROP) <> "CHAR") AND ((TYPEOF SH_PROP) <> "NULL") THEN
    SH_PROP:=CFORMAT SH_PROP
  ENDIF
SHOW_PROPS:=SHOW_PROPS,"  ",SH_PROP
ENDIF
ENDIF
UNTIL XX = 11
ENDDO
;=========  2  ================

DRAWTEXT SHOW_PROPS;"L";"";"";TXT_CRD

               IF (LENGTH KEY_INFO) >= 11 THEN
                DRAWPROMPT"USER PROPERTY FOUND";"B"
                IF (KEY_INFO[9]) <> "" THEN
                  DRAWPROMPT"(USTRING)";"L"
                  U_STRING:=PROPVALUE 127;NODE_KEYS
                  ""
                  "<BRON><TAB>USTRING PROPERTY VALUE ===>  ",U_STRING,"<BROFF>"
                ENDIF

                IF (KEY_INFO[10]) <> "" THEN
                  DRAWPROMPT"(UINTEGER)";"R"
                  U_INTEGER:=PROPVALUE 126;NODE_KEYS
                  ""
                  "<BRON><TAB>UINTEGER PROPERTY VALUE ===>  ",U_INTEGER,"<BROFF>"
                ENDIF

                IF (KEY_INFO[11]) <> "" THEN
                  NUM:=(KEY_INFO[11]) INDICESOF "["
                   SZ_NUM:=SIZE NUM
                   IF (SZ_NUM) > 1 THEN
                     PROPS:="<TAB>"
                     N:=0
                     DO
                     N:=N + 1
                     INFO:=KEY_INFO[11]
                     PROP:="<TAB>",INFO[IOTA ((NUM[N]), ((NUM[(N + 1)]) - 1))]
                     PROPS:=PROPS,"<CR>",PROP
                     UNTIL N = (SZ_NUM - 1)
                     ENDDO
                     PROPS:=PROPS,"<CR><TAB>",(INFO[IOTA ((NUM[(N + 1)]),^
                     (SIZE INFO))])
                   ELSE
```

```
                    PROPS:="<TAB>",KEY_INFO[11]
                  ENDIF
                 ""
                  "<BRON><TAB>USER DEFINED PROPERTY AS FOLLOWS:<BROFF>"
                  ""
                  "<BRON>",PROPS,"<BROFF>"
                 ELSE
                  "NO USER PROPERTIES FOUND"
                 ENDIF

        ENDIF


           ENDIF

          ELSE
           ""
           "<BRON><BEL>NO ELEMENTS FOUND<BROFF>"
         DRAWPROMPT "NO ELEMENTS FOUND";"B"
          ENDIF

         ENDIF
ERASEPROMPT"(USTRING)";"L"
ERASEPROMPT"(UINTEGER)";"R"
ERASEPROMPT"USER PROPERTY FOUND";"B"
ERASEPROMPT"NO ELEMENTS FOUND";"B"
ERASEPROMPT"NO USER PROPERTIES FOUND";"B"

        UNTIL NODE_LOCATION = ""
        ENDDO


REMOVE
WINOPTIONS OPTS
VKIND VKINDS
SKIND SKINDS
VLAYER VLAYERS
SLAYER SLAYERS
VDTYPE VDTYPES
VNTYPE VNTYPES
SNTYPE SNTYPES
PLEXMODE 1
ENDSUB
```

Syntax:

var := REVERSE array

(array = vector or matrix)

Function:

This monadic function reverses the order of elements in a vector, or reverses the column order in an array.

Example:

```
? AA:= 1 2 3 4 5 6 7 8
? REVERSE AA
      8. 7. 6. 5. 4. 3. 2. 1.
?
? BB:=2 3 RESHAPE (IOTA 6)
? BB
1 2 3
4 5 6
?
? REVERSE BB
3 2 1
6 5 4
?
```

Syntax:

var := n ROTATE array

or

var := n ROTATE [I] array

Function:

This is a dyadic function that, in the simple case, rotates n number of the elements of the argument. The index coordinate I indicates, for arrays only, the dimension to rotate within.

Note: At this time, the format "n ROTATE [I]" may only be used from within a GPLII program and NOT interactively.

Example:

```
? AA : = 1 2 3 4 5 6 7 8
? 3 ROTATE AA
   4. 5. 6. 7. 8. 1. 2. 3.
?
? #3 ROTATE AA
   6. 7. 8. 1. 2. 3. 4. 5.
?
? BB := 2 4 RESHAPE (IOTA 8)
? BB
1 2 3 4
5 6 7 8
?
? 1 ROTATE BB
2 3 4 1
6 7 8 5
?

? #1 ROTATE BB
4 1 2 3
8 5 6 7
```

Program example:

RESULT := 1 ROTATE [1] BB

RESULT will equal 5 6 7 8
1 2 3 4

In this example, the rows, rather than the columns have been rotated.

Syntax:

Niladic

```
ROUND (CR)
Enter number to round: (whole number)
Enter number of places past decimal point: (integer)
```

Monadic

```
Var := ROUND C; D (CR)
Z
```

Where Z is the value of C rounded to the decimal place specified by D.

Function:

Assigns the variable the value of the first argument, rounded to the decimal place specified by the second argument. The second argument must be an n integer whose value, when 1, designates that the first argument is to be rounded off to one place to the right of the decimal point. When the second argument is 2, the variable is assigned a value rounded to two places to the right of the decimal point; when it is 3, the variable is assigned a value rounded to three places to the right of the decimal point, etc.

Similarly, if the second argument is 0, the variable is assigned the value of the first argument, rounded to the one's decimal place. If the second argument is 01, the variable is assigned the value of the first argument rounded to the ten's decimal place, and if the value of the first argument is -2, the variable is assigned the value of the first argument, rounded to the hundred's decimal place, etc.

See also DROUND.

Restrictions:

Integers should be limited to a size of 8 characters maximum. Numbers out of range may cause ROUND to return unpredictable results and incorrect responses.

Program Information:

```
TYPEOF  = Real
SIZE    = Same as SIZE of first argument
LENGTH  = 1
SHAPE   = Same as SHAPE of first argument
```

Example:

Niladic

```
? ROUND
Enter number to round: 7837.8395
Enter number of places past decimal point: 3
7837.8395
?
```

Monadic

```
ROUND 7837.8395; -3
7000
?
```

Syntax:

Var := SCALARINPUT "prompt"(CR)
or
Var := SCALARINPUT "prompt"; default(CR)
or
Var := SCALARINPUT "prompt"; default; minimum(CR)
or
Var := SCALARINPUT "prompt"; default; minimum; maximum(CR)

Function:

Prompts the user to enter a scalar numeric value, accepts a response from the input keyboard, and assigns the reponse value to the variable. It also provides for optional default, minimum, and maximum response values.

The first argument specifies the prompt to be displayed on the QTY. SCALARINPUT automatically appends a colon and two spaces to the prompt.

The second argument specifies a default value for the response. If the user inputs a null value, the default value will be assigned to the variable. The default will appear in the prompt enclosed in parenthesis.

The third and fourth arguments specify minimum and maximum allowable values for the user's response. It is possible to specify a maximum without specifying a minimum by making the third argument null (" ") while supplying a numeric value for the fourth.

Note: If a character value or any numeric values which are not within the specified range are input, the system will produce an error message and reprompt for input.

Example:

```
NILADIC PROCEDURE SCALTEST
EXTERNAL NUM; NUMS; LAYS

NUMS:=SCALARINPUT"PLEASE ENTER VIEW LAYER NUMBER"
NUM:=SCALARINPUT"PLEASE ENTER SELECT LAYER NUMBER";63
LAYS:=SCALARINPUT"PLEASE ENTER VIEW/SELECT DATATYPE";63;0;63


VLAYER NUMS
SLAYER NUM
SDTYPE LAYS
VDTYPE LAYS


"VIEW/SELECT DATATYPE:  ";LAYS
```

```
"VIEW LAYER:   ";NUMS
"SELECT LAYER:   ";NUM
ENDSUB
```

```
?SCALTEST
PLEASE ENTER VIEW LAYER NUMBER: 1
PLEASE ENTER SELECT LAYER NUMBER (63.): 44
PLEASE ENTER VIEW/SELECT DATATYPE (63.): 68
        Maximum Value is 63.
PLEASE ENTER VIEW/SELECT DATATYE (63.): -15
        Minimum Value is 0.
PLESE ENTER VIEW/SELECT DATATYPE (63.): 45
VIEW LAYER: 1.
SELECT LAYER: 44.
?
```

```
TYPEOF = REAL
LENGTH= 1
RANK   = 1
SIZE   = 1
SHAPE  = 1
```

Syntax:

var := vector TAKE array

Function:

This dyadic function returns as a result the number of values in "array" specified by "vector". The first argument size MUST equal the RANK of the second argument; i.e., if the second argument is an array, the first argument must be a SIZE 2 vector whose values do not exceed the corresponding dimensions of the second argument.

Example:

```
? AA := 1 2 3 4 5 6 7 8
? 3 TAKE AA
1. 2. 3.
?
? #3 TAKE AA
6. 7. 8.
?
? BB := 3 4 RESHAPE (IOTA 12)
? BB
1  2  3  4
5  6  7  8
9  10 11 12
?
? 1 2 TAKE BB
1 2
? 2 3 TAKE BB
1 2 3
5 6 7
? 3 4 TAKE BB
1  2  3  4
5  6  7  8
9  10 11 12
? 1 4 TAKE BB
1 2 3 4
? 3 1 TAKE BB
1
5
9
? #2 2 TAKE BB
5  6
9  10
?
```

Program Example:

For a program example please refer to the GPLII program.
AUTODONUT.GS located under the command LINEINTERSECT in this section.

Syntax:

var := TRANSPOSE array

Function:

This monadic function reverses the index coordinates defining "array". This function has NO effect on data structures less than RANK 2.

Example:

```
? AA := 3 5 RESHAPE (IOTA 15)
? AA
1  2  3  4  5
6  7  8  9  10
11 12 13 14 15
?
? BB := TRANSPOSE AA
? BB
1  6  11
2  7  12
3  8  13
4  9  14
5  10 15
?
```

Syntax:

VECTORCOUNT (CR)

or

VAR := VECTORCOUNT (CR)

Function:

Returns a number representing the number of vectors in the current display window. The count will also include the number of vectors found in the menu that is currently displayed. To receive an accurate account of the vectors in the display window, the command RSCREEN" " (CR) should be used.

When VECTORCOUNT is used as a niladic function, the result assigned to the variable will be an INTEGER2 scalar.

Example:

```
? BSTRUCT "TEST"
? RT
Creating a Boundary
Layer 32, Datatype 0
(Horizontal-first)
Ce 1: 0 0
Ce 2: 10 10
Ce 1: (CR)
?

? GET 0 0
? COORDS
ce[1] = (0., 0.)
ce[2] = (10., 0.)
ce[3] = (10., 10.)
ce[4] = (0., 10.)
ce[5] = (0., 0.)

? PUT
? VECTORCOUNT
2074 vectors

? RSCREEN " "
? VECTORCOUNT
   4 vectors
? VECS:=VECTORCOUNT
? VECS
  4

? RSCREEN "CALMAMHD"
? VECS:=VECTORCOUNT
? VECS
2074
```

```
TYPE OF = INTEGER2
LENGTH  = 1
SHAPE   = 1
SIZE    = 1
RANK    = 1
```

GPLII DEBUGGER

GDSII Release 5.1 introduces a set of GPLII enhancements which are intended to reduce the amount of time spent on program development and debugging. When an error is encountered in a GPLII program, error traceback information is displayed which includes the name of the routine in control at the time of the error and the corresponding line number within that routine. If the error occurs in a nested subroutine, the entire GPLII calling stack, including line numbers is displayed. To utilize the debugging capabilities, programs must be compiled (or recompiled) under Release 5.1 or later.

Related procedures are as follows:

ERRTRAP

Implements an error trapping facility that allows the regaining of control after a system error has occurred. The syntax is similar to GOTO. For example:

ERRTRAP label

Label may be any GPLII label that is defined in the program. When an error occurs, the current routine is tested to see if it has an ERRTRAP that is active. If so, an error message is printed and the program restarts at the label specified. If there are not ERRTRAPs in the routine or any of its calling routines, the error is fatal, a traceback message is printed and execution is aborted. If the source file (.GS) of the program is currently in the GDSII text editor buffer, a window is displayed and the line with the error is made the current line for editing. An ERRTRAP statement remains in effect until another ERRTRAP is encountered within the program or until the end of the routine. ERRTRAP without an argument removes the current ERRTRAP in force.

BREAKPOINT

Interrupts a program before the execution of any GPLII statement. After breakpoint is entered, a special command mode is entered. The word BREAKPOINT displays on the QTY and the symbol " ?" appears as the prompt. Breakpoint mode allows all local variables to be examined and altered for debugging. Execution may be continued from this point or redirected to another statement within the current routine. If the source file is currently in the GDSII text editor buffer, the line indicated as the breakpoint becomes the current line and the window is displayed.

BREAKPOINT is called interactively and may be used in any one of three ways:

Called niladically, BREAKPOINT prints out the name of the routine containing the breakpoint and the line number on which it appears. For example:

```
? BREAKPOINT (CR)
  Breakpoint at Line 11 in TEST
```

Called monadically, with two arguments, sets the breakpoint to the line specified. The first argument in the list is the program name and the second is the line number that is to be defined as the breakpoint.

For example:

```
? BREAKPOINT "TEST" ; 11
```

Called monadically with 0 as the argument, removes the current breakpoint.

The following procedures may be invoked only from BREAKPOINT mode.

PROCEED

Continues execution of the program until a fatal error occurs, a breakpoint is encountered, or the program is completed.

STEP

Allows the user to step through the program line by line, causing a newbreakpoint after each executable statement in the program. If the current routine is in the GDSII text editor buffer, the breakpoint line becomes the current line and the text window is displayed.

TOLINE

Redirects execution to a specified line in the program and continues the program until a fatal error occurs, a breakpoint is encounterd, or the program is completed.

Example:

```
?
? G
File Name: WINMARK.GS
     1  NILADIC PROCEDURE WINMARK
     2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
     3
? T
```

```
      1  NILADIC PROCEDURE WINMARK
      2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
      3
      4  LL_CRD:=EXPINPUT"PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:  "
      5  UR_CRD:=EXPINPUT"PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:  "
      6
      7  KEY_NUMS:=MSELECT "";"";"";"";"";LL_CRD,UR_CRD
      8  IF KEY_NUMS <> "" THEN
      9    KEYMARK KEY_NUMS
     10  ENDIF
     11
     12  ENDSUB
?
? WINMARK
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   3. 8.
PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   24.5 25.5
?
? BREAKPOINT"WINMARK";1
? WINMARK

>>>BREAKPOINT At Line 1 in GPL II Routine WINMARK
      1  NILADIC PROCEDURE WINMARK
      2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
      3
>? STEP

>>>BREAKPOINT At Line 4 in GPL II Routine WINMARK
      2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
      3
      4  LL_CRD:=EXPINPUT"PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:  "
      5  UR_CRD:=EXPINPUT"PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:  "
      6
>? STEP
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   2.5 8.5

>>>BREAKPOINT At Line 5 in GPL II Routine WINMARK
      3
      4  LL_CRD:=EXPINPUT"PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:  "
      5  UR_CRD:=EXPINPUT"PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:  "
      6
      7  KEY_NUMS:=MSELECT "";"";"";"";"";LL_CRD,UR_CRD
>? STEP
PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   24.5 28.5

>>>BREAKPOINT At Line 7 in GPL II Routine WINMARK
      5  UR_CRD:=EXPINPUT"PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:  "
      6
      7  KEY_NUMS:=MSELECT "";"";"";"";"";LL_CRD,UR_CRD
      8  IF KEY_NUMS <> "" THEN
      9    KEYMARK KEY_NUMS
>? STEP

>>>BREAKPOINT At Line 8 in GPL II Routine WINMARK
      6
      7  KEY_NUMS:=MSELECT "";"";"";"";"";LL_CRD,UR_CRD
      8  IF KEY_NUMS <> "" THEN
      9    KEYMARK KEY_NUMS
     10  ENDIF

? BREAKPOINT"WINMARK";1

>>>BREAKPOINT At Line 1 in GPL II Routine WINMARK
      1  NILADIC PROCEDURE WINMARK
      2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
      3
>? TOLINE 4
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   2.5 9.
```

```
PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   34. 38.

?
? BREAKPOINT"WINMARK";1
?
? BREAKPOINT
Breakpoint at Line 1 in WINMARK
?
? WINMARK

>>>BREAKPOINT At Line 1 in GPL II Routine WINMARK
     1  NILADIC PROCEDURE WINMARK
     2  EXTERNAL KEY_NUMS; LL_CRD; UR_CRD
     3
>? PROCEED
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   1.5 7.5
PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   44.5 35.
?
?
? BREAKPOINT"WINMARK";5
? BREAKPOINT
Breakpoint at Line 5 in WINMARK
? WINMARK
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   2.5 9.

>>>BREAKPOINT At Line 5 in GPL II Routine WINMARK
>? TOLINE 4
PLEASE ENTER LOWER LEFT WINDOW COORDIANTE:   -1. -1.

>>>BREAKPOINT At Line 5 in GPL II Routine WINMARK
>? STEP
PLEASE ENTER UPPER RIGTH WINDOW COORDINATE:   20. 22.

>>>BREAKPOINT At Line 7 in GPL II Routine WINMARK
>? PROCEED
?
```

WORKSTATION CONFIGURATION

When the GDSII user first sits down at a GDSII workstation, they must consider the current station configuration.

Every designer, digitizer or engineer has their own particular needs and requirements for station configuration. This flexibility of the workstation is a necessity.

A user may determine which layers, datatypes or element types may be selectable and/or visible according to their own personal needs. Color schemes may be set up for a particular design process. These are only a few of the station parameters which might be established by the user prior to any actual data entry. This "station setup" can involve a substantial amount of time if it must be performed everytime the user needs to rest the workstation parameters to reflect their own needs. For some time, Calma has provided the GDSII user with programs such as RESTART.GS which may be used to reset the station parameters to "Calma default". Another such program is CALMAMENU.GS, used to load a menu to the workstation and dfunction buttons on the keyboard.

While these programs have been beneficial to the user in the area of workstation configuration, they are in no way complete. The user must still set up those parameters which reflect their personal needs.

Both CALMAMENU and RESTART are GPLII programs. It is much more expedient, both for the user and the system, to be able to use these programs rather than having to type in all the commands that the programs execute. For this reason, GPLII can greatly enhance the process of setting up the GDSII workstation parameters.

Programs used to set up the user's workstations can range from very simple command automation type programs to very elaborate programs that may even write other GPLII setup programs to be used later.

This flexibility in programming allows the user to develop a program that is very specific to their needs.

In the following example, the setup program uses both RESTART.GS and CALMAMENU.GS as subroutines and then sets some other station parameters. This is an example of a very simple program. Consider, however, how many commands are being executed by the programs and what it would require to type them all in interactively.

```
NILADIC PROCEDURE MYSETUP
EXTERNAL NILADIC PROCEDURE CALMAMENU; RESTART
EXTERNAL N

:THIS PROGRAM WILL SET UP THE GDSII WORKSTATION

LOAD "CALMAMENU"
```

```
LOAD "RESTART"

RESTART
CALMAMENU

VLAYER 1-15 63
SLAYER 1-15 63
VKIND"BDPCPBTX"

N:=(-1)
DO
N:=N + 1
ITEM N
BOUNDARY
LAYER N
ORTHINT
HORIZFIRST
DATATYPE 0
SETDEFAULTS

UNTIL N = 7
ENDDO

ITEM 8
SREF
MAG 1
REFL "N"
ANGLE 0

ITEM 9
TEXT
ANGLE 0
MAG .5
FONT 0
TJUST"T";"L"

ITEM 0

GEDMODE 1
GSCALE .5
GRID 2

ENDSUB
```

GPLII can perform a wide range of tasks for the user in the area of workstation configuration. In the next example, the program T$WSETUP.GS may be executed by the user to actually generate a GPLII setup program. The user only needs to set up the station parameters. Next, the user may execute the program and it will "read" the current parameters and then write a GPLII program with a specified name. From then on the user has a setup program built specifically for their own needs.

```
NILADIC PROCEDURE T$WSETUP
EXTERNAL CHK;GPLPROG;IFARK;FILNAM;TEMPNAM
:CREATED BY:  TED PAONE


FILNAM := NAMESOPEN
IF FILNAM = "" THEN
  DO
    FILNAM := TEXTINPUT "Enter Program Name:  "
    UNTIL FILNAM <> "" ENDDO
  ELSE FILNAM := FILNAM[1]
  TEMPNAM := FILNAM[IOTA ((FILNAM INDEXOF ".") - 1)]
  FILNAM := TEXTINPUT ("Enter Program Name (<BRON>",TEMPNAM,"<BROFF>):  ")
  IF FILNAM = "" THEN FILNAM := TEMPNAM  ENDIF
:
  ENDIF
:
:CHANGE THIS
IF (FILEINFO (FILNAM,".GS")) <> "" THEN
  FDELETE (FILNAM,".GS")  ENDIF
:
GPLPROG := "NILADIC PROCEDURE ",FILNAM,"<CR>:",GTIME
GPLPROG := GPLPROG,"<CR>LOCAL IFARK<CR>"
:
CHK := TEXTINPUT "Colors, Line Styles, and Fills (<BRON>Y<BROFF>):  "
IF CHK = "" THEN CHK := "Y"  ENDIF
IF (,CHK)[1] IN ("Yy") THEN
:
:Color, Style, and Fill modes.
  IFARK := COLORBY
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"COLORBY 0<CR>"
    ELSE GPLPROG := GPLPROG,"COLORBY 1<CR>" ENDIF
:
  IFARK := STYLEBY
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"STYLEBY 0<CR>"
    ELSE  GPLPROG := GPLPROG,"STYLEBY 1<CR>"
    ENDIF
:
  IFARK := FILLBY
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"FILLBY 0<CR>"
    ELSE  GPLPROG := GPLPROG,"FILLBY 1<CR>"
    ENDIF
:
:Colors
IFARK := SETCOLORS
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[1],""";",(FORMAT IFARK[2])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[3],""";",(FORMAT IFARK[4])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[5],""";",(FORMAT IFARK[6])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[7],""";",(FORMAT IFARK[8])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[9],""";",(FORMAT IFARK[10])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[11],""";",(FORMAT IFARK[12])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[13],""";",(FORMAT IFARK[14])
GPLPROG := GPLPROG,"SETCOLORS """,IFARK[15],""";",(FORMAT IFARK[16])
:
:If there are too many layers to fit in one line of text within the program,
: ie. more than 32, the line must be split into 2.

