L1 MOS

PGU
Graphics Management Package
Programmer Guide

olivetti

PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, via Jervis - 10015 Ivrea (Italy)

PREFACE

This manual describes the PGU COBOL, Compiled BASIC, PASCAL+ and FORTRAN interface which are available on the OLIVETTI L1 systems. It is intended for programmers who wish to write programs which incorporate graphics.

SUMMARY

Chapter 1 of the manual explains the available graphics features and the terminology.

Chapters 2, 3 and 4 contain a detailed description of the PGU COBOL, Compiled BASIC, and PASCAL+ interfaces, including all the data formats, respectively.

Chapter 5 contains a detailed description of all the PGU COBOL, Compiled BASIC and PASCAL+ interface functions, in alphabetical order.

Chapter 6 contains a detailed description of all the PGU FORTRAN interface functions, in alphabetical order.

Appendix A describes the CHANGEFONT utility and Appendix B contains a list of the system reply codes.

REFERENCES

Read first...

Introduction to MOS - Code 4002130 G (Vol. 2)

For further information read...

FORTRAN Language - Reference Manual - Code 4004490 M (Vol. 6C)

Compiled BASIC Language - Reference Manual - Code 4004350 X (Vol. 6E)

COBOL Language - Reference Manual - Code 4004290 H (Vol. 6D)

PASCAL+ Language - Reference Manual - Code 4002460 R (Vol. 9A)

FORTRAN - Program Preparation and Execution - Code 4004510 F (Vol. 6C)

Compiled BASIC - Program Preparation and Execution - Code 4002180 M (Vol. 6E)

COBOL - Program Preparation and Execution - Code 4004310 T (Vol. 6D)

PASCAL+ - Program Preparation and Execution - Code 4002480 T (Vol.9A)

RS232/CL Interface - Programmer Guide - Code 4004450 H (Vol. 6G)

Program Development tools - Reference Manual - Code 4002790 S (Vol. 6F)

MOS - Programmer Guide - Code 4002570 L (Vol. 6G)

MOS - Operating Guide - Code 4036160 T (Vol. 3)

Glossary/Glossario - Code 4002140 H (Vol. 1)

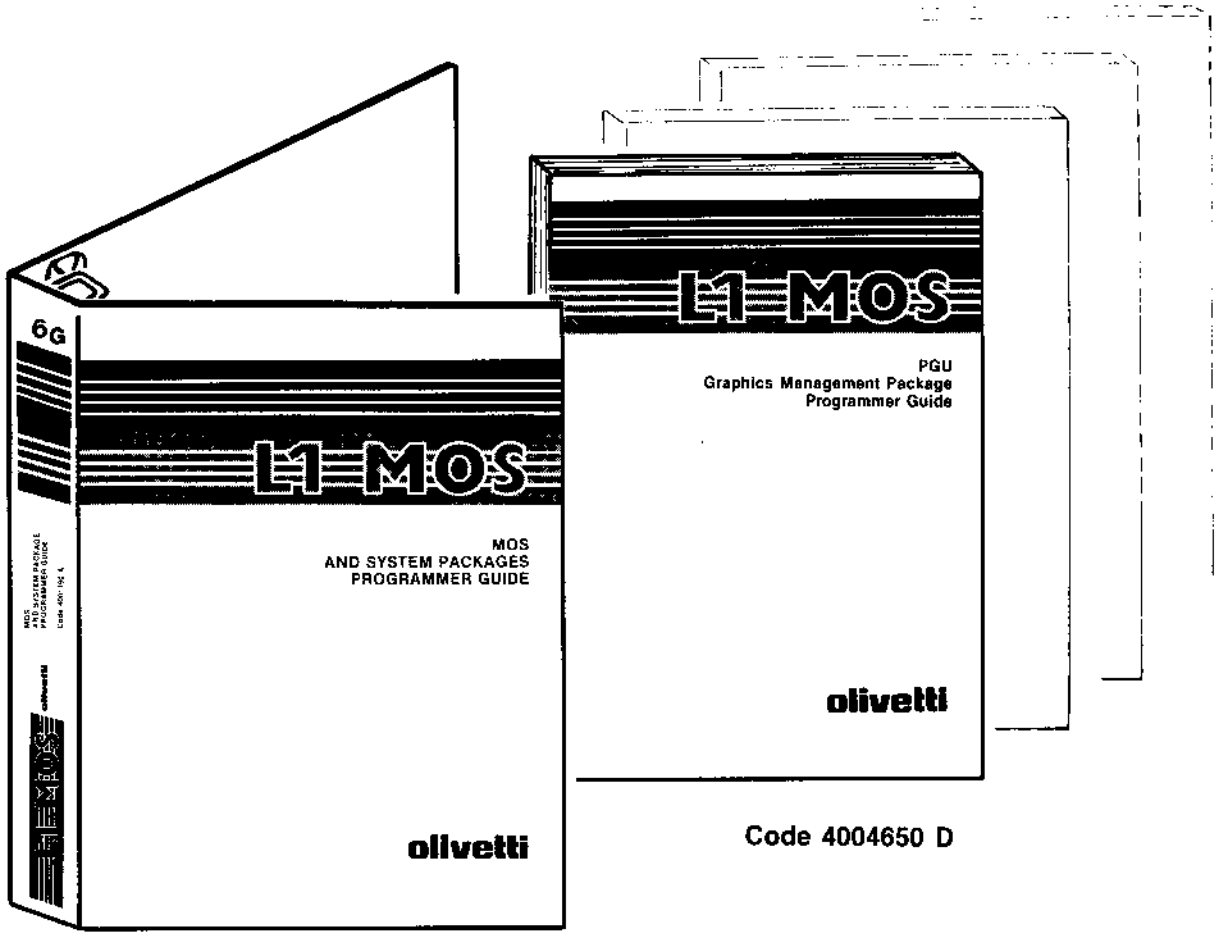
FIRST EDITION: September 1984 - Release 4.0

SECOND EDITION: February 1985 - Release 4.1

UPDATES: May 1985 - Release 5.0

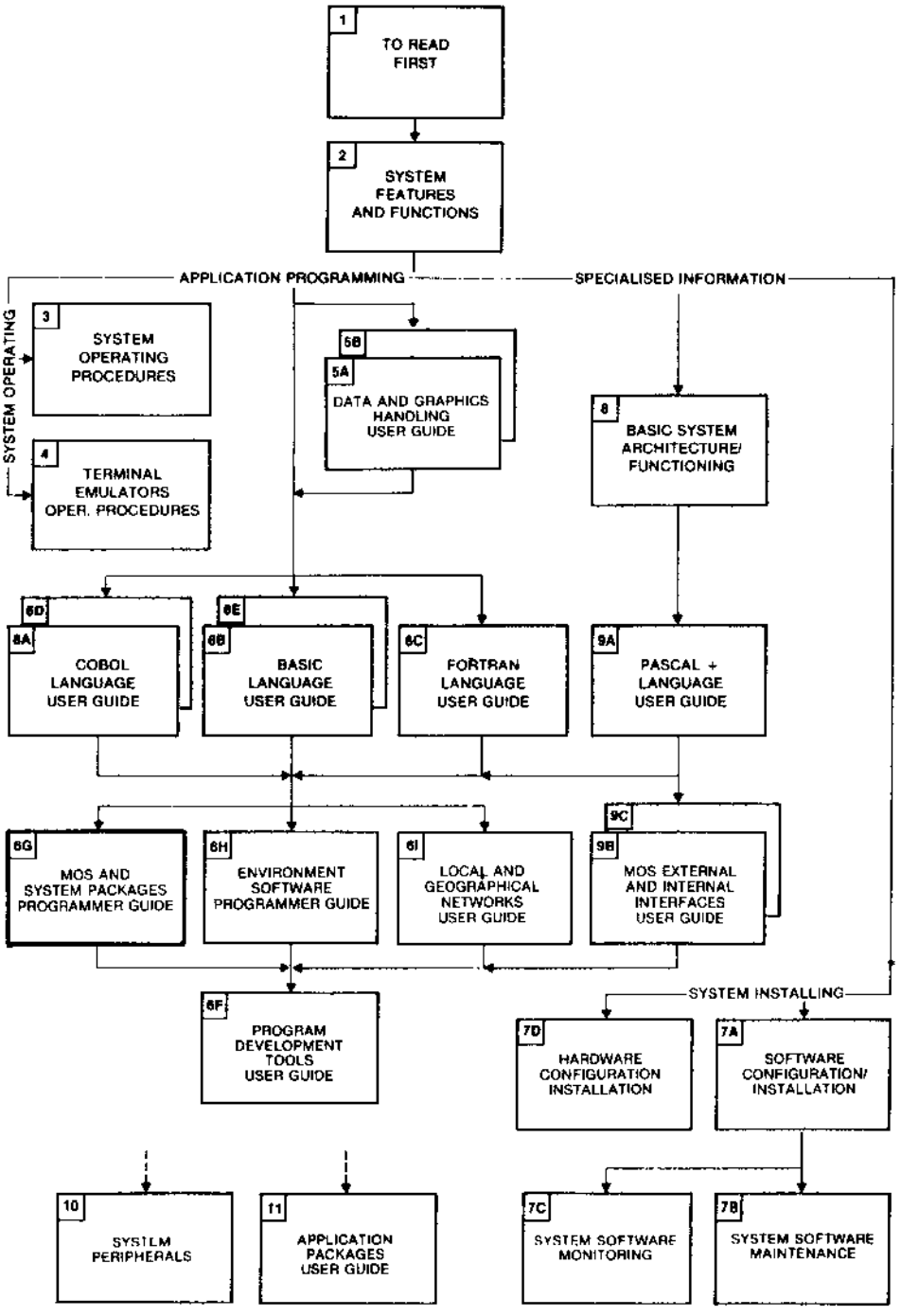
March 1987 - Release 5.2

4004653 N



Code 4001190 A

Code 4004650 D


1
TO READ FIRST
2
SYSTEM FEATURES AND FUNCTIONS
APPLICATION PROGRAMMING
SPECIALISED INFORMATION
SYSTEM OPERATING
3
SYSTEM OPERATING PROCEDURES
4
TERMINAL EMULATORS OPER. PROCEDURES
5B
5A
DATA AND GRAPHICS HANDLING USER GUIDE
8
BASIC SYSTEM ARCHITECTURE/ FUNCTIONING
6D
6A
COBOL LANGUAGE USER GUIDE
6E
6B
BASIC LANGUAGE USER GUIDE
6C
FORTRAN LANGUAGE USER GUIDE
9A
PASCAL + LANGUAGE USER GUIDE
6G
MOS AND SYSTEM PACKAGES PROGRAMMER GUIDE
6H
ENVIRONMENT SOFTWARE PROGRAMMER GUIDE
6I
LOCAL AND GEOGRAPHICAL NETWORKS USER GUIDE
9C
9B
MOS EXTERNAL AND INTERNAL INTERFACES USER GUIDE
6F
PROGRAM DEVELOPMENT TOOLS USER GUIDE
SYSTEM INSTALLING
7D
HARDWARE CONFIGURATION INSTALLATION
7A
SOFTWARE CONFIGURATION/ INSTALLATION
10
SYSTEM PERIPHERALS
11
APPLICATION PACKAGES USER GUIDE
7C
SYSTEM SOFTWARE MONITORING
7B
SYSTEM SOFTWARE MAINTENANCE

CONTENTS

1. INTRODUCTION

The L1 MOS PGU graphics package offers a set of features for developing two dimensional graphic applications. The functions and procedures of this package are described in alphabetical order in Chapter 5 for the COBOL, BASIC and PASCAL+ interfaces and in Chapter 6 for the FORTRAN interface.

The functional capabilities of the PGU graphics package may be subdivided into fourteen general classes: control, output, attributes, windows, graphics cursors, coordinate transformations, image files, elements, segments, graphics hard copy, inquiry, graphics work stations, alphanumeric I/O and a class of special procedures. A description of each class is given separately in what follows.

Control capabilities include opening and closing the PGU session, enabling and disabling buffer use, emptying the buffer and executing the commands stored in it. The PGU may either be set to the graphics or alphanumeric state, and both the graphics and alphanumeric memory can be displayed. The default bounds and the units of measure of the main window may be changed.

Output procedures are used to produce visible display entities, such as displaying one or more points, text strings, drawing lines, polylines, polygons, boxes, circles, ellipses, or any closed image and optionally filling it with a pattern or in one of the eight colors available (if the color controller is present). Geometrical figures such as arcs, sectors or segments of circles and ellipses may also be drawn. There is a function available for moving the current graphics position, one for clearing the contents of the current window, clearing the screen and resetting the initial conditions.

The **attributes** procedures allow the user to set and define modal attributes for succeeding output procedures. The foreground and background colors or the filling pattern may be set to one of eight colors if the color controller is present (black, red, green, yellow, blue, magenta, cyan,and white) and to one of two colors if the monochromatic controller is present (green and black). The output color is also influenced by defining a logic operator or by enabling/disabling specific bit-planes. The line style attribute specifies one of eight line styles, seven of which may be redefined by the user. A number of attributes pertain to text procedures: one of nine character fonts may be chosen to display text strings (string precision, simple, double and complex block letters, complex italics, cyrillic, greek, gothic and user-definable font). Once a character string is displayed it can be magnified, slanted and the spacing between consecutive characters may be modified.

The screen may be divided into rectangular areas called **windows**. Windows are independent of each other and there can be a maximum of sixteen on the screen at the same time. The PGU allows the user to create, close and

select a window. It is also possible to close all windows except for window number 1, the main window, which can never be closed. The PGU graphics package also offers the **viewport** facility which enables a specified area on the tablet and screen for graphics input/output. The origin of the coordinate system is placed in a specified point and the units of measure are set to 1 along both axes. The whole screen is enabled for alphanumeric output. If the window structure has been created the viewport structure logically destroys it but the contents of the screen is not cleared. The limits of the viewport must be within the physical limits of the bit-map, and thus:

- 0 <= x <= 1279 and 0 <= y <= 799 for the color controller
- 0 <= x <= 639 and 0 <= y <= 399 for the 32k monochromatic controller
- 0 <= x <= 2971 and 0 <= y <= 2971 for the tablet

The PGU graphics package offers eight **graphics cursors,** one of which is user-definable. It is possible to select one graphics cursor and move it within the current window, change its size or disable it.

Coordinate data is subjected to **coordinate transformations** such as changing the units of measure and the origin of the coordinate system, translating the origin, rotating the axes and varying the number of pixels per unit of measure.

The PGU **image file** mechanism can store and retrieve special files which contain graphics commands which produce images to be displayed more than once. These files may be opened, closed, executed, listed and the image displayed from an image file may be deleted. It is also possible to execute search and/ or modify operations on the image file by using **editing sessions.**

The functions and procedures used for creating an image are called **elements.** The PGU graphics package contains functions and procedures which execute operations on the element, such as returning its identifier, deleting it and assigning it to a segment.

A **segment** is a collection of elements. The PGU graphics package contains functions and procedures for executing the graphics commands contained in segments, activating new segments, changing segments identifiers, and deleting images displayed via the graphics commands contained in segments.

A **graphics hard copy** of the screen contents can be obtained via the procedures provided for this purpose. The hard copy session must be opened and closed when necessary. It is also possible to set the functioning mode for the graphics printers PR 15, PR 38C, PR 1480 with colour, PR 1550 and PR 1580.

For every feature that may be set by the PGU graphics package, there is an **inquiry** function which permits the user to test its value. The screen type and controller, and printer model used can be inquired. The number of alphanumeric rows and columns which make up a window and the window number are values which can be returned by the PGU. The units of measure

on the x and y axes, the rotation angle formed by the axes of the coordinate system with reference to the default coordinate system (which has a horizontal x axis and a vertical y axes) may be tested. The current foreground and background colors, the description of the fill pattern, the color of a pixel (the term pixel comes from picture element, i.e. the smallest visible entity on the screen), the current graphics position, the current line style are all values which can be inquired at any point in the application program. The bounds and the units of measure of the main window may be tested.

There is also a group of **special procedures** which permit the storing of an image in an array or in a file and displaying what stored when necessary, changing the color codes, displaying the selected part of the bit-map, selecting one of the four bit-map pages and zooming the display. The bit-map memory is an auxiliary RAM memory. If the monochromatic work station is present the bit-map memory may either be of 32k or 128k. In the latter case the bit-map memory is subdivided into four pages of 32k each. The number of pixels which may be represented on the screen are stored in one page. Thus, if a four page bit-map memory is present four complete screen images may be stored. If the color work station is present the bit-map memory is of 128k*3 bytes (one for each color plane) and is not divided into pages.

The PGU graphics package also offers a set of **alphanumeric I/O procedures** for defining a field to contain a string entered via the keyboard, displaying a string, changing visual attributes, returning characters, enabling/disabling the text cursor, and drawing a rectangle to be filled with particular alphanumeric attributes.

Graphics work stations (see the section GRAPHICS WORK STATIONS below) may also be handled by the PGU. When the graphics session is started the input/output graphics work station (the screen/keyboard) is automatically opened. During the graphics session it may be enabled and disabled but never closed. It is also possible to open, close, enable and disable up to four input graphics work stations (the only type available in the present release is the tablet).

GRAPHICS CONCEPTS

Graphical output generated by the L1 MOS PGU Graphics Package comprises two general classes of functions: **output** and **attributes.**

Output procedures are abstractions of basic actions that graphics devices can perform, like drawing lines and locating cursors. Output primitives are defined in a two-dimensional user coordinate space (known as world coordinates, see below). The units of measure and the coordinates of the user coordinate space are established by the application program.

The attributes functions determine the characteristics that an output function will possess when displayed on an output device; e.g., line class, color, intercharacter spacing, etc. The attributes are set modally; i.e., they establish a current value that is assigned to subsequently generated output.

Coordinate data is subjected to transformations that perform a mapping between two coordinate systems, namely:

- **World coordinates,** defined by the user that establish the scaling basis on which the graphical output is described. The world coordinate space definition determines how the coordinates from the application program shall be placed within a window. When a new window is created, it will have the same world coordinate space definition as the parent window, but since the proportions of the two windows have changed, the shape of subsequent output to those windows will change too. The world coordinate space defines a window within the Cartesian plane. The window contains an origin, that is the meeting point of the two axes whose coordinates are (0,0) if no translation is executed.

- **Device coordinates** range from 0 to 639 pixels on the x-axis and from 0 to 399 scanlines on the y-axis (scanline = a row of pixels) for the monochromatic controller, and from 0 to 1279 pixels on the x-axis and from 0 to 799 scanlines on the y-axis for the color controller. Each coordinate pair addresses only one specific pixel.

The world coordinates can be used by most of the functions. The output procedures and attributes are automatically mapped from the user's world coordinate space to the device coordinates via a transformation which is transparent to the user.

The L1 MOS PGU Graphics Package maintains **two current positions,** one for text and one for graphics, and **two cursors,** one for text and one for graphics. Only one of the two cursors (or neither, if so specified) is displayed at any one time. The text current position and the text cursor are always at the same logical location: the point at which the next text output will appear. The graphics current position (the point from which the next graphics output will begin) and the graphics cursor may not coincide. The graphics cursor may be used as an echo symbol to indicate a position on the screen that reflects the values entered by an input device. The current graphics position is used in many but not all graphics output routines, e.g., POLYLINE will establish its own starting point but moves the current position along as it draws, leaving it at the final point. The circle and ellipse procedure moves the current position to the center of the circle or ellipse.

Some graphics procedures may use **absolute coordinates** or **relative coordinates,** other functions may only use absolute coordinates. The distinction is that absolute coordinates are distances along the x and y axes from the origin of the Cartesian plane, while relative coordinates are distances along the x and y axes from the current position.

