CERN Program Library Long Writeups Q120 and Y251

HIGZ

High Level Interface to Graphics and Zebra

User's Guide

HPLOT

User's Guide

Application Software Group

Computing and Networks Division

CERN Geneva, Switzerland

```
CERN Program Library Office
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel.  +41 22 767 4951
Fax.  +41 22 767 7155
Bitnet:  CERNLIB@CERNVM
DECnet:  VXCERN::CERNLIB (node 22.190)
Internet:  CERNLIB@CERNVM.CERN.CH
```


Contact Person: Olivier Couet /CN (couet@cern.cern.ch)
Technical Realization: Michel Goossens /CN (goossens@cern.cern.ch)

Edition – June 1995

Preliminary remarks

This guide conbines the user documentation for both the HIGZ (Part I) and HPLOT (Part II) packages. They are implemented on various mainframes (e.g. IBM VM/CMS, Cray and VAX/VMS) and Unix workstations (e.g. HP, Apollo, Ultrix, IBM RS6000, Silicon Graphics and Sun).

HIGZ has been designed to provide basic graphics functions similar to GKS. HPLOT is a histogram plotting and editing system closely linked to HBOOK.

notation

Throughout this manual, all the GKS like functions are indicated as follows:

GKS `CALL` **GKSLIKE** `(parameters)`

Type of the subroutine parameters is defined by their initial letter following the usual Fortran conventions:

- parameters starting with the letter `I` through `N` are `INTEGER`.
- parameters starting with the letter `A` through `H` and `O` through `Z` are `REAL`.
- in addition to the above, parameters starting with the sequence `CH` are of type `CHARACTER`.

In the description of the routines a `*` following the name of a parameter indicates that this is an **output** parameter (e.g. `OUTPAR*`). If another `*` precedes a parameter in the calling sequence, the parameter in question is both an **input** and **output** parameter (e.g. `*IOPAR*`).

Examples are in `monotype face` and strings to be input by the user are `underlined`. In the index the page where a routine is defined is in **bold**, page numbers where a routine is referenced are in normal type.

This document has been produced using LaTeX [1] with the `cernman` style option, developed at CERN. A compressed PostScript file `higz.ps`, containing a complete printable version of this manual, can be obtained from any CERN machine by anonymous ftp as follows (commands to be typed by the user are underlined):

```
ftp asisftp.cern.ch
Connected to asis00.cern.ch.
Name (asis01:username):  anonymous
Password:  your_mailaddress
ftp> cd cernlib/doc/ps.dir
ftp> binary
ftp> get higz.ps.gz
ftp> quit
gunzip higz.ps.gz
```

Table of Contents

List of Figures

List of Tables

Part I

HIGZ – Reference Section

Chapter 1: Introduction

The present document describes the HIGZ package (High level Interface to Graphics and ZEBRA). The package is a part of larger system PAW (Physics Analysis Workstation)[2], and it was originally implemented in order to provide a graphics interface to PAW. However HIGZ can also be used independently.

Graphics packages like GKS [3] mediate the transition from user programs (applications) to devices in a standardized way. The European effort to restrict High Energy Physics users to using only one such package (at least for the 2D graphics), GKS, will yield portability of application programs between systems on which GKS is installed, and will make the application programs largely device-independent.

These packages, however, have limitations. They do not foresee an acceptable way of recording large volumes of graphical information in compact form with a convenient access method, for later manipulation. The GKS metafile is conceived as a vehicle to communicate series of pictures between computers, but not for their subsequent manipulation. Also, the acceptance of GKS, in particular by Laboratories outside Europe, is still rather modest, and thus it is not a standard that the High Energy Physics community can restrict itself to exclusively. We believe that the following requirements must be met by the graphical output of PAW:

1. The PAW picture data base must be fully transportable.
2. It must have easily accessible units (pictures) for later manipulation.
3. The picture data base must be as compact as possible, and accessible in direct access mode.
4. The picture data base must be independent from the underlying graphics package and, a fortiori, from different implementations of the same graphics package.

These requirements are not restricted to PAW. They are common to many applications existing or under development. We therefore define below an interface package called HIGZ, written in the context of PAW, and aiming at graphics applications of any nature, provided the level of functionality is similar. This package is basically a thin layer between the user program (application) and an underlying graphics package, offering the following advantages:

1. An interface to a standard memory management system (ZEBRA) [4], and through it a mechanism to store graphics data in a way which makes their organization and subsequent editing possible and easy. The picture data base is also highly condensed and fully transportable. A picture editor is part of the package. It allows merging of pictures, editing of basic graphics primitives, operations onto HIGZ structures, etc.
2. A GKS like user interface to the graphics package, keeping the program independent of the underlying graphics package installed.

The level of HIGZ was deliberately chosen to be close to GKS and as basic as possible. This makes the interface to GKS a very simple one, and preserves full compatibility with the most important underlying graphics packages. HIGZ does not introduce new basic graphics features, and does not duplicate GKS functions. On the other hand, some graphic macroprimitives are implemented, providing very frequently used functions, such as graphs, circles and axis. The user will also be able to call GKS directly in parallel with the use of HIGZ.

Many of the underlying GKS concepts used by HIGZ, e.g. the concepts of workstations and viewports, are well explained in [3] and in [5].

HIGZ is presently interfaced to several versions of GKS. The version of GKS can be selected at compilation time by PATCHY control statements. On the CERN central computers the GKS-GRAL version is implemented. The list of the different GKS versions, and of the values of GKS version-dependent parameters are specified in the appendix.

HIGZ is also interfaced the most important graphics packages such as PHIGS, DI3000, GDDM (IBM), GPR (APOLLO), GL (Silicon Graphics). Simple interfaces to the Tektronix/FALCO terminal and to the X Window System on all the modern workstations are also available.

Thoughout this manual the graphics package on top of which HIGZ is installed is referenced as "underlying graphics package". When HIGZ has initialized the underlying graphics package, the application program can call it directly. For example, if the underlying graphics package is GKS, the application program can access the segmentation facilities, but this will be not seen by HIGZ. For all the additional functionalities provided by the underlying graphics package, HIGZ is transparent.

The X Window System interface is now one of the most frequently used on workstations but also on mainframes like VAXes or IBM/VM machines. It has the advantages of a great portability, good performances, and the possibility to be used remotely through a network. The HIGZ interfaces to the X Window System is a small layer callable by Fortran providing a convenient way to access the basic Xlib facilities from Fortran. This interface is described in the chapter: **The** X Window System **interface routines**.

Most modern underlying graphics packages usable from HIGZ provide PostScript drivers. These drivers can be used through HIGZ, but a good uniform interface to PostScript is so important that HIGZ has its own native PostScript driver independent from the underlying graphics package used (see section 3.3.1).

In order to produce similar outputs even with different underlying graphics packages, HIGZ has its own line styles, hatches, marker types and text independent from the underlying graphics package. Thus it is possible to use all the basic tools even on a very simple terminal (for example a FALCO).

1.1 Functionality

The HIGZ system is subdivided into three main sets of functions:

1. Basic graphics functions (`I...` routines), interfacing to the underlying graphics package, with calling sequences identical to those of GKS.
2. Higher-level macroprimitives (`IG...` routines), and the related control routines.
3. Memory management function (`IZ...` routines), interfacing to the memory management system (ZEBRA).

The `IG...` and the `I...` functions act on the screen and/or on the data structure in main storage. All graphics functions producing a graphics object are able to direct the output:

- to the display device
- to the data management system
- to both

These actions are controlled by a switch set by the routine `IGZSET`.

The `IZ...` functions are the memory management functions. They act on the data structure in main storage and on the data stored on disk. This is particularly useful during an interactive session, as the user is able to "replay" pictures previously created, with no need to recall the application program, but just accessing the picture data base.

Chapter 2: Overall control routines

2.1 Control routines

2.1.1 Initialization

```
CALL IGINIT (NWHIGZ)
```

Action: This routine initializes HIGZ. This must be the first function to be used in the HIGZ package.

Parameter description:

`NWHIGZ` Minimal ZEBRA dynamic space in memory for the HIGZ division; A value of 0, indicates that allocation will be done automatically. `NWHIGZ` must be less than `NWORDS-5000` where `NWORDS` is the size of the common block `PAWC` (see below).

The ZEBRA memory allocation must be defined in the application program with the common block:

```
COMMON/PAWC/RPAW(NWORDS)
```

If HIGZ is used outside the context of PAW the routine `MZPAW` must be called in the main program in order to initialize the ZEBRA package [4], before calling `IGINIT`. Note that packages like HBOOK[6], HPLOT[7], PAW[2] and KUIP[8] call `MZPAW` directly and therefore the user should not issue such a call. These packages store dynamic structures in the same common `/PAWC/`.

```
CALL MZPAW(NWORDS,'M')
```

2.1.2 Termination

```
CALL IGEND
```

Action: This routine terminates HIGZ. This must be the last call to be issued in a HIGZ session. `IGEND` deactivates and closes all open workstations. It also closes the basic graphics package by calling `IDAWK`, `ICLWK`, `ICLKS`.

2.1.3 Graphic package control

```
CALL IGSSE (IERRF,KWTYPE)
```

Action: In general, the initialization of the underlaying graphics package consists in several calls to different routines, in order to set the environment parameters. For user's convenience and for most applications, `IGSSE` initializes the standard graphic package environment. In particular, the default primitives attributes and the default window, viewport, workstation window and workstation viewport are initialized. Sophisticated applications may need to call the specialized basic control routines, namely `IOPKS`, `IOPWK`, `IACWK`, `ISWKWN` and `ISWKVP`, instead of using `IGSSE`. `IGSSE` opens only a single workstation.

Parameter description:

`IERRF` Error file logical unit number.

`KWTYPE` Workstation type. See the description of `IOPWK` section 3.1.3.

`IGSSE` calls the following routines:

`IOPKS`	See section 3.1.1.
`IOPWK(1,KONID,KWTYPE)`	See section 3.1.3.
`IACWK(1)`	See section 3.1.6.

Note that `KONID` is initialized in `IGSSE` depending on the underlying graphics package used. In general `KONID` is set to 1.

The workstation window and viewport are also initialized in `IGSSE` as follows:

```
CALL ISWKWN(1,0.,1.,0.,1.)
CALL ISWKVP (1,0.,XMAX,0.,YMAX)
```

where `XMAX` and `YMAX` are the screen dimensions in pixels.

The following primitives attributes are initialized:

Attributes names	Default values
Polyline colour index	1
Line type	1
Line width	1.0
Polymarker colour index	1
Marker type	1
Marker scale factor	1.0
Fill area colour index	1
Fill area interior style	0
Fill area style index	1
Character height	0.01
Character up vector	0.0,1.0
Text alignment	0,0
Text font and precision	0,2
Text colour index	1
Clipping indicator	1
GKS Aspect source flag	Individual attributes

The first heigh elements of the colour table are initialized as follow:

Colour indeces	Colour
0	White
1	Black
2	Red
3	Green
4	Blue
5	Yellow
6	Magenta
7	Cyan

In addition to this initialization role, IGSSE, when it is used in the context of the telnetg program, allows to open the connection between the remote machine and the local one even if the X Window System is not available. This is done by giving to IGSSE the negative value of the local workstation type.

2.1.4 Display control

Many terminals provide different modes: for example a Tektronix emulation mode (or graphics mode) and a VT100 emulation mode (or alphanumeric mode). Some terminals have (additionally) two overlayed screens: a graphics screen and an alphanumeric screen (or dialog scroll). If a Fortran input is requested, the operating system generally displays a prompt (for example "CMS READ"), which belongs to the alphanumeric screen in VT100 emulation mode.

HIGZ provides two functions to switch between these modes and to enable Fortran input and output. In some systems (e.g. IBM's VM/CMS) it is essential that all Fortran input/output be performed in alphanumeric mode, else an abend will occur.

Graphic mode

```
CALL IGSG (KWKID)
```

Action: This routine takes the terminal back into graphics mode and enables graphics input/output. This task is in general performed automatically by all the basic graphics routines.

Parameter description:

KWKID Workstation identifier

Alphanumeric mode

```
CALL IGSA (KWKID)
```

Action: This routine takes the terminal out of graphics mode into alphanumeric mode. On terminals like Pericom Graphics the bell is rung and the user has to press the <CR> key to continue.

Parameter description:

KWKID Workstation identifier

2.2 The minimal HIGZ program

We are now able to write the minimal HIGZ program which only opens and closes HIGZ without doing any graphics. All the graphics routines described in the rest of this manual will be placed between the call to IGSSE and the call to IGEND.

The minimal HIGZ program

```
      PROGRAM MINIMAL
*
      PARAMETER (NWPAW=20000)
```

```
      COMMON/PAWC/RPAW(NWPAW)
*                              Initialize storage in /PAWC/.
      CALL MZPAW(NWPAW,'M')
*                              Initialize HIGZ.
      CALL IGINIT(0)
*                              Set standard environment.
*                              Errors are written to standard output (UNIT 6).
*                              Workstation type is 1.
      CALL IGSSE(6,1)
*                              Deactivate and close all open workstations
*                              Close HIGZ.
      CALL IGEND
*
      END
```

Note that by default the MZPAW routine does a verbose initialization of ZEBRA. To have a quiet initialization the single call to MZPAW should be replaced by:

```
      CALL MZEBRA(-3)
      CALL MZPAW(NWPAW,' ')
```

Warning: on the IBM VM/CMS systems, a:

```
      CALL INITC
```

is mandatory in the main program to force the loading of the C library.

Chapter 3: The basic graphics routines

3.1 Control

3.1.1 Graphic package open

GKS `CALL` **IOPKS** `(IERRF)`

Action: This routine initializes the graphic package for use. It should be the first of all graphic package routines called by the user program, just after the call to `IGINIT`. The opposite of `IOPKS` is `ICLKS`.

This routine is called by `IGSSE` and it must **NOT** be called if `IGSSE` has been already invoked.

Parameter description:

`IERRF` Logical unit number of the file for recording error messages. If `IERRF` is equal to `6`, the error messages are printed on the screen otherwise they are redirected to the file `higz.err` or to the error file opened by the underlying graphics package.

3.1.2 Graphic package close

GKS `CALL` **ICLKS**

Action: This routine terminates the usage of the graphic package. It is the opposite of `IOPKS`. The routine `ICLKS` should be called only when there are no open workstations (see `ICLWK`). Note that `IGEND` calls `ICLKS` automatically.

3.1.3 Workstation open

GKS `CALL` **IOPWK** `(KWKID,KONID,KWTYPE)`

Action: This routine initializes a workstation for use. It is usually the second of all graphic package routines called by the user program. Note that more than one workstation may be opened at the same time. A workstation means a terminal, a graphics window, or a metafile (see section 3.3.1). The opposite of `IOPWK` is `ICLWK`. Note that `IGSSE` opens and activates the workstation number 1 (see section 2.1.3), `IGMETA` use the workstation number 2 (see section 3.3.1).

Parameter description:

`KWKID` Workstation identifier. It must be used in subsequent calls to activate or deactivate the workstation (`IACWK` and `IDAWK`), to clear it (`ICLRWK`), or to close it (`ICLWK`). `KWKID` is also used in certain inquiry or option setting routines.

`KONID` Connection identifier. It is a system-specific identifier related to the access way to the graphics device. HIGZ doesn't use it and pass it directly to the underlying graphics package. If the workstation to be opened is a metafile, `KONID` is the logical unit number on which the Fortran file has been opened (see section 3.3.1) in this case it can be any number smaller than 100.

`KWTYPE` Workstation type. It selects which type of workstation has to be opened. `KWTYPE` must be among the predefined types that are supported by the underlying graphics package (see the appendix B). With the X11, GPR, and GL versions of HIGZ the `KWTYPE` corresponds to a line number in the file `higz_windows.dat` (or `HIGZWIN DATA` on `IBM/VM` machines). When `IOPWK` is called, it tries to open the file `higz_windows.dat` in the working directory. If it does not succeed it tries in the HOME directory. If it doesn't succeed again it creates this file in the home directory as follows:

```
0000 0000 0600 0600
        .
        .
        .
0000 0000 0600 0600
```

where the lines define each of the workstation types (from 1 to 10) with the x-margin (left), y-margin (top), x-size (width) and y-size (height) of the corresponding window in pixels. Using the X11 version the output is redirected (like for all X11 applications) to the display defined via the environment variable `DISPLAY`.

3.1.4 Get workstation type

CALL **IGWKTY** (KWTYPE*)

Action: This routine gets the workstation type from the standard input.

Parameter description:

`KWTYPE` Workstation type. A call to this routine will prompt the user with:

`Workstation type (?=HELP) <CR>=1`

Just typing `CR` will return the default value in `KWTYPE`. The value of the default depends on the HIGZ installation. Typing `?` will give a short help listing on all the different possible workstation types. Any other answer will be interpreted as a new workstation type. Note that with the X11 version of HIGZ the routine `IGWKTY` will accept a workstation type like: `n.hostname` where `n` is the line number in the file `higz_windows.dat` and `hostname` is the name of the machine on which the graphics will be displayed. In this way it is not necessary to define the variable `DISPLAY` before using HIGZ.

– If a workstation type like `n.hostname` is entered, the `hostname` is written at the end of the line number `n` of the file `higz_windows.dat`.
– If the workstation type `n` is entered and if a `hostname` is present on the line number `n` of the file `higz_windows.dat`, the graphics will be redirected to the machine `hostname`.
– If the workstation type `n` is entered and if a `hostname` is not on the line number `n` of the file `higz_windows.dat`, the graphics will be redirected to the machine defined by the variable `DISPLAY`.
– If the workstation type `n.` is entered and if a `hostname` is present on the line number `n` of the file `higz_windows.dat`, the graphics will be redirected to the machine defined by the variable `DISPLAY` and `hostname` is removed from the line `n` in `higz_windows.dat`.

Remark:

In the file `higz_windows.dat`, it is possible to specify the name of the window just after the `hostname`.

3.1.5 Workstation close

GKS `CALL **ICLWK** (KWKID)`

Action: This routine terminates the usage of the workstation. It is the opposite of `IOPWK`.

Parameter description:

`KWKID` Workstation identifier defined in `IOPWK`.

3.1.6 Workstation activation

GKS `CALL **IACWK** (KWKID)`

Action: This routine prepares a previously opened workstation (see `IOPWK`) to receive output primitives. It must always be used for workstations on which one wishes to draw primitives. In addition, `IACWK` and its opposite `IDAWK` are used with multiple workstations to control which of them will receive any new primitives.

Parameter description:

`KWKID` Workstation identifier defined in `IOPWK`.

3.1.7 Workstation deactivation

GKS `CALL **IDAWK** (KWKID)`

Action: This routine deactivates an active workstation. It is the opposite of `IACWK`. It must always be used before closing a workstation previously activated. In addition, `IACWK` and `IDAWK` are used when multiple workstations are open to control which of them receive any new primitives.

Parameter description:

`KWKID` Workstation identifier.

3.1.8 Update workstation

GKS `CALL **IUWK** (KWKID,IRFLG)`

Action: This routine updates the workstation KWKID. It send all buffered output to the screen. In the X11 version of HIGZ, this routine allows to flush the X11 buffer. This routine is usually called with the first parameter equal to 0 and the second to 1.

Parameter description:

`KWKID` Workstation identifier. `KWKID = 0` updates all the current open workstations.

`IRFLG` Regeneration flag:

- `0` postpone update workstation (only when the underlying graphics package is GKS)
- `1` refresh entire display
- `2` update current view

3.1.9 Update workstation and go to alphanumeric mode

CALL **IGTERM**

Action: Very often application programs require to update the open workstations and then return to the alphanumeric mode. This routine without parameters, provides these two actions. Essentially it performs the following calls:

```
CALL IUWK(0,1)
CALL IGSA(0)
```

3.1.10 Workstation clear

GKS CALL **ICLRWK** (KWKID,KOFL)

Action: This routine clears the output area of a workstation which has been previously opened.

Parameter description:

KWKID Workstation identifier. On a softcopy device (e.g. a terminal), the output area is cleared. On a hardcopy device, the paper is advanced, so that a fresh area is available for drawing. If KWKID =0 then all active workstations are cleared.

KOFL Flag controlling the operation of routine ICLRWK on a workstation for which the output area is already cleared. Possible values are:

- 0 If there has been no output since the previous ICLRWK, nothing happens.
- 1 The output medium is advanced or cleared in any cases.

If a change has been requested in the workstation transformation (via ISWKVP or ISWKWN), the workstation transformation is recalculated when ICLRWK is called.

With the GPR, GL, and X11 versions of HIGZ, if the window size has changed, the new size will be automatically taken into account after a clear workstation.

3.2 The coordinate systems and transformations

The coordinate systems and transformations are the same as for GKS. Three coordinate systems are used, namely the world coordinates (WC), normalized device coordinates (NDC) and device coordinates (DC) systems. Two transformations are then necessary, the normalization transformation (NT) going from world coordinates to normalized device coordinates space and the workstation transformation (WT) going from normalized device coordinates to device coordinates space.

The normalized device coordinates space is a fixed space, a square whose bottom left corner (the origin) has the coordinates (0.,0.) and the top right corner has the coordinates (1.,1.).

The mapping from normalized device coordinates to device coordinates and in general the knowledge of device parameters is supplied by default by the standard initialization function (but user callable routines are also provided). The complete viewing pipeline is described on figure 3.1.

For devices with variable windowing capabilities, HIGZ gives the possibility to change dynamically or after a clear (see ICLRWK) the device viewport, and to inform the basic graphics package of this via the routine IGQWK (see section 6.2).

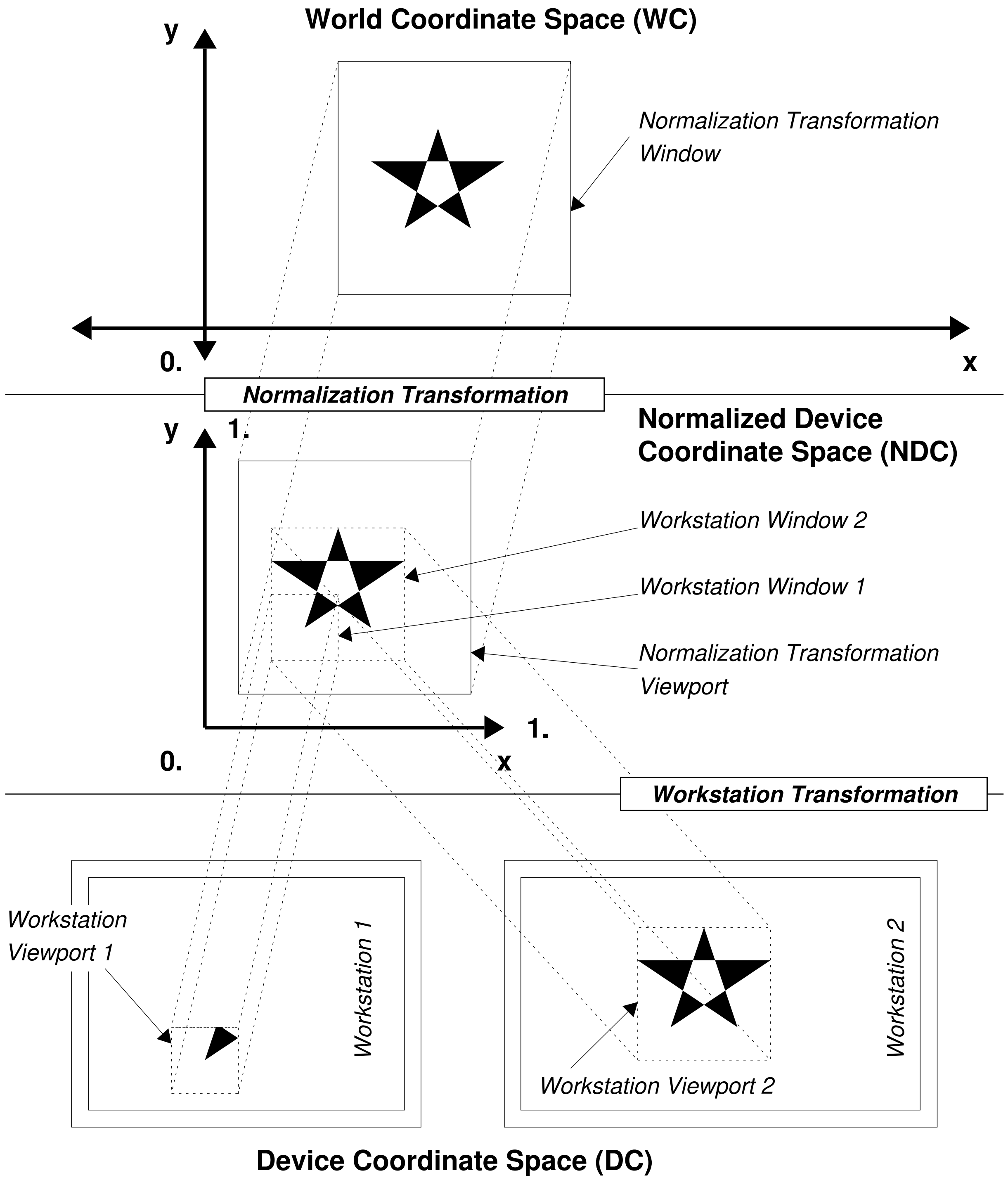


Figure 3.1: Normalization and Workstation Transformations.

3.2.1 Workstation window definition

GKS `CALL **ISWKWN** (KWKID,XMIN,XMAX,YMIN,YMAX)`

Action: This routine defines a workstation window in the normalized device coordinates space. It sets the (requested) workstation window on a previously opened workstation. The workstation window, specified in normalized device coordinates (i.e., `0.-1.` by `0.-1.`) is the portion of normalized device coordinates space that the application wishes to appear on the given workstation. This permits primitives which are created when multiple workstations are active to be clipped and scaled differently on the different workstations.

The workstation window (together with the workstation viewport and the rule that the aspect ratio of the workstation window must be preserved) determines the mapping (uniform scale with translation) from normalized device coordinates to device coordinates.

The requested workstation window becomes the current workstation window either during the invocation of `ISWKWN` (if the display surface is empty or if it does not cause an implicit regeneration) or at some later time (for example, during an invocation of `ICLRWK`).

Parameter description:

`KWKID`	Workstation identifier
`XMIN`	X coordinate of the lower left hand corner in NDC space.
`XMAX`	X coordinate of the upper right hand corner in NDC space.
`YMIN`	Y coordinate of the lower left hand corner in NDC space.
`YMAX`	Y coordinate of the upper right hand corner in NDC space.

The four last parameters must be between `0.0` and `1.0` (inclusive) and must satisfy `XMIN < XMAX` and `YMIN < YMAX`.

3.2.2 Workstation viewport definition

GKS `CALL **ISWKVP** (KWKID,XMIN,XMAX,YMIN,YMAX)`

Action: This routine sets the (requested) workstation viewport on a previously opened workstation. The workstation viewport, specified in device coordinates, is the portion of the maximum available display surface that the application wishes to use (see section 6.2).

The workstation viewport (together with the workstation window and the rule that aspect ratios must be preserved) also determines the mapping (uniform scaling with translation) from normalized device coordinates to device coordinates.

The requested workstation viewport becomes the current workstation viewport either during the invocation of `ISWKVP` (if the display surface is empty or if it does not cause an implicit regeneration) or at some later time (for example, during an invocation of `ICLRWK`). The device coordinates region specified by the parameters must be contained in or equal to the maximum available display surface. The initial requested workstation viewport is the entire display surface.

Parameter description:

KWKID	Workstation identifier
XMIN	X coordinate of the lower left hand corner in DC space
XMAX	X coordinate of the upper right hand corner in DC space
YMIN	Y coordinate of the lower left hand corner in DC space
YMAX	Y coordinate of the upper right hand corner in DC space

The last four parameters must satisfy the conditions `XMIN < XMAX` and `YMIN < YMAX`.

3.2.3 Normalization Transformation window definition

GKS `CALL` **ISWN** `(NT,XMIN,XMAX,YMIN,YMAX)`

Action: This routine sets the boundaries of the window of a normalization transformation. The window must be specified in world coordinates. The boundaries of the window, together with the boundaries of the viewport (which are in normalized device coordinates) determine a transformation from world coordinates to normalized device coordinates consisting of separate X and Y scale factors and a translation in two dimensions. The normalization transformation is selected by using routine `ISELNT`.

Parameter description:

NT	Normalization transformation index (`0<NT<1000000`).
XMIN	X coordinate of the lower left hand corner in WC space.
XMAX	X coordinate of the upper right hand corner in WC space.
YMIN	Y coordinate of the lower left hand corner in WC space.
YMAX	Y coordinate of the upper right hand corner in WC space.

The last four parameters must satisfy the conditions `XMIN < XMAX` and `YMIN < YMAX`.

3.2.4 Normalization Transformation viewport definition

GKS `CALL` **ISVP** `(NT,XMIN,XMAX,YMIN,YMAX)`

Action: This routine sets the boundaries of the viewport of a normalization transformation. The viewport must be specified in normalized device coordinates. The boundaries of the viewport have two roles:

1. Together with the boundaries of the window (which are in world coordinates) they determine a transformation from world coordinates to normalized device coordinates consisting of separate X and Y scale factors and a translation in two dimensions.
2. When the clipping indicator is 1 (see `ISCLIP`), primitives are clipped to the boundary of the viewport (once the primitives are transformed to normalized device coordinates)

The normalization transformation is selected with the routine `ISELNT`.

Parameter description:

NT	Normalization transformation index (0<NT<1000000).
XMIN	X coordinate of the lower left hand corner in NDC space (0.0 ≤XMIN≤ 1.0).
XMAX	X coordinate of the upper right hand corner in NDC space (0.0 ≤XMAX≤ 1.0).
YMIN	Y coordinate of the lower left hand corner in NDC space (0.0 ≤YMIN≤ 1.0).
YMAX	Y coordinate of the upper right hand corner in NDC space (0.0 ≤YMAX≤ 1.0).

The last four parameters must satisfy the conditions XMIN < XMAX and YMIN < YMAX.

3.2.5 Normalization transformation selection

GKS
```
CALL ISELNT (NT)
```

Action: This routine selects the normalization transformation to be used when world coordinates must be mapped to or from normalized device coordinates (NDC). These mappings usually take place during invocations of primitives (IFA, IPL, IPM, and ITX) and during graphics input (IRQLC).

Transformation 0 always has a window and a viewport that are the unit square (0.-1. by 0.-1.) and cannot be changed with ISVP or ISWN. Transformation 0 is selected by default.

Parameter description:

NT	Normalization transformation index (0<NT<1000000). The number of transformations is limited to 50.

3.2.6 Simplified way to define the viewing pipeline

Very often the user of a graphics package wants to define the dimensions of the physical output in centimeters and centered on the output devices (screen or paper). This can be done with HIGZ with simply one call to the routine IGRNG.

```
CALL IGRNG (XSIZE,YSIZE)
```

Action: This routine is used to determine the physical dimensions (in centimeter) and to optimize the aspect ratio and the centering of a picture. If the X or Y dimension of output device are smaller than XSIZE or YSIZE, a scaling factor is applied to the final size of the picture but the aspect ratio is kept. When an Encapsulated PostScript workstation is active, a call to this routine is mandatory in order to define the size of the picture (e.g the PostScript BoundingBox).

Parameter description:

XSIZE	Picture size in centimeters in the X direction.
YSIZE	Picture size in centimeters in the Y direction.

After a call to IGRNG the normalization transformation number 1 is selected. For this reason in all the HIGZ routines, the normalization transformation number 1 is assumed to be a centimeter transformation. It is not recommended to define this transformation (via ISWN, ISVP and ISELNT) outside IGRNG. In particular when PostScript files are used, the PostScript driver assumes that the setting of the normalization transformation 1 has been done via IGRNG.

After a call to IGRNG some useful value to convert centimeters into normalized device coordinates, are available in the common QUEST.

RQUEST(11)	Ratio to convert cm into normalized device coordinates.
RQUEST(12)	left position of the normalization transformation 1 viewport in normalized device coordinates.
RQUEST(13)	bottom position of the normalization transformation 1 viewport in normalized device coordinates.
RQUEST(14)	width of the normalization transformation 1 viewport in normalized device coordinates.
RQUEST(15)	height of the normalization transformation 1 viewport in normalized device coordinates.

For more details, see examples on pages 126 and 135.

3.3 Metafile control and printing

A special ASCII file called metafile is needed in order to produce pictures on paper. The metafiles are managed via all workstation control routines previously described. The general sequence of actions to use metafiles is:

```
- Open a FORTRAN file
- Open a workstation (IOPWK) with the type metafile
- Activate the workstation
- Produce some graphics
- Deactivate the workstation
- Close the workstation
```

3.3.1 Simplified metafile control

The routine IGMETA is provided in order to minimize the number of calls to specialized HIGZ workstation control routines and to improve the portability of applications. This routine opens, activates, deactivates or closes a metafile.

```
CALL IGMETA (LUN,KWTYPE)
```

Action: This routine permits the selection of a metafile, offering a choice of graphic output to the screen and/or a metafile.

Parameter description:

LUN Metafile logical unit number

LUN>0	The subsequent graphic output will be directed to both screen and metafile.
LUN<0	The subsequent graphic output will be directed to the metafile only.
LUN=0	Any previously open metafile is deactivated, and further graphic output will be directed to the screen only.
LUN=999	Any previously open metafile is deactivated and closed, and further graphic output will be directed to the screen only. PostScript metafiles need to be closed in order to be printed.

KWTYPE Workstation type. If KWTYPE = 0, then IGMETA selects automatically the default workstation type. This defaults workstations depend on the underlying graphics package used (e.g. -111 for HIGZ/X11 or 4 for GKS-GRAL).

3.3.2 PostScript metafile type

In addition to the metafile type provided by the underlaying graphics package (for example 4 with GKS-GRAL), PostScript workstation types are also available independently from the underlying graphics package used allowing generation of high quality outputs. The PostScript workstation types have the following format:

`-[Format][Nx][Ny][Type]`

Where:

`Format` Is an integer between 0 and 99 which defines the format of the paper.
Example: if `Format`=3 the paper is in the standard A3 format. `Format`=4 and `Format`=0 are the same and define an A4 page. The A0 format is selected by `Format`=99. The US format Letter is selected by `Format`=100. The US format Legal is selected by `Format`=200. The US format Ledger is selected by `Format`=300.

`Nx, Ny` Specify respectively the number of zones on the x and y axis. `Nx` and `Ny` are integers between 1 and 9.

`Type` Can be equal to:

- 1 Portrait mode with a small margin at the bottom of the page.
- 2 Landscape mode with a small margin at the bottom of the page.
- 4 Portrait mode with a large margin at the bottom of the page.
- 5 Landscape mode with a large margin at the bottom of the page.
  The large margin is useful for some PostScript printers (very often for the colour printers) as they need more space to grip the paper for mechanical reasons.
  Note that some PostScript colour printers can also use the so called "special A4" format permitting the full usage of the A4 area; in this case larger margins are not necessary and `Type`=1 or 2 can be used.
- 3 Encapsulated PostScript. This `Type` permits the generation of files which can be included in other documents, for example in LaTeX files. Note that with this `Type`, `Nx` and `Ny` must always be equal to 1, and `Format` has no meaning. The size of the picture must be specified by the user via the `IGRNG` routine. Therefore the workstation type for Encapsulated PostScript is -113. For example if the name of an Encapsulated PostScript file is `example.eps`, the inclusion of this file into a LaTeX file will be possible via (in the LaTeX file):

```
\begin{figure}
\epsffile{example.eps}
\caption{Example of Encapsulated PostScript in LaTeX.}
\label{EXAMPLE}
\end{figure}
```

  Note that all the figures in this manual are included in this way.

With `Type=1,2,4` and 5 the pictures are centered on the page, and the usable area on paper is proportional to the dimensions of A4 format.

Examples:

`-111` or `-4111` defines an A4 page not divided. `-6322` define an A6 landscape page divided in 3 columns and 2 rows.

1	2	3
4	5	6

The first picture will be drawn in the area 1. If the program clears the screen via ICLRWK, the graphics output will appear in the next area in the order defined above. If a page is filled, a new page is used with the same grid. Note that empty pages are not printed in order to save paper.

Ignoring formats smaller than A12, the total number of possible different PostScript workstation types is: $4 \times 9 \times 9 \times 13 + 1 = 4213$!

3.3.3 Usage of PostScript metafiles in an user application program

This section gives three examples showing the different ways of managing PostScript files. The first example is the more general way, using IOPWK, IACWK and IGQWK (see section 6.2). The second example shows how to use the IGMETA routine. The last example use IGRNG and IGMETA.

Example 1 : IOPWK, IACWK and IGQWK

```
      DIMENSION R(2)
*
*              Open a Fortran file. Note that on VAX/VMS machines
*              a CARRIAGECONTROL='LIST' in the open statement is needed.
*
      OPEN(UNIT=10,FILE='test1.ps',FORM='FORMATTED',STATUS='UNKNOWN')
*
*              Open and activate a workstation with the PostScript metafile
*              type -111 and with the workstation ID 5.
*              Note that the UNIT used to open the Fortran (here 10)
*              is given as second parameter.
*
      CALL IOPWK(5,10,-111)
      CALL IACWK(5)
*
*              Get the size of the available space on paper. This is
*              now possible because the Format is known.
*
      CALL IGQWK(5,'MXDS',R)
*
*              Compute the size of the viewport according to the paper
*              size. Note that if the screen has not the same RATIO the
*              picture on screen and on paper will be different. In this
*              case the user must inquire the screen size and compute
*              a new viewport with this size and redraw on the screen
*              with the metafile deactivated.
*
      XV=R(1)/R(2)/2.
      YV=XV
      CALL ISVP(2,0.,XV,0.,YV)
      CALL ISWN(2,X1,X2,Y1,Y2)
      CALL ISELNT(2)
          .
          .
        Drawing
          .
```

```
          .
*
*             Deactivate and close the metafile
*
      CALL IDAWK(5)
      CALL ICLWK(5)
      CLOSE(10)
```

Example 2 : `IGMETA`

```
      DIMENSION R(2)
      OPEN(UNIT=10,FILE='test2.ps',FORM='FORMATTED',STATUS='UNKNOWN')
*
*             IGMETA permits the opening and activating of the metafile
*
      CALL IGMETA(10,-111)
      CALL IGQWK(2,'MXDS',R)
      XV=MIN(1.,R(1)/R(2))
      YV=MIN(1.,R(2)/R(1))
      CALL ISVP(2,0.,XV,0.,YV)
      CALL ISWN(2,X1,X2,Y1,Y2)
      CALL ISELNT(2)
          .
          .
        Drawing
          .
          .
*
*             Deactivate the metafile
*
      CALL IGMETA(0,0)
*
*             Close the metafile
*
      CALL ICLWK(2)
      CLOSE(10)
```

Example 3 : `IGRNG`

```
      DIMENSION R(2)
      OPEN(UNIT=10,FILE='test4.ps',FORM='FORMATTED',STATUS='UNKNOWN')
      CALL IGMETA(-10,-111)
*
*             IGRNG defines a size in cm centered on the page.
*             Even if the RATIO of the screen and the RATIO of
*             the paper are not the same the picture will appear
*             exactly the same on both.
*             Note that in the case of Encapsulated PostScript~(-113)
*             a call to IGRNG is mandatory.
*
```

```
      CALL IGRNG(10.,10.)
          .
          .
        Drawing
          .
          .
      CALL IGMETA(0,0)
      CALL ICLWK(2)
      CLOSE(10)
```

3.3.4 LaTeX metafile type

HIGZ is able to produce metafiles which are ready to be included in LaTeX documents. These metafiles make use of the `\picture` environment. Compared to other possibilities of merging graphics into documents, LaTeX metafiles have a number of advantages:

- The `dvi` file is fully transportable as `\special` commands are not used. This file can be output on any device for which a driver exists. Documents can be written, formatted, and previewed on workstations while the `dvi` file can be sent via the network to a central server for printing.
- The metafile can be also merged into the LaTeX file to keep the full document in a single file.
- The power of LaTeX in text processing can be used in the primitive `ITX` for example to generate complicated mathematical formulae on a document.

LaTeX metafile capabilities

The capabilities of the `\picture` environment are basically limited to drawing straight horizontal or vertical lines. Slanted lines do exist but only in a limited number of slopes and a minimum length of $\approx 4\mathrm{mm}$. Therefore slanted lines have to be approximated by small steps of straight lines where the step size should be close to the printer resolution.

The workstation type for LaTeX metafiles is -777 for embedded files or -778 for stand-alone files. Coordinates written to the metafile are integer numbers assuming a grid spacing of $0.1\mathrm{mm}$. Therefore the settings for `XSIZE` and `YSIZE` should approximately correspond to the final picture size.

Line and marker types

Line types 1 through 4 and marker types 1 through 5 are supported.

Text fonts

In addition to the software characters the font numbers -1 through -8 at precision 0 can be used. They map to the TeX fonts Roman, *Emphatic*, **Bold**, *Italic*, *Slanted*, Sans Serif, SMALL CAPS, and `Typewriter`, respectively.

TeX fonts look much nicer and are faster to generate than software characters generated by `IGTEXT`, but the disadvantage is that they are available in horizontal orientation only and the character size does not scale with the picture size.

When using TeX fonts the `IGTEXT` control characters “`<>[]"#^?!`” are interpreted to obtain superscripts, greek letters, and other special characters. If a text string contains a “`\`” or “`{`” the remaining part is written verbatim into the metafile. This allows to use TEX formatting commands for elaborate displays. Of course "{" and "}" must be properly matched.

The whole text is typeset in math mode which does not allow a change of fontsize in between. In order to format a formula on a larger size the formula text must be preceded by "`{}$\large$`".

Configuration parameters

To some extent, the appearance of a picture can be changed at formatting time by defining configuration parameters (in the LATEX file) which have the following default values:

```
\newdimen\higzunit \higzunit=0pt
\newcount\higzstep \higzstep=2
\newcount\higzdraft \higzdraft=0
```

By default the picture is automatically scaled to fill the full page width. The picture size can be changed by setting `\higzunit` to the wanted grid spacing, e.g. to get the true `XSIZE`:

```
\newdimen\higzunit \higzunit=0.1mm
```

Slanted lines are approximated by straight lines along the major axis. The step size along the minor axis is `\higzstep`×`\unitlength`. By setting `\higzstep=1` curves will look smoother but if line segments come too close to the printer resolution the `dvi` driver may choose not to display them. A larger value will result in faster formatting requiring less TEX memory.

Setting `\higzdraft=1` replaces the actual picture by an empty box of the same size to save formatting time during drafting.

3.4 Examples: the routines START and FINISH

The two routines used to produce the figures appearing this manual are described in this section. They are good examples of a simple, but frequent, usage of HIGZ.

The first one: `START`, initializes HIGZ, opens an Encapsulated PostScript file and set the size of the figure according to the input parameters.

The second one: `FINISH`, closes the Encapsulated PostScript file and terminates HIGZ.

The routine START

```
      SUBROUTINE START(NAME,X,Y)
      CHARACTER*(*) NAME
      PARAMETER (NWORDS=50000)
      COMMON /PAWC/ RPAW(NWORDS)
      CALL MZEBRA(-3)
      CALL MZPAW(NWORDS,' ')
      CALL IGINIT(0)
      CALL IGWKTY(ITYPE)
      CALL IGSSE(6,ITYPE)
      OPEN(UNIT=10,FILE=NAME//'.EPS',FORM='FORMATTED',STATUS='UNKNOWN')
      CALL IGMETA(10,-113)
      CALL IGRNG(X,Y)
      END
```

In the routine FINISH the call to the routine IGTERM is not mandatory but is useful to flush the graphics buffer especially in the case of the X11 interface (see section 3.1.9).

The routine FINISH

```
      SUBROUTINE FINISH
      CALL IGMETA(0,0)
      CALL ICLWK(2)
      CLOSE(10)
      CALL IGTERM
      CALL IGEND
      END
```

3.5 The basic output primitives

In HIGZ there are four basic output primitives: the polyline (IPL), the fill area (IFA), the polymarker (IPM) and the text (ITX). In all routines described in this section the coordinates are given in the world coordinates system.

3.5.1 Polyline

GKS
```
CALL IPL (N,X,Y)
```

Action: This routine draws a polyline on the currently active workstations (there must be at least one). The polyline connects N points (N$\geq$2) by means of N-1 line segments. The X and Y coordinates of the points are in two N-dimensional arrays.

The appearance of a polyline is controlled by the current "polyline colour index" (see ISPLCI section 3.6.2), the current "line type" (see ISLN section 3.6.5) and the current "line width" (see ISLWSC section 3.6.6).

Parameter description:

N	Number of points.
X	Array of dimension N containing the x coordinates in WC space.
Y	Array of dimension N containing the y coordinates in WC space.

3.5.2 Multiline

```
CALL IML (N,X,Y)
```

Action: This routine draws a multiline on the currently active workstations (there must be at least one). The multiline connects N points (N$\geq$2) two by two. The X and Y coordinates of the points are in two N-dimensional arrays.

The appearance of a multiline is controlled by the current "polyline colour index" (see ISPLCI section 3.6.2), the current "line type" (see ISLN section 3.6.5) and the current "line width" (see ISLWSC section 3.6.6).

Parameter description:

N	Number of points.
X	Array of dimension N containing the x coordinates in WC space.
Y	Array of dimension N containing the y coordinates in WC space.

3.5.3 Polymarker

GKS `CALL IPM (N,X,Y)`

Action: This routine draws a polymarker on the currently active workstations (there must be at least one). Markers are placed at `N` points ($N \geq 1$), whose x and y coordinates are given in two `N`-dimensional arrays.

The appearance of a polymarker is controlled by the current "polymarker colour index" (see `ISPMCI` section 3.6.2), the current "marker type" (see `ISMK` section 3.6.7) and the current "marker scale factor" (see `ISMKSC` section 3.6.8).

Parameter description:

`N`	Number of points.
`X`	Array of dimension `N` containing the x coordinates in WC space.
`Y`	Array of dimension `N` containing the y coordinates in WC space.

3.5.4 Fill area

GKS `CALL IFA (N,X,Y)`

Action: This routine draws a filled area on the currently active workstations (there must be at least one). The "perimeter" of the filled area has `N` points ($N \geq 3$) whose x and y coordinates are given in two `N`-dimensional arrays.

The appearance of a filled area is controlled by the current "filled area colour index" (see `ISFACI` section 3.6.2), the current "filled area interior style" (see `ISFAIS` section 3.6.3) and the current "filled area style index" (see `ISFASI` section 3.6.3).

Parameter description:

`N`	Number of points.
`X`	Array of dimension `N` containing the x coordinates in WC space.
`Y`	Array of dimension `N` containing the y coordinates in WC space.

3.5.5 Text

GKS `CALL ITX (X,Y,CHARS)`

Action: This routine draws a text string on the currently active workstations (there must be at least one).

The appearance of the text is controlled by attributes set by the current "text colour index" (see `ISTXCI` section 3.6.2), the current "character height" (see `ISCHH` section 3.6.10), the current "text orientation" (see `ISCHUP` section 3.6.11 and the option `TANG` of the routine `IGSET` section 4.11), the current "text alignment" (see `ISTXAL` section 3.6.9) and the current "text font and precision" (see `ISTXFP` section 3.6.12).

Parameter description:

`X` X coordinate in WC space.

`Y` Y coordinate in WC space.

`CHARS` `CHARACTER` variable containing the text to be displayed. Only the following characters are allowed to appear in `CHARS`:

```
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[-]^_
abcdefghijklmnopqrstuvwxyz{|}~
and the space.
```

Software characters (i.e. drawn with lines and not provided by the hardware) can be produced with routine `IGTEXT`.

3.6 The output attributes

3.6.1 Clipping

GKS `CALL` **ISCLIP** `(ICLSW)`

Action: This routine sets the "clipping indicator" for use by future invocations of `IFA`, `IPL`, `IPM` and `ITX`. The clipping indicator specifies where primitives should be clipped.

Parameter description:

`ICLSW` Clipping indicator

- `1` Primitives should be clipped at the boundary of the normalization transformation viewport.
- `0` Primitives should be clipped at the edge of the normalized device coordinates space.

3.6.2 Colour management

Colour representation

Each colour is defined by an index and percentages of red, green and blue. Once a colour is defined it can be used via a reference to its index. If a requested colour index is not available on a workstation, colour index `1` is used when primitives are created.

GKS `CALL` **ISCR** `(KWKID,ICI,CR,CG,CB)`

Action: This routine sets the colour representation (red/green/blue) of the colour index on a previously opened workstation. On workstations using colour tables, this function can change the image immediately. On workstations lacking such tables, this new colour definition will be taken into account in the next use of this colour.

Parameter description:

`KWKID` Workstation identifier

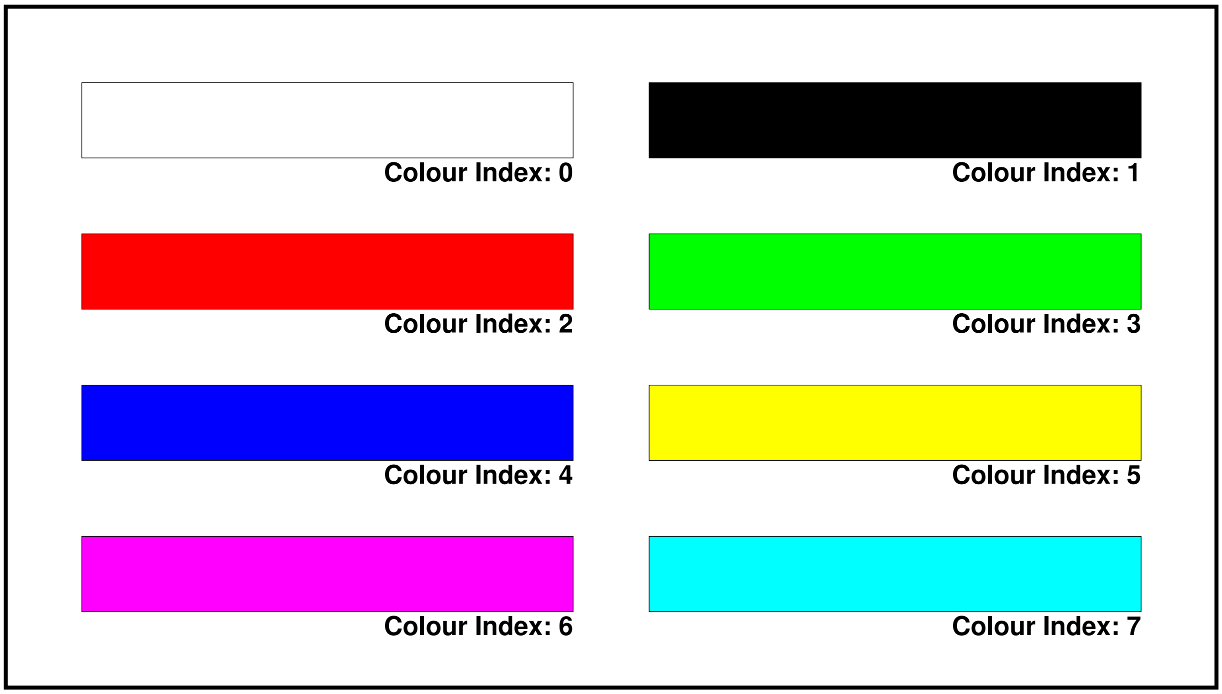


Figure 3.2: PostScript grey level simulation of the eight basic colours.

`ICI`	Colour index.
`CR`	Intensity of red `0.`$\leq$`CR`$\leq$`1.`
`CG`	Intensity of green `0.`$\leq$`CG`$\leq$`1.`
`CB`	Intensity of blue `0.`$\leq$`CB`$\leq$`1.`

By default `IGSSE` initialize the first eight colour indices are defined as follows:

Index	Colour	Red	Green	Blue
0	Background colour (White)	1.	1.	1.
1	Foreground colour (Black)	0.	0.	0.
2	Red	1.	1.	1.
3	Green	0.	1.	0.
4	Dark blue	0.	0.	1.
5	Yellow	1.	1.	0.
6	Magenta (red-purple)	1.	0.	1.
7	Cyan (light blue)	0.	1.	1.

When a PostScript file is printed on a black and white PostScript printer, a grey level simulation of the colours is used according to the figure 3.2.

Polyline colour index.

GKS `CALL` **ISPLCI** `(ICOLI)`

Action: This routine sets the polyline colour index attribute for use by future invocations of `IPL`. The routine `IGSET` (see section 4.11) can also be used with the parameter `PLCI`.

Parameter description:

`ICOLI` Polyline colour index.

Polymarker colour index.

GKS `CALL` **ISPMCI** `(ICOLI)`

Action: This routine sets the polymarker colour index attribute for use by future invocations of `IPM`. The routine `IGSET` (see section 4.11) can also be used with the parameter `PMCI`.

Parameter description:

`ICOLI` Polymarker colour index.

Fill area colour index.

GKS `CALL` **ISFACI** `(ICOLI)`

Action: This routine sets the fill area colour index attribute for use by future invocations of `IFA`. The routine `IGSET` (see section 4.11) can also be used with the parameter `FACI`.

Parameter description:

`ICOLI` Fill area colour index.

Text colour index.

GKS `CALL` **ISTXCI** `(ICOLI)`

Action: This routine sets the text colour index attribute for use by future invocations of `ITX`. The routine `IGSET` (see section 4.11) can also be used with the parameter `TXCI`.

Parameter description:

`ICOLI` Text colour index.

3.6.3 Fill area interior style

GKS `CALL` **ISFAIS** `(INTS)`

Action: This routine sets the fill area interior style attribute for use by future invocations of `IFA`. The routine `IGSET` (see section 4.11) can also be used with the parameter `FAIS`.

Parameter description:

`INTS` Fill area interior style. Possible values are:

- `0` Hollow: the perimeter of the filled area, after clipping, is drawn using solid lines.
- `1` Solid: the area is filled solidly.
- `2` Pattern: the area is filled with a dot-dashed pattern.
- `3` Hatched: the area is filled according to the current value of the fill area style index.

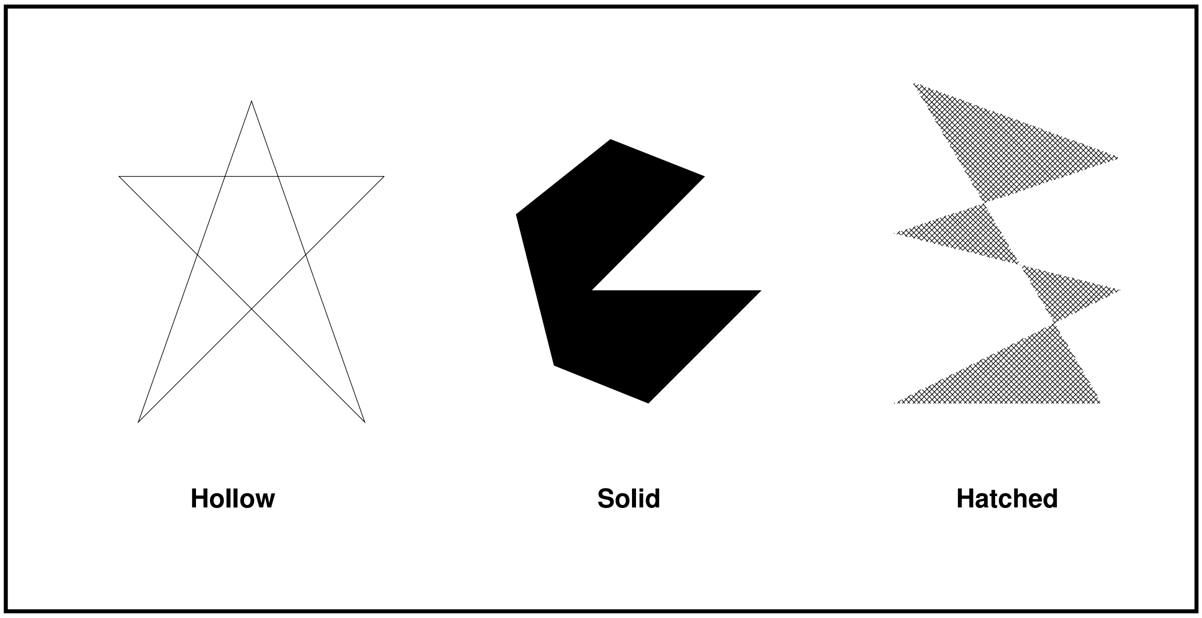


Figure 3.3: Example of fill area interior style.

3.6.4 Fill area style index.

GKS `CALL` **ISFASI** `(ISTYLI)`

Action: This routine sets the fill area style index for pattern and hatch styles. The routine `IGSET` (see section 4.11) can also be used with the parameter `FASI`.

Parameter description:

`ISTYLI` Fill area style index. This value depends on the underlying graphics package used.

In addition to the underlying graphics package dependent Fill area style indices, HIGZ provides a set of hatches independent from the underlying graphics package used. This fill area styles are indicated by a value greater than 100. The fill area style index is coded on three digits `ijk`.

`i` Distance between lines in the hatch.

`j` Angle between 90 and 180 degrees.

`k` Angle between 0 and 90 degrees.

Digit i	Distance	Digit j	Angle	Digit k	Angle
		0	180 deg	0	0 deg
1	$\approx$ 0.75 mm	1	170 deg	1	10 deg
2	$\approx$ 1.50 mm	2	160 deg	2	20 deg
3	$\approx$ 2.25 mm	3	150 deg	3	30 deg
4	$\approx$ 3.00 mm	4	135 deg	4	45 deg
5	$\approx$ 3.75 mm	5	not drawn	5	not drawn
6	$\approx$ 4.50 mm	6	120 deg	6	60 deg
7	$\approx$ 5.25 mm	7	110 deg	7	70 deg
8	$\approx$ 6.00 mm	8	100 deg	8	80 deg
9	$\approx$ 6.75 mm	9	90 deg	9	90 deg

For example `190` will set the interior of fill areas to be hatched with lines at 0 and 90 degrees ($\approx$ 0.75 mm spacing) and `444` will set the interior of fill areas to be hatched with lines at +45 and -45 degrees ($\approx$ 3 mm spacing).