  IFARK := RED
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
```

```
    GPLPROG := GPLPROG,"RED ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"RED ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"RED ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := BLUE
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"BLUE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"BLUE ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"BLUE ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := GREEN
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"GREEN ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"GREEN ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"GREEN ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := MAGENTA
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"MAGENTA ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"MAGENTA ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"MAGENTA ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := CYAN
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"CYAN ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"CYAN ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"CYAN ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := YELLOW
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"YELLOW ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"YELLOW ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"YELLOW ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := WHITE
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"WHITE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"WHITE ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"WHITE ",(FORMAT IFARK)  ENDIF
ENDIF
:
:Fills and flayers.
```

```
  IFARK := FILLA
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"FILLA ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"FILLA ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"FILLA ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := FILLB
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"FILLB ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"FILLB ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"FILLB ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := FILLC
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"FILLC ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"FILLC ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"FILLC ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := FILLD
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"FILLD ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"FILLD ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"FILLD ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := FLAYERON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"FLAYER ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"FLAYERON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"FLAYER ",(FORMAT IFARK)  ENDIF
ENDIF
:
:Line Types.
  IFARK := SOLID
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"SOLID ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"SOLID ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"SOLID ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := BROKEN
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"BROKEN ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"BROKEN ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
```

```
    GPLPROG := GPLPROG,"BROKEN ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := DASHED
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"DASHED ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"DASHED ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"DASHED ",(FORMAT IFARK)  ENDIF
ENDIF
:
  IFARK := DOTTED
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"DOTTED ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"DOTTED ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"DOTTED ",(FORMAT IFARK)  ENDIF
ENDIF
:
ENDIF
:
:
:Viewed Layers, types, and kinds.
CHK := TEXTINPUT "View Paramenters(<BRON>Y<BROFF>):  "
IF CHK = "" THEN CHK := "Y"  ENDIF
IF (,CHK)[1] IN ("Yy") THEN
  IFARK := VLAYERON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"VLAYER ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"VLAYERON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"VLAYER ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := VDTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"VDTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"VDTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"VDTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := VTTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"VTTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"VTTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"VTTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := VNTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"VNTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"VNTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
```

```
    GPLPROG := GPLPROG,"VNTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := VBTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"VBTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"VBTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"VBTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := VKINDON
IF IFARK <> "" THEN
  GPLPROG := GPLPROG,"VKIND """,IFARK ,"""<CR>"
ENDIF
:
  IFARK := MARKSIZE
  GPLPROG := GPLPROG,"MARKSIZE ",(FORMAT IFARK)
  IFARK := SLIMSIZE
  GPLPROG := GPLPROG,"SLIMSIZE ",(FORMAT IFARK)
  ENDIF
:
:SELECTABLES
CHK := TEXTINPUT "Selectable Parameters(<BRON>Y<BROFF>):  "
IF CHK = "" THEN CHK := "Y"  ENDIF
IF (,CHK)[1] IN ("Yy") THEN
  IFARK := SLAYERON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"SLAYER ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"SLAYERON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"SLAYER ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := SDTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"SDTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"SDTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"SDTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := STTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"STTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"STTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"STTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := SNTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"SNTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"SNTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"SNTYPE ",(FORMAT IFARK) ENDIF
```

```
ENDIF
:
  IFARK := SBTYPEON
IF IFARK <> "" THEN
  IF (SIZE IFARK) > 32 THEN
    GPLPROG := GPLPROG,"SBTYPE ",(FORMAT (IFARK[IOTA 32]))
    GPLPROG := GPLPROG,"SBTYPEON ",(FORMAT (IFARK[IOTA (33,(SIZE IFARK))]))
  ELSE
    GPLPROG := GPLPROG,"SBTYPE ",(FORMAT IFARK) ENDIF
ENDIF
:
  IFARK := SKINDON
IF IFARK <> "" THEN
  GPLPROG := GPLPROG,"SKIND """,IFARK ,"""<CR>"
  ENDIF
ENDIF
:
:MODES
CHK := TEXTINPUT "Modes (<BRON>Y<BROFF>):  "
IF CHK = "" THEN CHK := "Y"  ENDIF
IF (,CHK)[1] IN "Yy" THEN
  IFARK := RTDIGMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"RTDIGMODE 0<CR>"
    ELSE GPLPROG := GPLPROG,"RTDIGMODE 1<CR>"  ENDIF
:
  IFARK := DEFMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"DEFMODE 0<CR>"
    ELSE GPLPROG := GPLPROG,"DEFMODE 1<CR>"  ENDIF
:
  IFARK := GEDMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"GEDMODE 0<CR>"
    ELSE GPLPROG := GPLPROG,"GEDMODE 1<CR>"  ENDIF
:
  IFARK := INTCHECKMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"INTCHECKMODE 0<CR>"
    ELSE GPLPROG := GPLPROG,"INTCHECKMODE 1<CR>"  ENDIF
:
  IFARK := VMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"VMODE 0<CR>"
    ELSE GPLPROG := GPLPROG,"VMODE 1<CR>"  ENDIF
:
  IFARK := SNAPMODE
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"SNAPMODE 0<CR>"
    ELSE  IFARK := SNAPSIZE
    GPLPROG := GPLPROG,("SNAPMODE 1<CR>SNAPSIZE ",(FORMAT IFARK),"REDRAW<CR>")  ENDIF
  IFARK := AUTOHOLD
  IF IFARK = 0 THEN
    GPLPROG := GPLPROG,"AUTOHOLD 0<CR>"
    ELSE GPLPROG := GPLPROG,"AUTOHOLD 1<CR>"  ENDIF
ENDIF
:
:
:ADD ITEMS
:
:MENU
```

```
CHK := TEXTINPUT "MENU Paramenters (<BRON>Y<BROFF>):  "
IF CHK = "" THEN CHK := "Y"  ENDIF
IF (,CHK)[1] IN "Yy" THEN
:
:Buttom Menu
  IFARK := RMENU
  IF (LENGTH IFARK) = 5 THEN
    GPLPROG := GPLPROG,"RMENU """,IFARK[1],""";",(CFORMAT IFARK[2]),";",(CFORMAT IFARK[3])
    GPLPROG := GPLPROG,";",(CFORMAT IFARK[4]),";",(FORMAT IFARK[5])  ENDIF
:
:Menu Graphics
  IFARK := RSCREEN
  IF IFARK <> "" THEN
    GPLPROG := GPLPROG,"RSCREEN """,IFARK,"""<CR>"
  ENDIF
:
:Tablet
  IFARK := SETTAB
  IF IFARK <> "" THEN
    GPLPROG := GPLPROG,"SETTAB ",(FORMAT IFARK)  ENDIF
ENDIF
:
:
GPLPROG := GPLPROG,"<CR>ENDSUB<CR>"
"Compiling <BRON>",FILNAM,"<BROFF>."
(FILNAM,".GS") SAVE GPLPROG
GPL FILNAM;"N"
LOAD FILNAM
:
ENDSUB
```

Another area of workstation configuration that can enhance the users performance is that of writing a custom menu rather than using such menus as CALMAMENU.

In the following program, CUSMENU.GS, the user is lead through the menu building process by the program. The program then writes all the necessary files needed to create a custom menu.

```
:PROGRAM NAME:     CUSMENU.GS
:CREATED BY:       CHRIS MARTIN
:DATE:             JULY 13, 1984

:PURPOSE:    THIS PROGRAM WILL ALLOW THE USER TO CREATE
:            THEIR OWN CUSTOM MENU

:RELEASE:    5.0.1

:FILES NEEDED:     CALMAFONT.TX
:                  CUSMENU.DB
:                  CUSMENU.G*

:TO CREATE THE CUSMENU.DB LIBRARY:  MAKE A COPY OF THE CALMAMENU.DB
```

```
:LIBRARY WHICH CONTAINS A MENU FOR EACH TYPE OF SCREEN. DELETE ALL
:OF THE MENU GRAPHICS IN EACH STRUCTURE BUT KEEP THE GRID THAT
:DEFINES THE BUTTONS. MAINTAIN THE SAME STRUCTURE NAMES.


:AT THIS TIME, ONLY THE COORDINATE LISTS FOR THE MHD AND THE MRD TYPE
:SCREENS HAVE BEEN CALCULATED.



NILADIC PROCEDURE CUSMENU
EXTERNAL MONADIC FUNCTION CLEANUP
EXTERNAL NILADIC PROCEDURE RESTART
EXTERNAL MONADIC FUNCTION CFMT
EXTERNAL TAB_TYPE
EXTERNAL MENU_FILE;MN;CC;RC;COM_NUM;ROW_CNT;COL_CNT;IN_COL;IN_ROW
EXTERNAL N_COLS;N_ROWS;NEWCOM;GEN
EXTERNAL GENS;B_FILE;E_FILE;NEW_FILE
EXTERNAL MENU_LIB;INDI
EXTERNAL NEW_TEXT;SIGN
EXTERNAL  LIB_NAME;ANS;OP_LIB;LIB;CRDS;CRTS
EXTERNAL  COMS;S_TXT;TXT;TXT1;TXT2;TXT3;N;X;FIND;CRD
EXTERNAL  CIR_NUM;SIN_NUM;CIR_INFO;CIR_CRD
EXTERNAL C_CNT;R_CNT;CCNTS;RCNTS;COM_MANDS;XX;INCR;SPC
EXTERNAL MTXT;ETXT;BTXT;S_DSH;DSH



TAB_TYPE:=TABTYPE
IF TAB_TYPE = 3 THEN
LOAD "RESTART"
"STATION WILL BE RESTARTED FOR TALOS WEDGE TABLET"
RESTART
ENDIF
""
""
""
""
""
""
"        **************************************************"
""
"        THIS PROGRAM WILL ALLOW YOU TO CREATE A CUSTOM MENU"
"        <BRON>THIS PROGRAM ONLY SUPPORTS THE MHD AND MRD SCREEN<BROFF>"
"        **************************************************"
""
""
""
""
""
""
""
""
""
""
SLEEP 5
"*********************  RESTRICTIONS  ******************************"
""
"1.      IF YOU WISH TO DEFINE ONE OF THE MENU BUTTONS AS A CARRIAGE"
"        RETURN, YOU MUST TYPE IT IN AS FOLLOWS:"
"        C/R"
```

```
"        FOLLOWED BY A CARRIAGE RETURN"
""
"2.      IF YOU WISH TO LEAVE A BUTTON UNDEFINED, YOU MAY ENTER A"
"        CARRIAGE RETURN OR TYPE IN <QT>NONE<QT>"
""
""
""
""
""
SLEEP 10
"3.      COMMANDS MAY BE BROKEN INTO AS MANY AS 3 LINES; 4 CHARACTERS PER"
"        LINE. IF YOU WISH TO <QT>BREAK<QT> A COMMAND INTO MORE THAN ONE"
"        LINE, YOU MAY USE DASHES (I.E., DIS-PLAY-STAT). IF YOU DO NOT "
"        USE DASHES TO DEFINE THE BREAKS, THE PROGRAM WILL CREATE 4 CHAR-"
"        ACTER LINES."
""
"4.      COMMANDS MAY BE ABBREVIATED OR GPL PROGRAM NAMES MAY BE USED TO"
"        DEFINE A BUTTON. THE PROGRAM WILL ALERT YOU THAT THE COMMAND IS"
"        UNDEFINED, BUT ALLOW YOU TO MAKE THE ASSIGNMENT. THE PROGRAM WILL"
"        PROMPT AT A LATER TIME TO DEFINE THE COMMAND."
"        EXAMPLE:   <QT>ID-CLR<QT>.....FOR.....IDCLEAR COMMAND"
"                   <QT>GET-ST<QT>.....FOR.....GPLII PROGRAM"
""
"*****************************************************************"
""
ANS:=TEXTINPUT"PLEASE STRIKE ANY KEY TO CONTINUE:   "
""

FLAYEROFF
SOLID


:SECTION 1: DETERMINING USER'S LIBRARY


BEGIN:
  MENU_LIB:=TEXTINPUT "MENU LIBRARY NAME:   "
    IF MENU_LIB <> "" THEN
      IF (FILEINFO (MENU_LIB,".DB")) = "" THEN
        "CREATING NEW MENU LIBRARY:   ",MENU_LIB,".DB"
        FCOPY "CUSMENU.DB";(MENU_LIB,".DB")
        OPENLIB MENU_LIB
        ELSE
        "(((ERROR))) LIBRARY ALREADY EXISTS."
        GOTO BEGIN
      ENDIF
    ELSE
      ""
      "(((ERROR))) PLEASE SPECIFY MENU LIBRARY NAME"
      ""
      GOTO BEGIN
    ENDIF



:SECTION 2: SETTING THE WORK STATION
```

```
IF (STADPYINFO "CUR") <> "" THEN
  TSTRUCT
ENDIF

IF (STADPYINFO "MEN") <> "" THEN
RMENU""
RSCREEN ""
ENDIF

CRTS:=CRTTYPE

SWITCH CRTS OF
  CASE 0:       :OSTRUCT "TEKMENU"
                "NOT IMPLEMENTED AT THIS TIME"
                GOTO FINI
  CASE 1:       OSTRUCT "MRDMENU"
                CRDS:=24.25,880.9;24.25,928.7;24.25,976.5;947.9,23.8;^
                947.9,71.4;947.9,119;947.9,166.6;947.9,214.2;947.9,261.8;^
                947.9,309.4;947.9,357;947.9,404.6;947.9,452.2;947.9,499.8;^
                947.9,547.4;947.9,595;947.9,642.6;947.9,690.2;947.9,737.8;^
                947.9,785.4;947.9,833
  CASE 2:       :OSTRUCT "VMDMENU"
                "NOT IMPLEMENTED AT THIS TIME"
                GOTO FINI
  CASE 3:       OSTRUCT "MHDMENU"
                CRDS:=28.415,879.86;28.415,927.42;28.415,974.98;^
                1108.27,23.78;1108.27,71.34;1108.27,118.9;1108.27,166.46;^
                1108.27,214.02;1108.27,261.58;1108.27,309.14;^
                1108.27,356.7;1108.27,404.26;1108.27,451.82;^
                1108.27,499.38;1108.27,546.94;1108.27,594.5;^
                1108.27,642.06;1108.27,689.62;1108.27,737.18;^
                1108.27,784.74;1108.27,832.3
  CASE 4:       :OSTRUCT "BVDMENU"
                "NOT IMPLEMENTED AT THIS TIME"
                GOTO FINI
  OUT:
                "STATION TYPE NOT IMPLEMENTED"
                "      PROGRAM ABORTED        "
                GOTO FINI
ENDSWITCH

:SECTION 3: SETTING STATION PARAMETERS

AUTOHOLD 0
BLUE 1-62
RED 63
WHITE 0
VLAYER
SLAYER
VKIND
SKIND
VDTYPE
SDTYPE

VKINDOFF"SOTOSEAE"

VIEW
ZOOM 1.1
```

```
:SECTION 4:  SETTING UP ITEMS


DEFMODE 0
ITEM 0
TEXT
MAG 16
ANGLE 0
LAYER 0
FONT 0
TJUST "M";"C"
REFL "N"
SETDEFAULTS


FLAYER 63
FILLD 63

ITEM 1
SREF
SNAME "SIGN"
MAG 1
REFL "N"
SETDEFAULTS


CE 555, 414
PUT
SIN_NUM:=MSELECT 2;"";"";"";"SIGN"




DEFMODE 1


:SECTION 5: START ENTERING COMMANDS FOR MENU


:COMS:=FETCH "COMMNDS.CC"
COMS:=SYSN


COM_MANDS:=""
CCNTS:=22;3
RCNTS:=3;21
SPC:=56.83;57

N:=0
XX:=0
DO
XX:=XX + 1
C_CNT:=CCNTS[XX]
R_CNT:=RCNTS[XX]
```

```
INCR:=SPC[XX]

DO
N:=N + 1
CRD:=CRDS[N]

ITEM 1
SNAME"CIRCLE"
CE CRD
PUT

CIR_NUM:=MSELECT 2;"";"";"";"CIRCLE"
CIR_INFO:=GETEL CIR_NUM

ITEM 0

        X:=0
        DO
        X:=X + 1
        START:
        TXT:=TEXTINPUT "BUTTON DEFINITION:  "
        ""
        IF TXT <> "" AND TXT <> "NONE" THEN




S_TXT:=SIZE TXT
IF (TXT INDICESOF "-") <> "" THEN
"               FOUND DASH..."
        DSH:=TXT INDICESOF "-"
        S_DSH:=SIZE DSH
                IF S_DSH = 1 THEN
                  BTXT:=TXT[IOTA (1,(DSH - 1))],"<CR>"
                  ETXT:=TXT[IOTA ((DSH + 1),S_TXT)]
                  MAG 13
                  TXT:=BTXT,ETXT
                ENDIF
                IF S_DSH = 2 THEN
                  BTXT:=TXT[IOTA (1, (DSH[1] - 1))],"<CR>"
                  MTXT:=TXT[IOTA ((DSH[1] + 1), (DSH[2] - 1))],"<CR>"
                  ETXT:=TXT[IOTA ((DSH[2] + 1), S_TXT)]
                  MAG 10
                  TXT:=BTXT,MTXT,ETXT
                ENDIF


NEW_TEXT:=CLEANUP TXT
FIND:=""
FIND:=COMS INDICESOF (" ",NEW_TEXT," ")
IF FIND = "" THEN
        "NOT A DEFINED COMMAND..."
        ANS:=TEXTINPUT"DO YOU STILL WISH TO DEFINE (Y OR N):  "
        IF NOT (ANS[1] IN "YY") THEN
                GOTO START
        ENDIF
ENDIF
```

```
ELSE
        FIND:=""
          FIND:=COMS INDICESOF (" ",TXT," ")
            IF FIND <> "" THEN
"               NO DASH..."

                IF S_TXT > 4 AND S_TXT <= 8 THEN
                  TXT1:=TXT[IOTA (1,4)],"<CR>"
                  TXT2:=TXT[IOTA (5,S_TXT)]
                  TXT:=TXT1,TXT2
                MAG 13
                ENDIF
                IF S_TXT > 8 AND S_TXT <= 12 THEN
                  TXT1:=TXT[IOTA (1,4)],"<CR>"
                  TXT2:=TXT[IOTA (5,8)],"<CR>"
                  TXT3:=TXT[IOTA (8,S_TXT)]
                  TXT:=TXT1,TXT2,TXT3
                  MAG 10
                ENDIF
              ELSE
               "NOT A DEFINED COMMAND..."
                ANS:=TEXTINPUT "DO YOU STILL WISH TO DEFINE (Y OR N):   "
                IF NOT (ANS[1] IN "Yy") THEN
                  GOTO START
                ENDIF
                S_TXT:=SIZE TXT
                  IF S_TXT > 4 AND S_TXT <= 8 THEN
                    TXT1:=TXT[IOTA (1,4)],"<CR>"
                    TXT2:=TXT[IOTA (5,S_TXT)]
                    TXT:=TXT1,TXT2
                  MAG 13
                  ENDIF
                  IF S_TXT > 8 AND S_TXT <= 12 THEN
                    TXT1:=TXT[IOTA (1,4)],"<CR>"
                    TXT2:=TXT[IOTA (5,8)],"<CR>"
                    TXT3:=TXT[IOTA (8,S_TXT)]
                    TXT:=TXT1,TXT2,TXT3
                    MAG 10
                  ENDIF
              ENDIF
ENDIF
        ELSE
        TXT:=":::::::::"
        MAG 20
        ENDIF


ENTERTEXT TXT

CE CRD
PUT
CRD[1]:=CRD[1] + INCR
CIR_INFO[6]:=CRD
PUTEL CIR_INFO
TXT:=CLEANUP TXT
COM_MANDS:=COM_MANDS;TXT
UNTIL X = C_CNT

        ENDDO
```

```
DATA DELETE"Y";CIR_NUM

UNTIL N = R_CNT
ENDDO

UNTIL XX = 2
ENDDO



""
""
"NOW COMPILING BUTTON MENU FILE..."
""
""
MENU_FILE:=";FILE NAME:   ",MENU_LIB,".BM<CR>"
MENU_FILE:=MENU_FILE,";SINGLE LEVEL MENU<CR>"
MENU_FILE:=MENU_FILE,";DATE:   ",(CFMT DAY),"<CR>;<CR>"
MENU_FILE:=MENU_FILE,"MENU 1: ROWS=3, COLS=22<CR>"
MENU_FILE:=MENU_FILE,"MENU 2: ROWS=18, COLS=3<CR>"
MENU_FILE:=MENU_FILE,";<CR>"

MN:=0
CC:=RC:= (-1)
COM_NUM:=1
ROW_CNT:=2;17
COL_CNT:=21;2

IN_COL:=IN_ROW:=0

DO
IN_COL:=IN_COL + 1
IN_ROW:=IN_ROW + 1

N_COLS:=COL_CNT[IN_COL]
N_ROWS:=ROW_CNT[IN_ROW]

MN:=MN + 1
RC:= (-1)
        DO
        RC:=RC + 1
        CC:= (-1)
                DO
                CC:=CC + 1
                COM_NUM:=COM_NUM + 1
                MENU_FILE:=MENU_FILE,"[A",(CFMT MN),"R",(CFMT RC),"C",(CFMT CC),"]<CR>"
                NEWCOM:=CLEANUP COM_MANDS[COM_NUM]
                IF NEWCOM = ":::::::::" THEN
                  NEWCOM:=";UNASSIGNED"
                MENU_FILE:=MENU_FILE,"'",NEWCOM,"'","<CR>"
                ENDIF
                IF NEWCOM = "CR" OR NEWCOM = "C/R" THEN
                NEWCOM:="<74>CR<76>"
                MENU_FILE:=MENU_FILE,"'",NEWCOM,"'","<CR>"
                ENDIF


IF NEWCOM <> ";UNASSIGNED" AND NEWCOM <> "<74>CR<76>" THEN

        IF (COMS INDICESOF (" ",NEWCOM," ")) <> "" THEN
```

```
        MENU_FILE:=MENU_FILE,"'",NEWCOM,"<74>CR<76>'<CR>"
        ELSE
        ""
        "<BRON>COMMAND UNDEFINED:   ",NEWCOM,"<BROFF>"
        ""
        ANS:=TEXTINPUT"DO YOU WISH TO RE-DEFINE (Y OR N):   "
                IF (ANS[1] IN "Yy") THEN
                NEWCOM:=TEXTINPUT"COMMAND:   "
                MENU_FILE:=MENU_FILE,"'",NEWCOM,"<74>CR<76>'<CR>"
                ELSE
                MENU_FILE:=MENU_FILE,"'",NEWCOM,"<74>CR<76>'<CR>"
                ENDIF
        ENDIF
ENDIF
                UNTIL CC = N_COLS
                ENDDO

        UNTIL RC = N_ROWS
        ENDDO
UNTIL MN = 2
ENDDO

(MENU_LIB,".BM") SAVE MENU_FILE


SIGN:=MSELECT 2;"";"";"";"SIGN"
DATADELETE "Y";SIGN
IF CRTS = 3 THEN
WSCREEN MENU_LIB;.863 .856;0,0 1250.26,998.76
ELIF CRTS = 1 THEN
WSCREEN MENU_LIB;.866 .856;0,0 1067,1000
ENDIF
CLOSELIB
REDRAW



GEN:=FETCH"GENMENU.GS"
X:=1
DO
GENS:=GEN INDICESOF "GENMENU"
INDI:=GENS[X]

B_FILE:=GEN[IOTA (1,(INDI - 1))],MENU_LIB
E_FILE:=GEN[IOTA ((INDI + 7),(SIZE GEN))]

GEN:=B_FILE,E_FILE

UNTIL (SIZE GENS) = 1
ENDDO

(MENU_LIB,".GS") SAVE GEN
GPL MENU_LIB;"N"


X:=0
DO
X:=X + 1
"<CR>"
UNTIL X = 10
```

```
ENDDO
"<BRON>PROGRAM COMPLETE...<BROFF>"
""
"<BRON>YOUR NEW MENU IS:  ",MENU_LIB,"<BROFF>"
""
"<BRON>TYPE:  ",MENU_LIB," TO LOAD MENU TO YOUR STATION<BROFF>"
""

CALL MENU_LIB
FINI:
ENDSUB




MONADIC FUNCTION NUM_CHAR:=CFMT IN_NUM
LOCAL NUM_CHAR; IN_NUM

EXTERNAL OUT_NUM

OUT_NUM:=CFORMAT IN_NUM
OUT_NUM:=OUT_NUM[IOTA (1,((SIZE OUT_NUM) - 1))]
NUM_CHAR:=OUT_NUM
ENDSUB




:PROGRAM NAME:   CLEANUP.GS
:CREATED BY:     CHRIS MARTIN
:DATE:           JUNE 18, 1984

:RELEASE 5.0.1

:PURPOSE:   THIS PROGRAM WILL CLEAN UP THE COMMANDS USED TO
:DEFINE THE BUTTONS IN THE CUSMENU PROGRAM. IT WILL REMOVE
:THE CARRIAGE RETURNS FROM THE COMMANDS WHICH HAVE BEEN
:BROKEN UP TO FIT THE BUTTON AREA ON THE MENU.

