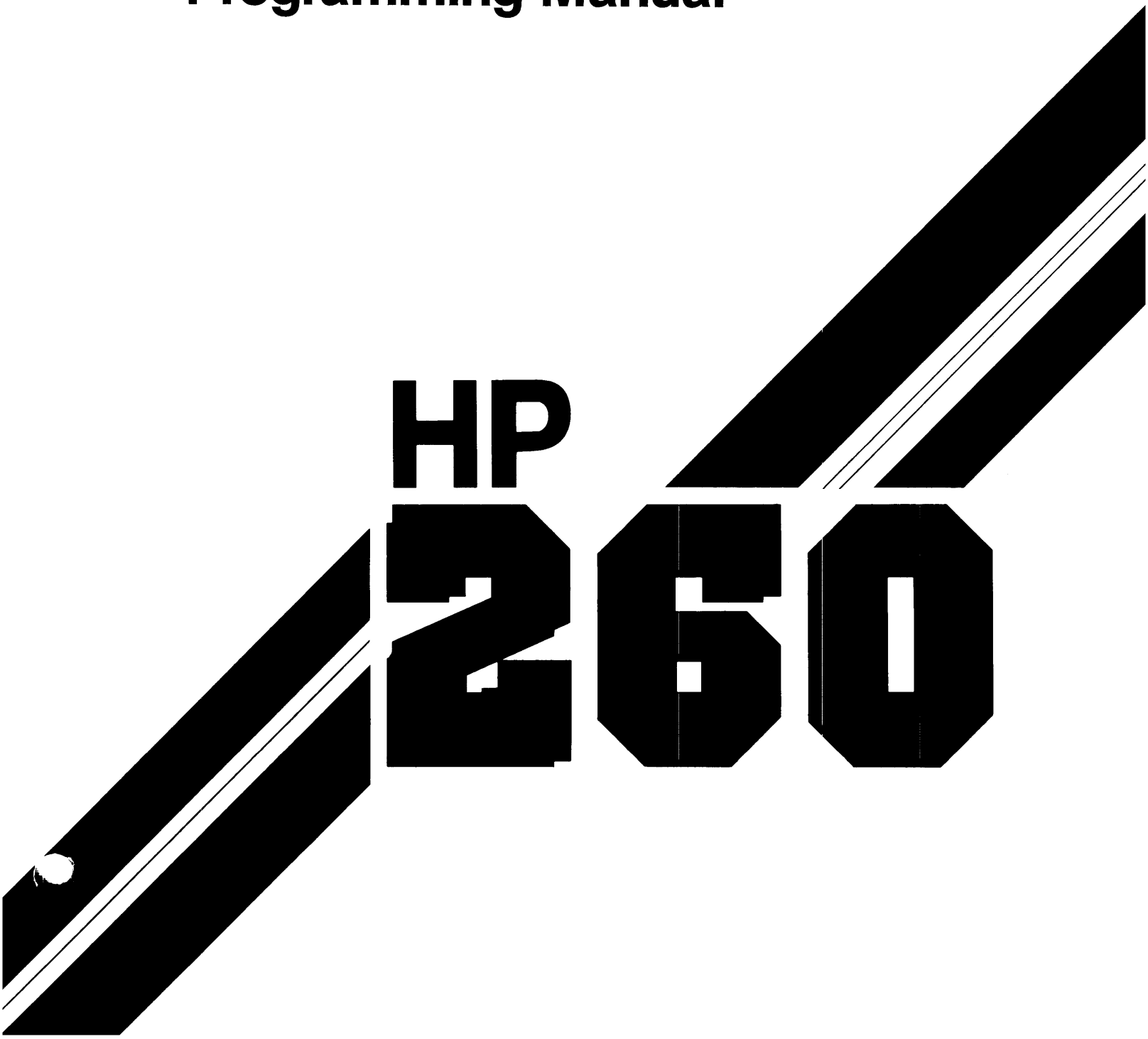
HEWLETT-PACKARD

GPL/260 Programming Manual



HP 260 Computer Systems

GPL

Programming Manual

HERRENBERGER STRASSE 130, D-7030 BOEBLINGEN

Part No. 45261-90064
E0986

Printed in Federal Republic of Germany 09/86

FEDERAL COMMUNICATION COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT
(for U.S.A. only)

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

CONTENTS

CONTENTS (continued)

Section 9
DRAWING TEXT STRINGS

Section 10
USING SOFTWARE CHARACTERS

Section 11
VIEWING TRANSFORMATIONS

CONTENTS (continued)

CONTENTS (continued)

Appendix E
SUMMARY OF GPL SUBPROGRAMS

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Jul 1985	B.07.00
Second Edition	Sep 1986	B.08.00

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Pages	Date
all	Sep 1986

INTRODUCTION TO GPL

SECTION 1

NOTE

Graphics Plotter Library system software is included with the operating system and system software of the following computer systems:

- all HP 250 Business Computer Systems
- all HP 260 Business Computer Systems

This manual describes the operation of GPL system software supplied with the most recent version operating system and system software of the previously mentioned computer systems. (At this edition of the GPL Programmer's Manual, B.08 is the most recent version of operating system and system software for these computer systems.)