The figure 3.4 shows some examples of HIGZ portable hatch styles. On this figure, the first column shows the nine different possible spacing (digit `i`), the second column shows the angle between `90` and `180` degrees (digit `j`), and the third column shows the angle between `0` and `90` degrees (digit `k`).

The number of possible hatch styles is: $9 \times 10 \times 10 = 900$.

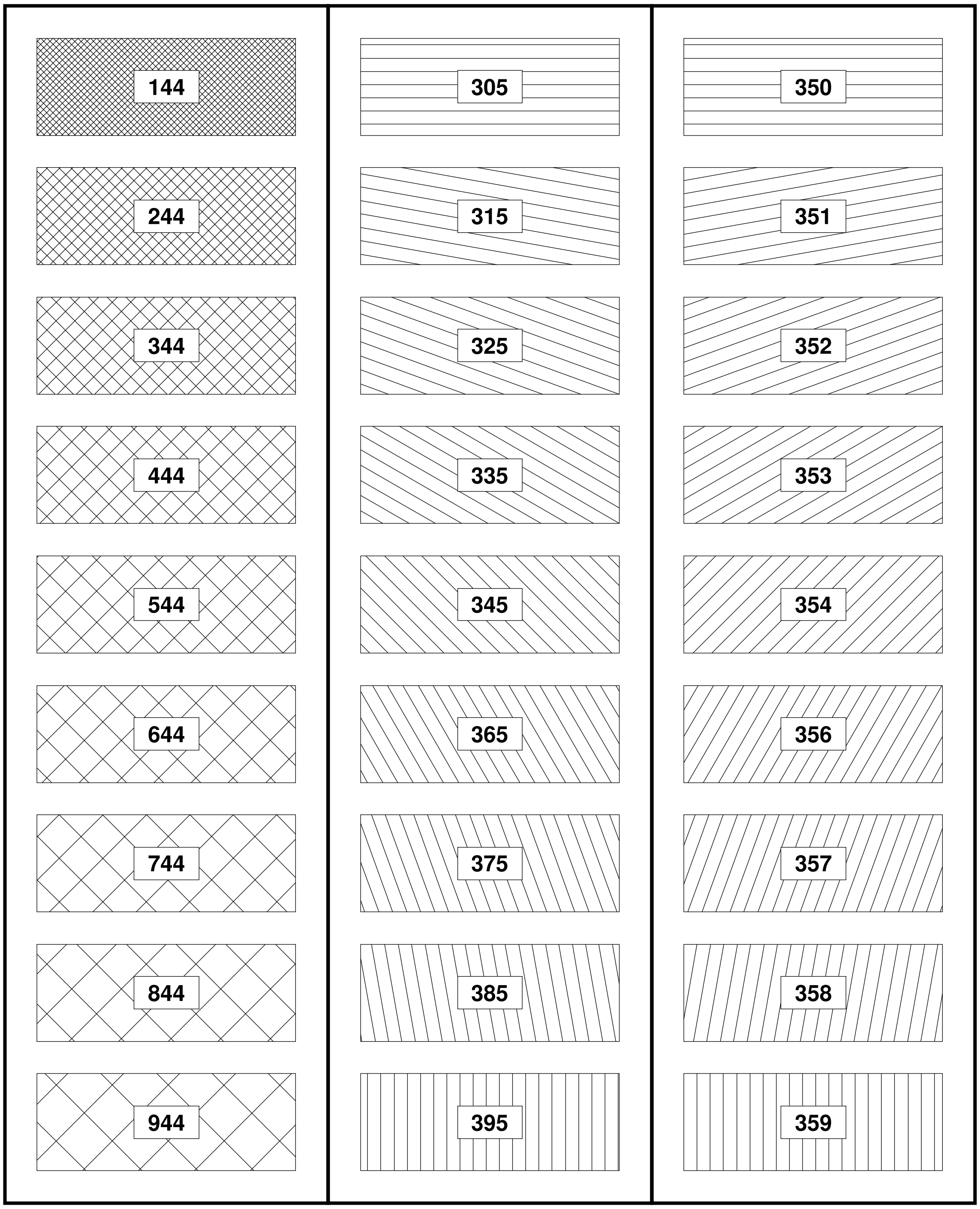


Figure 3.4: HIGZ portable fill area hatch styles.

3.6.5 Line type.

GKS `CALL ISLN (LTYPE)`

Action: This routine sets the line type attribute for use by future invocations of IPL. All workstations support at least line types 1 through 4 (see figure 3.5). Other line types may be supported. If a requested line type is not supported on a workstation, line type 1 is used when polylines are created. The routine IGSET (see section 4.11) can also be used with the parameter LTYP.

Parameter description:

LTYPE Line type (positive number).

- 1 Solid lines
- 2 Dashed lines
- 3 Dotted lines
- 4 Dashed-dotted lines

Note that line type values are dependent upon the underlying graphics package used. For the user's convenience, HIGZ defines a number of line types, indicated in the figure 3.5, which are independent from the basic graphics package used.

3.6.6 Line width scale factor.

GKS `CALL ISLWSC (WIDTH)`

Action: This routine sets the width of a line for use by future invocations of the polyline drawing routine IPL. The actual line width is determined by a nominal line width (workstation-dependent) multiplied by the line width scale factor. The nominal line width is one pixel on screens. On PostScript printers the nominal line width is one "dot". Therefore the width of a line can vary from a printer to another depending on the printer definition (300 dots per inch, 400 dots per inch etc.). The figure 3.6 shows some examples of various line width. The routine IGSET (see section 4.11) can also be used with the parameter LWID.

Parameter description:

WIDTH Line width scale factor.

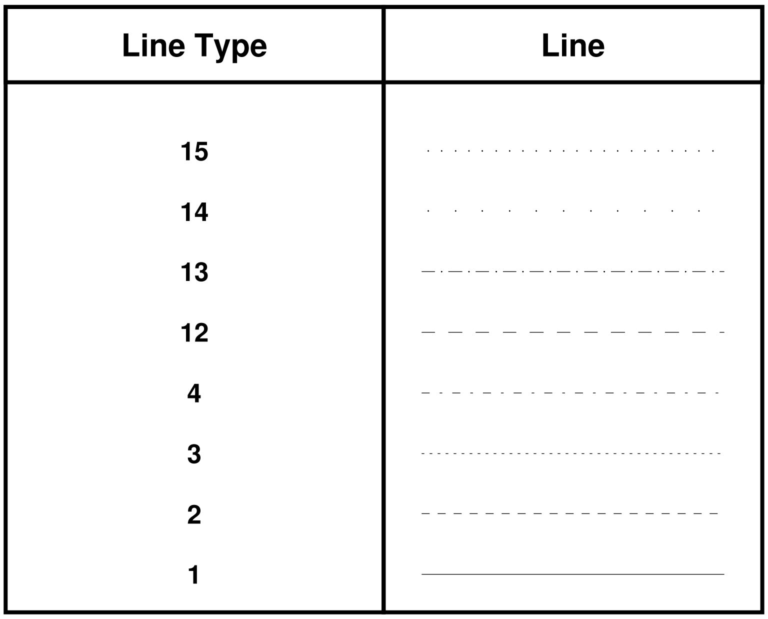


Figure 3.5: Line styles available.

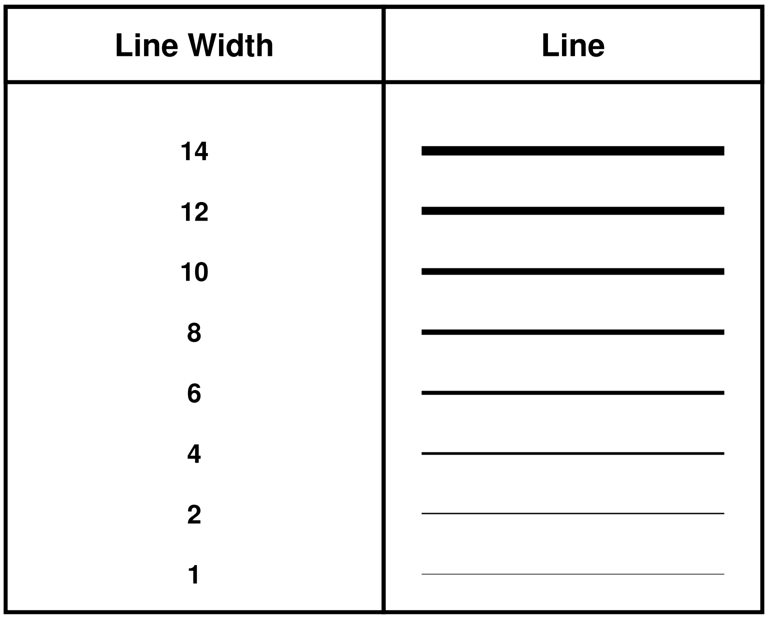


Figure 3.6: Examples of line width.

3.6.7 Marker type

GKS `CALL ISMK (MTYPE)`

Action: This routine sets the marker type attribute for use by future invocations of IPM. All workstations support at least the marker types 1 through 5 (see below). More marker types may be supported by the underlying graphics package. Marker types 20 to 31 are also defined, according to the figure 3.7, and are independent from the underlying graphics package used. If a requested marker type is not supported on a workstation, marker type 1 (a point) is used when polymarkers are created. The routine IGSET (see section 4.11) can also be used with the parameter MTYP.

Parameter description:

MTYPE Marker type (positive number)

- 1 Point shape ($\cdot$).
- 2 Plus shape ($+$).
- 3 Asterisk shape ($*$).
- 4 Circle shape ($\circ$).
- 5 X shape ($\times$).

3.6.8 Marker scale factor.

GKS `CALL ISMKSC (SSFM)`

Action: This routine sets the marker scale factor. This scale factor is applied on the nominal size of the marker. On all workstation, except PostScript files, the marker type 1 is not scalable. The routine IGSET (see section 4.11) can also be used with the parameter MSCF.

Parameter description:

SSFM Scale factor applied to markers. (≥ 0.)

Marker Type	Marker
31	✳
30	☆
29	★
28	✙
27	◇
26	△
25	□
24	○
23	▼
22	▲
21	■
20	●

Figure 3.7: HIGZ Marker type (20-31).

Marker Scale Factor	Marker
5	★
4	★
3	★
2	★
1	★

Figure 3.8: Examples marker scale factor.

3.6.9 Text alignment.

GKS `CALL **ISTXAL** (ITXALH,ITXALV)`

Action: This routine sets the text alignment attribute for use by future invocations of ITX. Text alignment controls the placement of the character string with respect to the position specified in the call to ITX. Horizontal alignment specifies which end of the string (or its geometric center) is aligned with the point specified in ITX. For a given horizontal alignment, the vertical alignment controls whether the tops of tall characters or the bottoms of capital letters line up with the point specified (see figure 3.9). The routine IGSET (see section 4.11) can also be used with the parameter TXAL.

Parameter description:

ITXALH Horizontal alignment specifier (0≤ITXALH≤3)

- 0 Left end of string at point specified (normal).
- 1 Same as 0.
- 2 Center of string at point specified.
- 3 Right end of string at point specified.

ITXALV Vertical alignment specifier (0≤ITXALV≤5)

- 0 Base of the characters (normal).
- 1 Top of tallest characters.
- 2 Same as 2.
- 3 Middle of tallest characters.

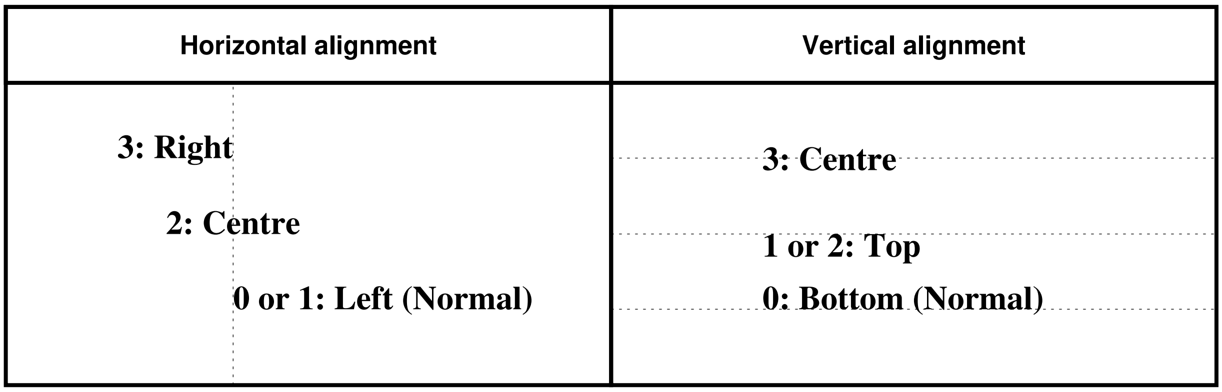


Figure 3.9: Text alignment.

3.6.10 Character height

GKS `CALL **ISCHH** (CHH)`

Action: This routine sets the character height attribute for use by future invocations of `ITX`. The routine `IGSET` (see section 4.11) can also be used with the parameter `CHHE`.

Parameter description:

`CHH` Character height. The default set by `IGSSE` is `0.01`. The height is given in world coordinates and it must be positive.

3.6.11 Character up vector.

GKS `CALL **ISCHUP** (RCHUX, RCHUY)`

Action: This routine sets the "character up vector" attribute for use by future invocations of `ITX`. The angle of the text can also be specified via the `IGSET` routine with the parameter `TANG`.

Parameter description:

`RCHUX` Character up vector in world coordinates (x part).

`RCHUY` Character up vector in world coordinates (y part).

The size of the vector specified is immaterial, but $\mathtt{CHUX}^2\mathtt{+CHUY}^2\mathtt{>1.E-20}$.

3.6.12 Text font and precision.

GKS `CALL **ISTXFP** (IFONT,IPREC)`

Action: This routine sets the text font and precision attributes for use by future invocations of `ITX`. The text font parameter selects among possible character shapes, as a roman font, a sans-serif font, etc. The text precision parameter specifies how closely HIGZ (and also the underlying graphics package) must follow the current size and orientation attributes. String precision is most liberal, stroke precision is most strict. Character precision is in the middle. The routine `IGSET` (see section 4.11) can also be used with the parameter `TXFP`.

Parameter description:

`IFONT` Text font. The value of `IFONT` depends on the underlying graphics package used.

`IPREC` Text precision ($0 \leq \mathtt{IPREC} \leq 2$).

Note that font number 0, with precision 2, is always available, independently from the underlying graphics package used and allows to access the `IGTEXT` facilities from `ITX`. If a font is not available on a workstation, or it is supported but not with the requested precision, font 1 is used, with precision 0.

The PostScript text fonts

With PostScript workstation types, the text produced by ITX can be generated with PostScript fonts. The figure 3.12 shows all the PostScript fonts available on most PostScript printers. Note that the fonts -15 to -24 are the same than -1 to -14, but they are drawn in hollow mode.

The ZapfDingbats font is not available on all PostScript printers. On such printers a reference to this font will produce an error message. The correspondence between ASCII and ZapfDingbats font is given on figures 3.13 and 3.14. IGTEXT control characters are taken into account. In addition the character ~ switches to the ZapfDingbats character set.

List of escape characters and their meaning			
<	go to lower case (optional)	>	go to upper case (optional)
[	go to greek (Roman = default)	]	end of greek
”	go to special symbols	#	end of special symbols
~	go to ZapfDingbats	#	end of ZapfDingbats
↑	go to superscript	?	go to subscript
!	go to normal level of script	&	backspace one character
$	termination character (optional)	@	escape

The PostScript fonts can be used with precision 0 or precision 1. On the screen, a PostScript font used with precision 1 appears like the IGTEXT characters, with precision 0 its appears as hardware character (X11 fonts). In both cases the PostScript file is the same. Note that characters can also be entered directly in lower or upper case instead of using the escape characters < and >. The escape characters can be themselves escaped with the @ sign i.e: '@<' produce a < symbol.

Example of PostScript text (result in figure 3.10)

```
      program psex1
      call start('psex1',16.,5.)
      call igset('LWID',6.)
      call igbox(0.,16.,0.,5.)
      call igset('CHHE',0.5)
      call igset('TXAL',3.)
      call igset('TXFP',-130.)
*
      call itx(3.,4.,'K\355nstler in den gr\345\373ten st\311dten')
      call itx(3.,3.,'\253\265 l''\372uvre on conna\333t l''artisan\273')
      call itx(3.,2.,'\(proverbe fran\321ais\).')
      call itx(3.,1.,
     + '\252\241Ma\337ana\41 \322ag&\306!das&\313!\272, '
     + //'dit l''\3231\325ve.')
*
      call finish
      end
```

Künstler in den größten Städten
«À l’œuvre on connaît l’artisan»
(proverbe français).
“¡Mañana! Çağdaş”, dit l’élève.

Figure 3.10: PostScript fonts usage (1).

Example of PostScript text and maths (result in figure 3.11)

```
      program psex2
      call start('psex2',16.,5.)
      call igset('LWID',6.)
      call igbox(0.,16.,0.,5.)
      call igset('CHHE',0.5)
      call igset('TXAL',23.)
      call igset('TXFP',-130.)
*
      call itx(8.,4.,
     +'e^+!e^-! "5# Z^o! "5# ll&^-!, qq&^\261!')
      call itx(8.,3.,
     +'| a&^[\256]! \267 b&^[\256]! | = [\345] a^i?jk!+b^kj?i')
      call itx(8.,2.,
     + 'i ("d#?[m!y]&^\261![g^m]! + m [y]&^\261! ) = 0'//
     + '" r# (~r# + m^2!) [y] = 0')
      call itx(8.,1.,
     + 'L?em! = e J^[m]?em! A?[m]! , J^[m]?em!=l&^\261![ g?m]!l , '//
     + 'M^j?i! = [\345&?a]! A?[a! t^a]j?i! ')
*
      call finish
      end
```

$$e^{+}e^{-} \rightarrow Z^{o} \rightarrow l\bar{l},\ q\bar{q}$$

$$|\,\vec{a} \bullet \vec{b}\,| = \sum a^{i}_{jk} + b^{kj}_{i}$$

$$i\,(\partial_{\mu}\bar{\psi}\gamma^{\mu} + m\,\bar{\psi}\,) = 0 \Leftrightarrow (\Box + m^{2})\,\psi = 0$$

$$L_{em} = e\,J^{\mu}_{em}\,A_{\mu}\ ,\ J^{\mu}_{em} = \bar{l}\,\gamma_{\mu}l\ ,\ M^{j}_{i} = \sum_{\alpha} A_{\alpha}\,\tau^{\alpha j}_{i}$$

Figure 3.11: PostScript fonts usage (2).

Font/Prec	PostScript Font Style	
-1/0	*ABCDEFghijlk0123456789*	Times-Italic
-2/0	**ABCDEFghijlk0123456789**	Times-Bold
-3/0	***ABCDEFghijlk0123456789***	Times-BoldItalic
-4/0	ABCDEFghijlk0123456789	Helvetica
-5/0	*ABCDEFghijlk0123456789*	Helvetica-Oblique
-6/0	**ABCDEFghijlk0123456789**	Helvetica-Bold
-7/0	***ABCDEFghijlk0123456789***	Helvetica-BoldOblique
-8/0	ABCDEFghijlk0123456789	Courier
-9/0	*ABCDEFghijlk0123456789*	Courier-Oblique
-10/0	**ABCDEFghijlk0123456789**	Courier-Bold
-11/0	***ABCDEFghijlk0123456789***	Courier-BoldOblique
-12/0	ΑΒΗΔΕΦγχιιλκ0123456789	Symbol
-13/0	ABCDEFghijlk0123456789	Times-Roman
-14/0	✡✢✣✤✥✦❇❈❉❊●❋✐✑✒✓✔✕✖✗✘✙	ZapfDingbats
-15/0	*ABCDEFghijlk0123456789*	Times-Italic
-16/0	ABCDEFghijlk0123456789	Times-Bold
-17/0	*ABCDEFghijlk0123456789*	Times-BoldItalic
-18/0	ABCDEFghijlk0123456789	Helvetica
-19/0	*ABCDEFghijlk0123456789*	Helvetica-Oblique
-20/0	ABCDEFghijlk0123456789	Helvetica-Bold
-21/0	*ABCDEFghijlk0123456789*	Helvetica-BoldOblique
-22/0	ΑΒΧΔΕΦγηιφλκ0123456789	Symbol
-23/0	ABCDEFghijlk0123456789	Times-Roman
-24/0	✡✢✣✤✥✦❇❈❉❊●❋✐✑✒✓✔✕✖✗✘✙	ZapfDingbats

Figure 3.12: PostScript text fonts.

Input	*Upper Roman*	*Upper Greek*	*Upper Special*	*Upper Zapf*
A	A	Α	±	✡
B	B	Β	⌊	✢
C	C	Η	∃	✣
D	D	Δ	∀	✤
E	E	Ε	!	✥
F	F	Φ	#	✦
G	G	Γ	>	✧
H	H	Χ	?	★
I	I	Ι	∫	✩
J	J	Ι	:	✪
K	K	Κ	;	✫
L	L	Λ	<	✬
M	M	Μ	[	✭
N	N	Ν	]	✮
O	O	Ο	≥	✯
P	P	Π	{	✰
Q	Q	Θ	}	✱
R	R	Ρ	√	✲
S	S	Σ	♠	✳
T	T	Τ	♥	✴
U	U	Υ	♦	✵
V	V	Χ	♣	✶
W	W	Ω	≤	✷
X	X	Ξ	×	✸
Y	Y	Ψ	%	✹
Z	Z	Ζ	∞	✺
0	0	0	⊗	✐
1	1	1	⊕	✑
2	2	2	∅	✒
3	3	3	◊	✓
4	4	4	•	✔
5	5	5	→	✕
6	6	6	↑	✖
7	7	7	←	✗
8	8	8	↓	✘
9	9	9	↔	✙
.	.	.	·	✎
,	,	,	,	✌
+	+	+	+	☞
-	-	−	∠	✍
***	*	*	〈	☛
/	/	/	÷	✏
=	=	=	≠	✝
(	(	(	≡	✈
)	)	)	″	✉
{	{	{	{	❛
}	}	}	}	❜

Input	*Lower Roman*	*Lower Greek*	*Lower Special*	*Lower Zapf*
a	a	α	≈	❁
b	b	β	≅	❂
c	c	η	⊥	❃
d	d	δ	∂	❄
e	e	ε	ƒ	❅
f	f	ϕ	∩	❆
g	g	γ	∪	❇
h	h	χ	⊃	❈
i	i	ι	⊇	❉
j	j	ι	⊄	❊
k	k	κ	⊂	❋
l	l	λ	⊆	●
m	m	μ	∈	❍
n	n	ν	∉	■
o	o	ο	∇	❏
p	p	π	∧	❐
q	q	θ	∨	❑
r	r	ρ	⇔	❒
s	s	σ	⇐	▲
t	t	τ	⇑	▼
u	u	υ	⇒	◆
v	v	χ	⇓	❖
w	w	ω	&	◗
x	x	ξ	ϖ	❘
y	y	ψ	~	❙
z	z	ζ	ℵ	❚
:	:	:	:	✚
;	;	;	;	✛
**	\\	∴	∴	✼
_	_	_	_	✿
|	|	|	|	❜
%	%	%	%	☎
\047	’	∍	∍	✇
\074	<	<	<	✜
\076	>	>	>	✞
\133	[	[	[	✻
\135	]	]	]	✽
\042	"	∀	∀	✂
\043	#	#	#	✃
\136	^	⊥	⊥	✾
\077	?	?	?	✟
\041	!	!	!	✁
\046	&	&	&	✆
\044	$	∃	∃	✄
\176	~	~	~	❞

Figure 3.13: PostScript characters (1).

Input	*Upper Roman*	*Upper Greek*	*Upper Special*	*Upper Zapf*
\241	¡	ϒ	ϒ	❡
\242	¢	′	′	❢
\243	£	≤	≤	❣
\244	⁄	⁄	⁄	❤
\245	¥	∞	∞	❥
\246	ƒ	ƒ	ƒ	❦
\247	§	♣	♣	❧
\250	¤	♦	♦	♣
\251	'	♥	♥	♦
\252	“	♠	♠	♥
\253	«	↔	↔	♠
\254	‹	←	←	①
\255	›	↑	↑	②
\256	ﬁ	→	→	③
\257	ﬂ	↓	↓	④
\260	à	°	°	⑤
\261	–	±	±	⑥
\262	†	″	″	⑦
\263	‡	≥	≥	⑧
\264	·	×	×	⑨
\265	À	∝	∝	⑩
\266	¶	∂	∂	❶
\267	•	•	•	❷
\270	‚	÷	÷	❸
\271	„	≠	≠	❹
\272	”	≡	≡	❺
\273	»	≈	≈	❻
\274	…	…	…	❼
\275	‰	⏐	⏐	❽
\276	â	⎯	⎯	❾
\277	¿	↵	↵	❿
\300	Å	ℵ	ℵ	➀
\301	`	ℑ	ℑ	➁
\302	´	ℜ	ℜ	➂
\303	ˆ	℘	℘	➃
\304	˜	⊗	⊗	➄
\305	¯	⊕	⊕	➅
\306	˘	∅	∅	➆
\307	˙	∩	∩	➇
\310	¨	∪	∪	➈
\311	ä	⊃	⊃	➉
\312	˚	⊇	⊇	➊
\313		⊄	⊄	➋
\314	Å	⊂	⊂	➌
\315	˝	⊆	⊆	➍
\316	˛ˇ	∈	∈	➎
\317		∉	∉	➏

Input	*Lower Roman*	*Lower Greek*	*Lower Special*	*Lower Zapf*
\321	ç	∇	∇	➑
\322	Ç	®	®	➒
\323	é	©	©	➓
\324	É	™	™	➔
\325	è	∏	∏	→
\326	È	√	√	↔
\327	ê	⋅	⋅	↕
\330	Ê	¬	¬	➘
\331	ë	∧	∧	➙
\332	Ë	∨	∨	➚
\333	î	⇔	⇔	➛
\334	Î	⇐	⇐	➜
\335	ï	⇑	⇑	➝
\336	Ï	⇒	⇒	➞
\337	ñ	⇓	⇓	➟
\340	Ñ	◊	◊	➠
\341	Æ	〈	〈	➡
\342	ô	®	®	➢
\343	ª	©	©	➣
\344	Ô	™	™	➤
\345	ö	∑	∑	➥
\346	Ö	⎛	⎛	➦
\347	û	⎜	⎜	➧
\350	Ł	⎝	⎝	➨
\351	Ø	⎡	⎡	➩
\352	Œ	⎢	⎢	➪
\353	º	⎣	⎣	➫
\354	Û	⎧	⎧	➬
\355	ü	⎨	⎨	➭
\356	Ü	⎩	⎩	➮
\357	å	⎪	⎪	➯
\360				
\361	æ	〉	〉	➱
\362	Å	∫	∫	➲
\363	ÿ	⌠	⌠	➳
\364	Ÿ	⎮	⎮	➴
\365	ı	⌡	⌡	➵
\366	á	⎞	⎞	➶
\367	Á	⎟	⎟	➷
\370	ł	⎠	⎠	➸
\371	ø	⎤	⎤	➹
\372	œ	⎥	⎥	➺
\373	ß	⎦	⎦	➻
\374	ù	⎫	⎫	➼
\375	Ù	⎬	⎬	➽
\376		⎭	⎭	➾
\377				

Figure 3.14: PostScript characters (2).

Chapter 4: The graphic macroprimitives

In addition to the standard set of basic graphic primitives describe in the previous chapter, HIGZ provides also a set of graphic ”macroprimitives”. These graphic macroprimitives are included in HIGZ for three main reasons:

1. Functionality: it is easier to define a circle with its center and its radius than to compute all the necessary points to draw a polyline.
2. Precision, for instance a circle has to be stored as a circle and not as a sequence of polylines.
3. Compactness of the graphic data base.

4.1 Drawing a box

```
CALL IGBOX  (X1,X2,Y1,Y2)
```

Action: This routine fills a rectangle according to the “fill area colour index” (see section 3.6.2), the “fill area interior style” (see section 3.6.3), and the “fill area style index” (see section 3.6.4) attributes. The border is never drawn unless the interior style is hollow or the routine `IGSET` has been called with ’`BORD`’ and `VAL = 1.`. As it is shown on the figure 4.1, the border of the rectangle is drawn according to the values of the “line width scale factor” (see section 3.6.6) and the “polyline colour index” (see section 3.6.2) attributes, whereas the “line type” is always solid (see section 3.6.5).

Parameter description:

`X1`	X coordinate of 1st corner of the rectangle in WC.
`X2`	X coordinate of 2nd corner of the rectangle in WC.
`Y1`	Y coordinate of 1st corner of the rectangle in WC.
`Y2`	Y coordinate of 2nd corner of the rectangle in WC.

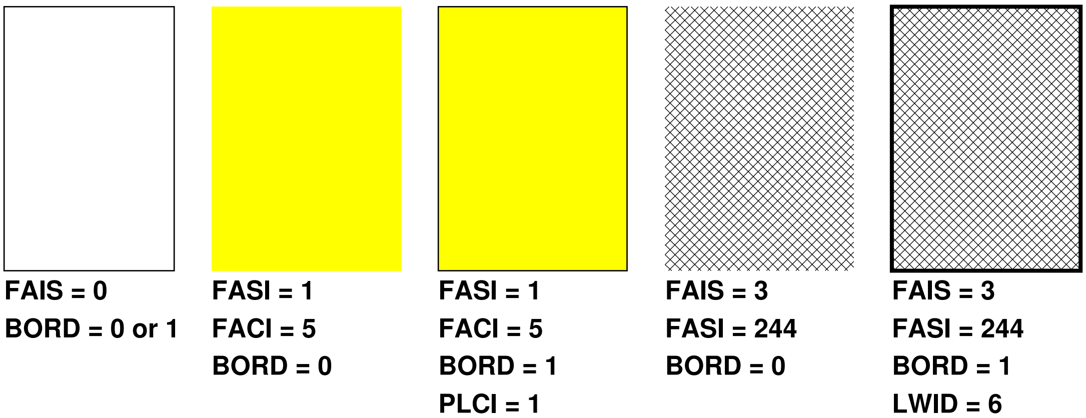


Figure 4.1: Action of the fill area and polyline attributes on `IGBOX`.

4.2 Drawing a frame

```
CALL IGFBOX  (X1,X2,Y1,Y2,X3,X4,Y3,Y4)
```

Action: This routine fills a frame according to the “fill area colour index” (see section 3.6.2), the “fill area interior style” (see section 3.6.3), and the “fill area style index” (see section 3.6.4) attributes. The border is never drawn unless the interior style is hollow or the routine `IGSET` has been called with '`BORD`' and `VAL = 1.`. Like for the `IGBOX` primitive (see figure 4.1), the border of the frame is drawn according to the values of the “line width scale factor” (see section 3.6.6) and the “polyline colour index” (see section 3.6.2) attributes, whereas the “line type” is always solid (see section 3.6.5).

Parameter description:

`X1` X coordinate of 1st corner of the outer rectangle in WC.

`X2` X coordinate of 2nd corner of the outer rectangle in WC.

`Y1` Y coordinate of 1st corner of the outer rectangle in WC.

`Y2` Y coordinate of 2nd corner of the outer rectangle in WC.

`X3` X coordinate of 1st corner of the inner rectangle in WC.

`X4` X coordinate of 2nd corner of the inner rectangle in WC.

`Y3` Y coordinate of 1st corner of the inner rectangle in WC.

`Y4` Y coordinate of 2nd corner of the inner rectangle in WC.

The figure 4.2 describes the usage of the `IGFBOX` parameters.

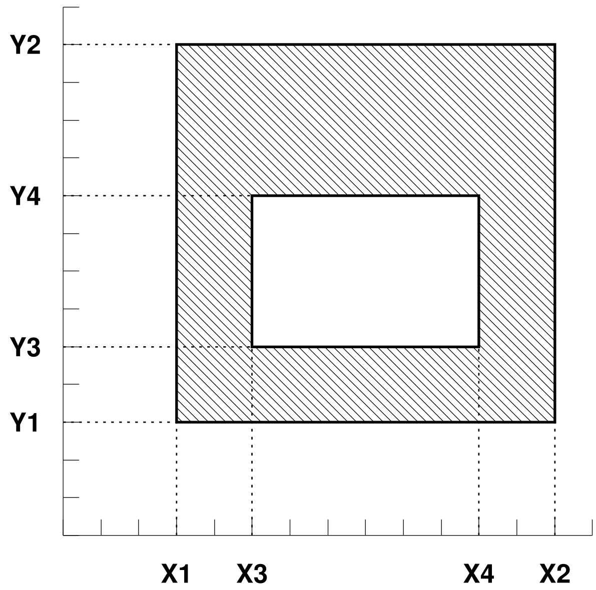


Figure 4.2: Example of `IGFBOX` usage

4.3 Drawing a paving block

```
CALL IGPAVE (X1,X2,Y1,Y2,DZ,ISBOX,ISFRAM,CHOPT)
```

Action: This routine draws a paving-block according to the value of `CHOPT`. `ISBOX (ISFRAM)` may be `1000+ICOLOR` where `ICOLOR` is the colour index of the box (frame), or `2000+IPAT` where `IPAT` is the pattern index of the box (frame), otherwise the style index. If `ISBOX(ISFRAM)=0`, only the box contour is drawn with the current polyline attributes. By default the Top and the Right frames are drawn. `CHOPT='TR'`.

Parameter description:

`X1`	X bottom left corner of box.
`X2`	X top right corner of box.
`Y1`	Y bottom left corner of box.
`Y2`	Y top right corner of box.
`DZ`	Box width.
`ISBOX`	Box style.
`ISFRAM`	Frame style.
`CHOPT`	Character option.

`'T'`	The top of the frame is drawn.
`'B'`	The bottom of the frame is drawn.
`'R'`	The right part of the frame is drawn.
`'L'`	The left part of the frame is drawn.
`'-'`	Reverse sense for the shadow drawing (see figure 4.3).
`'S'`	The frame is drawn like the “Shadow” of the inside box.
`'P'`	Cut the top of the shadow (see figure 4.3).
`'K'`	The paving-block is drawn like a button (see figure 4.3).
`'D'`	Delete. The paving block is drawn in the background colour.

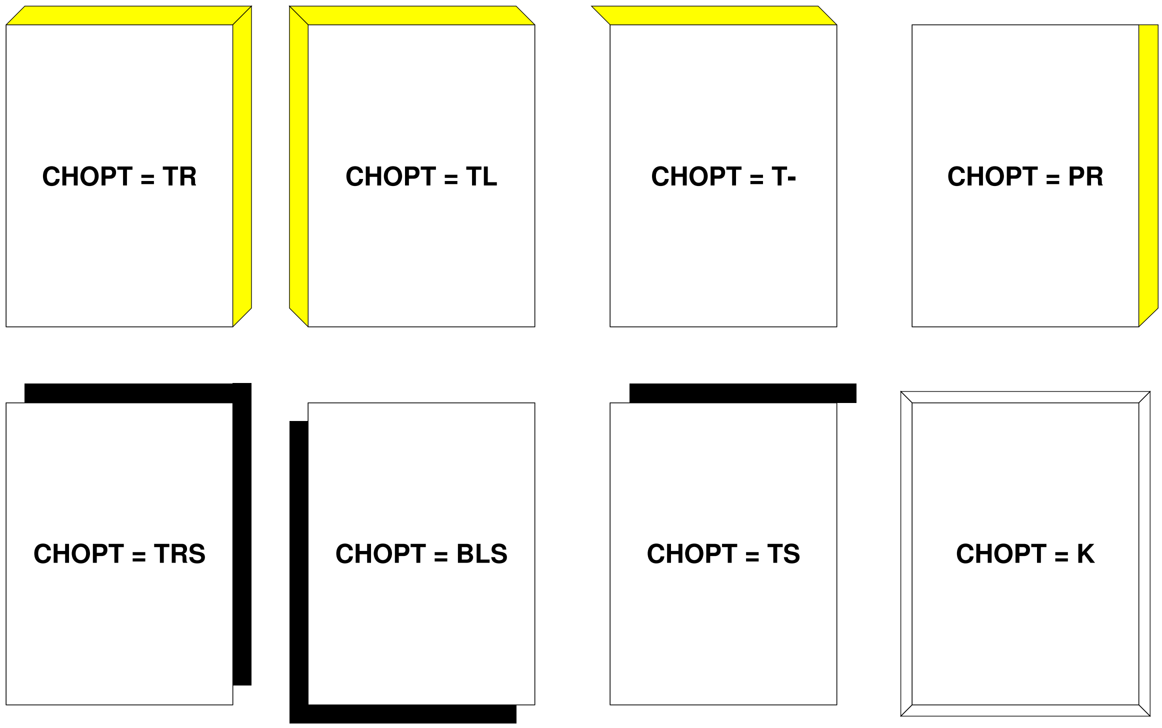


Figure 4.3: Examples of `IGPAVE` usage

4.4 Drawing an arc

```
CALL IGARC (XC,YC,R1,R2,PHIMIN,PHIMAX)
```

Action: This routine draws one or two arcs of a circle. If the two radii are not equal the area between the two arcs is filled according to the fill area interior style index and the fill area style index. The border is never drawn unless the interior style is hollow or the routine `IGSET` has been called with `BORD` and `VAL = 1`. If the arc's radii are equal only one arc is drawn.

Parameter description:

`XC`	X coordinate of the arc's center in world coordinate space.
`YC`	Y coordinate of the arc's center in world coordinate space.
`R1`	Radius of first arc.
`R2`	Radius of second arc.
`PHIMIN`	Starting angle (degrees.)
`PHIMAX`	Final angle (degrees.)

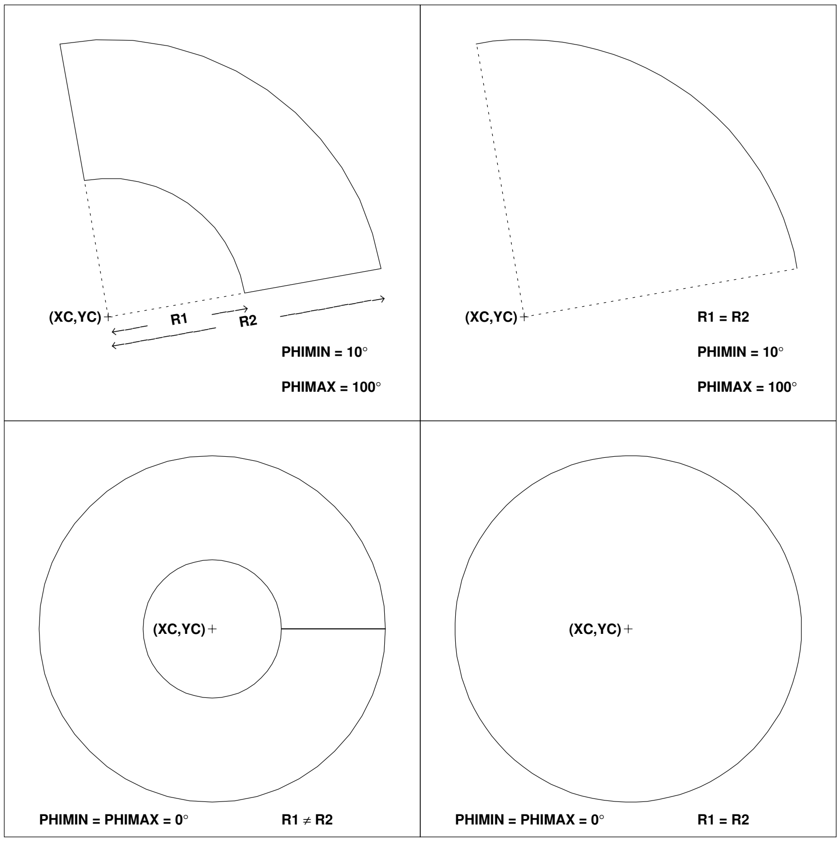


Figure 4.4: Examples of `IGARC`

4.5 Drawing a graph

```
CALL IGRAPH (N,X,Y,CHOPT)
```

Action: This routine draws (in the current normalization transformation) a graph with several possible presentations.

Parameter description:

`N`	Number of components in the arrays `X` and `Y`.
`X`	Array of dimension `N` containing the x coordinates in world coordinates space of the graph to be drawn.
`Y`	Array of dimension `N` containing the y coordinates in world coordinates space of the graph to be drawn.
`CHOPT`	`CHARACTER` variable specifying the options chosen (multiple simultaneous options are possible).

`'R'`	The graph is Rotated, i.e. the values in `X` are used for the ordinate and the values in `Y` for the abscissa (default is the contrary).
`'L'`	All points are connected with a straight line. (default)
`'F'`	A Fill area is drawn through the points with the current fill area attributes. The border is never drawn unless the fill area interior style is hollow or the routine `IGSET` has been called with 'BORD' and `VAL = 1.`.
`'C'`	The values in `Y` are plotted in the form of a smooth curve. Spline approximation algorithms are used. This option can be used with option F in order to draw a smooth fill area.
`'*'`	A star is plotted at every point.
`'P'`	A marker is plotted at every point, according to the current polymarker attributes.
`'B'`	The values in `Y` are plotted in the form of bars. The width of the bar is by default 50% of the interval `X(I)-X(I-1)`. This percentage can be changed by calling `IGSET` with option `BARW`.
`'A'`	X and Y axes are drawn on the border of the current normalization transformation.
`'GX'`	Logarithmic scale on the X axis.
`'GY'`	Logarithmic scale on the Y axis.

Example of GRAPH drawing (see result on figure 4.5)

```
      program  graph
      character*4 chopt(4)
      dimension x(9),y(9)
      parameter (xsize=16.,ysize=20.)
      data x/0.,.6,.3,.2,-.3,.3,-.2,-.3,-.6/
      data y/0.,-.2,-.7,-.9,-.2,.2,.9,.7,.2/
      data chopt/'AL*','AC*','AF*','ACF*'/
*
      call start('graph',xsize,ysize)
*
*              Viewports definition
*
      xnorm  = min(1.,xsize/ysize)
      xnorm2 = xnorm/2.
      ynorm  = min(1.,ysize/xsize)
      ynorm2 = ynorm/2.
      rmarg  = 0.05
      rmarg2 = rmarg/2.
      call isvp(10,rmarg,xnorm2-rmarg2,ynorm2+rmarg2,ynorm-rmarg)
      call isvp(20,xnorm2+rmarg2,xnorm-rmarg,ynorm2+rmarg2,ynorm-rmarg)
      call isvp(30,rmarg,xnorm2-rmarg2,rmarg,ynorm2-rmarg2)
      call isvp(40,xnorm2+rmarg2,xnorm-rmarg,rmarg,ynorm2-rmarg2)
*
*              Some attributes setting
*
      call isclip(0)
      call igset('FASI',244.)
      call igset('BORD',1.)
      call igset('CHHE',.05)
*
*              GRAPH drawing
*
      do i=1,4
         call iswn(10*i,-1.,1.,-1.,1.)
         call iselnt(10*i)
         call igset('FAIS',0.)
         call igbox(-1.,1.,-1.,1.)
         call itx(.3,.9,'CHOPT = '''//CHOPT(I)//'''')
         call igset('FAIS',3.)
         call igraph(9,x,y,chopt(i))
      enddo
      call finish
*
      end
```

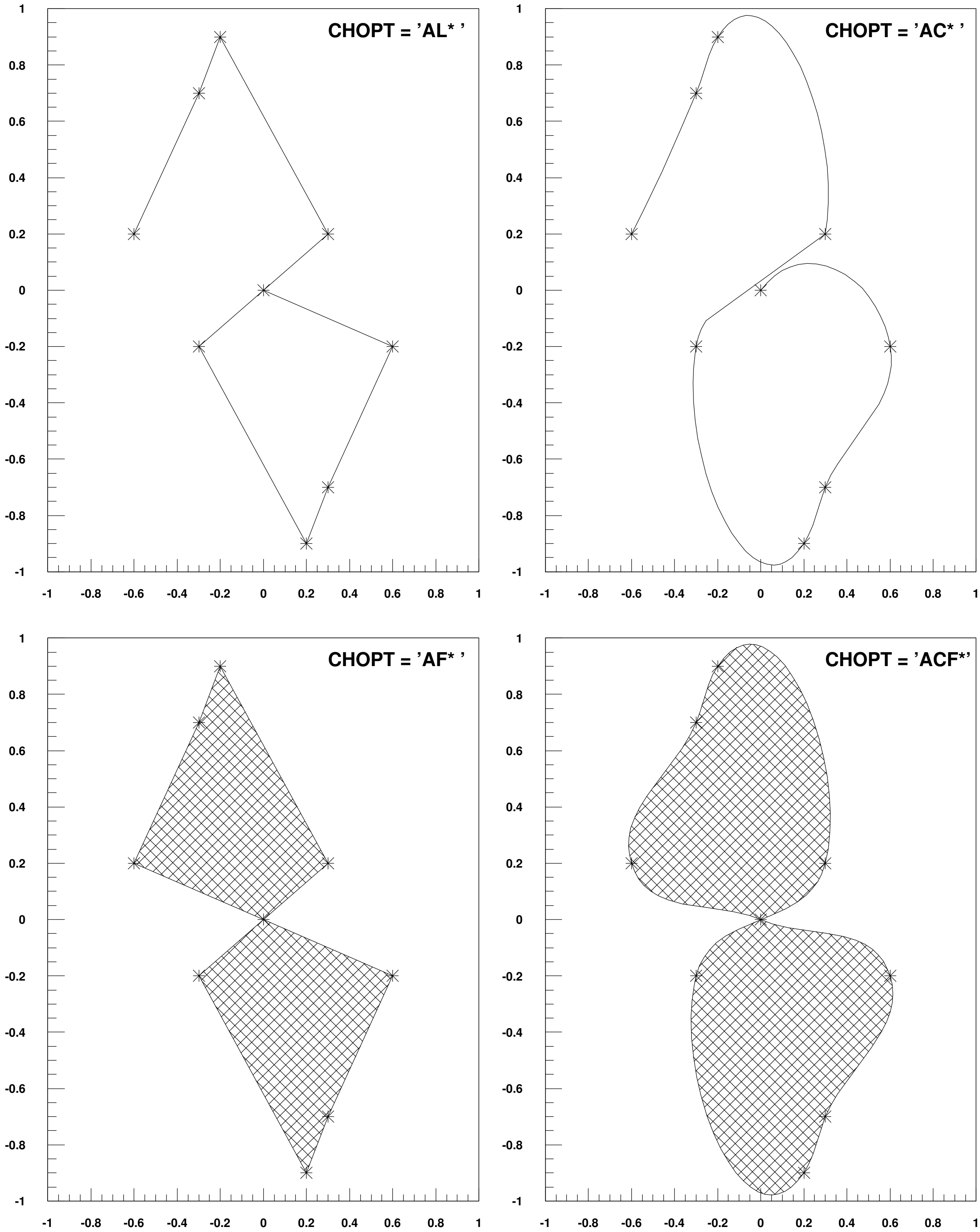


Figure 4.5: Example of `IGRAPH` using `L`, `C`, `F` and $*$ options.

4.6 Drawing a histogram

```
CALL IGHIST (N,X,Y,CHOPT)
```

Action: This routine draws a one-dimensional histogram with several possible presentations chosen by the user (histograms, bars, columns, smoothed graphs, etc.).

Parameter description:

N Number of bins in X and/or Y.

X Is either an array of dimension N containing x coordinates or a two-dimensional array with (XMIN,XMAX) (world coordinates space).

Y Is either an array of dimension N containing y coordinates or a two-dimensional array with (YMIN,YMAX) (world coordinates space).

CHOPT CHARACTER variable specifying the options selected (Multiple simultaneous options are possible). Note that the number of components needed in the array X and/or Y may depend on the value of CHOPT.

'R' The histogram is Rotated, i.e. the values in X are used for the ordinate and the values in Y for the abscissa (default is the contrary). If option 'R' is selected (and option 'N' is not selected), the user must give:

- 2 values for Y (Y(1)=YMIN and Y(2)=YMAX)
- N values for X, one for each bin.

Otherwise the user must give:

- N values for Y, one for each bin.
- 2 values for X (X(1)=XMIN and X(2)=XMAX)

For option 'N' see below.

'N' Non equidistant bins (default is equidistant). The arrays X and Y must be dimensioned as follows:
If option R is not selected (default) then the user must give:

- (N+1) values for X (the limits of the bins).
- N values for Y, one for each bin.

Otherwise the user must give:

- (N+1) values for Y (the limits of the bins).
- N values for X, one for each bin.

'H' An histogram is drawn as a contour (default).

'F' The area delimited by the histogram is filled according to the fill area interior style and the fill area style index or colour index. The contour is not drawn unless the 'H' option is also selected.

'C' A smooth curve is drawn across the points at the center of each bin of the histogram.

'L' A straight line is drawn across the points at the center of each bin of the histogram.

`'*'`	A star is plotted at the center of each bin of the histogram.
`'P'`	The current polymarker is plotted at the center of each bin of the histogram.
`'B'`	A bar chart with equidistant bins is drawn as fill areas (contours are drawn). The bar origin and the bar width can be controlled by routine `IGSET` and the `BARO` and `BARW` options.
`'A'`	The x and y axes are drawn.
`'GX'`	Logarithmic scale on the X axis.
`'GY'`	Logarithmic scale on the Y axis.

Example of HISTOGRAM drawing (see result on figure 4.6)

```
      program hist
      dimension x(2),y(10)
      parameter (xsize=16.,ysize=20.)
      data y/10.,30.,50.,400.,700.,900.,110.,90.,100.,40./
      data x/1.,1000./
*
      call start('hist',xsize,ysize)
*
*              Viewports definition
*
      xnorm  = min(1.,xsize/ysize)
      xnorm2 = xnorm/2.
      ynorm  = min(1.,ysize/xsize)
      ynorm2 = ynorm/2.
      rmarg  = 0.05
      rmarg2 = rmarg/2.
      call isvp(10,rmarg,xnorm2-rmarg2,ynorm2+rmarg2,ynorm-rmarg)
      call isvp(20,xnorm2+rmarg2,xnorm-rmarg,ynorm2+rmarg2,ynorm-rmarg)
      call isvp(30,rmarg,xnorm2-rmarg2,rmarg,ynorm2-rmarg2)
      call isvp(40,xnorm2+rmarg2,xnorm-rmarg,rmarg,ynorm2-rmarg2)
*
*              Some attributes setting
*
      call isclip(0)
      call igset('FASI',244.)
      call igset('FAIS',3.)
      call igset('CHHE',50.)
*
*              HISTOGRAM drawing
*
      call iswn(10,1.,1000.,1.,1000.)
      call iselnt(10)
      call ighist(10,x,y,'AHC*')
*
      call iswn(20,1.,1000.,1.,1000.)
      call iselnt(20)
      call ighist(10,x,y,'AB')
*
      call iswn(30,1.,1000.,log10(1.),log10(1000.))
      call iselnt(30)
      call ighist(10,x,y,'AHFGY')
*
      call iswn(40,log10(1.),log10(1000.),1.,1000.)
      call iselnt(40)
      call ighist(10,x,y,'AHFGX')
*
      call finish
      end
```

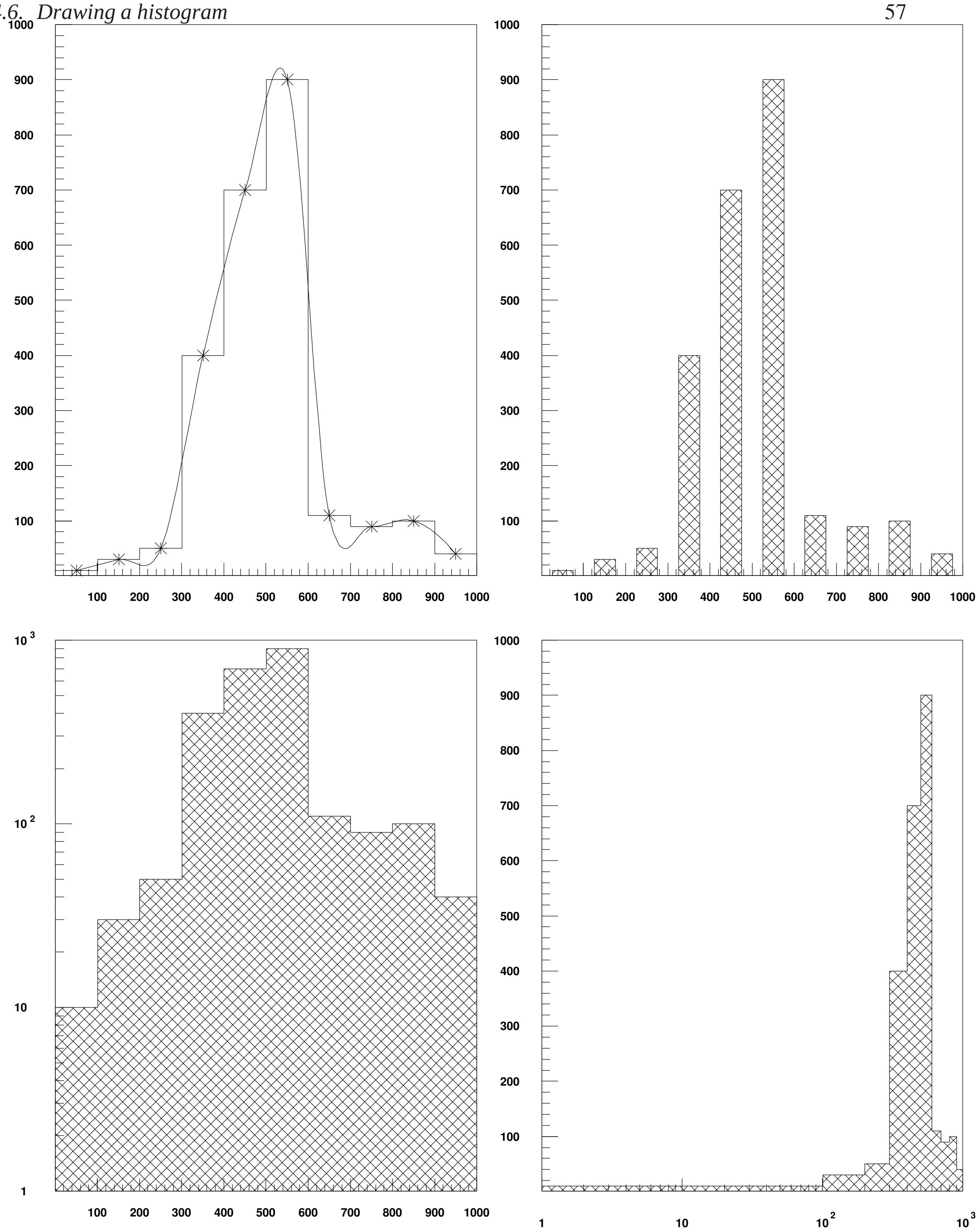


Figure 4.6: Examples of IGHIST usage.

4.7 Bidimensional matrix drawing

```
CALL IGTABL (NX,NY,V,NPAR,PAR,CHOPT)
```

Action: This routine draws a 2D matrix (i.e. table) according to the values of CHOPT and PAR. The PAR input parameter could be specified to change the aspect of the plot (see the description below). The position of the plot on the screen is given by the viewport of the current normalization transformation currently selected (the window is not used and could be anything).

Parameter description:

NX	Number of cells in X.
NY	Number of cells in Y.
V(NX,NY)	Content of the cells.
NPAR	Number of parameters in PAR.
PAR(NPAR)	Array of real parameter. If PAR(i)=0. or NPAR<i a default value is taken.
CHOPT	CHARACTER variable specifying the options selected. The possible value of CHOPT and the associate values of PAR are describe below. The default value of CHOPT is 'P'.