MONADIC FUNCTION CMND:=CLEANUP INPUT_CMND
LOCAL CMND; INPUT_CMND
EXTERNAL S_CMND; CHK; COM_MANDS; B_CMND; M_CMND; E_CMND

S_CMND:=SIZE INPUT_CMND
CHK:=INPUT_CMND INDICESOF "<CR>"

IF CHK <> "" AND CHK <> S_CMND THEN
        IF (SIZE CHK) = 1 THEN
                B_CMND:=INPUT_CMND[IOTA (1, (CHK - 1))]
                E_CMND:=INPUT_CMND[IOTA ((CHK + 1), S_CMND)]
                CMND:=B_CMND,E_CMND
        ENDIF

        IF (SIZE CHK) = 2 THEN
                B_CMND:=INPUT_CMND[IOTA (1, (CHK[1] - 1))]
```

```
                M_CMND:=INPUT_CMND[IOTA ((CHK[1] + 1),(CHK[2] - 1))]
                E_CMND:=INPUT_CMND[IOTA ((CHK[2] + 1),S_CMND)]
                CMND:=B_CMND,M_CMND,E_CMND
        ENDIF

        ELSE
        CMND:=INPUT_CMND
ENDIF

ENDSUB
```

GDSII DATABASE INTERFACE

Why is GPLII necessary?

Perhaps the most useful aspect of GPLII is its capability to efficiently interface with the GDSII database. This is not to say that GPLII is not a useful tool throughout the entire GDSII environment. It is, in fact, an integral part of the GDSII graphics system.

However, if any one capability of GPLII may be described as the true "power" of this programming language, it is in the area of database manipulation. With GPLII as a CAD tool we, as users, no longer have to rely on ITEMs and/or ID groups in order to modify the database. We can use GPLII to perform modifications in the database with minimal input from the user. Thus reducing the chances for operator error.

Not only may we use GPLII to perform modifications within a single structure, we may also instruct the GPLII program to perform a number of specific modifications throughout the entire database. This capability provides a more efficient use of the GDSII system.

In this section, we are going to discuss the commands and functions that allow GPLII to interface with the GDSII database. We will also be examining several example programs.

GPLII FUNCTIONS

GPLII Function: NSELECT

Niladic Syntax: var := NSELECT(CR)

Function: Assigns the keys of all selectable elements in the open structure to the variable as an integer2. Check the WINOPTIONS.

Returns a null if no elements are selectable.

Program Information:

```
TYPEOF  = Integer2 or Null
SIZE    = n Keys (0 if null)
LENGTH  = 1
SHAPE   = n Keys (0 if null)
```

Example:

```
? NKEYS := NSELECT(CR)
?
```

Assigns all selectable keys to NKEYS.

GPLII Function: MSELECT

Monadic Syntax: var := MSELECT selection pattern(CR)

Function: Assigns all of the keys in the open structure that are specified by the selection pattern to the variable as an integer2. The selection pattern must include list element one. Any other list elements contained in the selection pattern define the parameters the elements specified in list element one must meet in order to be selected by MSELECT. Any parameters that do not apply to the specified type(s) will be ignored.

The order of the selection pattern must conform to the order of the list elements in the element list:

list[-3] - do not sort keys ← NOTE! THIS IS MUCH FASTER
list[-2] - database only
list[-1] - id group
list[1] - element type
list[2] - layer number (Boundary, Path, Text, Box, and Node) or else "NULL"
list[3] - datatype, texttype, boxtype, structype, else "NULL"
list[4] - width (+n)/abswidth (-n), or array shape, or else "NULL"
(list[4])[1] - if array shape, number of columns
(list[4])[2] - if array shape, number of rows
list[5] - structure name, or text string, or else "NULL"
list[6] - element coordinate(2), array extent
list[7] - transformation specifier property, else "NULL"
(list[7])[1] - reflection (1), no reflection (0)
(list[7])[2] - angle (+n)/absangle (-n)
(list[7])[3] - mag (+n)/absangle (-n)
list[8] - pathtype or font number, else "NULL"
list[9] - Property number 127 (user string) else "NULL"
list[10] - Property number 126 (user integer) else "NULL"
list[11] - All other property values ([property number] value [property number] value . . .)

MSELECT can select non-selectable elements. If window coordinates (list element six) have not been specified, the coordinates of the maximum data area will be assigned automatically. If no window is specified when "EX" is a WINOPTION, no keys will be scanned.

Wild cards * and - are allowed in a character vector pattern. Semicolons (;) must separate list elements, and double quotes (" ") enclose character vectors.

Program Information:

TYPEOF = Integer2 or Null
SIZE = n Keys (0 if null)
LENGTH = 1
SHAPE = n Keys (0 if null)

Examples:

? BKEYS := MSELECT 3; 15; 3(CR)
?
Assigns the keys of all Boundaries on layer 15 with datatype 3 to BKEYS.

? MKEYS := MSELECT 2; ; ; ; "BUF1"; ; 0 45 1(CR)
?
Assigns the keys of all SREF's named BUF1 with 0 reflection, an angle of 45 degrees, and a magnification of 1 to MKEYS.

? ALLKEYS := MSELECT 1 2(CR)
?
Assigns the keys of all AREF's and SREF's to ALLKEYS.

GPLII Function: DSELECT

Dyadic Syntax: var := keys DSELECT selection pattern(CR)

Function: Assigns all of the specified keys that meet the selection pattern criteria to the variable as an integer2 vector. The selection pattern must include list element one. Any other list elements contained in the selection pattern define the parameters the elements specified in list element one must meet in order to be selected by DSELECT. Any parameters that do not apply to the specified element type(s) will be ignored.

The order of the selection pattern must conform to the order or the list elements in the element list:

NOBTYPE

list[1]	- element type
list[2]	- layer number (Boundary, Path, Text, Box, and Node) or else "NULL"
list[3]	- datatype, texttype, boxtype, structype, else "NULL"
list[4]	- WIDTH (+ n)/Abswidth (-n), array shape, else "NULL"
(list[4])[1]	- if array shape, number of columns
(list[4])[2]	- if array shape, number of rows
list[5]	- structure name or text string else "NULL"
list[6]	- element coordinates or array extents
list[7]	- transformation specifier property else "NULL"
(list[7])[1]	- reflection (1), no reflection (0)
(list[7])[2]	- angle (+ n)/absangle (-n)
(list[7])[3]	- mag (+ n)/absmag (-n)
list[8]	- pathtype or font number else "NULL"
list[9]	- property number 127 (user string) else "NULL"
list[10]	- property number 126 (user integer) else "NULL"
list[11]	- all other property values (formatted property number value property number value . . .)

wild cards - and * are allowed in a character vector pattern. Semicolons (;) must separate list elements and double quotes (" ") enclose character vectors.

Program Information:

TYPEOF	= INTEGER2 or NULL
SIZE	= n (0 if NULL)
LENGTH	= 1
SHAPE	= n (0 if NULL)

Examples:

```
? BOFB := IDKEYS DSELECT 3; 12; 0(CR)
?
```

or

```
? BKEYS:= IDKEYS(CR)
? BOFB := BKEYS DSELECT 3; 12.0(CR)
?
```

Assigns the keys of all the elements in the identified group that are Boundaries on layer 12 with datatype 0 to BOFB.

```
? SKEYS := MSELECT 2(CR)
? MKEYS := SKEYS DSELECT 2; ; ; ; "SSR06-"(CR)
?
```

Assigns the keys of all SREF's whose names begin with SSR06 to MKEYS.

GPLII Function: USELECT

Monadic Syntax: var := USELECT selection pattern(CR)

Function: USELECT (Update user Property Select) selects the nearest data element(s) and update the user property value specified by the pattern criteria. USELECT as a function returns the total number of data elements updated. The pattern criteria is outlined as follows:

[1] The first element in the list specifies the element type. The element type is represented as an integer value 1 through 7. Where 1 = AREF, 2 = SREF, 3 = BOUNDARY, 4 = PATH, 5 = TEXT, 6 = NODE, and 7 = BOX. If more than one data element is to be scanned, a vector can be specified. Besides the element number 1 through 7, a 0 can be used to display the trapsize of the window to be scanned. The trapsize is determined by the SNAPSIZE value and is centered around the coordinate given in the sixth element in the list. See below.

[2] The second element in the list specifies the element layer. The argument can be an integer scalar (single layer) or an integer vector (multiple layers). If multiple layers are to be specified, the dash (-) cannot be used. Instead, used the GPLII operator IOTA. See examples below.

[3] The third element in the list specifies the element type. The element type can be an integer scalar or an integer vector. Syntax wise, it is the same as layer. See example below.

[4] The fourth element in the list specifies the width or a range of widths (n,m) of a PATH element. Currently this element list is not implemented and a null must be specified.

[5] The fifth element in the list specifies the structure name(s) or text string(s). If more than one structure name or text string is to be scanned a space must be used as a delimiter to separate the names. Currently the fifth element in the list is not implemented and a null argument must be given. Wild card patterns can be used. Wild card patterns are as follows:

*(asterisk) Match any one character or match any number of characters depending on the number of asterisks being specified. For example:

"**CDE*"	"ABCDEFG"	(no match)
	"ABCDEF"	(match)
	"1ABCDEF"	(no match)
	"X1CDE2"	(match)

-(dash) Match any number of characters. For example:

"-X-Y"	"XY"	(match)
	"YX"	(no match)
	"EXPKEY"	(match)
	"3XY7"	(no match)

@(at) Match any single upper or lower case characters. For example:

"A@B@"	"AaBb"	(match)
	"AABb"	(no match)
	"abcd"	(match)
	"ABCD"	(no match)

(Pound) Match any single ASCII numeric digits 0 through 9. For example:

"#A#B"	"1A1B"	(match)
	"A1B1"	(no match)
	"1ABB"	(no match)

^(ESCAPE) Match any single character which is a special character as an actual character, such as spaces, dash, asterisk, @, #, commas, semicolons, and ^. The ^ must preceed these special characters. For example:

"A space B"	= "A^spaceB"
"A*B"	= "A^*B"
"AB-1"	= "AB^-1"
"@7.5"	= "^@7.5"
"1^2"	= "1^^2"

Note: for double quotes used double, double quotes e.g. "hi" = ""hi"".

[6] The sixth element in the list specifies the coordinate location to scan for the data element. Note, be sure that the size of SNAPSIZE is not too large or too small. The coordinate specified is to be the center of the trapsize window.

[7] The seventh element in the list specifies the transformation specifier. The argument is entered as a vector where the first value specifies the reflection (0 = no reflection, 1 = reflected), second value specifies the angle (+n = relative, -n = absolute), and the third value specifies the magnification (+n = relative, -n = absolute). Currently this is not implemented. A null must be specified.

[8] The eigth element in the list specifies a vector indicating the pathtype, font number, and text justification. Currently this is not implemented. A null must be specified.

[9] The ninth element in the list specifies the APN (Associated Property Number). This indicates that the elements being scanned, for example, a TEXT and a NODE of the same plex having the same property number and value will be updated with the new property value specified in the eleventh element of the list (see below). If the element is a TEXT, then the associated type (the TEXTTYPE which has been already specified in the third element of the list (see above) will be updated with the same value as a text string.

[10] The tenth element in the list specifies the property number that is to be updated. For example, in a pin node the property number 11 gets the new property value and the text with the corresponding texttype 11 gets the value as a text string.

[11] The eleventh element in the list specifies the property value.

[12] The twelfth element in the list specifies the PLEXMODE where, 0 = PLEXMODE OFF and 1 = PLEXMODE ON.

Examples:

```
? USELECT 5 6 0; 15 4; 11 58; ; ; (ce); ; ; 15; 11; "n"; 0 (CR)
  n
?
```

Scan for the nearest TEXT and NODE element, turn on and draw the trapsize; scan for the layers 15 and 4 with datatype 11 or 58, or texttype 11 or 58; ; ; at the nearest coordinate location; ; ; with the APN (associated property number) 15; update property number 11; and update the textstring of the text with texttype 11; and turn off the plex mode.

```
? USELECT IOTA 0 4; IOTA 1 5; 11 16; ; ; (ce); ; ; 17; 20; "n"; 0
?
```

Scan for elements 1 2 3 4 and draw trapsize; on layers 1 through 5; with types 11 and 16; ; ; at this location; ; ; with APN 17; with property number 20; width is new property value; with plexmode off.

GPLII Function: GETEL

Monadic Syntax: var := GETEL key(CR)

Function: Assigns the element list of the element whose key is specified to the variable. The type of element specified determines which of these list elements will be included in the element list:

(list[1])[1]	element type
(list[1])[2]	element key
(list[1])[3]	element bucket (Item number, -1 if identified, else -2)
(list[1])[4]	plex number, if present
list[2]	layer nymber (Boundary, Path, Text, Box or Node) else "NULL"
list[3]	datatype, texttype, boxtype, structype, else "NULL"
list[4]	width (+n), abswidth (-n), array shape or else "NULL"
(list[4])[1]	if array shape, number of columns
(list[4])[2]	if array shape, number of rows
list[5]	structure name, text string, else "NULL"
list[6]	element coordinates or array extents
list[7]	transformation specifier property else "NULL"
(list[7])[1]	reflection (1)/no reflection (0)
(list[7])[2]	angle (+ n)/ABSANGLE (-n)
(list[7])[3]	mag (+ n)/ABSMAG (-n)
list[8]	pathtype, or font number, else "NULL"
list[9]	property number 127 (user string), else "NULL"
list[10]	property number 126 (user integer), else "NULL"
list[11]	All other property values ([property number] value [property number] value . . .)

PUTEL returns this list directly to the data base. Use NSELECT, MSELECT, DSELECT, CEKEY, IDKEYS, or GET to find the element's key.

GEDELEMENT assigns the same list from an Item.

Program Information:

TYPEOF = LIST
SIZE = N/A
LENGTH = Minimum of 6 maximum of 11
SHAPE = N/A

Example:

? EXPEL := GETEL 43766(CR)
?
Assigns information above key 43766 to EXPEL, for instance:

```
? EXPEL(CR)
  3 43766 -1 13 0 0 0
                1 0
                1 1
                0 1
                0 0
```

GPLII Function: PUTEL

Syntax: PUTEL elementlist(CR)

Guidelines:

1. A structure must be open for editing.
2. The specified element cannot already have been gotten.
3. The length of the element list determines the list elements that can be subscripted and changed.
4. PUTEL is usually used in a GPLII program with the element list assigned to a variable.

Function: Updates the element list of the specified element and places the modified list directly into the data base.

See GETEL and CEKEY.

Example:

```
? ELIST := (GETEL (CEKEY)) (CE/CR)
? ELIST [2] := 3 (CR)
? PUTEL ELIST(CR)
?
```

Gets the specified element, explodes a copy of its element list, assigns the list to ELIST, changes the layer number to 3, and places the modified list into the data base.

MAGL IB.GS	Magnifies boundaries, paths and text within a specified database library. The magnification occurs on a structure-by-structure basis. The magnification amount will be specified by the user.
FINDANGLE.GS	This program reports the key number of any SREFs found with an absolute angle. The report will contain the structure in which the SREF was located with the key number of the SREF.
CM$EDIT.GS	This program allows the user to edit text strings in the database much the way the "CH" command works with the GDSII text editor.
SCRIBE.GS	This is an auto-scribe program that will generate a fracturable scribe structure that may be referenced on the device.
PGTITLES.GS	This program will generate the necessary P. G. titles. A fracturable text font must first be built.

```
:PROGRAM NAME:    MAGLIB
:CREATED BY:      CHRIS MARTIN

NILADIC PROCEDURE MAGLIB
EXTERNAL LIBRARY_NAME;CNT;MAG_AMOUNT;KEYS
EXTERNAL STRUCTURE_LIST

SKIND
VKIND
VLAYER
SLAYER
SDTYPE
STTYPE
SBTYPE
SNTYPE
SKINDOFF"SOAOND"
VKINDOFF"SOAOND"

LEVEL 0

LIBRARY_NAME:=TEXTINPUT"ENTER LIBRARY NAME:   "
IF LIBRARY_NAME <> "" THEN
  OPENLIB LIBRARY_NAME
  STRUCTURE_LIST:=STRUCLIST "-"
```

```
  MAG_AMOUNT:=EXPINPUT"ENTER MAGNIFICATION AMOUNT:   "
  IF MAG_AMOUNT <> "" THEN
    CNT:=0
    DO
    CNT:=CNT + 1
    OSTRUCT (STRUCTURE_LIST[CNT])
    "NOW WORKING IN:   <BRON>",(STRUCTURE_LIST[CNT]),"<BROFF>"
    KEYS:=MSELECT 3 4 5
    IF KEYS <> "" THEN
      DATAMAGNIFY MAG_AMOUNT; KEYS
    ENDIF
    UNTIL CNT = (LENGTH STRUCTURE_LIST)
    ENDDO
  ENDIF
ENDIF

LEVEL 10
CLOSELIB

ENDSUB


----------------------------------------------------------------


:Program Name:          FINDANGLE
:Created by:            Chris Martin
:Date:                  Dec. 4, 1983
:Release:               4.0.9 v6

:This porgram will search the data base library for all AREFs
:and SREFs that have been referenced with an ABSOLUTE ANGLE.
:This program will report the key numbers associated to the
:SREFs and AREFs that are found.


NILADIC PROCEDURE FINDANGLE
EXTERNAL ABS_KEYS; LIB; LIB_LIST; OLD_SKINDS; OLD_VKINDS; N; NN; KEYS
EXTERNAL STRUC; KEY_NUM; KEY_INFO; NUM_KEYS; RAM; CNT; LOGS

ABS_KEYS:=""
LOGS:=LOG
IF LOGS <> "" THEN
ENDLOG
ENDIF
IF (FILEINFO"ABSLOG.LG") <> "" THEN
FDELETE"ABSLOG.LG"
LOG"ABSLOG"
ELSE
LOG"ABSLOG"
ENDIF


LIB:=TEXTINPUT "LIBRARY NAME:   "
IF LIB <> "" THEN
OPENLIB LIB
ELSE
GOTO ENDIT
ENDIF
```

```
""
"NOW MAKING A LIST OF STRUCTURE NAMES"
"WORKING...PLEASE WAIT"
""
LIB_LIST:=STRUCLIST ""
OLD_SKINDS:=SKIND
OLD_VKINDS:=VKIND
SKIND"SOSEA0AE"
VKIND"SOSEA0AE"

AUTOHOLD 0
LEVEL 0

IF LIB_LIST <> "" THEN
""
"A REPORT IS BEING GENERATED IN FILE: ABSLOG.LG<BEL>"
""
N:=0
DO
N:=N + 1
STRUC:=LIB_LIST[N]
""
"NOW WORKING IN:<BRON>  ",STRUC,"<BROFF>"
OSTRUCT STRUC
      KEYS:=MSELECT 1 2
        IF KEYS <> "" THEN
        CNT:=(SIZE KEYS)
        NN:=0
        DO
          NN:=NN + 1
          KEY_NUM:=KEYS[NN]
          KEY_INFO:=GETEL KEY_NUM
          RAM:=KEY_INFO[7]
           IF (RAM[2]) < 0 THEN
           ABS_KEYS:=ABS_KEYS UNION KEY_NUM
           KEY_NUM
           ENDIF
        UNTIL NN = CNT
        ENDDO
        ENDIF
UNTIL N = (LENGTH LIB_LIST)
ENDDO
ENDIF

NUM_KEYS:=(SIZE ABS_KEYS)
"THERE WERE <BRON>";NUM_KEYS;" <BROFF>KEY NUMBER(S) LOCATED"
""
ENDIT:
SKIND OLD_SKINDS
VKIND OLD_VKINDS
LEVEL 10
"PROGRAM COMPLETE<BEL>"
ENDSUB      :FINDANGLE.GS
```

```
NILADIC PROCEDURE CM$EDIT
EXTERNAL CRD; LL; UR; KEY_NUMS; SIZE_KEYS; N; KEYNUM
EXTERNAL LIB; STR; TXT; KEYINFO; OLD; NEW; FIND; SIZE_OLD; SIZE_NEW
EXTERNAL BEG; END; SIZE_TXT; ANS
EXTERNAL GLOB; NN; FIND_NUM
EXTERNAL TO_DO;SZ_FIND;NUMS;OCUR;FIND_NUMS

LIB := OPENLIB
IF (LENGTH LIB) = 3 THEN
  LIB:=LIB[1]
ENDIF
LIB := TEXTINPUT "Enter the library to open (",LIB,"): "
IF (LIB <> "") THEN
   OPENLIB LIB
ENDIF

STR := OSTRUCT
STR := TEXTINPUT "Enter the structure to open (",STR,"): "
IF (STR <> "") THEN
   OSTRUCT STR
ENDIF

DO
   START:
   CRD := EXPINPUT "DIGITIZE COORDINATE"
   IF (CRD <> "") THEN
      LL := (CRD[1] - 5), (CRD[2] - 5)
      UR := (CRD[1] + 5), (CRD[2] + 5)
   ELSE
      GOTO FINIS
   ENDIF

   KEY_NUMS := MSELECT 5;"";"";"";"";LL,UR
   SIZE_KEYS := SIZE KEY_NUMS
   IF (SIZE_KEYS <> "") THEN
      N := 0
      DO
         N := N + 1
         IF (SIZE_KEYS > 1) THEN
            KEYNUM := KEY_NUMS[N]
         ELSE
            KEYNUM := KEY_NUMS
         ENDIF
         ID KEYNUM
         ANS := TEXTINPUT "IS THIS THE CORRECT TEXT (Y)?  "
         IF (ANS = "" OR ANS = "Y") THEN
            IDCLEAR
            KEYINFO := GETEL (KEYNUM)
            TXT := KEYINFO [5]
            SIZE_TXT := SIZE TXT
            DO
              OLD := TEXTINPUT "OLD TEXT STRING:  "
            UNTIL OLD <> ""
            ENDDO

            DO
              NEW := TEXTINPUT "NEW TEXT STRING:  "
            UNTIL NEW <> ""
            ENDDO

            FIND := TXT INDICESOF OLD
```

```
                SIZE_OLD:=SIZE OLD
                SIZE_NEW:=SIZE NEW

!=== ======  NEW PART ========================================


IF (SIZE FIND) > 1 THEN
  TO_DO:=TEXTINPUT" LOCAL <BRON>(L)<BROFF> OR GLOBAL <BRON>(G)<BROFF> CHANGE: "
    IF TO_DO <> "" THEN
      SWITCH, TO_DO OF
        CASE"L":
          SZ_FIND:= SIZE FIND
          NUMS:=IOTA SZ_FIND
          OCUR:=EXPINPUT"WHICH OCCURANCE (",(CFORMAT NUMS),"):  "
          IF OCUR <> "" THEN
           FIND:=FIND[OCUR]
          ENDIF
                  IF (FIND <> 1) AND (FIND <> (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND - 1))]
                     END := TXT [IOTA ((FIND + SIZE_OLD), SIZE_TXT)]
                     KEYINFO[5] := (BEG, NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = 1) THEN
                     END := TXT [IOTA ((SIZE_OLD + 1), SIZE_TXT)]
                     KEYINFO[5] := (NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND_NUM - 1))]
                     KEYINFO[5] := (BEG, NEW)
                     PUTEL KEYINFO
                  ENDIF


        CASE"G":
          FIND_NUMS := FIND
          NUMS := SIZE FIND_NUMS
          NN := 0
          DO
            NN := NN + 1
            KEYINFO := GETEL KEYNUM
          TXT := KEYINFO[5]
            FIND := TXT INDICESOF OLD
            IF (SIZE FIND) > 1 THEN
              FIND := FIND[1]
            ENDIF
                  IF (FIND <> 1) AND (FIND <> (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND - 1))]
                     END := TXT [IOTA ((FIND + SIZE_OLD), SIZE_TXT)]
                     KEYINFO[5] := (BEG, NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = 1) THEN
                     END := TXT [IOTA ((SIZE_OLD + 1), SIZE_TXT)]
                     KEYINFO[5] := (NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND_NUM - 1))]
                     KEYINFO[5] := (BEG, NEW)
                     PUTEL KEYINFO
                  ENDIF
```

```
BEG: =""
END: =""
UNTIL  NN = NUMS
ENDDO

ENDSWITCH

ELSE
  GOTO START
ENDIF

ELSE




                  IF (FIND <> 1) AND (FIND <> (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND - 1))]
                     END := TXT [IOTA ((FIND + SIZE_OLD), SIZE_TXT)]
                     KEYINFO[5] := (BEG, NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = 1) THEN
                     END := TXT [IOTA ((SIZE_OLD + 1), SIZE_TXT)]
                     KEYINFO[5] := (NEW, END)
                     PUTEL KEYINFO
                  ELIF (FIND = (SIZE_TXT - (SIZE_OLD - 1))) THEN
                     BEG := TXT [IOTA (1, (FIND_NUM - 1))]
                     KEYINFO[5] := (BEG, NEW)
                     PUTEL KEYINFO
                  ENDIF
               ENDIF
            ELSE
         IDCLEAR
      ENDIF

      UNTIL (N = SIZE_KEYS)
      ENDDO
   ELSE
      "NO TEXT FOUND"
      GOTO START
   ENDIF
UNTIL (CRD = "")
ENDDO

FINIS:
ENDSUB
```

```
:PROGRAM NAME:   SCRIBE.GS
:CREATED BY:     CHRIS MARTIN
:DATE:           NOV. 29, 1984

:THIS IS AN AUTO-SCRIBE PROGRAM.
NILADIC PROCEDURE SCRIBE
EXTERNAL CNT;XDSU;YDSU;DEL0;DELS;X1;X2;X3;X4;Y1;Y2;Y3;Y4
EXTERNAL LIB;STRUC;INF;XDSM;SWID;SCF;LAYS;NLAY;CLAY;DELIN;DELI
EXTERNAL CORD;YDSM;CORDS;PTS

PTS:=0
AUTOHOLD 0
LIB:=OPENLIB
IF LIB = "" THEN
  DO
  LIB:=TEXTINPUT"LIBRARY NAME:   "
  ""
  UNTIL LIB <> ""
  ENDDO
ELSE
LIB:=LIB[1]
ENDIF

STRUC:=TEXTINPUT"SCRIBE STRUCTURE NAME (SCRIBE):   "
IF STRUC = "" THEN
STRUC:="SCRIBE"
ENDIF
IF ("" <> STRUCLIST STRUC) THEN
  ""
  "<BRON><BEL>(((ERROR)))",STRUC," ALREADY EXISTS<BROFF>"
  GOTO FIN
ENDIF

XDSM:=EXPINPUT"FINAL X DIMENSION IN MILS:   "
IF XDSM = "" THEN
  GOTO FIN
ENDIF

YDSM:=EXPINPUT"FINAL Y DIMENSION IN MILS:   "
IF YDSM = "" THEN
  GOTO FIN
ENDIF

SWID:=EXPINPUT"FINAL SCRIBE WIDTH IN MICRONS (100):   "
IF SWID = "" THEN
  SWID:=100
ENDIF

SCF:=EXPINPUT"SCALE FACTOR (1):   "
IF SCF = "" THEN
  SCF:=1
ENDIF

LAYS:=EXPINPUT"SCRIBE LINE LAYERS (I.E., 1 2 3):   "
IF LAYS = "" THEN
  GOTO FIN
ENDIF
NLAY:=SIZE ,LAYS

LP6:
DELIN:=EXPINPUT"FINAL DELTAS IN SAME ORDER AS ABOVE (0):   "
IF DELIN = "" THEN
  DELIN:=NLAY RESHAPE 0
ENDIF

IF NLAY <> (SIZE DELIN) THEN
```

```
  "<BRON>(((ERROR)))NO. OF DELTAS MUST BE EQUAL TO NO. OF LAYERS<BROFF>"
  GOTO LP6
ENDIF

CLAY:=EXPINPUT"CHIPS SIZE REFERENCE LAYER (63):  "
IF CLAY = "" THEN
  CLAY:=63
ENDIF

XDSU:=.25*(FLOOR ((XDSM*50.8)%SCF))
YDSU:=.25*(FLOOR ((YDSM*50.8)%SCF))
DEL0:=.25*(CEILING ((SWID*2)%SCF))
DELS:=.25*(FLOOR ((DELIN*4)%SCF))
X4:=XDSU + DEL0
X1:=- X4
Y4:=YDSU + DEL0
Y1:=- Y4
PUTALL
BSTRUCT STRUC
DATATYPE 0
STRAIGHT
BOUNDARY
LAYER CLAY