Note that not every plotting device is supported on each computer system. To find out if a particular plotting device is supported on your computer system contact your third party representative or your local HP Sales Office.

In this manual, the phrase "your computer system" refers to the HP 250 and HP 260 Business Computer Systems.

WHAT IS GPL?

GPL is a BASIC-language subprogram library callable from BASIC for drawing pictures on HP Graphic Plotters. The library is intended for the application programmer who wants to include simple plotter graphics in his application and wants easy access to the plotters. GPL is implemented in BASIC and therefore uses some application program memory space.

WHAT IS COMPUTER GRAPHICS?

In the simplest case, "Computer Graphics" is the output of pictures on certain computer peripherals. You may view these pictures on CRT's (graphics terminals) or graphics plotters. In business applications, output computer graphics may be used to produce graphs, charts, maps, floor plans, and engineering drawings.

In more complicated applications, sophisticated peripherals may be used for interactive, real time input and output of pictures. This is called "Interactive Computer Graphics".

GPL is oriented to the output of hardcopy pictures on HP Graphics Plotters. Facilities for storing pictures in files for later output on a device are included.

GPL is not intended to support interactive computer graphics and does not contain facilities for graphics input.

PAYOFF OF COMPUTER GRAPHICS

Computer graphics is a good tool for displaying and analyzing data. For example, you can spot trends and problem areas with a chart much faster than with a list of numbers. The analysis is also more easily communicated to others in graphical form. For large audiences, you can draw charts on overhead slide acetates and use an overhead projector to view the charts.

GPL is ideal for producing charts and overhead acetate slides in typical business situations.

CAPABILITIES OF GPL

Key features include:

- Easy to use, mnemonic syntax

```
CALL Gpl_move(G$,X,Y) — moves pen to (X,Y)
CALL Gpl_draw(G$,X,Y) — draws a straight line to (X,Y)
CALL Gpl_text(G$,"HELLO") — draws HELLO at pen location
```

In the above calls, G$ is a string which contains all temporary parameters passed between GPL calls.

- Device-independent programs and application manuals
- Oriented towards business applications

  Emphasis on charts, graphs, and overhead slides

- Is compatible with HP RS-232-C Graphics Plotters
- Powerful

  Nine line styles, selection of multi-colored pens, control of pen velocity

  Numerous decorative character fonts, control of character width, height, and rotation

  Device-independent coordinate system - world (problem) coordinates, logical plotter

  Rotation, translation, and scaling of entire plots

  File storage of pictures - convenient for multiple copies of pictures

- First Day Productivity

  It's easy to get started. Instructions are complete.

GPL was designed for the business environment. It may not work for all applications because extra computer memory is occupied by extensive code to make GPL easy to use.