CHOPT = 'P' **Polymarker (scatter plot)**		
PAR **index**	PAR **values**	**default**
1	Marker type see ISMK.	1.
2	Maximum number of random points per cell	50.
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

Example of polymarker option drawing (see result on figure 4.7)

```
Program Scatter
parameter (nx=50,ny=50)
dimension v(nx,ny)
Call Start('scatter',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
```

```
          x=x+s
       enddo
       y=y+s
       x=-10.
    enddo
    Call Igtabl(nx,ny,v,0,0,'PA')
    Call Finish
    end
```

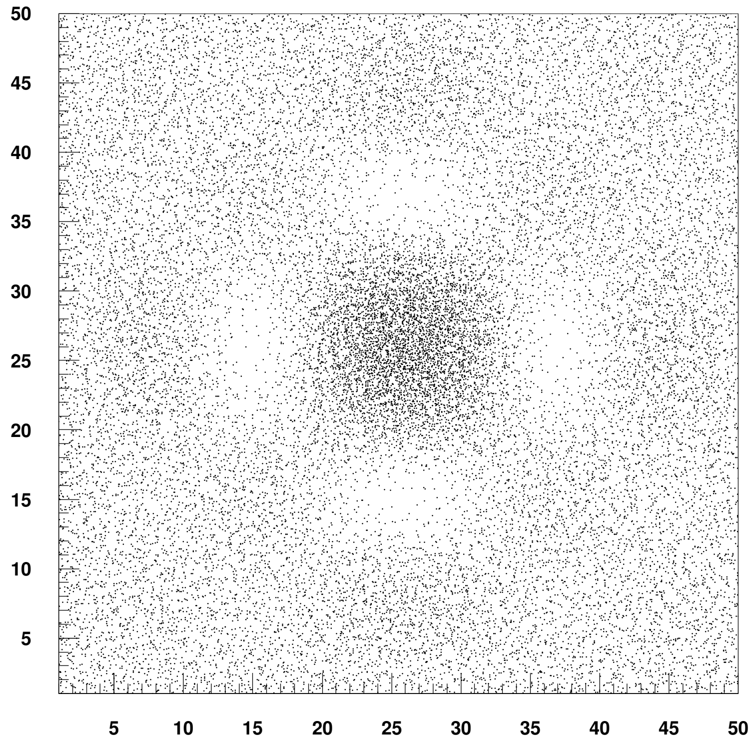


Figure 4.7: Example of the IGTABL Polymarker option

CHOPT = 'B' **Boxes**		
PAR **index**	PAR **values**	**default**
1	Not used	
2	Not used	
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

Example of box option drawing (see result on figure 4.8)

```
Program Boxes
parameter (nx=50,ny=50)
dimension v(nx,ny)
Call Start('boxes',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
Call Igtabl(nx,ny,v,0,0,'BA')
Call Finish
end
```

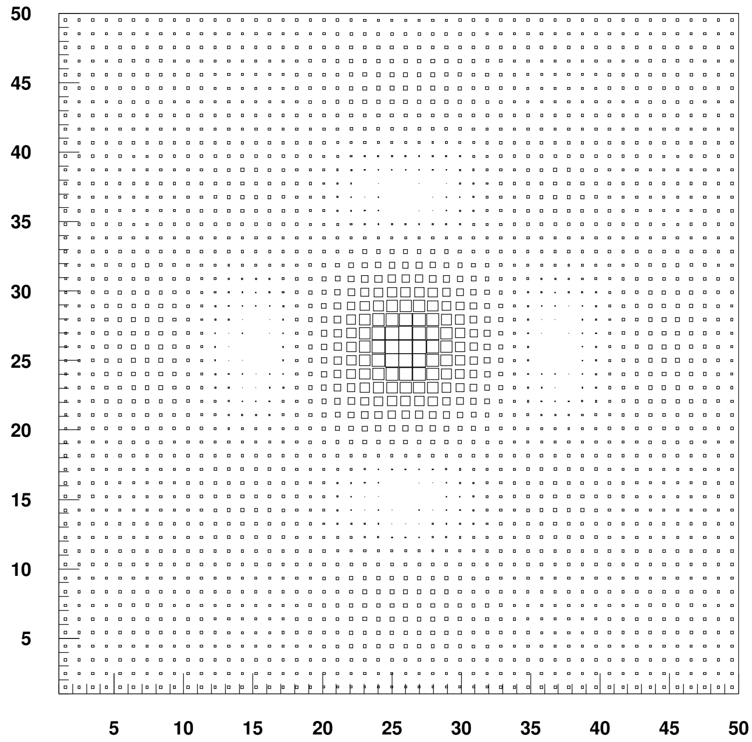


Figure 4.8: Example of the `IGTABL` Boxes option

CHOPT = 'R' **aRrows**		
PAR **index**	PAR **values**	**default**
1	Not used	
2	Not used	
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

Example of arrow option drawing (see result on figure 4.9)

```
Program Arrows
parameter (nx=50,ny=50)
dimension v(nx,ny)
Call Start('arrows',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
Call Igtabl(nx,ny,v,0,0,'RA')
Call Finish
end
```

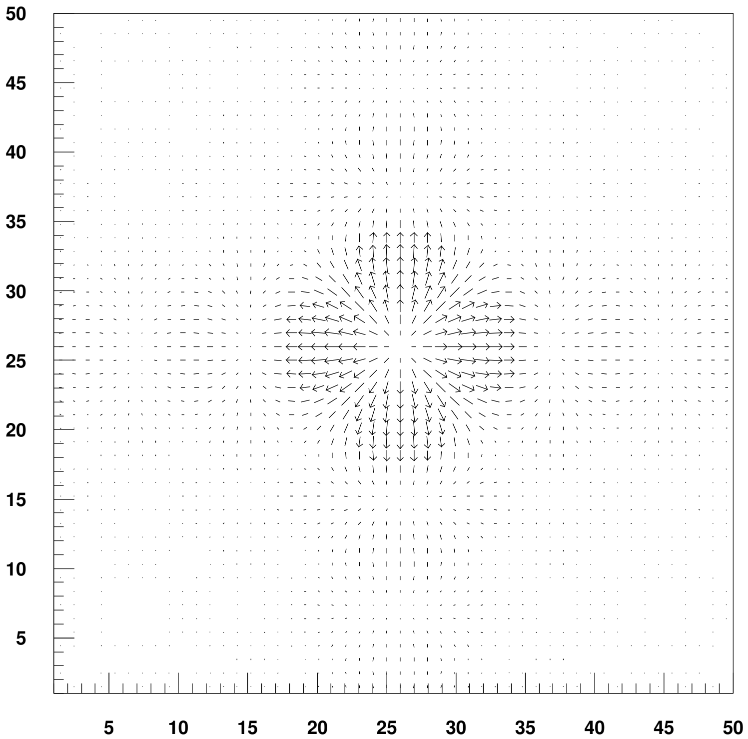


Figure 4.9: Example of the IGTABL aRrows option

CHOPT = 'C' **Contour plot**		
PAR **index**	PAR **values**	**default**
1	Nlevel (min=2 max=50). If NPAR ¡ 11 the level are equidistants.	20.
2	0 use colour to distinguish contours. Line type used is 1.	0.
	1.XXX use line style to distinguish contours. Colour index used is XXX.	
	2.XXX line style and colour are the same for all contours. (LitXXX).	
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY
11 to NPAR	Levels to be drawn	

Example of contour option drawing (see result on figure 4.10)

```
Program Contour
parameter (nx=50,ny=50)
dimension v(nx,ny)
Call Start('contour',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
Call Igtabl(nx,ny,v,0,0,'CA')
Call Finish
end
```

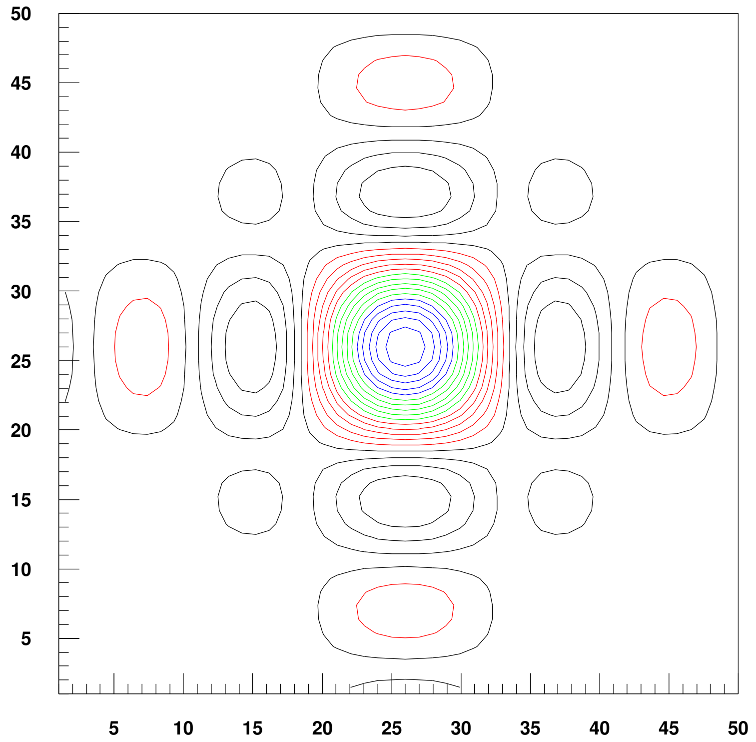


Figure 4.10: Example of the `IGTABL` Contour option

When a table is drawn as a contour plot, it is possible to define in the PAR vector which iso-contours are drawn. The user can specify as many as he want starting form the index 11 in the vector PAR as it is shown in the following example.

Example of contour option drawing with a define set of levels (see result on figure 4.11)

```
Program Contour_Level
parameter (nx=50,ny=50)
dimension v(nx,ny)
dimension par(14)
Call Start('contourl',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
par(11)=-19.
par(12)=0.
par(13)=50.
par(14)=90.
Call Igtabl(nx,ny,v,14,par,'CA')
Call Finish
end
```

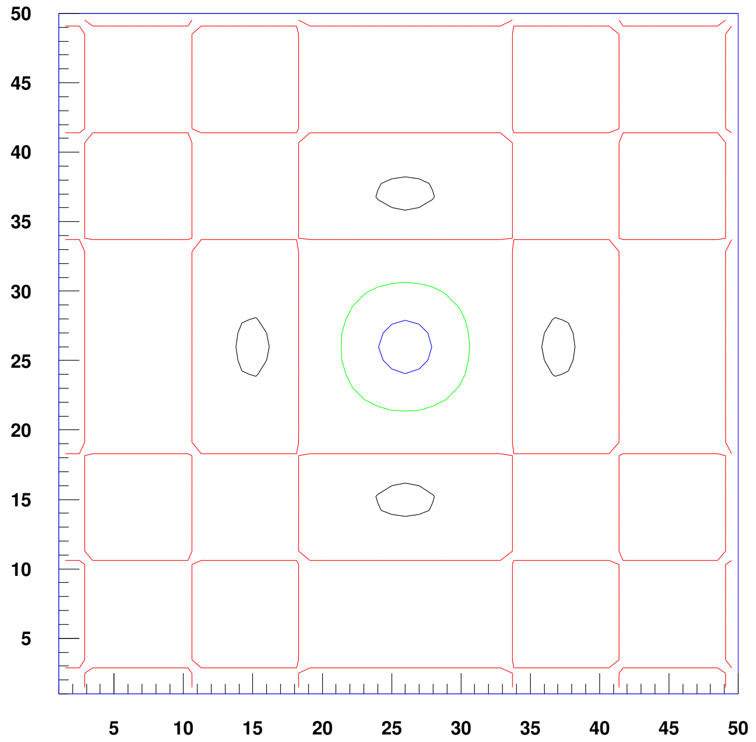


Figure 4.11: Example of the IGTABL Contour option with a define set of levels

CHOPT = 'COL' **COLour plot**		
PAR **index**	PAR **values**	**default**
1	0 use the standard 8 colours	0.
2	...	
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

Example of colour option drawing (see result on figure 4.12)

```
Program Colour
parameter (nx=50,ny=50)
dimension v(nx,ny)
Call Start('colour',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
Call Igtabl(nx,ny,v,0,0,'COL,A,Z')
Call Finish
end
```

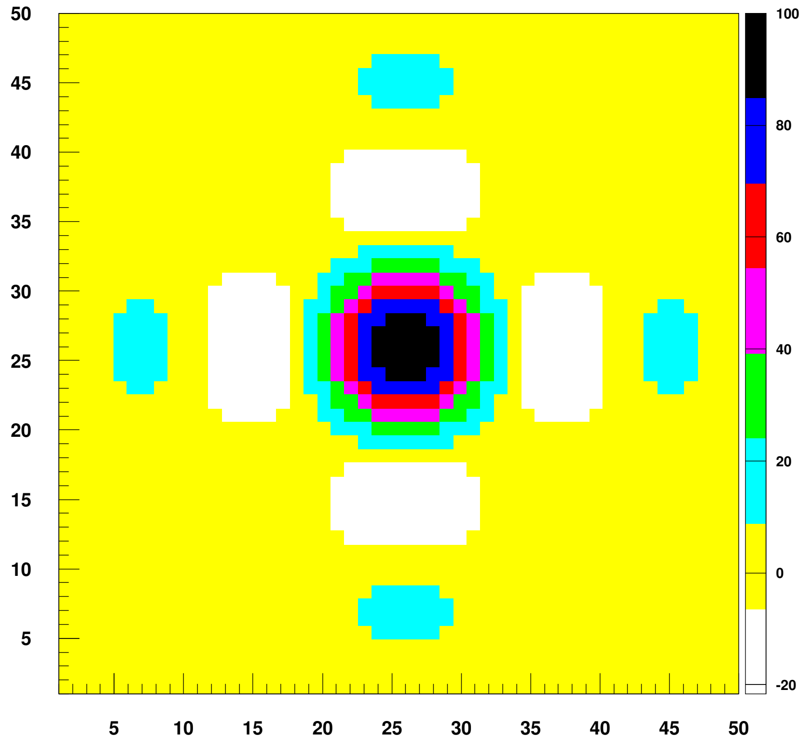


Figure 4.12: Example of the `IGTABL` COLour option

`CHOPT = 'T'` **Text**		
`PAR` **index**	`PAR` **values**	**default**
1	Text font	`1.`
2	Text Precision	`0.`
3	XMIN Lowest X-axis label	`IXMIN`
4	XMAX Highest Y-axis label	`IXMAX`
5	YMIN Lowest Y-axis label	`IYMIN`
6	YMAX Highest Y-axis label	`IYMAX`
7	ZMIN Lowest Z value	`ZMIN`
8	ZMAX Highest Z value	`ZMAX`
9	1000*IXMIN + IXMAX (Useful for ZOOM)	`1-NX`
10	1000*IYMIN + IYMAX (Useful for ZOOM)	`1-NY`

Example of text option drawing (see result on figure 4.13)

```
Program Tabt
parameter (nx=30,ny=30)
dimension v(nx,ny)
Call Start('tabt',9.,9.)
Call Igbox(0.,9.,0.,9.)
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      v(i,j)=100.*sin(x)/x*sin(y)/y
      x=x+s
   enddo
   y=y+s
   x=-10.
enddo
Call Igtabl(nx,ny,v,0,0,'TA')
Call Finish
end
```

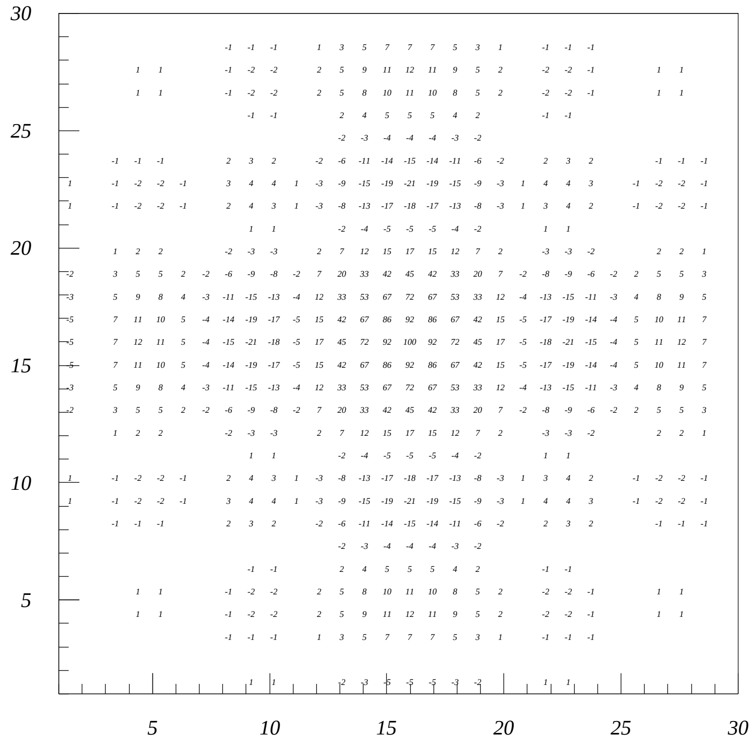


Figure 4.13: Example of the IGTABL Text option

CHOPT = 'K' **character**		
PAR **index**	PAR **values**	**default**
1	Text font	1.
2	Text Precision	0.
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

Example of character option drawing (see result on figure 4.14)

```
      Program Tabk
      parameter (nx=30,ny=30)
      dimension v(nx,ny)
      Call Start('tabk',9.,9.)
      Call Igbox(0.,9.,0.,9.)
      x=-10.
      y=-10.
      s=20./float(nx)
      do i=1,nx
         do j=1,ny
            v(i,j)=100.*sin(x)/x*sin(y)/y
            x=x+s
         enddo
         y=y+s
         x=-10.
      enddo
      Call Igtabl(nx,ny,v,0,0,'KA')
      Call Finish
      end
```

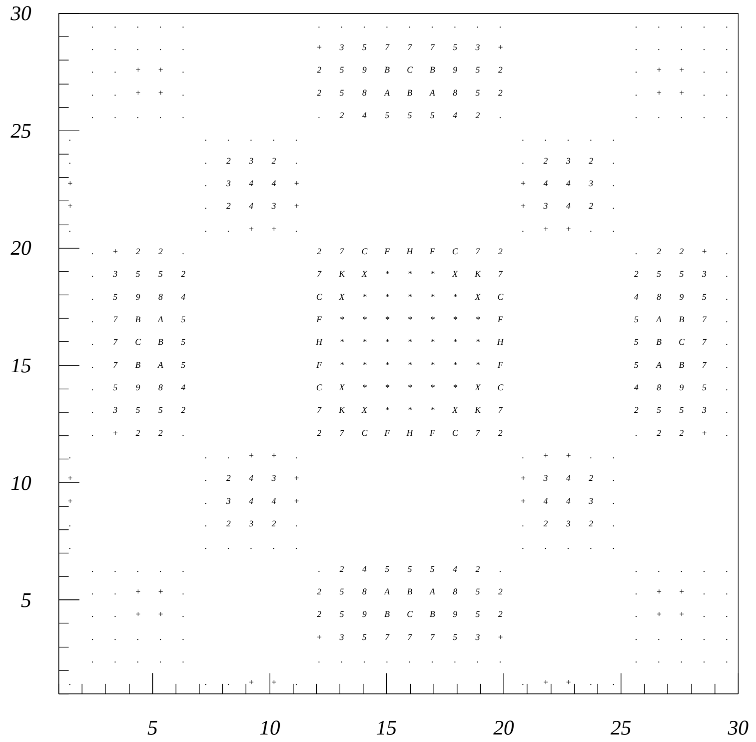


Figure 4.14: Example of the IGTABL character K option

`CHOPT = 'L'` **Lego (mode 0)**		
`CHOPT = 'L1'` **Lego with colours (mode 1)**		
`CHOPT = 'L2'` **Lego with colours (mode 2)**		
`CHOPT = 'B'` **Lego with BARO and BARW**		
`CHOPT = 'S'` **Surface (mode 0)**		
`CHOPT = 'S1'` **Surface with colours (mode 1)**		
`CHOPT = 'S2'` **Surface with colours (mode 2)**		
`CHOPT = 'S3'` **Surface with contour plot on top (mode 3)**		
`CHOPT = 'S4'` **Surface with Gouraud shading (mode 4)**		
`CHOPT = 'CYL'` **Cylindrical for lego and surface**		
`CHOPT = 'SPH'` **Spherical for lego and surface**		
`CHOPT = 'PSD'` **Pseudo rapidity for lego and surface**		
`PAR` **index**	`PAR` **values**	**default**
1	THETA	30.
2	PHI	30.
3	XMIN Lowest X-axis label	IXMIN
4	XMAX Highest Y-axis label	IXMAX
5	YMIN Lowest Y-axis label	IYMIN
6	YMAX Highest Y-axis label	IYMAX
7	ZMIN Lowest Z value	ZMIN
8	ZMAX Highest Z value	ZMAX
9	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
10	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY
11	NDVX	510.00
12	NDVY	510.00
13	NDVZ	510.00
14	XCOL	1.00
15	YCOL	1.00
16	ZCOL	1.00
17	XTIC	0.02
18	YTIC	0.02
19	ZTIC	0.02
20	VSIZ	0.02
21	VFON	2.00
22	XVAL	0.02
23	YVAL	0.02
24	ZVAL	0.04
25	Colour palette for SURF1, LEGO1, SURF2, LEGO2 and SURF3 options	

Table 4.1: Values of the `IGTABL` Lego and Surface option

Examples of Lego and Surface drawing (see results on figures 4.15 to 4.22)

```
Program Lego
Call Start_Lego('lego'    ,1.,'L,A')
Call Start_Lego('lego1'   ,5.,'L1')
Call Start_Lego('lego2'   ,5.,'L2')
Call Start_Lego('surf'    ,1.,'S')
Call Start_Lego('surf1'   ,5.,'S1')
Call Start_Lego('surf2'   ,5.,'S2')
Call Start_Lego('surf3'   ,5.,'S3')
Call Start_Lego('surf4'   ,2.,'S4')
Call Start_Lego('surfpol' ,1.,'S,POL')
Call Start_Lego('surfcyl' ,1.,'S,CYL')
Call Start_Lego('surfsph' ,1.,'S,SPH')
Call Start_Lego('surfpsd' ,1.,'S,PSD')
End

Subroutine Start_Lego(Name,Color,Option)
common /quest/ rquest(100)
character*(*) name,Option
parameter (nx=30,ny=30)
dimension v(nx,ny)
dimension par(29)
call vzero(par,29)
par(1)  = 30.
par(2)  = 23.
par(3)  = -10.
par(4)  = 10.
par(5)  = -10
par(6)  = 10.
par(9)  = 1030.
par(10) = 1030.
par(11) = 510.
par(12) = 510.
par(13) = 510.
par(14) = 1.
par(15) = 1.
par(16) = 1.
par(20) = 0.05
par(21) = -61.
par(22) = .1
par(23) = .15
par(24) = .1
par(25) = Color
par(26) = 7.
par(27) = 5.
par(28) = 6.
par(29) = 3.
x=-10.
y=-10.
s=20./float(nx)
do i=1,nx
   do j=1,ny
      if(x.ne.0..and.y.ne.0)then
         v(i,j)=100.*sin(x)/x*sin(y)/y
      else
         v(i,j)=100.
      endif
```

```
         x=x+s
      enddo
      y=y+s
      x=-10.
   enddo
   Call Start(NAME,10.5,10.5)
   r  = rquest(11)
   xl = rquest(12)
   yb = rquest(13)
   Call Iswn(10,0.,9.,0.,9.)
   Call Isvp(10,1.5*R+XL,10.5*R+XL,1.5*R+YB,10.5*r+YB)
   Call Iselnt(10)
   Call Igtabl(nx,ny,v,29,par,Option)
   Call Igterm
   Call Finish
   end
```

Note that the options POL, CYL, SPH, and PSD can be used together with any lego or surface options.

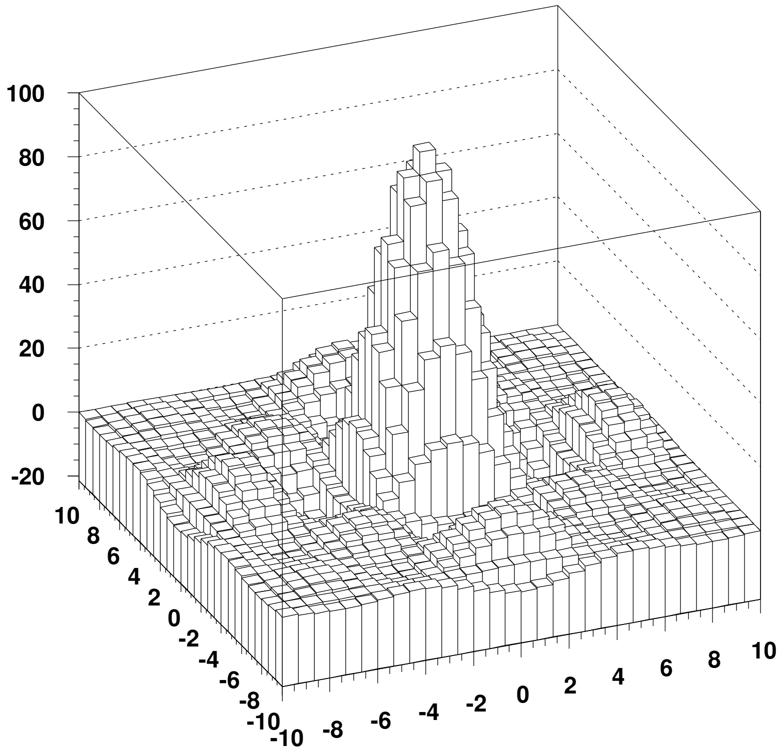


Figure 4.15: Example of the IGTABL Lego option

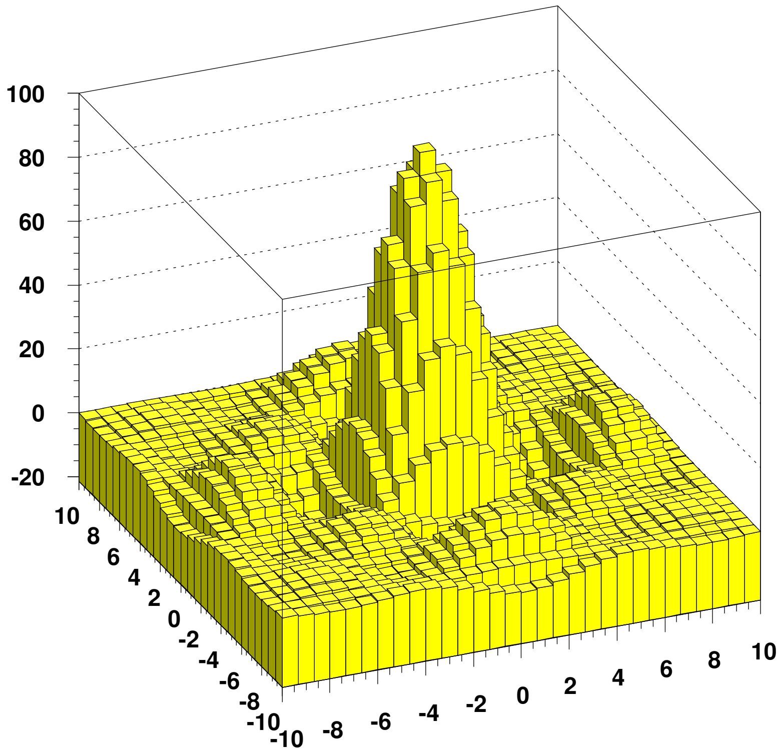


Figure 4.16: Example of the IGTABL Lego L1 option

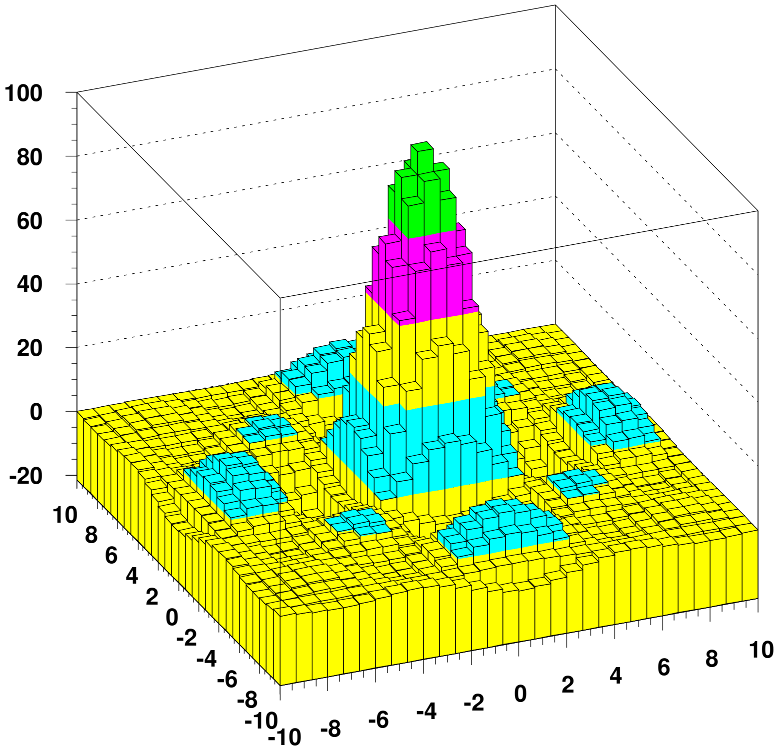


Figure 4.17: Example of the `IGTABL` Lego `L2` option

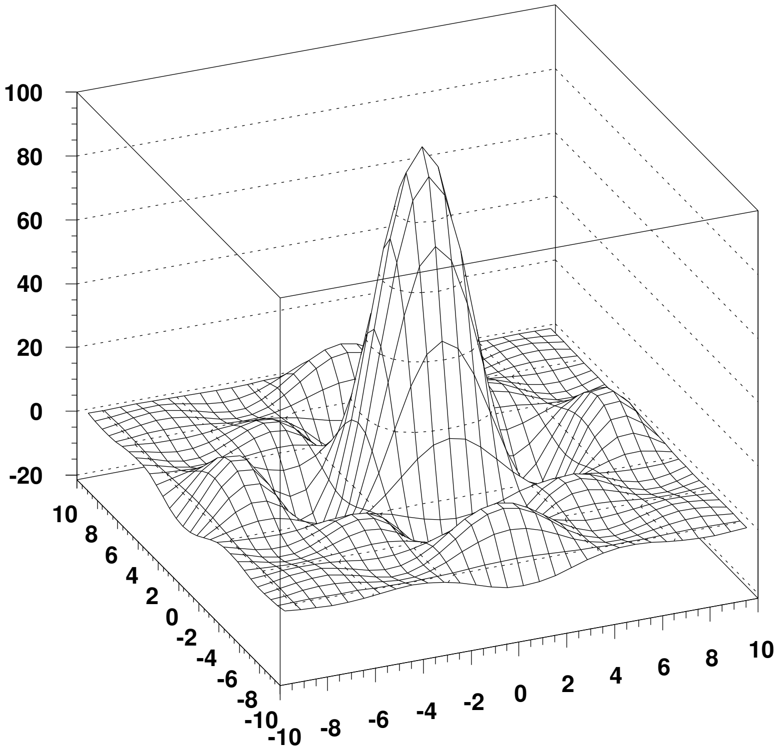


Figure 4.18: Example of the `IGTABL` Surface option

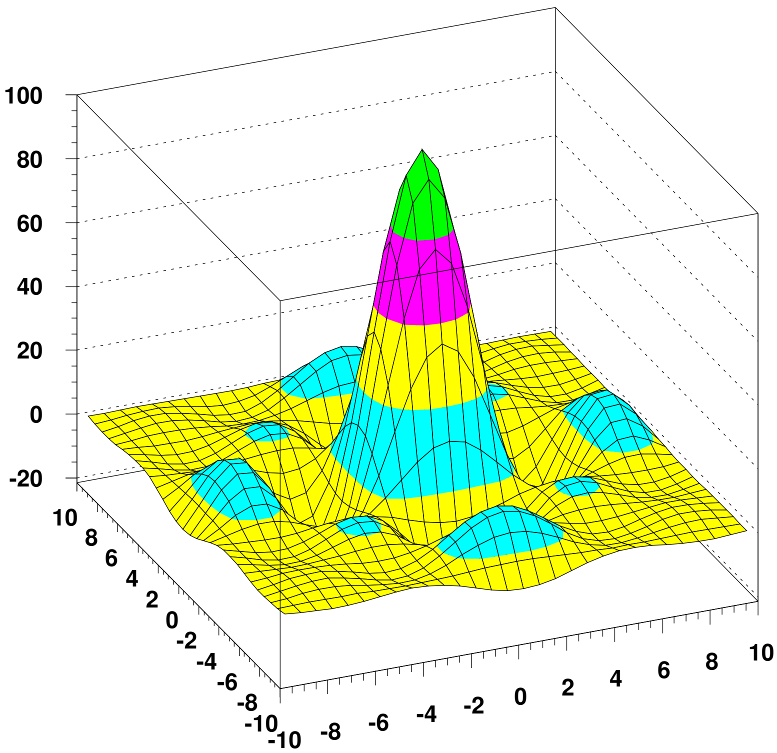


Figure 4.19: Example of the `IGTABL` Surface `S1` option

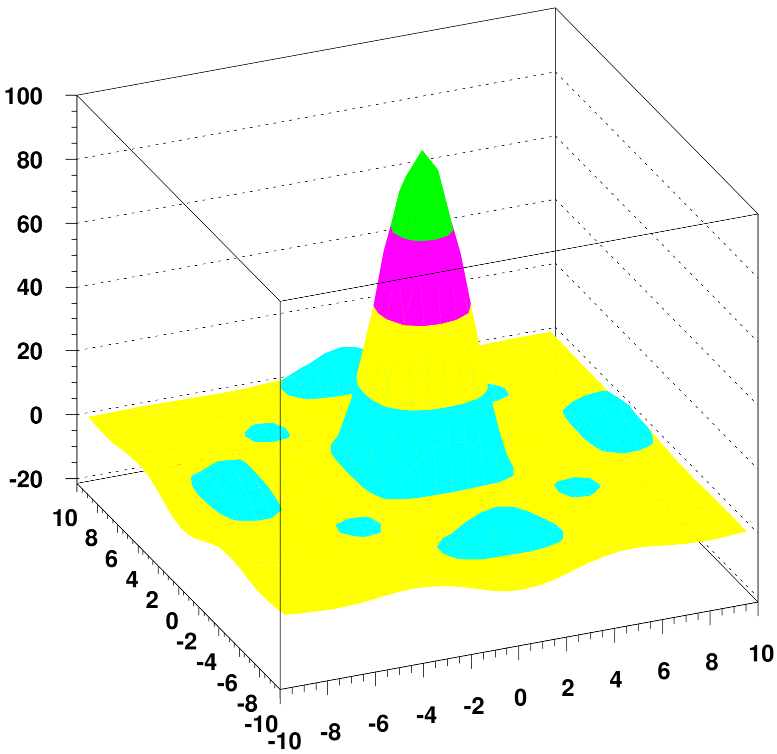


Figure 4.20: Example of the `IGTABL` Surface `S2` option

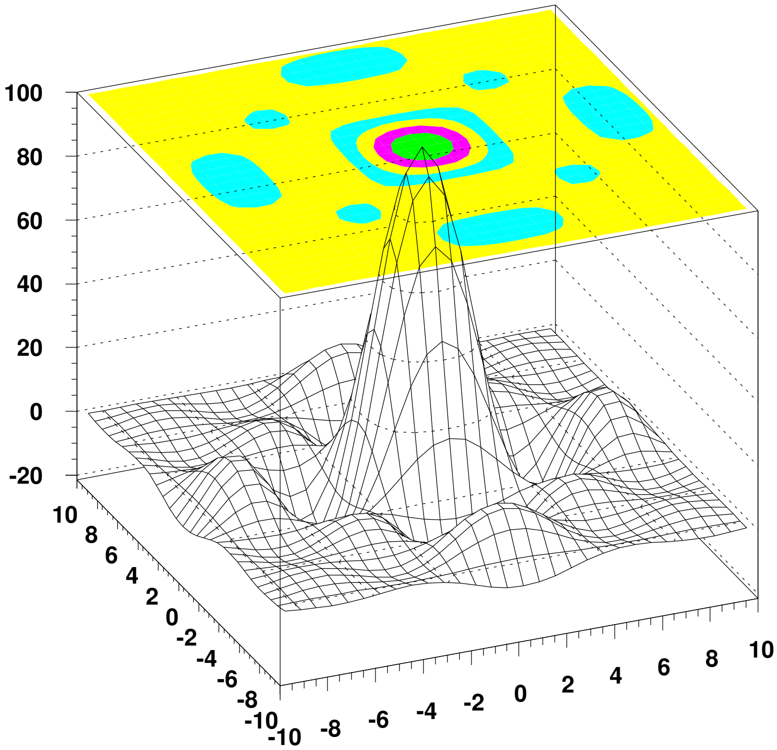


Figure 4.21: Example of the `IGTABL` Surface `S3` option

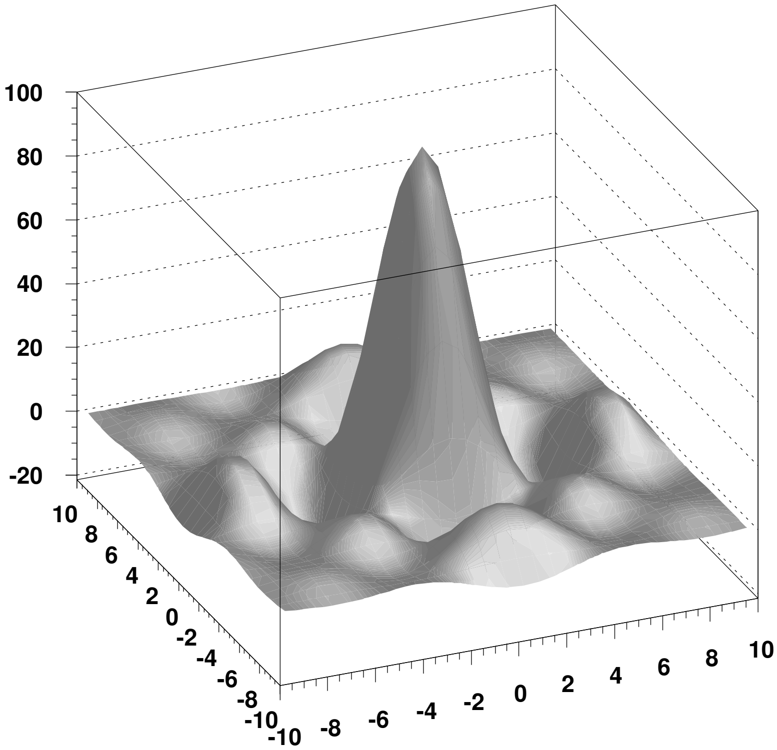


Figure 4.22: Example of the `IGTABL` Surface `S4` option

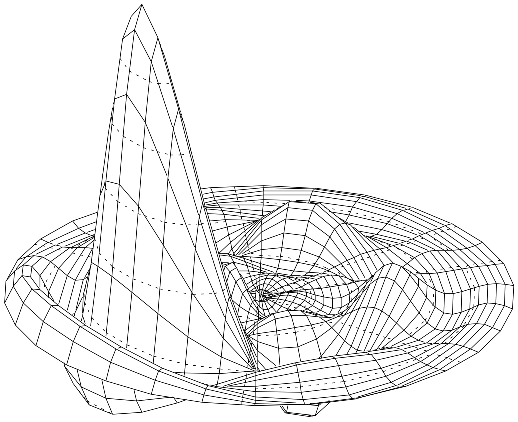


Figure 4.23: Example of the IGTABL Surface SPOL option

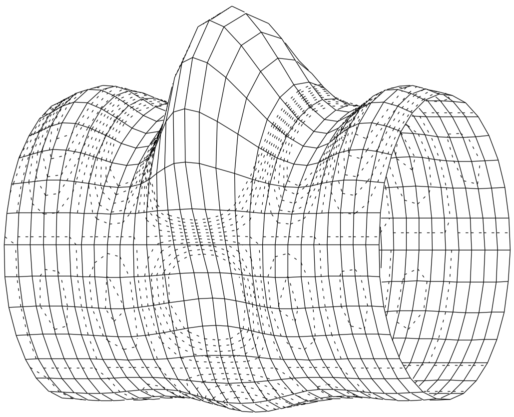


Figure 4.24: Example of the IGTABL Surface SCYL option

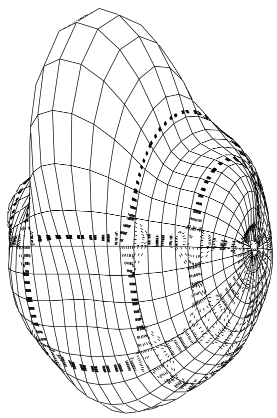


Figure 4.25: Example of the IGTABL Surface SSPH option

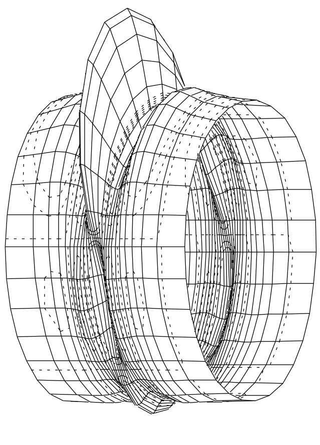


Figure 4.26: Example of the IGTABL Surface SPSD option

CHOPT	Description
'H'	Data are compacted as in HPLOT.
'GX'	loG on X coordinates. A log world coordinates must be defined before.
'GY'	loG on Y coordinates. A log world coordinates must be defined before.
'GZ'	loG on Z coordinates.
'A'	2nd vertical axis (legos and Surfaces only) axis (for the 2D representations).
'+'	For stacked histograms (legos).
'Z'	Allows to display the Z scale.
'E'	Draw the errors with colors. Used with the option S1, S2, L1 the colors are mapped on the errors not on the content.
'FB'	With LEGO or SURFACE, it suppress the Front-Box
'BB'	With LEGO or SURFACE, it suppress the Back-Box

Table 4.2: Other options for IGTABL

Example of stacked lego plots drawing (see result on figure 4.27)

```
      program stack
      parameter (nx=10,ny=10)
      parameter (npar=25)
      dimension v1(nx,ny),v2(nx,ny),v3(nx,ny)
      dimension par(npar)
      call vzero(par,npar)
      par(1)  = 30.
      par(2)  = 23.
      par(3)  = -10.
      par(4)  = 10.
      par(5)  = -10
      par(6)  = 10.
      par(9)  = 1000. + nx
      par(10) = 1000. + ny
      par(11) = 510.
      par(12) = 510.
      par(13) = 510.
      par(14) = 1.
      par(15) = 1.
      par(16) = 1.
      par(20) = 0.05
      par(21) = -61.
      par(22) = .1
      par(23) = .15
      par(24) = .1
*
*              Matrices filling
*
      do i=1,nx
         do j=1,ny
```

```
            v1(i,j)=float(i)
            v2(i,j)=float(i+j)
            v3(i,j)=float(j)
         enddo
      enddo
*
*              Stack drawing
*
      call start('stack',9.,9.)
      call igset('BARW',0.5)
      par(25) = 2.
      call igtabl(nx,ny,v1,npar,par,'+')
      par(25) = 5.
      call igtabl(nx,ny,v2,npar,par,'+')
      par(25) = 3.
      call igtabl(nx,ny,v3,npar,par,'LB1A')
      call igterm
      call finish
*
      end
```

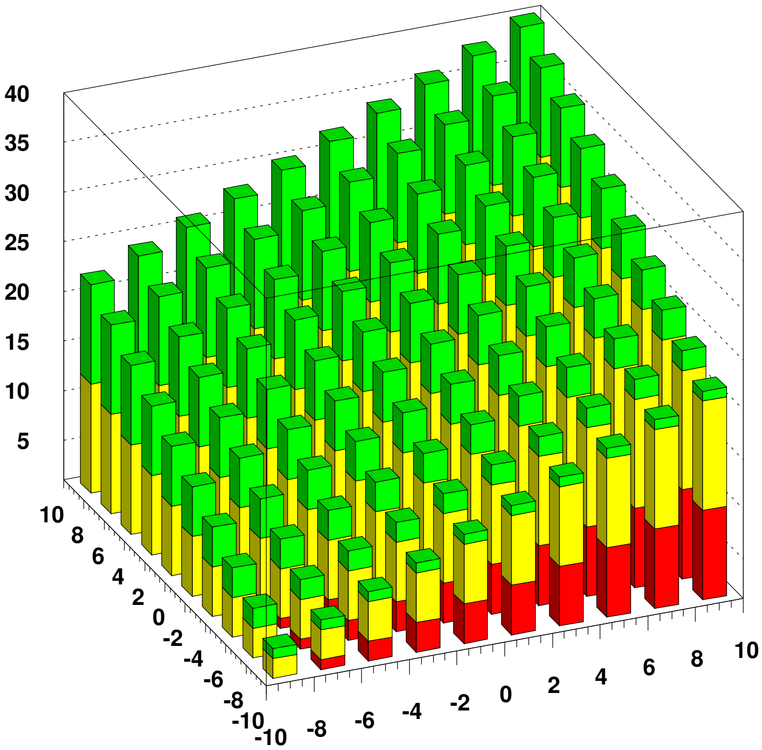


Figure 4.27: Example of stacked lego plots

4.8 Drawing a pie chart

```
CALL IGPIE (X0,Y0,RADIUS,N,VALUES,CHOPT,IAO,IAS,IAC)
```

Action: This routine draws a graph in form of a pie chart.

Parameter description:

`X0` X coordinate of the center of the pie chart.

`Y0` Y coordinate of the center of the pie chart.

`RADIUS` Radius of the pie chart.

`N` Number of entries in the array `VALUES`

`VALUES` Array of dimension `N` containing the values determining the size of the slices in the pie.

`CHOPT` Character variable specifying the combination of options desired:

- `'C'` Colours array is present.
- `'L'` Alphanumeric labels are required (see section 4.9.1).
- `'O'` Offset array is present.
- `'N'` The label of each slice will be the corresponding numeric value in array `VALUES`.
- `'P'` The label of each slice will be in expressed in percentage.
- `'S'` Style array is present.
- `'H'` Force the labels size to be the current character height. Without this option the labels size is computed automatically.
- `'R'` Draw the labels aligned on the radius of each slice.

`IAO` Array of dimension `N` containing offsets of the corresponding slice in percentage of the radius.

`IAS` Array of dimension `N` containing the interior style index for every slice.

`IAC` Array of dimension `N` containing the colour index for every slice.

Example of `PIE CHART` drawing (see result on figure 4.28

```
program  pie
dimension v(8),iao(8),ias(8)
data v /1.,1.8,2.9,1.,1.8,2.9,1.,1.8/
data iao /0,0,0,20,0,0,20,0/
data ias /205,295,245,244,254,245,244,245/
call start('pie',12.,9.)
call isclip(0)
call igbox(0.,12.,0.,9.)
call igset('BORD',1.)
call igpie(3.,6.,2.,8,v,'OSN',iao,ias,0)
call igpie(9.,6.,2.,8,v,'OSP',iao,ias,0)
call igset('TXAL',23.)
call igset('CHHE',0.3)
call itx(3.,3.,'CHOPT = ''OSN''')
call itx(9.,3.,'CHOPT = ''OSP''')
call itx(6.,2.,'IAO = 0,0,0,20,0,0,20,0')
call itx(6.,1.,'IAS = 205,295,245,244,254,245,244,245')
call finish
end
```

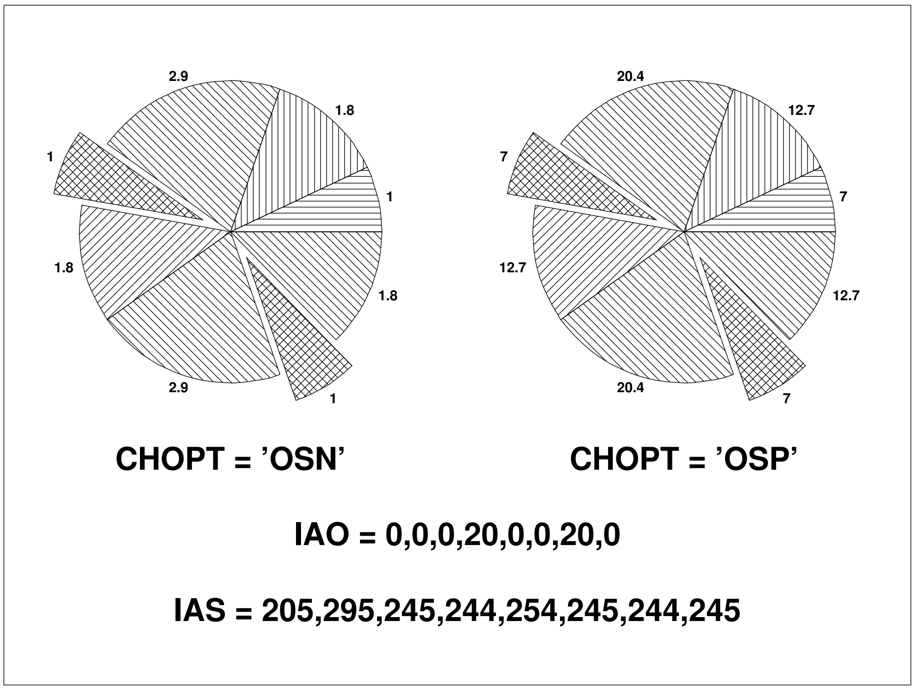


Figure 4.28: Examples of `IGPIE`

4.9 Drawing axes

```
CALL IGAXIS (X0,X1,Y0,Y1,WMIN,WMAX,NDIV,CHOPT)
```

Action: This routines allows the user to draw axes on a picture.

Parameter description:

- `X0` X coordinate of the origin of the axis in world coordinates space.
- `X1` X coordinate of the end of the axis in world coordinates space.
- `Y0` Y coordinate of the origin of the axis in world coordinates space.
- `Y1` Y coordinate of the end of the axis in world coordinates space.
- `WMIN` Lowest value for the tick mark labels written on the axis.
- `WMAX` Highest value for the tick mark labels written on the axis.
- `NDIV` Number of divisions. calculated according to the following convention:

  `NDIV = N1 + 100*N2 + 10000*N3`where,

  - `N1` Number of primary divisions.
  - `N2` Number of second order divisions.
  - `N3` Number of third order divisions.

  Examples:

  - `NDIV=0` No tick marks.
  - `NDIV=2` produces 2 divisions with one tick mark in the middle of the axis.

  Note that, in case numeric labels are requested, `N1` indicates the maximum number of primary divisions. An appropriate algorithm calculates a number of primary divisions less or equal to N1, in order to obtain "reasonable" labels. Option `'N'` in `CHOPT` forces `N1` to be used as the **exact** number of primary divisions.

- `CHOPT` Character variable specifying the combinations of options desired.

  General options

  - `'G'` LoGarithmic scale, default is linear.
  - `'B'` Blank axis, i.e. the base line constituting the axis is not drawn. However tick marks and labels are drawn. Useful when superimposing two axes.
  - `'A'` An arrow is drawn at the end of the axis (position `WMAX`).
  - `'N'` N1 will be used as exact number of divisions.

  Orientation of the tick marks on the axis

  Tick marks are normally drawn on the positive side of the axis. However, if the axis is vertical, i.e. if `X0=X1`, then they are drawn on the "negative" side. Their orientation can be selected by `CHOPT`.

  - `'+'` Tick marks are drawn on the positive side of the axis (default).
  - `'-'` Tick marks are drawn on the negative side of the axis.

Specifying `'+-'` will draw tick marks on **both** sides of the axis.

Orientation of tick marks and labels in the working space

Tick marks are normally drawn orthogonal to the axis. However, in case of an oblique axis, they can be drawn vertically.

`'V'` Tick marks are drawn Vertically (default is perpendicular to axis).

Labeling an axis

An axis is normally labeled, unless specified otherwise:

`'U'` Unlabeled axis (default is labeled).

Position of labels on an axis

Labels are normally drawn on the side opposite to the tick marks, unless specified otherwise:

`'='` Labels are drawn on the same side as the tick marks.

Orientation of labels on an axis.

Labels are normally drawn parallel to the axis.
However if the axis is vertical, i.e. if `X0=X1`, then the labels are drawn orthogonally. If the axis is horizontal, i.e. if `Y0=Y1`, then the labels are Parallel to the axis:

`'P'` Labels are drawn Parallel to the axis

`'O'` Labels are drawn Orthogonal to the axis.

Position of labels with respect to the tick marks.

Labels are centered on tick marks. However, if the axis is vertical (`X0=X1`), then they are right adjusted.

`'R'` Labels are Right adjusted on a tick mark.

`'L'` Labels are Left adjusted on a tick mark.

`'C'` Labels are centered on tick a mark. (default)

Direction of labels

The default writing direction of labels is from **left to right**.

`'Y'` Writing direction is **downwards**.

Format of labels

Training blanks in the label strings are stripped, and then the label is correctly aligned. If the last character of the string is a dot `'.'`, it is also stripped by default.

`'.'` The dot at the end of a string is mandatory.

Type of labels

Labels are by default numeric.

`'T'` The labels are alphanumeric text strings. In this case 12 default values are provided, namely the 3-character abbreviations of the names of the months: `'JAN'`, `'FEB'`, `'MAR'`,.... These values can be modified by calling the routine `IGLBL` (see section 4.9.1).

Optional grid

An optional grid (cross-wires) can be drawn as a prolongation of the primary tick marks.

`'W'` Draw cross-wires at the position of the primary tick marks. The length of the grid can be defined, in world coordinates, with the `IGSET` parameter `AWLN`. The current line type is used to draw the grid.

Intrinsic parameters
The default values for HIGZ intrinsic parameter settings are shown below expressed as a percentage of the length of the axis (world coordinates):

`Primary tick marks:`	3.0 %
`Secondary tick marks:`	1.5 %
`Third order tick marks:`	.75 %
`Length of the arrow:`	3.0 %
`Width of the arrow:`	.75 %
`Characters height for labels:`	2.0 %
`Characters spacing:`	40% of the character height
`Labels offset:`	4.0 %

The size of the secondary tick marks is always 50% of the primary ones. The size of the third order tick marks is always 50% of the secondary ones.
These values can be changed by calls to routine `IGSET`. The default value is used **unless** the corresponding option is selected by `CHOPT`:

- `'D'` The distance between the labels and the axis (the offset) is given by the preceding call to `IGSET` with the parameter `LAOF`.
- `'H'` The size (height) of the labels is given by the preceding call to `IGSET` with the parameter `LASI`.
- `'S'` The size of the tick marks is given by the preceding call to `IGSET` with the parameter `TMSI`.

4.9.1 Control of Alphanumeric labels

`CALL` **IGLBL** `(NLBL,CHLBL)`

Action: This routine must be called to alter the values of the alphanumeric labels used in `IGAXIS`.

Parameter description:

`NLBL` Number of alphanumeric labels specified in array `CHLBL`. The number of labels is limited to 50.

`CHLBL` `CHARACTER` array containing the new values for the alphanumeric labels. The maximal length of each label is 32 characters.

Example of AXIS drawing (see result on figure 4.29)

```
program axis
call start('axis',12.,12.)
call igbox(0.,12.,0.,12.)
call igaxis (1.,11.,1.,1.,0.,100.,510,'A')
call igaxis (1.,11.,3.,3.,1.,10000.,510,'G')
call igaxis (1.,11.,5.,5.,0.,12.,11,'NATY')
call igaxis (1.,11.,6.,6.,-100.,0.,510,'A')
call igaxis (11.,1.,7.,7.,-100.,0.,810,'A+-')
call igaxis (1.,11.,8.,11.,0.,1234567.,615,'A')
call igaxis (6.,11.,8.5,8.5,-3.14,0.,50505,'AN')
call finish
end
```

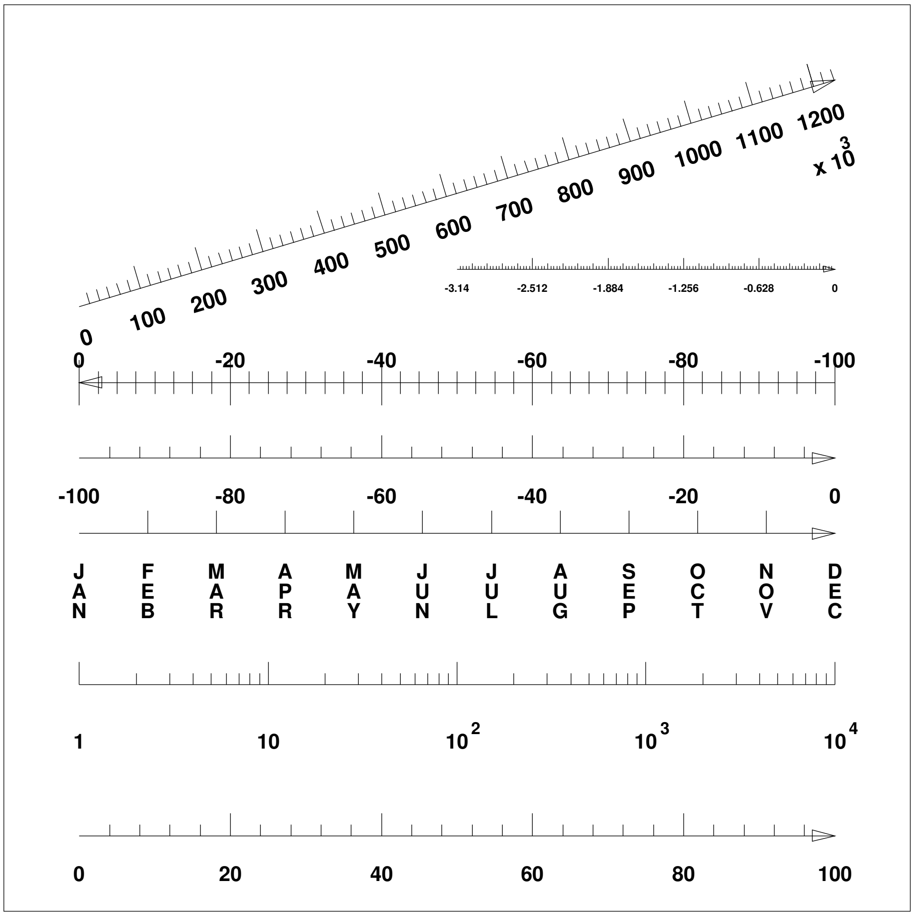


Figure 4.29: Examples of IGAXIS

4.10 Drawing software characters

```
CALL IGTEXT (X,Y,CHARS,SIZE,ANGLE,CHOPT)
```

Action: This routine draws a software character text, independently from underlying graphics package used by HIGZ. `IGTEXT` can produce over 300 different graphic signs. The way in which software characters are defined is via a string of valid Fortran characters, intermixed by other valid Fortran characters, acting as “escape” characters (e.g. a change of alphabet, upper or lower case). The string is interpreted by `IGTEXT` and the resulting characters are defined according to the figure 4.30, which shows the list of available software characters. This routine allows the user to mix different types of characters (roman, greek, special, upper and lower case, sub and superscript). There are a total of 10 control characters.

Parameter description:

`X`	x coordinate in world coordinatesspace.
`Y`	y coordinate in world coordinatesspace.
`CHARS`	`CHARACTER` variable containing the text to be displayed.
`SIZE`	Size of the text in world coordinatesspace.
`ANGLE`	Inclination angle of the text inclination in degrees.
`CHOPT`	`CHARACTER` variable specifying the text alignment:

`'L'`	The text is Left adjusted starting at the point `(X,Y)`.
`'C'`	The text is Centered around the the point `(X,Y)`.
`'R'`	The text is Right adjusted ending at the point `(X,Y)`.
`'S'`	The text size (length) is returned in `ANGLE`.

Note that it is not possible to align vertically the text produce by `IGTEXT`. The way to align vertically software text is to use `ITX` with the font `0` and precision `2` (see `ISTXFP`).