Most of the output routines are affected by the current **color attributes** and the color **logic-operator attribute.** There is a graphics foreground color which determines the color of graphic output (lines, circles, dots, etc.) and a background color. These attributes are selected from the range of colors available on the specific configuration.

The colors available on the eight-color system are: black, red, green, yellow, blue, magenta, cyan and white. The monochrome system provides two colors, green and black.

The logic-operator attribute determines the resultant output color, considering the type of graphics routine (text or graphics), the setting of the foreground, background or graphics color attribute and the color of the target pixels in the window. There are six logic operators and each one acts on all pixels in determining what the final color shall be. The action occurs one pixel at a time, using the color of the target pixel and that of the new graphics output as operands.

An **element** is a function or procedure which is significant in creating an image. Elements are identified by a code; the following table associates elements with their corresponding codes.

ELEMENT CODE	COBOL ELEMENT	BASIC ELEMENT	FORTRAN ELEMENT	PASCAL ELEMENT
0	ARC	ARC	ARC	ARC
1	CIRCLE	CIRCLE	CIRCLE	CIRCLE
2	COLOR	COLOR	COLOR	COLOR
3	LINETO	LINETO	LINE	LINETO
3	SEGMENT	SEGMENT	SGMENT	SEGMENT
3	BOX	BOX	BOX	BOX
3	BOXFILL	BOXFILL	BOX	BOXFILL
4	-	-	PRESET	-
5	DOT	SHOWDOT	PSET	DOT
6	ROTATE	ROTATE	ROTATE	ROTATE
7	SCALE	SCALE	SCALE	SCALE
8	TRANSLATE	TRANSLATE	TRANSL	TRANSLATE
9	STRFONT	STRFONT	FONT	STRFONT
9	STRMAGNIFICA	STRMAGNIFICA	FONT	STRMAGNIFICATION
9	STRDIRECTION	STRDIRECTION	FONT	STRDIRECTION
9	STRSPACING	STRSPACING	FONT	STRSPACING
10	STRING	STRING	LABEL	STRING
11	INK	INK	INK	INK
11	PAINTSHAPE	PAINTSHAPE	PAINT	PAINTSHAPE
11	PAINTCSHAPE	PAINTCSHAPE	PAINT	PAINTCSHAPE
11	PAINT	PAINT	INK, PAINT	PAINT
11	PAINTCOLOR	PAINTCOLOR	INK, PAINT	PAINTCOLOR
12	POLYLINE	POLYLINE	POLYLN	POLYLINE
13	POLYDOT	POLYDOT	POLYPX	POLYDOT
14	SELECTSTYLE	SELECTSTYLE	STYLE	SELECTSTYLE
14	DEFSTYLE	DEFSTYLE	DEFSTY	DEFSTYLE
14	DEFWIDTH	DEFWIDTH	DEFSTY	DEFWIDTH
15	MASK	MASK	MASK	MASK
16	MOVETO	MOVETO	MOVE	MOVETO
17	SETINSIDE	SETINSIDE	INSIDE	SETINSIDE
18	CIRCLEFILL	CIRCLEFILL	CIRCLF	CIRCLEFILL
19	SCALEAXES	SCALEAXES	SCALAX	SCALEAXES
20	POLYGON	POLYGON	POLYGN	POLYGON

Tab. 1-1 COBOL, BASIC, FORTRAN and PASCAL+ Elements

A **segment** is a collection of elements and is identified by a number in the range 0 to 254 inclusive. An image file is a collection of segments. All the elements called between the opening and closing of an image file are registered on the file and are logically associated with the active segment (only one segment at a time is active). When the image file is opened segment 0 is activated. There is a graphics function available for changing the active segment. If more than one segment is to be displayed then the elements of those segments are executed in the same order they appear within the image file.

Once an image file is recorded it may be modified by using the **editing** functions. An editing session can not be opened during the creation of an image file, that is between the opening and closing of an image file, and vice versa. The editing of an image may be carried out either from file or from memory.

In the first case, each time a search and/or modify procedure is called the file containing the image is read and/or modified. The deleted elements are simply marked and the space they occupy is not left free. A specific function is to be used for compacting the file and for removing the deleted elements. There are no limits to the length of the image file.

In the second case, the file is loaded into memory when the editing session is opened and is rewritten when the editing session is closed. All the search and/or modify operations are executed in memory. The deleted elements are not registered on file when the editing session is closed. The maximum length of an image file which may be modified in memory is 64,000 bytes.

GRAPHICS WORK STATIONS

The concept of a graphics work station is that of an abstract graphical work station which provides the logical interface by means of which the application program controls graphics physical devices. A graphics input/output work station generally is the graphics screen/keyboard, a graphics input work station may be physical devices such as a tablet, a mouse, etc., and a graphics output work station may be a plotter, a graphic printer, etc. From now on, the terminology "graphics work station" will be abbreviated to "work station".

The types of work station that can currently be used are: the integrated graphics video/keyboard, the M30, the M31 and OLIVETTI PC, used as graphics I/O work stations, the tablet, used as a graphics input work station, and the graphics printer, used as a graphics output work station.

More than one work station may be opened and active at one time. When the graphics session is opened the input/output work station is automatically opened and its identifier is 0. This work station may only be enabled or disabled (it can not be closed). If this work station is disabled all the procedures executed before its activation are not sent to the work station but are registered on the image file (if it is being created). Four input work stations may be opened (besides work station 0) and may be supported by one or more tablets (one tablet may contain up to four logical work stations). These input work stations permit the user to choose a point and to return its world coordinates. If work station 0 is

used this is obtained via the following steps:

- Selection of the cursor (if necessary).
- Execution of the procedure(s) which moves the cursor and returns its position.
- Use of the arrow keys.
- Selection of any key to return the cursor's position.

If any other work station is used the point selection and its world coordinates are obtained via the following steps:

- Selection of the cursor (if necessary).
- Movement of the stylus (or four button cursor) on the tablet.
- Pressing the stylus on the tablet (or one of the buttons on the four button cursor).

The procedures VIEW and SCALE have an effect on all active work stations. It is possible to map the whole tablet or part of it on the screen by executing these two procedures and by properly activating and disactivating the work stations. Let's suppose that the area on the tablet identified by the diagonal whose extremes are the points (20,20) and (120,120) (work station 1) is to be mapped on the screen in the area whose diagonal is identified by the points (500,250) and (600,350) and that both work stations are enabled. The following steps are to be executed:

- Disable work station 0.
- Select the input area on the tablet via the VIEW procedure.
- Disable work station 1.
- Enable work station 0.
- Select the output area on the screen via the VIEW procedure.
- Enable work station 1.
- Execute the SCALE procedure if the units of measure on the two work stations are to be the same.

It is to be noted that if the number of units of measure of the area created via the VIEW procedure on the input work station is greater than those on the output work station then the chosen point may be within the input work station area but not within the output one. The cursor is then left on the last position on which it can be displayed but its coordinates are returned.

PGU FUNCTIONAL GROUPS

The functions of the PGU Graphics Package can be grouped into thirteen functional groups, as shown in the following table. The name of each procedure for the COBOL, BASIC and PASCAL+ interfaces is given under the heading GLOBAL NAME, whereas for the FORTRAN interface the specific PGU FORTRAN function is specified. The global name is the COBOL, BASIC and PASCAL+ procedure name except for DOT (BASIC name SHOWDOT), REF (PASCAL+ name Refr) and STRMAGNIFICA (PASCAL+ name StrMagnification).

FUNCTIONAL GROUP	GLOBAL NAME	FORTRAN FUNCTION
CONTROL		
Initialise PGU	OPENPGU	-
Close PGU	CLOSEPGU	-
Change limits of main window	BOUNDS	BOUNDS
Enable buffer use	BSET	BSET
Disable buffer use	BRESET	BRESET
Empty buffer	FLUSH	BFLUSH
Set alphanumeric state	SETALPHA	-
Set graphic state	SETGRAPH	-
Display text/graphic bit-map	REF	REFRS
OUTPUT		
Move current position	MOVETO	MOVE
Display a point	DOT	PSET
Delete a point	-	PRESET
Display points	POLYDOT	POLYPX
Display a segment	SEGMENT	SGMENT
Display a segment	LINETO	LINE
Display a rectangle	BOX	BOX
Display filled rectangle	BOXFILL	BOX
Display connected segments	POLYLINE	POLYLN
Display filled polygon	POLYGON	POLYGN
Display arc, sector or segment	ARC	ARC
Display circle or ellipse	CIRCLE	CIRCLE
Display filled circle or ellipse	CIRCLEFILL	CIRCLF
Color closed area	INK	INK
Fill area with pattern	PAINT	PAINT, INK
Fill area with color pattern	PAINTCOLOR	PAINT, INK
Display a string	STRING	LABEL
Clear current window	CLS	CLRWIN
Reinitialise PGU	CLEAR	CLEAR

Tab. 1-2 PGU Graphics Procedures - Functional Groups (cont.)

FUNCTIONAL GROUP	GLOBAL NAME	FORTRAN FUNCTION
ATTRIBUTES		
Change foreground and background	COLOR	COLOR
Specify enabled bit-planes	MASK	MASK
Define logic operator	LOGICACTION	not separate function
Redefine line style	DEFSTYLE	DEFSTY
Change line width	DEFWIDTH	DEFSTY
Select line style	SELECTSTYLE	STYLE
Relative/absolute coordinates	COORDMODE	not separate function
Define pattern	PAINTSHAPE	PAINT
Define color pattern	PAINTCSHAPE	PAINT
Choose filling mode	SETINSIDE	INSIDE
Choose font file	STRFONT	FONT
Change character height and width	STRMAGNIFICA	FONT
Slant character string	STRDIRECTION	FONT
Change character spacing	STRSPACING	FONT
WINDOWS		
Create new window	CREATEWINDOW	NEWWIN
Close window	CLOSEWINDOW	CLSWIN
Close all windows	CLOSEALLWIND	CLSWIN
Select window	SELECTWINDOW	SETWIN
Create viewport	VIEW	VIEW
GRAPHICS CURSORS		
Select graphics cursor	CURSORPOINT	CURSOR
Move graphics cursor	CURSORMOVE	GIN
Define cursor shape	CURSORSHAPE	DEFCUR
Change cursor size	CURSORSIZE	DEFCUR
Disable graphics cursor	CURSOROFF	DEFCUR
COORDINATE TRANSFORMATION		
Change units of measure	SCALE	SCALE
Cancel ROTATE and TRANSLATE	SCALE	SRESET
Change number of pixels per unit of measure	SCALEAXES	SCALAX
Translate origin	TRANSLATE	TRANSL
Rotate axes	ROTATE	ROTATE

Tab. 1-2 PGU Graphics Procedures - Functional Groups (cont.)

FUNCTIONAL GROUP	GLOBAL NAME	FORTRAN FUNCTION
IMAGE PROCEDURES		
Open image file	OPENIMAGE	OPNIMG
Close image file or file in editing	CLOSEIMAGE	CLSIMG
Execute image file or file in editing	IMAGE, REDRAWALL	IMAGE
Delete image from screen	DELIMAGE	IMAGE
Compact image file or file in editing	IMAGECOMPACT	IMGCMP
List image file or file in editing	LISTIMAGE	LSTIMG
Open editing session on image file	OPENIMGEDIT	OPNED
ELEMENTS		
Return element identifier	SEARCHELEM	SRCELM
Delete an element	DELETEELEM	DELELM
Delete elements	DELETEALLELE	DELALL
Assign element to segment	EXCHANGEELEM	EXCELM
Assign elements to segment	EXCHALLELEM	EXCALL
SEGMENTS		
Execute segments contained in an image file or in an image in editing	IMAGES, REDRAW	IMGSEG
Activate a segment	CHANGEACTSEG	CHNSEG
Change segment identifier	CHANGESEGID	CHSGID
Delete elements of a segment	DELETESEGM	DELSEG
Delete displayed segments image	DELIMAGES	IMGSEG
GRAPHICS HARD-COPY (See the following note)		
Open graphic hard copy session	OPENGRPR	OPNPRT
Close graphic hard copy session	CLOSEGRPR	CLSPRT
Hard copy	GRAPHICPRINT	SPRINT
Set graphics printer functioning mode	SETGRPRMODE	GPRMOD

Tab. 1-2 PGU Graphics Procedures - Functional Groups (cont.)

FUNCTIONAL GROUP	GLOBAL NAME	FORTRAN FUNCTION
INQUIRY		
Return screen type	INQDEVTYPE	INQI
Return printer type	INQGRPRTYPE	INQI
Return limits of main window	INQBOUNDS	INQI
Return window information	INQWINDOW	INQW
Return units of measure	INQMUNITS	INQF
Return rotation angle	INQROTATION	INQF
Return current graphics position	INQCURPOS	INQF
Return current colors	INQCOLOR	INQI
Return current line style	INQLSTYLE	INQI
Return current fill pattern	INQPSHAPE	INQI
Return color of a pixel	POINTCOLOR	POINT
Return device coordinate	-	SCALEX, SCALEY
SPECIAL		
Store display in array	GET	GET, GET2
Store display on file	GETANDSAVE	GET,GET2
Display contents of array	PUT	PUT, PUT1, PUT2
Display contents of file	LOADANDPUT	PUT, PUT1, PUT2
Change color codes	CHCOLOR	CHCOL
Display part of bit-map	PAN	PAN
Select one bit-map page	PAGE	PAGE
Zoom display	ZOOM	ZOOM
ALPHANUMERIC I/O		
Define field for string	INPUTFIELD	-
Display string	OUTPUTFIELD	-
Change attributes	SETATR	-
Read character	READCHAR	READCH
Test character	TESTCHAR	TESTCH
Enable text cursor	ALPHACURSON	ACON
Disable text cursor	ALPHACURSOFF	ACOFF
Draw rectangle	FRAME	FRAME, FRAME1, FRAME2
GRAPHICS WORK STATIONS		
Open graphics work station	OPENGRWS	OPGRWS
Close graphics work stations	CLOSEGRWS	CLGRWS
Enable graphics work station	ACTIVGRWS	ACGRWS
Disable graphics work station	DEACTIVGRWS	DAGRWS

Tab. 1-2 PGU Graphics Procedures - Functional Groups

Note: It is possible to print the contents of the whole screen or part of it on graphics printers of type /A (serial interface with OLIVETTI standard and connectable to the L1 MOS system or to the PC under work station emulation with the program OLIEMU) or on printers of type /B (parallel interface with IBM standard and connectable to the PC in work station emulation with the program WSELAN).

The graphics printers of type /A that can be used are the following: PR 15, PR 17, PR 19, PR 38, PR 38C, PR 1450, PR 1470, PR 1480, PR 1480 with colour, PR 1490, PR 1550, PR 1570, PR 1580 and PR 2400.

The graphics printers of type /B that can be used are the following: PR 11B, PR 13B, PR 15B, PR 17B, PR 19B, PR 38B, PR 38BC, DM 280, DM 290, DM 580 and DM 590.

USE OF THE PGU GRAPHICS PACKAGE

The file structure which is necessary for correct use of the PGU when using an integrated work station or the M30 or M31 used as work stations is:

```
/IPL/DPC/GRAPHICS/PGU
/IPL/DPC/GRAPHICS/FONT0
/IPL/DPC/GRAPHICS/FONT1
/IPL/DPC/GRAPHICS/FONT2
/IPL/DPC/GRAPHICS/FONT3
/IPL/DPC/GRAPHICS/FONT4
/IPL/DPC/GRAPHICS/FONT5
/IPL/DPC/GRAPHICS/FONT6
/IPL/DPC/GRAPHICS/FONT7
/IPL/DPC/GRAPHICS/FONT8 ( this file is present only if created and
                          defined by the user )
```

The file /IPL/DPC/GRAPHICS/PGU contains the PGU graphics package code. The other files contain the description of the character fonts available and they need not be in the directory /IPL/DPC/GRAPHICS but may also be placed elsewhere and with different names from the given ones. In this case, it is necessary to connect them to the global names FONT0, FONT1, ..., FONT8 via the Shell command CONN.

For the file structure which is necessary for the use of PGU when using a PC as a work station, see the section "Personal Computers used as Graphics Work Stations".

The file structures which must be linked or included are specified in chapters 2, 3 and 4 for the COBOL, Compiled BASIC and PASCAL+ programming languages, respectively, and in chapter 6 for the FORTRAN interface.

M30 AND M31 USED AS GRAPHICS WORK STATIONS

M30s and M31s used as L1 MOS work stations with graphics capabilities can recognise the PGU graphics commands. However, there are some limits on the use of the procedures, as follows:

- The use of the following procedures (FORTAN functions) is not allowed:
  - GET (GET,GET2)
  - GETANDSAVE (GET,GET2)
  - GRAPHICPRINT (SPRINT)
  - POINTCOLOR (POINT)
  - ZOOM (ZOOM) if colour is not available
- The hardware coordinate ranges are from 0 to 639 on the x-axis and from 0 to 399 on the y-axis for the M31 and monochromatic M30, and from 0 to 1279 on the x-axis and from 0 to 799 on the y-axis for the M30 with colour.
- The viewport bounds are: 0 <= x <= 639 and 0 <= y <= 399 for the M31 and the monochromatic M30, 0 <= x <= 1279 and 0 <= y <= 799 for the M30 with colour.

Also, if the colour controller is present:

- The bit-map graphic memory is not managed separately from the alphanumeric memory. Graphics and text are therefore managed together, and they cannot be erased selectively.
- The PAGE procedure (PAGE) cannot be used.

For general information on the use of the M30 and M31 as work stations with an L1 MOS system, see the relevant parts of the manual MOS, Programmer Guide.

PERSONAL COMPUTERS USED AS GRAPHICS WORK STATIONS

The PC emulating an L1 MOS work station with graphics capabilities can recognise the PGU graphics commands. The term "PC" indicates the OLIVETTI Personal Computers: M19, M24, M24 SP and M28.

To enable graphics emulation the programs GIOM (Graphical Input/Output Manager) or GIOMR (Graphical Input/Output Manager Reduced) must be present on the PC in addition to L1WSE, WSELAN or OLIEMU emulation programs. The GIOM module, unlike the GIOMR, allows unloading of the CPU and memory of the L1 MOS system, since the "Output" graphics functions are performed on the PC.

When GIOM is used, in order to use PGU on the PC used as a work station, it is sufficient to have the file **/IPL/DPC/GRAPHICS/PGU_** available on the L1 MOS instead of the file **/IPL/DPC/GRAPHICS/PGU** , while the files **FONT...FONT8** must not be present on the LI MOS system, but must reside on the PC in the directory **GRAPHICS**

The following table summarises the above:

INTEGRATED WORK STATION OR M30/M31 AS W.S.	PC + GIOM	PC + GIOMR
on L1 : PGU FONT0...FONT8	on L1 : PGU_ on PC : L1WSE or WSELAN or OLIEMU GIOM FONT0...FONT8	on L1 : PGU FONT0..FONT8 on PC : L1WSE or WSELAN or OL1EMU GIOMR

Tab. 1-3 Software Modules Required for Graphics Capabilities

For colour graphic emulation, the EGC (Enhanced Graphic Colour) board must also be installed on the PC, and to enable the hard copy feature on type /8 printers, either the HCBM or HCBC module is necessary: the former for use with monochromatic printers and the latter for use with colour printers.

For general information on the use of the PC as an L1 MOS work station, see the relevant parts of the manuals MOS Programmer Guide and MOS Operating Guide.

Limitations on the PC Used as a Graphics Work Station

Depending on the type of hardware used, there are limits on the use of PGU on a PC emulating a work station. The limits are as follows:

- The hardware coordinate ranges are from 0 to 639 on the x-axis and from 0 to 399 on the y-axis.
- The viewport bounds are 0 <= x <= 639 and 0 <= y <= 399.
- The following procedures (FORTRAN functions) are ignored by the emulator program or are subject to limitations:

  - DEFWIDTH (DEFSTY limited) if the GIOMR program is used on the PC
  - PAGE (PAGE)
  - PAN (PAN)
  - ZOOM (ZOOM)

- Colour graphics capabilities are not available on the M19 PC.