CORDS:=2 2 RESHAPE (-XDSU,-YDSU,XDSU,YDSU)
RT CORDS
CNT:=0
DO
CNT:=CNT + 1
LAYER LAYS[CNT]
DELI:=DEL0 + DELS[CNT]
X3:=XDSU - DELI
X2:=- X3
Y3:=YDSU - DELI
Y2:=- Y3

PTS:=X1,Y1,X4,Y1,X4,Y4,0,Y4,0,Y3,X3,Y3,X3,Y2,X2,Y2
PTS:=13 2 RESHAPE (PTS,X2,Y3,0,Y3,0,Y4,X1,Y4,X1,Y1)
CE PTS
PUT
UNTIL CNT = NLAY
ENDDO

TSTRUCT

IF ("" <> FILEINFO "SCRIBE.LG") THEN
  FDELETE "SCRIBE.LG"
ENDIF

LOG"SCRIBE.LG"
"SCRIBE PROGRAM"
"==========="
""
"LIBRARY NAME:  ",LIB
"SCRIBE STRUCTURE:  ",STRUC
""
"DATABASE DIE SIZE:  ",(CFORMAT (XDSU%12.7))," X ",(CFORMAT (YDSU%12.7))," MILS"
"                    ",(CFORMAT (XDSU*2))," X ",(CFORMAT (YDSU*2))," MICRONS"
"LAYERS:  ";LAYS
"DATABASE DELTAS:  ";DELS
"DIE SIZE REF. LAYER:  ";CLAY
```

```
""
"SCALE FACTOR:   ";SCF
""
ENDLOG
FCOPY "SCRIBE.LG";"$LPT"
FIN:
"END OF PROGRAM"
ENDSUB


-----------------------------------------------------------------------



NILADIC PROCEDURE PGTITLES
EXTERNAL LIB;STR_NAME;SK;VK;SL;VL;N;INPUT_LAYERS
EXTERNAL INPUT_LAYER;DEV_NAME;MASK_ID;MASK_IDS
EXTERNAL TEXT_SIZE;TEXT_MAG;TEXT_CRD;TODAY;PG_LAYER;PG_TEXT

SK:=SKIND
VK:=VKIND
SL:=SLAYER
VL:=VLAYER

SKIND
VKIND
SLAYER
VLAYER


LIB:=OPENLIB
IF LIB = "" THEN
  DO
  LIB:=TEXTINPUT"LIBRARY NAME:   "
  UNTIL LIB <> ""
  ENDDO
  OPENLIB LIB
ENDIF

STR_NAME:=OSTRUCT
IF STR_NAME = "" THEN
  DO
  STR_NAME:=TEXTINPUT"STRUCTURE NAME:   "
  UNTIL STR_NAME <> ""
  ENDDO
  OSTRUCT STR_NAME
  VIEW
ENDIF

BINDFONTS 3;"PGFONT"
DEFMODE 0


""
""
INPUT_LAYERS:=EXPINPUT"INPUT FIRST LAYER NUMBER:   "
IF INPUT_LAYERS <> "" THEN
  DO
  INPUT_LAYER:=EXPINPUT"INPUT NEXT LAYER NUMBER (END):   "
  IF INPUT_LAYER <> "" THEN
  INPUT_LAYERS:=INPUT_LAYERS;INPUT_LAYER
```

```
  ENDIF
  UNTIL INPUT_LAYER = ""
  ENDDO
ENDIF

""
DO
DEV_NAME:=TEXTINPUT"DEVICE NAME:   "
UNTIL DEV_NAME <> ""
ENDDO
DEV_NAME:=DEV_NAME," - "

MASK_IDS:=TEXTINPUT"ENTER MASK ID FOR LAYER ",^
         ,(CFORMAT INPUT_LAYERS[1])," :   "
IF MASK_IDS <> "" THEN
  N:=1
  DO
  N:=N + 1
  MASK_ID:=TEXTINPUT"ENTER MASK ID FOR LAYER ",^
          ,(CFORMAT INPUT_LAYERS[N])," :   "
  MASK_IDS:=MASK_IDS;MASK_ID
  UNTIL N = (LENGTH INPUT_LAYERS)
  ENDDO
ENDIF


""
""
TEXT_SIZE:=EXPINPUT"TEXT HEIGHT (25):   "
IF (TEXT_SIZE = "") OR (TEXT_SIZE = 25) THEN
  TEXT_MAG:=1
ELSE
  TEXT_MAG:=TEXT_SIZE % 25
  TEXT_MAG:=1 DROUND TEXT_MAG
ENDIF


ITEM 0
TEXT
MAG TEXT_MAG
ANGLE 0
FONT 3
TJUST "B";"C"

DO
TEXT_CRD:=EXPINPUT"INDICATE BOTTOM/CENTER TITLE LOCATION:   "
UNTIL TEXT_CRD <> ""
ENDDO

TODAY:=DAY
N:=0
DO
N:=N + 1
PG_LAYER:=INPUT_LAYERS[N]
PG_TEXT:=DEV_NAME,MASK_IDS[N]," - ",(CFORMAT TODAY)

LAYER PG_LAYER
ENTERTEXT PG_TEXT

CE TEXT_CRD
PUT
```

```
UNTIL N = (LENGTH INPUT_LAYERS)
ENDDO

VKIND VK
SKIND SK
VLAYER VL
SLAYER SL

"PROGRAM COMPLETE"
""
ENDSUB
```

SECTION V GPLII INTERFACE TO THE BACKGROUND JOB SYSTEM

GPLII INTERFACE TO THE BACKGROUND JOB SYSTEM

All of the various forms of background jobs (plots, PGs, DRCs, etc.) may be entered into the job system by GPLII programs. The syntax of JOBCREATE will be one argument in the form of a list with three character vectors.

The first character vector is either the job template (.JT) file to be used or a user file which contains a previously saved job.

The second argument is either a null character vector (" ") to run the job or the name of a user file in which to save the job. Said user file may already exist. If so, note that it will be overwritten even though it may have a P attribute.

The third character vector is the parameter collector. This employs strange and different syntax than that with which the GDSII operator is familiar.

Example:

```
parametercollector='parameter'
INFORM_LIBRARY='NEWLIB.DB'
MAG_TAPE_LOADED='Y'
```

Parameters collectors are the key words which means they are reserved for a particular application. Each collects a different parameter for the job. Then they are concatenated and fed to JOBCREATE in the third position of the argument.

Only parameters which have no default value are absolutely required. For example, when running a P. G. tape, one could omit the scale factor and it would default to 10, but could not omit the library name or the structure name.

```
To run a job:    JOBCREATE "XCLI";"";"CMD_LINE='DAILYDUMP'"

To save a job:   JOBCREATE "XCLI";"DAILY.JB";"CMD_LINE='DAILYDUMP'"

To edit a job:   JOBCREATE "DAILY.JB";"DAILY.JB";"CMD_LINE='DAILYSAVE'"

To run to saved job:    JOBCREATE "DAILY.JB"
```

Syntax: JOBCREATE "jobfile1.QT"; "jobfile2.QT"; "Keyword= 'new arg'" (CR)

Function:

This Helpfile describes the format of the parameter list used for JOBCREATE.

Existing parameters prompted for by JOBCREATE can be specified by using the appropriate Keyword. Keywords (listed below by Jobtype) specify parameters, or parameter changes, for a particular job.

In the in-line format described by JOBPARAMS, JOBCREATE will accept up to 3 arguments, entered on the command input line. Arguments are separated by semi-colons, and are enclosed in double quotes if they are in strings. A keyword must be separated from the argument by an equal sign (=). Spaces and CR may be used as separators within the parameter list, but not within a Keyword or argument.

The first argument is the name of the jobtype file to be executed.

The second argument is the name of the new Jobtype file. This file will contain a copy of the original job, but will reflect the changes specified in the third argument. This allows the parameters of the existing job to be changed in a new file while a copy of the original format is retained. If a null string is entered for the second argument, and the Job Monitor is running, the job will be immediately started.

The third argument contains the keywords and arguments to specify the changes to the existing job file. To use more than one keyword or argument after the third argument, assign the third and other desired arguments and keywords to a variable and use the variable in place of the third argument.

Example: A GDS1T02 job was set up in job file G2.QT. The following command is issued:

```
JOBCREATE "G2.QT"; "GT.QT"; "GDS1_LIBRARY='CHIPSII'" <CR>
```

The job specified in G2.QT is copied into file GT.QT with a GDS Library of CHIPSII. "GDS1_LIBRARY" is the keyword, "CHIPSII" is the argument.

Keywords and their arguments are listed below:

GENERAL JOB PARAMETERS

"PRIORITY"

Changing this parameter will change the priority of the job. The correct format of the argument is the letter A, B, C, or D. ("PRIORITY='A'")

"JOB_NAME"

The Jobname should be a string of up to 10 characters.
("JOB_NAME='RULESCHECK'")

"MAG_TAPE_LOADED"

The argument should be Y or N depending on whether the mag tape

is loaded. ("MAG_TAPE_LOADED='Y'")

PLOTTER AND PG OUTPUT PARAMETERS

"LIBRARY"

Use this parameter to enter the library for all Plotter, PG, and Rulescheck jobs. GDS1T02, GDS2T01, Inform, and Outform all have specific parameters to their programs to enter the libraries with. (See INFORM and OUTFORM Paramaters below.) (LIBRARY = 'GOODIC5')

"STRUCTURE"

Use this parameter to enter structure names for the PG, PLOT, Rulescheck Jobs. GDS1T02, GDS2T01, Inform, and Outform have their own specialized parameters. (See INFORM and OUTFORM parameters below.) ("STRUCTURE='34'")

"MIR"

MIR will mirror data about the Y- parallel axis ("MIR")

"ROT"

ROT will rotate data 90-degrees clockwise ("ROT")

"CEN"

CEN will automatically center the plot ("CEN")

"PLT0"

This simply selects plotter option PLT0 ("PLT0")

"PLT1"

This simply selects plotter option PLT1 ("PLT1")

"PLT2"

This simply selects plotter option PLT2 ("PLT2")

"NOPLT"

This simply selects plotter option NOPLT ("NOPLT")

"MT0"

This simply selects plotter option MT0 ("MT0")

"MT1"

This simply selects plotter option MT1 ("MT1")

"MT2"

This simply selects plotter option MT2 ("MT2")

"NOMTA"

This simply selects plotter option NOMTA (no magtape - "NOMTA")

"NOOPT"

This cancels all plotter options except Magtape and Plotter options ("NOOPT")

"UPDATE"

This updates an existing tape ("UPDATE")

"AUTOGO"

This option means there will be no pause in the on_line plotter ("AUTOGO")

"SEGMENT"

Number of groups per exploder segments
("SEGMENT='4'" for 4 groups per segment)

"DIAG"

This option means diagnostics only - no tape. ("DIAG")

"EROV"

This option means error override - write tape in spite of errors! ("EROV")

"LAYER_GROUPS"

This is how you enter the layer groups for a job. If you entered "LAYER_GROUPS=1-4 5 6 7-12; 14 15 16-20 22; 24-28 30" there would be three plotter passes: #1 - 1-12, #2 - 14-20 22, #3 - 24-28 30.

"WINDOW"

The plot window x and y coordinates.
("WINDOW=1 -1; 10 10")
note: TAB is illegal in this case.

"SCALE"

The scalefactor. ("SCALE=100")

PARAMETERS FOR PG AND E-BEAM READ-IN

"PG_LIBRARY"

Library name to receive PG data ("PG_LIBRARY='IC5'")

"PG_STRUCTURE"

Structure name to receive PG data ("PG_STRUCTURE='34A'")

"PG_SCALE"

Scale at which PG tape was written ("PG_SCALE=400")

"PG_USE_TAPE_LAYERS"

This tells whether or not to use the layer numbers found on the tape when reading a David Mann PG tape. ("PG_USE_TAPE_LAYERS='Y'" means use the layer numbers found on the tape)

"PG_LAYER_INCREMENT"

Layer number increment for PG read-in. The layer number receiving the data will be the tape file number + N, where N is the argument. ("PG_LAYER_INCREMENT=20")

** INFORM PARAMETERS **

"INFORM_INPUT_FILE"

This is the input file, from which the progrm is to receive data. It consists of a tape drive and a file number. ("INFORM_INPUT_FILE='MT0:0'")

"INFORM_LIBRARY"

This is the library name to receive data. ("INFORM_LIBRARY='QAPLOTLIB'")

"INFORM_LIB_SIZE"

The library to receive data should be at lease 200 user units in size. ("INFORM_LIB_SIZE=200")

"INFORM_DBU_SIZE"

This is the size of database units for output. ("INFORM_DBU_SIZE='MIL'")

"INFORM_STRUCTURES"

These are the specific structures to be converted. ("INFORM_STRUCTURES='-'")

"INFORM_USER_DBUS"

This is the size of database units per user units. ("INFORM_USER_DBUS='1000'")

"INFORM_OPTIONS"

These options are fully described in the INFORM helpfile. ("INFORM_OPTIONS='UV'")
Note: Use with extreme caution; may cause system crash.

** OUTFORM PARAMETERS **

"OUTFORM_LIBRARY"

```
    This is the library to be converted.  ("OUTFORM_LIBRARY='IC5'")

"OUTFORM_FILE"

    The library name which is output to the stream file into the LIBNAME
    record.  Refer to STREAM.DC for more information.
    ("OUTFORM_FILE='SOMENAME'")

"OUTFORM_OUTPUT_FILE"

    The output file consists of a tape drive and a file number.
    ("OUTFORM_OUTPUT_FILE='MT0:0'")

 "OUTFORM_OPTIONS"

    These are described in detail in the OUTFORM helpfile.
    ("OUTFORM_OPTIONS='V'")

"OUTFORM_STRUCTURES"

    These are the specific structures to be converted.
    ("OUTFORM_STRUCTURES='-'")

All the above parameters are described in detail in the OUTFORM Helpfile.

------------------------------------------------------------------------

** STREAMOUT PARAMETERS **

"STREAMOUT_LIBRARY"

    This is the library to be converted.  ("STREAMOUT_LIBRARY='IC5'")

"STREAMOUT_FILE"

    The library name which is output to the stream file into the LIBNAME
    record.  Refer to STREAM.DC for more information.
    ("STREAMOUT_FILE='SOMENAME'")

"STREAMOUT_OUTPUT_FILE"

    The output file consists of a tape drive and a file number.
    ("STREAMOUT_OUTPUT_FILE='MT0:0'")

 "STREAMOUT_OPTIONS"

    These are described in detail in the OUTFORM helpfile.
    ("STREAMOUT_OPTIONS='V'")

"STREAMOUT_STRUCTURES"

    These are the specific structures to be converted.
    ("STREAMOUT_STRUCTURES='-'")

"STREAMOUT_MASK"

    Specifies the layers and data types as two vectors separated with
    a semi-colon.  Each set of layers and data types defines one mask.
    Several STREAMOUT_MASK parameters may be specified, and in this case
    the first parameter modifies the first mask, etc.
    ("STREAMOUT_MASK='1 4-6; 1-5'")
```

All above parameters are described in detail in the STREAMOUT Helpfile.

PARAMETERS FOR E-BEAM OUTPUT

NILADIC COMMANDS

"INVERT"

Acceptable only with a Toshiba job. This option produces reverse image output for the layer specified. ("INVERT")

"RETICLE"

Acceptable only with a Toshiba job. This option puts the output to reticle rather than master. ("RETICLE")

"DOT_SIZE"

Acceptable only with an ETEC job. This option specifies the output dot size in microns. ("DOT_SIZE=.25")

"STRIPE_HEIGHT"

Acceptable only with an ETEC job. This option specifies the output stripe height in address units. ("STRIPE_HEIGHT=256")

MONADIC COMMANDS

"CHIP = n m"

Acceptable only with a Toshiba job. Specifies the size of the largest chip to be plotted, where n and m are chip X and Y dimensions, respectively. If used, this option must be entered after WINDOW and SHIFT options if they are desired. Note: If in-line JOBCREATE is used only to save the job descriptor file (i.e., the second option argument is not " "), then the CHIP option will be overwritten by a default. ("CHIP = 20 25")

"COMMENT n = <character string>"

Acceptable with Toshiba and ST1FMI jobs. Allows user to assign a comment to each layer group (n). The character string may contain up to 14 characters. ("COMMENT 1 = GRP1 CMNT")

"COMPACT"

Outputs structure references and arrays in compact format. ("COMPACT")

"FNAME n = <filename>"

Acceptable only with a Toshiba job. Allows users to assign a filename to a layer group (n). The maximum number of characters for the filename is 12. ("FNAME 1 = GPR1NAME")

"INV n = <answer>"

This option produces reverse image output for the layer group,

(n), specified. "Answer" can be either yes (Y) or no (N). If used, this option must be entered after "Layer Groups" and "INVERT". ("INV 1 = Y")

"SHIFT = n m"

Acceptable only with a Toshiba job. Shifts all input coordinates by a specified distance (n) and (m) where n and m are displacements along the X and Y axes in microns. If used, this option shoud be entered after "ROT", if rotation is desired. ("SHIFT = 2 3.5")

"TOSHIBA_MODE = <mode>"

Acceptable only with a Toshiba job. The mode may be specified as A, B or C. This command should be entered before the "Scale" option. ("TOSHIBA MODE = A")

"DTYPE = <datatypes>"

Acceptable only with ETEC jobs. This option selects mask data by datatype. Valid datatypes range from 0 to 63. The format of datatype is the same as the format of arguments in "Layer Groups", except that semicolons (;) are not permitted. The single list of datatypes applies to all layer groups. ("DTYPE = 0-22 23")

"MODE12"

Acceptable only with an ETEC job. Causes output to be in ETEC I/II format, rather than extended address format. ("MODE12")

"OVERSIZE n = m"

Acceptable only with an ETEC job. This option specifies the amount of oversizing (in microns) to be done on each boundary or path for each layer group. The layer group number is specified as n, and m is the amount of oversizing. This option should be entered after "Layer Groups" and "DOTSIZE" if these two options are desired. ("OVERSIZE 1 = .5, OVERSIZE 2 = .75")

"PTYPE = <pathtypes>"

Acceptable only with an ETEC job. Selects data by pathtype. This option will affect path and text data that has the specified pathtype(s). Valid pathtypes are 0, 1 and 2. Note: Pathtype 1 is plotted as pathtype 2. ("PTYPE = 2")

"TTYPE = <texttypes>"

Acceptable only with an ETEC job. This option selects data by texttype. Valid texttypes are 0 to 63. The format of texttypes is the same as that of datatypes. (TTYPE = 0 10-12 15)

"UPDATE"

Updates a previously created tape by adding new data to the end of it and updating directory information. ("UPDATE")

PARAMETERS FOR PHOTOPLOTTERS OUTPUT

"BED_OFFSET = n m"

Acceptable only with a Photoplotters job. Sets Bed Offset option where n is the offset of X and m is the offset of Y, in inches or centimeters. ("BED OFFSET = 5 3.5")

"FILM_SIZE = n m"

Acceptable only with a Photoplotters job. This option sets the film size. n and m are the bed size in inches or centimeters. ("FILM SIZE = 65 45")

"FLASH_SORT = <n>X or = <n>Y

Acceptable only with a Photoplotters job. Sets flash band sort width and direction where n is the width. ("FLASH SORT = '2X'")

"FORMATTER = <formatter filename>"

Acceptable only with a Photoplotters job. This option sets the name of the formatter, if it is different from the default. ("FORMATTER = A1FORMAT")

"TOOL_DESCRIPTOR = <character string>"

Acceptable only with a Photoplotters job. The character string in this option sets the name of the Tool Descriptor filename if it is different from the default. ("TOOL DESCRIPTOR = 1TOOLDES")

**GDS1TO2 PARAMETERS **

"GDS1_LIBRARY"

For the GDS1 Library ("GDS1_LIBRARY='WARGAMES'")

"GDS2_LIBRARY"

For the GDS2 Library ("GDS2_Library='IC5'")

"GDS1_UNIT_SIZE"

GDS1 Physical Unit Size - M, CM, MM, MICRON, A, INCH, MIL ('GDS1_UNIT_SIZE='4 MICRON'")

"GDS1_LAYERS"

Layer number of conversion ("GDS1_LAYERS='4'")

"LINE_TABLE"

Linetable name ("LINE_TABLE='GRAPEFRUIT.LF'")

**GDS2TO1 PARAMETERS **

"GDS2TO1_LIBRARY"

```
    GDS2 Library.   ("GDS2T01_LIBRARY='IC5'")

"GDS2T01_STRUCTURE"

    GDS2T01 Structure mask.   ("GDS2T01_STRUCTURE='34-'")

"GDS2T01_ORIGIN"

    Coordinates to become origin of GDS data.
    ("GDS2T01_ORIGIN=400 400")

"GDS2T01_ORIGIN"

    Coordinates to become origin of GDS data.
    ("GDS2T01_ORIGIN=400 400")

"GDS2T01_LAYERS"

    Layers to be converted.
    ("GDS2T01_LAYERS='1-5 9 18 21-29'")

"GDS2T01_UNIT_SIZE"

    GDS1 physical unit size - M,CM, MM< MICRON, A, INCH, MIL.
    ("GDS2T01 UNIT SIZE = '5 MICRON'")

-------------------------------------------------------------------------

** APPLTOGDS2 KEYWORDS **

For a description of each prompt, see the Helpfile Annex APPLTOGDS2.

"APPL_LIBRARY"

    Same as "APPLE Library File" in interactive mode.
    ("APPL_LIBRARY='MT0:0'")

"APPL_GDS2_LIBRARY"

    Same as prompt "GDS2 Library Name".
    ("APPL_GDS2_LIBRARY='APPLELIB'")

"APPL_DB4"

    Number of Apple Database units per Physical Unit.
    ("APPL_DB4='30'")

"APPL_UNIT_SIZE"

    Same as "APPLE Physical Unit Size."
    ("APPL_UNIT_SIZE='M,MM,CM,MICRON,A INCH,MIL'")

"APPL_LAYERS"

    Same as "Layer Number." ("APPL_LAYERS='3,4,11'")

"APPL_STRUCTURE_MASK"

    Same as "Structure Mask."  ("APPL_STRUCTURE_MASK='-'")

"CREATE_BINS"
```

```
    Same as "Create Apple Bins Data."  ("CREATE_BINS='Y'")

 "EXPLODE_FX"

    Same as "Explode FX Rectangles?"  ("EXPLODE_FX='N'")

-------------------------------------------------------------------------

** READMP2D KEYWORDS **

"BANNING_IN_FILE"

   Name of tape input file for graphics.
   ("BANNING_INPUT_FILE='MT0:1'")

"STREAM_OUT_FILE"

   Name of disk or tape output file for graphics.
   ("STREAM_OUT_FILE='MT1:0'")

"SAVE_PINS"

   Determines whether or not to save the pin data file on disk.
   ("SAVE_PINS='Y'")

"PIN_IN_FILE"

   Name of tape input file for text.  ("PIN_IN_FILE='MT0:0'")

"PIN_OUT_FILE"

   Name of disk output file for text.
   ("PIN_OUT_FILE='BACKGROUND:STREAMFILE.SF'")

"MP2D_UNITS"

   Metric (M) or English (E) data.  ("MP2D_UNITS='E'")

"MP2D_SCALE"

   A positive number to determine scale factor.  ("MP2D_SCALE='1.0'")

-------------------------------------------------------------------------

** CLI DUMP AND LOAD PARAMETERS **

"DUMP_TO_FILE"

    The section of the tape to receive dump.  ("DUMP_TO_FILE='MT0:2'")


"DUMP_FILES"

    The files to be dumped.  ("DUMP_FILES='FILES. <AL,SR,BK,HP>'")

"LOAD_FROM_FILE"

    The section of the tape to Load from.  ("LOAD_FROM_FILE='MT0:2'")
```

```
"LOAD_FILES"

    The files to load.  ("LOAD_FILES='ROOTIE.RT TOOTIE.RT'")

------------------------------------------------------------------------

** PARAMETERS FOR STICKS SPACER **

"SPACER_IN_LIB"

    Spacer Input Library Name.  ("SPACER_IN_LIB='GDSII:CMOS$LIB.DB'")

"SPACER_OUT_LIB"

    Spacer Output Library Name.  ("SPACER_OUT_LIB='GDSII:CMOS$LIB.DB'")

"SPACER_IN_STRUC"

    Input Structure Name.  ("SPACER_IN_STRUC='DECODER'")

"SPACER_OUT_STRUC"

    Output Structure Name.  ("SPACER_OUT_STRUC='DECODER$'")

"SPACER_PASS_SEQ"

    Spacing Pass Sequence.  ("SPACER_PASS_SEQ='LD'")

"SPACER_TERM"

    Terminate if Overconstraint Found in First Pass.
    ("SPACER_TERM=N")

"SPACER_DIAG"

    Diagonal Check in Last Pass.  ("SPACER_DIAG=N")

"SPACER_VERIFY"

    Verify Input/Output Structure.  ("SPACER_VERIFY=N")

------------------------------------------------------------------------

** PARAMETERS FOR BACKGROUND CLI JOBS **

"CMD_LINE"

    A command line to execute.
    ("CMD_LINE='LOG;GMEM; ENDLOG;RENAME LOG.CM MINE.SR'")
------------------------------------------------------------------------

** Parameters for the DESIGN RULES CHECK job (DRC) **

"LIBRARY"

    The library file to do the design rule checking in.
    ("LIBRARY='IC'")

"STRUCTURE"

    The structure on which the design rule checks are to be
```

```
        performed.   ("STRUCTURE='IC5'")

    "ERRLIB"

        The library to which the errors from the design rules check will
        be put.   ("ERRLIB='ICER'")

    "ERRSTRUCT"

        The structure to which the DRC errors will be placed.
        ("ERRSTRUCT='IC5_ER'")

    "WINDOW"

        The window in which the design rule checks will be done.
        ("WINDOW= -1 4.89; 3.17 10")

    "DRYRUN"

        Whether to go ahead with the design rule checking after the
        compilation ("DRYRUN='Y'") or "DRYRUN='NO'") or ("DRYRUN='N'")
        or ("DRYRUN='YES'")

    "MAC_FILE"

        The file containing the Mask Analysis Commands, which is to be
        compiled (and executed if not a dry run) ("MAC_FILE='CHECKIT'")
        Note:  the extension ".ML" is assumed if no extension is given.
```

==========

Example programs using IN-LINE JOB PARAMETERS:

```
;Program Name:       SIZIT
;Created by:         Chris Martin
;Date:               Feb. 17, 1984
;
;Release:            4.1.0
;
;Purpose:            This program will write the necessary GPLII
;                    files for UNDERSIZING and OVERSIZING through
;                    DRC.
;
;       Revised:  April 4, 1985
;       The program will now write the necessary JOBCREATE and offer a
;       printout of the program to the user.