List of escape characters and their meaning			
<	go to lower case	>	go to upper case (default)
[	go to greek (Roman = default)	]	end of greek
”	go to special symbols	#	end of special symbols
↑	go to superscript	?	go to subscript
!	go to normal level of script	&	backspace one character
$	termination character (optional)		

Note that characters can be also entered directly in lower case or upper case instead of using the control characters `<` and `>`.

The boldface characters may be simulated by setting the attributes `'PASS'` and `'CSHI'` with `IGSET`. The meaning of these attributes is the following: Every stroke used to display the character is repeated `PASS` times, at a distance (in percentage of the character height) given by `CSHI`.

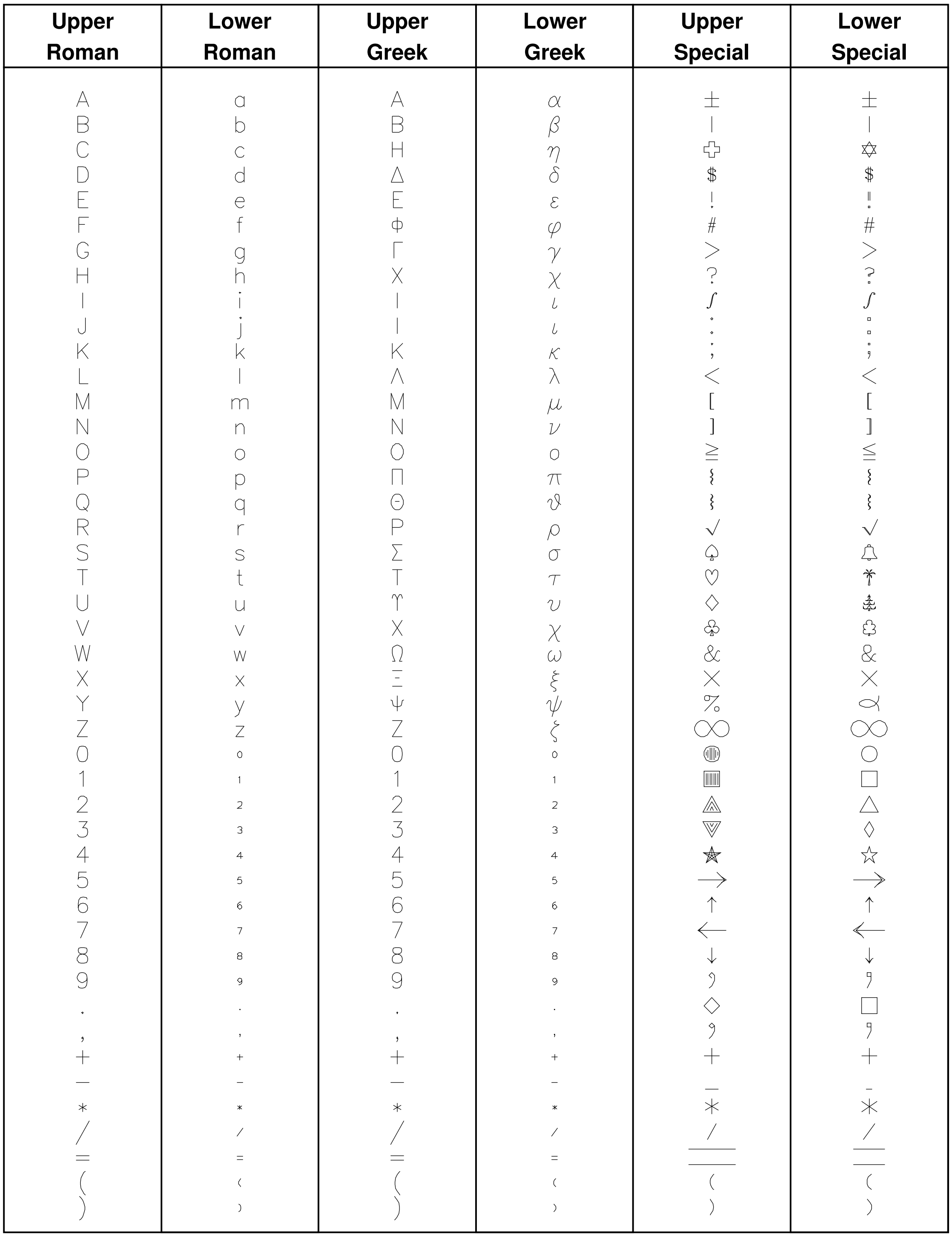


Figure 4.30: Characters available in `IGTEXT`

4.11 Setting attributes

`CALL` **IGSET** `(CHNAME,VAL)`

Action: Routine used to set the value of attributes related to primitives and/or macroprimitives. The first parameter is the mnemonic name of the parameter, the second is the value to be assigned. Note that all the basic primitives attributes can also be set with this routine.

`CHNAME` Character variable specifying the name of the parameter to be set (type `CHARACTER*4`). This is an UPPERCASE character string.

`VAL` **Floating point** value of the parameter (must be specified as a `REAL` number).
A value of `0.0` indicates that the parameter value must be reset to its default value.

CHNAME	VAL
'FAIS'	Fill Area Interior Style (0.,1.,2.,3.). See ISFAIS
'FASI'	Fill Area Style Index. See ISFASI
'LTYP'	Line TYPe. See ISLN
'BASL'	BAsic Segment Length. See ISLN
'LWID'	Line WIDth. See ISLWSC
'MTYP'	Marker TYPe. See ISMK
'MSCF'	Marker SCale Factor. See ISMKSC
'PLCI'	PolyLine Colour Index. See ISPLCI
'PMCI'	PolyMarker Colour Index. See ISPMCI
'FACI'	Fill Area Colour Index. See ISFACI
'TXCI'	TeXt Colour Index. See ISTXCI
'TXAL'	10*(horizontal alignment) + (vertical alignment). See ISTXAL
'CHHE'	CHaracter HEight. See ISCHH
'TANG'	Text ANGle (used to calculate the Character up vector). See ISCHUP
'TXFP'	10*(TeXt Font) + (TeXt Precision). See ISTXFP
'TMSI'	Tick Marks SIze (in world coordinates). See IGAXIS
'LASI'	LAbels SIze (in world coordinates). See IGAXIS
'LAOF'	LAbels OFfset. See IGAXIS
'AWLN'	Axis Wire LeNght. See IGAXIS
'PASS'	Text width (given by number of PASSes) of characters drawn by IGTEXT. The width is simulated by shifting the "pen" slightly at each pass.
'CSHI'	Distance between each shifted drawing of the character (in percentage of the character height) for characters drawn by IGTEXT
'BORD'	0. The border in IGBOX, IGFBOX and IGARC is not drawn. 1. The border in IGBOX, IGFBOX and IGARC is drawn.
'PICT'	Starting number for the automatic naming of pictures.
'AURZ'	1. The last current picture is automatically saved on disk when a new picture is created see IZPICT.
'*'	All attributes are set to their default values.
'SHOW'	The current value and the default of the parameters controlled by IGSET are displayed.
'BARO'	Offset of the left edge of the bar with respect to the left margin of the bin for a bar chart (expressed as a fraction of the bin width). See IGHIST
'BARW'	Width of the bar in a bar chart (expressed as a fraction of the bin width). See IGHIST
'NCOL'	Number of entry in the COLour map.
'CLIP'	Clipping mode: 1.=on 0.=off
CHNAME	**VAL (For X11 interface only)**
'DRMD'	Drawing mode: 1.=copy 2.=xor 3.=invert
'SYNC'	Synchronise the graphics in X11 1.=yes 0.=no
'2BUF'	10*(WKID)+(double buffer mode: 1.=on 0.=off)

Table 4.3: Overview of IGSET parameters

Chapter 5: The input routines

5.1 Cursor input

5.1.1 The Generic Routine

```
GKS   CALL IRQLC (KWKID,LCDNR,ISTAT*,NT*,*PX*,*PY*)
```

Action: This routine returns the (x,y) position of the cursor in world coordinates, and the index the normalization transformation. Its calling sequence is compatible with the equivalent GKS routine.

Parameter description:

KWKID   Workstation identifier.

LCDNR   Locator device.

- 1   Keyboard.
- 2   Graphic tablet.

With the X11 driver LCDNR can have the following values:

- 10   tracking cross
- 20   cross-hair
- 30   rubber circle
- 40   rubber band
- 50   rubber rectangle
- 99   for X11 only: the screen coordinates are taken in PX and PY and used to compute NT, PX, and PY.
- >0   request mode
- <0   sample mode

ISTAT   Return status.

- 0   Graphic input has been canceled.
- 1   A point was located and its coordinates are recorded in PX and PY.

NT   Index of the normalization transformation.

PX   X coordinate of position of locator

PY   Y coordinate of position of locator

5.1.2 The Two Points Routine

```
CALL IGLOC2 (KWKID,*NT*,X1*,Y1*,X2*,Y2*,ISTAT*,CHOPT)
```

Action: This routine returns the graphic cursor position in world coordinates space of two points and the corresponding normalization transformation number. Rubberbanding is used to visualize the area (box) delimited by the two points.

Parameter description:

`KWKID` Workstation identifier

`NT` Index of the normalization transformationsee(`CHOPT`).

`X1` X coordinate of the cursor position in world coordinates space of the first point.

`Y1` Y coordinate of the cursor position in world coordinates space of the first point.

`X2` X coordinate of the cursor position in world coordinates space of the second point.

`Y2` Y coordinate of the cursor position in world coordinates space of the second point.

`ISTAT` Return status:

- `0` Graphic input has been canceled.
- `1` Two points were located and their coordinates are recorded in `X1, Y1, X2, Y2`.

`CHOPT` `CHARACTER` variable specifying the option desired:

- `' '` `NT` is an output parameter.
- `'P'` `NT` is an input and output parameter. In this case, `NT` contains on input the normalization transformation index with the highest priority.

5.1.3 How to get the position both in normalized device coordinates and world coordinates space

```
CALL IGLOC (ICURS,NT*,IBN*,XNDC*,YNDC*,XWC*,YWC*)
```

Action: It is sometimes useful to get a point position both in normalized device coordinates and world coordinates space at the same time. This routine allows to do this for the workstation 1.

`ICURS` Cursor type.

`NT` normalization transformation number.

`IBN` Button number:

- `0` Right button of the mouse.
- `1` Left button of the mouse.
- `3` Middle button of the mouse only for the X11 interface.

`XNDC` X coordinate of the cursor position in normalized device coordinates space.

`YNDC` Y coordinate of the cursor position in normalized device coordinates space.

`XWC` X coordinate of the cursor position in world coordinates space.

`YWC` Y coordinate of the cursor position in world coordinates space.

5.2 Keyboard input

GKS `CALL IRQST (KWKID,ISTDNR,ISTAT*,L*,*STR*)`

Action: This routine returns a character string typed on the keyboard.

Parameter description:

KWKID	Workstation identifier. If KWKID is negative, the parameters RQUEST(81), RQUEST(82), RQUEST(91), and RQUEST(92) given via the QUEST COMMON specify a box in normalized device coordinates in which the request string will be done. If HIGZ is installed with GKS an "initialise string" is performed.
ISTDNR	Device number
ISTAT	Return status. 0: Break and 1: OK
L	Number of characters returned
STR	Character string returned. It should be initialized with the to be edited.

Note that in the routines IRQLC and IRQST the parameter ISTAT may be used to identify the button number of the mouse.

5.3 Menus Input

`CALL IGMENU (MN,CHTIT,*X1*,*X2*,*Y1*,*Y2*,NBU,CHUSER,N,CHITEM, CHDEF,-CHVAL*,ICHOIC*,CHOPT)`

Action: This routine displays a menu and returns the user's choice in the variable ICHOIC according to the option chosen. This routine works only on one menu: the menu management must be performed by the application program but this routine provides some facilities to manage several menus simultaneously.

Parameter description:

MN	Menu number. To use segment capabilities of the workstation. If MN=0 the segments are not used.
CHTIT	Menu title.
X1	X coordinate of lower left hand corner of menu box
Y1	Y coordinate of lower left hand corner of menu box
X2	X coordinate of upper right hand corner of menu box
Y2	Y coordinate of upper right hand corner of menu box
NBU	Number of User squares.
CHUSER	CHARACTER array of length NBU containing the text in the users' squares. The last line of the menu is split into NBU boxes.
N	Number of items.

CHITEM CHARACTER array of length N containing the text for the items.

CHDEF CHARACTER array of length N containing the text for the parameters. If CHOPT='P' the menu is split into two columns. The left column contains the items and the right column the default value of the corresponding item. CHDEF(I) (1<I≤N) is a character string which contains the possible values of the item number I: CHDEF(I)='value1, value2, value3,..., valueN'. If CHDEF(I)=' ' there are no default values.

CHVAL* CHARACTER array of length N into which parameter values are written. If CHOPT='P' then CHVAL(I) contains the parameter value for item I.

ICHOIC Choice number. The description of the possible values returned in ICHOIC is given in the following table:

0	Outside of the menu
-100	Title bar
-1,NBU	User keys
-1000	Right button of the mouse clicked
> 0	Item number

CHOPT CHARACTER variable specifying the option(s) selected.

The square at the left of the title bar moves and resizes the menu. The square at the right of the title bar moves the menu.

'H'	The picked item is highlighted. The last choice number must be given in ICHOIC.
'D'	Display the menu.
'C'	Permit a choice in the displayed menu.
'E'	Erase the menu.
'P'	The menu is a menu with parameters.
'R'	Return the current position of the menu in `X1,X2,Y1,Y2`.
'S'	Software characters are used to draw the text in the menu.
'U'	Update the user text in the user squares with the value in `CHUSER`. The user square number is given in ICHOIC. The options 'U' and 'H' are incompatible because they used both ICHOIC as input parameter.
'M'	Menu drawn on a Metafile.
'Z'	Menu stored in the ZEBRA picture.
'N'	The last input position is used to find the menu item. With this option choices can be made in several menus at the same time using a `DO` loop as shown below. `NBMENU` is the number of menus on the screen.
'B'	A rubberbanding box is used for the locator.
'T'	The title bar is not drawn, then the menu can not be moved interactively.
'W'	The menu is drawn with Width.
'A'	The menu is drawn with shAdow.
'V'	Draw only the vertical part of width or shadow.
'O'	Like option 'V' but the width or shadow is aligned on the menu frame.
'I'	Input menu. A parameter menu is displayed and `IGMENU` is entered directly in request string. This is useful to perform a request string without a very complicated initialization part.
'K'	Key menu. The user keys are drawn as key.

Table 5.1: Options for `IGMENU`

5.3.1 Example

This example program shows how `IGMENU` can manage several menus at the same time.

How to manage several menus

```
      PROGRAM MENU
*
      COMMON /PAWC/H(50000)
      PARAMETER (NBMENU=3)
      CHARACTER*10 CHU, CHI, CHD, CHV, CHTIT, CHOPT
      CHARACTER*80 TEXT
      CHARACTER*16 CHLOC(3)
      DIMENSION CHU(3),NBU(NBMENU),NBI(NBMENU)
```

```
      DIMENSION CHI(3),CHD(3),CHV(3),CHTIT(NBMENU)
      DIMENSION X1(NBMENU),X2(NBMENU),Y1(NBMENU),Y2(NBMENU)
*     Last choice in the menu NB i (useful for HIghligth)
      DIMENSION ICCH(NBMENU)
      DATA CHU /'Quit','Exit','GED'/
      DATA CHI /'Choice 1', '|Choice 2', 'Choice 3'/
*.-------------------------------------
*
*
*       Initialize HIGZ
*
      CALL MZEBRA(-3)
      CALL MZPAW(50000,' ')
      CALL IGINIT(0)
      CALL IGWKTY(KWKTYP)
      CALL IGSSE(6,KWKTYP)
      CALL ISELNT(0)
      CALL MESSAGE('Example of the IGMENU usage in multiple input')
*
*       Initialize and display menu number 1
*
  1   ICCH(1)=0
      X1(1)=0.14
      X2(1)=0.35
      Y1(1)=0.1
      Y2(1)=0.25
      NBU(1)=2
      NBI(1)=3
      CHTIT(1)='MENU 1'
      CALL IGMENU (0,CHTIT(1),X1(1),X2(1),Y1(1),Y2(1),NBU(1),CHU,
     +             NBI(1),CHI,CHD,CHV,ICH,'S   D')
*
*       Initialize and display menu number 2
*
      ICCH(2)=0
      X1(2)=0.3
      X2(2)=0.56
      Y1(2)=0.3
      Y2(2)=0.45
      NBU(2)=2
      NBI(2)=3
      CHTIT(2)='MENU 2'
      CALL IGMENU (0,CHTIT(2),X1(2),X2(2),Y1(2),Y2(2),NBU(2),CHU,
     +             NBI(2),CHI,CHD,CHV,ICH,'S   D')
*
*       Initialize and display menu number 3
*
      ICCH(3)=0
      X1(3)=0.05
      X2(3)=0.95
      NBU(3)=3
      NBI(3)=0
      CHTIT(3)='MENU 3'
      Y1(3)=0.9
      Y2(3)=0.935
      CALL IGMENU (0,CHTIT(1),X1(3),X2(3),Y1(3),Y2(3),NBU(3),CHU,
     +             NBI(3),CHI,CHD,CHV,ICH,'ST  D')
```

```
*
*       Initialize the current menu number
*
      IMENU=3
*
*       Request in the current menu
*
   10 CONTINUE
      IF(IMENU.LT.3)THEN
         CHOPT='S   CR'
      ELSE
         CHOPT='ST  C'
      ENDIF
      ICH=ICCH(IMENU)
      CALL IGMENU (0,CHTIT(IMENU),X1(IMENU),X2(IMENU),
     +             Y1(IMENU),Y2(IMENU),NBU(IMENU),CHU,
     +             NBI(IMENU),CHI,CHD,CHV,ICH,CHOPT)
*
*       If the choice is outside the menu (ICH=0), we search here
*       if the input is in an other menu (CHOPT='N')
*
      IF(ICH.EQ.0)THEN
         DO 20  I=1,NBMENU
            IF(I.LT.3)THEN
               CHOPT='S CRN'
            ELSE
               CHOPT='SCTNKU'
            ENDIF
            ICH=ICCH(I)
            CALL IGMENU (0,CHTIT(I),X1(I),X2(I),Y1(I),Y2(I),
     +                   NBU(I),CHU,
     +                   NBI(I),CHI,CHD,CHV,ICH,CHOPT)
            IF(ICH.NE.0)THEN
               IMENU=I
               GOTO 30
            ENDIF
   20    CONTINUE
*
*       After the DO loop the input is outside all menus
*
         CALL MESSAGE('Outside the menus')
         GOTO 10
      ENDIF
      ICCH(IMENU)=ICH
*
*       Analyses the result
*
   30 CONTINUE
      IF(ICH.GT.0)THEN
         WRITE(TEXT,'(''Menu : '',I1,'', choice : '',I1)')IMENU,ICH
         CALL MESSAGE(TEXT)
         GOTO 10
      ENDIF
      IF(ICH.EQ.-100)THEN
         WRITE(TEXT,'(''Menu : '',I1,'', title bar'')')IMENU
         CALL MESSAGE(TEXT)
         GOTO 10
```

```
      ENDIF
      IF(ICH.EQ.-1000)THEN
         CALL MESSAGE('Right button of the mouse')
         GOTO 10
      ENDIF
      IF(ICH.EQ.-1)THEN
         WRITE(TEXT,'(''QUIT from menu : '',I1)')IMENU
         CALL MESSAGE(TEXT)
         CALL IGEND
         GOTO 999
      ENDIF
      IF(ICH.EQ.-2)THEN
         WRITE(TEXT,'(''EXIT from menu : '',I1)')IMENU
         CALL MESSAGE(TEXT)
         CALL IGEND
         GOTO 999
      ENDIF
      IF(ICH.EQ.-3)THEN
         CALL MESSAGE('Invoke the Graphics Editor')
         CALL IZPICT('*','S')
         CALL IZPICT('P1','M')
         CALL IGRNG(20.,20.)
         CALL IZGED('P1','S')
         GOTO 1
      ENDIF
      IF(ICH.LT.0)THEN
        WRITE(TEXT,'(''Menu : '',I1,'', choice : '',I2)')IMENU,ICH
         CALL MESSAGE(TEXT)
         GOTO 10
      ENDIF
*
  999 END

      SUBROUTINE MESSAGE(TEXT)
      CHARACTER*(*) TEXT
      CALL IGZSET('G')
      CALL ISELNT(0)
      CALL IGSET('FACI',0.)
      CALL IGSET('FAIS',1.)
      CALL IGSET('BORD',1.)
      CALL IGBOX(0.,1.,0.,0.04)
      CALL IGSET('TXAL',23.)
      CALL IGSET('CHHE',0.02)
      CALL IGSET('TXFP',-100.)
      CALL ITX(0.5,0.02,TEXT)
      call iuwk(0,0)
      END
```

Chapter 6: The inquiry functions

6.1 Inquiry the current attributes values

```
CALL IGQ (PNAME,*RVAL*)
```

Action: This routine inquires the value of attribute PNAME and returns in into RVAL.

Parameter description:

PNAME Attribute name

RVAL Returned value. See the description below.

PNAME	RVAL description
'FAIS'	RVAL(1)=Fill Area Interior Style (0,1,2,3)
'FASI'	RVAL(1)=Fill Area Style Index
'LTYP'	RVAL(1)=Line TYPe
'BASL'	RVAL(1)=BAsic Segment Length
'LWID'	RVAL(1)=Line WIDth
'MTYP'	RVAL(1)=Marker TYPe
'MSCF'	RVAL(1)=Marker SCale Factor
'PLCI'	RVAL(1)=PolyLine Colour Index
'PMCI'	RVAL(1)=PolyMarker Colour Index
'FACI'	RVAL(1)=Fill Area Colour Index
'TXCI'	RVAL(1)=TeXt Colour Index
'TXAL'	RVAL(1)=Alignment horizontal RVAL(2)=Alignment vertical
'CHHE'	RVAL(1)=CHaracter HEight
'TANG'	RVAL(1)=Text ANGle
'TXFP'	RVAL(1)=TeXt Font RVAL(2)=TeXt Precision
'TMSI'	RVAL(1)=Tick Marks SIze (in world coordinates)
'LASI'	RVAL(1)=LAbels SIze (in world coordinates)
'LAOF'	RVAL(1)=LAbels OFfset
'PASS'	RVAL(1)=IGTEXT Width
'CSHI'	RVAL(1)=IGTEXT Shift
'BORD'	RVAL(1)=Border for IGBOX, IGFBOX and IGARC (0=No , 1=Yes)
'BARO'	RVAL(1)=IGHIST or IGRAPH BAR charts Offset (%)
'BARW'	RVAL(1)=IGHIST or IGRAPH BAR charts Width (%)
'AWLN'	RVAL(1)=Axis Wire LeNght
'DIME'	RVAL(1)=2D or 3D
'NCOL'	RVAL(1)=Number of entry in the COLour map.
'RGB '	RVAL(1)=Index (Input) RVAL(2)=Red RVAL(3)=Green RVAL(4)=Blue

Table 6.1: Description of the IGQ parameters

6.2 General inquiry function

```
CALL IGQWK (IWKID,PNAME,RVAL*)
```

Action: This routine inquires the values of attribute PNAME and returns it into RVAL.

Parameter description:

IWKID Workstation identifier.

PNAME Attribute name.

RVAL Returned value. See the description below.

PNAME	RVAL description	RVAL dimension
'MXDS'	Maximal display surface (XMAX YMAX)	2
'NTNB'	Current NT number	1
'NTWN'	Current window parameter	4
'NTVP'	Current viewport parameter	4
'DVOL'	Display volume in 3D	3
'ACTI'	1. if IWKID is active, 0. if not	1
'OPEN'	1. if IWKID is open, 0. if not	1
'NBWK'	Number and list of open workstations	11
'2BUF'	1. if the double buffer is on, 0. if not	11
'HWCO'	Number of colours supported by the hardware	11
'WIID'	Window identifier associated to IWKID.	1

Table 6.2: Description of the IGQWK parameters

Chapter 7: Graphical data structures: the IZ routines

7.1 Picture management routines

When options Z or GZ are selected with IGZSET, HIGZ intercepts all calls to the graphics package and stores them into the **current picture** in memory. Each picture is a ZEBRA data structure. Several pictures can coexist at the same time in memory as a ZEBRA linear chain of banks. If a program using pictures receive the message 'Not enough space in memory' some pictures must be deleted or the size of the PAWC common block can also be increased.

With IZPICT and option C one picture becomes the current picture. New primitives can be added and existing structures can be edited with the graphics editor IZGED.

Pictures are identified by a unique name PNAME. Pictures in memory can be saved into ZEBRA/RZ direct access files for later manipulation. Tools exist to transport such files across different computers. HIGZ metafiles are extremely compact compared to GKS metafiles.

One can, for example, generate a HIGZ/RZ metafile at CERN using the HIGZ/GKS-GRAL system, transport these files using BITNET to FNAL and interpret/edit the pictures using the HIGZ/DI3000 system. HIGZ metafiles can be opened/closed several times and new pictures added or modified. Many cycles (versions) of a picture can be stored.

Note that when HPLOT is used, pictures are automatically generated by calling HPLOPT('ZFL',1) and have names PICT1, PICT2, etc. . If a HPLOPT('ZFL1',1) only the last created picture is kept in memory with the name PICT00.

7.1.1 Operation mode control

```
CALL IGZSET (CHOPT)
```

Action: Routine IGZSET sets an internal flag, which determines whether the HIGZ output should be directed to the workstation, to ZEBRA or to both.

Parameter description:

CHOPT Character variable specifying the option

- 'G' Graphics mode only (default).
- 'Z' ZEBRA mode only.
- 'GZ' Both.
- 'S' Save the current state.
- 'R' Restore the last state saved.

Note that when a picture is created with the routine IZPICT the ''Z'' mode is automatically turned on.

7.1.2 Pictures manipulation

```
CALL IZPICT (*PNAME*,CHOPT)
```

Action: This routine allows an HIGZ user to manipulate HIGZ pictures in memory.

Parameter description:

`*PNAME*` `CHARACTER` variable containing the picture's name.

`CHOPT` `CHARACTER` variable specifying the option(s) desired:

- `'M'` Create a new picture in memory with name `PNAME`. An empty structure is created in memory and becomes the current picture. If `PNAME = ' '`, the picture is automatically named `"PICTnnn"` with the starting value for `nnn` either `0` (default), or the value defined by a previous call to `IGSET` with parameter `PICT`.
- `'D'` Display picture `PNAME` in memory.
- `'S'` Scratch picture `PNAME` from memory. If `PNAME=' '` the current picture is deleted.
- `'N'` The Next picture in memory (i.e. the one following the current one) becomes the current picture. If the current picture is the last one in memory, the first picture in memory becomes the current picture.
- `'L'` List the pictures in memory, following the sequence of their storage in memory.
- `'F'` The First picture in memory becomes the current picture.
- `'P'` Print the picture data structure. Useful to debug programs.
- `'C'` Sets the Current picture. All calls to HIGZ graphic functions are stored in the current structure according to the option selected by `IGZSET`.
- `'R'` Retrieve the name of what will be the current picture after the call to `IZPICT`. The name of the current picture is returned in `PNAME`.
- `'G'` Returns in `PNAME` the name of the displayed picture.
- `'Q'` Quiet: no error message printed.
- `'O'` Set the priority order of the normalisation transformations (in the picture `PNAME`) according to the order of creation.

A call to `IZPICT` with one of the options `'M'`, `'N'`, `'F'` or `'C'` automatically sets option `'Z'` of `IGZSET`. In this case the picture following the current one (in the linear chain of pictures in memory) becomes the current picture and is displayed.

7.2 Copying and renaming pictures

```
CALL IZCOPY (PNAME1,PNAME2,CHOPT)
```

Action: This routines allows pictures to be copied or renamed.

Parameter description:

`PNAME1` `CHARACTER` variable with the first picture's name.

`PNAME2` `CHARACTER` variable with the second picture's name.

`CHOPT` Character variable specifying the option desired:

- `'C'` Copy picture `PNAME1` to a new picture called `PNAME2`.
- `'R'` Rename picture `PNAME1` to `PNAME2`.

7.3 Merging pictures

`CALL` **IZMERG** `(PNAME,X0,Y0,SCALE,CHOPT)`

Action: This routine allows a picture to be merged with the current one.

Parameter description:

`PNAME`	`CHARACTER` variable with the picture's name.
`X0`	x coordinate in normalized device coordinates where pictures have to be merged.
`Y0`	y coordinate in normalized device coordinates where pictures have to be merged.
`SCALE`	Scale factor to be applied to picture `PNAME` (`0.`≤`SCALE`≤`1.`).
`CHOPT`	Character variable specifying the option desired
	`'D'` The new current picture is displayed before the merge operation.

7.4 Interface with the graphic editor

```
CALL IZGED (PNAME,CHOPT)
```

Action: This routine invokes the graphics editor. The picture's name is displayed on the screen and a graphic menu is presented. It contains options to add/modify/delete/merge structures within the picture.

Parameter description:

`PNAME` `CHARACTER` variable with the picture's name.

`CHOPT` Character variable specifying the option(s) desired

- `'S'` the menu are drawn with Software characters.
- `'A'` the menu are drawn with shAdow.

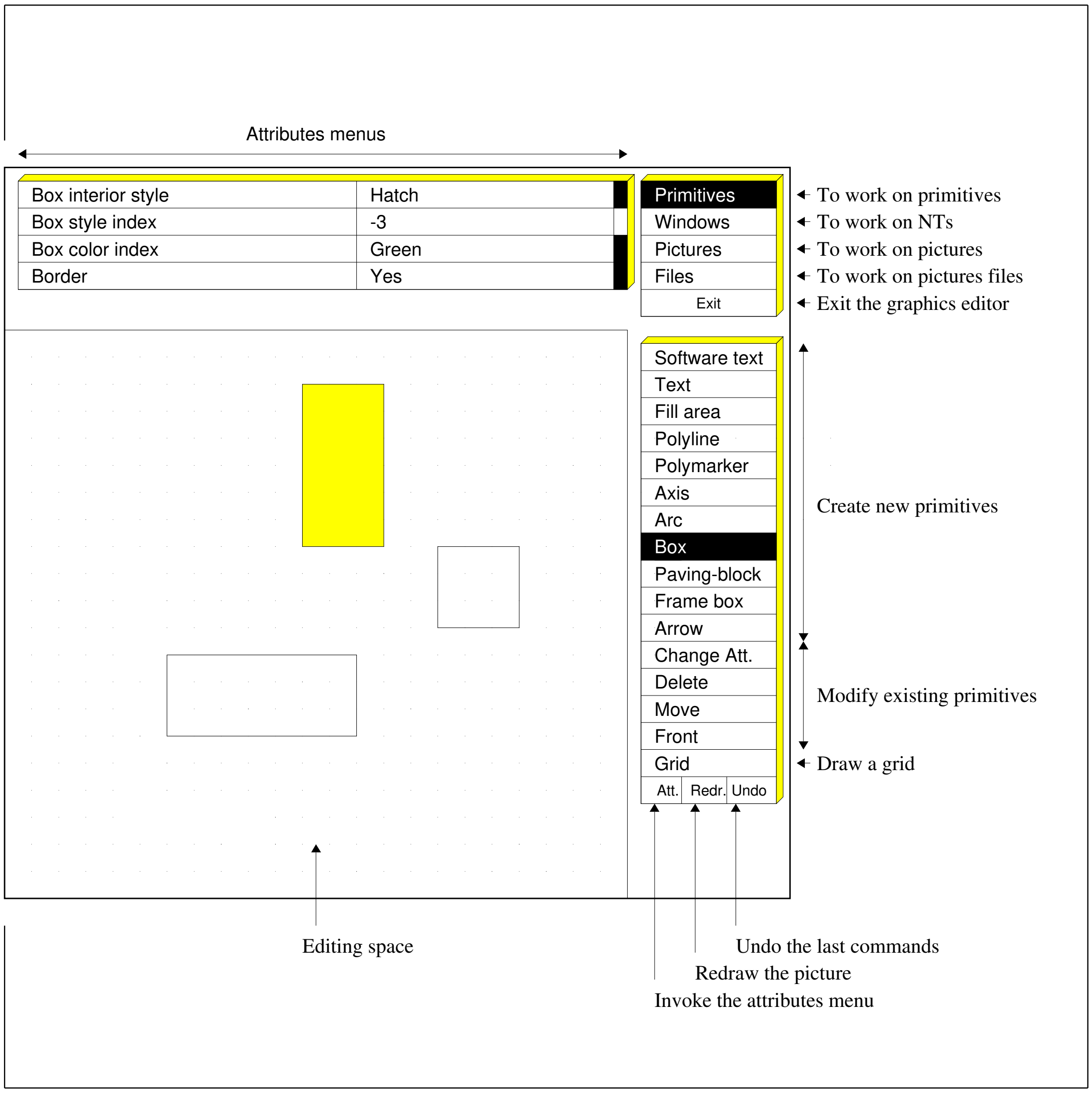


Figure 7.1: The graphics editor

Chapter 8: Structure and picking in the HIGZ pictures

8.1 Tree structure in HIGZ pictures

```
CALL IGPID (LEVEL,NAME,PID,CHOPT)
```

Action: This routine allows to define *primitives identifiers* in the HIGZ pictures. With this routine it is posible to define a tree structure inside HIGZ pictures.

Parameter description:

`LEVEL`	Level number
`NAME`	Primitives names
`PID`	Primitives identifier
`CHOPT`	Character options
	`' '` the level becomes `LEVEL`
	`'U'` one level Up
	`'D'` one level Down

In the HIGZ pictures, all the primitives stored sequentially **after** a *primitive identifier* are stamped with the `LEVEL`, `NAME` and `PID` defined by this *primitive identifier*. The level number allows to define a tree structure in the HIGZ picture.

Some errors are prevented when using `IGPID`. One of these errors is illustrated in the following: if level 5 (for example) is requested when the current level number is 1, then levels 2, 3 and 4 are not defined and routine `IGPID` returns an error message.

8.2 Picking in HIGZ pictures

```
CALL IGPICK (NT*,*X*,*Y*,NBLEV*,NAME*,ID*,CHOPT)
```

Action: This routine allows to return the level number, the name and the identifier of a picked primitive.

Parameter description:

`NT`	Normalization transformation
`X, Y`	Cursor position
`NBLEV`	Number of levels
`NAME(NBLEV)`	Names of the primitives
`ID(NBLEV)`	Identifiers of the primitives
`CHOPT`	Character options (Not used at present)

In addition it is possible to get information via the common `QUEST`.

IQUEST(60)	Adress of the picked primitive in the bank NT. If IADR=0, nothing has been picked.
IQUEST(61)	Primitive type

6	IGHIST
7	IPM with one point
8	IPL with two points
9	IPL
10	IPM
11	IFA
12	ITX
13	IGBOX
14	IGFBOX
15	IGARC
16	IGAXIS
17	IGTEXT
18	IML
20	IGTABL

By default the level 0 is the *Normalisation transformation* level.

8.3 Self structured primitives

It can be very inefficient to call IGPID and IPM with 1 point if many hundreds of points have to be drawn. Routine IPMID solves this problem.

```
CALL IPMID (N,X,Y,LEVEL,ID)
```

Action: This routine behave like IPM excepted that in the HIGZ picture each point is stamped with an identifier.

Parameter description:

N	Number of points
X(N)	X coordinates
Y(N)	Y coordinates
LEVEL	Level number
ID(N)	Points identifier

Example of structured picture (see result on figure 8.1)

```
      program pick
*
      common /pawc/ h(900000)
      character*8 chpid(15)
      dimension ipid(15)
*
      parameter (npts=20)
      dimension xz(86),yz(86)
      dimension x(npts),y(npts),id(npts)
      dimension xf1(3),yf1(3)
      dimension xf2(3),yf2(3)
      dimension xf3(3),yf3(3)
      dimension xf4(3),yf4(3)
      dimension xf5(3),yf5(3)
      data xf1/0.5,0.5,3.0/
      data yf1/0.5,4.0,0.5/
      data xf2/1.0,1.0,2.5/
      data yf2/1.0,3.5,1.0/
      data xf3/1.5,1.5,2.0/
      data yf3/1.5,3.0,1.5/
      data xf4/4.5,4.5,4.0/
      data yf4/1.0,4.0,2.0/
      data xf5/3.0,3.0,1.2/
      data yf5/4.0,4.5,1.1/
      data xz/
     +   0.6250,0.6875,0.9063,0.7500,0.7500,0.6875,0.6250,0.6875
     +  ,0.7500,0.8750,0.9688,1.0313,1.1563,1.2500,1.3125,1.5000
     +  ,1.6875,1.9375,2.0000,2.1250,2.1875,2.1875,2.2500,2.2500
     +  ,2.4375,2.4375,2.4688,2.5313,2.5313,2.5000,2.6250,2.6250
     +  ,2.7500,2.7188,2.7188,2.7188,2.9375,3.4375,3.7500,4.0625
     +  ,4.1250,4.0625,4.1250,4.1875,4.3125,4.3125,4.3125,4.3438
     +  ,4.3125,4.4375,4.5000,4.4375,4.4375,4.5625,4.5938,4.7188
     +  ,4.7813,4.7500,4.5313,4.5000,4.6250,4.6875,4.7188,4.7500
     +  ,4.8750,4.9625,4.9063,4.7500,4.6875,4.6563,4.3750,3.6875
     +  ,3.0625,2.8125,2.4375,2.0313,1.6563,1.4688,1.3438,1.3750
     +  ,1.4375,1.2500,1.1250,1.0000,0.8750,0.6250/
      data yz/
     +   4.8750,4.6563,4.3750,4.1250,3.8750,3.6250,3.4375,3.3125
     +  ,3.1875,3.1563,3.2188,3.3438,3.5000,3.5938,3.6875,3.5625
     +  ,3.3125,3.0938,2.8438,2.7000,2.2188,1.8750,1.2813,1.0625
     +  ,1.0625,1.8750,2.5000,2.4688,2.1875,1.9688,1.5000,1.2500
     +  ,1.2500,1.5313,2.0625,2.6250,2.5938,2.6563,2.7500,3.0000
     +  ,2.7188,2.1250,1.6563,1.4375,1.4688,1.6250,2.0313,2.3125
     +  ,2.6250,2.3125,2.0625,1.6250,1.5000,1.5000,1.6250,2.0313
     +  ,2.3125,2.5000,2.7500,2.9375,3.2500,3.6250,3.2500,2.8125
     +  ,2.6250,2.6875,3.0625,3.5625,3.8750,4.0625,4.1875,4.1250
     +  ,4.0313,4.0938,4.0625,4.2500,4.4875,4.5000,4.4688,4.6875
     +  ,4.8750,4.7188,4.5250,4.4688,4.7188,4.8750/
      data nz/86/
*
      call mzebra(-3)
      call mzpaw(900000,' ')
      call iginit(0)
      call igsse(6,1)
*
*          Create an HIGZ picture
```

```
*
      call izpict('PICT','M')
*
*          Define a new normalization transformation for each new object
*
      call iswn(10,0.,5.,0.,5.)
      call isvp(10,0.05,0.4,0.5,0.8)
      call iselnt(10)
      call igpid(1,'Zebra-axis',1,' ')
      call ipl(nz,xz,yz)
      call igaxis(.5,4.5,.5,.5,0.,1.,10,' ')
*
      call iswn(20,0.,7.,0.,7.)
      call isvp(20,0.5,0.8,0.5,0.8)
      call iselnt(20)
      call ismk(3)
      do j=1,7
         call ispmci(j)
         call igpid(1,'Ntuple',j,' ')
         do k=1,10
            do i=1,npts
               x(i)  = rndm(.01234)+float(j-1)
               y(i)  = 7.*rndm(.01234)
               id(i) = i
            enddo
            call ipmid(npts,x,y,2,id)
         enddo
      enddo
*
      call iswn(30,0.,5.,0.,5.)
      call isvp(30,0.05,0.4,0.1,0.4)
      call iselnt(30)
      call isfais(1)
      call igpid(1,'Red',1,' ')
      call isfaci(2)
      call ifa(3,xf1,yf1)
      call igpid(2,'Green',2,' ')
      call isfaci(3)
      call ifa(3,xf2,yf2)
      call igpid(3,'Blue',3,' ')
      call isfaci(4)
      call ifa(3,xf3,yf3)
      call igpid(1,'Yellow',4,' ')
      call isfaci(5)
      call ifa(3,xf4,yf4)
      call igpid(1,'Magenta',5,' ')
      call isfaci(6)
      call ifa(3,xf5,yf5)
*
      call iswn(40,0.,5.,0.,5.)
      call isvp(40,0.5,0.85,0.1,0.4)
      call iselnt(40)
      call isfais(3)
      call isfasi(344)
      call isfaci(1)
      call igpid(1,'Zebra-fill',2,' ')
      call ifa(nz-1,xz,yz)
```

```
      call igpid(2,'Zebra-line',2,' ')
      call ipl(nz,xz,yz)
*
*             Picking loop
*
  10  call irqlc(1,1,ibut,nt,xloc,yloc)
      call igpick(nt,xloc,yloc,nblev,chpid,ipid,' ')
      print*,'==> Normalization Transformation: ',NT
      do i=1,nblev
         print*, '    Level: ',i,' Name: ',chpid(i),' ID: ',ipid(i)
      enddo
      if(ibut.ne.0)goto 10
*
      call igend
      end
```

The program `pick` produce the following output if six "click" are done like on the figure 8.1.

Output produce by the program `pick`

```
 ==> Normalization Transformation:  40
     Level:  1 Name: Zebra-fi ID:  2
 ==> Normalization Transformation:  30
     Level:  1 Name: Red      ID:  1
     Level:  2 Name: Green    ID:  2
 ==> Normalization Transformation:  30
     Level:  1 Name: Red      ID:  1
     Level:  2 Name: Green    ID:  2
     Level:  3 Name: Blue     ID:  3
 ==> Normalization Transformation:  30
     Level:  1 Name: Yellow   ID:  4
 ==> Normalization Transformation:  20
     Level:  1 Name: Ntuple   ID:  4
     Level:  2 Name: POINT    ID:  4
 ==> Normalization Transformation:  20
     Level:  1 Name: Ntuple   ID:  4
     Level:  2 Name: POINT    ID:  4
```

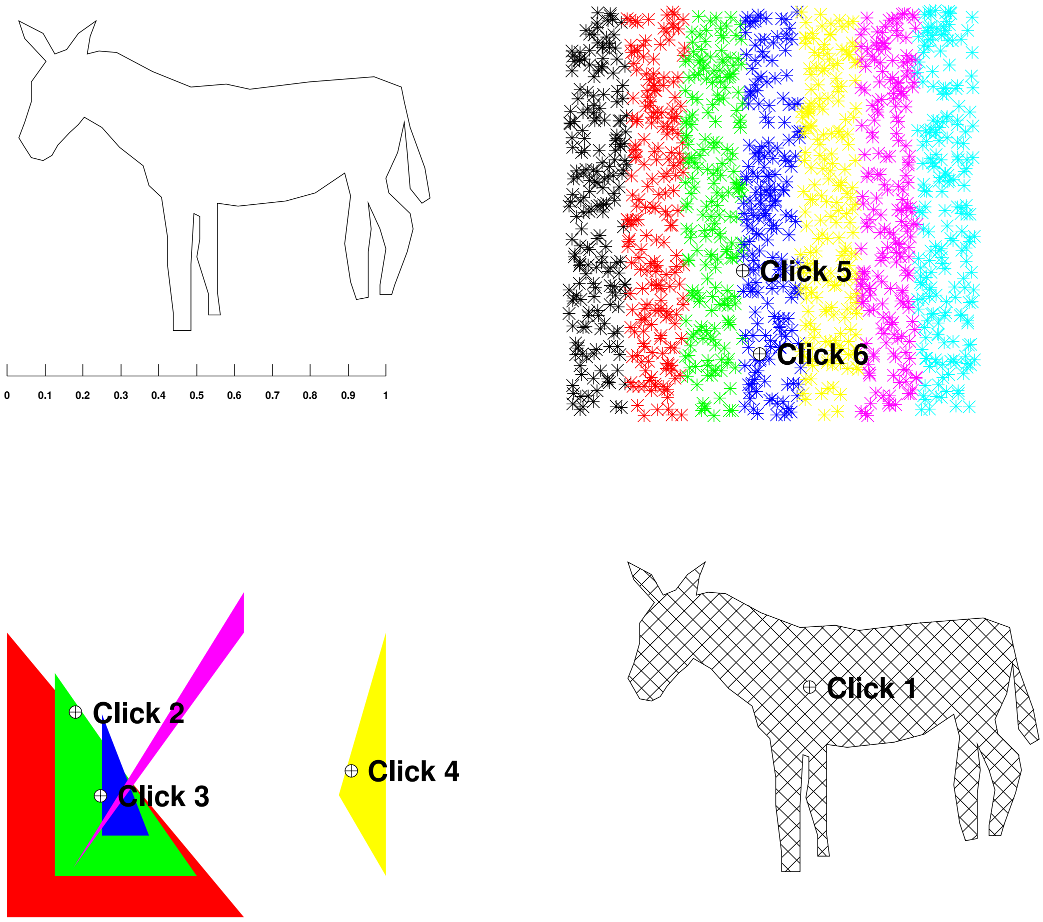


Figure 8.1: A structured picture

Chapter 9: Storing pictures on ZEBRA/RZ direct access files

The routines described in this chapter allow the HIGZ user to store pictures on disk and retrieve them. The pictures created on disk by a given HIGZ program can be used by other HIGZ application programs. Facilities to list the contents of a RZ directory, to purge old cycles, create subdirectories, etc. are available in the ZEBRA/RZ package.

9.1 Interface routines

`CALL` **IZFILE** `(LUN,CHDIR,CHOPT)`

Action: This routine declares a pre-open direct acces file to be ZEBRA/RZ file.

Parameter description:

`LUN`	Logical unit number.
`CHDIR`	`CHARACTER` variable specifying the name of the top directory.
`CHOPT`	`CHARACTER` variable specifying the option(s) desired:

- `'N'` Creates a **N**ew RZ file with top directory name `CHDIR`
- `' '` Open an existing RZ file with read only access.
- `'U'` Open an existing RZ file in **U**pdate mode.
- `'A'` Pictures are **A**utomatically saved on disk.

When option `'A'` is given or when option `AURZ` is activated by `IGSET`, pictures are automatically saved into the RZ file. In this case, there is only one picture in memory (the current picture). The last current picture is written on disk when `IGEND` is called.

`CALL` **IZOPEN** `(LUN,CHDIR,CFNAME,CHOPT,*LRECL*,ISTAT*)`

Action: Open a HIGZ/RZ picture file. This routine open a direct access file and call `IZFILE`. For more details see the description of the ZEBRA routine `RZOPEN` in the ZEBRA manual.

Parameter description:

`LUN`	Logical unit number.
`CHDIR`	`CHARACTER` variable specifying the name of the top directory.
`CFNAME`	File name.
`CHOPT`	`CHARACTER` variable specifying the option(s) desired:

- `'N'` Creates a **N**ew RZ file with top directory name `CHDIR`
- `' '` Open an existing RZ file with read only access.
- `'U'` Open an existing RZ file in **U**pdate mode.
- `'A'` Pictures are **A**utomatically saved on disk.

`LRECL` Integer variable specifying the record length of the file in machine words. If a value of zero (0) is specified `IZOPEN` will attempt to obtain the correct record length from the file itself. A value of zero must not be specified for new files.

`ISTAT` Integer variable in which the status code is returned.

When option `'A'` is given or when option `AURZ` is activated by `IGSET`, pictures are automatically saved into the RZ file. In this case, there is only one picture in memory (the current picture). The last current picture is written on disk when `IGEND` is called.

CALL **IZIN** (PNAME,ICYCLE)

Action: This routine reads a picture from an RZ data file and puts it in memory.

Parameter description:

`PNAME` `CHARACTER` variable specifying the name of picture to be read.

`ICYCLE` Cycle number of the picture to be read. If `ICYCLE` is greater than the highest existing cycle number on the RZ file, then the picture with the highest cycle number is read.

CALL **IZOUT** (PNAME,ICYCLE*)

Action: This routine writes on an RZ data file a memory resident picture.

Parameter description:

`PNAME` `CHARACTER` variable specifying the name of picture to be written.

`ICYCLE*` Cycle number of the picture written. If a picture with name `PNAME` does not already exist on the output file, then a value for `ICYCLE` of `1` is returned, otherwise a value one higher than the (previous) highest cycle number on the file.

CALL **IZSCR** (PNAME,ICYCLE)

Action: This routine deletes (scratches) a picture from an RZ data file.

Parameter description:

`PNAME` `CHARACTER` variable specifying the name of picture to be deleted.

`ICYCLE` Cycle number of the picture to be deleted.

Chapter 10: Miscellaneous functions

User routines, whose functionality is often needed (e.g. displaying a message), but which cannot be classified easily in any of the previous chapters will be described in this chapter.

10.1 Display a message on the screen

```
CALL IGMESS (N,CHMESS,CHTIT,CHOPT)
```

Action: This routine allows to display a message. The X11 version of HIGZ displays the message in a separated window.

Parameter description:

`N`	Number of lines in the message.
`CHMESS(N)`	Message to be displayed.
`CHTIT`	Window title.
`CHOPT`	Options.

`'P'`	Print the array `CHMESS` and open the message window if necessary.
`'C'`	Close the message window.
`'T'`	Print the array `CHMESS` on standard output.
`'D'`	Delete the message window.

10.2 Display a colour map

```
CALL IGCOLM (X1,X2,Y1,Y2,IC1,IC2,ZMIN,ZMAX,CHOPT)
```

Action: This routine allows to display a colour map on the screen from the coulour index `IC1` to the colour index `IC2`.

Parameter description:

`X1`	X coordinate of 1st corner of the rectangle in world coordinates.
`X2`	X coordinate of 2nd corner of the rectangle in world coordinates.
`Y1`	Y coordinate of 1st corner of the rectangle in world coordinates.
`Y2`	Y coordinate of 2nd corner of the rectangle in world coordinates.
`IC1`	First colour index.
`IC2`	Last colour index
`ZMIN`	Minimum Z value.
`ZMAX`	Maximum Z value.
`CHOPT`	Options.

`'C'`	Draw the levels with **C**olours.

`'B'`	Draw the levels with **B**oxes.
`'A'`	Draw the **A**xis.
`'H'`	Draw the map **H**orizontally (default is vertically).
`'G'`	Logarithmic scale is used to draw the axis.
`'P'`	`IC1` is the dimension of the `INTEGER` array `IC2` in which a list of colour indeces is given.
`'L'`	The current palette is used. `IC1` and `IC2` are not used.

10.3 Conversion between Colour systems

10.3.1 RGB to HLS

```
CALL IGRTOH (CR,CB,CG,CH*,CL*,CS*)
```

Action: This routine convert a RGB colour into an HLS colour.

Parameter description:

CR	Red value $0. \leq CR \leq 1.$
CG	Green value $0. \leq CG \leq 1.$
CB	Blue value $0. \leq CB \leq 1.$
CH	Hue value $0. \leq CH \leq 360.$
CL	Light value $0. \leq CL \leq 1.$
CS	Saturation value $0. \leq CS \leq 1.$

10.3.2 HLS to RGB

```
CALL IGHTOR (CH,CL,CS,CR*,CB*,CG*)
```

Action: This routine convert a HLS colour into an RGB colour.

Parameter description:

CH	Hue value $0. \leq CH \leq 360.$
CL	Light value $0. \leq CL \leq 1.$
CS	Saturation value $0. \leq CS \leq 1.$
CR	Red value $0. \leq CR \leq 1.$
CG	Green value $0. \leq CG \leq 1.$
CB	Blue value $0. \leq CB \leq 1.$

10.4 Conversion between character string and numbers

Often it is necessary to convert a Fortran character string into a number (integer or real) or vice versa. For example, routine IGMENU returns some parameters as character strings and it is often necessary to convert these into numbers. Also, to print graphically the result of a computation with ITX it is necessary to convert a number into a character string. The routines described in this paragraph allow these kinds of conversions.

10.4.1 Character to integer

```
CALL IZCTOI (CHVAL,IVAL*)
```

Action: Converts the character string CHVAL into the integer IVAL.

Parameter description:

CHVAL Character string.

IVAL Integer.

10.4.2 Character to real

```
CALL IZCTOR (CHVAL,RVAL*)
```

Action: Converts the character string CHVAL into the real RVAL.

Parameter description:

CHVAL Character string.

RVAL Real.

10.4.3 Integer to character

```
CALL IZITOC (IVAL,CHVAL*)
```

Action: Converts the integer IVAL into character string CHVAL.

Parameter description:

IVAL Integer.

CHVAL Character string.

10.4.4 Real to character

```
CALL IZRTOC (RVAL,CHVAL*)
```

Action: Converts the real RVAL into character string CHVAL.

Parameter description:

RVAL Real.

CHVAL Character string.

Chapter 11: Examples of HIGZ output

The graphical results of the examples below are reproduced directly from the PostScript output of and introduced into this manual.