- The distance between the pixels (luminous dots) on the graphics screen of the integrated work station is the same both horizontally and vertically, whereas this is not true for the PC. Pictures which are to be displayed on an integrated L1 MOS work station are therefore displayed on the PC with a scalar ratio (y = 6/5x).

  This is only a visual difference, however, since the dimensions of the picture in terms of pixels are correct.

  An INQDEV command can be inserted at the beginning of the program to identify whether the terminal being used is a PC or not. If it is, the parameters can be modified by means of an appropriate multiplication factor using the SCALE procedure.

- The definition of the styles for line tracing is different from that used on the L1 MOS system. However, the emulation programs modify the style in such a way as to provide the PC with a format which is similar to, though not exactly the same as, that of the L1 MOS system.

Furthermore, if the EGC board is not present on the PC there are the following limitations:

- The bit-map graphic memory on the PC is not managed separately from the alphanumeric memory. Graphics and text are therefore managed together, with the following consequences:
  - The contents of the screen cannot be erased selectively
  - Scrolling of the alphanumeric part of the screen also implies scrolling of the graphic part.

- All the commands referring to the alphanumeric screen attributes are ignored.

It should be noted that with the L1WSE emulator program none of the printers can be managed.

LANGUAGE INTERFACES - INITIAL CONDITIONS

The PGU COBOL, BASIC, PASCAL+ and FORTRAN interfaces are initialised to the following operating mode:

- Work station 0 is opened and enabled.
- Window number 1 (the main window) occupies the whole screen.
- The screen world coordinates range from 0 to 639 on the x axis and from 0 to 399 on the y axis, coinciding with the device coordinates.
- The tablet world coordinates range from 0 to 2971 on the x axis and from 0 to 2971 on the y axis, coinciding with the device coordinates.
- The origin of the device coordinate system is placed in the lower left hand corner of the screen, the x-axis is horizontal and the y-axis is vertical.
- Coordinate data is interpreted as absolute coordinates.
- Color depends on the system configuration:

  a) The monochrome system sets the background color to 0 = black and the graphics color to 1 = green.

  b) The eight-color system sets 0 = black to the background color, 1 = red to the graphics color, 2 = green, 3 = yellow, 4 = blue, 5 = magenta, 6 = cyan, 7 = white.

- The logic operator is "replace" for the COBOL, BASIC and PASCAL+ interfaces and "noact" for the FORTRAN interface, which display graphics output in the chosen color.
- The line style is solid line, 1 pixel wide.
- Buffer use is enabled.
- No graphics cursor is displayed.
- Page 1 of the bit-map memory is displayed.
- All three bit-planes are enabled (color system).
- The fill pattern is the sequence of characters %FF.
- The graphics position is (0.0, 0.0).
- The font file is FONT1 (simple block letters).
- Strings are displayed horizontally and each character is 10 pixels high (the width of each character and the spacing between two consecutive characters vary from one character to another).

2. COBOL INTERFACE

This chapter contains the following information:

- The general structure of a COBOL call for the PGU interface; a detailed description for each procedure call is found in Chapter 5.

- The description of all the parameters used by the procedures of the COBOL interface.

COBOL CALL

The structure of a COBOL procedure call is:

```
CALL "literal" USING parameter_list.
```

where "literal" is the name of the COBOL interface graphics procedure and "parameter list" is the list of parameters. A complete list of all these names can be found in Chapter 1, Tab. 1-2.

The PGU COBOL procedures are contained in the COBOL run-time, COB_RTS. The application program must link the library EXTINTF.LIB which includes the PGU COBOL interface. This is already done in the automatic compile-link procedure which uses the standard RTLINK file of directives for the linker.

PARAMETER DESCRIPTION

The following is a list of all the parameters of the PGU COBOL interface. The parameters are listed in alphabetical order and for each one the structure and description is furnished.

It is to be noted that:

- The character underscore (_) contained in the name of some COBOL parameters is to be substituted with a dash (-) because it is not accepted at program compile time.

- The names of some parameters are COBOL keywords (e.g. character, column, data, error, etc.) and must thus be substituted with other names (e.g. character1, column1, data1, error1, etc.).

action_verb

77 action_verb PIC 9(2) COMP.

Logic operator code.

angl1

77 angl1 COMP-1.

Initial angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis.

angl2

77 angl2 COMP-1.

Final angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis.

angle

77 angle COMP-1.

Angle, measured in radians from the positive x-axis to the positive y-axis, between the x-axis and the direction of the string.

angle_rotation

77 angle_rotation COMP-1.

Rotation angle.

BM_backColor

77 BM_backColor PIC S9(4) COMP.

Value which has an effect only if the parameter "screenType" specifies alphanumeric text and graphics. It specifies the color code on the graphic bit-map considered as background color of the alphanumeric text.

prMode

```
01  prMode.
    02  resolution_1550  PIC S9(4) COMP.
    02  color            PIC S9(4) COMP.
    02  density_1580     PIC S9(4) COMP.
    02  zoom_1580        PIC S9(4) COMP.
    02  end_flag         PIC S9(4) COMP.
```

Graphic printer functioning mode.

print_type

```
77 print_type PIC 9(2) COMP.
```

Specifies whether a positive or negative hard copy is to be obtained.

printerId

```
77 printerId PIC S9(9) COMP.
```

System identifier of the printer furnished by MOS and is significant only if the workstation printer is opened in exclusive mode.

printerType

```
77 printerType PIC S9(4) COMP.
```

Specifies the printer to be used.

putType

```
77 putType PIC 9(2) COMP.
```

Specifies where the image is to be displayed.

quadrant

```
77 quadrant PIC 9(2) COMP.
```

Specifies in which part of the parent window the new window is to be opened.

r1

```
77 r1 PIC S9(4) COMP.
```

Alphanumeric coordinate along the y-axis (the row) of the first end of the diagonal of the rectangle.

r2

77 r2 PIC S9(4) COMP.

Alphanumeric coordinate along the y-axis (the row) of the second end of the diagonal of the rectangle.

radius

77 radius COMP-1.

Radius of the circle or the horizontal semi-axis of the ellipse in world coordinates.

ratio

77 ratio COMP-1.

Ratio between the vertical and horizontal axes of an ellipse or ratio between the character height and width. The initial value is 1.

row

77 row PIC S9(4) COMP.

Number of the screen row from where the input field starts or from where the output field will be displayed.

screen

77 screen PIC 9(2) COMP.

Specifies whether the hard copy is of the whole screen or of the current window.

screenType

77 screenType PIC S9(4) COMP.

Specifies whether the color to be changed is associated with alphanumeric text and graphics or only graphics.

3. COMPILED BASIC INTERFACE

This chapter contains the following information:

- The general structure of a Compiled BASIC call for the PGU interface; a detailed description for each procedure call is found in Chapter 5.
- The description of all the parameters used by the BASIC interface.

BASIC CALL

The structure of a Compiled BASIC procedure call is:

```
CALL ProcedureName (parameter_list)
```

where ProcedureName is the name of the Compiled BASIC interface graphics procedure. A complete list of all these names can be found in Chapter 1, Tab.1-2.

Each PGU procedure must be declared as follows:

```
DECLARE SYSTEM PROCEDURE ProcedureName (parameter_list)
```

A complete list of all the procedure declarations is found at the end of this chapter.

The PGU BASIC interface (BA_PGU_ui.obj) is contained in the library extintf.lib which is present under the directory /IPL/DPC/BAS_RTS.

PARAMETER DESCRIPTION

The BASIC compiler takes care of converting input numeric parameter types according to what stated in the parameter declaration.

The following is a list of all the parameters of the PGU Compiled BASIC interface. The parameters are listed in alphabetical order and for each one the structure and description is given.

The symbolic value 10 has been used to indicate the line number.

action_verb$

10 declare string action_verb$*1

Logic operator code. The value must be given in the hexadecimal format.

angl1

10 declare numeric angl1*S

Initial angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis.

angl2

10 declare numeric angl2*S

Final angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis.

angle

10 declare numeric angle*S

Angle, measured in radians from the positive x-axis to the positive y-axis, between the x-axis and the direction of the string.

angle_rotation

10 declare numeric angle_rotation*S

Rotation angle.

BM_backColor

10 declare numeric BM_backColor*I

Value which has an effect only if the parameter "screenType" specifies alphanumeric text and graphics. It specifies the color code on the graphic bit-map considered as background color of the alphanumeric text.

4. PASCAL+ INTERFACE

This chapter contains the following information:

- The general structure of a PASCAL+ call for the PGU interface; a detailed description for each procedure call is found in Chapter 5.
- The description of all the parameters used by the PASCAL+ interface.

PASCAL+ CALL

The structure of a procedure call and that of a function call is:

```
ProcedureName ( parameter_list );

error := FunctionName ( parameter_list );
```

where "ProcedureName" and "FunctionName" is the name of the PASCAL+ interface graphics procedure or function respectively. A complete list of all these names can be found in Chapter 1, Tab.1-2. The variable "error" contains the function reply code. "parameter_list" is the list of parameters the procedure or function requires. In the following, a detailed description of each parameter is given.

The application program may include the module "PGU.d" in order to access the PASCAL+ graphics procedures and functions declarations and the parameter types and must link the "P_PGU_ui.obj" PASCAL+ interface.

PARAMETER DESCRIPTION

The following is a list of all the parameters of the PGU PASCAL+ interface. The parameters are listed in alphabetical order and for each one the structure and description is furnished.

action_verb

```
T_action = ( replace_, xor_, and_, not_, or_, erase_ );
action_verb : T_action;
```

Logic operator code. Passed by value.

angl1

angl1 : real;

Initial angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis. Passed by value.

angl2

angl2 : real;

Final angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis. Passed by value.

angle

angle : real;

Angle, measured in radians from the positive x-axis to the positive y-axis, between the x-axis and the direction of the string. Passed by value.

angle_rotation

angle_rotation : real;

Rotation angle. Passed by value.

BM_backColor

BM_backColor : integer;

Value which has an effect only if the parameter "screenType" specifies alphanumeric text and graphics. It specifies the color code on the graphic bit-map considered as background color of the alphanumeric text. Passed by value.

O_mode

```
T_ImgOpenMode = (rewrite, append);
O_mode : T_ImgOpenMode;
```

Specifies how the image file is to be recorded. Passed by value.

OdataSize

```
OdataSize : integer;
```

Number of characters to be used for the initialisation of the input field. Passed by value.

old_atr

```
byte = -128..127;
old_atr :byte;
```

Specifies which attributes are to be disabled. Passed by value.

openMode

```
T_PrOpenMode = (notOpen, alreadyOpen , systemPrt);
openMode : T_PrOpenMode;
```

Specifies the graphics printer open mode. Passed by value.

openPgu

```
T_PguOpenMode = (i_t, e_t, i_s, e_s, b_m);
openPgu : T_PguOpenMode;
```

Specifies the PGU initialisation mode. Passed by value.

option

```
T_arcOption : (arc_, seg_, sect_, sectFill_);
option : T_arcOption;
```

Value specifying whether an arc, a segment, a sector, or a filled sector of a circle or an ellipse is to be drawn. Passed by value.

page_number

```
page_number : integer;
```

Page to be selected. Passed by value.

paint_Cshape

```
byte = -128..127;
T_colorShape = packed array [1..30] of byte;
paint_Cshape : T_colorShape;
```

Thirty byte array specifying the new fill character. Passed by VAR.

paint_shape

```
byte = -128..127;
T_shape =packed array [1..10] of byte;
paint_shape : T_shape;
```

Ten byte array specifying the new fill character. Passed by VAR.

pattern

```
byte = -128..127;
T_style = packed array [1..8] of byte;
pattern : T_style;
```

Eight byte array containing the description of the line style. Passed by VAR.

pmask

```
pmask : integer;
```

Bit_planes mask. Passed by value.

poly_shape

```
T_point = record
            x : real;
            y : real
          end;
poly_shape : packed array [min..max:integer] of T_point;
```

Array containing the world coordinates of the points to be displayed or connected. The maximum number of points to be displayed or connected is 50. Passed by VAR.

position

```
position : integer;
```

Specifies the position (row or column) where the parent window is to be split or the position of the element in the segment or file. Passed by value.

prMode

```
T_grPrMode = record
               resolution_1550 : integer;
               color           : integer;
               density_1580    : integer;
               zoom_1580       : integer;
               endFlag         : integer
             end;
prMode : T_grPrMode;
```

Graphic printer functioning mode.

print_type

```
T_print_mode = (normal, reverse);
print_type : T_print_mode;
```

Specifies whether a positive or negative hard copy is to be obtained. Passed by value.

printerId

```
R_id = record
       end;
T_id = ^R_id;
t_systemId = record
               net   : T_id;
               local : T_id
             end;
printerId : t_systemId;
```

System identifier of the printer furnished by MOS and is significant only if the workstation printer is opened in exclusive mode. Passed by value.

printerType

```
printerType : integer;
```

Specifies the printer to be used. Passed by VAR.

putType

```
T_put_mode= (none, first, firstandsecond);
putType : T_put_mode;
```

Specifies where the image is to be displayed. Passed by value.

quadrant

```
T_quadrant = (top, bottom, left, right);
quadrant : T_quadrant;
```

Specifies in which part of the parent window the new window is to be opened. Passed by value.

r1

```
r1 : integer;
```

Alphanumeric coordinate along the y-axis (the row) of the first end of the diagonal of the rectangle. Passed by value.

r2

```
r2 : integer;
```

Alphanumeric coordinate along the y-axis (the row) of the second end of the diagonal of the rectangle. Passed by value.

radius

```
radius : real;
```

Radius of the circle or the horizontal semi-axis of the ellipse in world coordinates. Passed by value.

ymin

ymin : integer;

Device coordinate along the y-axis of the lower left hand corner of the viewport. Passed by value.

zlevel

zlevel : integer;

Zoom level. Passed by value.

This procedure draws an arc, a sector, a segment, or a filled sector of a circle or an ellipse.

COBOL Call

```
CALL "ARC" USING error, x_centre, y_centre, radius, ratio, angl1, angl2,
                 option.
```

BASIC Call

```
CALL ARC ( error, x_centre, y_centre, radius, ratio, angl1, angl2, option$ )
```

PASCAL Call

```
error := Arc ( x_centre, y_centre, radius, ratio, angl1, angl2, option );
```

PARAMETER	DESCRIPTION
INPUT	
x_centre	World coordinate along the x-axis of the centre of the circle or ellipse whose arc is to be drawn.
y_centre	World coordinate along the y-axis of the centre of the circle whose arc is to be drawn.
radius	Radius of the circle or the horizontal semi-axis of the ellipse in world coordinates. This value must be greater than zero and equal to or greater than one pixel.
ratio	Ratio between the vertical and horizontal axes of an ellipse. This value must be greater than zero. The value one is the ratio for a circumference if the scale along the x and y axes coincide.
angl1	Initial angle (of an arc, sector or segment of a circle or an ellipse) measured in radians from the positive x-axis to the positive y-axis. The value must belong to the range -2π to 2π.
angl2	Final angle (of an arc, sector or segment of a circle or or an ellipse) measured in radians from the positive x-axis to the positive y-axis. The value must belong to the range -2π to 2π.
option	Value specifying whether an arc, a segment, a sector, or a filled sector of a circle or an ellipse is to be drawn. It may be set to one of the following values: 0 arc 1 segment 2 sector 3 filled sector See characteristic 3

OUTPUT

error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
17	Out of volume space.
19	Division by zero.
20	Overflow.

Characteristics

1. After the execution of this procedure the current graphics position coincides with the centre of the circle or ellipse.

2. After the transformation from device coordinates to world coordinates has been applied, the values of the parameters "radius" and "ratio" must be equal to or greater than one pixel.

3. If the "option" parameter is set to 3 then the sector is filled with the current foreground color (if the SETINSIDE procedure has not been executed) or with what specified by the SETINSIDE procedure which selects the filling mode.

This procedure clears text and/or graphics from the current window.

COBOL Call

```
CALL "CLS" USING screen_type.
```

BASIC Call

```
CALL CLS ( screen_type$ )
```

PASCAL Call

```
error := Cls ( screen_type );
```

PARAMETER	DESCRIPTION
INPUT	
screen_type	Specifies whether text and/or graphics is to be cleared from the current window. The values are: 0 both text and graphics 1 only text 2 only graphics
OUTPUT	
error	Returned by the PASCAL+ interface. It is always 0.

Characteristics

1. If the "openPgu" parameter of the OPENPGU procedure has been set to 1 or 3 (meaning that procedures other than those of the PGU are used for alphanumeric I/O) then the only value allowed for the parameter "screen_type" of this procedure is 2 (only graphics).

2. The text cursor is placed in the upper left hand corner of the current window. The graphics cursor and the current graphics position are placed in the origin of the world coordinates system.

3. This procedure has no effect when using an integrated work station with colour, or the M30 with colour or the monochromatic PC used as work stations, since the graphics bit-map and alphanumeric text cannot be displayed separately on these.

COBOL Example

```
MOVE 2 TO SCREEN_TYPE.
CALL "CLS" USING SCREEN_TYPE.
```

COLOR

This procedure changes the foreground and the background colors for the current window.

COBOL Call

```
CALL "COLOR" USING error, foreground_color, background_color.
```

BASIC Call

```
CALL COLOR ( error, foreground_color, background_color )
```

PASCAL Call

```
error := Color ( foreground_color, background_color );
```

PARAMETER	DESCRIPTION
INPUT	
foreground_color	Foreground color code and must be in the range -1 to 7 inclusive. See characteristics 1 and 2.
background_color	Background color code and must be in the range -1 to 7 inclusive. See characteristics 1 and 2.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
17	Out of volume space.

This procedure moves the graphics cursor within the current window with the aid of the arrow keys (↑ , ↓ , →, ←) for the screen-keyboard and via the stylus or the four-button cursor for the tablet. If the enabled work station is the screen-keyboard, the ASCII code of the first key pressed after the arrow keys and the graphics cursor position are returned. If the enabled work station is a logical work station on the tablet then the ASCII code returned (after having pressed the stylus or a button on the four-button cursor) is related to the logical work station.

COBOL Call

```
CALL "CURSORMOVE" USING error, shift_step, lastchar, x_cursor, y_cursor.
```

BASIC Call

```
CALL CURSORMOVE ( error, shift_step, lastchar$, x_cursor, y_cursor )
```

PASCAL Call

```
error := CursorMove ( shift_step, lastchar, x_cursor, y_cursor );
```

PARAMETER	DESCRIPTION
INPUT	
shift_step	Number of pixels the cursor will move each time an arrow key is entered. This number of pixels is the cursor step.
OUTPUT	
lastchar	ASCII code of the first character entered after the arrow keys or ASCII code related to the logical work station.
x_cursor	World coordinate along the x-axis of the graphics cursor position.
y_cursor	World coordinate along the y-axis of the graphics cursor position.
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
24	Line error.

Characteristics

1. Pressing the CONTROL key together with an arrow key moves the cursor by 1 pixel.

2. This procedure suspends the execution of the application program and waits for input from the keyboard.

3. If the enabled work station is a logical work station on the tablet then the ASCII code returned (after having pressed the stylus or a button on the four-button cursor) indicates the logical work station as follows:

ASCII CODE RETURNED	LOGICAL WORK STATION IDENTIFIER
128	Lowest value.
144	Next to lowest value.
160	Next to highest value.
176	Highest value.