CONFIGURING THE PLOTTER

SECTION 2

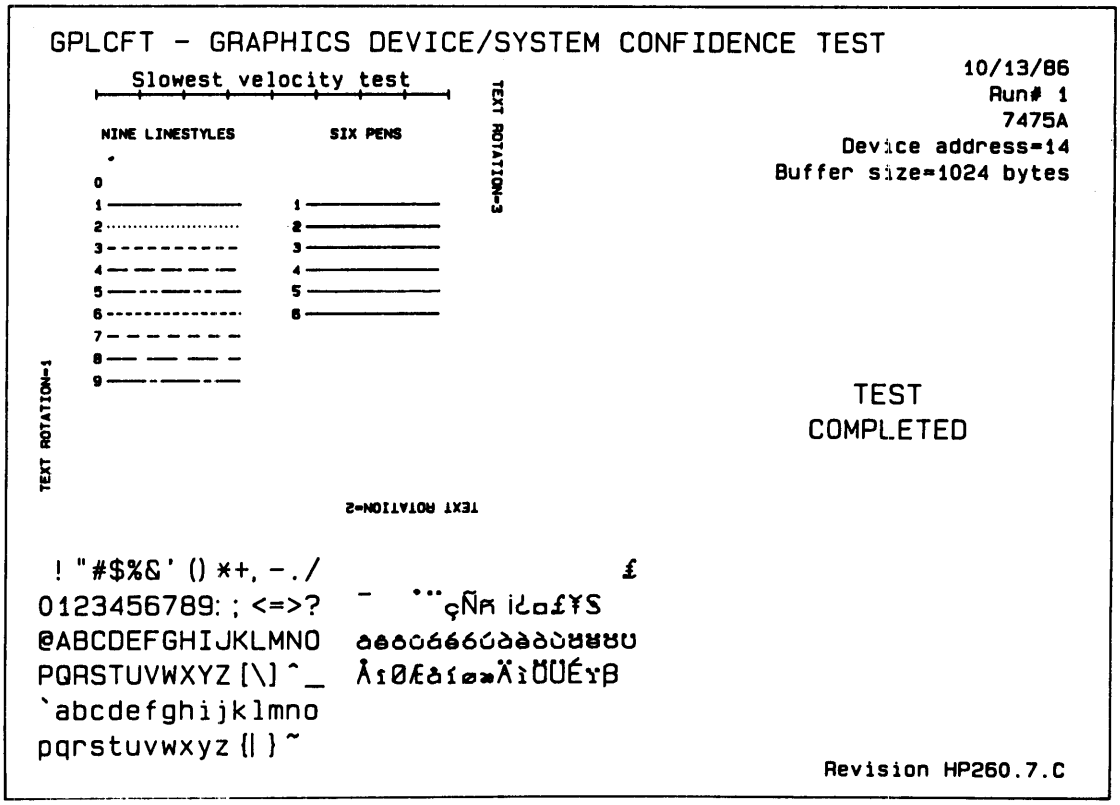
Before learning how to program with GPL, connect the plotter to your computer system. Instructions appear in "Operating and Managing"(part number 45260-900020). Plotter installation includes hardware connection and software configuration.

SOFTWARE CONFIGURATION

Run the CONFIG utility program, described in the "Utilities Manual" (part number 4526-90061). GPL requires configuration of the PACK and TIO DROM's. In the "Asynchronous Port Configuration" function, configure the plotter as a 26xx terminal, format 8N1.

HARDWARE CONNECTION

See "Operating and Managing" for hardware configuration. Be sure to perform a plotter self-test and system confidence test.



LOAD THE GRAPHICS SOFTWARE FROM DISC

The GPLLOD program transfers GPL files from one medium to another. GPL is supplied on one of the following types of distribution medium:

- the Operating System Tape Cartridge
- the 8-inch floppy labeled GPL
- the two 3.5-inch microfloppies labeled GPL LIBRARY and GPL TEXT

Use the following procedure to copy the GPL files from your distribution medium to your fixed disc.

```
RUN "GPLLOD"
```

Press the softkey corresponding to the device from which the files are to be transferred. Then press the softkey corresponding to the device on which the files will be stored.

GPLLOD asks what should be transferred:

GPL25X LIBRARY — All GPL files except those for software-generated character sets.

SOFTWARE TEXT — The files required to plot software-generated characters.

ALL — Both of the above.

Now press the softkey corresponding to the group of files you want to transfer.

"Operating and Managing" contains some troubleshooting suggestions. Follow these suggestions when a problem occurs.

PROGRAMMING GUIDELINES

SECTION 3

Before learning how to program with GPL, you must understand several concepts relating to the way it works. The disc contains a sample program to help you understand the structure of a GPL program.

NAMING CONVENTIONS

The names of all GPL calls begin with the prefix "Gpl__". The names of all GPL internal (nonuser) subprograms and functions begin with the prefix "Gpli". You should name all of your application subprograms and functions with a prefix other than "Gpl__" or "Gpli".

GPL$ STRING

The first parameter in most GPL calls is a 1200 character string that contains all parameters that are passed between the GPL subprogram calls. In all of the examples in this manual, the string will be called Gpl$. To dimension the string, use the command:

```
DIM Gpl$[1200]
```

This variable name must be passed between all modules that use GPL subprograms. It is important that this string variable be dimensioned to be exactly 1200 characters. Attempting to pass a string of length other than 1200 between GPL subprograms will result in an error.

PARAMETERS IN GPL SUBPROGRAMS

All numeric parameters in the GPL subprograms parameter list must be of type REAL.

USE OF REQUEST AND RELEASE

GPL uses TIO whenever the device address is in the range 11 to 20. Each time data is to be sent to the plotter, GPL inquires about available plotter buffer space. If the buffer is almost full, GPL waits until buffer space is available. TIO requires that the device be REQUEST'ed before a Gpl__plotteris call, and RELEASE'd after a Gpl__terminate call. This lets other users access the device. It is the responsibility of the application program, not GPL, to perform REQUEST and RELEASE operations.