HIGZ test program

```
      PROGRAM HIGZEX
*.==========>
*.
*.           HIGZ TEST PROGRAM
*.
*..=========>
      COMMON/PAWC/H(20000)
      LOGICAL INTRAC
      CHARACTER*80 STR
      CHARACTER*(*) HZFILE
+SELF,IF=IBM,IF=-PSCRIPT.
      PARAMETER (HZFILE='/HIGZ METAFILE')
+SELF,IF=IBM,IF=PSCRIPT.
      PARAMETER (HZFILE='/HIGZ PS')
+SELF,IF=-IBM,IF=-PSCRIPT.
      PARAMETER (HZFILE='higz.metafile')
+SELF,IF=-IBM,IF=PSCRIPT.
      PARAMETER (HZFILE='higz.ps')
+SELF.

*.-------------------------------------------
*
+SELF,IF=IBM.
      CALL ERRSET(151,999,-1)
+SELF,IF=IBM,IF=X11.
      CALL INITC()
+SELF.
      OPEN(10,FILE=HZFILE,FORM='FORMATTED',STATUS='UNKNOWN')
      CALL MZEBRA(-3)
      CALL MZPAW(20000,' ')
      CALL IGINIT(0)
      IF(.NOT.INTRAC(DUMMY))THEN
         INTER=0
         KWTYPE=0
      ELSE
         CALL IGWKTY(KWTYPE)
         INTER=1
      ENDIF
      CALL IGSSE(6,KWTYPE)
      IF(INTER.EQ.0)GOTO 10
      CALL HIEX1
      CALL IRQST(1,1,ISTAT,NCH,STR)
*
*          Switch to alpha mode. Note that IGSSE has preset the
*          workstation identifier to 1
*
      CALL IGSA (1)
*
      PRINT *, ' Example 1 completed'
      CALL HIEX2
```

```
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IGSA (1)
      PRINT *, ' Example 2 completed'
*
      CALL HIEX3
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IGSA (1)
      PRINT *, ' Example 3 completed'
*
      CALL HIEX4
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IGSA (1)
      PRINT *, ' Example 4 completed'
*
   10 CALL HIEX5
      IF(INTER.EQ.0)GOTO 20
      CALL IGSA (1)
      PRINT *, ' Example 5 completed'
*
*          Replay some pictures from the HIGZ metafile
*
      CALL HIEX6
      CALL IGSA (1)
      PRINT *, ' Example 6 completed'
*
   20 CALL IGEND
      END
```

Example of basic HIGZ. Polylines and fill areas

```
      SUBROUTINE HIEX1
*
      COMMON /QUEST/ RQUEST(100)
      DIMENSION XZ(86),YZ(86)
      DATA XZ/
     +   0.6250,0.6875,0.9063,0.7500,0.7500,0.6875,0.6250,0.6875
     +  ,0.7500,0.8750,0.9688,1.0313,1.1563,1.2500,1.3125,1.5000
     +  ,1.6875,1.9375,2.0000,2.1250,2.1875,2.1875,2.2500,2.2500
     +  ,2.4375,2.4375,2.4688,2.5313,2.5313,2.5000,2.6250,2.6250
     +  ,2.7500,2.7188,2.7188,2.7188,2.9375,3.4375,3.7500,4.0625
     +  ,4.1250,4.0625,4.1250,4.1875,4.3125,4.3125,4.3125,4.3438
     +  ,4.3125,4.4375,4.5000,4.4375,4.4375,4.5625,4.5938,4.7188
     +  ,4.7813,4.7500,4.5313,4.5000,4.6250,4.6875,4.7188,4.7500
     +  ,4.8750,4.9625,4.9063,4.7500,4.6875,4.6563,4.3750,3.6875
     +  ,3.0625,2.8125,2.4375,2.0313,1.6563,1.4688,1.3438,1.3750
     +  ,1.4375,1.2500,1.1250,1.0000,0.8750,0.6250/
      DATA YZ/
     +   4.8750,4.6563,4.3750,4.1250,3.8750,3.6250,3.4375,3.3125
     +  ,3.1875,3.1563,3.2188,3.3438,3.5000,3.5938,3.6875,3.5625
     +  ,3.3125,3.0938,2.8438,2.7000,2.2188,1.8750,1.2813,1.0625
     +  ,1.0625,1.8750,2.5000,2.4688,2.1875,1.9688,1.5000,1.2500
     +  ,1.2500,1.5313,2.0625,2.6250,2.5938,2.6563,2.7500,3.0000
     +  ,2.7188,2.1250,1.6563,1.4375,1.4688,1.6250,2.0313,2.3125
     +  ,2.6250,2.3125,2.0625,1.6250,1.5000,1.5000,1.6250,2.0313
     +  ,2.3125,2.5000,2.7500,2.9375,3.2500,3.6250,3.2500,2.8125
     +  ,2.6250,2.6875,3.0625,3.5625,3.8750,4.0625,4.1875,4.1250
     +  ,4.0313,4.0938,4.0625,4.2500,4.4875,4.5000,4.4688,4.6875
     +  ,4.8750,4.7188,4.5250,4.4688,4.7188,4.8750/
      DATA NZ/86/
*
*          Define the size of the Picture in cm
*
      CALL ICLRWK(0,1)
      CALL IGRNG(14.5,14.5)
      R  = RQUEST(11)
      XL = RQUEST(12)
      YB = RQUEST(13)
      CALL IGBOX(0.,14.5,0.,14.5)
      CALL IGTEXT(7.25,13.5,'HIGZ example 1',0.6,0.,'C')
*
*          Define a new Normalization transformation for each new object
*          The viewports are set in the centimeter space defined by IGRNG
*
      CALL ISWN(10,0.,5.,0.,5.)
      CALL ISVP(10,0.5*R+XL,6.5*r+XL,6.5*R+YB,11.5*r+YB)
      CALL ISELNT(10)
      CALL IPL(NZ,XZ,YZ)
*
      CALL ISWN(20,0.,5.,0.,5.)
      CALL ISVP(20,7.5*R+XL,14.*r+XL,6.5*R+YB,11.5*r+YB)
      CALL ISELNT(20)
      CALL ISMK(29)
      CALL IPM(NZ-1,XZ,YZ)
      CALL IPL(NZ  ,XZ,YZ)
*
```

```
      CALL ISWN(30,0.,5.,0.,5.)
      CALL ISVP(30,0.5*R+XL,6.5*r+XL,0.5*R+YB,5.5*r+YB)
      CALL ISELNT(30)
      CALL ISFAIS(3)
      CALL ISFASI(256)
      CALL IFA(NZ-1,XZ,YZ)
*
      CALL ISWN(40,0.,5.,0.,5.)
      CALL ISVP(40,7.5*R+XL,14.*r+XL,0.5*R+YB,5.5*r+YB)
      CALL ISELNT(40)
      CALL ISFASI(290)
      CALL IFA(NZ-1,XZ,YZ)
      CALL ISFAIS(0)
      CALL IFA(NZ-1,XZ,YZ)
*
      END
```

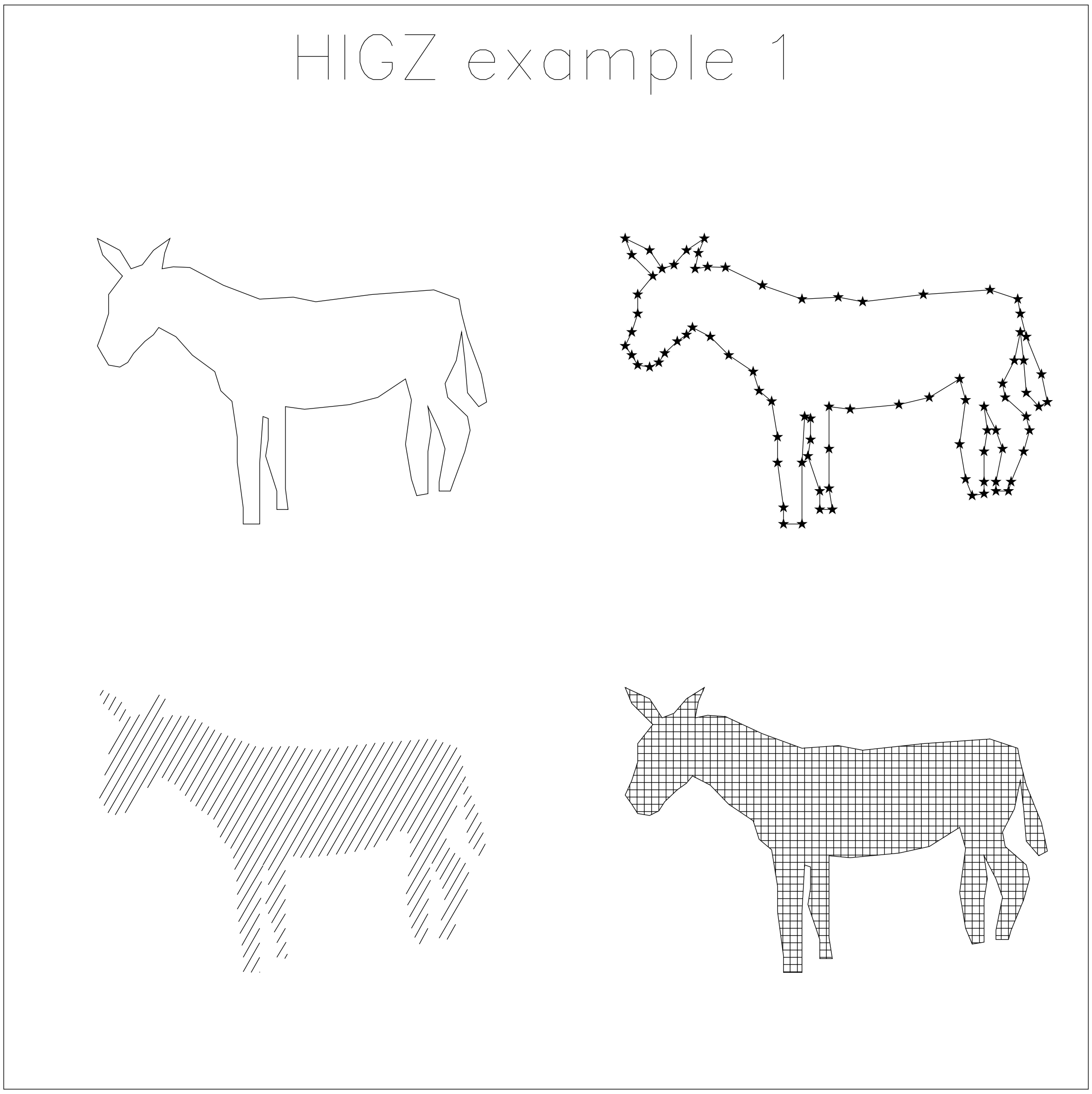


Figure 11.1: Result of first HIGZ example

Example to plot the table of HIGZ software characters

```
      SUBROUTINE HIEX2
*
      CHARACTER*6 KD1,KD2
      CHARACTER*45 KDG
      CHARACTER*3 KTEXT
      CHARACTER*1 CHOPT
      DIMENSION XPOS(6),X(5),Y(5)
      DATA KD1/' < < <'/
      DATA KD2/'  [[""'/
      DATA KDG/'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,+-*/=()'/
      DATA XLONG,YTOP/16.,24./
      DATA SIZE,ANGLE/0.3,0./
*
      CALL IGRNG(20.,24.)
      CALL ICLRWK(0,1)
*
      XW = XLONG/12.
      DO 10 I = 1,6
         XPOS(I) = (2*I-1)*XW + 2.5
   10 CONTINUE
*
*              Draw the frame
*
      YLONG  = 46*1.5*SIZE + 5*1.5*SIZE
      X(1)   = XPOS(1) - XW
      X(2)   = XPOS(6) + XW
      X(3)   = X(2)
      X(4)   = X(1)
      X(5)   = X(1)
      Y(1)   = YTOP
      Y(2)   = Y(1)
      Y(3)   = Y(1) - YLONG
      Y(4)   = Y(3)
      Y(5)   = Y(1)
      CALL IPL(5,X,Y)
      DO 20 I = 1,5
         X(1)   = XPOS(I) + XW
         X(2)   = X(1)
         Y(1)   = YTOP
         Y(2)   = Y(1) - YLONG
         CALL IPL(2,X,Y)
   20 CONTINUE
      X(1)   = XPOS(1) - XW
      X(2)   = XPOS(6) + XW
      Y(1)   = YTOP - 5.*SIZE
      Y(2)   = Y(1)
      CALL IPL(2,X,Y)
*
*             Draw box titles
*
      Y1     = YTOP - 2.*SIZE
      Y2     = Y1 - 2.*SIZE
      CHOPT='C'
      CALL IGTEXT(XPOS(1),Y1,'Upper'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(1),Y2,'Roman'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(2),Y1,'Lower'  ,SIZE,ANGLE,CHOPT)
```

```
      CALL IGTEXT(XPOS(2),Y2,'Roman'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(3),Y1,'Upper'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(3),Y2,'Greek'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(4),Y1,'L<OWER' ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(4),Y2,'G<REEK' ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(5),Y1,'U<PPER' ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(5),Y2,'Special',SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(6),Y1,'Lower'  ,SIZE,ANGLE,CHOPT)
      CALL IGTEXT(XPOS(6),Y2,'Special',SIZE,ANGLE,CHOPT)
*
      YP = YTOP - 6.*SIZE
      DO 40 I = 1,45
         YP = YP - 1.5*SIZE
         DO 30 J = 1,6
            KTEXT=KD1(J:J)//KD2(J:J)//KDG(I:I)
            CALL IGTEXT(XPOS(J),YP,KTEXT,SIZE,ANGLE,CHOPT)
  30     CONTINUE
  40  CONTINUE
*
      END
```

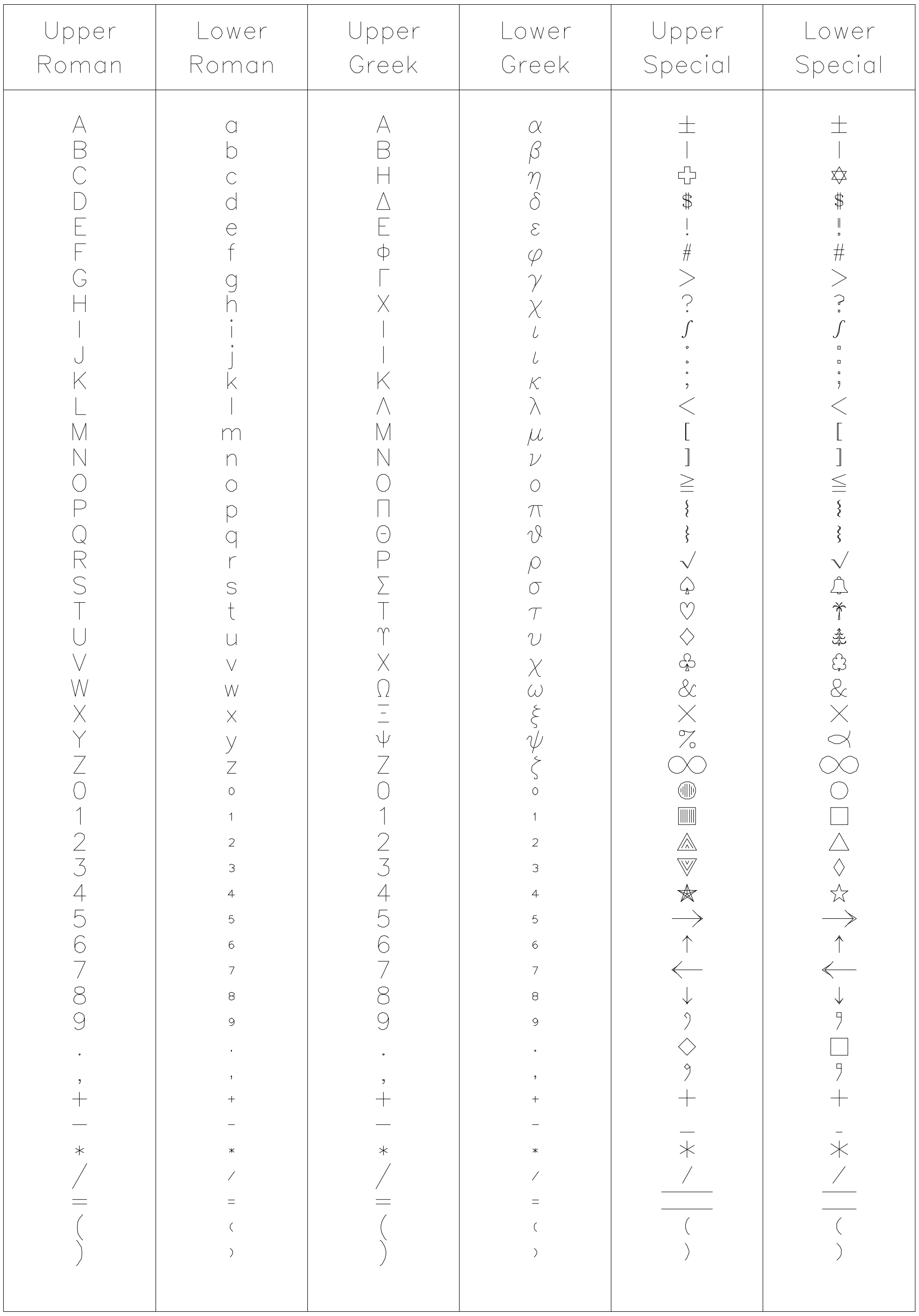


Figure 11.2: Result of plotting HIGZ software characters

Advanced example to draw text (based on a PAW macro from W.Walk)

```
      SUBROUTINE HIEX3
*
      DIMENSION X(3),Y(3)
*
      CALL IGRNG(14.6,18.)
      CALL ICLRWK(0,1)
      CALL IGBOX(0.,14.6,0.,18.)
      CALL IGSET('PASS',10.)
      CALL IGSET('CSHI',0.005)
      CALL ISFAIS(1)
      CALL ISTXCI(1)
      CALL ISTXFP(-104,2)
      CALL ISCHH(0.6)
      CALL ISTXAL(2,0)
      CALL ITX(7.3,17.,'Exclusive Toponium Decays')
      CALL ISTXFP(0,2)
      CALL ISFACI(1)
      CALL IGBOX(5.,7.,15.,14.9)
      CALL IGBOX(5.,7.,3.,2.9)
      CALL IGBOX(3.,5.,14.,13.9)
      CALL IGBOX(3.,5.,2.,1.9)
      CALL IGBOX(10.,12.,13.,12.9)
      CALL IGBOX(10.,12.,12.,11.9)
      CALL IGBOX(10.,12.,11.,10.9)
      CALL IGBOX(6.,8.,12.4,12.3)
      CALL ISPLCI(3)
      X(1)=6.
      X(2)=11.
      X(3)=6.
      Y(1)=15.
      Y(2)=13.
      Y(3)=3.
      CALL IPL(3,X,Y)
      Y(2)=12.
      CALL IPL(3,X,Y)
      Y(2)=11.
      CALL IPL(3,X,Y)
      CALL ISPLCI(2)
      X(2)=4.
      Y(2)=14.
      CALL IPL(3,X,Y)
      Y(2)=2.
      CALL IPL(3,X,Y)
      CALL ISPLCI(4)
      X(2)=X(3)
      Y(2)=1.5
      CALL IPL(2,X(2),Y(2))
      X(1)=X(2)-0.2
      X(3)=X(2)+0.2
      Y(1)=Y(2)+0.3
      Y(3)=Y(1)
      CALL IPL(3,X,Y)
      CALL ISTXCI(4)
      CALL IGTEXT(6.,0.5,'e^+!e^-! or [m]^+![m]^-!',0.5,0.,'C')
      CALL IGTEXT(6.,15.2,'2^3!S?1--!',0.5,0.,'C')
      CALL IGTEXT(6.,3.2,'1^3!S?1--!',0.5,0.,'C')
```

```
      CALL IGTEXT(11.,13.2,'1^3!P?2++!',0.5,0.,'C')
      CALL IGTEXT(11.,12.2,'1^3!P?1++!',0.5,0.,'C')
      CALL IGTEXT(11.,11.2,'1^3!P?0++!',0.5,0.,'C')
      CALL IGTEXT(7.,12.6,'1^1!P?1+-!',0.5,0.,'C')
      CALL IGTEXT(4.,14.2,'2^1!S?0-+!',0.5,0.,'C')
      CALL IGTEXT(4., 2.2,'1^1!S?0-+!',0.5,0.,'C')
      CALL ISTXCI(6)
      CALL IGTEXT(4.5,15.,'[Q]?2S!',0.5,0.,'R')
      CALL IGTEXT(7.5,2.75,'[Q]?1S! (80 GeV)',0.5,0.,'L')
      CALL IGTEXT(2.5,13.75,'[c]?t!&^,!',0.5,0.,'R')
      CALL IGTEXT(2.5,1.75,'[c]?t!',0.5,0.,'R')
      CALL IGTEXT(12.5,13.,'[h]^2!&?t!',0.5,0.,'L')
      CALL IGTEXT(12.5,12.,'[h]^1!&?t!',0.5,0.,'L')
      CALL IGTEXT(12.5,11.,'[h]^0!&?t!',0.5,0.,'L')
      CALL ISTXCI(3)
      CALL IGTEXT(1.,9.,'E1',0.5,0.,'C')
      CALL ISTXCI(2)
      CALL IGTEXT(3.,9.,'M1',0.5,0.,'C')
      CALL ISTXCI(3)
      CALL IGTEXT(8.8,14.8,'100 MeV',0.4,0.,'L')
      CALL IGTEXT(8.5,6.,'800 MeV',0.4,0.,'L')
      CALL ISTXCI(6)
      CALL IGTEXT(9.4,14.2,'BR 2"Y',0.3,0.,'L')
      CALL IGTEXT(8.9,5.4,'BR 30"Y',0.3,0.,'L')
      CALL IGSET('*',0.)
*
      END
```

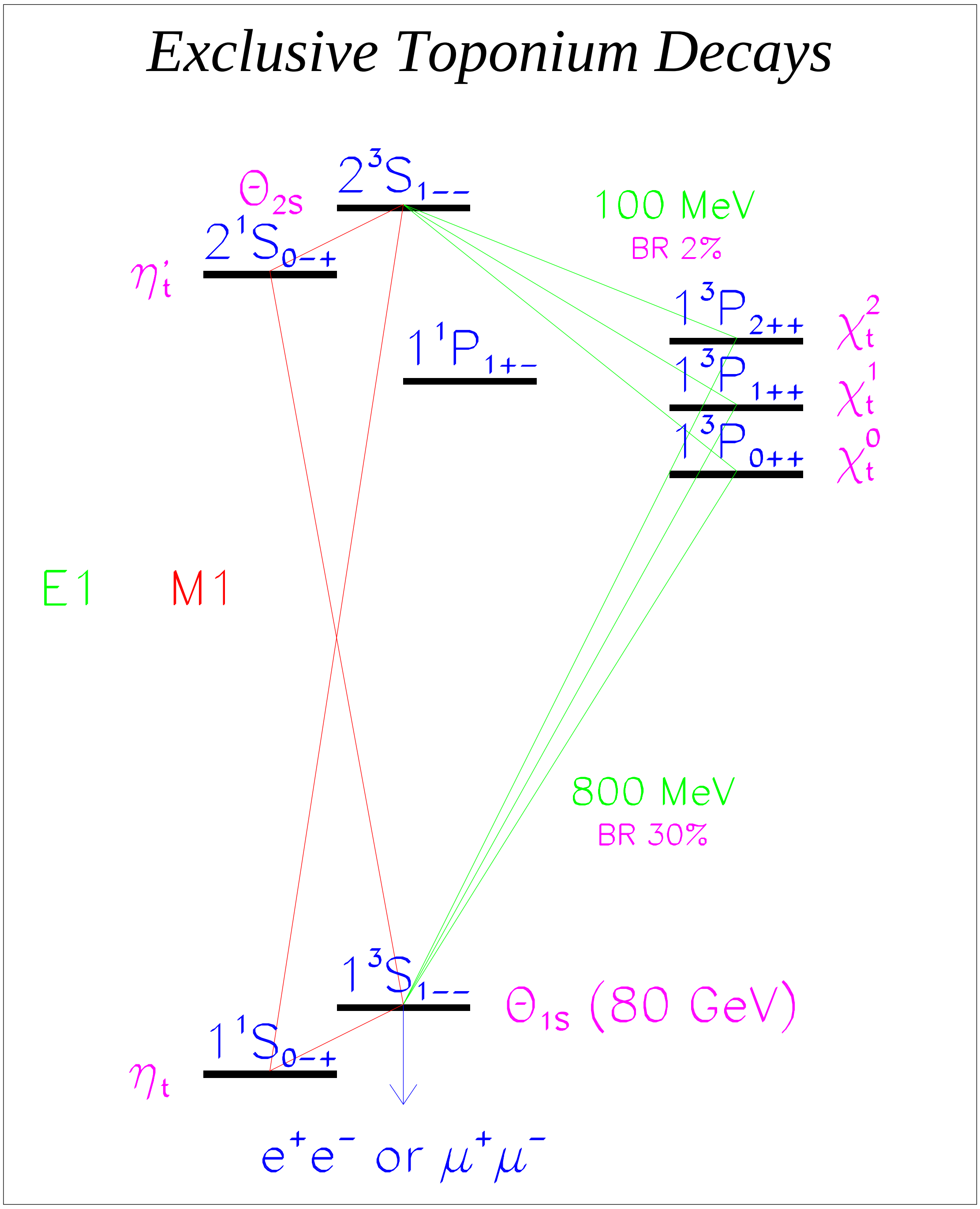


Figure 11.3: Result of HIGZ example 3 (toponium decay scheme)

Examples of graphs and histograms

```
      SUBROUTINE HIEX4
*
      COMMON /QUEST/ RQUEST(100)
      DIMENSION X(10),Y(10),V(10)
      DATA Y/2.,3.,5.,4.,7.,10.,11.,9.,10.,4./
      DATA X/0.,16.,8*0./
      DATA V/-1.5,1.,2.,4.,4.5,6.,9.,10.,14.,17./
*
      CALL IGRNG(15.,18.)
      R  = RQUEST(11)
      XL = RQUEST(12)
      YB = RQUEST(13)
      CALL ICLRWK(0,1)
      CALL ISTXFP(-13,1)
*
      CALL ISWN(10,0.,18.,-1.,12.)
      CALL ISVP(10,8.*R+XL,14.*R+XL,11.*R+YB,17.*R+YB)
      CALL ISELNT(10)
      CALL ISMK(29)
      CALL IGHIST(10,X,Y,'AHCP')
*
      CALL ISWN(20,0.,18.,0.,12.)
      CALL ISVP(20,R+XL,7.*R+XL,11.*R+YB,17.*R+YB)
      CALL ISELNT(20)
      CALL IGHIST(10,X,Y,'AB')
*
      CALL ISWN(30,-4.,19.,-1.,13.)
      CALL ISVP(30,R+XL,14.*R+XL,R+YB,10.*R+YB)
      CALL ISELNT(30)
      CALL IGAXIS(-3.,19.,1.,1.,-3.,19.,20510,' ')
      CALL IGSET('LASI',0.5)
      CALL IGAXIS(-3.,-3.,1.,12.,1.,12.,510,'H')
      CALL ISMK(21)
      CALL IGRAPH(10,V,Y,'LP')
      CALL ISLN(2)
      CALL IGRAPH(10,V,Y,'C')
      CALL IGSET('*',0.)
*
      END
```

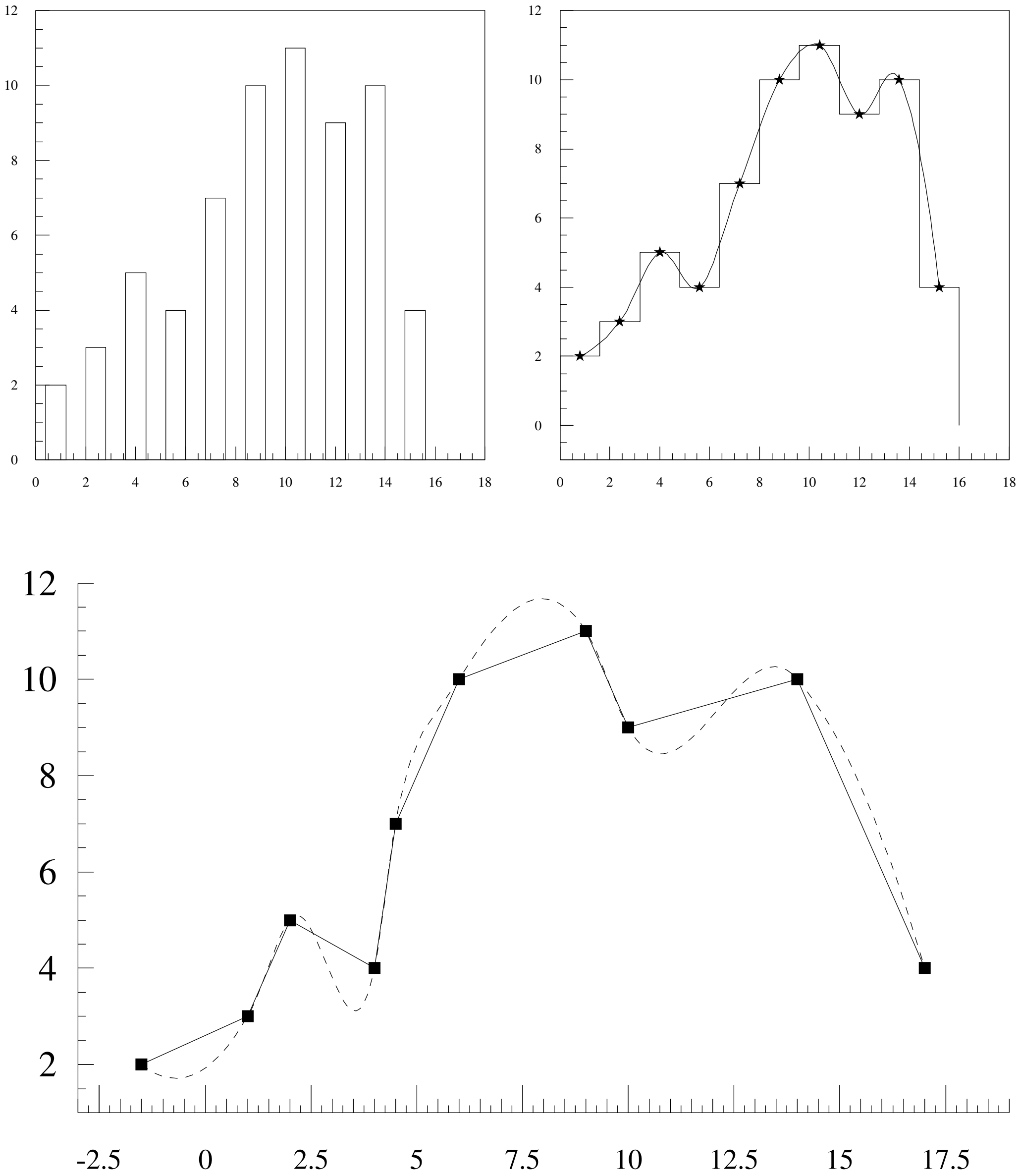


Figure 11.4: Result of HIGZ example 4 (graphs and histograms)

Example using HIGZ and PostScript metafiles

```
      SUBROUTINE HIEX5
*
*          Open HIGZ metafile
*          and repeat previous examples
*
      PRINT *,' Writing higz metafile'
      CALL IGZSET('Z')
      CALL IZOPEN(1,'Pictures','higz.rz','AN',1024,ISTAT)
      CALL IZPICT('ZEBRA','M')
      CALL HIEX1
      CALL IZPICT('SOFT-TABLE','M')
      CALL HIEX2
      CALL IZPICT('TOPONIUM','M')
      CALL HIEX3
      CALL IZPICT('GRAPH','M')
      CALL HIEX4
      CALL IZOUT('GRAPH',ICYCLE)
      CALL IGSA (1)
*
*          Open PostScript metafile
*          and repeat previous examples
*
      PRINT *,' Writing PostScript metafile'
      CALL IGZSET('G')
      CALL IGMETA(-10,0)
      CALL HIEX1
      CALL HIEX2
      CALL HIEX3
      CALL HIEX4
      CALL IGMETA(0,0)
*
      END
```

Display pictures in HIGZ files and invoke the HIGZ editor

```
      SUBROUTINE HIEX6
*
      CHARACTER*10 STR
      DATA ICYCLE/999/
*
*           List contents of the ZEBRA/RZ file
*
      CALL RZLDIR(' ',' ')
*
*           Read some pictures into memory and display
*
      CALL IGSET('AURZ',0.)
      CALL IZIN('ZEBRA',ICYCLE)
      CALL IZPICT('ZEBRA','D')
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IZIN('TOPONIUM',ICYCLE)
      CALL IZPICT('TOPONIUM','D')
      CALL IRQST(1,1,ISTAT,NCH,STR)
*
*           Edit PICT4
*           Select options in the graphics menu
*           For example select the item ARROW in the
*           menu 'PRIMITIVES', select the type of arrow
*           by clicking in the box 'ATTR' and try to superimpose
*           a double-arrow on the picture.
*           Try to change the font and the font size for the top graphs
*           Note that the HIGZ graphics editor can be invoked
*           from PAW (PICTURE/MODIFY command).
*
      CALL IZGED('GRAPH',' ')
*
      END
```

Part II

HPLOT – Reference Section

Chapter 12: Introduction

HPLOT is a Fortran callable facility for producing HBOOK[6] output on graphic devices other than the line printer. Its main design objective is to be able to produce drawings and slides of a quality suitable for talks and publications. To this end, it does not produce all the numeric information of the HBOOK output routines (which give what can be regarded as working histograms) but, on the other hand, it is not restricted by the line printer resolution or character size. The reader is of course supposed to be familiar with the HBOOK package.

The present version of HPLOT has been developed in the context of the Physics Analysis Workstation project PAW[2].

HPLOT can be used either in **BATCH** mode or interactively with PAW. When running in **BATCH**, one can write a metafile via the HIGZ/GKS packages and interpret these metafiles using the standard utilities such as GRCONV, GRVIEW and GRPLOT (see e.g. [3, 5]). PostScript file can also be produce with the native HIGZ PostScript driver. This way is certainly now the most popular because it doesn't need any translation programs to generate the paper output.

Users are strongly encouraged to use the PAW system to make good quality pictures. The HPLOT functionality described in this manual is available interactively in PAW.

12.1 A simple example

As an introductory example to HPLOT consider an already existing program using HBOOK, where one wants to plot all created histograms saving all pictures into a GKS or PostScript metafile.

Simple HPLOT **program**

```
      PROGRAM TEST
      COMMON/PAWC/H(20000)
*
      CALL HLIMIT(20000)          ! Initialize HBOOK
      CALL HBOOK1(...             ! Book and fill histograms with HBOOK
      CALL HBOOK2(...
*
      CALL HISTDO                 ! Print all histograms on lineprinter
*
      CALL HPLINT(0)              ! Initialize HPLOT
      CALL HPLOT(0,' ',' ',0)     ! Write all histograms to metafile
      CALL HPLEND                 ! Close HPLOT
      END
```

On VM/CMS a file definition:

FILEDEF 3 DISK HPLOT METAFILE A (RECFM F LRECL 80

must have been made beforehand for the output metafile HPLOT METAFILE. The latter can be visualized on various devices as desired, e.g. with the GRVIEW utility if it is a GKS metafile or with any PostScript previewer if it is a PostScript file.

Chapter 13: Reference Guide

13.1 Overview of HPLOT calls

Name	Action	Page
HPLABL	to define alphanumeric labels lists	144
HPLAER	to draw asymetric error bars	144
HPLARC	to draw an arc of circle	145
HPLAX	to add a comment to the axes	145
HPLBOX	to draw a box on the picture	146
HPLCAP	to switch or/off metafile output	146
HPLCOM	to add a comment	146
HPLCON	to draw a contour plot	147
HPLDO	to plot all histograms (like HISTDO)	147
HPLEGO	to plot a scatter plot as a 3 dim view	147
HPLEND	to terminate HPLOT	147
HPLERR	to draw error bars	148
HPLFRA	to define (and draw) a frame	148
HPLFUN	to draw a function	149
HPLGIV	to return size of the current zone	149
HPLINE	to draw straight lines	150
HPLINT	to initialize HPLOT	150
HPLKEY	to draw a symbol and its explanation	151
HPLNT	to plot a N-tuple	152
HPLNUL	to draw a picture or zone frame	153
HPLNXT	user routine called before each new frame	153
HPLOC	for graphics input	154
HPLOPT	to define options	155
HPLOT	to plot histograms or plots	157
HPLPRO	to plot a scatter plot and its projections	160
HPLPTO	wait after each plot	160
HPLSET	to redefine parameters	161
HPLSIZ	to set or read picture dimensions	169
HPLSOF	to draw software characters	169
HPLSUR	to plot a scatter plot as a 3 dim view	170
HPLSYM	to draw symbols on the picture	170
HPLTAB	to draw an histogram as a table	171
HPLTIT	to draw a title	190
HPLUSR	user routine called after each plot	190
HPLWIR	to draw cross-wires on a picture.	192
HPLZOM	to zoom a picture	193
HPLZON	to split the picture into zones	193

```
CALL HPLABL (NUM, NB, CHLAB)
```

Action: By default, labels used by axis are numeric labels. This routine, allows the user to define up to nine alphanumeric set of labels (numbered from `1` to `9`). These labels can then be used in subsequent calls producing axis. This routine limits the lenght of the alphanumeric labels at 32 characters.

Parameter description:

`NUM`	List number.
`NB`	Number of labels .
`CHLAB(NB)`	Array of `CHARACTER` defining the list contents.

See also `HPLSET`.

```
CALL HPLAER (XU ,YU, DXU1, DXU2, DYU1, DYU2, N, CHOPT, ISYM, USIZE)
```

Action: Allows the user to draw his own (asymetric) error bars on the picture. Error bars computed by HBOOK are automatically plotted by HPLOT. They can, however, be turned off via the routine `HPLOPT` with the option '`NEAH`' ("No Errors And Histogram"). The character with code `ISYM` is plotted at the point given by the coordinates `(XU,YU)`

Parameter description:

`XU`	Array of floating point numbers specifying the X-coordinate of the centre point of the error bars to be drawn.
`YU`	Array of floating point numbers specifying the Y-coordinate of the centre point of the error bars to be drawn.
`DXU1-DXU2`	Arrays of floating point numbers specifying the half length in the X direction of the error bars, i.e. the error bar is drawn from `XU(I) - DXU1(I)` to `XU(I) + DXU2(I)`.
`DYU1-DYU2`	Arrays of floating point numbers specifying the half length in the Y direction of the error bars, i.e. the error bar is drawn from `YU(I) - DYU1(I)` to `YU(I) + DYU2(I)`.
`N`	Length of the arrays `XU, YU, DXU1, DXU2, DYU1, DYU2`.
`CHOPT`	`CHARACTER` variable determining the coordinate system of the `XU...` coordinates: `' '` means that the coordinates are expressed in histogram coordinates (of the last drawn histogram). Error bars are drawn. `'C'` (or `'CM'` for compatibility) means that the coordinates are expressed in cm. `'W'` a new window is defined and axis are drawn. `'0'` error bars are drawn (default). `'1'` small lines at the end of the error bars are drawn. `'2'` error rectangles are drawn. `'3'` a filled area is drawn through the end points of the vertical error bars. `'4'` a smoothed filled area is drawn through the end points of the vertical error bars.
`ISYM`	Code of the symbol to be drawn at each point (see `HPLSYM`). `0` means that no symbols is printed.

USIZE Size of the symbol to be drawn at each point (see HPLSYM). 0 means that no symbols is printed.

Remarks:

- See also HPLERR.
- The options '0', '1', '2', '3' and '4' can be cumulated.
- HPLAER must be called after HPLFRA or HPLOT.

```
CALL HPLARC (XC, YC, RAD, PHI1, PHI2)
```

Action: Draws an arc of circle.

Parameter description:

XC	X coordinate of the centre of the arc in cm.
YC	Y coordinate of the centre of the arc in cm.
RAD	Radius of the arc in cm.
PHI1	The arc of circle is drawn from PHI1 to PHI2 (degrees).
PHI2	If PHI1 = PHI2 (0 for instance) then a complete circle is drawn.

Note that the line type can be changed with parameter DMODin HPLSET.

Remark:

HPLARC is only kept for compatibility with earlier versions. Users are encouraged to switch to the more powerful HIGZ routine IGARC.

```
CALL HPLAX (CHXTIT, CHYTIT)
```

Action: Prints titles along the X and/or Y axes of the plot.

Parameter description:

CHXTIT	Character string to be printed on the X axis. ' ' means that no label has to be drawn on the X axis.
CHYTIT	Character string to be printed on the Y axis. ' ' means that no label has to be drawn on the Y axis.

Remarks:

- Each title is printed either to the right and below the axis (X) or at the top and to the left (Y).
- The position of the axis labels may be redefined with HPLSET (XLAB and YLAB).
- The labels are only printed on an already existing picture, i.e. HPLAX must be called **after** HPLOT.

```
CALL HPLBOX (XLOW, YLOW, XUP, YUP, CHOPT)
```

Action: Draws a rectangular box on the picture. The area delimited by the rectangle is filled according to the fill area interior style index and fill area style index set in `HPLSET` with parameter `BTYP`, and to the fill area colour index set in `HPLSET` with parameter `BCOL`. The contour is always drawn.

Parameter description:

`XLOW` X coordinate of the lower left hand corner of the box.

`YLOW` Y coordinate of the lower left hand corner of the box.

`XUP` X coordinate of the upper right hand corner of the box.

`YUP` Y coordinate of the upper right hand corner of the box.

`CHOPT` Character variable determining the coordinate system of the `XLOW...` coordinates:

- `' '` means that the coordinates are expressed in histogram coordinates (of the last drawn histogram).
- `'C'` (or `'CM'` for compatibility) means that the coordinates are expressed in centimeters.

Remark:

`HPLBOX` must be called after `HPLFRA` or `HPLOT`.

```
CALL HPLCAP (IFILE)
```

Action: Changes the status of metafile and terminal output.

Parameter description:

`IFILE` Logical unit for the GKS metafile.

- `10` Enable terminal output and metafile output to Fortran unit `IFILE`
- `0` Enable terminal output only
- `-10` Enable metafile output to Fortran unit `IFILE` only.

Remark:

`HPLCAP` is only kept for compatibility with previous versions. It is now strongly recommended to use HIGZ Routine `IGMETA (IFILE, METAFILE-TYPE)`, with metafile types `4`, `-111`, `-112`, etc.
`HPLCAP` may be called at any time to redefine `IFILE`. In batch execution `IFILE` must always be negative.

```
CALL HPLCOM (XM, YM, CHTIT)
```

Action: Adds a comment on the picture.

Parameter description:

`XM` X coordinate (in cm) of the first character of the string to be drawn.

`YM` Y coordinate (in cm) of the first character of the string to be drawn.

`CHTIT` Character variable containing the string to draw.

`HPLCOM` is used to add comments to an existing picture, i.e. it must be called **after** `HPLOT`.
A more powerful routine (`HPLSOF`) permits to plot any character at a given size or angle. See also the HIGZ routines `IGTEXT` and `ITX`.

```
CALL HPLCON (ID, NLEVEL, IFLAG)
```

Action: Draws a contour plot from a 2 dim histogram.

Parameter description:

ID	Histogram identifier
NLEVEL	Number of contour lines
IFLAG	Option flag 0 Use colour to distinguish contours. 1 Use line style to distinguish contours. 2 Line style and colour are the same for all contours.

See also the routine HPLTAB.

```
CALL HPLDO (LUN)
```

Action: This routine is the HPLOT equivalent of HISTDO. It is equivalent to:

```
CALL HPLINT(LUN)
CALL HPLOT(0,' ',' ',0)
CALL HPLEND
```

```
CALL HPLEGO (ID,THETA, PHI)
```

Action: Plots two-dimensional histograms as solid objects viewed from infinity. The "object" can be rotated specifying the polar coordinates THETA and PHI.

Parameter description:

ID	histogram ID.
THETA	θ viewing angle in degrees.
PHI	ϕ viewing angle in degrees.

See also the routine HPLTAB.

```
CALL HPLEND
```

Action: Terminates the HPLOT package, and writes the termination page on the line printer. This gives the total number of plots produced and the number of plots stored as HIGZ pictures (see HPLOPT for option 'ZFL').

Remark:

HPLEND must be called after all other HPLOT routines.

```
CALL HPLERR (XU, YU, DXU, DYU, N, CHOPT, ISYM, USIZE)
```

Action: Allows the user to draw his own error bars on the picture. Error bars computed by HBOOK are automatically plotted by HPLOT. They can, however, be turned off via the routine HPLOPT with the option 'NEAH' ("No Errors And Histogram"). The character with code ISYM is plotted at the point given by the coordinates (XU,YU)

Parameter description:

XU Array of floating point numbers specifying the X-coordinate of the centre point of the error bars to be drawn.

YU Array of floating point numbers specifying the Y-coordinate of the centre point of the error bars to be drawn.

DXU Array of floating point numbers specifying the half length in the X direction of the error bars, i.e. the error bar is drawn from XU(I) - DXU(I) to XU(I) + DXU(I).

DYU Array of floating point numbers specifying the half length in the Y direction of the error bars, i.e. the error bar is drawn from YU(I) - DYU(I) to YU(I) + DYU(I).

N Length of the arrays XU, YU, DXU, DYU.

CHOPT CHARACTER variable determining the coordinate system of the XU... coordinates:

- ' ' means that the coordinates are expressed in histogram coordinates (of the last drawn histogram). Error bars are drawn.
- 'C' (or 'CM' for compatibility) means that the coordinates are expressed in centimeters.
- 'W' a new window is defined and axis are drawn.
- '0' error bars are drawn (default).
- '1' small lines at the end of the error bars are drawn.
- '2' error rectangles are drawn.
- '3' a filled area is drawn through the end points of the vertical error bars.
- '4' a smoothed filled area is drawn through the end points of the vertical error bars.

ISYM Code of the symbol to be drawn at each point (see HPLSYM). 0 means that no symbol is printed.

USIZE Size of the symbol to be drawn at each point (see HPLSYM). 0 means that no symbol is printed.

Remarks:

- See also HPLAER.
- The options '0', '1', '2', '3' and '4' can be cumulated.
- HPLERR must be called after HPLFRA or HPLOT.

```
CALL HPLFRA (X1, X2, Y1, Y2, CHOPT)
```

Action: Defines (and draws) a frame. By defaults axis labels and tick marks are drawn.

Parameter description:

`X1` X coordinate of the lower left hand corner of the frame.

`Y1` Y coordinate of the lower left hand corner of the frame.

`X2` X coordinate of the upper right hand corner of the frame.

`Y2` Y coordinate of the upper right hand corner of the frame.

`CHOPT` `CHARACTER` variable specifying the options desired:

- `'S'` A convenient way to redefine the frame for the current zone.
- `'A'` The axis labels and tick marks are not drawn.
- `'B'` The box around the histogram is not drawn.

```
CALL HPLFUN  (XU, YU, N, CHOPT)
```

Action: Draws a smooth curve (splines) on the picture. The curve will pass through all the points and will be smoothed to form a line as a function of `X`. If the option `AST` has been set on with the routine `HPLOPT`, each point `(XU(I),YU(I))` is stamped with a star.

Parameter description:

`XU` Array containing the X-coordinates of the points be to connected.

`YU` Array containing the Y-coordinates of the points be to connected.

`N` Dimension of the arrays `XU` and `YU`

`CHOPT` `CHARACTER` variable determining the coordinate system of the `XU, YU` coordinates.

- `' '` means that the coordinates are expressed in histogram coordinates (of the last drawn histogram).
- `'C'` (or `'CM'` for compatibility) means that the coordinates are expressed in centimeters.

Remarks:

- If `CHOPT = 'CM'`, `HPLGIV` can be used to determine the boundary of the current picture.
- The line type can be changed with parameter `DMOD` of `HPLSET`.
- No check is made in `HPLFUN` that the `XU` (`YU`) values are in ascending order.
- If `N<3`, routine `HPLINE` is called instead and a warning message is output.
- The limit `N<1002` must be satisfied[1].
- `HPLFUN` must be called after `HPLFRA` or `HPLOT`.
- See also the HIGZ routine `IGRAPH`.

```
CALL HPLGIV  (XL*, YL*, XH*, YH*)
```

Action: Returns the lower and upper coordinates of the current zone in cm.

Parameter description:

[1] to parameter `NMAX` defined in the Patchy `KEEP` sequence `HPL11` in the HPLOT source PAM file.

XL* X coordinate of the lower left hand corner of the current picture or zone.

YL* Y coordinate of the lower left hand corner of the current picture or zone.

XH* X coordinate of the upper right hand corner of the current picture or zone.

YH* Y coordinate of the upper right hand corner of the current picture or zone.

Remarks:

- HPLGIV must be called after HPLOT.
- See also the HIGZ routine IGQWK.

```
CALL HPLINE  (XU, YU, N, CHOPT)
```

Action: Draws a polyline on the picture.

Parameter description:

XU Array containing the X-coordinates of the points be to connected by straight lines.

YU Array containing the Y-coordinates of the points be to connected by straight lines.

N Dimension of the arrays XU and YU. Note that N-1 lines will be drawn.

CHOPT CHARACTER variable determining the coordinate system of the XU, YU coordinates:

- ' ' means that the coordinates are expressed in histogram coordinates (of the last drawn histogram).
- 'C' (or 'CM' for compatibility) means that the coordinates are expressed in centimeters.

Remarks:

- If CHOPT = 'CM', HPLGIV can be used to determine the boundary of the current picture.
- The line type can be changed with parameter DMOD of HPLSET.
- The limit N<1002 must be satisfied[2].
- See also the HIGZ routine IPL.
- HPLINE must be called after HPLFRA or HPLOT.

```
CALL HPLINT  (IWTYP)
```

Action: Initialises the HPLOT package and especially the graphic package environment (HIGZ).

Parameter description:

IWTYP Workstation type.See **appendix B** for the list of valid workstation types. If IWTYP=0 no graphics workstation will be open. This special value should be used when working in **batch** mode. In this case, to direct output to a metafile, use IGMETA.

[2] to parameter NMAX defined in the Patchy KEEP sequence HPL11 in the HPLOT source PAM file.

Remarks:

- The HPLOT error messages will appear on the same output file as the HBOOK error message file.
- The HBOOK result file can be changed by the HBOOK routine `HOUTPU`, and the HBOOK error message file can be changed by the HBOOK routine `HERMES`.
- `HPLINT` must be called **before** any other HPLOT routines, but **after** the HBOOK initialization routine `HLIMIT`.

```
CALL HPLKEY  (XC, YC, ISYM, CHTIT)
```

Action: Draws a symbol and its explanation. The symbol numbers are the same as for `HPLSYM`, and `HPLKEY` provides a convenient method of annotating the different symbols on a plot.

Parameter description:

`XC`	X coordinate (in cm) of the first character of the string preceded by the symbol `ISYM`.
`YM`	Y coordinate (in cm) of the first character of the string preceded by the symbol `ISYM`.
`ISYM`	Code of the symbol to be drawn (see `HPLSYM` for details).
`CHTIT`	`CHARACTER` variable containing the string to be drawn.

Remark:

For `HPLKEY` the "text" consists of the symbol followed by a space and then the characters of `CHTIT`, which will be in the same size as for comments (routine `HPLCOM`). This can be controlled by setting the value of the parameter `CSIZ` using routine `HPLSET`, which defines also the symbol size.

```
CALL HPLNT (IDN, ISEL, UWFUNC, IFROM, ITO, IVARX, IVARY)
```

Action: Draws two variables of an Ntuple as a scatterplot.

Parameter description:

IDN	Identifier of a Ntuple.
ISEL	Selection flag.
UWFUNC	Selection function.
IFROM	First event number.
ITO	Last event number.
IVARX	Number of the Ntuple variable to be plotted along X.
IVARY	Number of the Ntuple variable to be plotted along Y.

Routine HPLNT plots the correlation between two variables of an existing Ntuple IDN. For all events in the range IFROM to ITO the Ntuple variable with identifier IVARY will be plotted against the variable with identifier IVARX. A selection mechanism may be specified with the ISEL parameter. ISEL=0 means no selection. All events with numbers between IFROM to ITO included will be used in the plot. When ISEL is not zero, then an EXTERNAL user written function UWFUNC is called for each event with, as parameters the Ntuple array X and the value of ISEL. Routine UWFUNC should return the weight of the event. If UWFUNC=0 then the event is not included in the plot.

Example of the use of HPLNT

```
          . . .
      EXTERNAL WFUNC
*
*         To plot X(7) versus X(3) for the 5000 first events
*         of Ntuple 10 using the selection option 1.
*
      CALL HPLNT(10,1,WFUNC,1,5000,3,7)
          . . .
      FUNCTION WFUNC(X,ISEL)
      DIMENSION X(*)
      WFUNC=0.
      IF(ISEL.EQ.1)THEN
          IF(X(2)**2 +X(3)**2.LT.0.)WFUNC=1.
      ELSEIF(ISEL.EQ.2)THEN
          IF(X(2)**2 +X(3)**2.GT.5.)WFUNC=1.
      ELSE
          WFUNC=X(5)
      ENDIF
      END
```

Remarks:

- HPLNT works only on “Row Wise Ntuples”.
- In PAW, more possibilities are offer to draw Ntuples (including 3D).
- In an interactive PAW session the user function UWFUNC may be defined interactively using a Fortran syntax without recompilation and relinking.
- For more information about Ntuples, see the description of routine HBOOKN in the HBOOK manual.

CALL **HPLNUL**

Action: Draws a box in place of the histogram box and its contents.

Remark:

HPLNUL allows the user to draw a box for his own requirements. If windowing is in use (HPLZON), HPLNUL draws the box in the appropriate position. If windowing is not in use, or if HPLNUL draws a box on a new page, then the page number and the global title (if present) will also be drawn.

Routines HPLAX, HPLBOX, HPLCOM, HPLINE, HPLTIT, etc., can all be used to add information to the box. It is also possible to superimpose a histogram with:

```
CALL HPLOT(ID,'S',' ',0)
```

in which case no axis values or tick marks will be drawn.

CALL **HPLNXT**

Action: This is an HPLOT User routine. The user should not call it but provide, if he wishes, his own version to replace the do-nothing version automatically provided by HPLOT. This routine is called before each graphics clear screen operation i.e. it is intended to be used to pause an interactive program at the end of a graphics frame and, if required, to change program flow.

On some systems graphics input/output and Fortran input/output cannot be intermixed and in most systems Fortran input/output will simply start its text from wherever the graphics cursor was positioned. For these reasons an auxiliary HPLOT routine, called HPLPTO, to do simple text output and wait for input via graphics rather than Fortran has been provided.

Example of the use of HPLNXT

```
      SUBROUTINE HPLNXT
*         Optional user routine called before a new frame
      CHARACTER*30   STROUT,STRIN
*
      DATA STROUT/'TYPE QUIT OR RETURN'/
*         Issue a graphics prompt and read keyboard
      CALL HPLPTO(STROUT,STRIN)
*         Check for quit
      IF(STRIN.NE.'QUIT') RETURN
*         Clean up and stop
      CALL HPLEND
      STOP 99
      END
```

```
CALL HPLOC (NTPRI, NTLOC*, XLOC*, YLOC*, IDH*, ICX*, ICY*, ISTAT*)
```

Action: Picks a point on the current displayed picture and returns the information, related to the corresponding histogram. Picking is done with locator number 1.

Parameter description:

`NTPRI`	Normalisation transformation number with a priority. If `NTPRI<0` then automatic selection of `NTLOC`. If `NTPRI`≥`0` then transformation number `NTPRI` has priority.
`NTLOC`	Normalisation transformation number which has been picked.
`XLOC`	X coordinate in `NTLOC` units.
`YLOC`	Y coordinate in `NTLOC` units.
`IDH`	Histogram identifier corresponding to `NTLOC`.
`ICX`	Channel number in X for `IDH`.
`ICY`	Channel number in Y for `IDH` (if 2-dim histogram).
`ISTAT`	Locator return status

Remarks:

- `NTLOC` is returned with the value 0 when the point is outside the picture limits as defined by the `XSIZ/YSIZ` parameters. In this case `XLOC` and `YLOC` are given in Normalized Device Coordinates in the range `(0.,1.)`.
- `NTLOC` is returned with the value 1 when the point is somewhere on the picture, but not in a histogram box. In this case `XLOC` and `YLOC` are given in centimeters. To force `XLOC` and `YLOC` to be returned in centimeters independently of the position of the locator, set `NTPRI=1`.
- `NTLOC` returns values like 10, 20, 30, etc when the point is inside one of the histogram boxes as explained in chapter 14. In this case `XLOC` and `YLOC` are given in histogram coordinates.

```
CALL HPLOPT (CHOPT,N)
```

Action: Allows the user to change the options defined by default in HPLINT. HPLOPT can be called any number of times, each option remaining in effect until modified by a further call to `HPLOPT`.

Parameter description:

`CHOPT` `CHARACTER*4` array of options. Each word of the array defines a new option via a character string of four characters (see table below).

`N` Size of the array in words.

In table 13.1 the values in the column labelled **default** are those set at initialization by `HPLINT`.

Table 13.1: Overview of the `HPLOPT` options

Default	Alternative	Effect
' '	'A0', 'A1',...	Picture size. Predefined options are: A0, A1, A2, A3, A4, A5, A6
'NOPG'	'*P','**P', '***P'	Suppresses ('NOPG') or adds a 1, 2 or 3 digit page numbers to a plot (Each '*' stands for a digit). The page numbers are incremented automatically
'NEAH'	'EAH'	Plots Errors bars And Histogram, if both are present
'VERT'	'HORI'	Vertical or horizontal orientation of paper
'NAST'	'AST'	Functions are drawn with ('AST') or without ('NAST') asterisks in each channel.
'NCHA'	'CHA'	Scatter plot are plotted with dots randomised within each bin ('NCHA') or by printing a single character in the middle of the bin ('CHA')
'NHST'	'HSTA'	Filling statistics ('HSTA').
'SOFT'	'HARD'	Use SOFTware or HARDware characters
'TAB'	'NTAB'	tables (HTABLE) are plotted as tables ('TAB') or as scatter plots ('NTAB')
'NSQR'	'SQR'	The size of the histogram boxes is set to the largest square (SQR)
'HTIT'	'UTIT'	Option for printing titles. 'HTIT' means use the HBOOK titles, while 'UTIT' signals the use of user titles
'LINX'	'LOGX'	The scale for the X axis is linear or logarithmic.
'LINY'	'LOGY'	The scale for the Y axis is linear or logarithmic.
		Note that if in HBOOK the HIDOPT option 'LOGY' or HLOGAR was selected for a particular ID and if neither options 'LINY' nor 'LOGY' are selected then the scale will be logarithmic. If HLOGAR or HIDOPT with option 'LOGY' was called and the option 'LINY' is selected then the scale will be linear
'LINZ'	'LOGZ'	The scale for the Z axis is linear or logarithmic (for lego plots or surfaces).
'BOX'	'NBOX'	By default a rectangular box is drawn around a picture. 'NBOX' suppresses this box
'NTIC'	'TIC'	Cross-wires are drawn ('TIC') or not drawn ('NTIC') after each plot

Table 13.1: Overview of the HPLOPT options (continued)

Default	Alternative	Effect
'NSTA'	'STA'	Statistics information are printed ('STA') or not printed ('NSTA') on the picture
'NFIT'	'FIT'	Fit parameters are printed ('FIT') or not printed ('NFIT') on the picture
'NZFL'	'ZFL'	The picture is stored ('ZFL') or not stored ('NZFL') in a ZEBRA data base The procedure to create a HIGZ picture is given below.
'NZFL'	'ZFL1'	'ZFL1' has the same effect as 'ZFL', but only the picture last created is kept in memory.
'NPTO'	'PTO'	"Please Turn Over". With 'PTO' a carriage return is requested between each new plot.
'NBAR'	'BAR'	1-dimensional histograms are plotted as "Bar charts" ('BAR') or as contours ('NBAR')
'DVXR'	'DVXI'	Real ('DVXR') or integer ('DVXI') labels are computed for the X axis
'DVYR'	'DVYI'	Real ('DVYR') or integer ('DVYI') labels are computed for the Y axis
'GRID'	'NGRI'	Grid on X and Y axis
'NDAT'	'NDAT'	The date is printed or not on each plot
'NFIL'	'NFIL'	The file name is printed or not on each plot

Remarks:

- The parameters can be supplied in any order in array CHOPT. If two mutually exclusive options are given, the last one encountered is used i.e. CHOPT(2) takes precedence over CHOPT(1).
- The allowed range of metric paper sizes may be restricted at some installations by the physical size of the plotter.
- Once a value for the page number has been given, it will automatically be incremented for each new picture.
- If the options 'A3' or 'A4' are called, windowing is turned off (i.e. a call to HPLZON(1,1,1,' ') is performed). It is recommended that windowing is defined **after** HPLOPT to avoid this problem.
- When the option 'LOGX' is selected only the axes are drawn with a call to HPLOT or HPLTAB. This option is interesting when used with HPLERR, HPLAER, HPLSYM or HPLFUN.
- If option 'ZFL' is selected then all the subsequent graphics primitives are kept in memory to make a HIGZ picture. A name is automatically assigned to each HIGZ picture : PICT1, PICT2, ... Several pictures can be stored in memory. They can be saved in a ZEBRA/RZ direct access file and be modified with the HIGZ graphics editor. (See the HIGZ routines IZFILE, IZIN, IZOUT, IZPICT and IZGED and the last example at the end of the manual.)
- If option 'ZFL1' is selected only the last created picture is kept in memory.
- With the 'BAR' option parameter HTYP of HPLSET can be used to change the fill area interior style.
- If CHOPT(1) = 'SHOW' a list of all options and their current values is printed.

```
CALL HPLOT  (ID, CHOPT, CHCASE, NUM)
```

Action: Plots histogram ID.

Parameter description:

`ID`	Identifier of the histogram to be plotted. `ID=0` means plot all histograms.
`CHOPT`	`CHARACTER` variable containing the string of options.