COBOL Example

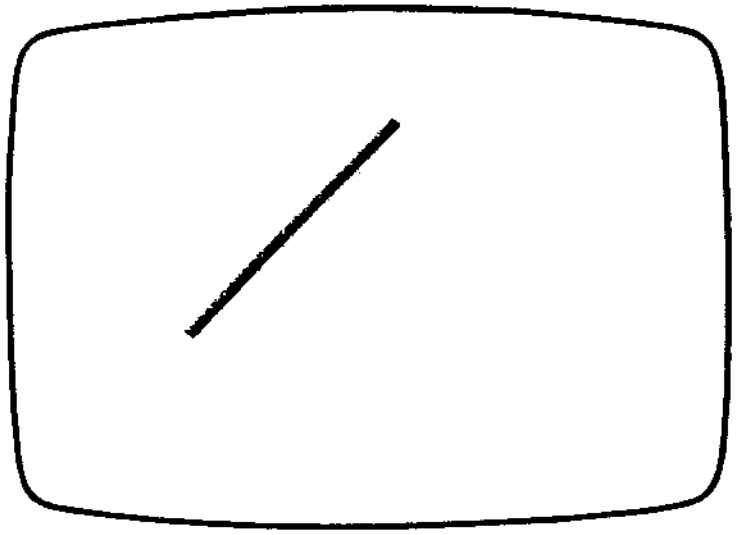
```
MOVE "0" TO LASTCHAR.
MOVE 0   TO ERROR1.
MOVE 10  TO SHIFT_STEP.
CALL "CURSORMOVE" USING ERROR1
                        SHIFT_STEP
                        LASTCHAR
                        X_CURSOR
                        Y_CURSOR.
```

Characteristic

This procedure has no effect on the PC emulating a work station with the GIOMR program.

COBOL Example

```
MOVE 15 TO WIDTH.
MOVE 0  TO ERROR1.
CALL "DEFWIDTH" USING ERROR1
                      WIDTH.
MOVE 150 TO X1.
MOVE 150 TO Y1.
MOVE 350 TO X2.
MOVE 350 TO Y2.
CALL "SEGMENT" USING ERROR1
                     X1
                     Y1
                     X2
                     Y2.
```



DELETEALLELE

This procedure deletes the specified elements from a segment.

COBOL Call

```
CALL "DELETEALLELE" USING error, segment_id, element_code.
```

BASIC Call

```
CALL DELETEALLELE ( error, segment_id, element_code )
```

PASCAL Call

```
error := DeleteAllEle ( segment_id, element_code );
```

PARAMETER	DESCRIPTION
INPUT	
segment_id	Segment to which the elements "element_code" belong. Value in the range -1 to 254; -1 means any segment.
element_code	Codes of the elements to be deleted. Value in the range -1 to 20; -1 means any element.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal function call.
10	Parameter out of range.
16	Invalid file operation.

This procedure stores in an array the contents of the whole current window or just that of a specified rectangular area.

COBOL Call

```
CALL "GET" USING error, getType, x1, y1, x2, y2, store, startIndex,
                 storeSize.
```

BASIC Call

```
CALL GET ( error, getType$, x1, y1, x2, y2, store, startIndex, storeSize )
```

PASCAL Call

```
error := Get ( getType, x1, y1, x2, y2, store, startIndex );
```

PARAMETER	DESCRIPTION
INPUT	
getType	Specifies whether the whole current window (0) or just a rectangular area (1) is to be stored. If this parameter is set to 0 then the coordinates (x1, y1) and (x2, y2) are ignored and the whole window is stored. If it is set to 1 then the coordinates (x1, y1) and (x2, y2) specify the ends of the diagonal of the rectangle to be stored.
x1	World coordinate along the x-axis of the first end of the diagonal which defines the rectangular area to be stored.
y1	World coordinate along the y-axis of the first end of the diagonal which defines the rectangular area to be stored.
x2	World coordinate along the x-axis of the second end of the diagonal which defines the rectangular area to be stored.
y2	World coordinate along the y-axis of the second end of the diagonal which defines the rectangular area to be stored.
startIndex	Ordinal value of the byte of "store" from where storing of the image starts. The first byte has ordinal value 0.
storeSize	Number of bytes to be used in the "store" array. In the PASCAL+ interface this parameter is not present because its value is obtained by subtracting "min" from "max", specified in the definition of the "store" array. See characteristic 2.

OUTPUT

error	Reply code. See the following table.
store	Array of bytes used to store the contents of the whole window or rectangular area.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
11	Invalid screen position.
19	Overflow.
20	Division by zero.

Characteristics

1. The coordinates (x1, y1) and (x2, y2) must fall within the current window.

2. The dimensions of the rectangular area to be stored is calculated as follows:

   "storeSize" = number_of_enabled_bit-planes * base/8 * height + 6

   where number_of_enabled_bit_planes (defined via the MASK procedure) is in the range 1 to 3 inclusive and base and height are measured in pixels.

3. With the M30 or M31 used as work stations this procedure cannot be used.

array is used for I/O and its dimensions must be such as to contain the whole image.
If "storeSize" is equal to 0 the image is directly stored on file. See characteristics 2 and 4.

OUTPUT

error	Reply code. See the following table.
store	Array of bytes used to store the contents of the whole window or rectangular area. See characteristic 3.

Reply Code

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
11	Invalid screen position.
13	Not enough memory.
15	File locked.
17	Out of volume space.
18	Device not ready.
19	Division by zero.
20	Overflow.

Characteristics

1. The coordinates (x1, y1) and (x2, y2) must fall within the current window.

2. The dimensions of the rectangular area to be stored is calculated as follows:

   "storeSize" = number_of_enabled_bit-planes * base/8 * height + 6

   where number_of_enabled_bit_planes (defined via the MASK procedure) is in the range 1 to 3 inclusive and base and height are measured in pixels.

3. When an image recorded on file is displayed, the array is used as an I/O buffer and therefore must be large enough to contain the whole file. After the execution of this procedure the "store" array may be used by subsequent PUT procedures (of the same image) without having to read the file each time.

4. In the PASCAL+ interface this parameter is not present because its value is obtained by subtracting "min" from "max", specified in the definition of the "filename" and "store" arrays.

5. This procedure cannot be used with the M30 or M31 used as work stations.

COBOL Example

```
MOVE 0    TO ERROR1.
MOVE 1    TO GETTYPE.
MOVE "/IPL/GRAPHICS/FILE" TO FILENAME.
MOVE 18   TO FNAMELENGTH.
MOVE 48           TO X1.
MOVE 48           TO Y1.
MOVE 250          TO X2.
MOVE 250          TO Y2.
MOVE 0            TO STARTINDEX.
MOVE 15156        TO STORESIZE.
CALL "GETANDSAVE" USING ERROR1
                        FILENAME
                        FNAMELENGTH
                        GETTYPE
                        X1
                        Y1
                        X2
                        Y2
                        STORE
                        STARTINDEX
                        STORESIZE.
```

This procedure carries out the hard copy of the whole screen or of the current window. If the color graphics screen and the printers PR 38C, PR 38BC, PR 1480 with color or DM 590 are used then a color hard copy of the screen is obtained.

COBOL Call

```
CALL "GRAPHICPRINT" USING error, screen, print_type, title, startIndex,
                          title_length.
```

BASIC Call

```
CALL GRAPHICPRINT ( error, screen, print_type$, title$, startIndex,
                    title_length )
```

PASCAL Call

```
error := GraphicPrint ( screen, print_type, title, startIndex,
                        title_length );
```

PARAMETER	DESCRIPTION
INPUT	
screen	Specifies whether the hard copy is to be of the whole screen (1) or of the current window (0).
print_type	Specifies whether a positive (0) or negative (1) hard copy is to be obtained. See the following characteristic.
title	Alphanumeric string containing a title to be printed before the hard copy.
startIndex	Ordinal value of the first character in "title" to be printed.
title_length	Title length, starting from "startIndex". It is an integer value in the range 0 to 60 inclusive. If this value is greater than zero then the characters contained in "title", starting from "startIndex", are printed before the hard copy.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
18	Device not ready.

Characteristics

1. If the "print-type" parameter has the value 0, the disactivated pixels will be printed. If it is 1, the activated pixels will be printed.

   If the parameter "print_type" is set to 1 and a color hard copy is possible then the complementary colors of those stored in the graphic bit-map are obtained, for example, green = 010 becomes magenta = 101.

2. If the number of pixels to be printed is greater than the number of printable dots, the image will be truncated at the edges.

3. For the graphics printers available see the OPENGPR procedure.

4. See the procedure SETGPRMODE for the selection of graphics modes relative to the printers PR 15, PR 38C, PR 1480 with colour, PR 1550 and PR 1580.

5. To obtain a hard copy on printers of type /B from a PC used as a work station it is necessary to have the modules HC8M and HCBC for hard copy on monochromatic or colour printers respectively.

6. When producing a hard copy on monochromatic printers from a colour video it must be noted that currently it is not possible to select the colours that will be printed. At present, if the "print_type" parameter is set to 0, only the colours which have the bit in their binary value that corresponds to the first activated bit-plane set to 0 are printed. If the "print_type" parameter is set to 1, only the colours which have this bit set to 1 are printed. In view of this, the user must decide in advance which bit-planes to activate and which colours to use to draw the image so that it can be printed.

   The following table, which shows the example of the "print_type" parameter set to 0 and all the bit-planes activated (parameter "newmask"=7 in the MASK procedure) illustrates the above problem.

COLOUR	BINARY VALUE			RESULT WITH ALL BIT-PLANES ACTIVATED AND "print_type"=0
	BIT-PLANE 3	BIT-PLANE 2	BIT-PLANE 1	
Black	0	0	0	Printed
Red	0	0	1	Not printed
Green	0	1	0	Printed
Yellow	0	1	1	Not printed
Blue	1	0	0	Printed
Magenta	1	0	1	Not printed
Cyan	1	1	0	Printed
White	1	1	1	Not printed

If the first bit-plane activated were bit-plane 2 (for example if instead of bit-plane 1 (for example, if the parameter "newmask"=6 in the MASK procedure), it can be seen from the preceding table that only the colours black, red, blue and magenta would be printed (bits equal to 0 in the bit-plane 2 column).

For the activation and disactivation of bit-planes and for the colours available as a result, see the MASK procedure.

7. This procedure cannot be used with the M30 or M31 used as work stations.

COBOL Example

```
MOVE 0       TO ERROR1.
MOVE 0       TO SCREEN1.
MOVE 0       TO PRINT_TYPE.
MOVE "TITLE" TO TITLE.
MOVE 0       TO STARTINDEX.
MOVE 5       TO TITLE_LENGTH.
CALL "GRAPHICPRINT" USING ERROR1
                          SCREEN1
                          PRINT_TYPE
                          TITLE
                          STARTINDEX
                          TITLE_LENGTH.
```

This procedure returns the screen type and controller used.

COBOL Call

```
CALL "INQDEVTYPE" USING dev_type.
```

BASIC Call

```
CALL INQDEVTYPE ( dev_type )
```

PASCAL Call

```
dev_type := InqDevType;
```

PARAMETER	DESCRIPTION

OUTPUT

dev_type — Integer value in the range -1 to 5 inclusive specifying the screen type and controller used. The values have the following meaning:

-1	Integrated alphanumeric screen.
0	Non-integrated screen (via RS232).
1	Integrated monochromatic graphics screen, monochromatic M30 or M31 used as work stations, with 32K controller (1 page of 640*400 pixels).
2	Integrated monochromatic graphics screen, monochromatic M30 or M31 used as work stations, with 128K controller (4 pages of 640*400 pixels).
3	Integrated graphics screen with colour or M30 with colour used as a work station.
4	Monochromatic PC used as a work station.
5	Colour PC used as a work station.

COBOL Example

```
CALL "INQDEVTYPE" USING DEV_TYPE.
CALL "SETALPHA".
DISPLAY "DEV_TYPE = " DEV_TYPE.
```

INQGRPRTYPE

This procedure returns the model of the printer used. It can be called only after having executed an OPENGRPR.

COBOL Call

```
CALL "INQGRPRTYPE" USING printerType.
```

BASIC Call

```
CALL INQGRPRTYPE ( printerType )
```

PASCAL Call

```
printerType := InqGrPrType;
```

PARAMETER	DESCRIPTION
OUTPUT	
printerType	Integer value in the range 0 to 7 inclusive or equal to 100, specifying the printer used. The values returned refer to the following printers: 0 printer not available or the OPENGRPR has not been executed 1 PR 1450 2 PR 1480, PR 1480 with colour 3 PR 2400 4 PR 15, PR 17, PR 1550, PR 1570 5 PR 19, PR 1470, PR 1490 6 PR 1580 7 PR 38, PR 38C 100 PR 11B, PR 13B, PR 15B, PR 17B, PR 19B, PR 38B, PR 38BC, DM 280, DM 290, DM 580, DM 590 connected to a PC.

COBOL Example

```
CALL "INQGRPRTYPE" USING PRINTERTYPE.
CALL "SETALPHA".
DISPLAY "PRINTER-TYPE = " PRINTERTYPE.
```

COBOL Example

```
           MOVE 100 TO X.
           MOVE 100 TO Y.
           MOVE 0   TO T-ERROR.
           CALL "MOVETO" USING T-ERROR
                               X
                               Y.
```

OPENGRPR

This procedure opens a work station printer or a system printer. The printers can be connected to the L1 MOS system or to PCs used as work stations.

COBOL Call

```
CALL "OPENGRPR" USING error, openMode, printerId, sysprtType.
```

BASIC Call

```
CALL OPENGRPR ( error, openMode$, sysprtType )
```

PASCAL Call

```
error := OpenGrPr ( openMode, printerId, sysprtType );
```

PARAMETER	DESCRIPTION
INPUT	
openMode	Integer value in the range 0 to 2 and specifies the graphics printer open mode. See the following characteristic.
sysprtType	System printer identifier (character "1",...,"8") and printer model (1,...,7). See "Characteristics".
OUTPUT	
error	Reply code. See the following table.
printerId	System identifier of the printer furnished by MOS and is significant only if the "openMode" parameter is set to 1.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
18	Device not ready.
22	Not graphic printer.

Characteristics

1. If the "openMode" parameter is set to 0, the printer is opened and connected as a work station printer; in this case the parameter "printerId" and the field containing the printer number in the "sysprtType" parameter are ignored.

   If the "openMode" parameter is set to 1, the printer must already be opened and connected and the parameter "printerId" contains the identifier supplied by MOS. In this case, the printer is a work station printer and the field containing the printer number in the "sysprtType" parameter is ignored. This method of opening must be used by those programs which open the work station in exclusive mode.

   If the "openMode" parameter is set to 2, the printer is opened as a system printer. In this case the "printerId" parameter is ignored.

2. A printer connected to a PC can only be used as a work station printer, and the "sysprtType" and "printerId" parameters are ignored.

3. The field containing the printer model in the "sysprtType" parameter is ignored if the connection allows the exchange of messages between the printer and the system (configuration parameters PRT_PRMODEF with the least significant byte equal to %00 and TANDEMCTLMO with the most significant byte equal to %01.

4. The following table shows the correspondences between the values of the field showing the printer model in the "sysprtType" parameter and the printers that can be used:

VALUE	PRINTERS
1	PR 1450
2	PR 1480 - 1480 with colour
3	PR 2400
4	PR 15 - PR 17 - PR 1550 - PR 1570
5	PR 19 - PR 1470
6	PR 1580
7	PR 38 - PR 38C

The printers listed above can be connected to the L1 MOS system or to a PC used as a work station via the OLIEMU program.

The following printers, on the other hand, are connected to the PC used as a work station via the WSELAN program: PR 11B, PR 13B, PR 15B, PR 17B, PR 19B, PR 38B, PR 38BC, DM 280, DM 290, DM 580, DM 590.

COBOL Example

```
MOVE 0   TO T-ERROR.
MOVE 2   TO OPENMODE.
MOVE "1" TO NUMBER1.
MOVE 3   TO MODEL.
CALL "OPENGRPR" USING T-ERROR
                      OPENMODE
                      PRINTERID
                      SYSPRTTYPE.
```

This procedure logically opens an input work station and, if necessary, it also opens the line connecting the PGU to the specified physical device on which the logical work station resides.

COBOL Call

```
CALL "OPENGRWS" USING error, wsId, wsType, lnDescLength, lnDescriptor,
                      wsDescLength, wsDescriptor.
```

BASIC Call

```
CALL OPENGRWS ( error, wsId, wsType, lnDescLength, lnDescriptor$,
                wsDescLength, wsDescriptor$ )
```

PASCAL Call

```
error := OpenGrWs ( wsId, wsType, lnDescLength, lnDescriptor, wsDescLength,
                    wsDescriptor );
```

PARAMETER	DESCRIPTION
INPUT	
wsId	Positive integer value specifying the identifier to be assigned to the logical work station.
wsType	Specifies the work station type. It must be set to 0, specifying an input work station (tablet).
lnDescLength	Number of characters to be considered in the "lnDescriptor" array. The minimum value is 5.
lnDescriptor	Array containing all the information necessary for opening the line which connects the system to the device on which the logical work station resides. See characteristic 1.
wsDescLength	Number of characters to be considered in the "wsDescriptor" array.
wsDescriptor	Array containing all the information about the model of the device on which the logical work station resides. See characteristic 2.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
10	Parameter out of range.
23	Line busy.
24	Line error.
25	Too many opened work stations.
27	Work station already opened or activated.

Characteristics

1. The array "lnDescriptor" has the following structure:

   - The first five characters specify the line to which the device is connected. At the moment the only connection method available is via RS232. The first two characters must be PT and the last three must be a value in the range from 001 to 113, which specifies the channel used.

   - The next seventeen characters specify the line transmission characteristics. For a detailed description of each character see the manual "RS232/CL Interface - Programmer Guide", OpenLn primitive. The default sequence is : '011010222<<100111' which may be specified by the character 'D'.

2. The array "wsDescriptor" has the following structure:

   - The first three characters specify the input device type. The only value allowed is 'TAB', meaning tablet, which is the default value and may be specified by the character 'D'.

   - The next twelve characters specify the characteristics of the tablet. The first five characters are the maximum coordinate along the x axis to be returned by the tablet, the next five characters are the maximum coordinate along the y axis to be returned by the tablet (the default value for all 10 characters is specified by the character 'D') and the last two characters give the tablet transmission mode which must be '01' (the default value for these two characters is specified by the character 'D'). The latter value indicates a transmission mode having the sequence XXXX, YYYY, F, CRLF where:

X is an ASCII BCD digit of the x coordinate
, is an ASCII comma
Y is an ASCII BCD digit of the y coordinate
, is an ASCII comma
F is the ASCII character returned by the function key pressed after having chosen the point whose coordinates are to be returned. It may be:
0 no key
1 key 0 of the four button cursor or the stylus
2 key 1 of the four button cursor
4 key 2 of the four button cursor
8 key 3 of the four button cursor

- The default value of this array is 'TAB029710297101'.

3. The parameters "lnDescriptor" and "wsDescriptor" set via this procedure must be consistent with the microswitch settings on the tablet. The default values given by the PGU are consistent with the following microswitch settings on the CALCOMP 2000 Series Digitizer:

Switch 1 = 000000000
Switch 2 = 0000CC00C
Switch 3 = 00000000C

where 0 means open and C means closed.

4. The logical work stations on the tablet may physically be separate or superimposed. In order to establish their size and position the procedures VIEW, ACTIVGRWS, DEACTIVGRWS must be used. For example, let's suppose that two logical work stations (identified by the values 1 and 2) are to be opened on the tablet. Suppose that work station 1 is to be identified by the coordinates (0,0), (2000,0), (0,2971), (2000,2971) and work station 2 by the coordinates (1500,0), (2971,0), (1500,2971), (2971,2971). The following procedures are necessary:

- DEACTIVGRWS to disable work station 0 (if enabled)
- OPENGRWS to open work station 1
- OPENGRWS to open work station 2
- ACTIVGRWS to enable work station 1
- VIEW to enable the area on work station 1 identified by the coordinates specified above
- DEACTIVGRWS to disable work station 1
- ACTIVGRWS to enable work station 2
- VIEW to enable the area on work station 2 identified by the coordinates specified above.