;       REQUIREMENT:     A JOBCREATE"DRC" must be set up prior to running this
;       program. It must be saved in a file called: DRCJOBS.JT. The jobcreate
;       should call for 'NO' DRC Parameters. A dummy name may be used for the
;       name of the GPL Procedure.
;       SEE JOBCREATE EXAMPLE AT END OF PROGRAM
NILADIC PROCEDURE SIZIT
EXTERNAL CHOICE
```

```
EXTERNAL OVSZ_DATA
EXTERNAL UNDRSZ_DATA
EXTERNAL OV_INPUT_LAYER
EXTERNAL OV_DATA
EXTERNAL N
EXTERNAL INPUT_LAYER
EXTERNAL FILE_NAME
EXTERNAL ANS
EXTERNAL DATA
EXTERNAL OV_TITLE
EXTERNAL OV_LINE
EXTERNAL OV_LINES
EXTERNAL XX
EXTERNAL NN
EXTERNAL UNDRSZ_LAYERS
EXTERNAL UN_OUTPUT_LAYERS
EXTERNAL AMOUNT
EXTERNAL X
EXTERNAL COMP_LAYER
EXTERNAL AMNT
EXTERNAL ONE_HALF_AMNT
EXTERNAL UN_OUT_LAYER
EXTERNAL TEST
EXTERNAL OVSZ_LAYERS
EXTERNAL OV_LAYER
EXTERNAL OV_OUTPUT_LAYERS
EXTERNAL OV_AMOUNT
EXTERNAL OV_MNT
EXTERNAL OV_AMNT
EXTERNAL OV_LAY
EXTERNAL LINES
EXTERNAL SPACES
EXTERNAL SIZE_SPACES
EXTERNAL TITLE
EXTERNAL LINE
EXTERNAL OLAY
EXTERNAL OV_MASK
EXTERNAL LAY
EXTERNAL OV_MASK_FILES
EXTERNAL MASK_FILES
EXTERNAL ZZ
EXTERNAL MSK
EXTERNAL IN_LIB;OUT_LIB;IN_STRUCT;OUT_STRUCT;NUM
EXTERNAL WIN;LL_CRD;UR_CRD;LIB_INFO;JNAME;JGPL
;
IF (FILEINFO "UNSZDATA.XX") <> "" THEN
FDELETE"UNSZDATA.XX"
ENDIF
IF (FILEINFO "OVSZDATA.XX") <> "" THEN
FDELETE"OVSZDATA.XX"
ENDIF


:JOBCREATE INFORMATION INPUT
"<BRON>THE INPUT AND OUTPUT LIBRARY NAMES MUST BE DIFFERENT<BROFF>"
""
""

PROG_START:
IN_LIB:=TEXTINPUT"INPUT LIBRARY NAME:   "
IF IN_LIB <> "" THEN
```

```
  IF NOT ("." IN IN_LIB) THEN
    IN_LIB:=IN_LIB,".DB"
  ENDIF
ENDIF
OUTLIB:

  IF (FILEINFO IN_LIB) <> "" THEN
    OUT_LIB:=TEXTINPUT"OUTPUT LIBRARY NAME (",("ERR",IN_LIB),"):   "
    IF OUT_LIB = "" THEN
      OUT_LIB:="GDSII:",("ERR",IN_LIB)
    ELSE
      OUT_LIB:="GDSII:",OUT_LIB
    ENDIF
    IF NOT ("." IN OUT_LIB) THEN
      OUT_LIB:=OUT_LIB,".DB"
    ENDIF

    IF (FILEINFO OUT_LIB) = "" THEN
      "(((ERROR))) OUTPUT LIBRARY DOES NOT EXIST"
      ""
      ANS:=TEXTINPUT"DO YOU WISH TO CREATE ERR",IN_LIB," (Y):   "
      IF (ANS = "") OR (ANS = "Y") THEN
        OPENLIB IN_LIB
        LIB_INFO:=INITLIB
        LIB_INFO[1]:="ERR",IN_LIB
        INITLIB LIB_INFO
        CLOSELIB
      ELSE
        GOTO OUTLIB
      ENDIF
      GOTO OUTLIB
    ENDIF
  ELSE
   "(((ERROR))) INPUT LIBRARY DOES NOT EXIST"
   ""
   GOTO PROG_START
 ENDIF


INSTRUCTURE:
IN_STRUCT:=TEXTINPUT"INPUT STRUCTURE NAME:   "
IF IN_STRUCT <> "" THEN
  IF (STRUCLIST IN_STRUCT;IN_LIB) <> "" THEN
    OUT_STRUCT:=TEXTINPUT"OUTPUT STRUCTURE NAME (",("ERR",IN_STRUCT),"):   "
    IF OUT_STRUCT = "" THEN
      OUT_STRUCT:="ERR",IN_STRUCT
    ENDIF
    ELSE
     "(((ERROR))) INPUT STRUCTURE DOES NOT EXIST"
     GOTO INSTRUCTURE
    ENDIF
  ELSE
   "(((ERROR))) INPUT STRUCTURE MUST BE SPECIFIED"
   GOTO INSTRUCTURE
ENDIF

WIN:=TEXTINPUT"COMPUTE WINDOW TO INCLUDE ALL (Y):   "
  IF (WIN <> "") AND (WIN <> "Y") AND (WIN <> "YES") THEN

  CRDS:
```

```
  LL_CRD:=EXPINPUT"ENTER LOWER LEFT COORDINATE:   "
    IF LL_CRD <> "" THEN
      UR_CRD:=EXPINPUT"ENTER UPPER RIGHT COORDIANTE:   "
        IF UR_CRD = "" THEN
          "(((ERROR))) ILLEGAL COORDINATE"
          GOTO CRDS
        ENDIF
    ELSE
     "(((ERROR))) ILLEGAL COORDINATE"
     GOTO CRDS
    ENDIF

  ELSE
   WIN:=""
  ENDIF


:END OF JOBCREATE INFORMATION


:
OVSZ_DATA:=""
UNDRSZ_DATA:=""
AUTOHOLD 0
CHOICES:
""
"Please indicate which operation you wish to perform:"
"           1 <===== UNDERSIZING"
"           2 <===== OVERSIZING"
"           3 <===== BOTH"
""
CHOICE:=TEXTINPUT"           <BRON>Operation Number:<BROFF>   "
IF CHOICE = "" AND NOT (CHOICE IN "123") THEN
  "<BEL><BRON>(((ERROR))) That operation does not exist.<BROFF>"
  GOTO CHOICES
ENDIF

IF CHOICE = "1" OR CHOICE = "3" THEN
UNDERSIZING:
""
"Please enter <BRON>LAYERS<BROFF> for <BRON>UNDERSIZING<BROFF>:"
""
UNDRSZ_LAYERS:=EXPINPUT "LAYER:   "
IF UNDRSZ_LAYERS <> "" AND (UNDRSZ_LAYERS >=0 AND UNDRSZ_LAYERS <=63) THEN
  UNDRSZ_LAYERS:=CFORMAT UNDRSZ_LAYERS
UNDRSZ_LAYERS:=UNDRSZ_LAYERS[IOTA (1, ((SIZE UNDRSZ_LAYERS) - 1))]
  DO
  ADDLAYERS:
  LAY:=EXPINPUT "LAYER (END):   "
  IF LAY <> "" AND (LAY < 0 OR LAY > 63) THEN
    "<BEL><BRON>(((ERROR))) Illegal Input. <BROFF>"
    GOTO ADDLAYERS
  ENDIF
  IF LAY >=0 AND LAY <=63 THEN
    LAY:=CFORMAT LAY
     LAY:=LAY[IOTA (1, ((SIZE LAY) - 1))]
    UNDRSZ_LAYERS:=UNDRSZ_LAYERS;LAY
  ENDIF
  UNTIL LAY = ""
  ENDDO
ELSE
```

```
"<BEL><BRON> You must specify input layers. <BROFF>"
GOTO UNDERSIZING
ENDIF


UN_OUTPUT_LAYERS:=""
AMOUNT:=""

""
"Please enter the <BRON>TOTAL UNDERSIZE<BROFF> in user units"
""
X:=0
DO
X:=X + 1
COMP_LAYER:=UNDRSZ_LAYERS[X]
USERUNITS:
""
AMNT:=EXPINPUT "Layer <BRON>",COMP_LAYER,":  <BROFF>"
IF AMNT <> "" AND AMNT >=0 THEN
  ONE_HALF_AMNT:=AMNT % 2
  AMNT:=CFORMAT ONE_HALF_AMNT
  AMOUNT:=AMOUNT;AMNT
ELSE
"<BEL><BRON>(((ERROR))) Illegal Input.<BROFF>"
GOTO USERUNITS
ENDIF

OUTPUTLAYERS:
UN_OUT_LAYER:=TEXTINPUT "Output layer for<BRON> ",COMP_LAYER,":  <BROFF>"
TEST:=EXECUTE UN_OUT_LAYER
IF UN_OUT_LAYER <> "" AND (TEST IN (IOTA 0 63)) THEN
  UN_OUTPUT_LAYERS:=UN_OUTPUT_LAYERS;UN_OUT_LAYER
ELSE
"<BEL><BRON> Illegal Input.<BROFF>"
GOTO OUTPUTLAYERS
ENDIF


UNTIL X = (LENGTH UNDRSZ_LAYERS)
ENDDO
ENDIF


IF CHOICE = "2" OR CHOICE = "3" THEN
OVERSIZING:
""
"Please enter <BRON>LAYERS<BROFF> for <BRON>OVERSIZING<BROFF>"
""
OVSZ_LAYERS:=EXPINPUT "Layer:  "
IF OVSZ_LAYERS <> "" AND (OVSZ_LAYERS >=0 AND OVSZ_LAYERS <=63) THEN
  OVSZ_LAYERS:=CFORMAT OVSZ_LAYERS
  OVSZ_LAYERS:=OVSZ_LAYERS[IOTA (1, ((SIZE OVSZ_LAYERS) - 1))]
DO
ADDLAYERS1:
OV_LAYER:=EXPINPUT "Layer (END):  "
  IF OV_LAYER <> "" AND (OV_LAYER < 0 OR OV_LAYER > 63) THEN
    "<BRON><BEL> Illegal Input. <BROFF>"
    GOTO ADDLAYERS1
  ENDIF
  IF OV_LAYER >=0 AND OV_LAYER <=63 THEN
    OV_LAYER:=CFORMAT OV_LAYER
```

```
      OV_LAYER:=OV_LAYER[IOTA (1, ((SIZE OV_LAYER) - 1))]
     OVSZ_LAYERS:=OVSZ_LAYERS;OV_LAYER
  ENDIF
UNTIL OV_LAYER = ""
ENDDO
ELSE
"<BRON><BEL>(((ERROR))) You must specify input layer(s). <BROFF>"
GOTO OVERSIZING
ENDIF



OV_OUTPUT_LAYERS:=""
OV_AMOUNT:=""


"Please enter the <BRON>TOTAL OVERSIZE<BROFF> in user units"
""
X:=0
DO
X:=X + 1
COMP_LAYER:=OVSZ_LAYERS[X]
OVSZLAYERS:
""
OV_AMNT:=EXPINPUT "Layer <BRON>",COMP_LAYER,":  <BROFF>"
IF OV_AMNT <> "" AND OV_AMNT >=0 THEN
  OV_MNT:=OV_AMNT % 2
  OV_AMNT:=CFORMAT OV_MNT
  OV_AMOUNT:=OV_AMOUNT;OV_AMNT
ELSE
"<BEL><BRON>(((ERROR))) Illegal Input.<BROFF>"
GOTO OVSZLAYERS
ENDIF

OVOUTLAYERS:
OV_LAY:=TEXTINPUT "Output layer for<BRON> ",COMP_LAYER,":  <BROFF>"
TEST:=EXECUTE OV_LAY
IF OV_LAY <> "" AND (TEST IN (IOTA 0 63)) THEN
  OV_OUTPUT_LAYERS:=OV_OUTPUT_LAYERS;OV_LAY
ELSE
"<BEL><BRON> Illegal Input.<BROFF>"
GOTO OVOUTLAYERS
ENDIF



UNTIL X = (LENGTH OVSZ_LAYERS)
ENDDO
ENDIF

IF CHOICE = "1" OR CHOICE = "3" THEN
LINES:=""
SPACES:="..........."
SIZE_SPACES:=SIZE SPACES

TITLE:="THE FOLLOWING UNDERSIZES WILL BE WRITTEN:<CR>"
TITLE:=TITLE,"<CR>"
TITLE:=TITLE,"INPUT LAYER : UNDERSIZE : OUTPUT LAYER<CR>"
TITLE:=TITLE,"<BRON>              (PER EDGE)<BROFF>"

TITLE:=TITLE,"<CR>=======================================<CR>"
```

```
NN:=0
XX:=1
DO
XX:=XX + 1
NN:=NN + 1
LINE:="      ",UNDRSZ_LAYERS[NN]
LINE:=LINE,SPACES[IOTA (((SIZE UNDRSZ_LAYERS[NN]) + 1), (SIZE SPACES))]
LINE:=LINE,"(",AMOUNT[XX],")"
LINE:=LINE,SPACES[IOTA (((SIZE AMOUNT[XX]) + 1), (SIZE SPACES))]
LINE:=LINE,UN_OUTPUT_LAYERS[XX],"<CR>"
LINES:=LINES,LINE
UNTIL NN = (LENGTH UNDRSZ_LAYERS)
ENDDO

LINES:=TITLE,LINES

LINES
"UNDERSIZES.XX" SAVE LINES
ENDIF



IF CHOICE = "2" OR CHOICE = "3" THEN
OV_LINES:=""
SPACES:="..........."
SIZE_SPACES:=SIZE SPACES

OV_TITLE:="THE FOLLOWING OVERSIZES WILL BE WRITTEN:<CR>"
OV_TITLE:=OV_TITLE,"<CR>"
OV_TITLE:=OV_TITLE,"INPUT LAYER : OVERSIZE : OUTPUT LAYER<CR>"
OV_TITLE:=OV_TITLE,"<BRON>                (PER EDGE)<BROFF>"

OV_TITLE:=OV_TITLE,"<CR>=========================================<CR>"

NN:=0
XX:=1
DO
XX:=XX + 1
NN:=NN + 1
OV_LINE:="      ",OVSZ_LAYERS[NN]
OV_LINE:=OV_LINE,SPACES[IOTA (((SIZE OVSZ_LAYERS[NN]) + 1), (SIZE SPACES))]
OV_LINE:=OV_LINE,"(",OV_AMOUNT[XX],")"
OV_LINE:=OV_LINE,SPACES[IOTA (((SIZE OV_AMOUNT[XX]) + 1), (SIZE SPACES))]
OV_LINE:=OV_LINE,OV_OUTPUT_LAYERS[XX],"<CR>"
OV_LINES:=OV_LINES,OV_LINE
UNTIL NN = (LENGTH OVSZ_LAYERS)
ENDDO
OV_LINES:=OV_TITLE,OV_LINES

OV_LINES
"OVERSIZES.XX" SAVE OV_LINES
ENDIF


ANS:=TEXTINPUT "Do you want a printout ? (Y):  "
IF ANS = ""  THEN
ANS:="Y"
ENDIF
IF  (ANS[1] IN "Yy") THEN
  IF  CHOICE = "1" OR CHOICE = "3" THEN
```

```
    FCOPY "UNDERSIZES.XX";"$LPT"
  ENDIF
  IF CHOICE = "2" OR CHOICE = "3" THEN
    FCOPY "OVERSIZES.XX";"$LPT"
  ENDIF
ENDIF

ONCEMORE:
FILE_NAME:=TEXTINPUT "Please enter program name:  "
IF FILE_NAME <> "" THEN
""
"<BRON>PLEASE STAND BY...<BROFF>"
""

G"NULL"
I"NILADIC PROCEDURE ",FILE_NAME,"<CR>"
A"!<CR>"
Z
IF CHOICE = "1" OR CHOICE = "3" THEN
MASK_FILES:="!<CR>"
ZZ:=0
DO
ZZ:=ZZ + 1
LAY:=UNDRSZ_LAYERS[ZZ]
MSK:="MASK FILE UN_LAY",LAY,"<CR>"
MSK:=MSK,"MASK FILE NEWUN_LAY",LAY,"<CR>"
MASK_FILES:=MASK_FILES,MSK
UNTIL ZZ = (LENGTH UNDRSZ_LAYERS)
ENDDO
"UNMASKS.XX" SAVE MASK_FILES
IFILE "UNMASKS.XX"
Z
IF (CHOICE = "1") AND (CHOICE <> "3") THEN
:JOB PARAMS INPUT INFORMATION

A "OPENLIB <QT>GDSII:",IN_LIB,"<QT><CR>"
A "OSTRUCT <QT>",IN_STRUCT,"<QT><CR>"
A "OUTPUTCHANGE <QT>",OUT_LIB,"<QT>;<QT>",OUT_STRUCT,"<QT><CR>"
Z
IF WIN = "" THEN
  A "OUTPUTVIEW <CR>"
ELSE
  A "OUTPUTVIEW ",(CFORMAT LL_CRD)," ",(CFORMAT UR_CRD),"<CR>"
ENDIF
A "! <CR>"
Z

ENDIF
:END OF JOB PARAMS INPUT

IF   CHOICE = "3" THEN
OV_MASK_FILES:="!<CR>"
ZZ:=0
DO
ZZ:=ZZ + 1
OLAY:=OVSZ_LAYERS[ZZ]
OV_MASK:="MASK FILE OV_LAY",OLAY,"<CR>"
OV_MASK:=OV_MASK,"MASK FILE NEWOV_LAY",OLAY,"<CR>"
OV_MASK_FILES:=OV_MASK_FILES,OV_MASK
UNTIL ZZ = (LENGTH OVSZ_LAYERS)
ENDDO
```

```
"OVMASKS.XX" SAVE OV_MASK_FILES
IFILE "OVMASKS.XX"
Z
:JOB PARAMS INPUT INFORMATION

A "OPENLIB <QT>GDSII:",IN_LIB,"<QT><CR>"
A "OSTRUCT <QT>",IN_STRUCT,"<QT><CR>"
A "OUTPUTCHANGE <QT>",OUT_LIB,"<QT>;<QT>",OUT_STRUCT,"<QT><CR>"
Z
IF WIN = "" THEN
  A "OUTPUTVIEW <CR>"
ELSE
  A "OUTPUTVIEW ",(CFORMAT LL_CRD),",",(CFORMAT UR_CRD),"<CR>"
ENDIF
A ":  <CR>"
Z
ENDIF


X:=0
N:=1
DO
X:=X + 1
N:=N + 1
INPUT_LAYER:="UN_LAY",UNDRSZ_LAYERS[X]
DATA:=":<CR>"
DATA:=INPUT_LAYER,":=INPUTMASK ",UNDRSZ_LAYERS[X]
DATA:=DATA,";(IOTA 0 63)<CR>"
DATA:=DATA,"NEW",INPUT_LAYER
DATA:=DATA,":=UNDERSIZE ",AMOUNT[N]
DATA:=DATA,";",INPUT_LAYER,"<CR>"
DATA:=DATA,"OUTPUTMASK NEW",INPUT_LAYER,";"
DATA:=DATA,UN_OUTPUT_LAYERS[N],";<QT>JOBLOG<QT>;  "
DATA:=DATA,"0;200<CR>"
DATA:=DATA,":<CR>"
DATA:=DATA,"MASKFREE ",INPUT_LAYER,"<CR>"
DATA:=DATA,"MASKFREE NEW",INPUT_LAYER,"<CR>"
DATA:=DATA,":<CR>"
UNDRSZ_DATA:=UNDRSZ_DATA,DATA
UNTIL X = (LENGTH UNDRSZ_LAYERS)
ENDDO



"UNSZDATA.XX" SAVE UNDRSZ_DATA
IFILE "UNSZDATA.XX"

ENDIF


IF    CHOICE = "2" THEN
OV_MASK_FILES:=":<CR>"
ZZ:=0
DO
ZZ:=ZZ + 1
OLAY:=OVSZ_LAYERS[ZZ]
OV_MASK:="MASK FILE OV_LAY",OLAY,"<CR>"
OV_MASK:=OV_MASK,"MASK FILE NEWOV_LAY",OLAY,"<CR>"
OV_MASK_FILES:=OV_MASK_FILES,OV_MASK
UNTIL ZZ = (LENGTH OVSZ_LAYERS)
ENDDO
```

```
"OVMASKS.XX" SAVE OV_MASK_FILES
IFILE "OVMASKS.XX"
Z
;JOB PARAMS INPUT INFORMATION

A "OPENLIB <QT>GDSII:",IN_LIB,"<QT><CR>"
A "OSTRUCT <QT>",IN_STRUCT,"<QT><CR>"
A "OUTPUTCHANGE <QT>",OUT_LIB,"<QT>;<QT>",OUT_STRUCT,"<QT><CR>"
Z
IF WIN = "" THEN
  A "OUTPUTVIEW <CR>"
ELSE
  A "OUTPUTVIEW ",(CFORMAT LL_CRD)," ",(CFORMAT UR_CRD),"<CR>"
ENDIF
A ";  <CR>"
Z

;END OF JOB PARAMS INPUT
ENDIF


IF CHOICE = "2" OR CHOICE = "3" THEN


X:=0
N:=1
DO
X:=X + 1
N:=N + 1
OV_INPUT_LAYER:="OV_LAY",OVSZ_LAYERS[X]
OV_DATA:=OV_INPUT_LAYER,":=INPUTMASK ",OVSZ_LAYERS[X]
OV_DATA:=OV_DATA,";(IOTA 0 63)<CR>"
OV_DATA:=OV_DATA,"NEW",OV_INPUT_LAYER
OV_DATA:=OV_DATA,":=OVERSIZE ",OV_AMOUNT[N]
OV_DATA:=OV_DATA,";",OV_INPUT_LAYER,"<CR>"
OV_DATA:=OV_DATA,"OUTPUTMASK NEW",OV_INPUT_LAYER,";"
OV_DATA:=OV_DATA,OV_OUTPUT_LAYERS[N],";<QT>JOBLOG<QT>;  "
OV_DATA:=OV_DATA,"0;200<CR>"
OV_DATA:=OV_DATA,";<CR>"
OV_DATA:=OV_DATA,"MASKFREE ",OV_INPUT_LAYER,"<CR>"
OV_DATA:=OV_DATA,"MASKFREE NEW",OV_INPUT_LAYER,"<CR>"
OV_DATA:=OV_DATA,";<CR>"
OVSZ_DATA:=OVSZ_DATA,OV_DATA
UNTIL X = (LENGTH OVSZ_LAYERS)
ENDDO



"OVSZDATA.XX" SAVE OVSZ_DATA
IFILE "OVSZDATA.XX"
ENDIF
ELSE
GOTO ONCEMORE
ENDIF



Z
A"ENDSUB<CR>"
X:=0
DO
```

```
X:=X + 1
"<CR>"
UNTIL X = 20
ENDDO
W (FILE_NAME,".GS")
GPL FILE_NAME;"N"
""
""
"<BRON>PROGRAM:  ",(FILE_NAME,".GS")," HAS BEEN CREATED<BROFF>"
""
""
""
ANS:=TEXTINPUT"<BRON>DO YOU WISH TO RUN DRC JOB (N):  <BROFF>"
IF (ANS = "") OR (ANS = "N") THEN
  JNAME:="JOB_NAME='",FILE_NAME,"'"
  JGPL:="GPL_PROCEDURE='",FILE_NAME,"'"
  JOBCREATE"DRCJOBS.JT";(FILE_NAME,".JC");(JNAME,JGPL)
  ""
  "JOB TEMPLATE FILE SAVED:  <BRON>",(FILE_NAME,".JC"),"<BROFF>"
  ""
ELSE
  JNAME:="JOB_NAME='",FILE_NAME,"'"
  JGPL:="GPL_PROCEDURE='",FILE_NAME,"'"
  JOBCREATE"DRCJOBS.JT";"";(JNAME,JGPL)
ENDIF


ANS:=TEXTINPUT"DO YOU WANT A PRINTOUT OF THE GPLII PROGRAM (Y):  "
IF (ANS = "") OR (ANS = "Y") THEN
""
"        1 ==> $LPT"
"        2 ==> $TTO"
NUM:=TEXTINPUT"<TAB><TAB>ENTER NUMBER:  "
IF NUM <> "" THEN
  SWITCH, NUM OF
    CASE "1":
     FCOPY (FILE_NAME,".GS");"$LPT"
    CASE "2":
     FCOPY (FILE_NAME,".GS");"$TTO"
  ENDSWITCH
ENDIF
ENDIF

"<BRON>SIZIT PROGRAM IS COMPLETE<BROFF>"
ENDSUB
THE GPLII PROGRAM "XXCM.GS" HAS ALREADY BEEN WRITTEN.
IT MAY BE ANY SIMPLE SUBROUTINE.