`' '`	The histogram contour is drawn (1 dim histograms).
`'H'`	The histogram contour is drawn (1 dim histograms).
`'L'`	Draw a Line connecting bin contents (1 dim histogram).
`'*'`	An asterisk is drawn at the center of each histogram channel.
`'P'`	The current polymarker is drawn at the center of each histogram channel.
`'C'`	The histogram contour is drawn as a smooth curve (the curve will pass through the center of each channel and will be smoothed to form a line).
`'B'`	Bar chart format selected for 1 dim histograms.
`'S'`	The current histogram is superimposed on the previous picture (title, axes, page number are not redrawn).
`'K'`	Keep histogram in memory (in a ZEBRA bank). This option needs to be requested for later update of histogram (option `'U'`) or for addition of several histograms (option `'+'`) if several zones (with `HPLZON`) are in use.
`'U'`	Update histogram with identifier `ID`. Useful for dynamic histograms (when the content of the histogram changes with time). The new histogram content is superimposed on the previous one, and the scale is changed (with new axis labels if necessary).
`'+'`	The contents of histogram `ID` is added to the contents of the histogram on the current picture.
`'-'`	Same as '+' but the contains of the histogram is substract.
`'+-'`	Draw the for each bin delta between 2 histograms
`'A'`	If specified, axis are not drawn
`'BOX'`	Draw 2D histograms with proportionnal Boxes
`'ARR'`	Draw 2D histograms with Arrows
`'COL'`	Draw 2D histograms with Colors
`'LEGO'`	Draw as a Lego plot
`'LEGO1'`	Draw as a Lego (mode 1 see `HPLTAB`)
`'LEGO2'`	Draw as a Lego (mode 2 see `HPLTAB`)
`'SURF'`	Draw as a Surface
`'SURF1'`	Draw as a Surface (mode 1 see `HPLTAB`)
`'SURF2'`	Draw as a Surface (mode 2 see `HPLTAB`)
`'CONT'`	Draw 2D histograms as a Contour plot
`'SCAT'`	Draw 2D histograms a Scatter plot
`'TEXT'`	Draw 2D histograms with the contains of each cell

'CHAR'	Draw 2D histograms with a character set
'ARR'	Draw 2D histograms with arrows
'HIST'	Draw only the histogram
'FUNC'	Draw only the function (for example in case of fit)
'E'	Errors with current marker type and size are drawn.

CHCASE 4-CHARACTER string to select possible projections of a 2 dimensional histogram, e.g. slices in X. Possible values are: HIST, PROX, PROY, BANX, BANY, SLIX, SLIY.

NUM Integer which permits, together with parameter CHCASE, to further specify a given selection, e.g. third slice in X.

Remarks:

- When superimposing histograms with CHOPT = 'S' the line style for drawing the straight lines of the histogram, error bars and function is changed as follow :

first histogram	______	solid line
second histogram	_ _ _	(dash,blank,dash,blank)
third histogram	. . .	(dot,blank,dot,blank)
fourth histogram	_._._	(dash,dot,dash,dot)
fifth histogram	.....	(dot,dot,dot,dot)

  If more than five histograms are superimposed, HPLOT will loop round the symbols again. If three histograms are to be superimposed, but the second histogram requested does not exist, the third histogram will still be plotted with the third symbol (. .). Similarly if the second histogram is a scatter plot, the third histogram will take the third symbol.
- One can force a particular type of line style by calling routine HPLSET with parameter DMOD, e.g. CALL HPLSET('DMOD', 4.0) will force all lines to be drawn in dash-dot mode.
- When option 'S' is selected, the histogram is drawn with the viewport and window parameters of the first histogram plotted in the current zone.
- Option 'BAR' in HPLOPT can be used instead of CHOPT = 'B' to plot all 1 dimensional histogram as “bar charts”.
- The fill area interior style and style index can be changed with parameter HTYP in HPLSET (this parameter has to be set to draw a histogram as a hatched surface instead of a contour).
- The colour (contour or surface) of the histogram can be changed with parameter HCOL in HPLSET.
- The current polymarker (CHOPT = 'P') can be changed by calling HIGZ routine IGSET (parameter MTYP).
- If options 'U' or '+' are selected, and if several zones are requested, option 'K' must be used when the first histogram is drawn.

Example of the use of the option K and U

```
      program dice
      common /pawc/ h(100000)
*.------------------------------------------
*
      call igwkty(kwtype)
      call hlimit(100000)
      call hplint(kwtype)
*
      n      = 1000
      ifirst = 1
      call hplset('HCOL',1001.)
      call hplset('NDVX',-11.05)
      call hplopt('STAT',1)
      call hbook1(3,'Playing with two dice',11,2.,13.,0.)
      do j=1,n
         ix1=6.*rndm(.01234)+1
         ix2=6.*rndm(.56789)+1
         call hfill(3,float(ix1+ix2),0.,1.)
         if (ifirst.eq.1) then
            call hplot(3,'BK',' ',0)
            ifirst=0
         else
            call hplot(3,'BU',' ',0)
         endif
         call igterm
      enddo
*
      end
```

Two random numbers between 1 and 6 are generated and the histogram is filled with the sum of this numbers to simulate dice playing. The first time the histogram is plotted the option "LitK" is used to keep in memory a copy of the histogram in order to update it later. With the "U" option, HPLOT looks at the current kept histogram contents and update the plot with the new contribution without redrawing everything. This mechanism is used in data acquisition. The statistics are also updated.

```
CALL HPLPRO (ID, CHXTIT, CHYTIT)
```

Action: Draws a scatter plot and its X and Y projections (if present) on a plot with 2 by 2 zones. Separate titles may be given to the projections if required.

Parameter description:

`ID`	The HBOOK identifier of a 2 Dim histogram.
`CHXTIT`	`CHARACTER` string containing the title to be printed for the X projection. `' '` requests to print the histogram title for the X projection (unless option '`UTIT`' has been selected, in which case no title will be printed).
`CHYTIT`	`CHARACTER` string containing the title to be printed for the Y projection. `' '` requests to print the histogram title for the Y projection (unless option '`UTIT`' has been selected, in which case no title will be printed).

Remarks:

- This routine sets the zone option on entry, and turns it off before returning, therefore subsequent plots will be plotted in the default "unzoned" manner.
- The scatter plot is drawn last so that if `HPLAX` is called after `HPLPRO`, the axis titles will appear on the scatter plot.
- If option '`UTIT`' is selected before calling `HPLPRO`, no title will be printed on the 2 dim histogram itself (the titles for the projections depend on `CHXTIT` and `CHYTIT`, not '`UTIT`'). Therefore, it is possible to supply a title for the 2-D histogram with `HPLTIT`.

```
CALL HPLPTO (STROUT, STRIN)
```

Action: Displays the `CHARACTER` variable specified in the bottom left hand corner of the screen during an interactive graphics session, waits for some user keyboard input and returns the input (which may be just carriage return) in a `CHARACTER` variable.

Parameter description:

`STROUT`	`CHARACTER` variable to be displayed. The maximum length allowed will depend on the underlying graphics package.
`STRIN`	`CHARACTER` variable returned to the user. The maximum length allowed will depend on the underlying graphics package.

Remark:

When called in interactive graphics mode this routine does nothing. It is primarily intended to be called from the user routine `HPLNXT` at the end of each graphics frame so that a user can pause between frames.

`CALL HPLSET (CHOPT, VAR)`

Action: Sets one HPLOT parameter (see table 13.2 for more details). Note that if HPLSET in invoked with a parameter not describe in the table 13.2, the HIGZ routine IGSET is invoked with the same parameter value. If the parameter value is again not correct for IGSET, then an error message is displayed.

Parameter description:

CHOPT CHARACTER variable of length 4 identifying the parameter to be redefined.

VAR New value for the parameter specified.

Remarks:

- If VAR = 0 the corresponding parameter is set to its default value.
- If CHOPT = '* ', all parameters listed in the table are set to their default value.
- If CHOPT = 'SHOW' a list of all parameters is printed.
- HMAX is given in percent (default value is 90%).
- The values given to the parameters PTYP, BTYP and HTYP are fill area interior style. These parameters are installation dependent and even device dependent. If one wants to get the same result on all devices, use numbers defined on on the figure 3.3. The parameters PCOL, BCOL, HCOL are equivalent to PTYP, BTYP, HTYP, respectively, but instead of changing the hatch style, they change the colour of the same areas.
- If PCOL, BCOL, HCOL are between 1 and 99, then only the contour of the corresponding area is changed. If they are between 1001 and 1099, then the surface is filled with the corresponding fill area colour index. For PCOL, BCOL or HCOL the corresponding value of the Fill Area Interior Style (for PTYP, BTYP, HTYP) is automatically set to 1 (solid).
- It is possible to specify with one HPLSET call both the border and the inside color for the Histogram, Box Page, and Function (HCOL, BCOL, PCOL, FCOL).

Example of HCOL specification

```
     Ex:
                  +---- 1 The Histogram is filled
                  |     0 Only the border is drawn
                  |+--- Border color (here 2) if the histogram is filled
                  ||++- Inside color (here 3) if the histogram is filled
                  ||||  Border color if the histogram is not filled
                  ||||
                  VVVV
 CALL HPLSET('HCOL',1203.)
```

The same mechanism is also available for FCOL, BCOL and PCOL.
If PCOL, BCOL, HCOL or FCOL are between 1 and 99, then only the contour of the corresponding area is changed. If they are between 1001 and 1099, then the surface is filled with the colour determined by the corresponding fill area colour index (1 to 99). If they are between 1199 and 1999, then the surface is filled with the colour determined by the corresponding fill area colour index (1 to 99) and the border is drawn with the corresponding line color index (1 to 9).

If one of the *COL is greater than 1000 the corresponding value of the Fill Area Interior Style (for HTYP, BTYP, PTYP or FTYP) is automatically set to 1 (solid).
In addition, BCOL has two digits after the dot. The first one specifies the colour of the zone box shadowing and the second the colour of the statistic box shadowing.

– TFON, GFON, VFON and LFON must be set according the following convention :

  'X'FON = 10*IFON + IPREC

  where IFON and IPREC correspond respectively to the HIGZ attributes for "Text Font" and "Precision".
– *SIZ, *TYP, *COL, *WID and *FON define respectivly all the text sizes, the fill area type, the colors, the line width and the text fonts with the same values.
– The label sets defined by the routine HPLABL can be used for axes on all plots produced by HPLOT via the NDVX, NDVY and NDVZ parameters. These parameters have the following structure:

Example of NDVX specification

```
    CALL HPLSET('NDVX',i)      e.g.   CALL HPLSET('NDVX',512.)
or
    CALL HPLSET('NDVX',i.jk)   e.g.   CALL HPLSET('NDVX',10.25)
```

In the first case the number i contains 100 times the number of secondary divisions plus the number of primary divisions. (e.g. 512 means 12 primary and 5 secondary division. By adding 10000 times N3 to i a third level of divisions is available.
In the second case the number in front of the dot (i) indicates the total number of divisions, the first digit following the dot (j) the label identifier: LABNUM (see HPLABL) (if this number is equal to 0 numeric labels are drawn). The second digit after the (k) dot indicates the position where the labels have to be drawn (i.e. the **text justification** parameter, in this case 5, indicating horizontally written text centered on the interval). Study figures 13.1 and 13.2 for details.
These two figures show that the labels can be centered on the tick marks (1 to 4) or on the divisions (5 to 8). If the labels are centered on the tick marks, note that the number of items defined by the routine HPLABL must be equal to the number of tick marks (which is equal to the number of divisions **plus one**), otherwise the last alphanumeric label on the axis will be undefined. By default, the number of primary divisions given by CALL HPLSET('NDVX',n), CALL HPLSET('NDVY',n) or CALL HPLSET('NDVZ',n) is optimized to have a reasonable labelling. If the number of divisions has to be exactly equal to the number given by HPLSET, a negative value must be used i.e.:

Forcing an exact number of divisions

```
    CALL HPLSET('NDVX',-i)      e.g.   CALL HPLSET('NDVX',-512.)
or
    CALL HPLSET('NDVX',-i.jk)   e.g.   CALL HPLSET('NDVX',-10.25)
```

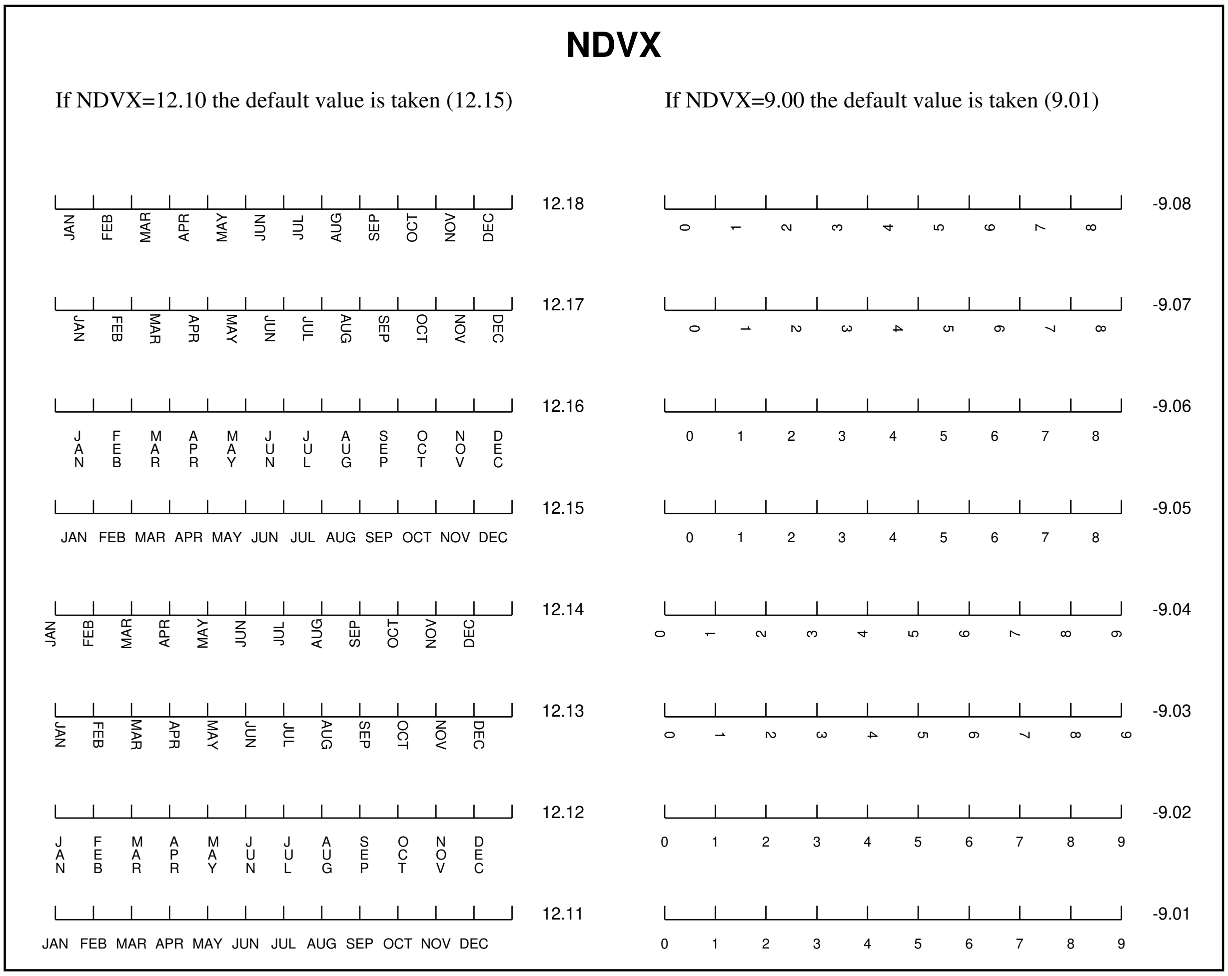


Figure 13.1: Example of labelling for horizontal axes

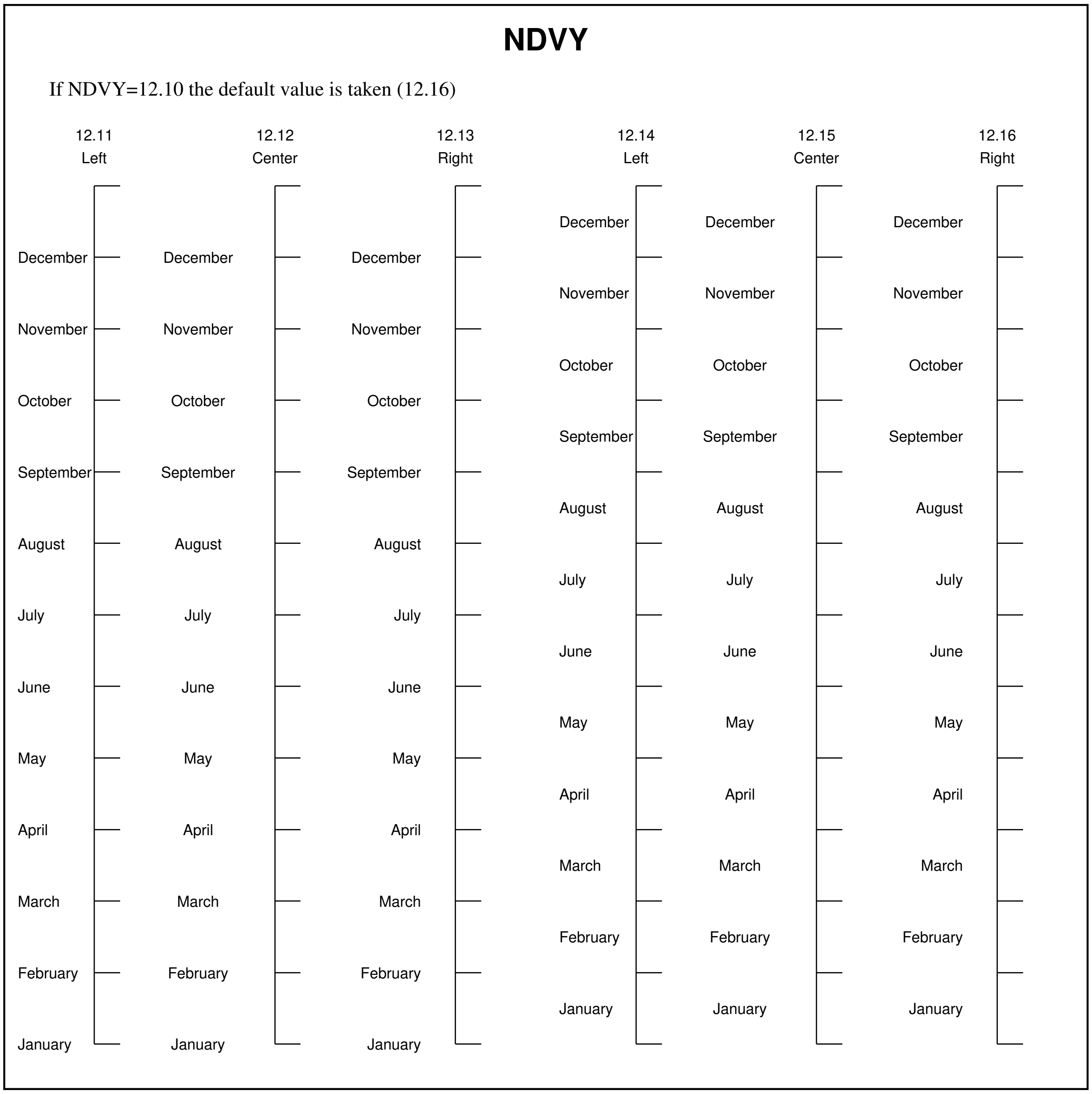


Figure 13.2: Example of labelling for vertical axes

Table 13.2: Overview of the HPLSET options

CHOPT	VAR **(default)**	**Explanation**
ASIZ	0.28 cm	axis label size
BARO	0.25	bar offset for “bar charts”
BARW	0.5	bar width for “bar charts”
BCOL	1	zone fill area colour index
BTYP	0	zone fill area style index
BWID	1	box line width
CFON	2	comment font (10*font+precision)
CSHI	0.03	character shift between two pass
CSIZ	0.28 cm	comment size
DASH	0.15	length of basic dashed segment for dashed lines
DATE	2	date position
DMOD	1	line style for histogram contour (see HPLOT)
ERRX	0.50	error on X (% of bin width)
FCOL	1	function fill area COLor
FILE	1	file name position
FIT	101	fit values to be plotted
FPGN	1	first PaGe Number
FTYP	0	function fill area TYPe
FWID	1	function line width
GFON	2	global title font (10*font+precision)
GRID	3	grid line type
GSIZ	0.28 cm	global title size
HCOL	1	histogram fill area colour index
HMAX	0.90	histogram maximum for scale (in percent)
HTYP	0	histogram fill area style index
HWID	1	histogram line width
KSIZ	0.28 cm	Hershey character size (cf. HPLKEY)
LFON	2	axis labels font (10*font+precision)
NDVX	10510.00	number of divisions for X axis
NDVY	10510.00	number of divisions for Y axis
NDVZ	10510.00	number of divisions for Z axis
PASS	1.	number of pass for software characters
PCOL	1	picture fill area colour index
PSIZ	0.28 cm	page number size
PTYP	0	picture fill area style index
PWID	1	picture line width
SMGR	0.	stat margin right (in percent)

Table 13.2: Overview of the `HPLSET` options (continued)

CHOPT	`VAR` **(default)**	**Explanation**
SMGU	0.	stat margin up (in percent)
SSIZ	0.28 cm	asterisk size (for functions)
STAT	1111	stat values to be plotted
TFON	2	general comments font (`10*font+precision`)
TSIZ	0.00 cm	histogram title size
VFON	2	axis values font (`10*font+precision`)
VSIZ	0.28 cm	axis values size
XCOL	1	X axis COLor
XLAB	1.40 cm	distance Y axis to labels
XMGL	2.00 cm	X margin left
XMGR	2.00 cm	X margin right
XSIZ	20.0 cm	length of picture along X
XTIC	0.30 cm	X axis tick mark length
XVAL	0.40 cm	distance Y axis to axis values
XWID	1	X ticks width
XWIN	2.00 cm	X space between zones
YCOL	1	Y axis COLor
YGTI	1.50 cm	Y position of global title
YHTI	1.20 cm	Y position of histogram title
YLAB	0.80 cm	distance X axis to labels
YMGL	2.00 cm	Y margin low
YMGU	2.00 cm	Y margin up
YNPG	0.60 cm	Y position for number of page
YSIZ	20.0 cm	length of picture along Y
YTIC	0.30 cm	Y axis tick mark length
YVAL	0.20 cm	distance X axis to axis values
YWID	1	Y ticks width
YWIN	2.00 cm	Y space between zones
2SIZ	0.28 cm	scatter plot and table character. size

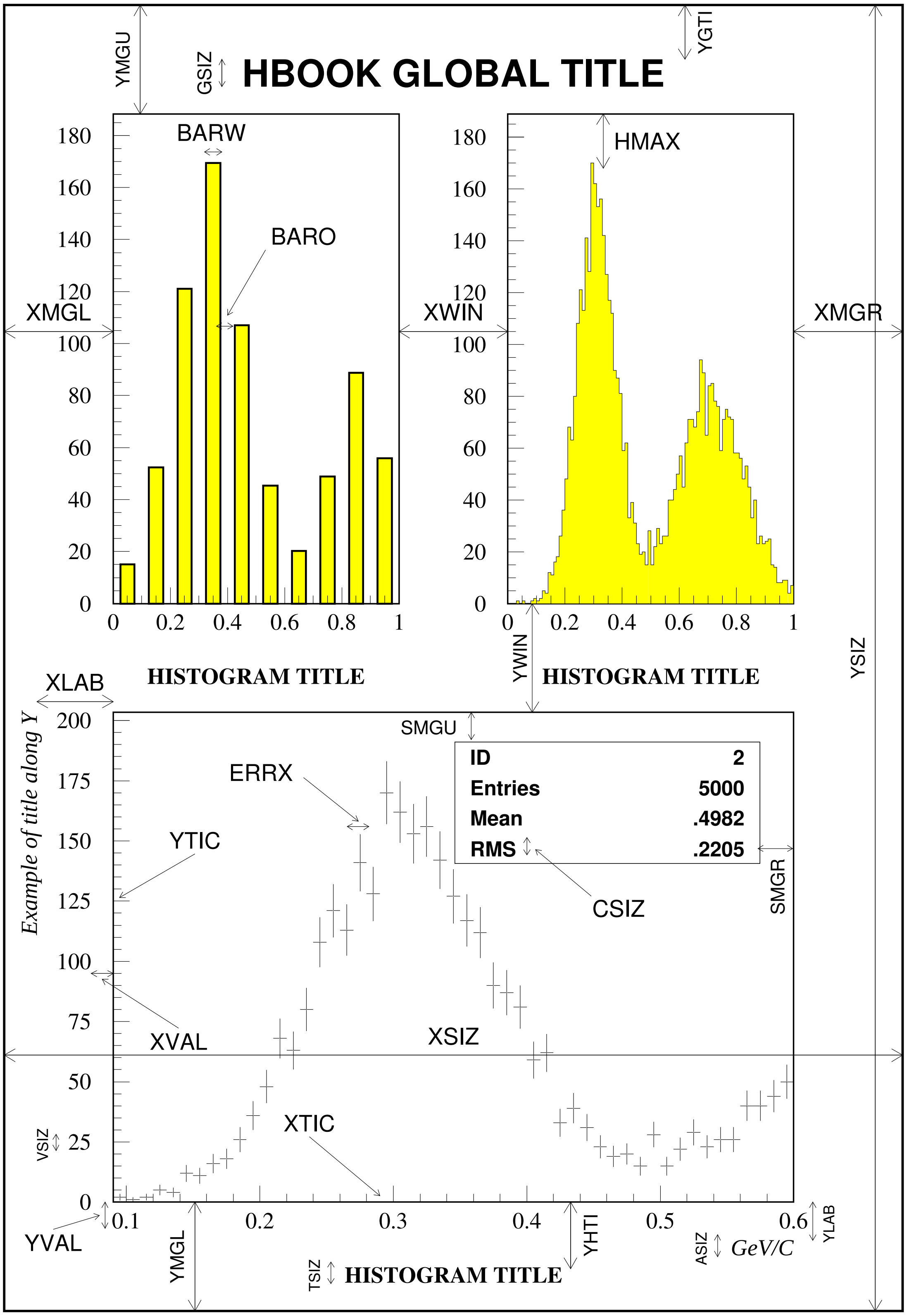


Figure 13.3: A graphical view of the HPLSET parameters.

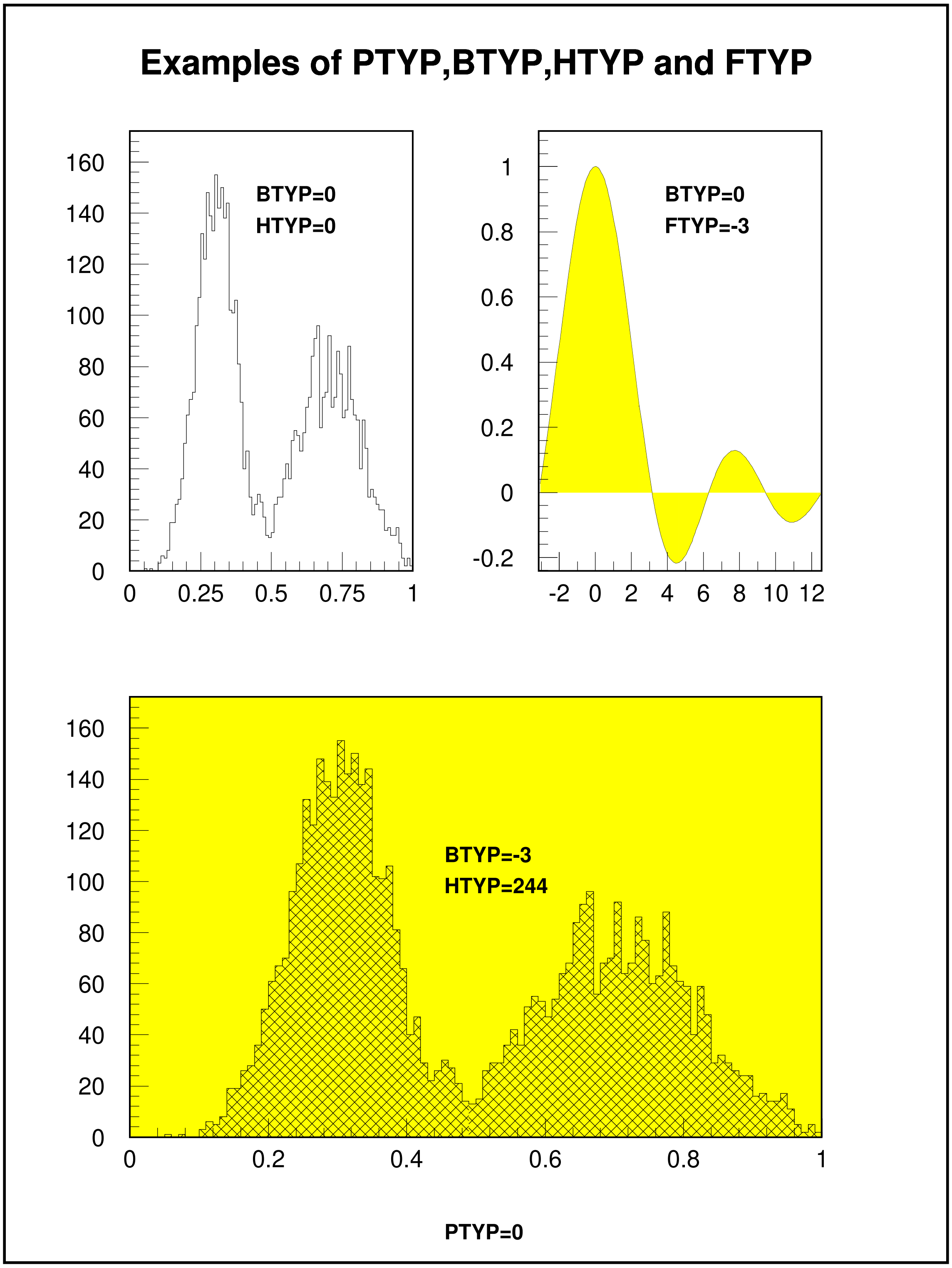


Figure 13.4: The HPLSET parameters PTYP, BTYP, HTYP

`CALL` **HPLSIZ** `(*XSIZE*, *YSIZE*, CHOPT)`

Action: Sets or reads picture size.

Parameter description:

`*XSIZE*` Size of the picture along X in centimeters

`*YSIZE*` Size of the picture along Y in centimeters

`CHOPT` `CHARACTER` variable specifying whether the picture size given as input or queried for output.

- `' '` Set the picture size (`XSIZE` and `YSIZE` are input parameters).
- `'R'` Read the picture size (`XSIZE` and `YSIZE` are output parameters).

`CALL` **HPLSOF** `(X, Y, CHTXT, SIZE, ANGLE, SIZMAX, IOPT)`

Action: Draw software characters.

Parameter description:

`X` X coordinate (in cm) of the first character of the string to be drawn.

`Y` Y coordinate (in cm) of the first character of the string to be drawn.

`CHTXT` `CHARACTER` variable containing the string to be drawn.

`SIZE` Size (in cm) for the characters.

`ANGLE` Rotation angle (in degrees) of the text to be drawn

`SIZMAX` Dummy (not used at present)

`IOPT` Integer specifying the option desired:

- `-1` First character of text is left adjusted to `X, Y`
- `0` Text is centered at `X,Y`
- `1` Last character of text is right adjusted at `X,Y`

List of escape characters and their meaning

`<` go to lower case

`>` go to upper case (default)

`[` go to greek (Roman = default)

`]` end of greek

`"` go to special symbols

`#` end of special symbols

`^` go to superscript

`?` go to subscript

`!` go to normal level of script

& backspace one character

$ termination character

Remarks:

- The order of alphabets is Roman, Greek and special.
- The way in which software characters are produced is to present a text as a string of characters which consists only of the allowed characters in Hollerith strings. This string is interpreted by routine HPLSOF as a string consisting both of control characters for such things as change of alphabet, upper and lower case, and others, and the equivalent of each character in the extended range given by a character in the limited set of 63 characters.
- Note that boldface characters may be simulated by with the PASS and CSHI attributes of HPLSET. The meaning of these attributes is the following: Every stroke used to display the character is repeated PASS times, at a distance (in percentage of the character height) given by CSHI.
- This routine directly invokes HIGZ routine IGTEXT. HPLSOF has been kept for compatibility with previous versions of HPLOT. Users are strongly invited to call HIGZ routine IGTEXT directly.

CALL **HPLSUR** (ID,THETA, PHI, MODE)

Action: Plots two dimensional histograms as solid objects viewed from infinity. The "object", can be rotated over a certain angle.

Parameter description:

ID	Histogram identifier.
THETA	Viewing angle θ in degrees.
PHI	Viewing angle ϕ in degrees.
MODE	Not used at present.

Remark:

See also the routine HPLTAB.

CALL **HPLSYM** (X, Y, N, ISYM, USIZE, CHOPT)

Action: Draws symbols or points on a picture.

Parameter description:

X	X coordinate of the center of the symbols to be drawn
Y	Y coordinate of the center of the symbols to be drawn
N	Dimension of arrays X and Y.
ISYM	Code of the symbol to be drawn (see below). If ISYM = 0 a point will be drawn.

USIZE Size of the symbol (in cm). If `USIZE = 0.` then the size of the symbol in cm will be taken from the current "Comment size", which can be changed with the parameter `CSIZ` of `HPLSET`.

CHOPT `CHARACTER` variable determining the coordinate system of `X` and `Y`.

- `' '` means that the coordinates are expressed in histogram coordinates (of the last drawn histogram). Error bars are drawn.
- `'C'` (or `'CM'` for compatibility) means that the coordinates are expressed in centimeters.

Remark:

Some symbols are meant to represent "blackened" symbols, but have to be drawn by a series of straight lines. Their effectiveness is therefore device-dependent. On PostScript files they are really filled. The symbol numbers correspond to the Hershey character set used by HIGZ routine `IGTEXT`, which can also be called directly to draw the same symbols or others.

CALL **HPLTAB** (ID, NPAR, PAR, CHOPT)

Action: Draws a table with the histogram ID according to the value of CHOPT.

Parameter description:

ID Histogram identifier.

NPAR Number of parameters in `PAR`.

PAR(NPAR) Array of real parameter. If `PAR(i)=0.` or `NPAR<i` a default value is taken.

CHOPT `CHARACTER` variable specifying the options selected. The possible value of `CHOPT` and the associate values of `PAR` are describe below. The default value of `CHOPT` is `'P'`.

`HPLTAB` **example**

```
      program hplotlego
*
      dimension par(6)
      common /pawc/ h(100000)
*.___________________________________________
*
      call igwkty(kwtype)
      call hlimit(100000)
      call hplint(kwtype)
      call hplmak
*
      call vzero(par,6)
      call hplsiz(9.,9.,' ')
      call hplset('YGTI',0.3)
      call hplset('XMGL',1.)
      call hplset('YMGL',2.)
      call hplset('XMGR',1.)
      call hplset('YMGU',0.5)
      call hplset('VSIZ',0.15)
      call hplset('YHTI',1.5)
      call hplset('MTYP',1.)
```

```
      call doeps(par,'SCAT')
      call doeps(par,'BOX')
      call doeps(par,'ARR')
      call doeps(par,'CONT')
      call doeps(par,'COL')
      call doeps(par,'TEXT')
      call doeps(par,'CHAR')
      par(1) = 30.
      par(2) = 30.
      call doeps(par,'LEGO')
      call doeps(par,'LEGO1')
      call doeps(par,'LEGO2')
      call doeps(par,'SURF')
      call doeps(par,'SURF1')
      call doeps(par,'SURF2')
      call doeps(par,'SURF3')
      call doeps(par,'SURF4')
      call doeps(par,'LEGOPOL')
      call doeps(par,'LEGOCYL')
      call doeps(par,'LEGOSPH')
      call doeps(par,'LEGOPSD')
      call doeps(par,'SURFPOL')
      call doeps(par,'SURFCYL')
      call doeps(par,'SURFSPH')
      call doeps(par,'SURFPSD')
      call hplend
      end

      subroutine doeps(par,chopt)
      character*(*) chopt
      character*32 name
      name     = 'hplot'
      name(6:) = chopt
      call cutol(name(6:))
      open(unit=10,file=name(1:lenocc(name))//'.eps'
     +,    form='formatted',status='unknown')
      call igmeta(10,-113)
      call hpltab(200,6,par,chopt)
      call igterm
      call igmeta(999,0)
      close(10)
      end

      subroutine hplmak
*
*  Creation of some histograms (based on HBOOK examples)
*
      common /hex2/ c1,c2,xm1,xm2,xs1,xs2
      external htfun1,htfun2
*.------------------------------------------
*
      c1  = 1.
      c2  = 0.5
      xm1 = 0.3
      xm2 = 0.7
      xs1 = 0.07
      xs2 = 0.12
```

```
*
      call hbfun2(200,'Test of 2-DIM plots',40,0.,1.,40,0.,1.,htfun2)
*
      end

      function htfun1(X)
      common /hex2/ c1,c2,xm1,xm2,xs1,xs2
*
      a1 = -0.5*((x-xm1)/xs1)**2
      a2 = -0.5*((x-xm2)/xs2)**2
      x1 = c1
      x2 = c2
      if(abs(a1).gt.1.e-4)x1 = c1*exp(a1)
      if(abs(a2).gt.1.e-4)x2 = c2*exp(a2)
      htfun1 = x1+x2
      end

      function htfun2(x,y)
      htfun2 = 100.*htfun1(x)*htfun1(y)
      end
```

CHOPT = 'SCAT' **Scatter plot**		
PAR **index**	PAR **values**	**default**
1	Marker type see ISMK.	1.
2	Maximum number of random points per cell	50.
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

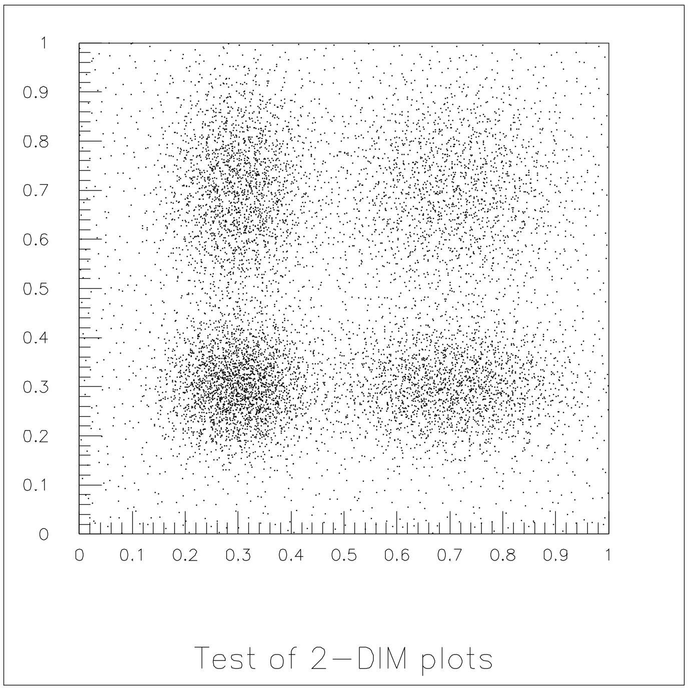


Figure 13.5: Example of HPLTAB with SCAT option

CHOPT = 'BOX' **Boxes**		
PAR **index**	PAR **values**	**default**
1	Not used	
2	Not used	
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

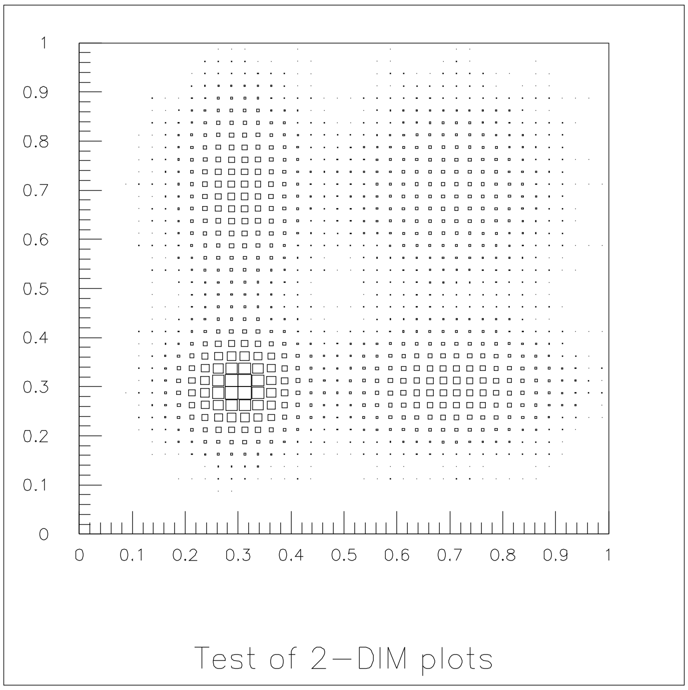


Figure 13.6: Example of HPLTAB with BOX option

CHOPT = 'ARR' **Arrows**		
PAR **index**	PAR **values**	**default**
1	Not used	
2	Not used	
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

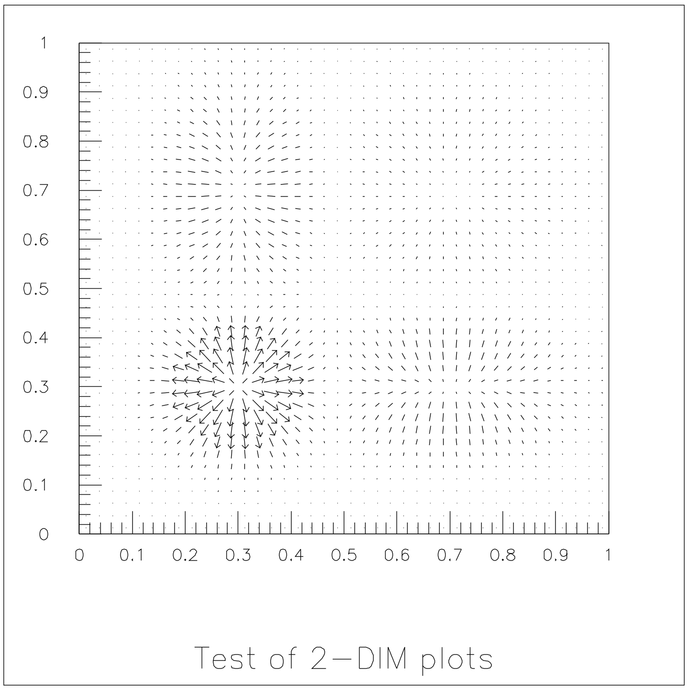


Figure 13.7: Example of HPLTAB with ARR option

`CHOPT = 'CONT'` **Contour plot**		
`PAR` **index**	`PAR` **values**	**default**
1	Nlevel (min=2 max=50)	`20.`
2	0 use colour to distinguish contours. Line type used is 1.	`0.`
	1 use line style to distinguish contours.	
	2 line style and colour are the same for all contours.	
	3 draw the contour with fill colored fill are.	
3	XMIN Lowest X-axis label	`IXMIN`
4	XMAX Highest Y-axis label	`IXMAX`
5	YMIN Lowest Y-axis label	`IYMIN`
6	YMAX Highest Y-axis label	`IYMAX`
7	ZMIN Lowest Z value	`ZMIN`
8	ZMAX Highest Z value	`ZMAX`
9	1000*IXMIN + IXMAX (Useful for ZOOM)	`1-NX`
10	1000*IYMIN + IYMAX (Useful for ZOOM)	`1-NY`

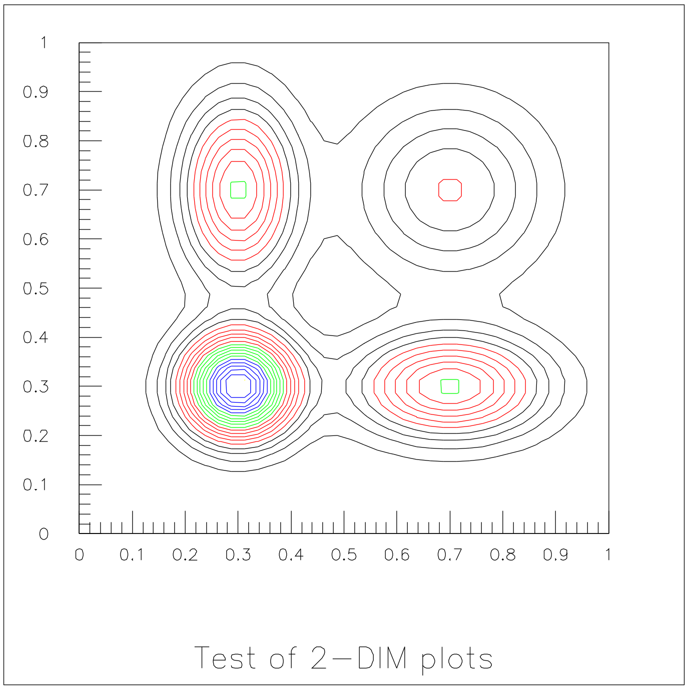


Figure 13.8: Example of `HPLTAB` with `CONT` option

CHOPT = 'COL' **COLour plot**		
PAR **index**	PAR **values**	**default**
1	0 use the standard 8 colours	0.
2	Not used	
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

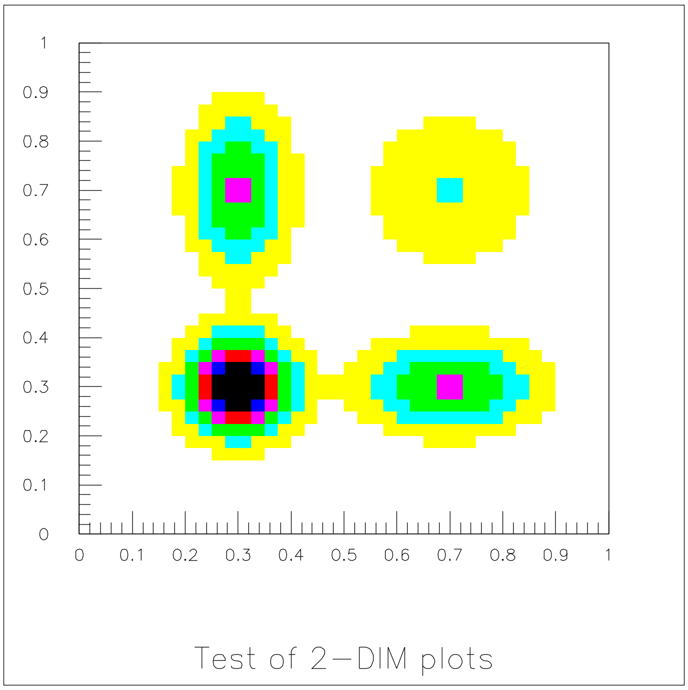


Figure 13.9: Example of HPLTAB with COL option

`CHOPT = 'TEXT'` **Table (Text)**		
`PAR` **index**	`PAR` **values**	**default**
1	Text font	`1.`
2	Text Precision	`0.`
3	`ZMIN` Lowest Z value	`ZMIN`
4	`ZMAX` Highest Z value	`ZMAX`
5	`1000*IXMIN + IXMAX` (Useful for ZOOM)	`1-NX`
6	`1000*IYMIN + IYMAX` (Useful for ZOOM)	`1-NY`

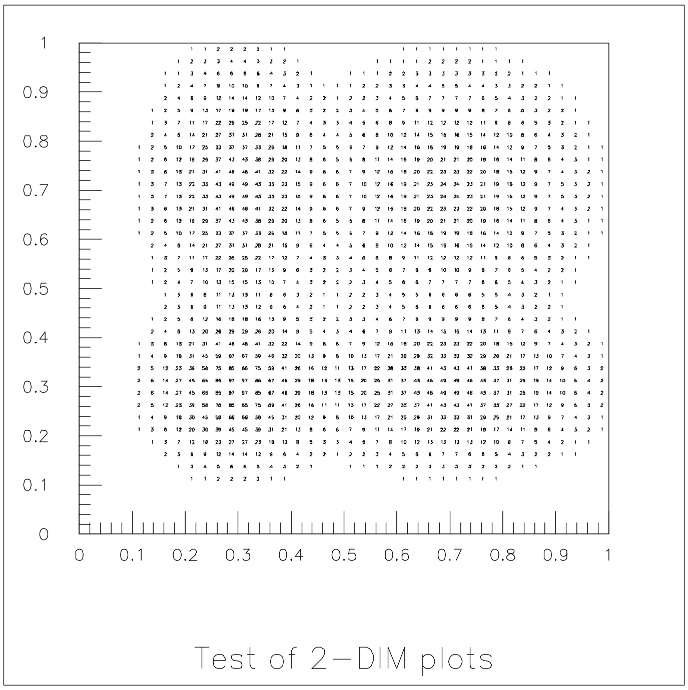


Figure 13.10: Example of `HPLTAB` with `TEXT` option

CHOPT = 'CHAR' **Character, the contains is one single character**		
PAR **index**	PAR **values**	**default**
1	Text font	1.
2	Text Precision	0.
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

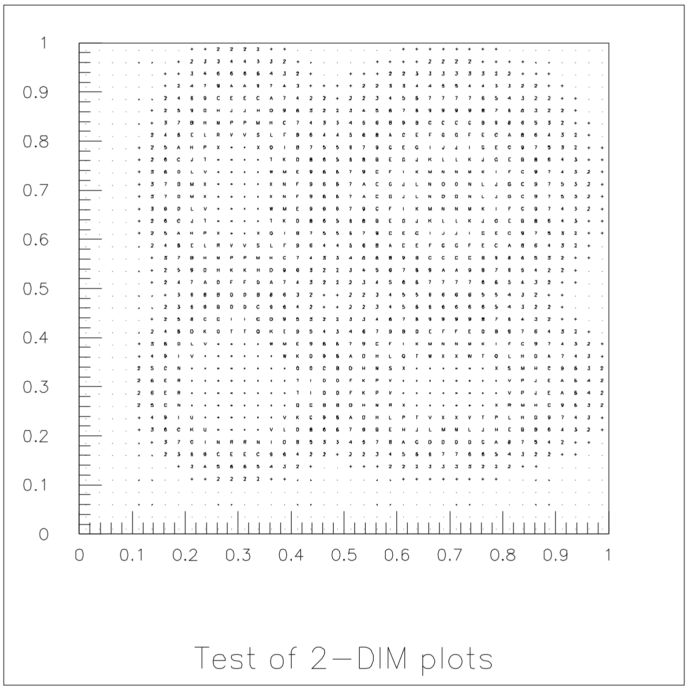


Figure 13.11: Example of HPLTAB with CHAR option

CHOPT = 'LEGO' **Lego (mode 0)**		
CHOPT = 'LEGO1' **Lego with colours (mode 1)**		
CHOPT = 'LEGO2' **Lego with colours (mode 2)**		
CHOPT = 'SURF' **Surface (mode 0)**		
CHOPT = 'SURF1' **Surface with colours (mode 1)**		
CHOPT = 'SURF2' **Surface with colours (mode 2)**		
CHOPT = 'SURF3' **Surface with contour plot on top (mode 3)**		
CHOPT = 'SURF4' **Surface with Gouraud shading (mode 4)**		
CHOPT = 'CYL' **Cylindrical for lego and surface**		
CHOPT = 'SPH' **Spherical for lego and surface**		
CHOPT = 'PSD' **Pseudo rapidity for lego and surface**		
PAR **index**	PAR **values**	**default**
1	Theta	30.
2	Phi	30.
3	ZMIN Lowest Z value	ZMIN
4	ZMAX Highest Z value	ZMAX
5	1000*IXMIN + IXMAX (Useful for ZOOM)	1-NX
6	1000*IYMIN + IYMAX (Useful for ZOOM)	1-NY