5. It is not possible to connect the tablet to the M30, M31 or PC used as work stations.

OPENIMAGE

Opens the specified file (image file) on which all successive graphics procedures will be recorded.

COBOL Call

```
CALL "OPENIMAGE" USING error, filename, fnlength, 0_mode.
```

BASIC Call

```
CALL OPENIMAGE ( error, filename$, fnlength, 0_mode$ )
```

PASCAL Call

```
error := OpenImage ( filename, 0_mode );
```

PARAMETER	DESCRIPTION
INPUT	
filename	Pathname of the image file to be opened. It is user defined and must be in the range 1 to 60 inclusive.
fnlength	Pathname length. See characteristic 5.
0_mode	Specifies whether the image file is to be recorded from the beginning (0) or in append mode (1). The append mode allows a previously recorded image to be completed.
OUTPUT	
error	Reply code. See the following table.

This procedure initialises the PGU variables and the bit-map. The input/output work station is opened and enabled. The screen is set according to the user's requirements and the existence of FONT files is tested. If there is no FONT file this procedure returns a warning.

COBOL Call

```
CALL  "OPENPGU"  USING error, openPgu.
```

BASIC Call

```
CALL OPENPGU ( error, openPgu$ )
```

PASCAL Call

```
error := OpenPGU ( openPgu );
```

PARAMETER	DESCRIPTION
INPUT	
openPgu	The values which can be assigned to this parameter are one of the following: 0 the PGU procedures are used for alphanumeric I/O and both text and bit-map may be displayed together. 1 procedures other than those of the PGU are used for alphanumeric I/O and both text and bit-map are displayed together. 2 the PGU procedures are used for alphanumeric I/O. Text and bit-map are displayed separately. 3 procedures other than those of the PGU are used for alphanumeric I/O. Text and bit-map are displayed separately. 4 Batch mode execution. See characteristic 1.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
-2	There are no font files.
-1	The system is not graphic.
0	Correct procedure execution.
1	System error.
2	PGU not found.
3	PGU already open.
4	PGU can not be loaded.
5	PGU can not be started.
13	Not enough memory.

Characteristics

1. If the values 0 or 2 are chosen then also text can use the window facility and "alphanumeric text" is automatically assumed.

   If the values 1 or 3 are chosen then the user must choose whether to use "alphanumeric text" or "graphics text".

   If the value 4 is chosen all the elements of an image file are executed in batch mode by using the Shell command BM. This mode is useful for executing image files which require a long execution time and the graphic work station is left free for other processes.

2. With an integrated work station with colour, or an M30 with colour or monochromatic PC used as work stations, the graphics bit-map and alphanumeric text are always displayed together.

3. If using an integrated monochromatic work station, or a monochromatic M30, an M31 or a PC with colour, used as work stations, and if the "openPgu" parameter has been set to 2 or 3 then in order to display the graphics bit-map (or the alphanumeric text) it is necessary to set (or reset) the hide-alpha attribute (attribute 96) in all the character cells. This is automatically done on the whole current window via the procedure REF.

COBOL Example

```
MOVE 1 TO OPENPGU.
MOVE 0 TO T-ERROR.
CALL "OPENPGU" USING T-ERROR
                     OPENPGU.
```

This procedure selects one of the four bit-map pages.

COBOL Call

```
CALL "PAGE" USING error, page_number.
```

BASIC Call

```
CALL PAGE ( error, page_number )
```

PASCAL Call

```
error := Page ( page_number );
```

PARAMETER	DESCRIPTION
INPUT	
page_number	Page to be selected. This is an integer value in the range 1 to 4 (see the PAN procedure).
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
10	Parameter out of range.

Characteristics

1. This procedure has effect only with integrated monochormatic work stations, and with the M31 and the monochromatic M30 used as work stations with a 128K byte bit-map memory (4 pages of 32k each).

2. All the graphics procedures executed after PAGE will produce output to be stored in the selected bit-map page. This does not necessarily correspond to the page displayed on the screen, which is selected via the PAN procedure.

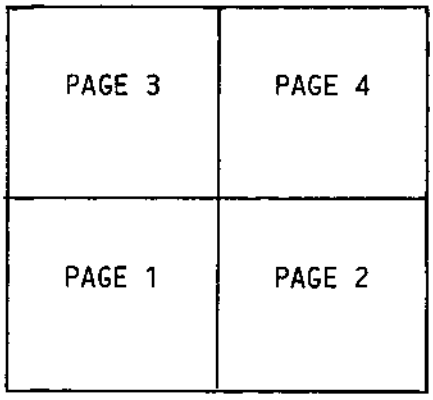
COBOL Example

```
MOVE 3 TO PAGE_NUMBER.
MOVE 0 TO T-ERROR.
CALL "PAGE" USING T-ERROR
                  PAGE_NUMBER.
```

Characteristics

1. This procedure is effective only on integrated monochromatic work stations, and on M31s and monochromatic M30s used as work stations that have a 128K byte bit-map memory (4 pages of 32K each), and on integrated work stations with colour, and M30s with colour used as work stations.

2. With integrated monochromatic work stations, M31s and monochromatic M30s used as work stations, the hardware device coordinates (x-pan,y-pan) can only have the values (0,0), (640,0), (0,400) and (640,400), corresponding to the lower left-hand corners of the four bit-map pages:



   See the PAGE procedure.

3. With integrated work stations with colour or M30s with colour used as work stations, it is possible to display any bit-map area of 640 x 400 pixels starting from the point specified by this procedure, considered as the origin.

COBOL Example

```
MOVE 0 TO T-ERROR.
MOVE 0 TO X_PAN.
MOVE 0 TO Y_PAN.
CALL "PAN" USING T-ERROR
                 X_PAN
                 Y_PAN.
```

POINTCOLOR

This procedure returns the color of the pixel nearest to the specified point.

COBOL Call

```
CALL "POINTCOLOR" USING x, y, color.
```

BASIC Call

```
CALL POINTCOLOR ( x, y, color )
```

PASCAL Call

```
color := PointColor ( x, y );
```

PARAMETER	DESCRIPTION
INPUT	
x	World coordinate along the x-axis of the pixel whose color value is to be known.
y	World coordinate along the y-axis of the pixel whose color value is to be known.
OUTPUT	
color	Color code of the pixel which is nearest to the point (x,y).

Characteristics

1. If the world coordinate (x,y) is outside the current window the value 0 is returned.
2. This procedure cannot be used with the M30 or M31 used as work stations.

COBOL Example

```
MOVE 200 TO X.
MOVE 200 TO Y.
CALL "POINTCOLOR" USING X
                        Y
                        COLOR.
```

2. This procedure has no effect on integrated work stations with colour, M30s with colour and monochromatic Pcs used as work stations, since it is not possible to display the graphics bit-map and alphanumeric text separately on these.

COBOL Example

```
MOVE 0 TO SCREEN_T.
CALL "REF" USING SCREEN_T.
```

ROTATE

This procedure rotates the graphic axes of the world coordinate system.

COBOL Call

```
CALL "ROTATE" USING error, angle_rotation.
```

BASIC Call

```
CALL ROTATE ( error, angle_rotation )
```

PASCAL Call

```
error := Rotate ( angle_rotation );
```

PARAMETER	DESCRIPTION
INPUT	
angle_rotation	Rotation angle in radians, measured from the the positive x-axis to the positive y-axis. The value must be in the range -2π and 2π.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
17	Out of volume space.
19	Division by zero.
20	Overflow.

2. An example for the monochromatic configuration may be 5A meaning that the character is displayed in reverse and blinks, with right line and underline attributes. The same value for the color configuration means that the character is displayed in black and blinks on a yellow background, with right line and underline attributes.

3. On a monochromatic PC used as a work station the alphanumeric visual attributes are ignored.

COBOL Example

```
MOVE 0       TO T-ERROR.
MOVE ""0""   TO OLD_ATR.
MOVE ""32""  TO NEW_ATR.
CALL "SETATR" USING T-ERROR
                    OLD_ATR
                    NEW_ATR.
```

SETGRPRMODE

This procedure sets the functioning mode for the graphic printers PR 15, PR 38C, PR 1480 with colour, PR 1550 and PR 1580. The OPENGRPR procedure must have been previously executed.

COBOL Call

```
CALL "SETGRPRMODE" USING error, prMode.
```

BASIC Call

```
CALL SETGRPRMODE ( error, prMode() )
```

PASCAL Call

```
error := SetGrPrMode ( prMode );
```

PARAMETER	DESCRIPTION
INPUT	
prMode	Five integers array in the range -1 to 1. Each integer value refers to a specific printer and to a particular functioning mode. See the following characteristic.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
10	Parameter out of range.

Characteristic

The first integer value refers to the printing resolution on the PR 15 and PR 1550 printers and specifies the maximum width of the bit-map area to be printed. It may be set to one of the following:

-1 : the current resolution is not changed

 0 : maximum width set to 640 pixels (initial value)

 1 : maximum width set to 560 pixels. The distance between the points printed horizontally and vertically coincides and thus the image to be printed is not altered.
If the image to be printed is wider than 560 pixels on the graphic bit-map then it will be clipped.

The second integer value refers to the PR 38C and the PR 1480 with colour and specifies whether a colour or a monochromatic print is to be obtained. It may be set to one of the following:

-1 : the current print mode is not changed

 0 : colour print (initial value)

 1 : monochromatic print.

The third integer value refers to the PR 1580 and specifies whether a normal or a high density print is wanted. It may be set to one of the following:

-1 : the current density is not changed

 0 : normal density print (initial value)

 1 : high density print.

The fourth integer value refers to the PR 1580 and specifies whether a normal or a zoomed print is to be obtained. It may be set to one of the following:

-1 : the current mode is not changed

 0 : normal print (initial value)

 1 : zoomed print (the image size is doubled).

The fifth integer value must be set to 0 and is the closing element of the array.

SETINSIDE

This procedure sets the filling mode for the geometric figures drawn via the ARC (parameter "option" set to 3), BOXFILL, CIRCLEFILL, and POLYGON procedures.

COBOL Call

```
CALL "SETINSIDE" USING error, inside_mode.
```

BASIC Call

```
CALL SETINSIDE ( error, inside_mode )
```

PASCAL Call

```
error := SetInside ( inside_mode );
```

PARAMETER	DESCRIPTION
INPUT	
inside_mode	Integer value specifying whether the geometric figure is to be filled with the current foreground color (0) or with a pattern (1). See the following characteristic.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
10	Parameter out of range.

Characteristic

If the parameter "inside_mode" is set to 0 then the current foreground color (either the initial one or the one set via the COLOR procedure) is used for filling the geometric figure. If it is set to 1 then the current fill pattern (either the initial one or the one set via the PAINTSHAPE or PAINTCSHAPE procedures) is used.

COBOL Example

```
MOVE 0 TO T-ERROR.
MOVE 0 TO INSIDE_MODE.
CALL "SETINSIDE" USING T-ERROR
                       INSIDE_MODE.
```

STRDIRECTION

This procedure changes the value of the angle formed by the string and the x-axis. It acts on the current window.

COBOL Call

```
CALL "STRDIRECTION" USING error, angle.
```

BASIC Call

```
CALL STRDIRECTION ( error, angle )
```

PASCAL Call

```
error := StrDirection ( angle );
```

PARAMETER	DESCRIPTION
INPUT	
angle	Angle, measured in radians from the positive x-axis to the positive y-axis, between the x-axis and the direction of the string. Its value must be in the range -2π and 2π. The initial value is 0, meaning that the string will be parallel to the x-axis.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
17	Out of volume space.

Characteristics

1. The available fonts are:

"fontnumber"	FONT FILE	FONT TYPE
0	FONT0	string precision
1	FONT1	simple block letters
2	FONT2	double block letters
3	FONT3	complex block letters
4	FONT4	complex italics
5	FONT5	cyrillic
6	FONT6	greek
7	FONT7	gothic
8	FONT8	user-defined

   The initial value is 1.

2. The user can select files with different descriptions (created via the CHANGEFONT utility, described in Appendix A) connecting them to the global names FONT0, FONT1, ..., FONT8 or directly substituting the files FONT0,...,FONT8 with his own. With integrated work stations, M30s, M31s and PCs with the GIOMR program, used as work stations, the files FONT0,...,FONT8 reside on the L1 MOS system in the directory /IPL/DPC/GRAPHICS, while with PCs used as work stations with the program GIOM, the files reside in the directory \GRAPHICS on the PC.

3. If "fontnumber" is set to 0 then the file FONT0 is used, containing a character set with the same precision as the alphanumeric characters. In this case, in order not to deform the characters, the current value of "ratio" (see procedure STRMAGNIFICA) is ignored (the initial value is used) and character magnification, spacing and direction are applied on the origins of each character.

   If routines for alphanumeric output are used it is not possible to print graphics and text together. Using FONT0 and the STRING procedure, besides printing graphics output it is possible to obtain a hard copy of graphics text with the same precision as the alphanumeric text.

4. If the specified font file does not exist or may not be read (errors 12, 13, 15, and 18) this procedure does not return an error message. It will be signalled by the String procedure.

COBOL Example

```
MOVE 0 TO FONTNUMBER.
MOVE 0 TO T-ERROR.
CALL "STRFONT" USING T-ERROR
                     FONTNUMBER.
IF T-ERROR NOT = 0 CALL "SETALPHA"
   DISPLAY "ERROR CODE = " T-ERROR.
MOVE 0           TO T-ERROR.
MOVE 8           TO LENGTH_STRING.
MOVE 0           TO STARTINDEX.
MOVE "OLIVETTI" TO STR.
MOVE 100         TO X.
MOVE 40          TO Y.
CALL "STRING" USING T-ERROR
                    LENGTH_STRING
                    STARTINDEX
                    STR
                    X
                    Y.
IF T-ERROR NOT = 0 CALL "SETALPHA"
   DISPLAY " ERROR CODE = " T-ERROR.
MOVE 1 TO FONTNUMBER.
MOVE 0 TO T-ERROR.
CALL "STRFONT" USING T-ERROR
                     FONTNUMBER.
IF T-ERROR NOT = 0 CALL "SETALPHA"
   DISPLAY " ERROR CODE = " T-ERROR.
MOVE 0           TO T-ERROR.
MOVE 8           TO LENGTH_STRING.
MOVE 0           TO STARTINDEX.
MOVE "OLIVETTI" TO STR.
MOVE 150         TO X.
MOVE 80          TO Y.
CALL "STRING" USING T-ERROR
                    LENGTH_STRING
                    STARTINDEX
                    STR
                    X
                    Y.
IF T-ERROR NOT = 0 CALL "SETALPHA"
   DISPLAY " ERROR CODE = " T-ERROR.
```

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.
12	Font not found.
13	Not enough memory.
15	File locked.
17	Out of volume space.
18	Device not ready.
19	Division by zero.
20	Overflow.

Characteristics

1. The character font, height, width, spacing, and direction are the current ones chosen via the procedures STRFONT, STRMAGNIFICA, STRSPACING, and STRDIRECTION. If these have not been executed the initial values are assumed.

2. The current value of ratio is ignored if the font is set to FONT0. In this case, each character is contained in a cell 8 pixels wide and 10 pixels high.

3. Each character is contained in a rectangular cell having:

   - width = W * magnification (W is equal to the width of each character; this value is contained in the description of each character and depends on its shape)

   - height = 10 * magnification * ratio.

4. The spacing between two characters next to each other is:
   spacing = W * magnification * spacing

5. If a character does not fit within the current window then it will not be displayed. This means that there will be no clipping operation on the single character.

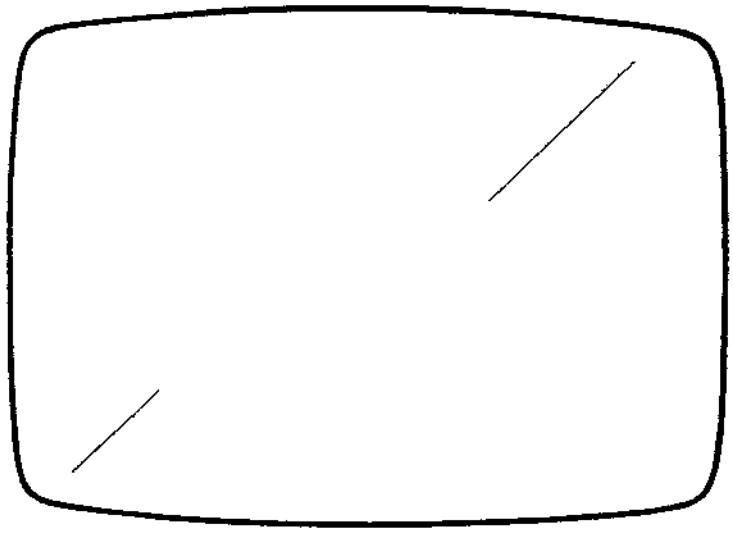
6. With a PC used as a work station, reply code 12 is not returned.

COBOL Example

```
MOVE 0           TO T-ERROR.
MOVE 8           TO LENGTH_STRING.
MOVE 0           TO STARTINDEX.
MOVE "OLIVETTI" TO STR.
MOVE 200         TO X.
MOVE 50          TO Y.
CALL "STRING" USING T-ERROR
                    LENGTH_STRING
                    STARTINDEX
                    STR
                    X
                    Y.
```

COBOL Example

```
MOVE 0   TO X1.
MOVE 0   TO Y1.
MOVE 50  TO X2.
MOVE 50  TO Y2.
CALL "SEGMENT" USING T-ERROR
                     X1
                     Y1
                     X2
                     Y2.
MOVE 300 TO SHIFT_HORIZONTAL.
MOVE 200 TO SHIFT_VERTICAL.
MOVE 0   TO T-ERROR.
CALL "TRANSLATE" USING T-ERROR
                       SHIFT_HORIZONTAL
                       SHIFT_VERTICAL.
MOVE 0   TO X1.
MOVE 0   TO Y1.
MOVE 100 TO X2.
MOVE 100 TO Y2.
CALL "SEGMENT" USING T-ERROR
                     X1
                     Y1
                     X2
                     Y2.
```



VIEW

This procedure enables the specified area on the tablet and screen for graphics input/output. It acts on all active work stations.

COBOL Call

```
CALL "VIEW" USING error, xmin, xmax, ymin, ymax.
```

BASIC Call

```
CALL VIEW ( error, xmin, xmax, ymin, ymax )
```

PASCAL Call

```
error := View ( xmin, xmax, ymin, ymax );
```

PARAMETER	DESCRIPTION
INPUT	
xmin	Device coordinate along the x-axis of the lower left hand corner of the viewport. If this parameter is set to -1 the limits of the viewport are: 0 <= x <= 1279 and 0 <= y <= 799 for integrated work stations with colour and for M30s with colour used as work stations. 0 <= x <= 639 and 0 < =y <= 399 for integrated monochromatic work stations, monochromatic M30s, for M31s and for PCs used as work stations. 0 < =x <= 2971 and 0 < =y <= 2971 for the tablet.
xmax	Device coordinate along the x-axis of the upper right hand corner of the viewport.
ymin	Device coordinate along the y-axis of the lower left hand corner of the viewport.
ymax	Device coordinate along the y-axis of the upper right hand corner of the viewport.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.

Characteristics

1. The origin of the world coordinate system is placed in (xmin,ymin) and the units of measure are set to 1 pixel along both axes.