? JOBCREATE
Job Type (XCLI): DRC
Priority Class (B): B
Jobname (DRC):
Turn on DRC verbose mode? (N):
Are DRC parameters necessary? (Y): N
GPL Object Procedure: XXCM
Run, Save, or Abort (RUN): SAVE
Filename?: DRCJOBS.JT
?
```

```
:PROGRAM:    AUTOPLOT
:BY:         PETER ERDMAN
:DATE:       3-12-85

NILADIC PROCEDURE AUTOPLOT
EXTERNAL VKND;SSZE;LLCRD;URCRD;PMAG;USTR;STRUC;TXT;UMAG
EXTERNAL DELX;DELY;TXPOS;FNT;INF;HGT;PAR;SDTP;SKND;SLAY

VKND:=VKIND
SKND:=SKIND
SLAY:=SLAYER
SDTP:=SDTYPE
SSZE:=SLIMSIZE

LLCRD:=EXPINPUT "ENTER <BRON>LOWER LEFT<BROFF> COORDINATE: "
URCRD:=EXPINPUT "ENTER <BRON>UPPER RIGHT<BROFF> COORDINATE: "
PMAG:=EXPINPUT "ENTER PLOT <BRON>MAG<BROFF>NIFICATION FACTOR: "
USTR:=TEXTINPUT "ENTER YOUR USER STRING: "
VKIND "BDTXPB"
TEXT
STRUC:=OSTRUCT
UMAG:=CFORMAT PMAG
TXT:=STRUC," at ",UMAG," X<CR>",USTR,"<CR>",GTIME
ENTERTEXT TXT
LAYER 59
DATATYPE 59
RT LLCRD, URCRD
TJUST"T";"L"
PATHTYPE 0
WIDTH 0
ANGLE 0
REFL "N"
FONT 3
DELX:=ABS(URCRD[1] - LLCRD[1])
DELY:=ABS(URCRD[2] - LLCRD[2])
TXPOS:=(LLCRD[1] + (.02*DELY)),(LLCRD[2] - (.02*DELY))
CE TXPOS
:FNT:=BINDFONTS
:FNT:=FNT[3]
:INF:=FONTINFO FNT
HGT:=1.4*3
PMAG:=((.12*DELY)%HGT)
MAG PMAG
PUT
OUTPUTVIEW LLCRD, URCRD
SLIMSIZE 0
PAR:="JOB_NAME='MS",STRUC,"' SCALE='",UMAG,"'"
JOBCREATE "Z$AUTOPLOT.JD";"";PAR
SLAYER 59
SDTYPE 59
SKIND "TXBD"
DATADELETE"Y"
SLAYER SLAY
SDTYPE SDTP
SKIND SKND
SLIMSIZE SSZE
LLCRD:=(LLCRD[1]-(.12*DELX)),(LLCRD[2]-(.12*DELY))
```

```
URCRD.=(URCRD[1]+(.12*DELX)),(URCRD[2]+(.12*DELY))
SETVIEW LLCRD,URCRD

ENDSUB


A JOBCREATE MUST FIRST BE SAVE IN FILE:  Z$AUTOPLOT.JD

EXAMPLE:

? JOBCREATE
Job Type (XCLI): OVERSA8242
Priority Class (B):
Jobname (OVERSA8242): CMPLOT
Plot Options (PLTR):
Library Name: GPL$CELLS
Structure Name: CMOS$NOR2
Plot Window
  CE1 (0.,0.):
  CE2 (0.,0.): 10 10
Scale Factor (103930.): 1000

Plot Size Will be        0.3937 By     0.3937 Inches
Layer Groups: 1 2 3 4 5 7 13 15
Assign fillcodes to layers
     L1 (1):
     L2 (2): 85
     L3 (86): 15
     L4 (16): 34
     L5 (35): 80
     L7 (81): 9
     L13 (10): 7
     L15 (8): 12
Run, Save, or Abort (RUN): SAVE
Filename?: Z$AUTOPLOT.JD
?
```

APPENDICES

APPENDIX A

GPLII PROGRAM FUNCTION LISTING

PROGRAM FUNCTION LISTING

MAGLIB	This program will process the GDSII database file on a structure-by-structure basis. It will magnify the selectable data elements by a specified amount.
FINDANGLE	This program will look in every structure within the database and locate any SREFs which have been placed with an ABSOLUTE ANGLE. It will report the element key number of the SREF and store the information in a log file.
CM$EDIT	This program may be used to edit text string nodes. It works much like the 'CH' command for the GDSII text editor.
SCRIBE	This program will build a scribe structure given the final die size, layers and interior delta distances for each scribe layer. It will produce a fracturable scribe that the final device may then be placed within.
PGTITLES	This program will set up titling for P.G. fracture for each layer to be fractured. A fracturable text font library MUST first be built.
MYSETUP	This program may be used to set up the GDSII workstation.
T$WSETUP	This program will "read" the current parameters of the GDSII workstation and write a GPLII set up program.
CUSMENU	This program will build a custom menu. The user need only input the commands that will be included in the menu and the program will write all of the necessary files. CALMAFONT must be present. At this time, this program is only implemented for MRD and MHD type display screens.
CFMT	This is a custom CFORMAT program used by the CUSMENU program.
CLEANUP	This program is a custom program written for the CUSMENU program. It locates dashes within a command string and removes them.
SIZIT	This program automates the process of writing UNDERSIZING and OVERSIZING DRC programs. It first writes the GPLII DRC program and then enters the necessary JOBCREATE to run the DRC.
AUTODONUT	This program will digitize a specified boundary into a donut shaped boundary by specifying the interior window.
AUTOPLOT	This program will write an OVERSA8242 JOBCREATE from the input of the user. The input will include the plot window and the scale. See the program example for further information.
AREA	This program will calculate the area of a specified boundary or path.

NUMPTS	This program will tell the user the number of vertices of a specified boundary.
SUBMAT	This program is a custom CFORMAT program that will eliminate the decimal from the end of a real number that has been changed to character data.
BLDFONTLIB	This program will copy a defining (BOX only) text structure and build a structure for each text font. This program would be used to simply copy any reference-type data throughout the database (i.e., the text font defining BOX).
SHOWCRDS	This program will display the order in which a specified boundary or path has been entered into the database.
LOCATE	This function will return the location of a character or set of characters within a vector.
ENCLOSEDBY	This program will determine if one layer is completely enclosed by another. See the program example for further information.
INSIDE	This function is used by the ENCLOSEDBY program. It will determine if a specified coordinate is "inside" an array of coordinates.
PLEXIT	This program is used by the ENCLOSEDBY program to PLEX one geometry to another.
PASSWORD	This program may be used as a simple security program. The user is prompted to enter a password before they may obtain access to the system.
FIRSTPNT	This program will display the first coordinate of a specified boundary or path.
WINMARK	This program will mark all elements within a specified window.
ROUNDIT	This DRC program will round geometries to a specified precision.
IDNODEPROP	This program will identify and display all the user properties that may have been assigned a specific NODE.
SCALTEST	This program is used to illustrate the usage of the SCALARINPUT function.

APPENDIX B

GPLII KEYWORDS

GPL KEYWORDS

ABS
ARCTAN
BOOLEAN
COS
DROP
EQ
EXTERNAL
ENDSWITCH
FUNCTION
GRASEDOWN
GLOBAL
IN
LN
LT
MAX
MASK
NEQ
NOT
PI
RANK
RESHAPE
(*) SIGNUM
SORT
SCALAR
THEN
XOR
WHILE
NILADIC
SCALAR
REAL
LOGICAL
ENDSUB

(+) ADDITION
(:=) ASSIGNMENT
CEILING
DEC
DYADIC
EXP
ELIF
FALSE
FILE
GRADEUP
GOTO
INDEXOF
LOGBASE
LIST
MIN
MOD
NOR
MONADIC
POWER
(,) RAVEL
REAL
SIN
SORTDOWN
(-) SUBTRACTION
TAKE
UNTIL
SWITCH
INTEGER
OF
TRUE
VECTOR

AND
ARRAY
CHAR
(%) DIVISION
DO
ENDDO
ELSE
FLOOR
GEQ
GT
(+) IDENTITY
IOTA
(;) LIST CONCATENATION
LOCAL
(*) MULTIPLICATION
NAND
(-) (#) NEGATION
OR
PROCEDURE
(%) RECIPROCAL
SHAPE
SIZE
[ ] SUBSCRIPTION
TAN
TYPEOF
(,) VECTOR CONCATENATION
OUT
CASE
ERRTRAP
ENDIF
CASE

APPENDIX C

GPLII ERROR MESSAGES

GPLII ERROR MESSAGES

The following list of GPLII error messages is current as of GDSII release 4.0.8. The explanations included with each message indicate the most common cause(s) of that message which may not be the same cause of the problem you are trying to solve.

Some pointers in debugging a program might include:

1. After making corrections in the text editor, remember to W, GPL, and LOAD. LOADing will only take place automatically if the program in question is not in the work area.

2. If you have an error message pointing to a statement that you KNOW is correct, fix the other errors first. Some errors cause other errors.

3. The compiler makes three passes through the source code, and will find different types of errors on each pass. Therefore, it is possible to get some "Pass One Errors" or "Pass Two Errors", correct them, and get more errors.

4. Interactively, elided output may be implemented to examine the contents of external variables. You will either get the value (indicating that the program made it at least to the initializaton of that variable) or the error "Variable Not Initialized". In the case of a counter, you can determine how many times a loop was completed.

   Local and global variables may not be examined after the program is complete, however, such a variable may be stored in an external variable which could be examined upon program completion.

5. On rare occasions, G - Code may be used to debug a program. This is generally used when the program has defied all your attempts at debugging.

6. Elided output is a must when it comes to debugging. It may be used in a program as well as interactively. Inside your program, it can tell you where the error in question is occurring. Suppose you do not know where the program is choking. You could put elided output flags every 10 lines, thereby narrowing the region in which the error exists to 10 lines:

```
.
.
.
0
code
code
code
code
code
code
code
code
code
code
10
code
code
code
code
code
code
code
code
code
code
20
code
code
code
.
.
.
30
.
.
.
40
.
.
.
```

Suppose the program example above looked like this when executed:

```
? PROGRAM
  0
  10
  20
***ERROR*** EOF ENCOUNTERED INSIDE STRING
```

It can now be determined that the program failed between the 20 and 30 flags. The next step might be to add more flags between the lines in that region until you have located the line in error. Using this technique can save you a lot of time hunting for a mistake.

All of the error messages are followed by a word or words indicating when such a message would appear. They are interpreted as follows:

1. interactive - during interactive manipulation of GPLII statements.
2. compilation - while compiling the program using the GPL command.
3. runtime - during the actual execution of the program.
4. load - while LOADing the program with the LOAD command.

1. ACCESS CONTROL BLOCK NOT OPEN
runtime, interactive

An attempt to subscript a scalar will cause this. If your program subscripts a variable, and there is any chance that the variable might turn out to be a scalar, then your program is susceptible.

Example:

```
.
.
.
VAR := 7
FAR := VAR[1]
.
.
.
```

A suggested workaround is to ravel the scalar into a vector first.

Example:

```
FAR := (,VAR)[1]
```

2. ARGUMENT n NOT NULL
runtime, interactive

The database element format list is unforgiving. If you wish to create an element in the database, be absolutely sure that the list is accurate.

For example, the seventh position of the element list is for the transformation specifier properties (REFL, ANGLE, MAG). A boundary does not have these parameters. If you attempt to create a boundary with a position 7, you will get this error.

Example:

```
? CORDS := 5 2 RESHAPE 0 0 10 0 10 10 0 10
? PUTEL 3 0 -1; 7; 0; ; ; CORDS; 0 0 1
```

It is correct to leave a null vector in position 7, or position 7 may be omitted entirely. When working interactively, it is irrelevent whether or not the quotes (" ") are used to indicate a null position. They are only necessary to satisfy the compiler's desire to see an argument on both sides of a semicolon (;).

Example: ? PUTEL 3 0 -1; 7; 0; ; ; CORDS; " "

is the same as:

? PUTEL 3 0 -1; 7; 0; ; ; CORDS

3. ARGUMENT LEFT ON STACK
runtime, interactive

A procedure may be declared as a function.

4. ARGUMENT REQUIRED
compilation

If the variable into which the argument of a monadic or dyadic program will be stored is omitted from the program declaration line, this message will appear.

Example: MONADIC FUNCTION OHMS := FUNC

Should read: MONADIC FUNCTION OHMS := FUNC ARG

5. ATTEMPT TO CREATE AN ILLEGAL ELEMENT
runtime, interactive

This message will appear if an attempt is made to do a PUTEL with an invalid list.

Example: PUTEL 3 0 -2; 7; 0; " "; " "; 7 0

In the example above, the boundary only has one coordinate.

6. CLEAR MUST NOT BE USED WITHIN A GPLII PROCEDURE
runtime

As of release 4, it is not possible to use CLEAR within a GPLII program.

7. CONTROL VARIABLE NOT LOCAL
compilation

If the variable used in a FOR loop is declared as external this error may appear.

Example:
```
EXTERNAL CONTROL_VAR

FOR CONTROL_VAR RANGE (IOTA 0 9) DO
    .
    .
    .
ENDDO
```

The FOR statement is not implemented yet anyway.

8. DIVISION BY ZERO
runtime, interactive

If an attempt to divide by 0 is made, it will cause this error.

Example:

```
? 242 70 19 % 0 7 0
DIVISION BY ZERO
DIVISION BY ZERO
  242. 10. 19.
```

9. DOUBLY DECLARED VARIABLE
compilation

A variable or subroutine has been declared more than once.

10. DYADIC OPERATOR REQUIRED
compilation

Two sequential expressions in a line require a dyadic operator between them.

11. ELEMENT DOES NOT EXIST
runtime, interactive

An attempt to use GETEL with a non-existent key will cause this.

Example:

```
GETEL 99
```

12. ELSE OUTSIDE IF
ENDIF OUTSIDE IF
compilation

You may have misspelled or omitted some other keyword such as an ENDDO or ENDSWITCH.

13. END OF FILE
runtime, interactive

Using DIGMODE with a null vector as an argument will cause this.

Example:

```
DIGMODE " "
```

Also, use of a DATA command with a null vector for the optional keys argument will cause this.

Example

```
DATAMOVE 0 0 10 0; " "
```

14. EOF ENCOUNTERED INSIDE STRING
compilation

The keyword ENDSUB (which is an End Of File) was found inside a literal character vector. This can be caused by leaving out a quote (").

Example:

```
.
.
.
ARG := EXPINPUT "Enter Layer Number:
.
.
.
ENDSUB
```

15. EXECUTION STACK DESTROYED
runtime, interactive

A certain form of strange syntax will cause this.

Example: ? (INITLIB)

16. EXECUTION STACK OVERFLOW
runtime

If an attempt is made to create more than 122 entries on the execution stack, this error will appear. Local and global variables take one entry each, subroutine calls take two. You may need to declare some variables external, as they do not take up any space on the execution stack.

17. EXPRESSION PARSED INCORRECTLY
compilation

Usually, this means the number of arguments used with a subroutine call is inconsistent with the declaration of that subroutine.

Example:

```
.
.
.
EXTERNAL MONADIC PROCEDURE TEST1

TEST1
.
.
.
```

Also very common, a function may be declared as a procedure.

This message may also appear if the compiler doesn't know what's wrong . . . in essence, an "all out case". See "parse" in the dictionary if necessary.

18. EXTRA ")"
compilation, interactive

A right parenthesis ( )) is found with no left parenthesis (( ) to match it.

Example: (AA)) [2; 2] := BB

This can also be caused by a subscripting error.

Example: (AA [2)

19. FOR VARIABLE MUST BE DECLARED LOCAL
runtime

FOR statements do not work. When they do, local or global variables must be used with them. Currently, if an attempt is made to implement the FOR statement with local or global variables, this error may be the result.

20. FPU OVERFLOW
runtime, interactive

The result of an arithmetic expression is larger than the capability of the Floating Point Unit to represent it.

21. GOTO INTO A STRUCTURED STATEMENT
compilation

This message will appear if a label is found within a structured statement such as an IF/THEM.

Example:

```
.
.
.
IF expression THEN
    .
    .
    .
    START:
    .
    .
    .
ENDIF
.
.
.
GOTO START
.
.
.
```

22. GPLII WORK AREA BIT MAP ALLOCATION ERROR
runtime, interactive

This is a bug. It may be necessary to use ^R as a last result.

23. ILLEGAL ARGUMENT DATA TYPE
runtime

This can be caused by using a variable that has the same name as a GPLII subroutine that is already in the work area.

Example:

```
NILADIC PROCEDURE SETUP

EXTERNAL CALMAMENU

CALMAMENU
.
.
.

? LOAD "CALMAMENU"
? SETUP
```

24. ILLEGAL ARGUMENT FOR CREEL
runtime, interactive

CREEL is a subroutine used in the CREation of an ELement. If an illegal value is submitted to PUTEL (which calls CREEL) this message may be the result.

Example: PUTEL 3; 67; 1; ; ; 5 2 RESHAPE 0 0 2 0 2 2 0 2

In the example above, an attempt was made to create a boundary on layer 67 which is of course absurd.

25. ILLEGAL ARGUMENT SHAPE
runtime, interactive

If the incorrect number of values is used in an argument, this error message will appear.

Example: VIEW 0 0 100 100 2 2

26. ILLEGAL CHARACTER
compilation

A name may have an illegal character in it. Names include; procedures, functions, variables, keywords, and labels. Legal characters for names include: alpha-numerics (A-Z, 0-9) dollar signs ($) and underscores (_).

Example: NILADIC PROCEDURE SETUP@

27. ILLEGAL DATA MODE
runtime, interactive

Characters were used where numerics were expected or vice-versa.
See also: ILLEGAL TYPE CONVERSION

28. ILLEGAL DATA RANK
runtime, interactive

An argument has the wrong number of dimensions, (scalar vs. vector vs. matrix vs. list).

Example: CE 5

Note: Sometimes this message will appear when in fact ILLEGAL DATA SHAPE would have been more appropriate.

29. ILLEGAL DATA SIZE
runtime, interactive

A list subscript is out of range.

Example:
```
? CC := "ABC"; 4 4 RESHAPE 1.6; IOTA 9; -98.6; "F"
? CC[6]
```

30. ILLEGAL DATA VALUE
runtime, interactive

A vector subscript out of range will cause this.

Example: (6330 6331 6332 6339)[5]

31. ILLEGAL ELEMENT FORMAT
runtime, interactive

An attempt to do a PUTEL with an improper element list will cause this message to appear.

Example: PUTEL " "

32. ILLEGAL ELEMENT TYPE
runtime, compilation

Use of PUTEL with an element type that is not in the range 1 <= n <= 7 will cause this.

Example: PUTEL 8 0 -2; 9; 3; " "; "GND"; 18. 4.

33. ILLEGAL EXECUTE REQUEST
runtime, interactive

EXECUTE is used to evalutate a character vector and return the numerical result of that vector. More simply stated, it turns characters into numbers. This error will appear if it is impossible to do so.

Correct Examples:
```
? EXECUTE "1 2 3"
  1. 2. 3.
? CORDS := EXECUTE "GEDCOORDS"
? CALL "REDRAW"
? CALL "A1 := VALUE"
```

Incorrect Examples:
```
? EXECUTE "ABC"
? EXECUTE "REDRAW"
```

34. ILLEGAL FORMAT REQUEST
runtime, interactive

An attempt to FORMAT a list will cause this error. CFORMAT may be used for lists, but FORMAT may not.

Incorrect Example:
```
? FORMAT "ABCDEF"; 1 2 3
```

Correct Example:
```
? CFORMAT "ABCDEF"; 1 2 3
ABCDEF 1. 2. 3.
```

35. ILLEGAL ITEM IN EXPRESSION - SKIPPING
compilation

If the equal sign (=) is omitted from an assignment statement this message will appear.

Example: DD: 7

36. ILLEGAL KIND/KEY/STATUS/PLEX [1]
runtime, interactive

You may have given key numbers to PUTEL instead of the complete element list.

Example:
```
? KEYS := MSELECT . . .
? EL := GETEL KEYS[1]

    . . . change . . .
    . . . modify . . .
    . . . whatever . . .

? PUTEL KEYS
```

Should read:
```
? PUTEL KEYS[1]
```

37. ILLEGAL NUMERIC INPUT
runtime, interactive

An attempt to use IOTA with a non-integer argument will cause this message to appear.

Example:
```
? IOTA 7.5
```

38. ILLEGAL STORE OPERATION
runtime, interactive

This message will result if an expression is found to the left of an assignment statement such as with a double subscript.

Example:
```
(VAR[3] )[2 3] := 7 8
```

A workaround to this is to use an intermediate variable.

Example
```
TEMP := VAR[3]
TEMP[2 3] := 7 8
VAR[3] := TEMP
```

39. ILLEGAL TYPE CONVERSION
runtime, interactive

Character data was used where numeric data was expected or vice-versa. GPLII will convert: logicals --> integers, logicals --> reals, and integers --> reals, automatically where necessary. This is known as "automatic type conversion:. However, not true with: characters --> numerics or numerics --> characters. Use CFORMAT to convert numeric data --> character data and EXECUTE to convert character data --> numeric data.

40. (IMPOSSIBLE) RUNTIME STACK OVERFLOW IN STATEMENT EXPRESSION
compilation

Be sure there is no statement between the SWITCH and the first CASE.

Example:

```
.
.
.
SWITCH
    expression                    <-- not allowed
    CASE " "
    .
    .
    .
    CASE 1; 2;
    .
    .
    .
ENDSWITCH
.
.
.
```

41. INACTIVATED LOCAL VARIABLE
interactive

An attempt to use a local or global variable interactively after a program completes will cause this. Only the values of external variables are retained in the work area. If the variable must be local and there is a need to examine it upon program completion, then one could assign the contents of the local variable to an external variable.

Example:

```
.
.
.
LOCAL EE
EXTERNAL FF
.
.
.
EE := ABS (.5 * SUM)
FF := EE
.
.
.
```

42. INVALID ACCESS KEY
runtime, interactive

This message will result in the event of the use of a non-existent element key with GETEL or PUTEL. Assuming there is no element 9940, consider this example.

Example: ? PUTEL 3 9940 -2; 7; 0; ; ; 0 0 7 0 7 7 0 7 0 0

If the key number in the element list is 0, PUTEL will assign a new key number and create the new element. However, if the key number is anything else, PUTEL looks for that element to delete it so it can create a new one to replace it. This error will appear if there is no element 9940.

Workarounds:

1. Change the key number to 0.
2. Do not delete 9940. PUTEL will delete it and build the new one for you.

43. KEYWORD NOT LEGAL IN THIS CONTEXT
interactive

There are a few select words which may only be used within the context of a GPLII program. Any attempt to use them interactively will result in this error. Some key words are:

EXTERNAL, NILADIC, MONADIC, DYADIC, PROCEDURE, FUNCTION, DO, UNTIL, ENDDO, WHILE, FOR, RANGE, IF, THEN, ELIF, ELSE, ENDIF, SWITCH, OF, CASE, OUT, ENDSWITCH, GOTO, ENDSUB

44. LEFT OPERAND REQUIRED
compilation

An operator requires a left argument.

Example: RESHAPE 7

45. MISSING ENDDO
compilation

The keyword ENDDO is missing from a DO loop.

Example:

```
.
.
.
DO
    .
    .
    .
UNTIL expression
.
.
.
```

See note 2 at the beginning of this list.

46. MISSING ENDIF
compilation

The keyword ENDIF is missing from an IF statement.

Example:

```
.
.
.
IF ANS[I] IN "Nn" THEN
    .
    .
    .
```

See note 2 at the beginning of this list.

47. MUST BE AT LEAST ONE VARIABLE DECLARATION
compilation

Just like the error says, you must declare something, anything, regardless of whether or not you use it. This is a limitation of the compiler.

Example:

```
NILADIC PROCEDURE SIMPLE
LOCAL USELESS

2 + 2

ENDSUB
```

48. NAME IDENTIFIER REQUIRED
":=" REQUIRED -- INSERTED
END OF LINE EXPECTED
compilation

This message will appear if the variable assignment is missing in the program declaration line of a function.

Example: MONADIC FUNCTION MITHRIL LEMBAS

Should read: MONADIC FUNCTION ALE := MITHRIL LEMBAS

49. NILADIC, MONADIC, OR DYADIC REQUIRED
compilation

The very first line in any GPLII program must be the program declaration describing the type of program it is. The only exceptions are comment lines and blank lines. There are six possible configurations for a program declaration line.

Examples:

```
NILADIC PROCEDURE PROGRAM
MONADIC PROCEDURE PROGRAM ARG
DYADIC PROCEDURE ARG1 PROGRAM ARG2
NILADIC FUNCTION RESULT := PROGRAM
MONADIC FUNCTION RESULT := PROGRAM ARG
DYADIC FUNCTION RESULT := ARG1 PROGRAM ARG2
```

50. NO FORMAT OUTPUT FIELD
runtime, interactive

One cause of this error is an attempt to print an integer vector which has a SIZE greater than or equial to 10,000 (SIZE VECT > = 10000). See also: NUMBER OUT OF RANGE

51. NUMBER OUT OF RANGE
runtime, interactive

This message will appear if an attempt is made to evaluate an expression having a greater value than the capacity of GPLII. See also: NO FORMAT OUTPUT FIELD.

52. ONLY ONE UNTIL ALLOWED AT END OF DO
compilation

If an assignment (:=) is used by mistake where a comparison was intended, this error may occur.

Example:

```
.
.
.
N := 0
DO
    N := N + 1
    .
    .
    .
UNTIL N := NUM ENDDO
.
.
.
```

53. OPERAND REQUIRED
interactive

This message indicates that an argument is completely missing.

Example: ? IOTA

54. PROCEDURE DOES NOT END WITH ENDSUB
compilation

The keyword ENDSUB has been omitted from the end of the program. This message will appear even though the program in question may be a function.

If there is an ENDSUB, and this error appears, it may be caused by something else. Correct the other errors first. That may fix the problem.

55. PROCEDURE NOT MONADIC OR DYADIC
execution

An attempt to use an argument with a niladic GPLII program will cause this error.

Example: ? RESTART 7

56. PROCEDURE OR FUNCTION REQUIRED
FATAL COMPILER ERROR -- QUIT
compilation

If in the program declaration line, the word PROCEDURE or FUNCTION is misspelled or omitted, this message will appear.

Example: NILADIC PROCEDURE SETUP

57. PROGRAM INCONSISTENT WITH EXISTING VERSION
load

This can be caused by having a program of a given type in the work area, then modifying it to a new program type, recompiling and attempting to reload.

Example:
- load a niladic procedure into the work area
- modify the program to be a monadic procedure
- recompile the program
- attempt to reload the program

Workarounds:
- rename the program

or

- CLEAR the work area before reloading the new type of program

58. RIGHT OPERAND REQUIRED
compilation, interactive

One cause of this error is to omit the right argument of a monadic or dyadic operator. The compiler will flag this error.

Example:

```
      .
      .
      .
"TWO LINES(CR)OF TEXT" INDEXOF
      .
      .
      .
```

A second cause is to omit the right argument of a dyadic operator when being used interactively.

Example: ? 5 2 RESHAPE

59. SEPARATOR REQUIRED BETWEEN CONSTANTS
compilation

See: DYADIC OPERATOR REQUIRED

60. STRING OVERFLOW
runtime, interactive

An attempt to display an integer vector having a SIZE in the range $6554 <= n <= 9999$ will cause this message to appear.

61. UNMATCHED " ("
compilation, interactive

A left parenthesis (( ) is found with no right parenthesis ( )) to match it.

Example: (HH[7]

62. UNRECOGNIZED ELEMENT TYPE
runtime, compilation

Use of PUTEL with an element type that is not in the range 1 <= n <= 7 will cause this.

Example:

```
PUTEL 8 0 -2; 9; 3; " "; "GND"; 18. 4.
```

63. VARIABLE HAS NOT BEEN INITIALIZED
runtime, interactive

If a variable is declared, but not initialized before it is used, this error message will result.

Example:

```
.
.
.
EXTERNAL HH; II

II := 2 % HH
.
.
.
```

The three steps required to implement a variable are:

1. declare
2. initialize
3. use

Example:

```
.
.
.
EXTERNAL HH; II

HH := 7

II := 2 % HH
.
.
.
```

64. VARIABLE NOT DECLARED
compilation

A variable or subroutine declaration has been omitted. A simple spelling error will cause this.

Example:

```
MONADIC FUNCTION RES := FORMC33 AFARG

EXTERNAL DYADIC FUNCTION MYDFUNC
LOCAL VALUE

VALUE := 0 0 1 MYDFUN 0 0 0
.
.
.
```

65. VARIABLE NOT DEFINED
interactive

An attempt to use a variable that is not in the work area will cause this error. Misspelling a command will also cause this error, therefore making it the very first error message most of us ever saw (and will probably never forget).

Example: ? OPENLOB

66. VIRTUAL POSITION ERROR
runtime, interactive

This is a bug. It may be necessary to use ^R as a last result.

67. WRONG NUMBER OF ARGUMENTS
execution

An attempt to execute a monadic or dyadic program without arguments will cause this message.

APPENDIX D

RELRPIM.PR

```
: VOICE COMMANDS

USER COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE TRAIN
EXTERNAL NILADIC, MONADIC PROCEDURE UPDATE
EXTERNAL NILADIC, MONADIC PROCEDURE UPLOAD
EXTERNAL NILADIC, MONADIC PROCEDURE DOWNLOAD
EXTERNAL NILADIC, MONADIC PROCEDURE SETREJECT
:EXTERNAL NILADIC, MONADIC PROCEDURE RECOGNIZE
EXTERNAL NILADIC PROCEDURE RESET
EXTERNAL NILADIC, MONADIC PROCEDURE VOICE
EXTERNAL NILADIC PROCEDURE ATTENTION
EXTERNAL NILADIC PROCEDURE RELAX

SYSTEM FUNCTIONS

: End of VOICEPRIM.PR
: CARDS COMMANDS

USER COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE TDCHECK

SYSTEM FUNCTIONS

: PHOTO PLOTTER AND NC DRILL COMMANDS
:    PHTOOL IS NOW DEFINED IN GDSIIPRIM.PR BECUASE IT IS ALSO TO BE
:    USED BY CHIPS

EXTERNAL NILADIC, MONADIC PROCEDURE NCTOOL

: NET LIST EXTRACTION COMMANDS

EXTERNAL NILADIC FUNCTION NXSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE NXTOOL
EXTERNAL NILADIC, MONADIC PROCEDURE NXREPORT

: INTERACTIVE ROUTER COMMANDS

EXTERNAL NILADIC PROCEDURE ROUTE

: END OF CARDSPRIM.PR
: CHIPS COMMANDS

USER COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE STRETCHCELL(STCELL)
EXTERNAL NILADIC FUNCTION, NILADIC PROCEDURE SCELL

SYSTEM FUNCTIONS

: END OF CHIPSPRIM.PR

:
: STICKS COMMANDS from STICKSPRIM.PR                    4/28/84
:
USER COMMANDS

EXTERNAL NILADIC, MONADIC FUNCTION STRCLASS
EXTERNAL NILADIC, MONADIC PROCEDURE SRF
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE BINDSRF
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION EXTERIOR
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION DIGASPECT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BSTICKS
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION EXTENSION
EXTERNAL NILADIC, MONADIC PROCEDURE CLONE
EXTERNAL NILADIC, MONADIC PROCEDURE FATWIRE
EXTERNAL NILADIC, MONADIC PROCEDURE PORTWIDTH
EXTERNAL MONADIC PROCEDURE SNAPEND
EXTERNAL NILADIC, MONADIC PROCEDURE FINDINSTANCES
EXTERNAL NILADIC FUNCTION IDCOUNT
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION SHOWSTATUS
EXTERNAL NILADIC FUNCTION BUTTCODE
EXTERNAL NILADIC, MONADIC PROCEDURE STATIONVS
EXTERNAL MONADIC FUNCTION SAVEVS
EXTERNAL MONADIC PROCEDURE RESTOREVS
EXTERNAL NILADIC PROCEDURE NOREDRAW
EXTERNAL MONADIC FUNCTION SCREENCOORD
EXTERNAL NILADIC PROCEDURE KINDS
EXTERNAL MONADIC FUNCTION KINDVS
EXTERNAL MONADIC FUNCTION SCALARINPUT (SCALRINPUT)
EXTERNAL MONADIC FUNCTION LOWESTOF
EXTERNAL MONADIC FUNCTION HIGHESTOF
EXTERNAL NILADIC, MONADIC FUNCTION CECOUNT
EXTERNAL DYADIC FUNCTION SUBSTRING (SBSTRING)
EXTERNAL NILADIC PROCEDURE EXON
EXTERNAL NILADIC PROCEDURE EXOFF
EXTERNAL MONADIC FUNCTION CELLKEYS
EXTERNAL MONADIC FUNCTION CELLKEY (CELKY)
EXTERNAL NILADIC PROCEDURE JOINWIRES
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BORDER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PORT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PORTNODE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION WIRE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FENCE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION HARDWIRE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION HARDFENCE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SYMGRAPH
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION EXTERIORTEXT (EXTEXT)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION INTERIORTEXT (INTEXT)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION USERCONSTRAINT (CONSTRAINT)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION DANGLEMARK
EXTERNAL DYADIC, MONADIC FUNCTION DSRULE
EXTERNAL NILADIC PROCEDURE MATERIALS
EXTERNAL NILADIC PROCEDURE OBJECTS
EXTERNAL NILADIC PROCEDURE SHOWERRORS (SHOERRID)
EXTERNAL NILADIC, MONADIC PROCEDURE DRAWSEGS
EXTERNAL NILADIC, MONADIC PROCEDURE DRAWCROSS
EXTERNAL NILADIC, MONADIC PROCEDURE DRAWARROW
EXTERNAL MONADIC PROCEDURE DRAWBOX
EXTERNAL MONADIC PROCEDURE DRAWTEXT
EXTERNAL MONADIC PROCEDURE DRAWPROMPT (DRWPROMPT)
EXTERNAL MONADIC FUNCTION DYNAMENU
EXTERNAL MONADIC FUNCTION CELLMENU
EXTERNAL MONADIC FUNCTION GETCELLMENU (GTCELLMEN)
EXTERNAL MONADIC PROCEDURE SAVECELLMENU (SAVCELLMEN)
EXTERNAL MONADIC FUNCTION LOCKCELLMENU (LOCKCELMEN)
EXTERNAL MONADIC PROCEDURE FREECELLMENU (FREECELMEN)
EXTERNAL MONADIC PROCEDURE ERASESEGS (ESEGS)
EXTERNAL MONADIC PROCEDURE ERASECROSS (ECROSS)
EXTERNAL MONADIC PROCEDURE ERASETEXT (ETEXT)
```

```
EXTERNAL MONADIC PROCEDURE ERASEPROMPT (EPROMPT)
EXTERNAL MONADIC PROCEDURE ERASEARROW (EARROW)
EXTERNAL MONADIC PROCEDURE ERASEBOX (EBOX)
EXTERNAL NILADIC, MONADIC PROCEDURE AUTOBORDER
EXTERNAL MONADIC FUNCTION EXTENT (XYEXTENT)
EXTERNAL MONADIC FUNCTION ENCIRCLE
EXTERNAL NILADIC, MONADIC PROCEDURE GETBORDER
EXTERNAL NILADIC, MONADIC PROCEDURE IDBORDERS
EXTERNAL NILADIC, MONADIC PROCEDURE SHOWBORDERS
EXTERNAL NILADIC, MONADIC PROCEDURE SHOWPORTS
EXTERNAL NILADIC, MONADIC PROCEDURE WIREPORTS (WTOPPATH)
EXTERNAL NILADIC, MONADIC PROCEDURE WIRENODEPORTS (WTOPNODE)
EXTERNAL NILADIC, MONADIC PROCEDURE COPYPORTS
EXTERNAL NILADIC, MONADIC PROCEDURE IDPORTS
EXTERNAL NILADIC, MONADIC PROCEDURE GETPORT
EXTERNAL MONADIC PROCEDURE NAMEPORTS
EXTERNAL NILADIC, MONADIC PROCEDURE ENDSPACER
EXTERNAL NILADIC, MONADIC PROCEDURE ATTACHWIRES (ATACHWIRES)
EXTERNAL DYADIC FUNCTION FINDCROSS (FNDCROSS)
EXTERNAL NILADIC, MONADIC PROCEDURE NAMECLANS
EXTERNAL NILADIC, MONADIC PROCEDURE SHOWCLANS
:
: PCELL STUFF
:
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BPCELL (PBEPC)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELL (PPCXX)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PCELLANGLE (PPCAN)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLCLEARPARAM (PCLPM)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PCELLDEFPATH (PDTXX)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLENDCAP (PPCEE)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PCELLLENGTH (PPALN)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLNATURAL (PPCNA)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLPLACENAIL (PPLNL)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLREMOVENAIL (PRMNL)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLTEXTPARAM (PTXPM)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PCELLVALID (PVDXX)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLSETX (PSEXS)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLSETY (PSEYS)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLSETW (PSEWD)
EXTERNAL NILADIC PROCEDURE PCELLSTRETCHMARK (PPCSM)
EXTERNAL NILADIC PROCEDURE PCELLVARPOLY (PPCVP)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLWMARK (PWDMR)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLWPARAM (PASWD)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLXMARK (PXSMR)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLXPARAM (PASXS)
EXTERNAL NILADIC, MONADIC PROCEDURE PCELLYMARK (PYSMR)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PCELLYPARAM (PASYS)
EXTERNAL MONADIC PROCEDURE PCELLAUTOSPACE (PAUSP)
EXTERNAL NILADIC PROCEDURE PCELLCOMPRESS (PPMSQ)
:
:SYSTEM FUNCTIONS
:
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STICKMODE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STICKSMODE (STICKMODE)
EXTERNAL NILADIC PROCEDURE SETSTICKS
EXTERNAL NILADIC PROCEDURE SETPCELL
EXTERNAL MONADIC PROCEDURE STICKSEDITOR (STIXEDITOR)
:
: End of STICKSPRIM.PR, nothing beyond this line
:
: GPL II (tm) EXTERNAL DEFINITION FILE
```

```
: GDSIIPRIM.PR DEFINES ALL GDS II SYSTEM PRIMITIVE COMMANDS
: COMMON TO ALL PRODUCTS GDSII, CHIPS, CARDS, AND STICKS.

: SPECIFIC COMMANDS FOR EACH PRODUCT ARE LISTED IN THE
: CORRESPONDING -PRIM.PR FILE NOW

: THE COMMANDS ARE GROUPED INTO TWO CATEGORIES.
: THE FIRST CATEGORY IS CALLED USER COMMANDS.  THESE COMMANDS WOULD
: NORMALLY BE USED BY THE OPERATOR IN AN INTERACTIVE SITUATION.
: THE SECOND CATEGORY IS CALLED SYSTEM COMMANDS.  THESE FUNCTIONS WOULD
: NORMALLY BE USED BY GPL II PROGRAMS
: USER COMMANDS ARE INCLUDED IN THE OUTPUT OF "COMMANDS" AND MAY BE
: EXPECTED TO HAVE HELP FILES.   SYSTEM COMMANDS ARE NOT DOCUMENTED ONLINE.

USER COMMANDS

: STATION CONFIGURATION CONTROL COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION AUTOHOLD (AHOLD)
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION BUTTON
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BUTTONMODE (BUTMODE)
:EXTERNAL NILADIC PROCEDURE BUTTONBEGIN (BTBEGIN)
:EXTERNAL NILADIC PROCEDURE BUTTONEND (BTEND)
EXTERNAL NILADIC, MONADIC PROCEDURE DFUNCTION
EXTERNAL NILADIC PROCEDURE HARDLOCKS
EXTERNAL NILADIC PROCEDURE LOWERCASE
EXTERNAL NILADIC PROCEDURE UPPERCASE
EXTERNAL NILADIC PROCEDURE UPLOWCASE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION RMENU
EXTERNAL NILADIC, MONADIC PROCEDURE WSCREEN
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION RSCREEN
EXTERNAL NILADIC, MONADIC PROCEDURE MENUMOVE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETTAB
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TABMODE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION MENU
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION INTERRUPTMODE (INRUPTS)

: VIEW CONTROL

EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VBTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VBTYPEOFF (VBTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VBTYPEON (VBTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VDTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VDTYPEOFF (VDTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VDTYPEON (VDTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VNTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VNTYPEOFF (VNTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VNTYPEON (VNTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VTTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VTTYPEOFF (VTTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VTTYPEON (VTTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VKIND
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VKINDOFF (VKNDF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VKINDON (VKNDN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VLAYER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VLAYEROFF (VLYRF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VLAYERON (VLYRN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLBY (FBY)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION COLORBY (CBY)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STYLEBY (SBY)
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SOLID
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION DOTTED
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION DASHED
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BROKEN
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETCOLORS
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION RED
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BLUE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION GREEN
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION YELLOW
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION MAGENTA
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION CYAN
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION WHITE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FLAYER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FLAYERON (FLYRN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FLAYEROFF (FLYRF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLA
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLB
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLC
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLD
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FILLSET
EXTERNAL NILADIC, MONADIC PROCEDURE MASKS
EXTERNAL NILADIC PROCEDURE DISPLAYSTAT (DSPSTAT)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VLEVELS
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION GRID
EXTERNAL NILADIC PROCEDURE REMOVE
EXTERNAL NILADIC PROCEDURE REDRAW
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VIEW
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETVIEW
EXTERNAL NILADIC, MONADIC PROCEDURE PAN
EXTERNAL NILADIC, MONADIC PROCEDURE VIEWMOVE (VWMOVE)
EXTERNAL NILADIC, MONADIC PROCEDURE ZOOM
EXTERNAL NILADIC, MONADIC PROCEDURE DATADRAW
EXTERNAL NILADIC, MONADIC PROCEDURE DATAREDRAW (DATADRAW)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION DATAVIEW
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION DATASETVIEW (DTSTVIEW)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAPAN (DTPAN)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAVIEWMOVE (DTVWMOVE)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAZOOM (DTZOOM)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VMODE(VWMODE)

EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION MARKSIZE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SLIMSIZE
EXTERNAL NILADIC PROCEDURE ITEMMARK
EXTERNAL NILADIC PROCEDURE BOUNDARYMARK (BNDMARK)
EXTERNAL NILADIC PROCEDURE PATHMARK
EXTERNAL NILADIC PROCEDURE TEXTMARK
EXTERNAL NILADIC, MONADIC PROCEDURE SREFMARK
EXTERNAL NILADIC, MONADIC PROCEDURE AREFMARK
EXTERNAL NILADIC PROCEDURE TEMPLATEMARK (TMPMARK)
EXTERNAL NILADIC PROCEDURE EXTERIORMARK (EXTMARK)
EXTERNAL MONADIC PROCEDURE KEYMARK
EXTERNAL NILADIC, MONADIC PROCEDURE MARK
EXTERNAL NILADIC, MONADIC PROCEDURE UNMARK
EXTERNAL NILADIC, MONADIC PROCEDURE COORDMARK (COMARK)
EXTERNAL NILADIC, MONADIC PROCEDURE RULER
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION VECTORCOUNT (VECCT)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION CIRCLESIZE (CIRCSIZE)

!EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SPLITVIEW
!EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETLVIEW (LV)
!EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETRVIEW (RV)
```

```
:EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VPAN
:EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VZOOM
:EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION VTRACK

: SCALE COMMANDS

EXTERNAL NILADIC PROCEDURE SSTAT
EXTERNAL NILADIC PROCEDURE ASCALE
EXTERNAL NILADIC PROCEDURE BSCALE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION GSCALE
EXTERNAL NILADIC, MONADIC PROCEDURE RSCALE
EXTERNAL NILADIC, MONADIC PROCEDURE WSCALE
EXTERNAL NILADIC PROCEDURE XSCALE
EXTERNAL NILADIC PROCEDURE YSCALE
EXTERNAL NILADIC PROCEDURE ZSCALE

: LIBRARY ACCESS COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE RECOVERLIB
EXTERNAL NILADIC, MONADIC PROCEDURE RCSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION INITLIB
EXTERNAL NILADIC, MONADIC PROCEDURE NEWUNITS
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION INFORM
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OUTFORM
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STREAMOUT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OPENLIB (OLIB)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OPENREF1 (OPEN1)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OPENREF2 (OPEN2)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TEMPREF1 (TREF1)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TEMPREF2 (TREF2)
EXTERNAL NILADIC, MONADIC PROCEDURE CLOSEREF1 (CLOS1)
EXTERNAL NILADIC, MONADIC PROCEDURE CLOSEREF2 (CLOS2)
EXTERNAL NILADIC, MONADIC PROCEDURE LISTREF1 (REF1LIB)
EXTERNAL NILADIC, MONADIC PROCEDURE LISTREF2 (REF2LIB)
EXTERNAL NILADIC, MONADIC PROCEDURE LISTLIB
EXTERNAL NILADIC, MONADIC PROCEDURE LISTGEN
EXTERNAL NILADIC, MONADIC PROCEDURE MERGELIB
EXTERNAL NILADIC PROCEDURE CLOSELIB (CLLIB)
EXTERNAL NILADIC, MONADIC PROCEDURE LSTAT
EXTERNAL NILADIC, MONADIC PROCEDURE SUMMARY
EXTERNAL NILADIC PROCEDURE TREE
EXTERNAL NILADIC, MONADIC PROCEDURE SHOWREFS
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION CONTEXT
EXTERNAL NILADIC, MONADIC PROCEDURE FINDDATA
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE CSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION DSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION ESTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE BACKUP
EXTERNAL NILADIC, MONADIC PROCEDURE EXPUNGE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE REVERT
EXTERNAL NILADIC, MONADIC PROCEDURE RSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE SSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE TSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE VSTRUCT
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION EDATA
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION ODATA
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION LEVEL
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SETLEVEL
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION EREF
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION OREF
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION ONAME
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION ENAME
EXTERNAL NILADIC PROCEDURE SAVEDISPLAY
EXTERNAL NILADIC, MONADIC PROCEDURE OPENDISPLAY
EXTERNAL NILADIC, MONADIC PROCEDURE INITDISPLAY
EXTERNAL NILADIC, MONADIC PROCEDURE LISTDISPLAY
EXTERNAL NILADIC PROCEDURE UPDISPLAY
EXTERNAL NILADIC, MONADIC PROCEDURE USTRUCT

: DATA ENTRY COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE RT
EXTERNAL NILADIC, MONADIC PROCEDURE R3
EXTERNAL NILADIC, MONADIC PROCEDURE SEGS

: DATA MOVEMENT COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SBTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SBTYPEOFF (SBTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SBTYPEON (SBTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SDTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SDTYPEOFF (SDTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SDTYPEON (SDTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SNTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SNTYPEOFF (SNTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SNTYPEON (SNTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STTYPEOFF (STTPF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION STTYPEON (STTPN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SKIND
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SKINDOFF (SKNDF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SKINDON (SKNDN)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SLAYER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SLAYEROFF (SLYRF)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SLAYERON (SLYRN)
EXTERNAL NILADIC, MONADIC PROCEDURE DATACOPY (DTCPY)
EXTERNAL NILADIC, MONADIC PROCEDURE DATADELETE (DTDEL)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAMAGNIFY (DTMAG)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAMOVE (DTMOV)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAOVERSIZE (DTOVR)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAROTATE (DTROT)
EXTERNAL NILADIC, MONADIC PROCEDURE DATAREFLECT (DTREF)
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION DATATYPE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION PATHTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWTEXTTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWDATATYPE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWNODETYPE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWBOXTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWLAYER
EXTERNAL NILADIC, MONADIC PROCEDURE NEWANGLE
EXTERNAL NILADIC, MONADIC PROCEDURE NEWMAG
EXTERNAL NILADIC, MONADIC PROCEDURE NEWWIDTH
EXTERNAL NILADIC, MONADIC PROCEDURE NEWFONT
EXTERNAL NILADIC, MONADIC PROCEDURE NEWPATHTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE EXPLODE

: IDENTIFIED GROUP COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE IDEXPLODE
EXTERNAL NILADIC PROCEDURE IDCLEAR
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE IDCOPY
EXTERNAL NILADIC, MONADIC PROCEDURE IDDELETE
EXTERNAL NILADIC, MONADIC PROCEDURE ID
EXTERNAL NILADIC, MONADIC PROCEDURE IDADD
EXTERNAL NILADIC, MONADIC PROCEDURE UNID
EXTERNAL NILADIC, MONADIC PROCEDURE IDCLAYER
EXTERNAL NILADIC, MONADIC PROCEDURE IDCDATATYPE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCNODETYPE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCBOXTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCTEXTTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCPATHTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCMAG
EXTERNAL NILADIC, MONADIC PROCEDURE IDCANGLE
EXTERNAL NILADIC, MONADIC PROCEDURE IDCWIDTH
EXTERNAL NILADIC, MONADIC PROCEDURE IDCFONT
EXTERNAL NILADIC PROCEDURE IDMARK
EXTERNAL NILADIC, MONADIC PROCEDURE IDREFLECT
EXTERNAL NILADIC, MONADIC PROCEDURE IDROTATE
EXTERNAL NILADIC, MONADIC PROCEDURE IDMOVE
EXTERNAL NILADIC, MONADIC PROCEDURE IDOVERSIZE
EXTERNAL NILADIC PROCEDURE IDPATHEXP
EXTERNAL NILADIC, MONADIC PROCEDURE IDWINDOW
EXTERNAL NILADIC, MONADIC PROCEDURE UNIDWINDOW (UNWIN)
EXTERNAL NILADIC, MONADIC PROCEDURE WINCOPY (WNCPY)
EXTERNAL NILADIC, MONADIC PROCEDURE WINFENCE
EXTERNAL NILADIC, MONADIC PROCEDURE WINDELETE (WNDEL)
EXTERNAL NILADIC, MONADIC PROCEDURE WINMOVE (WNMOV)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION WINOPTIONS
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION OPTIONS (WINOPTIONS)
EXTERNAL NILADIC, MONADIC PROCEDURE WINSTRETCH (WNSTR)