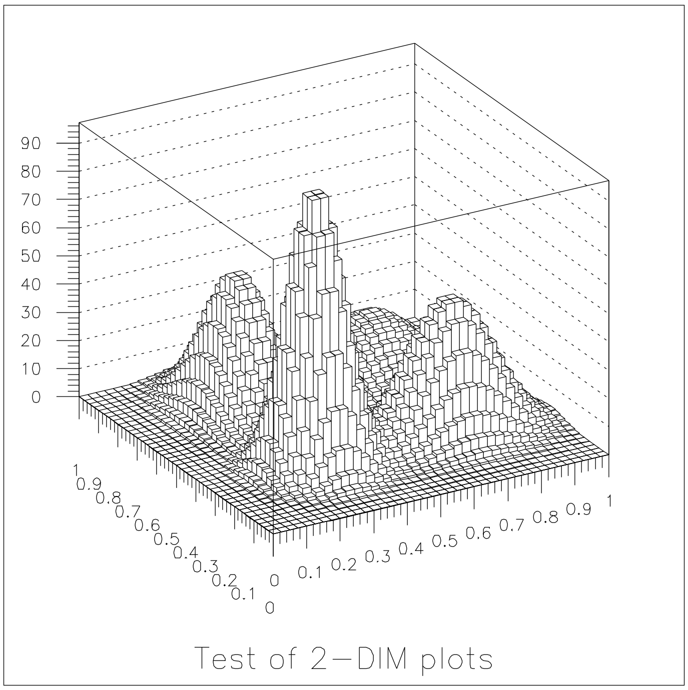


Figure 13.12: Example of `HPLTAB` with `LEGO` option

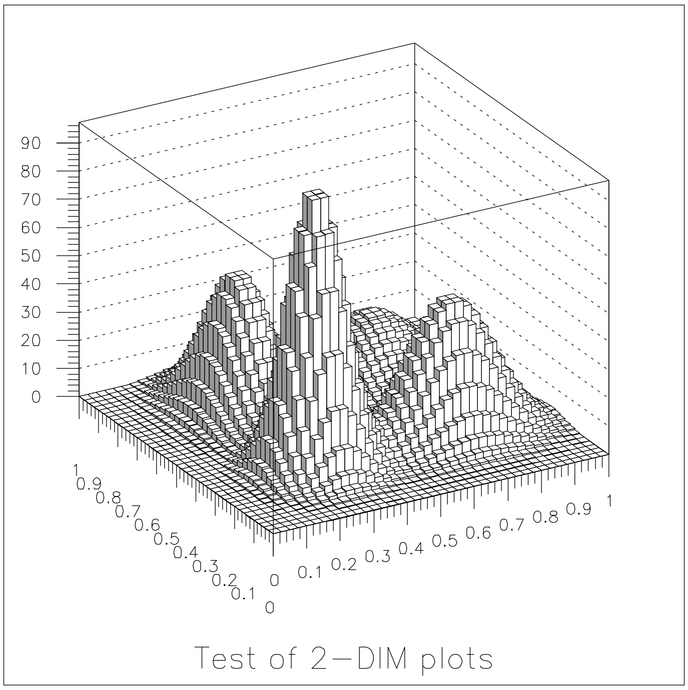


Figure 13.13: Example of `HPLTAB` with `LEGO1` option

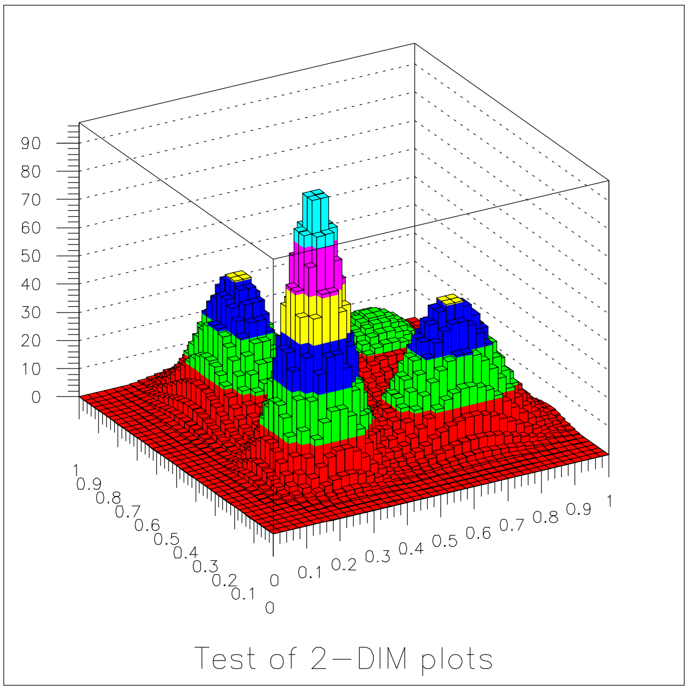


Figure 13.14: Example of `HPLTAB` with `LEGO2` option

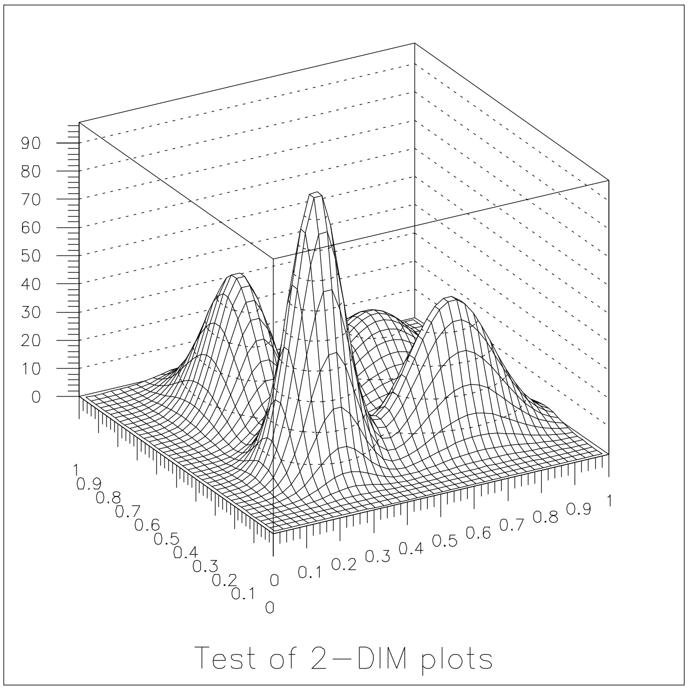


Figure 13.15: Example of `HPLTAB` with `SURF` option

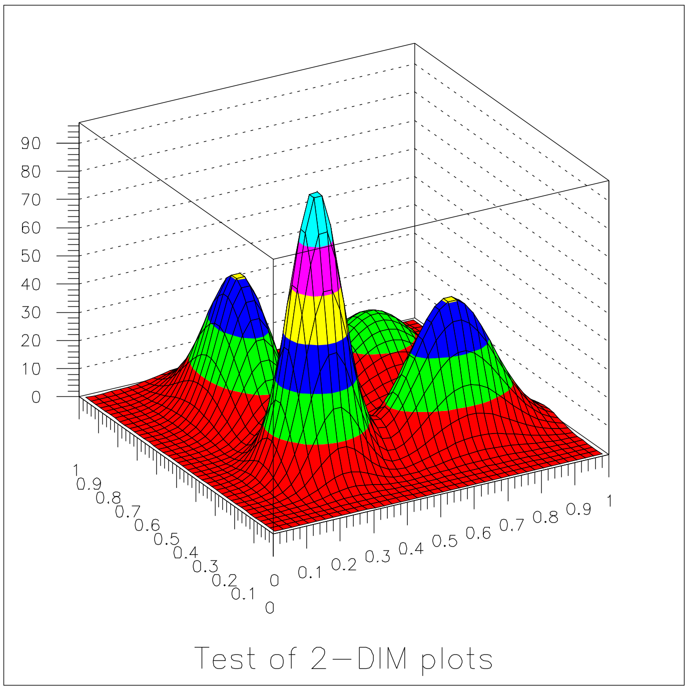


Figure 13.16: Example of `HPLTAB` with `SURF1` option

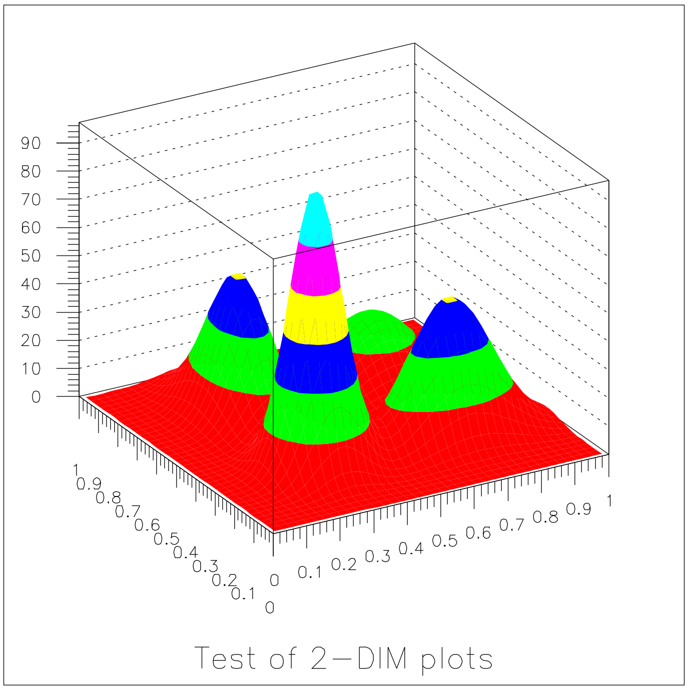


Figure 13.17: Example of `HPLTAB` with `SURF2` option

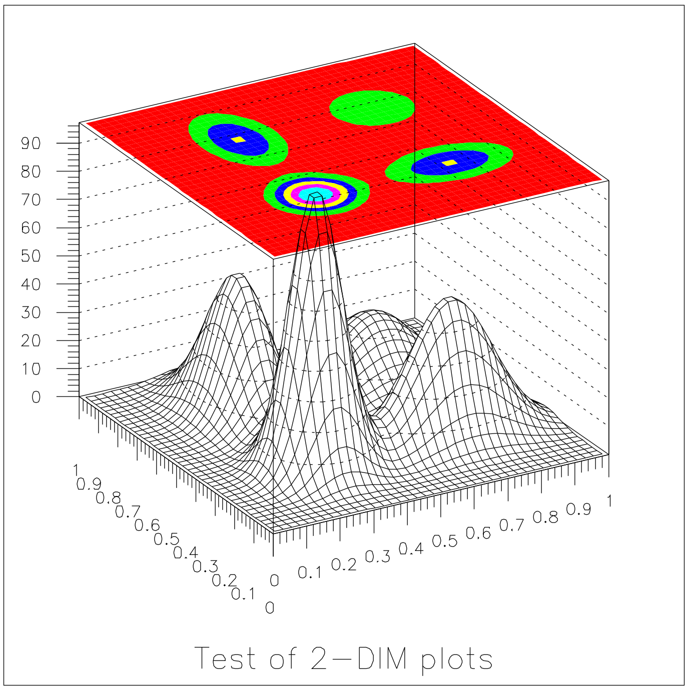


Figure 13.18: Example of `HPLTAB` with `SURF3` option

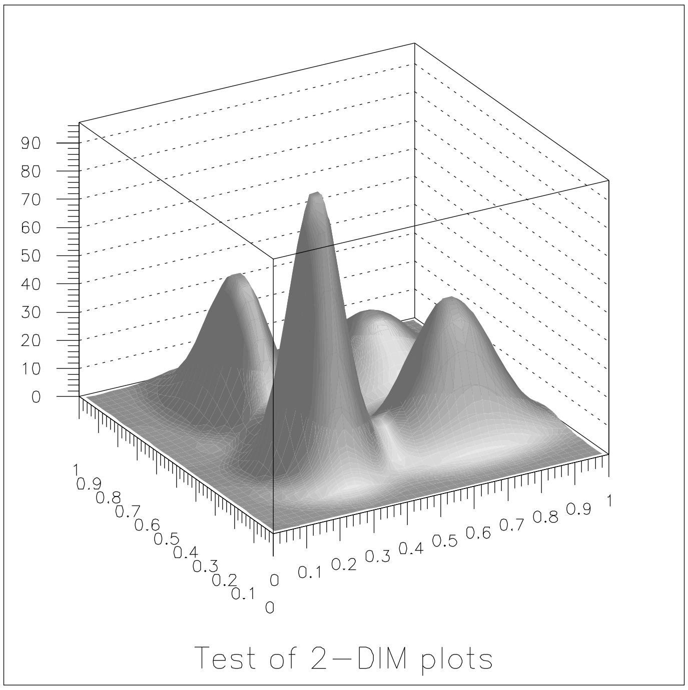


Figure 13.19: Example of `HPLTAB` with `SURF4` option

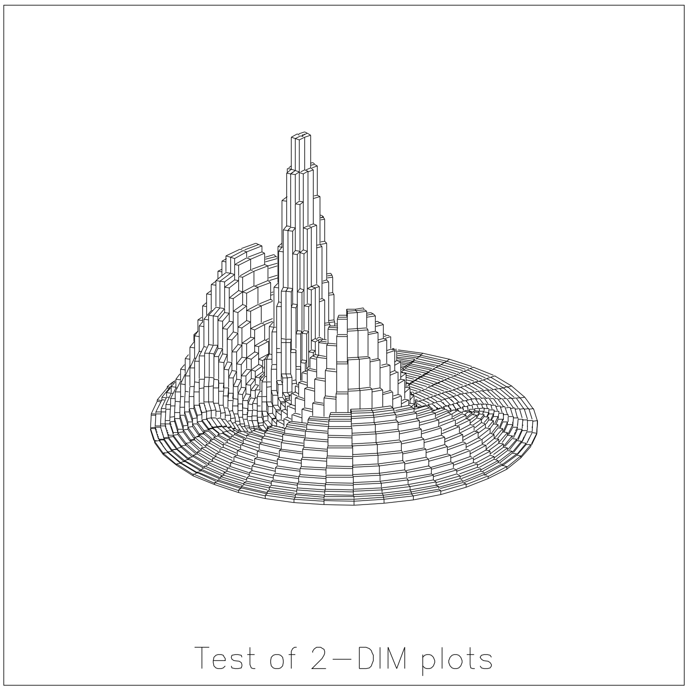


Figure 13.20: Example of `HPLTAB` with `LEGOPOL` option

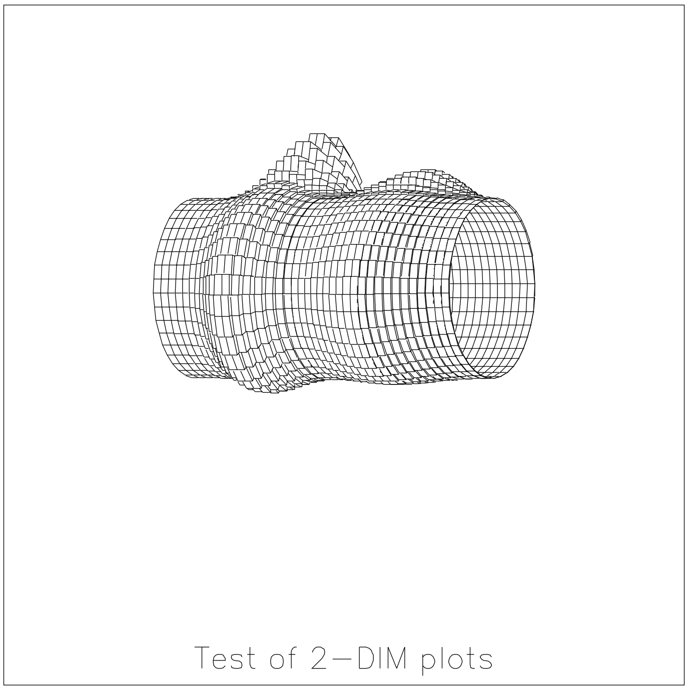


Figure 13.21: Example of `HPLTAB` with `LEGOCYL` option

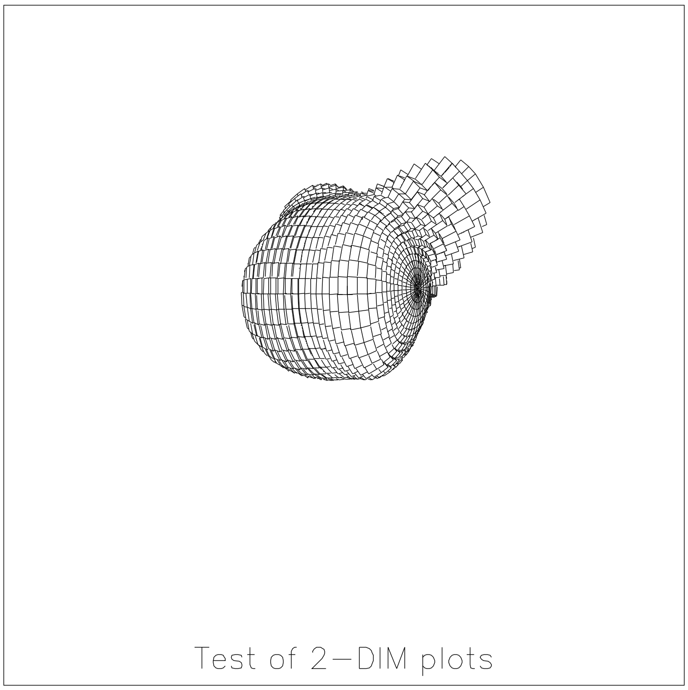


Figure 13.22: Example of HPLTAB with LEGOSPH option

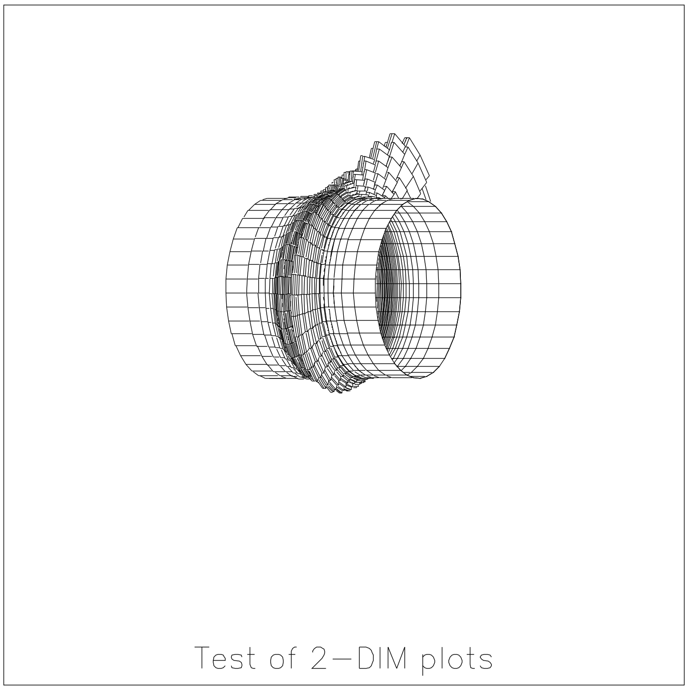


Figure 13.23: Example of HPLTAB with LEGOPSD option

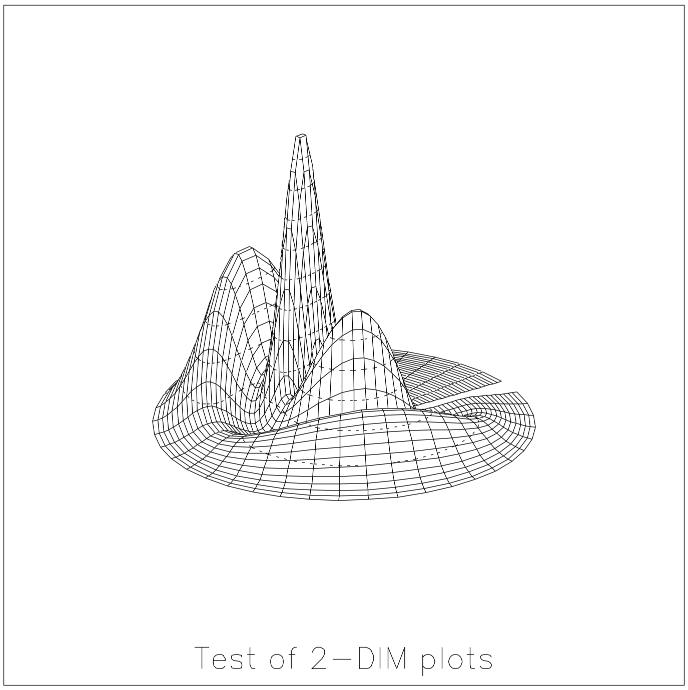


Figure 13.24: Example of `HPLTAB` with `SURFPOL` option

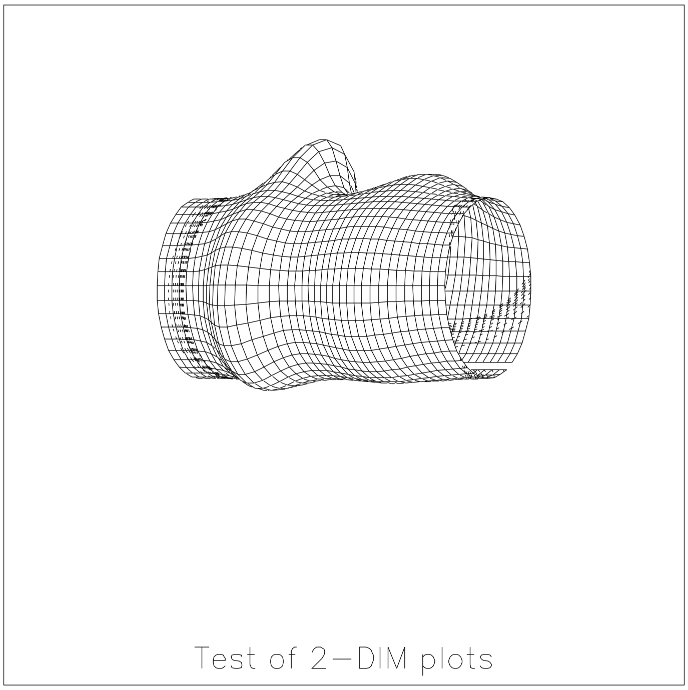


Figure 13.25: Example of `HPLTAB` with `SURFCYL` option

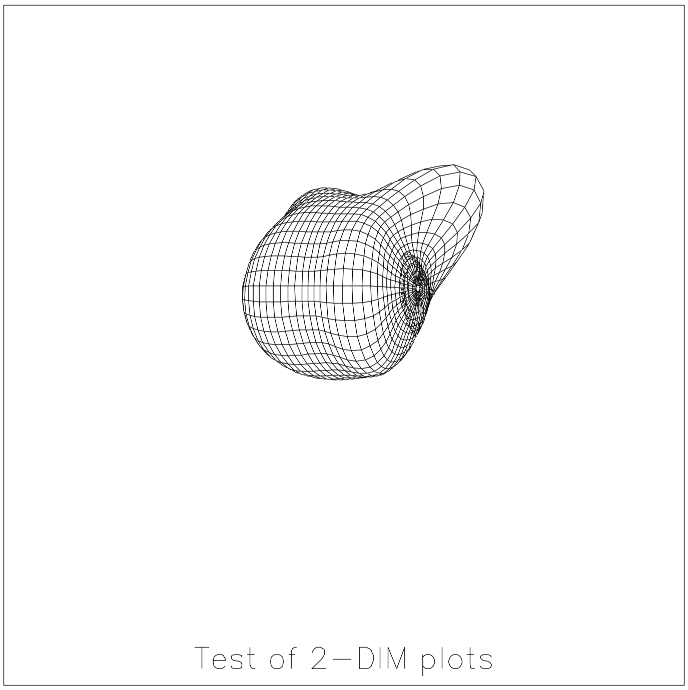


Figure 13.26: Example of `HPLTAB` with `SURFSPH` option

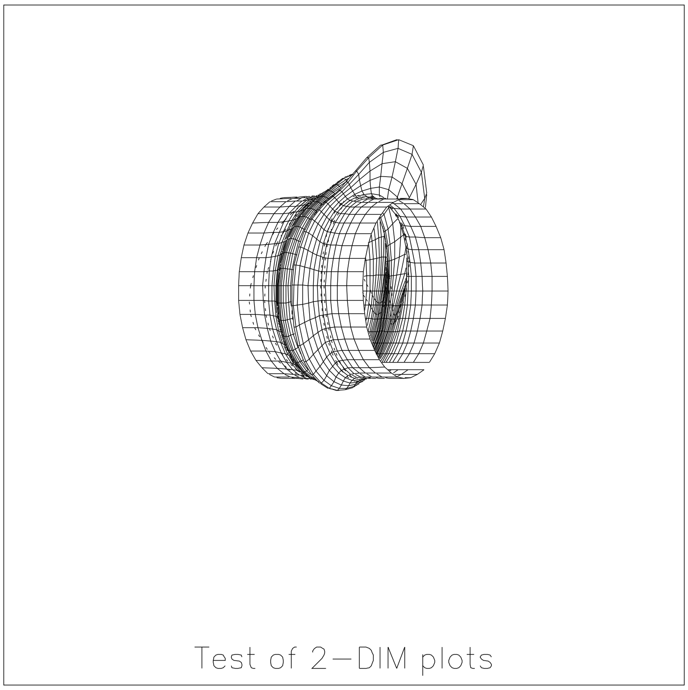


Figure 13.27: Example of `HPLTAB` with `SURFPSD` option

`CALL` **HPLTIT** `(CHTIT)`

Action: Writes a title for a histogram instead of the HBOOK title. The user must also turn off the option for printing the HBOOK title by setting the option 'UTIT'.

Parameter description:

CHTIT CHARACTER variable containing the title to be drawn (up to 80 characters).
If it contains a ";", two titles are drawn. One along the X-axis (the text before the ";"), and one along the Y-axis (the text after the ";").
' ' specifies that the HBOOK histogram title is to be used.

Remarks:

- HPLTIT must be called after HPLOT.
- Before calling HPLOT for the histogram to be titled, HPLOPT must be called with the option 'UTIT' otherwise the HBOOK histogram title will also be printed.
- The position of the title may be changed with HPLSET and its parameter 'YHTI'.

`CALL` **HPLUSR** `(ID, CHCASE, KID)`

Action: This is an HPLOT User Routine. The user should not call it, but provide his own subroutine HPLUSR, which will be called after each histogram has been plotted. To avoid problems with unresolved external references, a dummy routine HPLUSR is provided in the HPLOT library.

Parameter description:

ID Identifier of the histogram just plotted

CHCASE CHARACTER variable specifying the type of histogram which has just been plotted:

'1DIM'	1 dimensional histogram.
'2DIM'	2 dimensional histogram.
'TABL'	Table.
'3DIM'	2 dimensional histogram or table plotted with routine HPLSUR.
'SLIX'	Slice in X of a 2 dimensional histogram or table.
'SLIY'	Slice in Y of a 2 dimensional histogram or table.
'BANX'	Band in X of a 2 dimensional histogram or table.
'BANY'	Band in Y of a 2 dimensional histogram or table.
'PROX'	X projection of a 2 dimensional histogram or table.
'PROY'	Y projection of a 2 dimensional histogram or table.

KID Flag denoting how HPLUSR was invoked:
0 : invoked with call to HPLOT(0,,,).
1 : invoked with a specific histogram identifier ID.

Remarks:

– `HPLUSR` is particularly useful when used in conjunction with `HPLOT(0)` as it allows to assign to every histogram the same axis titles, etc.
– Another use is to provide a printout of all histograms plotted.
– Many HBOOK and HPLOT subroutines can be called from `HPLUSR`, but some could give problems and the following routines **can not** be called from within `HPLUSR`: `HPLOT`, `HPLINT`, `HPLEND`, `HPLPRO` and `HPLSUR`.
– The option routine `HPLOPT` can be called from `HPLUSR`, but the plot size should not be changed (i.e. do not call `HPLOPT` with arguments ’`HORI`’, ’`VERT`’, ’`A4`’, …).

Examples of the use of HPLUSR

A simple example

The user may require all histograms to have the same axis titles, but there be gaps in the numbering of the histogram identifiers `ID`, or one may not even know which identifiers are available. A `DO` loop involving calls to `HPLOT(ID)` and `HPLAX` is therefore difficult. `HPLUSR` can be used together with `HPLOT(0,' ',' ',0)`

Using `HPLUSR` to have identical axes titles

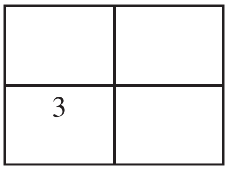
```
      SUBROUTINE HPLUSR(ID,CHCASE,KID)
      CALL HPLAX('Momentum (GeV/c)','Time of flight(nsec)')
      END
```

An example with zones

It may sometimes be required to perform a different action for different zones. As an example suppose one issues the following calls (with no call to `HPLZON` inside `HPLUSR`):

```
      CALL HPLZON(2,2,1,' ')
      CALL HPLOT(0,' ',' ',0)
```

Suppose also that for every histogram a comment must be written in the lower left hand corner (i.e. zone number 3 in our example):




```
      SUBROUTINE HPLUSR(ID,CHCASE,KID)
      CHARACTER*(*) CHCASE
      DATA IWIN /0/
      IWIN=IWIN+1
      K=MOD(IWIN,4)
      ⋮
      IF(K.EQ.3) CALL HPLCOM(......)
      END
```

If the comment has to appear in the histogram box, HPLGIV could be used to return the coordinates of the histogram box.

```
CALL HPLWIR (CHOPT,XVAL, YVAL, CHTICK)
```

Action: Draws "cross-wires" on a picture, optionally with tick marks and values. In the present context cross-wires are lines perpendicular to the X and/or Y axis.

Parameter description:

CHOPT CHARACTER variable specifying which cross-wires must be drawn and where to draw the values

' '	Tick marks are drawn on the edges of the picture.
'X'	Cross-wire drawn perpendicular to the X-axis.
'Y'	Cross-wire drawn perpendicular to the Y-axis.
'A'	Value drawn **A**bove cross-wire.
'B'	Value drawn **B**elow cross-wire.
'L'	Value drawn at **L**eft of cross-wire.
'R'	Value drawn at **R**ight of cross-wire.

XVAL Intersection on the X-axis.

YVAL Intersection on the Y-axis.

CHTICK CHARACTER variable specifying whether tick marks are required ('TICK').

Remarks:

- HPLWIR must be called after HPLOT.
- The values of XVAL and YVAL are always histogram coordinates.
- The tick marks will be drawn on both sides of the cross-wire, unless the cross-wires are requested on the boundary of the box surrounding the histogram (i.e. at the extreme limits of the drawn histogram). In this case tick marks will only be drawn inside the box.
- The character options 'A' (Above) and 'B' (Below) refer only to the cross-wires perpendicular to the Y axis, e.g.

```
CALL HPLWIR('YA',0.,3.14,'TICK')
CALL HPLWIR('Y' ,0.,3.14,' ')
```

  In each case only one cross-wire will be drawn.
- Similarly the character options 'L' (Left) and 'R' (Right) refer only to the cross-wires perpendicular to the X-axis.
- 'A', 'B', 'L' and 'R' have no effect unless CHTICK='TICK'
- It is possible to redefine the length of the tick marks on the X or Y axis by calling HPLSET with XTIC or YTIC
- The position of the axis values may be changed with HPLSET (XVAL or YVAL).
- The number of divisions and tick marks may be changed with HPLSET (NDVX or NDVY).

`CALL HPLZOM (ID, CHOPT, IMIN, IMAX)`

Action: Plots a 1 dimensional histogram between two channel numbers.

Parameter description:

ID	Identifier of a 1=dimensional histogram.
CHOPT	Options (as for routine HPLOT).
IMIN	First channel to be plotted. If IMIN≤0, then IMIN is assumed to be 1.
IMAX	Last channel to be plotted. If IMAX is greater than the number of channels, then IMAX is taken equal to the number of channels.

`CALL HPLZON (NXZON, NYZON, IFIRST, CHOPT)`

Action: Splits the picture into smaller parts, called zones. A complete histogram can be drawn in one of these zones.

Parameter description:

NXZON	Number of zones in the X direction.
NYZON	Number of zones in the Y direction
IFIRST	First zone to be plotted. A value of zero is equivalent to 1 and the first zone is selected.
CHOPT	CHARACTER variable specifying the options desired. 'S' Redefine zones on the same picture. ' ' The next call to HPLOT will start a new picture.

If both NXZON and NYZON are zero, then they are set to 1, if both NXZON and NYZON are reset to 1 and the zone option is turned off.

Remarks:

– Zones are numbered from left to right, starting at the top of the picture. For example with

```
CALL HPLZON(3,2,1,' ')
```

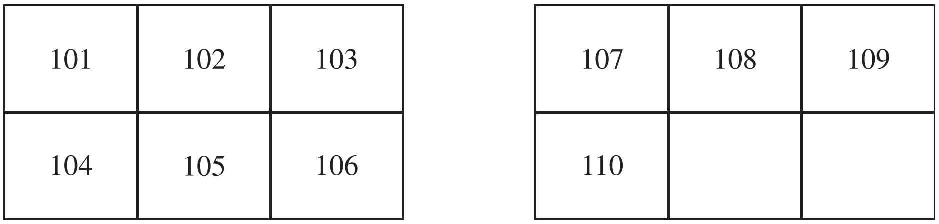
the zones are numbered as follows:

10	20	30
40	50	60

– The zone number is automatically incremented with each HPLOT call unless reset by a further call to HPLZON. If the zone number becomes larger than the maximum allowed on a picture, then the next histogram plotted will be at zone position 1 on a new picture. For example, assuming histograms 101 to 110 are 1 dimensional, then the following code:

```
      CALL HPLZON(3,2,1,'    ')
      DO 10 I=101,110
   10 CALL HPLOT(I,' ',' ',0)
```

gives:



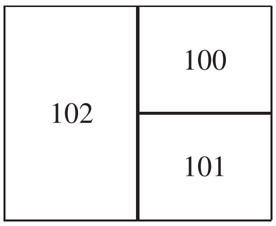
and a further call to HPLOT will start plotting below histogram 108.

- It is important to understand the difference between the effects of the 'S' options of HPLZON and HPLOT. The 'S' option of HPLOT allows histograms to be superimposed without redrawing axes or titles. The 'S' option of HPLZON allows the zone options to be reset on the current picture, and the next HPLOT call will plot a histogram complete with axes and titles. The 'S' option of HPLZON is normally used when plotting different sized zones on the same plot, or when forcing a histogram into a particular zone.
- Different sized zones can be plotted together on one picture with a series of HPLZON and HPLOT calls, all but the first containing the 'S' parameter in HPLZON.

  For example:

```
      CALL HPLZON(2,2,2,' ')
      CALL HPLOT(100,' ',' ',0)
      CALL HPLZON(2,2,4,'S')
      CALL HPLOT(101,' ',' ',0)
      CALL HPLZON(2,1,1,'S')
      CALL HPLOT(102,' ',' ',0)
```

  will give:



  This example also illustrates how one can force a histogram into a particular zone.
- To terminate the zone option:

```
      CALL HPLZON(1,1,1,' ')
```

  The next HPLOT call will start on a new picture.
- For scatter plots remember that:

```
      CALL HPLOT(ID,' ',' ',0)
```

  will give several pictures if slices/bands/projections are present. The above remarks must be read with this in mind.

Note that routine HPLZON must be called after HPLOPT if the options 'A3', 'A4', 'HORI' or 'VERT' are being requested and also after a call to HPLSET which defines the margin.

The distance between zones can be redefined using routine HPLSET and its options XWIN and YWIN.

Chapter 14: Technical Remarks

14.1 One-dimensional histograms

If HMAXIM, HMINIM and/or HCOMPA have **not** been called, a 1-dimensional histogram is scaled so that its maximum is at 90% of the available height. This maximum takes into account the HBOOK "functions" (if any) and error bars (if any). This can be changed with parameter HMAX in HPLSET (default value for HMAX is 0.9).

HPLOT always plots histograms from zero to the maximum (unless the minimum is negative). This differs from HBOOK which prints from the minimum to the maximum. This is not a serious problem, since the actual value of the contents is available with HBOOK, but HPLOT could produce a bin appearing to have zero contents when in fact it contains a very small value.

When the logarithmic scale in X is requested for a 1-dim histogram only the axe are drawn, not the contour.

14.2 HPLOT scatter plots

Two options are available for plotting scatter plots 'CHA' and 'NCHA'.

The first will print a character in the middle of each bin, corresponding to the contents of the bin. The result will be the same as with HBOOK - i.e. the contents are printed up to a value of 36 (or, up to the maximum allowed by the number of bits per channel that were set during booking), after which an asterisk is printed to denote overflow.

The second option 'NCHA' (set by default in HPLINT) will plot points randomly distributed within the bin. If the maximum content of any bin is 50 or less, the number of points plotted corresponds to the contents. If, however, the maximum content is greater than 50, then the number of points plotted will be normalised such that 50 points correspond to the maximum, (but a bin containing a value of 1.0 or greater will have at least one point plotted).

Note that logarithmic scales are ignored for scatterplots and tables.

14.3 Restrictions on the length of titles and text strings

To avoid text overflowing the limits of the picture, HPLOT will truncate text strings to fit the available space.

The truncation is performed by starting the text string as far to the left as possible (or, for Y axis titles, as low as possible). As many characters as possible are then drawn.

If the result is not what is required because of truncation the user can modify the output in several ways:

- The HBOOK global title can be redefined by calling HTITLE just before the relevant HPLOT call(s).
- The character sizes can be redefined with HPLSET.
- For "zoned" plots, the position or number of zones can be altered.
- The text position can be redefined with HPLSET.

14.4 Software characters

By default, HPLOT uses software characters. It is possible to switch between software and hardware characters by calling HPLOPT with the parameter 'SOFT' or 'HARD'. The advantages of using software characters are that they provide:

– Upper and lower case letters.
– Greek alphabet and special symbols.
– Superscripts and subscripts.
– Any size of letters at any angle.

The disadvantages are:

– Software characters take longer to plot.
– The size of the GKS metafile is much bigger.
– The necessary control characters make it tedious to mix Greek, Roman, upper case, lower case, etc.

14.5 Information about histograms

Four options (HPLOPT)are available to plot additional informations on HPLOT pictures: DATE, FILE, STAT and FIT.

```
* Plot date and hour on current HPLOT picture
      CALL HPLOPT('DATE',1)
* Plot file name of current histogram
      CALL HPLOPT('FILE',1)
* Plot statistics of current histogram
      CALL HPLOPT('STAT',1)
* Plot Fit parameters of current histogram
      CALL HPLOPT('FIT ',1)
```

For each of these option a corresponding HPLSET parameter is available:

```
      CALL HPLSET('DATE',r)
      CALL HPLSET('FILE',r)
```

where r defines the position of the date or file name:

r=1. Top left corner of page/current histogram (default for file).

r=2. Top right corner of page/current histogram (default for date).

r=3. Bottom left corner of page/current histogram.

r=4. Bottom right corner of page/current histogram.

For example the call:

```
      CALL HPLSET('DATE',3.)
```

sets the position of the date to the bottom left corner of the HPLOT pictures.

```
      CALL HPLSET('STAT',r)
```

where `r` corresponds to binary status bits `OURMEIA` as follows:

`O=1` Draw number of overflows
`U=1` Draw number of underflows
`R=1` Draw R.M.S.
`M=1` Draw mean value
`E=1` Draw number of entries
`I=1` Draw histogram identifier
`A=1` Draw the contents of all channels

For example the call:

```
CALL HPLSET('STAT',10.)
```

sets the statistics informations to be only the number of entries.

```
CALL HPLSET('FIT ',r)
```

where `r` corresponds to binary status bits `CEP` as follows:

`C=1` Draw χ^2
`E=1` Draw errors
`P=1` Draw fit parameters

For example to draw only the result of the χ^2 fit one would use:

```
CALL HPLSET('FIT ',100.)
```

For all these options, the **character size** is specified with the `HPLSET` parameter `'CSIZ'` and the character font used with the parameter `'CFON'`.

14.6 Normalization transformations

To build a picture, HPLOT uses the following normalization transformations:

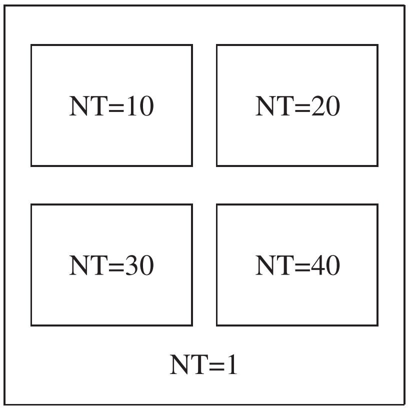
`NT=1` Defines a coordinate system in centimeters. It is used to define the picture size. normalization transformation 1 must be selected to draw text on the picture.

`NT=10,20,...` Used to draw pictures into zones. The coordinate system corresponds to histogram coordinates.

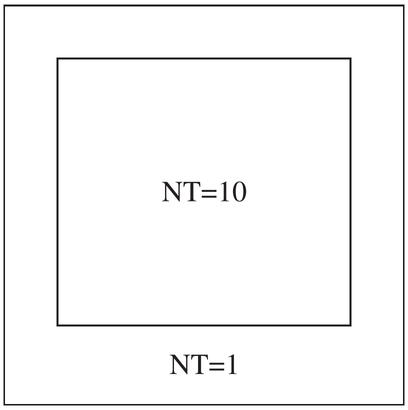
HIGZ routine `ISELNT` can be used to select one normalization transformation by the call:

```
CALL ISELNT(NT)
```

If `ZONE 2 2` is active, then:



If `ZONE 1 1` is active, then:



Chapter 15: Examples of HPLOT output

The examples are reproduced directly from the output of a PostScript metafile and introduced into the LaTeX file containing the HPLOT manual.

HPLOT test program

```
      PROGRAM HPLEXA
*
      CHARACTER*(*) HZFILE,HPFILE
+SELF,IF= IBM.
      PARAMETER (HZFILE='/HPLOT HIGZ')
+SELF,IF= IBM,IF=-PSCRIPT.
      PARAMETER (HPFILE='/HPLOT METAFILE')
+SELF,IF= IBM,IF= PSCRIPT.
      PARAMETER (HPFILE='/HPLOT PS')
+SELF,IF=-IBM.
      PARAMETER (HZFILE='hplot.higz')
+SELF,IF=-IBM,IF=-PSCRIPT.
      PARAMETER (HPFILE='hplot.metafile')
+SELF,IF=-IBM,IF= PSCRIPT.
      PARAMETER (HPFILE='hplot.ps')
+SELF.
      COMMON/PAWC/H(100000)
      LOGICAL INTRAC
*.___________________________________________
+SELF,IF=IBM,IF=X11.
      CALL INITC
+SELF,IF=APOLLO,UNIX,IBM,CRAY.
      OPEN(UNIT= 1,FILE=HZFILE,FORM='UNFORMATTED',RECL=4096,
     +     ACCESS='DIRECT',STATUS='UNKNOWN')
+SELF,IF=VAX
      OPEN(UNIT=1,FILE=HZFILE,FORM='UNFORMATTED',RECL=1024,
     +     ACCESS='DIRECT',SHARED,STATUS='UNKNOWN')
+SELF,IF=-VAX.
      OPEN(UNIT=10,FILE=HPFILE,FORM='FORMATTED',STATUS='UNKNOWN')
+SELF,IF= VAX.
      OPEN(UNIT=10,FILE=HPFILE,FORM='FORMATTED',SHARED,
     +     STATUS='UNKNOWN')
+SELF.
      IF(.NOT.INTRAC(DUMMY))THEN
         KWTYPE=0
      ELSE
         CALL IGWKTY(KWTYPE)
      ENDIF
      CALL TIMED(T0)
      CALL HLIMIT(100000)
      CALL HPLINT(KWTYPE)
      CALL HPLMAK
      IF(KWTYPE.NE.0)THEN
         CALL HPLOPT('PTO ',1)
         CALL HPLEX1
         CALL TIMED(T1)
         PRINT *, ' TIME FOR EXAMPLE 1 =',T1,'  SECONDS'
         CALL HPLEX2
         CALL TIMED(T2)
         PRINT *, ' TIME FOR EXAMPLE 2 =',T2,'  SECONDS'
```

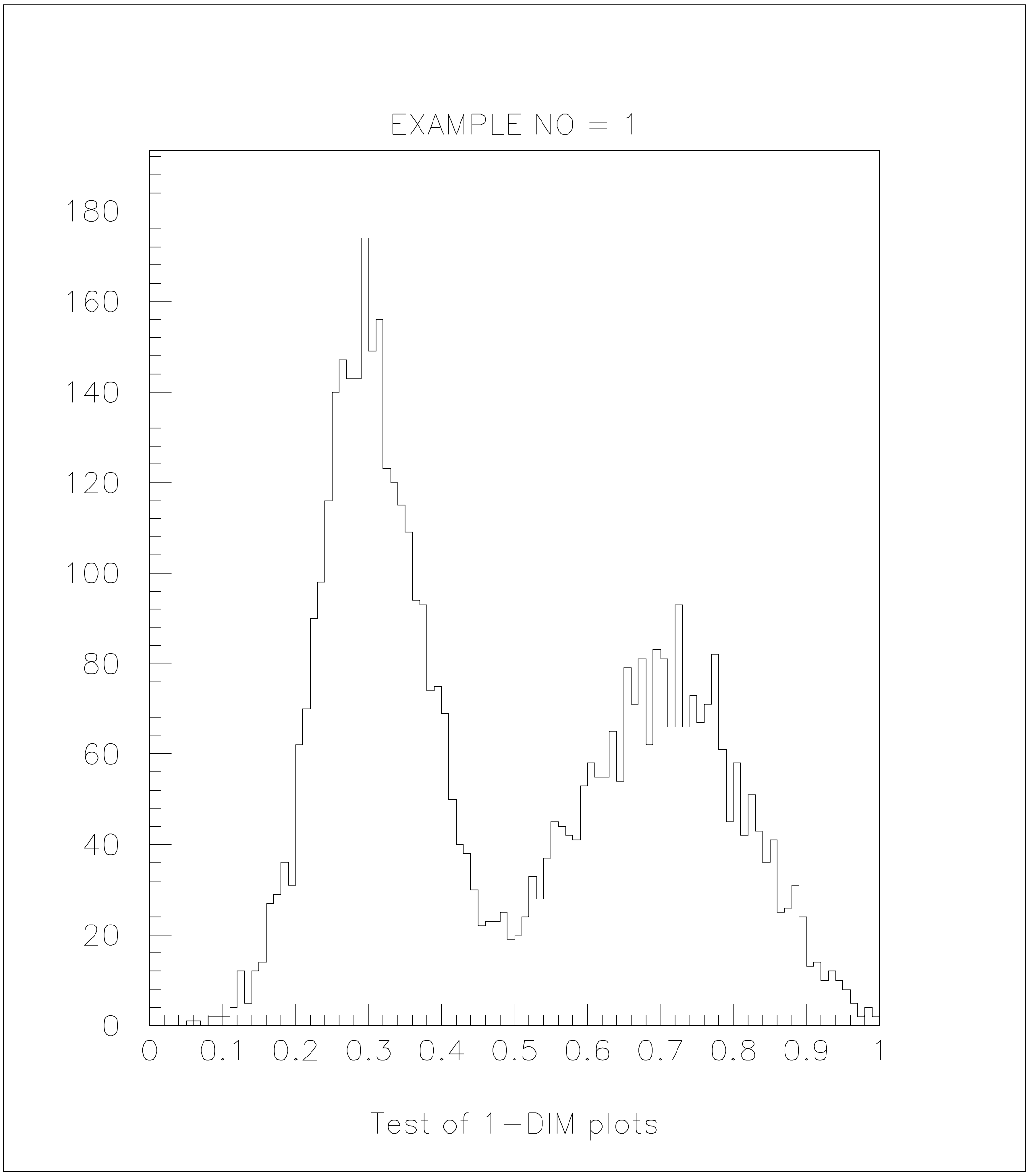
```
         CALL HPLEX3
         CALL TIMED(T3)
         PRINT *, ' TIME FOR EXAMPLE 3 =',T3,'  SECONDS'
         CALL HPLEX4
         CALL TIMED(T4)
         PRINT *, ' TIME FOR EXAMPLE 4 =',T4,'  SECONDS'
         CALL HPLEX5
         CALL TIMED(T5)
         PRINT *, ' TIME FOR EXAMPLE 5 =',T5,'  SECONDS'
      ENDIF
      CALL HPLOPT('NPTO',1)
*
*          Open HIGZ metafile
*          and repeat previous examples
*
      PRINT *,' WRITING HIGZ PICTURE FILE'
      CALL IGZSET('Z')
      CALL IZFILE(1,'HPLOT','NA')
      CALL HPLOPT('ZFL ',1)
      CALL HPLEX6
      CALL TIMED(T6)
      PRINT *, ' TIME TO WRITE HIGZ PICTURE FILE =',T6,'  SECONDS'
*
*          Open a GKS or PostScript metafile
*          and repeat previous examples
*
      PRINT *,' WRITING METAFILE (BE PATIENT !)'
      CALL IGZSET('G')
      CALL HPLOPT('NZFL',1)
      CALL HPLCAP(-10)
      CALL HPLEX6
      CALL TIMED(T7)
      PRINT *, ' TIME TO WRITE METAFILE =',T7,'  SECONDS'
*
*          Replay some pictures from the HIGZ picture file
*
      IF(KWTYPE.NE.0)THEN
         CALL HPLCAP(0)
         CALL HPLEX7
      ENDIF
*
      CALL HPLEND
      END
```

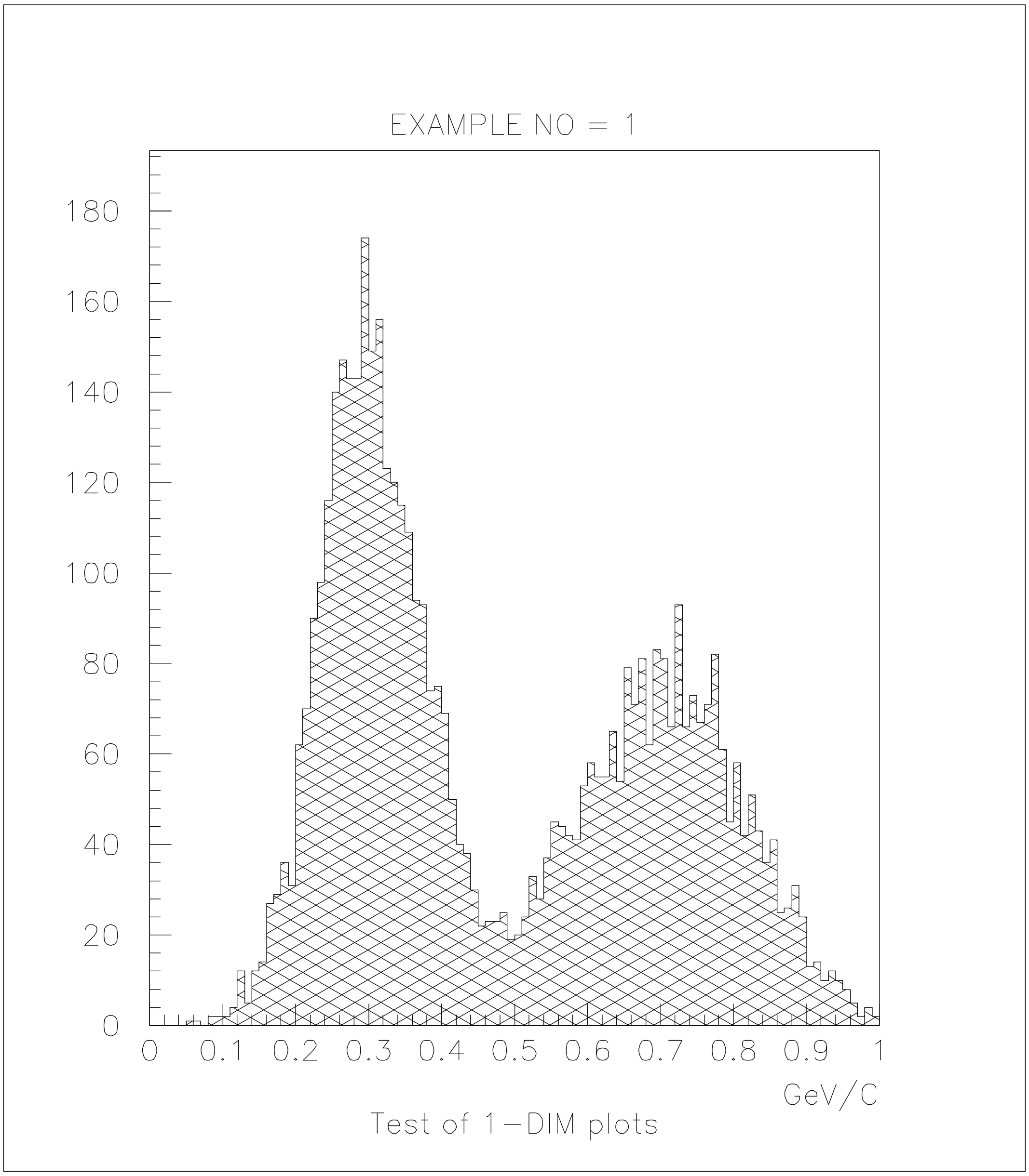
Creation of some histograms (based on HBOOK examples)

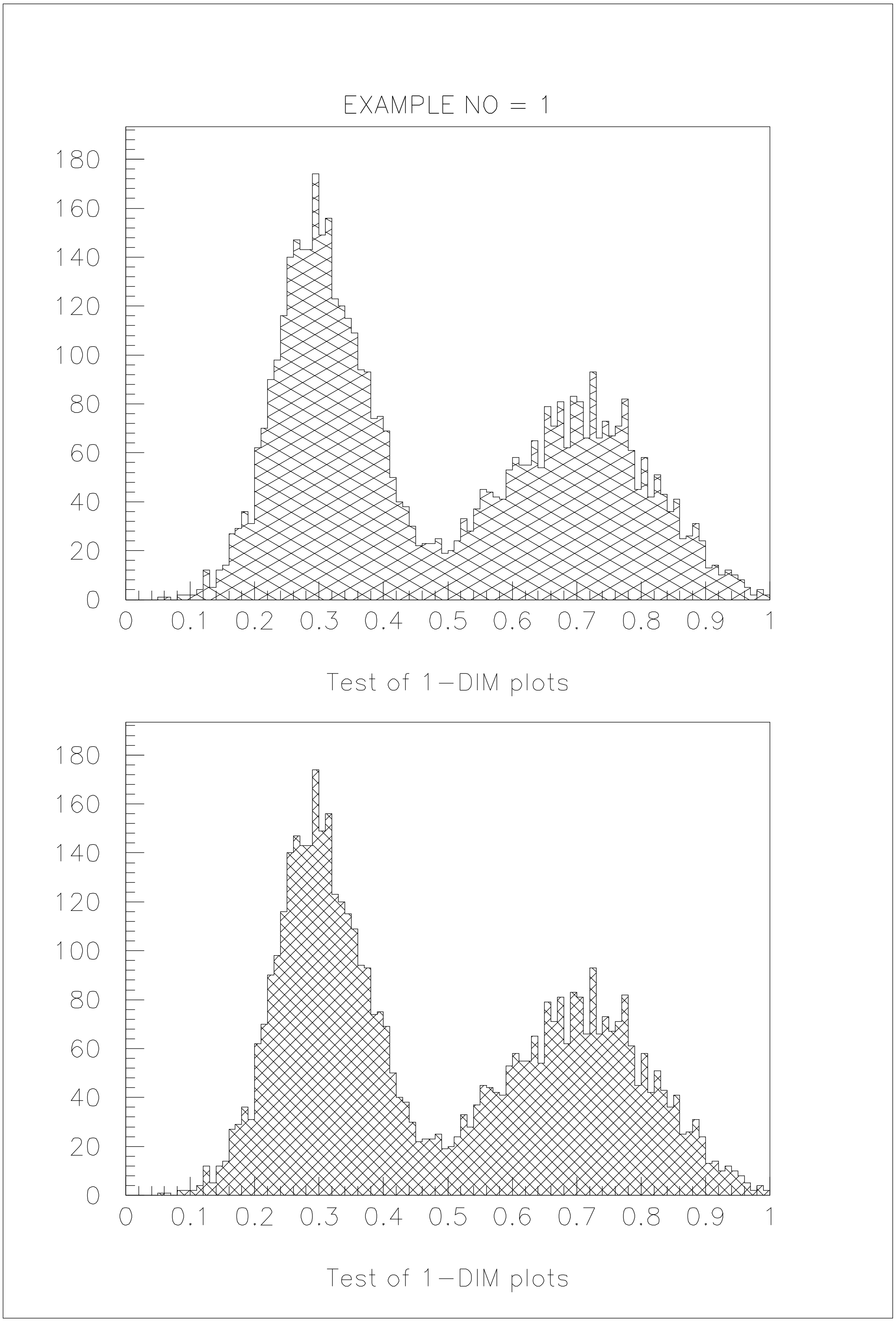
```
      SUBROUTINE HPLMAK
*
      COMMON/HEX2/C1,C2,XM1,XM2,XS1,XS2
      EXTERNAL HTFUN1,HTFUN2
*.------------------------------------------
*
*             BOOKING
*
      C1=1.
      C2=0.5
      XM1=0.3
      XM2=0.7
      XS1=0.07
      XS2=0.12
*
      CALL HBFUN1(100,'TEST OF HRNDM1',100,0.,1.,HTFUN1)
*
      CALL HBOOK1(110,'Test of 1-DIM plots',100,0.,1.,1000.)
*
      CALL HBFUN2(200,'Test of 2-DIM plots',40,0.,1.,40,0.,1.,HTFUN2)
      CALL HSCALE(200,0.)
*
*             FILLING
*
      DO 10 I=1,5000
         X=HRNDM1(100,I)
         CALL HFILL(110,X,0.,1.)
  10  CONTINUE
*
      END
      FUNCTION HTFUN1(X)
      COMMON/HEX2/C1,C2,XM1,XM2,XS1,XS2
*
      A1=-0.5*((X-XM1)/XS1)**2
      A2=-0.5*((X-XM2)/XS2)**2
      X1=C1
      X2=C2
      IF(ABS(A1).GT.1.E-4)X1=C1*EXP(A1)
      IF(ABS(A2).GT.1.E-4)X2=C2*EXP(A2)
      HTFUN1=X1+X2
      END
      FUNCTION HTFUN2(X,Y)
      HTFUN2=HTFUN1(X)*HTFUN1(Y)
      END
```

Examples of basic HPLOT : 1-DIM histograms

```
      SUBROUTINE HPLEX1
*
      CALL HTITLE('EXAMPLE NO = 1')
*
      CALL HPLSIZ(14.,16.,' ')
      CALL HPLOT(110,' ',' ',0)
      CALL HPLSET('HTYP',333.)
      CALL HPLOT(110,' ',' ',0)
      CALL HPLAX('GeV/C',' ')
      CALL HPLSIZ(14.5,21.4,' ')
      CALL HPLZON(1,2,1,' ')
      CALL HPLOT(110,' ',' ',0)
      CALL HPLSET('HTYP',244.)
      CALL HPLOT(110,' ',' ',0)
      CALL HPLSET('HTYP',0.)
      CALL HPLZON(1,1,1,' ')
*
      END
```


EXAMPLE NO = 1
180
160
140
120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 1-DIM plots

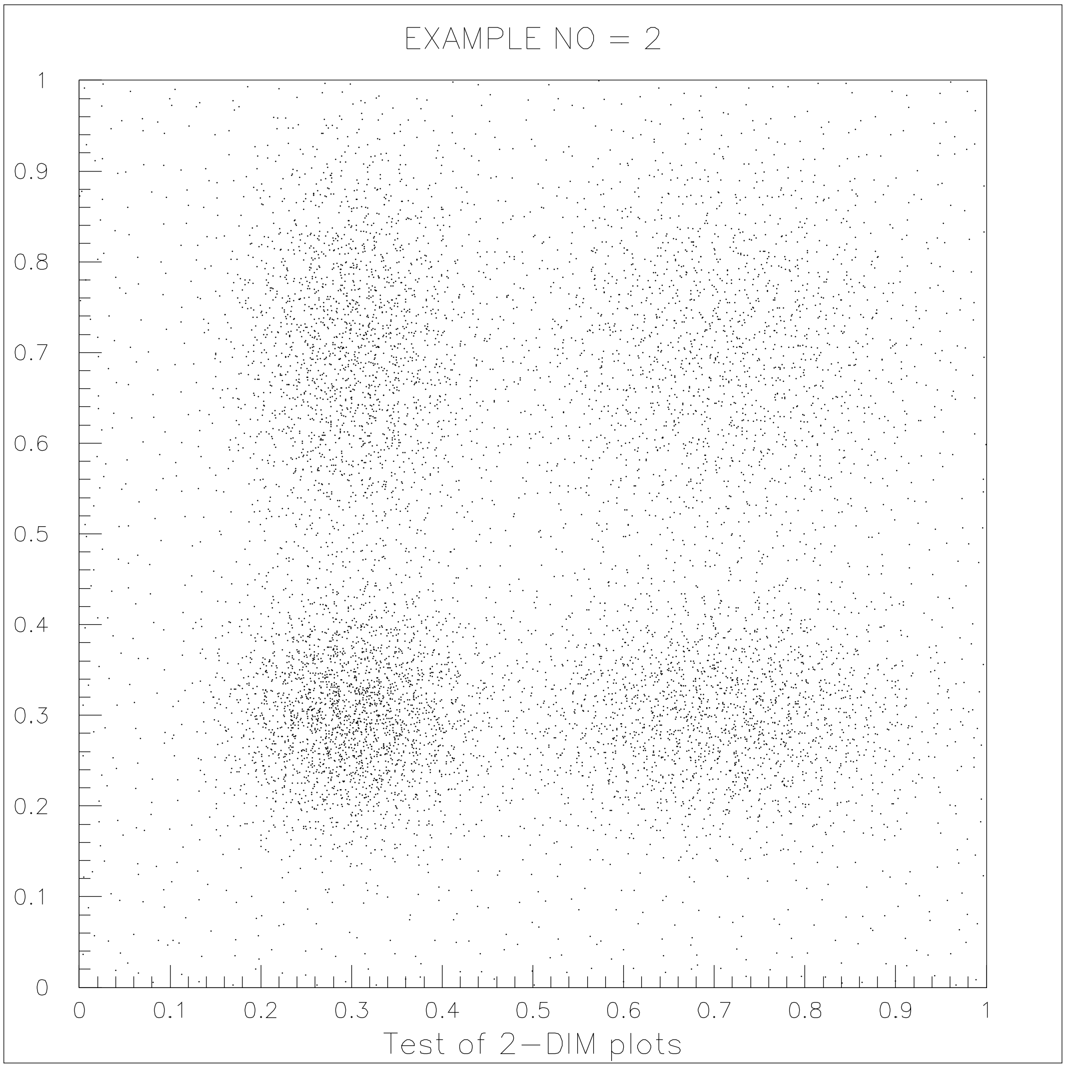

EXAMPLE NO = 1
180
160
140
120
100
80
60
40
20
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
GeV/C
Test of 1-DIM plots