2. The whole screen is enabled for alphanumeric output.

3. If the window structure has been created this procedure logically destroys it but the contents of the screen is not cleared. The PGU enters in the viewport state and an error is returned if the procedures CREATEWINDOW, CLOSEWINDOW, CLOSEALLWIND and SELECTWINDOW are called.

4. It is possible to exit the viewport state by executing the procedures BOUNDS or CLEAR.

5. The limits of the viewport on the screen must be within the physical limits of the bit-map, and thus:

   - 0 <= x <= 1279 and 0 <= y <= 799 for integrated work stations with colour and for M30s with colour used as work stations

   - 0 <= x <= 639 and 0 <= y <= 399 for integrated monochromatic work stations, for monochromatic M30s, for M31s and for PCs used as work stations.

   The limits of the viewport on the tablet must be within the values:
   0 <= x <= 2971 and 0 <= y <= 2971

6. For the screen, the minimum height and width for a viewport is 10 pixels x 8 pixels.

COBOL Example

```
MOVE 0   TO T-ERROR.
MOVE 0   TO XMIN.
MOVE 0   TO YMIN.
MOVE 639 TO XMAX.
MOVE 383 TO YMAX.
CALL "VIEW"  USING T-ERROR
                   XMIN
                   XMAX
                   YMIN
                   YMAX.
```

This procedure zooms the display according to the specified zoom level, starting from the lower left-hand corner.

COBOL Call

```
CALL "ZOOM" USING error, zlevel.
```

BASIC Call

```
CALL ZOOM ( error, zlevel )
```

PASCAL Call

```
error := Zoom ( zlevel );
```

PARAMETER	DESCRIPTION
INPUT	
zlevel	Specifies the zoom level. It is an integer in the range 0 to 15. The value 0 resets the image as it was before being zoomed, the value 1 doubles it, the value 2 triples it, etc.
OUTPUT	
error	Reply code. See the following table.

Reply Codes

The following reply codes are returned by the system:

NUMERIC REPLY VALUES	MEANING
0	Correct procedure execution.
6	Illegal procedure call.
10	Parameter out of range.

Characteristics

1. If the zoom level 0 is applied to an image that has not previously been zoomed, this procedure has no effect.

2. This procedure has no effect on integrated monochromatic work stations, or on monochromatic M30s, M31s and PCs used as work stations.

COBOL Example

```
MOVE 8 TO ZLEVEL.
MOVE 0 TO T-ERROR.
CALL "ZOOM"  USING  T-ERROR
                    ZLEVEL.
```

6. FORTRAN INTERFACE

This chapter describes the PGU FORTRAN interface functions in alphabetical order. These functions execute operations of the following types (see Tab. 1-2 for a detailed list):

- Control
- Output
- Attributes
- Window handling
- Cursor handling
- Coordinate scaling
- Image file handling
- Element handling
- Segment handling
- Graphic hard copy
- Inquiry
- Work station handling
- Special
- Alphanumeric I/O

Symbolic names have been assigned to many parameters of the FORTRAN functions. The user may choose to use these symbolic names or their integer values. In the function description both are furnished and they are separated by a slash, for example CHORD/1. The function names and the symbolic names may either be entered in lower case or in upper case letters. The real values to be assigned to the parameters are single-precision values unless otherwise specified.

The following information is given for each function:

- Its name
- A short description

- The function syntax
- A detailed description of each parameter
- Some possible characteristics
- An example of function call with the associated hard copy (when possible)

The user can decide, at the start of each program, whether to display the alphanumeric I/O and/or the contents of the graphic bitmap by entering

program name '-mode'

where mode has one of the following values:

- 0 : "together"
  The text and graphics are displayed together.
- 2 : "separate"
  The text and graphics are displayed in mutual exclusion.
- 4 : "batch" mode is used
  All the elements of an image file are executed in batch mode by using the Shell command BM. This mode is useful for executing image files which require a long execution time and the graphic work station is left free for other processes.

The default value is 0.

With an integrated work station with colour, an M30 with colour, or a monochromatic PC used as a work station, the graphics bit map and the alphanumeric text are always displayed together.

When the graphics session is opened, the work station 0 (the screen/keyboard) is opened and enabled.

Each FORTRAN function, program or subroutine must include the file

'/IPL/DPC/FORT/INCL/GRAPHIC.H'

which contains the definition of each graphics function and of all the parameters of the PGU graphics package.

For each function, the system returns a numeric value in a variable; see Appendix B for the meaning.

Errata Corrige: the symbolic parameter name MAGENTA is to be substituted with MAGENT in the following text.

These functions disable/enable the text cursor.

ACOFF ()
ACON ()

ARC

This function draws an arc, a segment, a sector or a filled sector of a circle or ellipse. The sector is filled with the specified colour (if the INSIDE function has not been executed) or as specified in the INSIDE function.

ARC (x , y , r , initial-angle , final-angle , choice , colour , axis-ratio , action-verb)

---

where:

x,y are real numeric expressions whose values give the coordinates of the centre of the circle or ellipse.

r is a real numeric expression which must have a positive value. This value gives the radius of the circle or the horizontal semi-axis of the ellipse.

initial-angle is a real numeric expression whose value must be between -2π and $+2\pi$. This value defines the initial angle of the arc, measured in radians, from the positive x-axis to the positive y-axis.

final-angle is a real numeric expression whose value must be between -2π and $+2\pi$. This value defines the final angle of the arc, measured in radians, from the positive x-axis to the positive y-axis.

choice is an integer numeric expression which allows an arc, a segment, a sector or a filled sector to be drawn. It can have the following values:

- DEFARC/0 : An arc is drawn.
- CHORD/1 : A segment is drawn.
- SECTOR/2 : A sector is drawn.
- SECFIL/3 : A filled sector is drawn.

colour is an integer numeric expression which selects the colour in which to draw or fill the arcs, segments and sectors. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off
  . REV/-1 : pixel on if previously off and vice versa.

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with the REV/-1 value.

axis-ratio is a real numeric expression which must have a positive value. This value defines the ratio between the vertical and horizontal axes of an ellipse. If "axis-ratio" has a value of 1, an arc, segment or sector is drawn (if the same unit of measure is used on both the x and y axes).

action-verb is an integer value which identifies the operation to be executed on each pixel in the figure. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

Each of these values defines the operation to be executed on each pixel of the curve.
They have the following meaning:

- NOACT/0 : This has no effect.

- SXOR/1 : The arcs, segments and sectors are drawn in the colour resulting from the XOR logical operation between the binary values of the current foreground colour code and the colour code of each pixel.

- SAND/2 : The arcs, segments and sectors are drawn in the colour resulting from the AND logical operation between the binary values of the current foreground colour code and the colour code of each pixel.

- SNOT/3 : The arcs, segments and sectors are drawn in the complementary colour of that specified by the "colour" parameter.

- SOR/4 : The arcs, segments and sectors are drawn in the colour resulting from the OR logical operation between the binary values of the current foreground colour code and the colour code of each pixel.

- PIXRES/5 : The arcs, segments and sectors are drawn in the current background colour.

- PIXSET/0 : The arcs, segments and sectors are drawn in the colour specified by the "colour" parameter.

Characteristics

1. The current graphic position is updated to the (x,y) point, which is the centre of the circle or ellipse whose arc, segment or sector is to be drawn.

2. The "r" and "axis-ratio" parameters must be selected so that the semi-axes are longer than, or equal to a pixel.

This function draws a rectangle whose diagonal is the segment joining the coordinates (x1,y1) and (x2,y2).

BOX (step , x1 , y1 , step , x2 , y2 , colour , action-verb , fill)

---

where:

step is an integer value which may be one of the following:

- RELAT/1 : the coordinates are considered relative to the current graphic position.
- ABSOL/0 : the coordinates are considered absolute (for the current window).

x1,y1 are real numeric expressions whose values specify the coordinates of an extreme of the segment.

x2,y2 are real numeric expressions whose values specify the coordinates of the other extreme of the segment.

colour is an integer numeric expression which specifies the colour with which to draw or fill the rectangle. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off
  . REV/-1 : pixel on if previously off, and vice versa.

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with the REV/-1 value.

action-verb is an integer numeric expression which identifies the operation to be executed on each pixel in the rectangle. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameter of the ARC function for the meaning of these values.

fill is an integer numeric expression which indicates whether the inside of the rectangle must be filled or not. The following values are allowed:

EMPTY/0 : The rectangle is not filled.
FILL/1 : The rectangle is filled with the colour specified by the "colour" parameter (if the INSIDE function has not been executed) or as specified in the INSIDE function.

Characteristic

The current graphic position is updated according to the second extreme specified.

Example

```
include '/IPL/DPC/FORT/INCL/GRAPHIC.H'
ires1 = box(0,0.,0.,0,250.,190.,1,0,1)
ires2 = box(0,250.,190.,0,500.,380.,1,0,1)
ires3 = box(0,185.,140.,0,320.,245.,1,0,0)
ires4 = box(0,320.,140.,0,500.,0.,1,0,1)
ires5 = box(1,-500.,380.,1,185.,-135.,1,0,1)
end
```

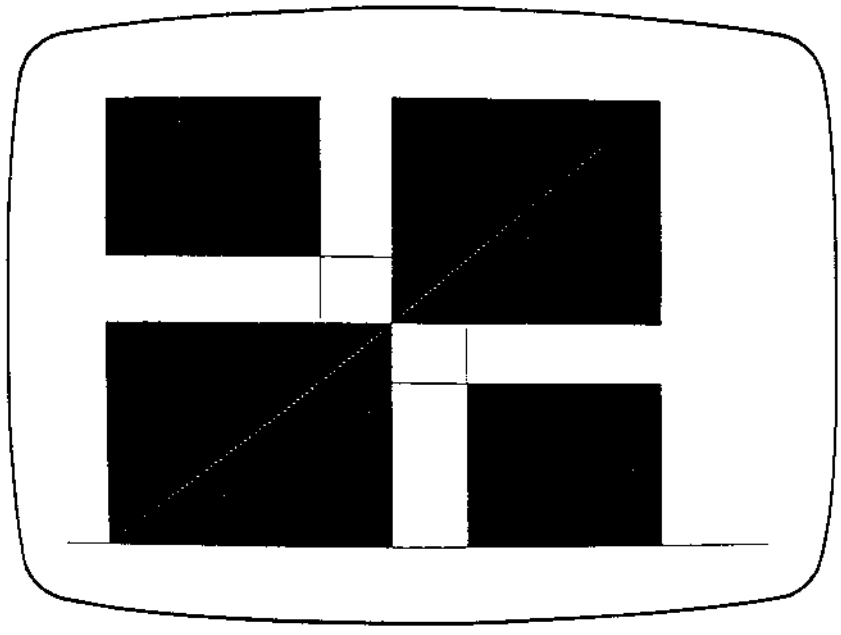


Fig. BOX-1 Rectangles

This function disables the buffer.

BRESET ()

Characteristic

After BRESET has been executed, each graphic command is executed and the result is immediately displayed. When it has been executed, BRESET remains active until the BSET function is executed.

Note

See the BFLUSH and BSET functions for buffer use.

This function draws a circle or an ellipse and fills it with the specified colour (if the INSIDE function has not been executed) or as specified in the INSIDE function.

CIRCLF (x , y , r , colour , axis-ratio , action-verb)

---

where:

x,y are real numeric expressions whose values give the coordinates of the centre of the circle or ellipse.

r is a real numeric expression which must have a positive value. This value gives the radius of the circle or the horizontal semi-axis of the ellipse.

colour is an integer numeric expression which specifies the colour in which to draw or fill the circle or ellipse. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off
  . REV/-1 : pixel on if previously off, and vice versa.

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with the REV/-1 value.

axis-ratio is a real numeric expression which must have a positive value. This value defines the ratio between the vertical and horizontal axes of an ellipse. If "axis-ratio" has the value 1, a circle is drawn (if the same unit of measurement has been selected on both the x and y axes).

action-verb is an integer numeric expression which identifies the operation to be executed on each pixel of the circle or ellipse. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameter of the ARC function for the meaning of these values.

Characteristics

1. The current graphic position is updated to the (x,y) point, which is the centre of the circle or ellipse.

2. The "r" and "axis-ratio" parameters must be selected so that the semi-axes are longer than, or equal to a pixel.

This function deletes the screen contents and restores the initial conditions (see Chapter 1).

CLEAR ()

Characteristic

The effects of BOUNDS and BRESET are not disabled when CLEAR is executed.

CLGRWS

This function closes the specified work station.

CLGRWS (wsid)

---

where:

wsid is an integer numeric expression which must have a positive value. It identifies the work station to be closed.

Characteristics

1. The identifier is set via the OPGRWS function.

2. The work station is disabled via this function (if it has not been disabled via the DAGRWS function) and closed. Its identifier is no longer recognised and may be used to open another work station.

3. The line connecting the PGU to the physical device containing the logical work station is closed if there is no other logical work station opened on the same physical device.

4. Work station 0 (the screen/keyboard) can not be closed.

This function disables the specified work station.

DAGRWS (wsid)

---

where:

wsid is an integer numeric expression which must have a positive value. It identifies the work station to be disabled.

Characteristic

The identifier is set via the OPGRWS function.

This function enables/disables the graphic cursor in the current window, specifying its position and shape.

DEFCUR (x , y , type , shape)

---

where:

x,y are real numeric expressions whose values identify the coordinates of the position where the cursor is to be displayed.

type is an integer numeric expression which can have one of the following meanings:

- OFF/0 : Cursor OFF.
- SW/1 : User cursor ON (it appears in the form defined for "shape").
- CROSS/2 : Cross cursor ON (the size is given in "shape").
- HW/3 : Hardware cursor ON.
- RUBBER/4 : Rubber band cursor ON.
- RECT/5 : Rubber rectangle cursor ON.
- SQUARE/6 : Square cursor ON (it appears in the form defined for "shape").
- RCROSS/7 : Rubber band cross cursor ON.
- CRECT/8 : Rubber rectangle cross cursor ON.

shape is an array of one or five INTEGER*2 elements. If the "type" parameter has a value of CROSS, SQUARE, RCROSS or CRECT, "shape" is an array of one INTEGER*2 element; the value of this element expresses the length (in pixels) of the side of the square or the arm of the cross.

If "type" has the SW value, "shape" is a five-element array where each element contains the representation of the state of the bits of two lines of the (8*10) bitmap defining the cursor.

The XOR operation is carried out on the screen's contents each time the cursor is moved.

Example

See the example given for the CURSOR function.

DEFSTY

This function redefines the style for drawing the straight lines in the current window. It has no effect on curved lines.

DEFSTY (style-number , shape)

---

where:

style-number is an integer numeric expression which can have a value between 1 and 7 for the monochromatic screen and between 1 and 10 for the colour screen. If "style-number" has a value of 1, the line is continuous; values 2 to 10 mean that the user can redefine the line style with the "shape" parameter.

shape is an array of four INTEGER*2 elements, which redefines the line style for the values from 1 and 7 of the parameter "style-number". Each element, expressed in hexadecimal, indicates the number of pixels to be enabled and how many are to remain disabled. An element containing 0 indicates the start of a repetition and each subsequent element is ignored. For example, the following expression of the "shape" parameter

%030A, %2518, %0000, %3048,...

indicates that the line will be a succession of 3 enabled pixels, 10 disabled pixels, 37 enbaled, 24 disabled. This will be repeated and subsequent elements ignored.

If "style-number" is 1 (a continuous line) the first element of the array defines the thickness of the line and must be an integer between 1 and 255 (see characteristic 2). If the parameter "style-number" is set to 8, 9 or 10 then "shape" is an INTEGER*2 element, expressed in hexadecimal format. The binary representation of each figure of this element specifies the pixels to be enabled (bit=1) and those to be disabled (bit=0).

Characteristics

1. When this function has been executed, the redefined style will automatically be selected for drawing the straight lines, without having to recall the STYLE function.

   See the STYLE function for initial line styles.

2. When using a PC in work station emulation using the GIOMR it is not possible to change the thickness of the line.

This function assigns the element, whose identifier is obtained with the SRCELM function, to a segment.

EXCELM (elemid , newseg)

---

where:

elemid is an integer variable which contains the identifier of the element to be assigned to the segment.

newseg is an integer numeric expression whose value must be between 0 and 254; it identifies the segment to which the element is to be assigned.

Characteristic

The absolute position of the elements in the file is not changed.

FONT

This function selects a font file for the current window.

FONT (fontnum , magnitude , ratio , spacing , direction)

---

where:

fontnum is an integer numeric expression whose value must be between 0 and 8. It specifies the index of the font file to be used.

The initial value is 1.

magnitude is a real numeric expression which must have a positive value. It specifies the character's magnification coefficient.

The initial value is 1.

ratio is a real numeric expression which must have a positive value. It specifies the ratio between the character's height and length.

The initial value is 1.

spacing is a real numeric expression. It specifies the spacing between contiguous characters.

The initial value is 1.

direction is a real numeric expression whose value must be between -2π and $+2\pi$. It specifies the angle, measured in radians from the positive x-axis to the positive y-axis, between the x axis and the direction of the string.

The initial value is 0.

Characteristics

1. The DEFATT/-1 value can be specified for the "magnitude", "ratio", "spacing" and "direction" parameters to use the initial values of the selected font file.

2. If the user has created the FONT8 font file, using the FONT function, he can select one of nine files containing different descriptions of the character sets which can be used with the LABEL function.

The nine available files are:

FONT0 : alphanumeric string precision

FONT1 : simple block letters

FONT2 : double block letters

FONT3 : complex block letters

FONT4 : complex italics

FONT5 : cyrillic

FONT6 : greek

FONT7 : gothic

FONT8 : present only if created by the user

The user can select files with differing descriptions (eventually created using the CHANGEFONT utility, described in Appendix A), connecting them to the global names FONT0,...,FONT8, or directly substituting the files FONT0,...,FONT8 with his own files. When using the integrated work station, or the M30, the M31 or the PC with the GIOMR program used as work stations, the files FONT0,...,FONT8 are located in the directory /IPL/DPC/GRAPHICS on the L1 MOS system. With the PC used as a work station with the GIOM program the files are present in the \GRAPHICS directory on the PC.

3. If a value of 0 is assigned to the "fontnum" parameter the FONT0 file is used, which contains a set of characters described with the same precision as the alphanumeric characters; in this case, in order not to deform the characters, the "magnitude" and "ratio" parameters are ignored and the origin of each character is rotated. With this selection, graphics and strings can be displayed and printed (with the SPRINT function) with the same precision as the alphanumeric text.

4. Each character, whose shape is coded in the selected font file, is represented in a rectangle whose size is equal to:

   height = "magnitude" * 10 * "ratio"

   width = LC * "magnitude"

   (where LC is the length associated with each character in the font file)

   The space between the characters is:

   LC * "magnitude" * "spacing"

FRAME

This function draws/deletes a rectangular frame whose size is equal to the current window.

FRAME (fillflag , atr , drawflag)

Characteristics

1. See the FRAME1 function for the possible values of the "fillflag", "atr" and "drawflag" parameters.

2. The frame is drawn using the line selected or defined by the user.

This function draws/deletes a frame whose diagonal is the specified (x1,y1) and (x2,y2) coordinates.

FRAME2 (step1 , x1 , y1 , step2 , x2 , y2 , fillflag , atr , drawflag)

Characteristics

1. See the FRAME1 function for the possible values of the "step1", "step2", "fillflag", "atr" and "drawflag" parameters.

2. The frame is drawn using the line selected or defined by the user.

GET

This function stores the contents of the current window in an integer array or can record them on a file.

GET (image , size , fname)

Characteristics

1. The parameters of the GET function have the same values and meaning as those of GET2.

2. This function cannot be used with the M30 or M31 used as work stations.

This function stores the contents of a rectangular area of the screen (bitmap) in an integer array or can record them on a file.