: GRAPHIC EDITOR

EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION PATHEDGE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION GEDMODE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION DEFMODE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION INTCHECKMODE
EXTERNAL NILADIC PROCEDURE SETDEFAULTS
EXTERNAL NILADIC PROCEDURE SHOWDEFAULTS
EXTERNAL NILADIC, MONADIC PROCEDURE ITEMROTATE
EXTERNAL NILADIC, MONADIC PROCEDURE ITEMREFLECT (IREFLECT)
EXTERNAL NILADIC, MONADIC PROCEDURE GETITEM
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ISTAT
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ITEM
EXTERNAL NILADIC, MONADIC PROCEDURE ITEMCOPY
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ABSANGLE
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ABSMAG
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ABSWIDTH
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ANGLE
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION AREF
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION ASHAPE
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION BOUNDARY
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION BOX
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BOXTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE CE
EXTERNAL NILADIC, MONADIC PROCEDURE, MONADIC FUNCTION COORDS
EXTERNAL NILADIC PROCEDURE EDGE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION ENTERTEXT (ENTEXT)
EXTERNAL NILADIC, MONADIC PROCEDURE REDO
EXTERNAL NILADIC, MONADIC PROCEDURE DVERT
EXTERNAL NILADIC PROCEDURE CLOSE (GCLOSE)
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE GET
EXTERNAL NILADIC, MONADIC PROCEDURE GETP
EXTERNAL NILADIC PROCEDURE UNGET
EXTERNAL NILADIC, MONADIC PROCEDURE MOVEPOINT
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION NODE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION NODETYPE
EXTERNAL NILADIC PROCEDURE OCTAGONAL
EXTERNAL NILADIC PROCEDURE ORTHINT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TEMPLATE(TMPLATE)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PLEX
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PLEXMODE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SNAPMODE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION SNAPSIZE
EXTERNAL NILADIC PROCEDURE STRAIGHT
EXTERNAL NILADIC PROCEDURE HORIZFIRST
EXTERNAL NILADIC PROCEDURE VERTFIRST
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION RTDIGMODE(RTDMODE)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION CURMODE

EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION FONT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BINDFONTS
EXTERNAL NILADIC, MONADIC PROCEDURE FONTINFO
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION TEXT
EXTERNAL NILADIC, MONADIC PROCEDURE TEXTFONT (FONTC)
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION TEXTTYPE
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TJUST

EXTERNAL NILADIC, MONADIC PROCEDURE SUBRT
EXTERNAL NILADIC, MONADIC PROCEDURE ADDRT
EXTERNAL NILADIC, MONADIC PROCEDURE INTRT
EXTERNAL NILADIC, MONADIC PROCEDURE CUTPATHIN (TPAI)
EXTERNAL NILADIC, MONADIC PROCEDURE CUTPATHOUT (TPAO)
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION MAG
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION LAYER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PUT
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION PUTALL
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PATH
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION WIDTH
EXTERNAL NILADIC, MONADIC FUNCTION SNAP
EXTERNAL NILADIC, MONADIC PROCEDURE UNDO
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION SNAME
EXTERNAL MONADIC PROCEDURE, NILADIC FUNCTION REFL
EXTERNAL NILADIC PROCEDURE ITEMREVERSE (RVRSE)
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION SREF
EXTERNAL MONADIC,  NILADIC PROCEDURE STRETCH (ST)
EXTERNAL NILADIC, MONADIC PROCEDURE MOVE

EXTERNAL NILADIC,MONADIC PROCEDURE WIPE
EXTERNAL NILADIC,MONADIC PROCEDURE WIPEALL
EXTERNAL NILADIC FUNCTION LASTCOORD (LASTC)
EXTERNAL NILADIC, MONADIC PROCEDURE ARC
EXTERNAL NILADIC, MONADIC PROCEDURE CIRCLE
EXTERNAL NILADIC, MONADIC PROCEDURE FILLET

: PROPERTY LISTS

EXTERNAL NILADIC, MONADIC PROCEDURE DEFATTR
EXTERNAL NILADIC, MONADIC PROCEDURE LISTATTR
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION BINDATTR
EXTERNAL NILADIC, MONADIC PROCEDURE USERPROP(PROPERTY)
EXTERNAL NILADIC PROCEDURE CLRPROP
```

```
EXTERNAL NILADIC, MONADIC FUNCTION PROPVALUE

: TEXT EDITOR

EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION ESTAT
EXTERNAL NILADIC, MONADIC PROCEDURE A
EXTERNAL NILADIC PROCEDURE B
EXTERNAL NILADIC, MONADIC PROCEDURE CH
EXTERNAL NILADIC, MONADIC PROCEDURE DB
EXTERNAL NILADIC, MONADIC PROCEDURE DE
EXTERNAL NILADIC, MONADIC PROCEDURE DLN
EXTERNAL NILADIC, MONADIC PROCEDURE DS
EXTERNAL NILADIC, MONADIC PROCEDURE G
EXTERNAL NILADIC, MONADIC PROCEDURE I
EXTERNAL NILADIC, MONADIC PROCEDURE IB
EXTERNAL NILADIC, MONADIC PROCEDURE ICOPY
EXTERNAL NILADIC, MONADIC PROCEDURE IE
EXTERNAL NILADIC, MONADIC PROCEDURE IFILE
EXTERNAL NILADIC, MONADIC PROCEDURE IMOVE
EXTERNAL NILADIC, MONADIC PROCEDURE J
EXTERNAL NILADIC PROCEDURE JOIN
EXTERNAL NILADIC, MONADIC PROCEDURE K
EXTERNAL NILADIC, MONADIC PROCEDURE L
EXTERNAL NILADIC, MONADIC PROCEDURE SPLIT (LSPLIT)
EXTERNAL NILADIC, MONADIC PROCEDURE S
EXTERNAL NILADIC, MONADIC PROCEDURE SB
EXTERNAL NILADIC, MONADIC PROCEDURE T
EXTERNAL NILADIC, MONADIC PROCEDURE W
EXTERNAL NILADIC PROCEDURE Z
EXTERNAL NILADIC PROCEDURE IITEM
EXTERNAL NILADIC PROCEDURE WITEM

: BACKGROUND JOB CONTROL

EXTERNAL NILADIC, MONADIC PROCEDURE JOBABORT (ABORT)
EXTERNAL MONADIC FUNCTION JOBENTER
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION JOBCREATE
: EXTERNAL MONADIC PROCEDURE JOBDEQUEUE
: EXTERNAL MONADIC PROCEDURE JOBENQUEUE
EXTERNAL NILADIC, MONADIC PROCEDURE JOBINFO
EXTERNAL NILADIC, MONADIC PROCEDURE JOBLOG
EXTERNAL NILADIC, MONADIC PROCEDURE JOBPRIORITY (PRIO)
EXTERNAL NILADIC, MONADIC PROCEDURE JOBSTART (RESTART)
EXTERNAL NILADIC, MONADIC PROCEDURE JOBSUSPEND (SUSPEND)
EXTERNAL NILADIC, MONADIC PROCEDURE PAUSE
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION PLOTSTAT
EXTERNAL NILADIC, MONADIC PROCEDURE RS

: SYSTEM UTILITIES

EXTERNAL NILADIC PROCEDURE SYSTAT
EXTERNAL NILADIC PROCEDURE SYSDPY
EXTERNAL NILADIC PROCEDURE STADPY (CONDPY)
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION MEMORY
EXTERNAL NILADIC, MONADIC PROCEDURE COMMANDS
EXTERNAL NILADIC, MONADIC PROCEDURE FUNCTIONS
EXTERNAL NILADIC, MONADIC PROCEDURE TYPE
EXTERNAL NILADIC, MONADIC PROCEDURE HELP
EXTERNAL NILADIC PROCEDURE DIRECTORY
EXTERNAL NILADIC PROCEDURE, NILADIC FUNCTION DISK
EXTERNAL NILADIC, MONADIC PROCEDURE FPRINT
```

```
EXTERNAL NILADIC, MONADIC PROCEDURE FLIST
EXTERNAL NILADIC, MONADIC PROCEDURE FDELETE
EXTERNAL NILADIC, MONADIC PROCEDURE FCHATR
EXTERNAL NILADIC, MONADIC PROCEDURE FCOPY
EXTERNAL NILADIC, MONADIC PROCEDURE FINIT
EXTERNAL NILADIC, MONADIC PROCEDURE FRELEASE
EXTERNAL NILADIC, MONADIC PROCEDURE FLINK
EXTERNAL NILADIC, MONADIC PROCEDURE FUNLINK
EXTERNAL NILADIC, MONADIC PROCEDURE FRENAME
EXTERNAL NILADIC, MONADIC PROCEDURE FCCONT
EXTERNAL NILADIC, MONADIC PROCEDURE START
EXTERNAL NILADIC PROCEDURE QUIT
EXTERNAL NILADIC, MONADIC PROCEDURE ABSTATION
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION LOG
EXTERNAL NILADIC PROCEDURE ENDLOG
EXTERNAL NILADIC, MONADIC PROCEDURE UPLIB
EXTERNAL NILADIC, MONADIC PROCEDURE UPFONT
EXTERNAL NILADIC PROCEDURE SPOOLQ
EXTERNAL NILADIC, MONADIC PROCEDURE, NILADIC FUNCTION TIME
EXTERNAL MONADIC PROCEDURE SLEEP
EXTERNAL NILADIC, MONADIC PROCEDURE HARDCOPY

: CalmaNet commands

EXTERNAL NILADIC, MONADIC PROCEDURE SETMEMORY
EXTERNAL NILADIC PROCEDURE GETMEMORY
EXTERNAL NILADIC, MONADIC PROCEDURE NFSTAT
EXTERNAL NILADIC PROCEDURE NFCONT
EXTERNAL NILADIC, MONADIC PROCEDURE NFMOVE
EXTERNAL NILADIC PROCEDURE SYSID
EXTERNAL NILADIC PROCEDURE AKNUM

: GPL COMMANDS

EXTERNAL NILADIC, MONADIC PROCEDURE GPL
EXTERNAL NILADIC, MONADIC PROCEDURE LOAD (LOADOBJ)
EXTERNAL NILADIC PROCEDURE NAMES
EXTERNAL NILADIC FUNCTION SYSN
EXTERNAL NILADIC PROCEDURE VARS
EXTERNAL NILADIC PROCEDURE SUBS
EXTERNAL NILADIC PROCEDURE WASTAT
EXTERNAL NILADIC PROCEDURE CLEAR (GXCLEAR)
EXTERNAL NILADIC, MONADIC PROCEDURE BREAKPOINT (BPOINT)
EXTERNAL NILADIC PROCEDURE PROCEED (PCEED)
EXTERNAL NILADIC PROCEDURE STEP (PSTEP)
EXTERNAL MONADIC PROCEDURE TOLINE

SYSTEM FUNCTIONS

: GPL INPUT OPERATORS

EXTERNAL NILADIC, MONADIC FUNCTION EXPINPUT
EXTERNAL NILADIC, MONADIC FUNCTION TEXTINPUT
EXTERNAL NILADIC, MONADIC FUNCTION DIGIN

: GPL COMPILER AND OUTPUT FORMATTER INTERFACE

EXTERNAL MONADIC PROCEDURE CALL
EXTERNAL MONADIC FUNCTION TRAPCALL (CALL)
EXTERNAL MONADIC FUNCTION TRAP CALL (CALL)
EXTERNAL MONADIC FUNCTION EXECUTE
```

```
EXTERNAL MONADIC FUNCTION FORMAT(GOUT)
EXTERNAL MONADIC FUNCTION CFORMAT(FFORMAT)

: GPL OPERATORS PENDING DEFINITION IN KEYWORDS

EXTERNAL DYADIC FUNCTION BITOR
EXTERNAL DYADIC FUNCTION BITAND
EXTERNAL DYADIC FUNCTION BITXOR
EXTERNAL DYADIC FUNCTION UNION
EXTERNAL DYADIC FUNCTION INTERSECT (INTERSECTION)
EXTERNAL DYADIC FUNCTION MEMBER
EXTERNAL DYADIC FUNCTION SUBSET
EXTERNAL DYADIC FUNCTION MINUS (DIFFERENCE)
EXTERNAL MONADIC FUNCTION ARCSIN (ARSIN)
EXTERNAL MONADIC FUNCTION ARCCOS (ARCOS)
EXTERNAL MONADIC FUNCTION POLAR
EXTERNAL MONADIC FUNCTION CARTESIAN
EXTERNAL NILADIC FUNCTION ENCLOSE
EXTERNAL NILADIC FUNCTION COORDFILE (ENCLOSE)
EXTERNAL MONADIC FUNCTION INEXCLUDE
EXTERNAL DYADIC FUNCTION DROUND (DYROUND)
EXTERNAL DYADIC FUNCTION CLOSEST TO (SHDISTANCE)
EXTERNAL DYADIC FUNCTION CLOSESTTO (SHDISTANCE)
EXTERNAL MONADIC FUNCTION MIN MAX (MINMAX)
EXTERNAL MONADIC FUNCTION MINMAX
EXTERNAL MONADIC FUNCTION LINE INTERSECT (LSINTERSECT)
EXTERNAL MONADIC FUNCTION LINEINTERSECT (LSINTERSECT)
EXTERNAL DYADIC FUNCTION INDICES OF (INDICES)
EXTERNAL DYADIC FUNCTION INDICESOF (INDICES)
EXTERNAL DYADIC FUNCTION INSIDE POLYGON (INSPOLY)
EXTERNAL DYADIC FUNCTION INSIDEBOUNDARY (INSPOLY)
EXTERNAL MONADIC FUNCTION POLYGON AREA (POLYAREA)
EXTERNAL MONADIC FUNCTION BOUNDARYAREA (POLYAREA)
EXTERNAL NILADIC, MONADIC FUNCTION ROUND
EXTERNAL NILADIC, MONADIC PROCEDURE ERROR (EUSER)

: GPL/CDOS INTERFACE

EXTERNAL NILADIC, MONADIC FUNCTION FILEINFO
EXTERNAL DYADIC, MONADIC FUNCTION FETCH
EXTERNAL DYADIC PROCEDURE SAVE
EXTERNAL NILADIC FUNCTION GTIME (GXTIME)
EXTERNAL NILADIC FUNCTION DAY (GXDAY)
EXTERNAL NILADIC FUNCTION TOD (GXTOD)
EXTERNAL NILADIC FUNCTION CRTTYPE
EXTERNAL NILADIC FUNCTION TABTYPE
EXTERNAL NILADIC FUNCTION STANUM

: GPL -- LIBRARY ACCESS FUNCTIONS

EXTERNAL NILADIC FUNCTION NAMESOPEN (NMOPEN)
EXTERNAL NILADIC, MONADIC FUNCTION NAMESBOUND (NMBOUND)
EXTERNAL NILADIC, MONADIC FUNCTION STRUCINFO (STRUINFO)
EXTERNAL NILADIC, MONADIC FUNCTION STRUCLIST (STRULIST)
EXTERNAL NILADIC, MONADIC FUNCTION STADPYINFO
EXTERNAL NILADIC FUNCTION STRUCOPEN (STRUOPEN)
: EXTERNAL NILADIC, MONADIC FUNCTION FONTINFO

: GPL/DATA BASE INTERFACE

EXTERNAL NILADIC, MONADIC FUNCTION KEYCE
```

```
EXTERNAL NILADIC, MONADIC FUNCTION CEKEY
EXTERNAL MONADIC FUNCTION LOCALTOGLOBAL
EXTERNAL MONADIC FUNCTION GLOBALTOLOCAL
EXTERNAL NILADIC, MONADIC FUNCTION DATAEXTENT (GDEXTENT)
EXTERNAL NILADIC FUNCTION VIEWEXTENT (GDVWEXT)
EXTERNAL NILADIC FUNCTION VIEWWINDOW (GDVWWIN)
EXTERNAL NILADIC FUNCTION IDKEYS (GDIDKEYS)
EXTERNAL NILADIC FUNCTION NSELECT (GDSELECT)
EXTERNAL MONADIC FUNCTION MSELECT (GDSELECT)
EXTERNAL DYADIC FUNCTION DSELECT (GDSELECT)
EXTERNAL MONADIC FUNCTION USELECT
EXTERNAL MONADIC FUNCTION GETEL (GDGETEL)
EXTERNAL MONADIC PROCEDURE PUTEL (GDPUTEL)
EXTERNAL MONADIC FUNCTION PATHBOUNDARY (PATHBNDARY)

: GPL/JOB SYSTEM INTERFACE

EXTERNAL NILADIC FUNCTION JOBID
: EXTERNAL NILADIC FUNCTION JOBQUEUE

: GPL/JOB PARAMETER FILE FILE INTERFACE

EXTERNAL MONADIC FUNCTION JOBREAD
EXTERNAL MONADIC PROCEDURE JOBWRITE
EXTERNAL MONADIC FUNCTION JOBRSTR
EXTERNAL MONADIC PROCEDURE JOBWSTR
EXTERNAL MONADIC FUNCTION GETREAL4 (GETR4)
EXTERNAL MONADIC FUNCTION GETREAL2 (GETR2)
EXTERNAL MONADIC FUNCTION GETINT2
EXTERNAL MONADIC FUNCTION PUTREAL4 (PUTR4)
EXTERNAL MONADIC FUNCTION PUTREAL2 (PUTR2)
EXTERNAL MONADIC FUNCTION PUTINT2

: GPL BACKGROUND EXPLODER COMMANDS.

EXTERNAL NILADIC PROCEDURE RTEXP
EXTERNAL NILADIC, MONADIC PROCEDURE BGEXP
EXTERNAL NILADIC, MONADIC PROCEDURE EXPTRACE
EXTERNAL NILADIC FUNCTION READVIEW (RDVIEW)
EXTERNAL MONADIC PROCEDURE CENTERLINE (CTLINE)
EXTERNAL MONADIC PROCEDURE VIEWOVERSIZE (EOCOMMAIN)
EXTERNAL NILADIC, MONADIC PROCEDURE OUTPUTVIEW(JIDRCVIEW)
EXTERNAL NILADIC, MONADIC PROCEDURE EKIND
EXTERNAL NILADIC, MONADIC PROCEDURE EKINDON (EKNDN)
EXTERNAL NILADIC, MONADIC PROCEDURE EKINDOFF (EKNDF)
EXTERNAL NILADIC, MONADIC PROCEDURE ELAYER
EXTERNAL NILADIC, MONADIC PROCEDURE ELAYERON (ELYRN)
EXTERNAL NILADIC, MONADIC PROCEDURE ELAYEROFF (ELYRF)

: GPL DRC INTERFACE

EXTERNAL NILADIC, MONADIC PROCEDURE DRCVERBOSE
EXTERNAL NILADIC PROCEDURE DRCTIMER
EXTERNAL NILADIC PROCEDURE VDRCINIT(VDRCINIT)
EXTERNAL NILADIC PROCEDURE VDRCSHUTDOWN(VDRCSHUTDOWN)
EXTERNAL NILADIC PROCEDURE VDRCRESTART(VDRCRESTART)
EXTERNAL MONADIC FUNCTION INPUTMASK(INMASK)
EXTERNAL MONADIC PROCEDURE OUTPUTMASK(OUTMASK)
EXTERNAL MONADIC PROCEDURE OUTPUTPATH(OUTPATH)
EXTERNAL MONADIC PROCEDURE CHECKEXTERIOR1(CHEX1)
EXTERNAL MONADIC PROCEDURE CHECKEXTERIOR2(CHEX2)
```

```
EXTERNAL MONADIC PROCEDURE CHECKINTERIOR(CHIN)
EXTERNAL MONADIC FUNCTION OVERSIZE(OVSZ)
EXTERNAL MONADIC FUNCTION UNDERSIZE(UNSZ)
EXTERNAL MONADIC FUNCTION CHECKAREA(CHAREA)
EXTERNAL MONADIC FUNCTION ROUNDMASK(RNDMASK)
EXTERNAL DYADIC FUNCTION MOR
EXTERNAL DYADIC FUNCTION MAND
EXTERNAL DYADIC FUNCTION MANDNOT
EXTERNAL DYADIC FUNCTION MXOR
EXTERNAL DYADIC FUNCTION ANDNOT(MANDNOT)
EXTERNAL MONADIC PROCEDURE OUTPUTCHANGE(JICHANGE)
EXTERNAL MONADIC PROCEDURE MASKFREE
EXTERNAL MONADIC PROCEDURE MASKSAVE(MASK2SAVE)
EXTERNAL MONADIC PROCEDURE VMASKSAVE(VX2MASKSAVE)
EXTERNAL MONADIC FUNCTION MASKRESTORE
EXTERNAL MONADIC FUNCTION VMASKRESTORE(VXMASKRESTORE)

: GPL/GED INTERFACE

EXTERNAL NILADIC FUNCTION GEDTYPE (GGKIND)
EXTERNAL NILADIC FUNCTION GEDKEY (GGKEY)
EXTERNAL NILADIC FUNCTION GEDPLEX (GGPLEX)
: EXTERNAL NILADIC FUNCTION GEDLINKS (GGLINKS)
EXTERNAL NILADIC FUNCTION GEDCOORDS (GGCOORDS)
EXTERNAL NILADIC FUNCTION GEDLAYER (GGLAYER)
EXTERNAL NILADIC FUNCTION GEDPATHTYPE (GGPATHTYPE)
EXTERNAL NILADIC FUNCTION GEDDATATYPE (GGDTYPE)
EXTERNAL NILADIC FUNCTION GEDTEXTTYPE (GGTTYPE)
EXTERNAL NILADIC FUNCTION GEDWIDTH (GGWIDTH)
EXTERNAL NILADIC FUNCTION GEDNAME (GGNAME)
EXTERNAL NILADIC FUNCTION GEDTRANSFORM (GGTRANSFORM)
EXTERNAL NILADIC FUNCTION GEDTEXT (GGTEXT)
EXTERNAL NILADIC FUNCTION GEDFONT (GGFONT)
EXTERNAL NILADIC FUNCTION GEDASHAPE (GGASHAPE)
EXTERNAL NILADIC FUNCTION GEDINTEGER (GGINTEGER)
EXTERNAL NILADIC FUNCTION GEDSTRING (GGSTRING)
EXTERNAL NILADIC FUNCTION GEDELEMENT (GGELEMENT)
EXTERNAL NILADIC FUNCTION GEDDIGMODE (GGDMODE)
EXTERNAL MONADIC PROCEDURE DIGMODE (GSDMODE)

: GPL Drawing Routine

EXTERNAL NILADIC, MONADIC PROCEDURE GPLDRAW   .

: ALL COMMANDS AFTER THIS POINT ARE EXCLUDED FROM THE "COMMANDS" LIST

EXTERNAL NILADIC, MONADIC PROCEDURE IDBADPLEX
EXTERNAL NILADIC PROCEDURE BUGCLASS
EXTERNAL NILADIC PROCEDURE ZTOOL
EXTERNAL NILADIC, MONADIC PROCEDURE PROPERTY
EXTERNAL NILADIC, MONADIC FUNCTION FILETEST
EXTERNAL NILADIC, MONADIC PROCEDURE NAMTABLE
EXTERNAL NILADIC, MONADIC PROCEDURE OVID
EXTERNAL NILADIC PROCEDURE DEBUG
EXTERNAL NILADIC PROCEDURE OCTUMD
EXTERNAL NILADIC PROCEDURE HEXUMD
EXTERNAL NILADIC, MONADIC PROCEDURE LOOKAHEAD
EXTERNAL NILADIC, MONADIC PROCEDURE VLT
EXTERNAL NILADIC,MONADIC FUNCTION LIBUNITS
EXTERNAL NILADIC,MONADIC FUNCTION LIBFONT
EXTERNAL NILADIC, MONADIC PROCEDURE CREATE(CRFILE)
```

```
:EXTERNAL NILADIC, MONADIC PROCEDURE SKETCH
:EXTERNAL NILADIC PROCEDURE CREF
EXTERNAL MONADIC FUNCTION SQUAREMASK(SQRMASK)
EXTERNAL NILADIC PROCEDURE TIMETEST

: INCLUDING PHTOOL INTO GDSIIPRIM.PR BECUASE IT IS BEING USED BY CHIPS
:     CUSTOMERS ALSO.
EXTERNAL NILADIC, MONADIC FUNCTION PHTOOL


: END OF GDSIIPRIM.PR
```

APPENDIX E

BIBLIOGRAPHY

BIBLIOGRAPHY

Peter Erdman, Sperry Corporation
GDSII Help Files
Ben Glick, Calma Marketing
Ted Paone, Calma Applications Engineer
Gretchen Wegman, Calma Technical Instructor/Developer

INDEX

INDEX