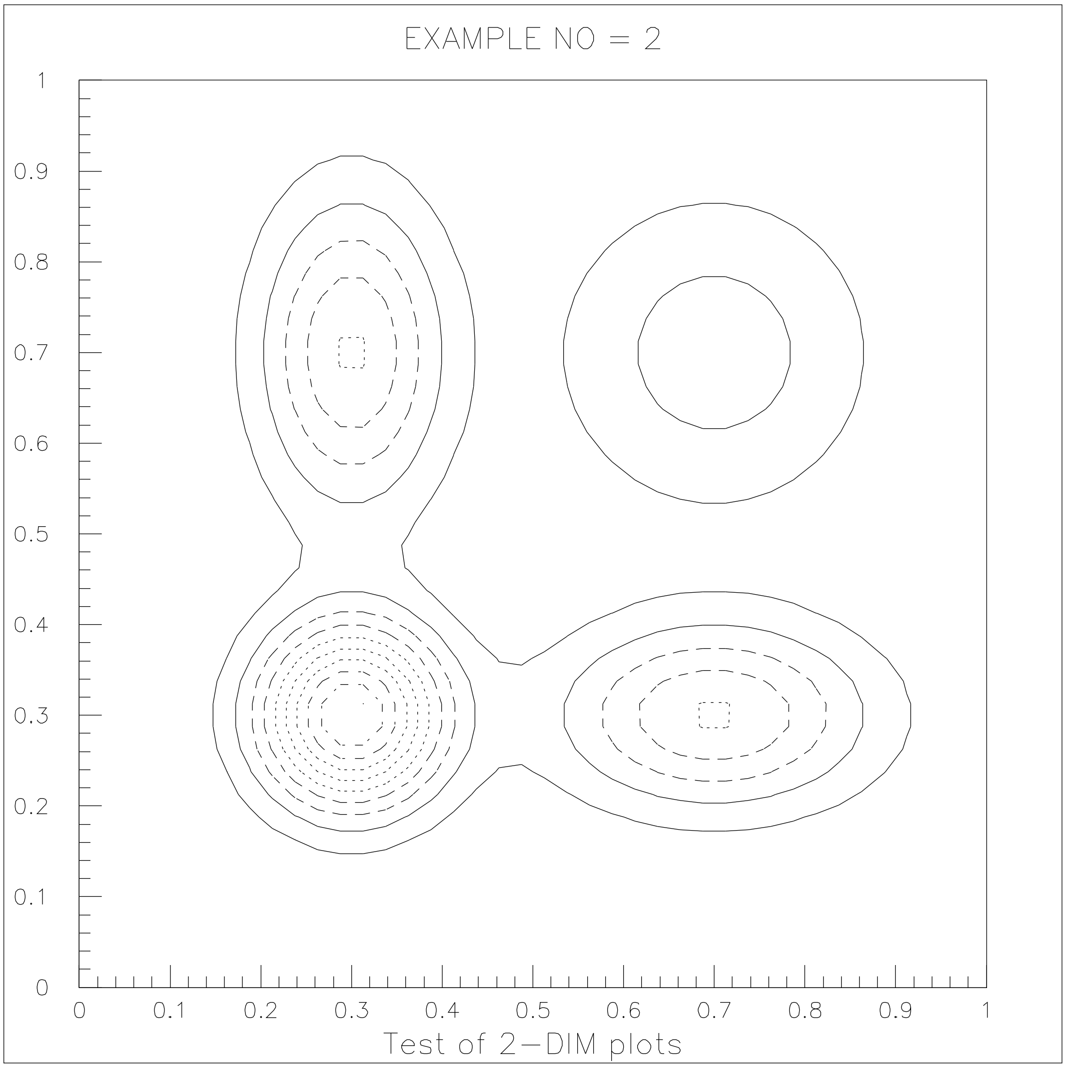


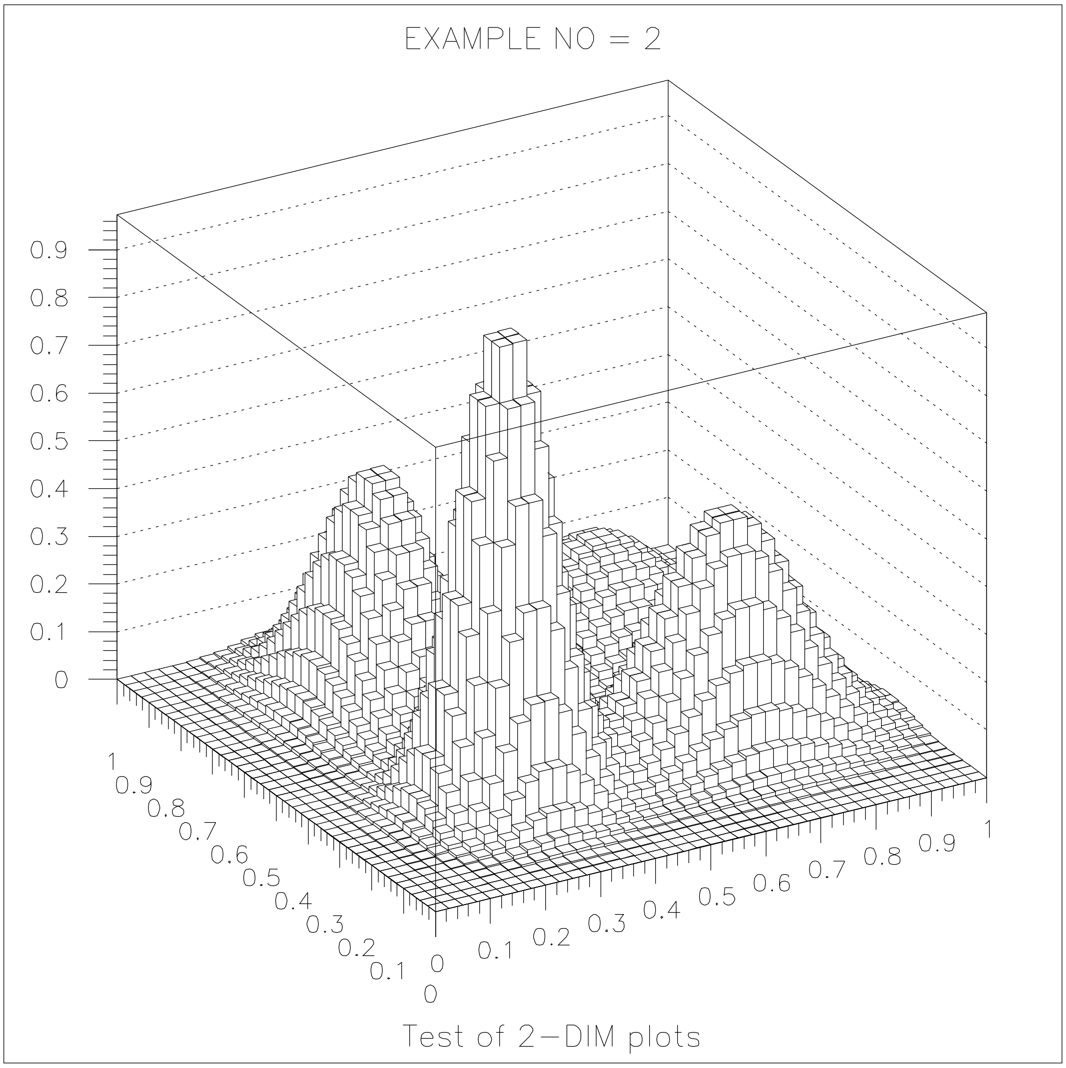
EXAMPLE NO = 1
180
160
140
120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 1-DIM plots
180
160
140
120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 1-DIM plots

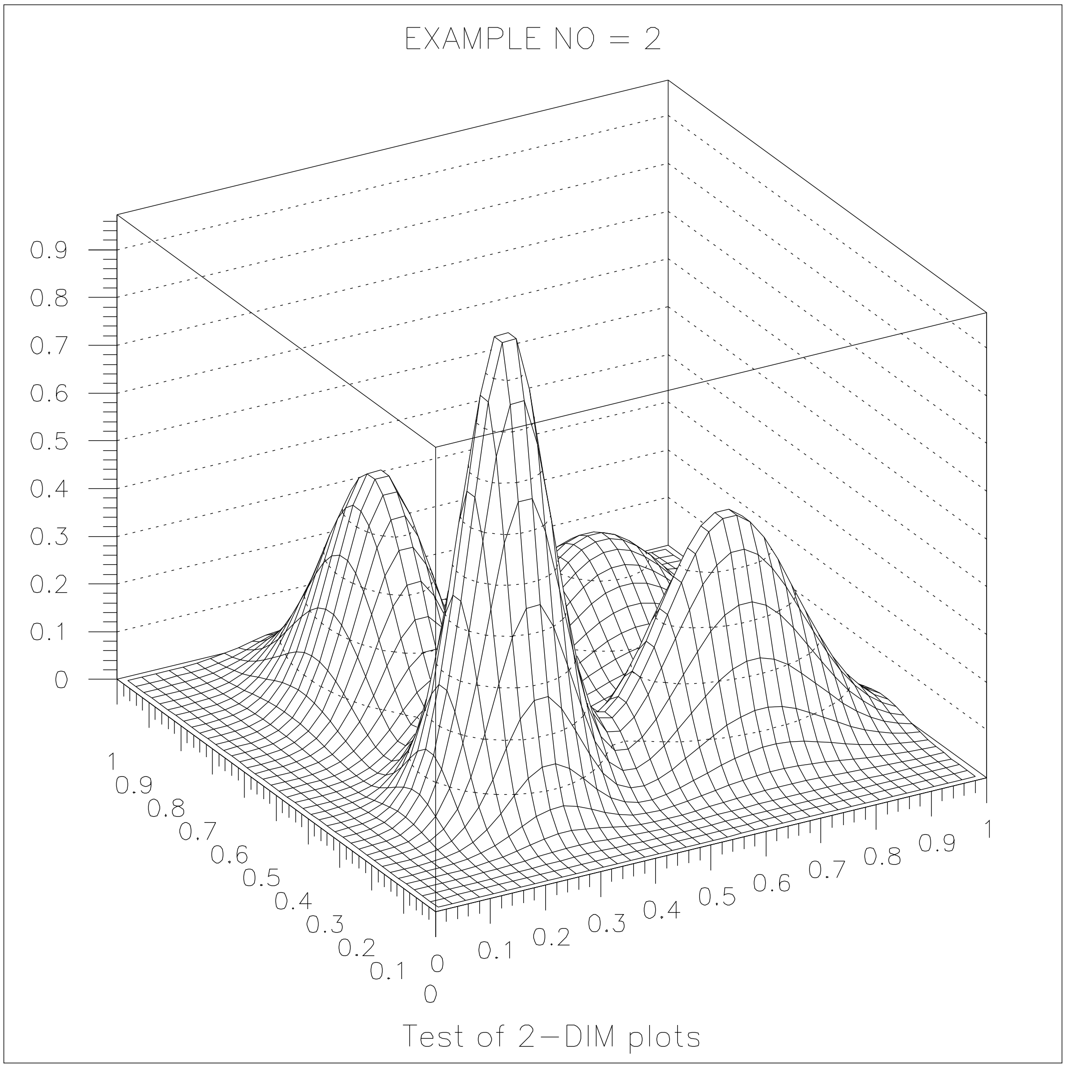
Examples of basic HPLOT : 2-DIM histograms

```
      SUBROUTINE HPLEX2
*
      CALL HTITLE('EXAMPLE NO = 2')
*
      CALL HPLSIZ(14.,14.,' ')
      CALL HPLSET('YGTI',0.3)
      CALL HPLSET('XMGL',1.)
      CALL HPLSET('YMGL',1.)
      CALL HPLSET('XMGR',1.)
      CALL HPLSET('YMGU',1.)
      CALL HPLSET('VSIZ',0.2)
      CALL HPLSET('YHTI',0.6)
      CALL IGSET('MTYP',1.)
      CALL HPLOT(200,' ',' ',0)
      CALL HPLCON(200,10,1)
      CALL HPLEGO(200,30.,30.)
      CALL HPLSUR(200,30.,30.,1)
*
      END
```


EXAMPLE NO = 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 2−DIM plots

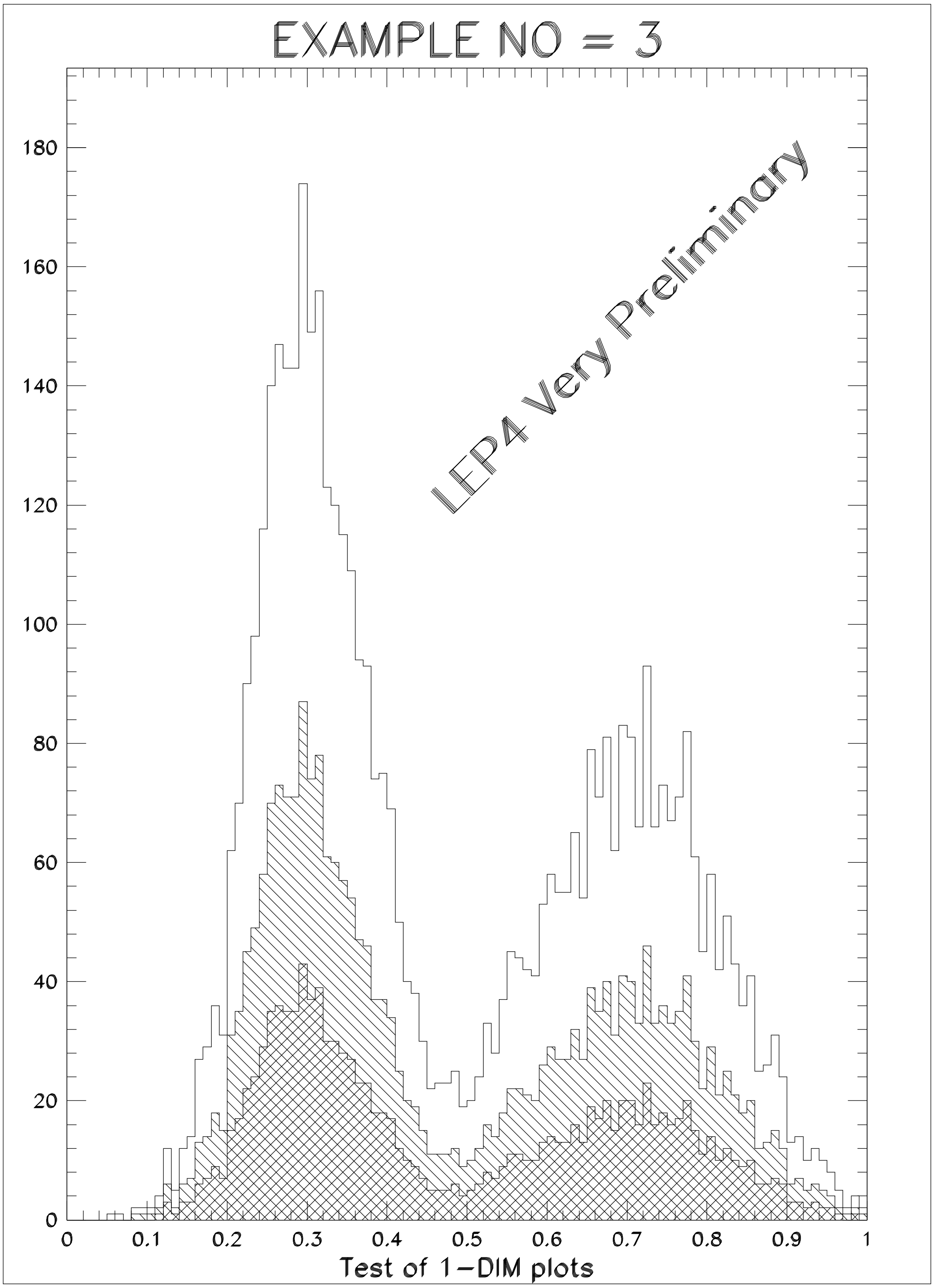

EXAMPLE NO = 2
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 2-DIM plots


EXAMPLE NO = 2
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 2-DIM plots


EXAMPLE NO = 2
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 2−DIM plots

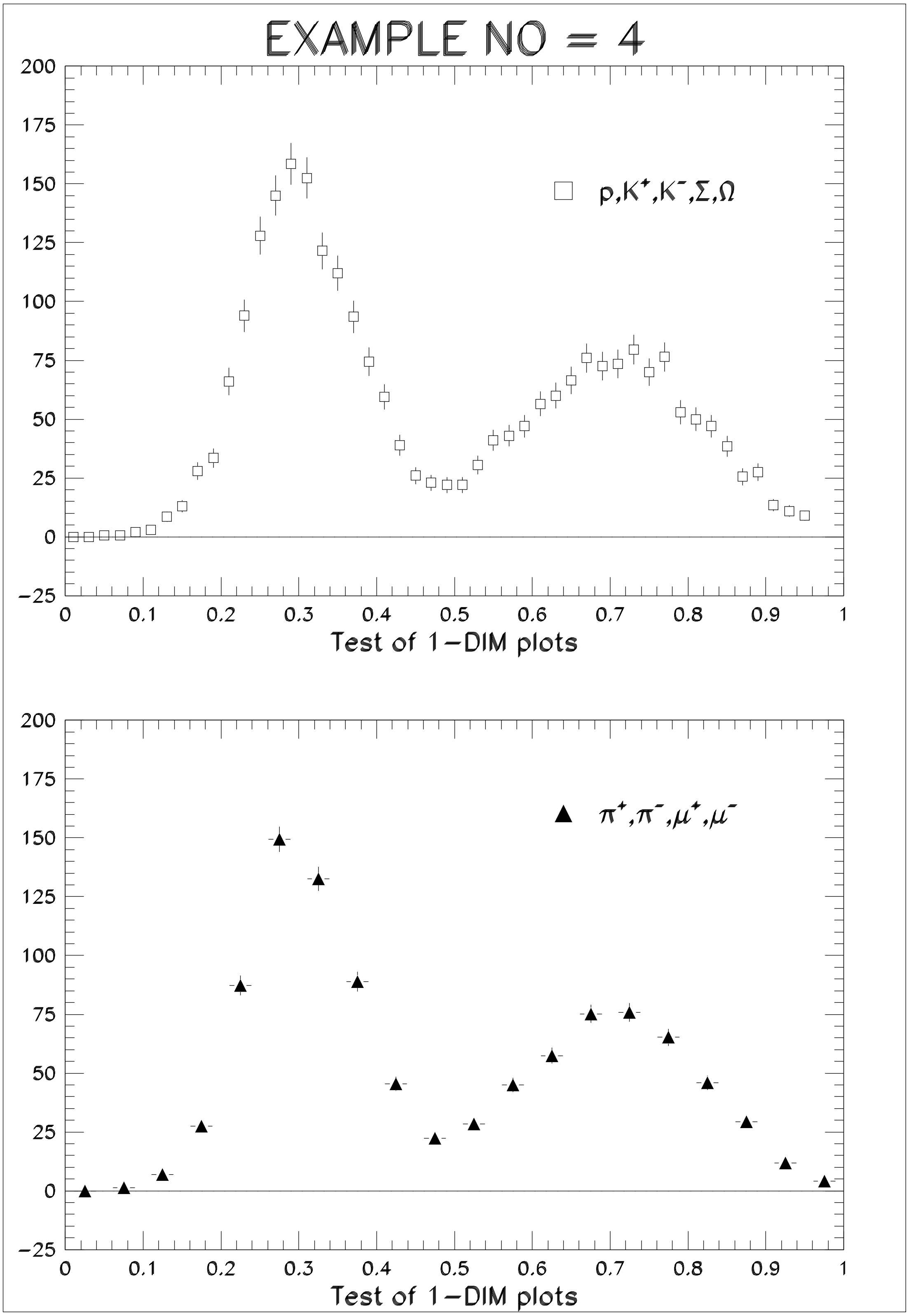
Examples of HPLOT options

```
      SUBROUTINE HPLEX3
*
      CALL HTITLE('EXAMPLE NO = 3')
*
      CALL HPLSIZ(14.5,20.,' ')
      CALL HPLSET('GSIZ',0.5)
      CALL HOPERA(110,'+',110,120,0.5,0.)
      CALL HOPERA(120,'+',120,130,0.5,0.)
      CALL HPLSET('PASS',5.)
      CALL HPLSET('CSHI',0.03)
      CALL HPLSET('XVAL',0.15)
      CALL HPLOPT('TIC ',1)
      CALL HPLOT(110,' ',' ',0)
      CALL HPLSET('HTYP',245.)
      CALL HPLOT(120,'S',' ',0)
      CALL HPLSET('HTYP',254.)
      CALL HPLOT(130,'S',' ',0)
      CALL HPLSOF(7.,12.,'LEP4 Very Preliminary',0.5,45.,99.,-1)
*
      END
```


EXAMPLE NO = 3
LEP4 Very Preliminary
0
20
40
60
80
100
120
140
160
180
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Test of 1-DIM plots

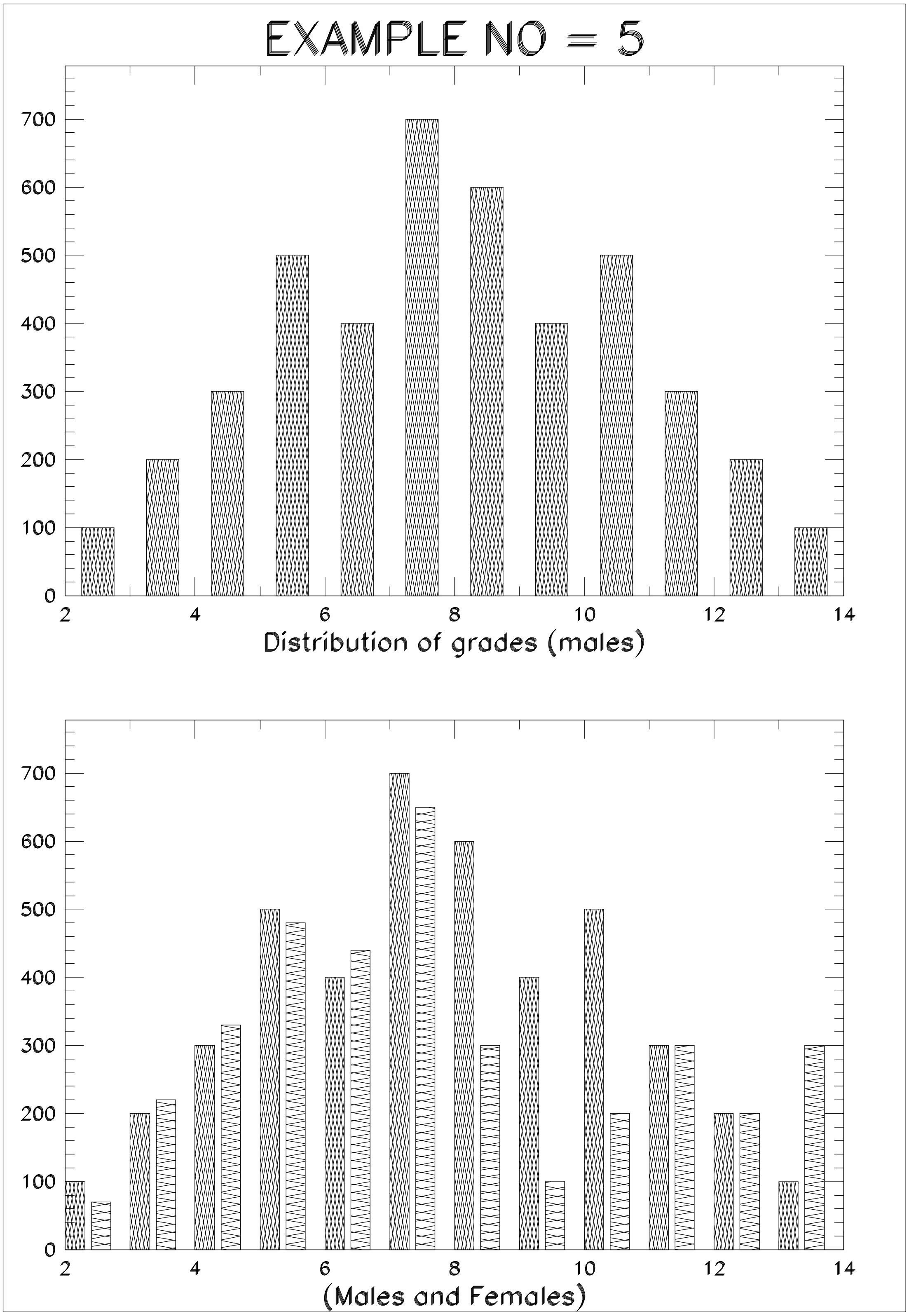
Examples of HPLOT options

```
      SUBROUTINE HPLEX4
*
      DIMENSION X(100),Y(100),EX(100),EY(100)
*
      CALL HTITLE('EXAMPLE NO = 4')
*
      CALL HCOPY(110,310,' ')
      CALL HRESET(310,' ')
      CALL HPLSET('XMGL',1.)
      CALL HPLSET('YMGL',1.)
      CALL HPLSET('XMGR',1.)
      CALL HPLSET('YMGU',1.)
      CALL HPLSET('VSIZ',0.2)
      CALL HPLSET('XVAL',0.15)
      CALL HPLSET('YGTI',0.3)
      CALL HPLSET('YHTI',0.6)
      CALL HPLSIZ(14.5,21.,' ')
      CALL HPLZON(1,2,1,' ')
      CALL HMAXIM(310,200.)
      CALL HMINIM(310,-25.)
      CALL HPLOT(310,' ',' ',0)
      CALL HREBIN(110,X,Y,EX,EY,50,1,100)
      CALL HPLERR(X,Y,EX,EY,48,' ',25,0.15)
      CALL HPLKEY(9.,18.,25,'p,K^+!,K^-!,[S,W')
*
      CALL HPLOT(310,' ',' ',0)
      CALL HREBIN(110,X,Y,EX,EY,20,1,100)
      CALL HPLERR(X,Y,EX,EY,20,' ',22,0.2)
      CALL HPLKEY(9.,8.,22,'[p^+!,p^-!,m^+!,m^-')
      CALL HDELET(120)
      CALL HDELET(130)
      CALL HDELET(310)
*
      END
```


EXAMPLE NO = 4
200
175
150
125
100
75
50
25
0
−25
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p,K⁺,K⁻,Σ,Ω
Test of 1−DIM plots
π⁺,π⁻,μ⁺,μ⁻
Test of 1−DIM plots

Examples of HPLOT options (BARS)

```
      SUBROUTINE HPLEX5
*
      DIMENSION XALL(12),XFEM(12)
      DATA XALL/
     +100.,200.,300.,500.,400.,700.,600.,400.,500.,300.,200.,100./
      DATA XFEM/
     + 70.,220.,330.,480.,440.,650.,300.,100.,200.,300.,200.,300./
*
      CALL HTITLE('EXAMPLE NO = 5')
*
      CALL HPLSET('YGTI',0.3)
      CALL HPLSIZ(14.5,21.,' ')
      CALL HPLZON(1,2,1,' ')
      CALL HBOOK1(1,'Distribution of grades (males)',12,2.,14.,0.)
      CALL HPAK(1,XALL)
      CALL HPLOPT('BAR ',1)
      CALL HPLSET('HTYP',188.)
      CALL HPLOT(1,' ',' ',0)
      CALL HRESET(1,'(Males and Females)')
      CALL HPAK(1,XALL)
      CALL HPLSET('BARO',0.)
      CALL HPLSET('BARW',0.3)
      CALL HPLOT(1,' ',' ',0)
      CALL HPLSET('HTYP',211.)
      CALL HPLSET('BARO',0.4)
      CALL HPAK(1,XFEM)
      CALL HPLOT(1,'SAME',' ',0)
      CALL HPLOPT('NBAR',1)
      CALL HDELET(1)
      CALL HPLSET('*',0.)
*
      END
```


EXAMPLE NO = 5
700
600
500
400
300
200
100
0
2
4
6
8
10
12
14
Distribution of grades (males)
700
600
500
400
300
200
100
0
2
4
6
8
10
12
14
(Males and Females)

Examples of HPLOT using GKS metafiles or HIGZ files

```
      SUBROUTINE HPLEX6
*
      CALL HPLEX1
      CALL HPLEX2
      CALL HPLEX3
      CALL HPLEX4
      CALL HPLEX5
      CALL HPLNUL
*
      END
```

Examples of HPLOT playing back HIGZ files

```
      SUBROUTINE HPLEX7
*
      CHARACTER*10 STR
      DATA ICYCLE/999/
*
      CALL RZLDIR(' ',' ')
      CALL IGSET('AURZ',0.)
      CALL IZIN('PICT1',ICYCLE)
      CALL IZPICT('PICT1','D')
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IZIN('PICT8',ICYCLE)
      CALL IZPICT('PICT8','D')
      CALL IRQST(1,1,ISTAT,NCH,STR)
      CALL IZIN('PICT9',ICYCLE)
      CALL IZPICT('PICT9','D')
      CALL IRQST(1,1,ISTAT,NCH,STR)
*
      END
```

Appendix A: The X Window System interface routines

The interface between HIGZ and the X Window System (X11) is done via a set of C routines callable from a Fortran program. This set of routine provide a low level interface to X11. It allows to write X11 programs with a small number of simple routines and it does not require the knowledge of the complete X11 library (Xlib). This set of interface routines are described in this appendix. The ”normal” HIGZ user does not need to read this chapter, which is useful only to write X11 drivers in Fortran.

A.1 X11 interface control routines

A.1.1 Open X11 display

```
I = IXOPNDS (LENHST,CHOST)
```

Action: Open the display. Returns -1 if the opening fails.

Parameter description:

`LENHST`	Host name length.
`CHOST`	Host name.

A.1.2 Open an X11 window

```
IWID = IXOPNWI (IX,IY,IW,IH,LENTIT,CHTIT,IFLAG)
```

Action: Open a X11 window and returns a window number which can be used with `IXSELWI`, `IXRSCWI`, `IXMOVWI`, `IXCLIP`, `IXNOCLI` etc ...
Returns `-1` if the window creation fails. It is possible to open up to 20 different windows.

Parameter description:

`IX`	Initial window X position in pixels.
`IY`	Initial window Y position in pixels.
`IW`	Initial window width in pixels.
`IH`	Initial window height in pixels.
`LENTIT`	Window title length.
`CHTIT`	Window title.
`IFLAG`	If it is not equal to 1 it allows to open a non Motif window even if `ixmotif` has been called.

The name of machine on which the program is running is automatically apend to the window title except if the window title begin with ”-”.

A.1.3 Select the current X11 window

`CALL` **IXSELWI** `(IWID)`

Action: Select the window to which subsequent output will be directed.

Parameter description:

`IWID` Window number returned by `IXOPNWI`.

A.1.4 Close an X11 window

`CALL` **IXCLSWI**

Action: Close the current window.

A.1.5 Close an X11 session

`CALL` **IXCLSDS**

Action: Close all opened windows and close the X11 connection.

A.1.6 Set X11 host name

`CALL` **IXSETHN** `(LENHST,CHOST)`

Action: Set host name.

Parameter description:

`LENHST` Host name length.

`CHOST` Host name.

A.1.7 Clear an X11 window

`CALL` **IXCLRWI**

Action: Clear the current window.

A.1.8 Update an X11 window

`CALL` **IXUPDWI** `(MODE)`

Action: Update the display and raise current window to top of stack. Synchronise client and server once (not permanent). Copy the pixmap on the window if the double buffer is on.

`MODE` Control the update mode.

`(1) or (11)`	The window is raised.
`(0) or (10)`	The window is not raised.
`(0) or (1)`	No synchronisation between client and server.
`(10) or (11)`	Synchronisation between client and server.

A.1.9 Resize an X11 window

```
CALL IXRSCWI (IWID,IW,IH)
```

Action: Resize (rescale) the X11 window IWID.

Parameter description:

`IWID` Window number returned by `IXOPNWI`.

`IW` New width.

`IH` New height.

A.1.10 Move an X11 window

```
CALL IXMOVWI (IWID,IX,IY)
```

Action: Move the X11 window IWID to the position (IX,IY).

Parameter description:

`IWID` Window number returned by `IXOPNWI`.

`IW` New X position.

`IH` New Y position.

A.1.11 Define the X11 clipping rectangle

```
CALL IXCLIP (IWID,IX,IY,IW,IH)
```

Action: Set clipping region for all windows.

Parameter description:

`IWID` Window number returned by `IXOPNWI`.

`IX` X clipping rectangle position.

`IY` Y clipping rectangle position.

`IW` Clipping rectangle width.

`IH` Clipping rectangle height.

A.1.12 Deactivate the X11 clipping rectangle

CALL **IXNOCLI** (IWID)

Action: Switch off the clipping rectangle.

IWID Window number returned by IXOPNWI.

Example

```
      program example
      character*8 machinename
      character*1 wait
*
      machinename='hphigz:0.0'              ! Open display
      if(ixopnds(10,machinename).ne.0)print*, 'Can''t open display'
*
      iwid=ixopnwi(0,0,200,200,5,'Hello')    !  Open window
*
      call ixselwi(iwid)                    ! Select window identified by iwinid
*
      call ixbox(10,190,10,190,0)           ! Draw boxes
      call ixbox(20,180,20,180,0)
      call ixbox(30,170,30,170,0)
*
      call ixupdwi(0)                       ! Flush X11 buffer and Wait
      read(*,'(a)') wait
*
      call ixclswi                          ! Close window
      call ixclsds                          ! Close display
*
      end
```

A.2 X11 output primitives

A.2.1 X11 lines

CALL **IXLINE** (N,IXY)

Action: Draw a line through all points.

Parameter description:

N Number of points

IXY(2,N) List of points. This is an INTEGER*2 array.

A.2.2 X11 markers

```
CALL IXMARKE (N,IXY)
```

Action: Draw a marker at each point.

Parameter description:

`N`	Number of points.
`IXY(2,N)`	List of points (INTEGER*2).

A.2.3 X11 fill area

```
CALL IXFLARE (N,IXY)
```

Action: Fill area described by polygon.

Parameter description:

`N`	Number of points.
`IXY(2,N)`	List of points (INTEGER*2).

A.2.4 X11 text

```
CALL IXTEXT (MODE,IX,IY,ANGLE,RMAGN,LENTXT,CHTEXT)
```

Action: Draw a text string using the current font.

Parameter description:

`MODE`	Drawing mode.
`MODE = 0`	The background is not drawn.
`MODE = 1`	The background is drawn.
`IX`	X text position.
`IY`	Y text position.
`ANGLE`	Text angle.
`RMAGN`	Magnification factor.
`LENTXT`	Text length.
`CHTEXT`	Text string.

A.3 X11 output attributes

A.3.1 X11 colour representation

```
CALL IXSETCO (INDEX,R,G,B)
```

Action: Set colour intensities for given colour index.

Parameter description:

`INDEX`	Colour index.
`R`	Red intensity between 0.0 and 1.0.
`G`	Green intensity between 0.0 and 1.0.
`B`	Blue intensity between 0.0 and 1.0.

A.3.2 X11 line width

```
CALL IXSETLN (IWID)
```

Action: Set line width.

Parameter description:

`IWID`	Line width in pixels.

A.3.3 X11 line style

```
CALL IXSETLS (N,IDASH)
```

Action: Set line style.

Parameter description:

`N` Length of dash list.

`IDASH(N)` Dash segment lengths.

- `N = 0` Use solid lines.
- `N > 0` Use dashed lines described by `DASH(N)`. For example `N=4, DASH=(6,3,1,3)` will produce a dashed-dotted line of 6 drawn pixels followed by 3 blank pixels, 1 drawn pixel and 3 blank pixels.

A.3.4 X11 lines colour

```
CALL IXSETLC (INDEX)
```

Action: Set colour index for lines.

Parameter description:

`INDEX`	Colour index defined by IXSETCOL.

A.3.5 X11 marker style

```
CALL IXSETMS (ITYPE,N,IXY)
```

Action: Set marker style.

Parameter description:

`ITYPE` Marker type

`N` Length of marker description.

`IXY(2,N)` List of points describing marker shape (INTEGER*2)

`N.EQ.0`	Marker is a single point.
`ITYPE = 0`	Marker is hollow circle of diameter `N`.
`ITYPE = 1`	Marker is filled circle of diameter `N`.
`ITYPE = 2`	Marker is a hollow polygon described by line `IXY`.
`ITYPE = 3`	Marker is a filled polygon described by line `IXY`.
`ITYPE = 4`	Marker is described by segmented line `IXY`.

Example: `ITYPE=4, N=4, XY=(-3,0,3,0,0,-3,0,3)` sets a plus shape of 7x7 pixels

A.3.6 X11 markers colour

```
CALL IXSETMC (INDEX)
```

Action: Set colour index for markers.

Parameter description:

`INDEX` Colour index defined by `IXSETCO`.

A.3.7 X11 fill area style

```
CALL IXSETFS (ISTYL,IFASI)
```

Action: Set fill area style.

Parameter description:

`ISTYL` fill area interior style hollow or solid.

`IFASI` fill area style index.

A.3.8 X11 fill area colour

```
CALL IXSETFC (INDEX)
```

Action: Set colour index for fill area.

Parameter description:

`INDEX` Colour index defined by `IXSETCO`.

A.3.9 X11 text alignment

```
CALL IXSETTA (IH,IV)
```

Action: Set text alignment.

Parameter description:

`IH` Horizontal alignment.

`IV` Vertical alignment.

A.3.10 X11 text fonts

```
CALL IXSETTF (MODE,LENFNT,CHFONT)
```

Action: Set text font to specified name. This function returns 0 if the specified font is found, 1 if not.

Parameter description:

`LENFNT` Font name length.

`CHFONT` Font name.

`MODE` Loading flag.

- `0` Search if the font exist.
- `1` Search the font and load it if it exist.

A.3.11 X11 text colour

```
CALL IXSETTC (INDEX)
```

Action: Set colour index for text.

Parameter description:

`INDEX` Colour index defined my IXSETCOL.

A.3.12 X11 text size

```
CALL IXTXTL (IW*,IH*,LENTXT,CHTEXT)
```

Action: Return the width and the height of a character string in the current font.

Parameter description:

`IW`	Text width.
`IH`	Text height.
`LENTXT`	Text length
`CHTEXT`	Message

A.3.13 X11 box

```
CALL IXBOX (IX1,IX2,IY1,IY2,MODE)
```

Action: Draw a box.

Parameter description:

`IX1`	X left down corner position.
`IY1`	Y left down corner position.
`IX2`	X right up corner position.
`IY2`	Y right up corner position.
`MODE`	drawing mode.

`MODE = 0`	Hollow.
`MODE = 1`	The Box is filled with the Fill area colour index.

A.3.14 X11 drawing mode

```
CALL IXDRMDE (MODE)
```

Action: Set the drawing mode

Parameter description:

`MODE`	Drawing mode.

`MODE = 1`	Copy.
`MODE = 2`	Xor.
`MODE = 3`	Invert.

Note that the Xor and Invert modes work only when the double buffer is off.

A.3.15 X11 synchronization

`CALL` **IXSYNC** `(MODE)`

Action: Set synchronization on or off. By default the X Window System bufferize all the graphics outputs (synchronization off). It is possible to switch off this capability with this routine (synchronization on) but the trafic on the network is more important and the speed of the graphics decrease by a factor 10 or 20 depending on the machine used.

Parameter description:

`MODE` synchronization ON or OFF.

- `1` ON.
- `0` OFF.

A.4 X11 input functions

A.4.1 X11 request locator

`CALL` **IXREQLO** `(MODE,ITYP,IX*,IY*)`

Action: Request locator input. Return button number (1=left, 2=middle, 3=right)

Parameter description:

`IX` Cursor position at the moment when the button is pressed.

`IY` Cursor position at the moment when the button is pressed.

`ITYP` Cursor type.

- `ITYP=1` Tracking cross.
- `ITYP=2` Cross-hair.
- `ITYP=3` Rubber circle.
- `ITYP=4` Rubber band.
- `ITYP=5` Rubber rectangle.

`MODE` Input mode.

- `MODE=0` Request.
- `MODE=1` Sample

A.4.2 X11 request string

`CALL` **IXREQST** `(IX,IY,LENTXT,*CHTEXT*)`

Action: Request a string input.The text is displayed and can be edited with Emacs-like keybinding return termination code (0 for ESC, 1 for RETURN)

Parameter description:

`IX,IY` Position where text is displayed.

`LENTXT` Text length.

`CHTEXT` Text displayed (input), edited text (output).

A.5 X11 inquiry routines

A.5.1 Get the window size

```
CALL IXGETGE (IWID,IX*,IY*,IW*,IH*)
```

Action: Returns position and size of Window `IWID`. If `IWID<0`, the size of the Display is returned in variables `IW` and `IH`.

Parameter description:

`IWID`	Window identifier.
`IX`	X Window position.
`IY`	Y Window position.
`IW`	Window or Display width.
`IH`	Window or Display height.

A.5.2 Get window identifier

```
CALL IXGETWI (IWID, IDG*)
```

Action: Returns the X11 window identifier (return by *XCreateWindow*) of the window identified IWID.

Parameter description:

`IWID`	Window number returned by `IXOPNWI`.
`IDG`	Window identifier.

A.5.3 Get the maximum number of planes

```
CALL IXGETPL (NBPLAN*)
```

Action: Returns the maximal number of planes of the display.

Parameter description:

`NBPLAN`	Number of planes.

A.6 Pixmap manipulation

A.6.1 Open a pixmap

```
I = IXOPNPX (IW, IH)
```

Action: Open a new pixmap, and return the pixmap adress.

Parameter description:

`IW`	Pixmap width.
`IH`	Pixmap height.

A.6.2 Close pixmap

```
CALL IXCLPX
```

Action: Close the current opened pixmap.

A.6.3 Copy pixmap

```
CALL IXCPPX  (IPIX, IX, IY)
```

Action: Copy the pixmap `IPIX` at the position (`IX`, `IY`) in the current window.

`IPIX` Pixmap adress.

`IX` X Pixmap position.

`IY` Y Pixmap position.

A.6.4 CLear pixmap

```
CALL IXCLRPX  (IPIX)
```

Action: Clear the pixmap IPIX.

`IPIX` Pixmap adress.

A.6.5 Remove pixmap

```
CALL IXRMPX  (IPIX)
```

Action: Remove the pixmap IPIX.

`IPIX` Pixmap adress.

A.6.6 Write pixmap on bitmap file

```
CALL IXWRPX  (IPIX,IW,IH,ILEN,CHPX)
```

Action: Write the pixmap IPX in the bitmap file CHPX.

`IPIX` Pixmap adress.

`IW` Pixmap width.

`IH` Pixmap height.

`ILEN` Pixmap name length.

`CHPX` Pixmap name.

A.6.7 Save a part of the screen in a pixmap

```
CALL IXWIPX (IPIX,IXPOS,IYPOS)
```

Action: Copy the area at the position IXPOS IYPOS in the current window in the pixmap IPIX. The area copied has the size of the pixmap IPIX.

`IPIX` Pixmap adress.

`IXPOS` X position in the current window.

`IYPOS` Y position in the current window.

A.6.8 Double buffer

```
CALL IXS2BUF (IWID, MODE)
```

Set the double buffer ON or OFF for the window IWID.

`IWID` Window identifier. 999 means all the opened windows.

`MODE` 1: double buffer is on, 0: double buffer is off.

A.7 HIGZ integration with Motif

```
CALL ixsdswi (dsp, win)
```

It is often useful to give to HIGZ (i.e. the `IX...` routines), the address of a pre-opened window in which the HIGZ output will be directed, in particular in the context of Motif applications written in C. This routine, callable from C, provides this facility.

Action: Allows to set the DISPLAY and the WINDOW address from outside HIGZ (e.g. a Motif program). The next call to `IXOPNWI` (after the call to `ixsdswi`) will use `dsp` and `win` to create the window.

Parameter description:

`dsp` Display.

`win` Window.

```
CALL ixmotif (dsp, motifopen, motifinit, motifclose)
```

Allows KUIP (or any other C package) to give to HIGZ the adress of three routines.

`dsp` Display.

`motifopen` to open a KUIP/Motif window

`motifinit` to initalize a KUIP/Motif window (add the callbacks)

`motifclose` to close a KUIP/Motif window

```
CALL IXMINIT (IWID)
```

Initialize the Motif windows (add the callbacks with `motifinit`).

`IWID` Window number returned by `IXOPNWI`.

Appendix B: HIGZ interface to graphic packages and calling sequences

B.1 Interfaces

HIGZ is presently interfaced to the following underlying graphics package:
- X11
- GL
- GPR
- PostScript
- FALCO terminals
- MSDOS graphics cards
- MacIntosh
- GKS-GRAL
- PLOT10-GKS
- MGKS
- NOVA-GKS
- DEC-GKS
- ATC-GKS
- SUN-GKS
- UNI-GKS
- DI3000 (interface developed at FNAL)
- GPHIGS (G5G PHIGS)
- GDDM

The underlying graphics package version can be selected at compilation time by PATCHY control statements.

B.2 Workstation types

B.2.1 BATCH Workstation Types

`0`	Alphanumeric terminal

B.2.2 HIGZ native Workstation Types

`1-10`	Describe in file higzwindows.dat (GPR, GL, X11, Mac/MPW)
`7878`	FALCO terminal
`7879`	xterm
`-111`	PostScript metafile (A4 Portrait)
`-112`	PostScript metafile (A4 Lansdcape)
`-3111`	PostScript metafile (A3 Portrait)
`-3112`	PostScript metafile (A3 Lansdcape)
`-99111`	PostScript metafile (A0 Portrait)
`-99112`	PostScript metafile (A0 Lansdcape)
`-100111`	PostScript metafile (Letter Portrait)

-100112	PostScript metafile (Letter Lansdcape)
-200111	PostScript metafile (Legal Portrait)
-200112	PostScript metafile (Legal Lansdcape)
-300111	PostScript metafile (Ledger Portrait)
-300112	PostScript metafile (Ledger Lansdcape)
-113	Encapsulated PostScript metafile
-777	LaTeX metafile

B.2.3 GKS-GRAL Workstation Types

4	Metafile Output
101	Tektronix 4010, 4014
102	Tektronix 4012
103	Tektronix 4014 with enhanced graphics option
121	Tektronix 4107, 4207, Pericom MX2000
122	Tektronix 4109
123	Tektronix 4111
125	Tektronix 4113
127	Tektronix 4115, Pericom MX8000
7800	MG600, MG200
7878	Falco, Pericom Graph Pac (old Pericom)
1020	VT240
1030	VT340
8601-6	Vaxstation GPX
10002	Apollo DNXXXX monochrome (GPR)
10003-4	Apollo DNXXXX colour (GPR)
9701-8	Apollo DNXXXX (GSR)
32120-9	X-Window

B.2.4 GKS-GRAL Workstation Types on IBM/NEWLIB

4714	IPS-Apollo-Workstation or X-Terminal
4725	Workstation/X-Terminal (2 Terminal Mode)
5003	IBM Graphic (GDDM) Terminal (e.g. 3192 G)
7878	Falco Infinity Terminal
470352	Atari-Workstation via 7171
471352	Atari-Workstation via 7171 (full window)
470353	Atari-Workstation via Terminal Server

471353	Atari-Workstation via Terminal Server(full window)
5005	Macintosh IBM Emulator
5010	IBM 3820 Laserprinter LI1 (portrait-format)
5020	IBM 3820 Laserprinter LI1 (landscape-format)
5011	IBM Matrix-Printer PRINTER2
13001	QMS Laserprinter L1 (portrait-Format)
13002	QMS Laserprinter L1 (landscape-format)

B.2.5 DEC-GKS Workstation Types

2	GKSM Output metafile
7	CGM Output metafile
13	VT240 color
14	VT240 monochrome
16	VT330
17	VT340 color
82	Tek 4107, 4207, Pericom MX2000
41	Vaxstation
211	Vaxstation running DECWindows
38	LN03 Plus Laser Printer
72	Tektronix 4014 Emulators
61	Postscript Printers

B.2.6 GKS2000 Workstation Types

445	Vaxstation
102	Tektronix 4014 Emulators
105	Macintosh Tektronix 4014 Emulator
191	Ramtek R25 4014 Emulator
324	LN03 Plus Laser Printer
601	Talaris Printers
700	Postscript Printers

B.2.7 SUN-GKS Workstation Types

3	GKS Metafile Output (ASCII)
4	SUN Workstation
6	HP-GL Output
7	PostScript Output
8	CGM Output
10	GKS Metafile Output (binary)

B.2.8 ATC-GKS Workstation Types

15nn	QMS Lasergrafix (TALARIS) 8.5x11
19nn	POSTSCRIPT 8.5x11 Printer
23nn	Tektronix 4105 Terminals
2400	Tektronix 4014 Terminal
2501	Tektronix 4010 Terminal
2502	C-ITOH Terminals (201,414)
2503	Retrographics VT640
2506	GRAPHON 140, 230 Terminal
25nn	other Tektronix 4010 Types
2600	DEC VT125 Terminal
2602	DEC VT240 Terminal
2603	DEC VT330 Terminal
2605	DEC VT340 Color Terminal
3100	Tektronix 4107 Terminal
3102	Tektronix 4205 Terminal
3104	Tektronix 4208 Terminal
315n	4107-4208 (software segments)
4300	VAXstation II (not in Version 3.2)
5300	X-Windows
5350	X-Windows with refresh
63nn	IMAGEN 8.5x11 Printer
66nn	Tektronix 4510 Color Rasterizer
10100	Binary Output CG Metafile
10110	Character Output CG Metafile
10120	Clear Text Output CG Metafile

B.2.9 MSDOS Workstation Types

4	320x200, 4 colors
5	320x200, 4 colors
6	640x200, 2 colors
13	320x200, 16 colors
14	640x200, 16 colors
15	640x350, 2 colors
16	640x350, 16 colors

`17`	640x480, 2 colors
`18`	640x480, 16 colors
`19`	320x200,256 colors
`SuperVGA`	800x600 16 colors
`41`	Tseng chipset
`98`	Video Seven
`88`	Paradise
`100`	Renaissance GRX

B.2.10 GDDM Workstation Types

`11`	3270 Family devices
`12`	5080 Display

B.2.11 GPHIGS Workstation Types

`8887`	GPHIGS-X11
`7176`	GPHIGS-GL
`8384`	GPHIGS-Starbase
`8871`	GPHIGS-Xgl
`7188`	GPEX

B.2.12 DI3000 Workstation Types

`-1`	Alphanumeric terminal
`0`	DI3000 metafile

Table B.1: Overview of HIGZ calling sequences

Calling Sequence	**Page**
HIGZ GKS like functions	
`CALL IACWK (KWKID)`	12
`CALL ICLKS`	10
`CALL ICLRWK (KWKID,KOFL)`	13
`CALL ICLWK (KWKID)`	12
`CALL IDAWK (KWKID)`	12
`CALL IFA (N,X,Y)`	26

Table B.1: Overview of HIGZ calling sequences (cont.)

Calling Sequence	Page
HIGZ functions	
CALL IGARC (XC,YC,R1,R2,PHIMIN,PHIMAX)	50
CALL IGAXIS (X0,X1,Y0,Y1,WMIN,WMAX,NDIV,CHOPT)	88
CALL IGBOX (X1,X2,Y1,Y2)	46
CALL IGCOLM (X1,X2,Y1,Y2,IC1,IC2,ZMIN,ZMAX,CHOPT)	120
CALL IGEND	6
CALL IGFBOX (X1,X2,Y1,Y2,X3,X4,Y3,Y4)	47
CALL IGHIST (N,X,Y,CHOPT)	54
CALL IGHTOR (CH,CL,CS,CR*,CB*,CG*)	122
CALL IGINIT (NWHIGZ)	6
CALL IGLBL (NLBL,CHLBL)	90
CALL IGLOC (ICURS,NT*,IBN*,XNDC*,YNDC*,XWC*,YWC*)	97
CALL IGLOC2 (KWKID,*NT*,X1*,Y1*,X2*,Y2*,ISTAT*,CHOPT)	96
CALL IGMESS (N,CHMESS,CHTIT,CHOPT)	120
CALL IGMETA (LUN,KWTYPE)	18
CALL IGPAVE (X1,X2,Y1,Y2,DZ,ISBOX,ISFRAM,CHOPT)	48
CALL IGPIE (X0,Y0,RADIUS,N,VALUES,CHOPT,IAO,IAS,IAC)	86
CALL IGQ (PNAME,*RVAL*)	104
CALL IGQWK (IWKID,PNAME,RVAL*)	106
CALL IGRAPH (N,X,Y,CHOPT)	51
CALL IGRNG (XSIZE,YSIZE)	17
CALL IGRTOH (CR,CB,CG,CH*,CL*,CS*)	122
CALL IGSA (KWKID)	8
CALL IGSET (CHNAME,VAL)	94
CALL IGSG (KWKID)	8
CALL IGSSE (IERRF,KWTYPE)	6
CALL IGTABL (NX,NY,V,NPAR,PAR,CHOPT)	58
CALL IGTERM	13
CALL IGTEXT (X,Y,CHARS,SIZE,ANGLE,CHOPT)	92
CALL IGWKTY (KWTYPE*)	11
CALL IGZSET (CHOPT)	107
HIGZ GKS like functions	
CALL IOPKS (IERRF)	10
CALL IOPWK (KWKID,KONID,KWTYPE)	10
CALL IPL (N,X,Y)	25
CALL IPM (N,X,Y)	26
CALL IRQLC (KWKID,LCDNR,ISTAT*,NT*,PX*,PY*)	96
CALL IRQST (KWKID,ISTDNR,ISTAT*,L*,STR*)	98
CALL ISCHH (CHH)	38
CALL ISCHUP (RCHUX,RCHUY)	38
CALL ISCLIP (ICLSW)	27
CALL ISCR (KWKID,ICI,CR,CG,CB)	27

Table B.1: Overview of HIGZ calling sequences (cont.)

Calling Sequence	Page
CALL ISELNT (NT)	17
CALL ISFACI (ICOLI)	29
CALL ISFAIS (INTS)	30
CALL ISFASI (ISTYLI)	31
CALL ISLN (LTYPE)	33
CALL ISLWSC (WIDTH)	33
CALL ISMK (MTYPE)	35
CALL ISMKSC (SSFM)	35
CALL ISPLCI (ICOLI)	29
CALL ISPMCI (ICOLI)	29
CALL ISTXAL (ITXALH,ITXALV)	37
CALL ISTXCI (ICOLI)	30
CALL ISTXFP (IFONT,IPREC)	38
CALL ISVP (NT,XMIN,XMAX,YMIN,YMAX)	16
CALL ISWKVP (KWKID,XMIN,XMAX,YMIN,YMAX)	15
CALL ISWKWN (KWKID,XMIN,XMAX,YMIN,YMAX)	15
CALL ISWN (NT,XMIN,XMAX,YMIN,YMAX)	16
CALL ITX (X,Y,CHARS)	26
CALL IUWK (KWKID,IRFLG)	12
HIGZ/IZ routines (pictures management)	
CALL IZCOPY (PNAME1,PNAME2,CHOPT)	108
CALL IZCTOI (CHVAL,IVAL*)	123
CALL IZCTOR (CHVAL,RVAL*)	123
CALL IZFILE (LUN,CHDIR,CHOPT)	118
CALL IZGED (PNAME,CHOPT)	110
CALL IZIN (PNAME,ICYCLE)	119
CALL IZITOC (IVAL,CHVAL*)	123
CALL IZMERG (PNAME,X0,Y0,SCALE,CHOPT)	109
CALL IZOUT (PNAME,ICYCLE*)	119
CALL IZPICT (*PNAME*,CHOPT)	108
CALL IZRTOC (RVAL,CHVAL*)	123
CALL IZSCR (PNAME,ICYCLE)	119
HIGZ/X11 interfaces routines	
CALL IXBOX (IX1,IX2,IY1,IY2,MODE)	226
CALL IXCLIP (X,Y,W,H)	220
CALL IXCLPX	229
CALL IXCLRWI	219
CALL IXCLSDS	219
CALL IXCLSWI	219
CALL IXCPPX (IPIX,IX,IY)	229
CALL IXDRMDE (MODE)	226

Table B.1: Overview of HIGZ calling sequences (cont.)

Calling Sequence	Page
CALL IXFLARE (N,IXY)	222
CALL IXGETCOL(INDEX,R*,G*,B*)	??
CALL IXGETGE (IWID,X*,Y*,W*,H*)	228
CALL IXGETWI (IWKID,IDG*)	228
CALL IXLINE (N,IXY)	221
CALL IXMARKE (N,IXY)	222
CALL IXNOCLI	221
INTEGER FUNCTION IXOPNDS (LENHST,CHOST)	218
INTEGER FUNCTION IXOPNPX (W,H)	228
INTEGER FUNCTION IXOPNWI (X,Y,W,H,LENTIT,CHTIT)	218
INTEGER FUNCTION IXREQLO (MODE,ITYP,IX*,IY*)	227
INTEGER FUNCTION IXREQST (IX,IY,LENTXT,*CHTEXT*)	227
ixsdswi (dsp,win) (this is a C routine)	230
CALL IXSELWI (WID)	219
CALL IXSETCO (INDEX,R,G,B)	223
CALL IXSETFC (INDEX)	225
CALL IXSETFS (ISTYL,IFASI)	224
CALL IXSETHN (LENHST,CHOST)	219
CALL IXSETLC (INDEX)	223
CALL IXSETLS (N,IDASH)	223
CALL IXSETLN (WIDTH)	223
CALL IXSETMS (ITYPE,N,IXY)	224
CALL IXSETMC (INDEX)	224
CALL IXSETTA (IH,IV)	225
CALL IXSETTC (INDEX)	225
INTEGER FUNCTION IXSETTF (MODE,LENFNT,CHFONT)	225
CALL IXSYNC (MODE)	227
CALL IXTEXT (MODE,IX,IY,ANGLE,RMAGN,LENTXT,CHTEXT)	222
CALL IXTXTL (IW*,IH*,LENTXT,CHTEXT)	226
CALL IXUPDWI	219

Bibliography

[1] L. Lamport. *LaTeX A Document Preparation System (2nd Edition)*. Addison-Wesley, 1994.

[2] CN/ASD Group. *PAW users guide, nProgram Library Q121*. CERN, October 1993.

[3] D. R. Myers. *GKS/GKS-3D Primer, nDD/US/110*. CERN, 1987.

[4] CN/ASD Group and J. Zoll/ECP. *ZEBRA Users Guide, nProgram Library Q100*. CERN, 1993.

[5] Graphics section. *Guide to computer graphics at CERN, nDD/US/1987*. CERN, 1990.

[6] CN/ASD Group. *HBOOK Users Guide (Version 4.21), nProgram Library Y250*. CERN, January 1994.

[7] R. Brun and H. Renshall. *HPLOT users guide, nProgram Library Y251*. CERN, 1990.

[8] CN/ASD Group. *KUIP – Kit for a User Interface Package, nProgram library I202*. CERN, January 1994.

Index