GET2 (x1 , y1 , x2 , y2 , image , size , fname)

---

where:

x1,y1,x2,y2 are real numeric expressions whose values identify the coordinates of the diagonal extremes of the rectangle whose contents are to be stored or recorded.

image is an INTEGER*2 array which is used for storing the contents of the rectangular screen area. The array elements are used in the following way:

- The first element will contain the rectanglès base.
- The second element will contain the rectanglès height.
- The third element must contain a flag which indicates whether the image to be stored is monochromatic or colour.

The remaining elements in the array will contain information on the state of the scanline bits making up the rectangular area. If "fname" is a CHARACTER*n expression, this parameter is considered an I/O buffer.

size is an integer numeric expression which indicates how many elements there will be in the array defined by the "image" parameter. If "size" is the value 0, the contents are only stored on file, using an I/O buffer created by the PGU.

fname is a CHARACTER*n string expression whose value is the name of the file on which the image is to be recorded. If the string expression is CHARACTER*0 or if it is a null string, the contents are only stored in the array.

Characteristics

1. The file on which the image is to be stored is automatically created (if it does not already exist), opened and recorded from the beginning. In order to contain the image of the selected rectangle, the array must have a minimum number of elements and the files minimum extension is a number of bytes obtained from the following equation:

   (number of enabled planes) * (base/8 * height) + 6

   where the number of enabled planes must be between 1 and 3.

2. The current graphic position is not updated.

3. This function cannot be used with the M30 or M31 used as work stations.

This function moves the graphic cursor inside the current window, via the arrow keys (↑,→,←,↓) for the screen-keyboard and via the stylus or the four-button cursor for the tablet. If the enabled work station is the screen-keyboard, the ASCII code of the first key pressed after the arrow keys is returned. If the enabled work station is a logical work station on the tablet then the ASCII code returned (after having pressed the stylus or a button on the four-button cursor) is related to the logical work station.

GIN (n-pixel , close-key)

---

where:

n-pixel is an integer numeric expression which specifies by how many pixels the cursor is to be moved.

close-key is a string variable in which the ASCII code of the first key pressed after the arrow keys or the ASCII code related to the logical work station is given.

Characteristics

1. If CONTROL or SHIFT (according to the keyboard type) is entered together with one of the arrow keys, the cursor moves one pixel.

2. This function is suspensive and the user remains in graphic input mode until an alphanumeric key is entered.

3. If the enabled work station is a logical work station on the tablet then the ASCII code returned (after having pressed the stylus or a button on the four-button cursor) indicates the logical work station as follows:

ASCII CODE RETURNED	LOGICAL WORK STATION IDENTIFIER
128	Lowest value.
144	Next to lowest value.
160	Next to highest value.
176	Highest value.

GPOS

This function gives the position of the graphic cursor in world coordinates.

GPOS (axis , position)

---

where:

axis is an integer numeric expression which specifies whether the coordinate value on the x or y axis is to be given in the "position" parameter. The following values are allowed:

- ASC/0 : Gives the graphic cursor's position on the x axis.

- ORD/1 : Gives the graphic cursor's position on the y axis.

position is a real variable where the graphic cursor's position on the requested axis is given.

This function sets the functioning mode for the graphic printers PR 15, PR 38C, PR 1480 with colour, PR 1550 and PR 1580.

GPRMOD (prmode)

---

where:

prmode is an array of five integers in the range -1 to 1. Each integer value refers to a specific printer and to a particular functioning mode. The five integers define in the order:

- the resolution : using the PR 15 or the PR 1550 this integer specifies the maximum width of the bit-map area to be printed. It may be set to one of the following:

  . -1 : the current resolution is not changed

  . 0 : maximum width set to 640 pixels (initial value)

  . 1 : maximum width set to 560 pixels.

- the colour : using the PR 38C or the PR 1480 with colour this integer specifies whether a colour or a monochromatic print is to be obtained. It may be set to one of the following:

  . -1 : the current print mode is not changed

  . 0 : colour print (initial value)

  . 1 : monochromatic print.

- the density : using the PR 1580 this integer specifies whether a normal or a high density print is required. It may be set to one of the following:

  . -1 : the current density is not changed

  . 0 : normal density print (initial value)

  . 1 : high density print.

- the zoom : using the PR 1580 this integer specifies whether a normal or a zoomed print is to be obtained. It may be set to one of the following:

  . -1 : the current mode is not changed

  . 0 : normal print (initial value)

  . 1 : zoomed print (the image size is doubled).

- the closing element : this integer must be set to 0 and is the closing element of the array.

Characteristics

1. The OPNPRT function must have been executed before this function.

2. If the first element of the array is set to 1, the distance between the points printed horizontally and vertically coincides and thus the image to be printed is not altered. If the image to be printed is wider than 560 pixels on the graphic bit-map then it will be clipped.

This function executes the graphic commands of the specified segments in the image file (or image in editing) in the current window.

IMGSEG (filename , action-verb , segments)

---

where:

filename is a CHARACTER expression whose value is the name of the image file. If "filename" is a null string and IMGSEG is called during an editing session, then the commands in the image in editing are executed.

action-verb is an integer numeric expression which can have the following values:

PIXSET/0

PIXRES/5

See the ARC function for the meaning of these values. If PIXRES/5 is used, all the colour attributes and logical actions in the file are ignored.

segments is an INTEGER*2 array which contains the identifiers of the segments. Its values are between 0 and 254, but the last element must be -1. If the first element is -1, then all the segments are displayed.

Characteristics

1. The graphic image is displayed with all the transformations (scale, rotate, translate) and attributes (colour, style, etc) present when the IMGSEG function is executed. These characteristics can be changed by inserting the appropriate commands in the image file. These changes only have a local effect when the IMGSEG function is executed.

2. One or more segments can be simultaneously displayed with this function, and in the second case the elements belonging to the segments to be displayed will be executed in the order in which they are recorded in the image file.

INK

This function fills an area outlined by a border of a particular colour or, if there is no closed area, the whole window with a specific colour or pattern.

INK (x , y , colour , border-colour , flag)

---

where:

x,y are real numeric expressions whose values specify the coordinates of the pixel from which to start filling.

colour is an integer value which identifies the colour with which to fill the window or a part of it. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)

border-colour is an integer numeric expression which specifies which colour is to be considered as the border of the area to be coloured. The values allowed are the same as for "colour".

If the value REV/-1 is specified, then any colour different from the colour used for filling is to be considered as the border.

flag is an integer numeric expression which can have the following values:

- COL/0 : The area is coloured.

- PAT/-1 : The area is filled with the current filling pattern.

Characteristics

1. Only the combinations (ON,ON) and (OFF,OFF) are allowed on the graphic monochromatic screens.

2. The current graphic position is not updated.

INQF

This function gives information on the unit of measurement used on the axes, the current graphic position and the rotation angle.

INQF (info , req)

---

where:

info is a two element REAL array which gives various types of information according to the value of the "req" parameter.

req is an integer numeric expression which can have the values between 0 and 2.

If its value is 0, then:

- info(1) : gives the unit of measurement on the x axis (in pixels)
- info(2) : gives the unit of measurement on the y axis (in pixels).

If its value is 1, then:

- info(1) : gives the abscissa of the current graphic position in world coordinates
- info(2) : gives the ordinate of the current graphic position in world coordinates.

If its value is 2, then:

- info(1) : gives the rotation angle (in radians).

This function gives information on the printing mode, screen controller, limits of the graphic screen and the line style.

INQI (info , req)

---

where:

info is a one, two, five or fifteen element INTEGER*2 array which gives information according to the value of the "req" parameter.

req is an integer numeric expression which can have the values between 0 and 5.

If its value is 0, then:

- info(1)= -1 : the graphic controller is not present
- info(1)= 0 : non-integrated terminal (via RS232)
- info(1)= 1 : a monochromatic integrated graphics screen, or a monochromatic M30 or an M31 used as a work station is present, with a 32K controller (1 page of 640 pixels * 400 scanlines)
- info(1)= 2 : a monochromatic integrated graphics screen, or a monochromatic M30 or an M31 used as a work station is present, with a 128K controller (4 pages of 640 pixels * 400 scanlines)
- info(1)= 3 : an integrated graphics screen with colour or an M30 with colour used as a work station is present
- info(1)= 4 : a monochromatic PC used as a work station is present
- info(1)= 5 : a PC with colour used as a work station is present.

If its value is 1, then:

- info(1)= 0 : the printer is unavailable or an OPNPRT has not been executed

- info(1)= 1 : the printer is the PR 1450

- info(1)= 2 : the printer is the PR 1480 - PR 1480 with colour

- info(1)= 3 : the printer is the PR 2400

- info(1)= 4 : the printer is the PR 15 - PR 17 - PR 1550 - PR 1570

- info(1)= 5 : the printer is the PR 19 - PR 1470 - PR 1490

- info(1)= 6 : the printer is the PR 1580

- info(1)= 7 : the printer is the PR 38 - PR 38C

- info(1)=100 : the printer is the PR 11B - PR 13B - PR 15B - PR 17B - PR 19B - PR 38B - PR 38BC - DM 280 - DM 290 - DM 580 - DM 590 connected to a PC.

If its value is 2, then:

- info(1) : gives the graphic screen extension on the x axis

- info(2) : gives the graphic screen extension on the y axis.

If its value is 3, then:

- info(1) : gives the current foreground colour

- info(2) : gives the current background colour.

If its value is 4, then:

- info(1) : gives the current style number (integer value between 1 and 10)

- info(2) : gives the line thickness (if style number is 1)

- info(2)..info(5) : gives the style description.

If its value is 5, then:

info(1)..info(15) : gives the filling character description (if the screen is monochromatic only info(1)..info(5)).

This function sets the filling mode for the geometric figures drawn via the ARC, CIRCLF, POLYGN and BOX functions.

INSIDE (inside_mode)

---

where:

inside_mode is an integer numeric expression specifying the filling mode:

- SOLID/0 : the geometric figure is to be filled with the colour specified in the relative function.

- PATTRN/1 : the geometric figure is to be filled with the current filling pattern.

Characteristics

1. The initialisation value is SOLID/0.

2. The filling pattern can be changed via the PAINT function.

LABEL

This function displays a character string in a given position.

LABEL (string , x , y , magnitude , rotation)

---

where:

string is a string of printable characters to be displayed, starting from the position specified by the x,y parameters.

x,y are real numeric expressions which identify the position of the first character in the string. This position coincides with the lower left hand corner of the first character's bitmap.

magnitude is a real numeric expression which must have a positive value. It represents the multiplication factor of the initial size of each character in the string. Each character's bitmap is a rectangle whose size is:

height : magnitude * 10 scanlines

width : magnitude * 6 + 2 pixels

rotation is a real numeric expression whose value must be between -2π and $+2\pi$. This value defines the angle, measured in radians from the positive x-axis to the positive y-axis, between the x-axis and the direction in which the string is to be displayed.

Characteristics

1. The LABEL function is the only way in which characters can be printed in graphic mode.

2. The string is displayed using the active font file.

3. The current graphic position is updated to the (x,y) point.

4. If the current font file is FONT0 the "magnitude" parameter is ignored, whereas the "rotation" parameter takes on a different meaning from that given previously. Only the initial point of each character in the string is rotated (the initial point coincides with the lower left hand corner of the bitmap containing the character). Each character in FONT0 is defined by an 8 * 10 pixel rectangle.

The "action verb" parameter has no effect with the REV/-1 value.

action-verb is an integer numeric expression which identifies the operation to be executed on each pixel in the line. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameters of the ARC function for the meaning of these values.

Characteristic

The current graphic position is updated to the second extreme of the line drawn.

Example

```
include '/IPL/DPC/FORT/INCL/GRAPHIC.H'
ires1 = sgment(0,0.,0.,0,50.,70.,1,0)
ires2 = line(0,100.,40.,1,0)
ires3 = line(0,150.,190.,1,0)
ires4 = line(0,200.,90.,1,0)
ires5 = line(0,250.,120.,1,0)
ires6 = line(0,300.,10.,1,0)
ires7 = sgment(0,0.,60.,0,300.,60.,1,0)
end
```

LSTIMG

This function generates a listing of the specified image file or the image in editing.

LSTIMG (listfilename , imagefilename , segments)

---

where:

listfilename is a CHARACTER expression whose value is the name of the file in which the listing is to be recorded. If "listfilename" is a null string, then the listing is printed using the printer previously opened with the OPNPRT call.

imagefilename is a CHARACTER expression whose value is the name of the image file to be listed. If "imagefilename" is a null string, then the image in editing is listed.

segments is an INTEGER*2 array which contains the identifiers of the segments to be listed. Its values are between 0 and 254, and the last element must be -1.

Characteristic

The listing is generated in a particular language. Each line in the listing has the following format:

EL. POS. SEGM. EL.ID. PARAM.

where:

EL.POS. is the absolute position of the element in the file (it can be used as an input parameter in the SRCELM function).

SEGM. is the number identifying the segment to which the element belongs.

EL.ID. is a string identifying the element.

PARAM. is a sequence of parameters associated to the element.

This function specifies which levels of the bitmap memory are to be disabled and which are to be enabled.

MASK (mask)

---

where:

mask is an integer numeric expression which can have the following meaning:

PARAMETER VALUE	BINARY VALUE	MEANING	COLOURS AVAILABLE
BLACK/0	000	all planes are disabled	black
RED/1	001	plane 1 is enabled	black, red
GREEN/2	010	plane 2 is enabled	black, green
YELLOW/3	011	planes 1 and 2 are enabled	black, red, green, yellow
BLUE/4	100	plane 3 is enabled	black, blue
MAGENTA/5	101	planes 1 and 3 are enabled	black, red blue, magenta
CYAN/6	110	planes 2 and 3 are enabled	black, blue green, cyan
WHITE/7	111	all planes are enabled.	all eight colours

Characteristics

1. This function has no effect on a monochromatic screen.

2. The initial value is BLACK/0.

MOVE

This function moves the current graphic position to the specified point.

MOVE (x , y)

---

where:

x,y are real numeric expressions whose values identify the new current graphic position.

This function opens an input work station and, if necessary, it also opens the line connecting the PGU to the specified physical device on which the logical work station resides.

OPGRWS (wsid , wstype , lndescriptor , wsdescriptor)

---

where:

wsid is an integer numeric expression which must have a positive value. It specifies the identifier to be assigned to the logical work station to be opened.

wstype is an integer numeric expression specifying the work station type. It must be set to 0 (at the moment), specifying an input work station.

lndescriptor is a CHARACTER expression containing all the information necessary for opening the line which connects the system to the device on which the logical work station resides. See characteristic 1.

wsdescriptor is a CHARACTER expression containing all the information about the model of the device on which the logical work station resides. See characteristic 2.

Characteristics

1. The "lndescriptor" parameter has the following structure:

   - The first field of five characters specifies the line to which the device is connected. At the moment the only connection method available is via RS232. The first two characters must be PT and the last three must be a value in the range from 001 to 113, which specifies the channel used.

   - The second field of seventeen characters specifies the line transmission characteristics. The default value is: 011010222<<100111. For a detailed description of the line transmission characteristics and the line number see the primitive OpenLn described in the manual RS232/CL Interface , Programmer Guide.

2. The "wsdescriptor" parameter has the following structure:

   - The first field of three characters specifies the input device type. The only value allowed at the moment is TAB, meaning tablet, which is the default value.

   - The second field of ten characters specifies the maximum coordinate along the x axis (5 characters) and the maximum coordinate along the y axis to be returned by the tablet. The default value for all 10 characters is 0297102971.

   - The third field of two characters specifies the tablet transmission mode which, at the moment, must be 01. This value indicates a transmission mode having the sequence:

     XXXX, YYYY, F, CRLF

     where:

     X is an ASCII BCD digit of the x coordinate
     , is an ASCII comma
     Y is an ASCII BCD digit of the y coordinate
     , is an ASCII comma
     F is the ASCII character returned by the function key pressed after having chosen the point whose coordinates are to be returned. It may be one of the following:
        0 no key has been pressed
        1 key 0 of the four button cursor has been pressed or the stylus has been pressed on the tablet
        2 key 1 of the four button cursor has been pressed
        4 key 2 of the four button cursor has been pressed
        8 key 3 of the four button cursor has been pressed
     CR is an ASCII Carriage Return
     LF is an ASCII Line Feed

3. Each field of the "lndescriptor" and "wsdescriptor" parameters for which the default value is to be assumed may be set to "D".

4. The "lndescriptor" and "wsdescriptor" parameters set via this procedure must be consistent with the microswitch settings on the tablet. The default values given by the PGU are consistent with the following microswitch settings on the CALCOMP '2000 Series Digitizer':

   Switch 1 = 000000000
   Switch 2 = 0C00CC00C
   Switch 3 = 00000000C

   where 0 means open and C means closed.

5. The logical work stations on the tablet may physically be separate or superimposed. In order to establish their size and position the VIEW, ACGRWS, and DAGRWS functions must be used. For example, let's suppose that two logical work stations (identified by the values 1 and 2) are to be opened on the tablet. Suppose that work station 1 is to be identified by the coordinates (0,0), (2000,0), (0,2971), (2000,2971) and work station 2 by the coordinates (1500,0), (2971,0), (1500,2971), (2971,2971). The following functions are necessary:

   - DAGRWS to disable work station 0 (if enabled).
   - OPGRWS to open work station 1.
   - OPGRWS to open work station 2.
   - ACGRWS to enable work station 1.
   - VIEW to enable the area on work station 1 identified by the coordinates specified above.
   - DAGRWS to disable work station 1.
   - ACGRWS to enable work station 2.
   - VIEW to enable the area on work station 2 identified by the coordinates specified above.

6. The tablet cannot be connected to the M30, the M31 or the PC used as work stations.

3. This function cannot be called if an image file is being edited.

4. Segment 0 is automatically activated when this function is executed.

OPNPRT

This function opens a work station printer or a system printer. The printers can be connected to the L1 MOS system or to a PC used as a work station.

OPNPRT (mode , sysprtnum , sysprtmod)

---

mode is an integer numeric expression which specifies the type of connection for the graphic printer. The values allowed, and their meanings, are the following:

- STD/0 : work station printer

- SYS/2 : system printer

sysprtnum is a CHARACTER value between 1 and 8. It specifies the number of the system printer to be connected. This value is ignored if the work station printer is to be connected.

sysprtmod is an integer value which specifies the model of the graphic printer used. The following values are allowed:

- PRT1/1 : PR 1450 printer

- PRT2/2 : PR 1480 - PR 1480 printer with colour

- PRT3/3 : PR 2400 printer

- PRT4/4 : PR 15 - PR 17 - PR 1550 - PR 1570 printer

- PRT5/5 : PR 1470 - PR 1490 - PR 19 printer

- PRT6/6 : PR 1580 printer

- PRT7/7 : PR 38 - PR 38C printer

This value is ignored if the PGU automatically recognizes the printer model; in other words, the connection allows messages to be exchanged between the printer and the system (configuration parameters PRT_PRMODEF with least significant byte equal to %00 and TANDEMCTLMO with most significant byte equal to %01), or when using printers connected to PCs used as work stations.

Characteristics

The printer models that can be used are:

- PR 15, PR 17, PR 19, PR 38, PR 38C, PR 1450, PR 1470, PR 1480, PR 1480 with colour, PR 1490, PR 1550, PR 1570, PR 1580, PR 2400, connectable to the L1 MOS system or to PCs used as work stations via the OLIEMU program.

- PR 11B, PR 13B, PR 15B,PR 17B, PR 19B, PR 38B, PR 38BC, DM 280, DM 290, DM 580, DM 590, connectable to PCs used as work stations via the WSELAN program.

This function selects one of the four pages of the bitmap memory.

PAGE (page-num)

---

where:

page-num is an integer numeric expression whose value must be between 1 and 4. It selects the bitmap page. See the PAN function.

Characteristics

1. This function is only effective with the integrated monochromatic work station, with the monochromatic M30 and the M31 used as work stations, with 128K of bitmap memory (4 pages of 32K each).

2. All the subsequent graphic functions after PAGE will produce an output which is stored in the selected memory page. This does not necessarily correspond to the page displayed on the screen. The page can be selected using the PAN function.

PAINT

This function fills an area delimited by a border of a particular colour, or, if there is no closed border, the entire current window, with a pattern specified and defined by the user.

PAINT (x , y , border-colour , mask , shape)

---

where:

x,y are real numeric expressions whose values specify the coordinates of the pixel from which to start filling.

border-colour is an integer numeric expression which specifies the colour which is to be considered as the border of the area to be filled. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off

  Values between 2 and 7 are interpreted as 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)

  If the value REV/-1 is specified, then any colour different from that used for the filling character is considered to be the border.

mask is an integer numeric expression or a character which specifies the number of elements in the array described by the "shape" parameter. The following values are allowed:

- RED/1, GREEN/2, BLUE/4	the array has 5 INTEGER*2 elements or 10 CHARACTER elements (10 bytes associated to planes 1, 2 or 3)
- YELLOW/3	the array has 10 INTEGER*2 elements or 5 INTEGER elements or 20 CHARACTER elements (20 bytes associated to planes 1 and 2)
- MAGENTA/5	the array has 10 INTEGER*2 elements or 5 INTEGER elements or 20 CHARACTER elements (20 bytes associated to planes 1 and 3)
- CYAN/6	the array has 10 INTEGER*2 elements or 5 INTEGER elements or 20 CHARACTER elements (20 bytes associated to planes 2 and 3)
- WHITE/7	the array has 15 INTEGER*2 elements or 30 CHARACTER elements (30 bytes associated to planes 1, 2 and 3)

shape is an INTEGER*2 array of 5/10/15 elements, or a CHARACTER array of 10/20/30 elements. It describes the bitmap (an 8*10 array) of the character with which to fill the area. Each byte describes a line of the rectangle, starting from the top. The bits with a value of 1 correspond to pixel on, and a value of 0 corresponds to pixel off.

Characteristics

1. If working on a monochromatic screen and an array has been defined with more than ten elements, only the first ten are considered as there is only one bitmap plane. The "mask" parameter is ignored. The "mask" parameter is ignored.

2. If the parameters "x" and "y" are defined as INTEGER*2 and set to the value -1 then this function only defines the filling character and does not automatically fill the closed area.

3. The filling pattern defined with this function becomes the current filling pattern.

Example

```
include '/IPL/DPC/FORT/INCL/GRAPHIC.H'
integer*2 shape1(5), shape2(5), shape3(5)
data (shape1(j),j=1,5)/0,0,6168,0,0/
data (shape2(j),j=1,5)/16416,4104,1799,2064,8256/
data (shape3(j),j=1,5)/6168,6399,-232,6168,6168/
ires1 = box(0,150.,150.,0,400.,300.,1,0,0)
ires2 = circle(150.,150.,80.,1,1.,0)
ires3 = circle(400.,300.,80.,1,1.,0)
ires4 = circle(150.,300.,80.,1,1.,0)
ires5 = circle(400.,150.,80.,1,1.,0)
ires6 = paint(200.,200.,1,1,shape1)
ires7 = paint(399.,298.,1,1,shape2)
ires8 = paint(250.,250.,1,1,shape3)
ires9 = ink(150.,310.,1,1,0)
ires10 = ink(410.,150.,1,1,0)
end
```

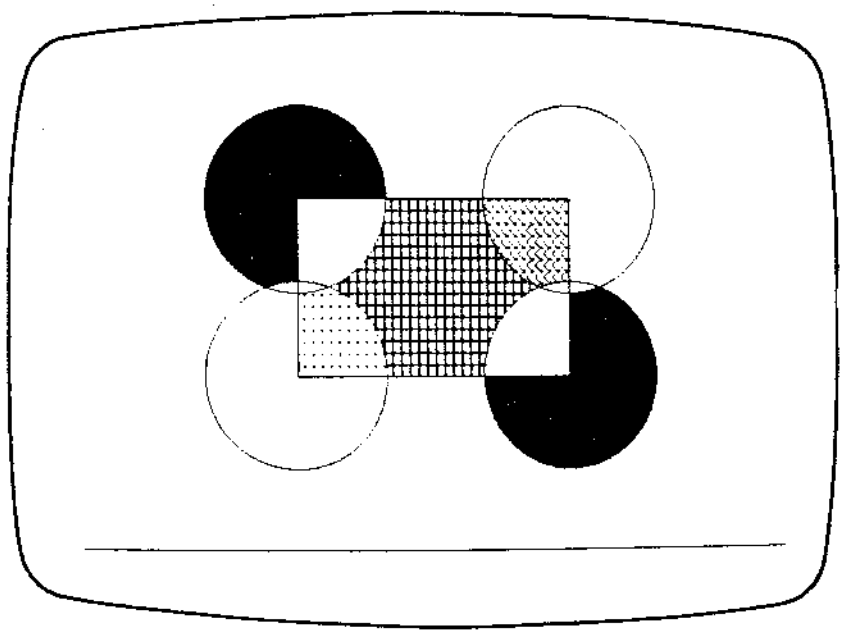


Fig. PAINT-1 Effects of the PAINT Function

This function displays the selected part of the bitmap.

PAN (x , y)

---

where:

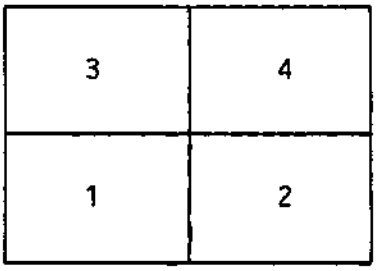
x is an integer numeric expression whose value must be between 0 and 1279. This parameter gives the device abscissa of the pixel which will be at the lower left hand corner of the screen and with which the whole screen will be filled.

y is an integer numeric expression whose value must be between 0 and 799. This parameter gives the device ordinate of the pixel which will be at the lower left hand corner of the screen and with which the whole screen will be filled.

Characteristic

The PAN function only has effect with integrated monochromatic work stations, and with monochromatic M30s and M31s used as work stations, with a 128K bit-map memory (four pages of 32K in which four complete screen images can be stored), with integrated colour work stations and with M30s with colour used as work stations.

In the case of integrated monochromatic work stations, and monochromatic M30s and M31s used as work stations, the x,y coordinates can only have the values (0,0), (640,0), (0,400) and (640,400). These values correspond to the lower left hand corners of the 4 pages on display.



In the case of integrated work stations with colour or M30s with colour used as work stations, any 640 * 400 pixel bit-map area can be displayed, starting from the specified point for the y axis and any point which is a multiple of 16 for the x axis.

POINT

This function gives the pixel's colour code with assigned coordinates.

POINT (x , y , colour-attribute)

---

where:

x,y are real numeric expressions whose values specify the coordinates of the pixel whose colour code, with which it has been activated, is required.

colour-attribute is an integer variable in which the colour code of the pixel at (x,y) is given.

Characteristics

1. The current graphic position is not updated.
2. This function cannot be used with an M30 or M31 used as a work station.

This function draws a polygon and fills it with the specified colour (if the INSIDE function has not been executed) or as specified in the INSIDE function.

POLYGN (n-points , coordinate-array , colour , action-verb)

---

where:

n-points is an integer numeric expression whose value gives the number of vertices in the polygon. The maximum value allowed is 50, and the minimum is 2.

coordinate-array is a real array; it contains the (x1,y1, x2,y2...) coordinates of the vertices of the polygon to be drawn.

colour is an integer value which identifies the colour with which to draw the polygon. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off
  . REV/-1 : pixel on if previously off, and vice versa.

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with REV/-1.

action-verb is an integer numeric expression which identifies the operation to be executed on each pixel in the polygon. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameter of the ARC function for the meaning of these values.

Characteristic

The current graphic position is updated to the last vertex of the polygon.

This function enables the pixel nearest to the x,y coordinates, using the specified colour.

PSET (x , y , colour , action-verb)

---

where:

x,y are real numeric expressions whose values specify the world coordinates of the pixel to be activated.

colour is an integer numeric expression which selects the colour with which to activate the pixel. The following values are allowed:

- for a graphic monochromatic screen:

  . ON/1 : pixel on
  . OFF/0 : pixel off
  . REV/-1 : pixel on if previously off, and vice versa.

  If the value given is between 2 and 7, it is interpreted as the value 1.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with REV/-1 value.

action-verb is an integer value which identifies the logical operation to be executed on each pixel. The following values are allowed:

- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameter of the ARC function for the meaning of these values.

Characteristic

The current graphic position is updated to the (x,y) point if this is inside the window.

This function gives the first available character in the input buffer without displaying it. It is suspensive.

READCH (char)

---

where:

char is a CHARACTER variable which will contain the first available character in the input buffer.

REFRS

This function enables/disables the HIDE ALPHA (96) attribute on the current window.

REFRS (screen-flag)

---

where:

screen-flag is an integer numeric expression which can have the following values:

- FALSE/0

- TRUE/1

If mode 2 (separate) has been selected at the start of the program, the value FALSE/0 indicates that the graphic bitmap must be displayed, while the value TRUE/1 indicates that the alphanumeric bitmap must be displayed.
If, however, mode 0 (together) is selected, the value FALSE/0 indicates that the alphanumeric bitmap must be displayed, while the value TRUE/1 indicates that this must not be displayed.

Characteristic

On integrated work stations with colour, M30s with colour and monochromatic PCs used as work stations, the alphanumeric text and the bitmap are always displayed together.

This function rotates the graphic axes of the world coordinate system (for the current window).

ROTATE (rotation-angle)

---

where:

rotation-angle is a real numeric expression whose value must be between -2π and $+2\pi$. This value defines the angle by which the graphic axes must be rotated. The angle must be given in radians and is measured from the positive x-axis to the positive y-axis. If the "rotation-angle" parameter has the integer value RESET/-1 the graphic axes are set in the initial position.

Characteristic

When ROTATE has been executed, all subsequent graphic functions are executed in relation to the rotated axes. ROTATE behaves in a particular way in the following situations:

- For the GET, PUT1 and PUT2 functions, only the extremes of the rectangle to be stored (or displayed) are rotated; the rectangle's base will always be parallel to the window border and not the rotated x axis.
- For the LABEL function, only the coordinates of the string's initial point are rotated.
- For the BOX function, if ROTATE is executed the fill parameter has no effect.
- If ROTATE is executed, SCALEX and SCALEY cannot be called.
- If SCALE is executed, the effects of ROTATE are cancelled.

SCALAX

This function changes the number of pixels per unit of measure for both the x and y axes. These may be returned via the INQF function.

SCALAX (x_coef , y_coef)

---

where:

x_coef is a real numeric expression whose value is multiplied by the number of pixels per unit of measure along the x axis.

y_coef is a real numeric expression whose value is multiplied by the number of pixels per unit of measure along the y axis.

Characteristic

If the value of "x_coef" and "y_coef" is 1 it has no effect on the axes, if their value is -1 they set the opposite direction of the axis, if their value is 0 the device coordinates are reset.

- for a graphic colour screen:

  . BLACK/0
  . RED/1
  . GREEN/2
  . YELLOW/3
  . BLUE/4
  . MAGENTA/5
  . CYAN/6
  . WHITE/7
  . DEFCOL/8 (current foreground colour)
  . REV/-1 (complementary colour to the one on the screen)

The "action-verb" parameter has no effect with REV/-1.

action-verb is an integer numeric expression which identifies the operation to be executed on each pixel in the line. The following values are allowed:

- NOACT/0
- SXOR/1
- SAND/2
- SNOT/3
- SOR/4
- PIXRES/5
- PIXSET/0

See the same parameter of the ARC function for the meaning of these values.

Characteristic

The graphic position is updated to the second extreme of the line drawn.

SPRINT

This function executes the hard copy (on the graphic printer) of the entire screen or a specified window. If the colour graphic screen and the printers PR 38C, PR 38BC, PR 1480 with colour or DM 590 are used, a colour hard copy of the screen is produced.

SPRINT (window-number , mode , title , timedate)

---

where:

window-number is an integer numeric expression which must have values from 1 to 16. It specifies the window for which the hard copy is to be executed. If the "window-number" parameter has the value SCREEN/0, the hard copy of the entire screen is executed.

mode is an integer numeric expression which describes the printer in chromatic terms. It can have one of two values:

- POS/1 : A printed point on the paper corresponds to an enabled pixel.

- NEG/0 : A printed point on the paper corresponds to a disabled pixel.

title is an alphanumeric string. It must not be longer than 60 characters or it will be clipped. The string is printed in the top left corner of the graphic output.

timedate is an integer numeric expression which allows the time and date to be printed in the top right corner of the output. The following values are allowed:

- DATEON/1 : The date and time are printed.

- DATEOF/0 : The date and time are not printed.

Characteristics

1. If the "mode" parameter has the value POS/1, and a colour hard copy can be printed, the complementary colours of those stored in the graphic bitmap are given (e.g. green (010) becomes magenta (101)).

2. If the printers PR 15, PR 38C, PR 1480 with colour, PR 1550 or PR 1580 are available, see also the GPRMOD function.

3. To obtain a hard copy on printers of type /B from a PC used as a work station, the modules HCBM and HCBC must be present on the PC, for hard copy on monochromatic or colour printers respectively.

4. In the case of hard copy on monochromatic printers from a colour video it must be noted that at present it is not possible to select the colours to be printed. At present, if the "mode" parameter is set to NEG/0, only the colours which have the bit in their binary value that corresponds to the first bit-plane activated set to 0 are printed. If the "mode" parameter is set to POS/1, only the colours which have this bit set to 1 are printed. In view of this, the user must decide in advance which bit-planes to activate and which colours to use to draw the image so that it can be printed.

   The following table, which shows the situation where the "mode" parameter is set to NEG/0 and all the bit-planes are activated ("mask" parameter = 7 in the MASK function), illustrates this point.

COLOUR	BINARY VALUE			RESULT WITH ALL BIT-PLANES ACTIVATED AND "mode"=0
	BIT-PLANE 3	BIT-PLANE 2	BIT-PLANE 1	
Black	0	0	0	Printed
Red	0	0	1	Not printed
Green	0	1	0	Printed
Yellow	0	1	1	Not printed
Blue	1	0	0	Printed
Magenta	1	0	1	Not printed
Cyan	1	1	0	Printed
White	1	1	1	Not printed

   When the first bit-plane activated is bit-plane 2 (for example, "mask" parameter = 6 in the MASK function) and not bit-plane 1, it can be seen from the table that only the colours black, red, blue and magenta would be printed (the bit in the bit-plane 2 column is equal to 0).

   For activation and deactivation of the bit-planes and for the colours available, see the MASK function.

5. This function cannot be used with an M30 or M31 used as a work station.

This function enables the areas (viewports) for the graphic input and output. These areas are marked by the parameters xmin, xmax, ymin and ymax.

VIEW (xmin , xmax , ymin , ymax)

---

where:

xmin, xmax, ymin, ymax are integer numeric expressions which give the size of the viewport. Their values must be between:

0 <= x <= 1279	0 <= y <= 799	for integrated work stations with colour and for M30s with colour used as work stations.
0 <= x <= 639	0 <= y <= 399	for integrated monochromatic work stations, for monochromatic M30s, for M31s and for PCs used as work stations.
0 <= x <= 2971	0 <= y <= 2971	for a tablet

The minimum value for the height and width of a viewport on the screen is 10*8 pixels. If the "xmin" parameter has the value RESET/-1 the largest viewport possible is opened.

Characteristics

1. The origin of the world coordinates is placed in the lower left hand corner of the viewport and the unit of measurement is equal to one on both axes. The entire screen can be used to display the alphanumeric output.

2. When VIEW is used, any previously created logical structure of the windows is deleted; consequently, each call to the SETWIN, CLSWIN and CLRWIN functions will give an error until the BOUNDS or CLEAR functions are called.

3. This function affects all the currently enabled work stations.

ZOOM

This function enlarges the displayed image, starting from the lower left hand corner and according to the specified magnification factor.

ZOOM (level)

---

where:

level is an integer numeric expression whose value must be between 0 and 15. It defines the magnification coefficient. The value 1 doubles the image, the value 2 triples it, etc. The value 0 resets the image as it was before being zoomed.

Characteristics

1. If the zoom level 0 is applied to an image that has not previously been zoomed, this function has no effect.

2. This function has no effect on integrated monochromatic work stations, or on monochromatic M30s, on M31s and on PCs used as work stations.

Title: L1 MOS System Software Maintenance - User Guide

Newsletter Code: 4002304 K

Date: May 1987

Publication Code: 4002300 M(1)

Previous Newsletters: 4002302 S
4002303 W

This Newsletter provides updated pages for the subject publication.

The last level completed on the attached form, Updating Status, indicates the pages to be added, removed, or replaced, the number of pages included, and the Newsletter Code. Pages marked with an asterisk should be removed from the publication. The form should be filed at the back of the publication as a permanent record of amended pages.

Each amended page is identified by the Newsletter Code shown above. Amended pages remain valid unless otherwise noted in a subsequent Newsletter. Modifications to text, figures, or tables are indicated by a vertical bar in the outside margin next to the change.

Summary of Amendments: for Rel 5.2

- New RAM DEBUGGER command: NOT MASKABLE INTERRUPT
- Updating of commands: READALTM, TMDUMP, DISKEDIT, HEXED, NOSE
- Updating of system and Shell maintenance commands error messages
- Description of the certified limits of the Shell maintenance commands.

Ing. C. Olivetti & C., S.p.A. — Direzione Documentazione — Via Jervis 77, — 10015 IVREA (TO) Italy

Printed in Italy

olivetti

UPDATING STATUS

LEVEL	DATE	UPDATED PAGES	PAGES	CODE
1	Sept. 84		150	4002300 M (1)
2	Jan. 85	Pref., vi, vii, 2-1, 2-2, 2-8, 2-12÷2-17, 4-1, LISTMSG-1÷4, NOSE-5, RCODE-1, TREECHECK-0-4÷7, VOLGC-0÷6, A-1÷A-16	45	4002302 S
3	Oct. 85	Pref., v÷viii, 1-2, 1-3, 2-2÷2-25, 3-1, 3-7-3÷17, 4-1÷4-6, CSIZE-1, DISKCHECK-7÷10, DISKEDIT-7÷13, HEXED-3, LISTCON-1-2-4-6-8÷10, LISTMSG-3-4, NOSE-1, PERFPROG-1÷17, TREECHECK-1-2-4÷8, TREEMEND-1-2, VOLGC-0-2-3-7÷11, A-1, A-6, A-12, A-15÷A-21	117	4002303 W
4	May 87	Manual tree, vi÷viii, Pref, 2-2, 2-8, 2-9, 2-17, 2-25, 3-1÷3-17, CSIZE-5, DISKCHECK-3, DISKCHECK-7*÷10*, DISKEDIT-1, DISKEDIT-1/A, 1/B, HEXED-1÷15, LISTMSG-2, LISTMSG-3*, LISTMSG-4*, NOSE-3/A÷3/D, NOSE-4, NOSE-34÷38, TREECHECK-1, TRECHECK-4, TRECHECK-5*÷TRECHECK-8*, TREEMEND-1, VOLGC-0÷3, VOLGC-7*÷VOLGC-11*, A-1÷31 , B-1÷27, C-1, C-2	126	4002304 K
5				
6				
7				

Pages marked * must be suppressed