LOADING AND DELETING THE GPL LIBRARY FILES

The GPL library occupies a large block of user memory. It is necessary to load and delete several GPL subprogram files during the execution of a program. This can all be done under program control. For example, you could use statements of the form:

```
DEL 5001,9999
LOAD SUB "GPL252",5001,1
```

at the beginning of your program to load the GPL initialization subprograms.

Clear the entire subprogram library from the user memory at the end of your program with the statement:

```
DEL 5001,9999
```

The GPL library is organized into logical subprogram files as follows:

FILE	Contains subprograms for:
GPL251	Device selection
GPL252	Initialization and plotting area specifications
GPL253	Plot setup and specifications
GPL254	Text and straight lines
GPL255	Straight lines
GPL256	Text
GPL257	Plot termination
GPL258	Determining width of text strings

SAMPLE PROGRAM

A sample application program, GPLE01, has been included in the GPL library for use with this chapter. The sample program draws a typical business line chart on any of the supported plotters. Extensive comments are included to help you read the program.

To run the program, perform the following steps:

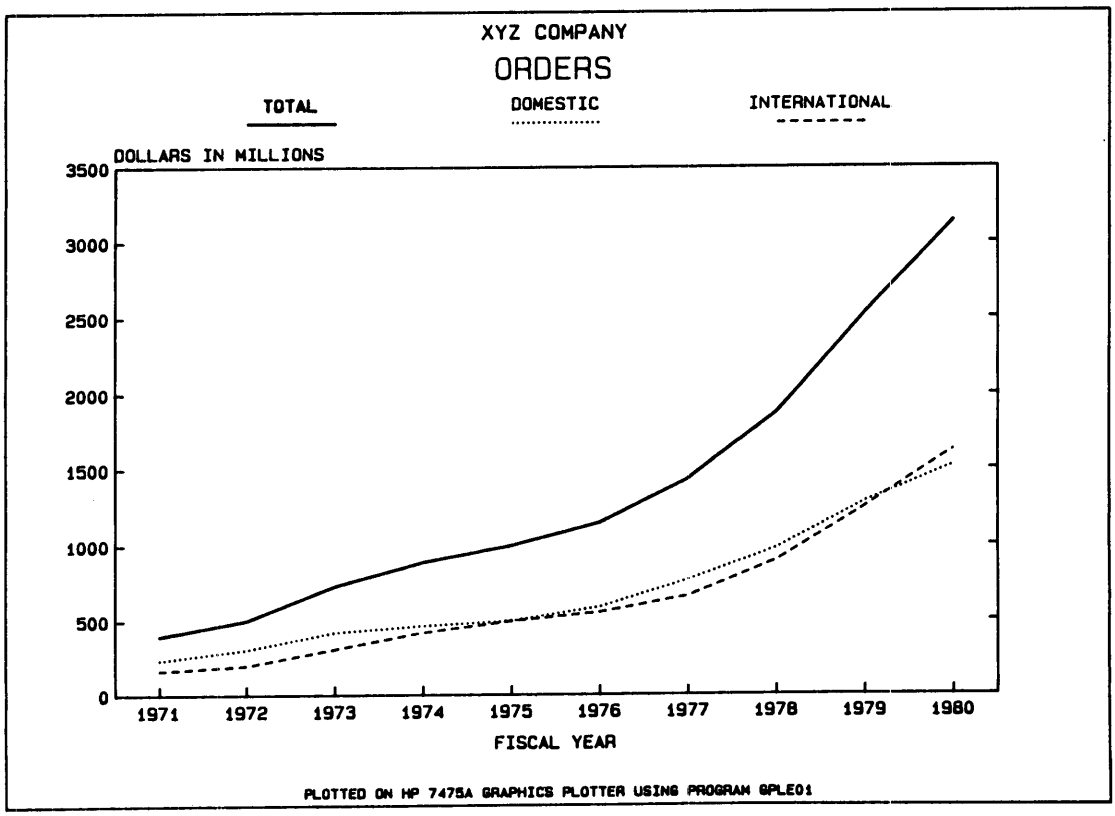
1) MSI ":<volume-spec for the disc containing the GPL files>"

2) RUN "GPLE01" and follow the instructions on the screen.

The program will ask for the plotter model number and device address. The address is 8 or 11 through 20.

Device address 8 will send the plotter protocol to the CRT display. First, run the program using 8 and compare the display with a listing of the program.

Now run the program, giving the address to which the plotter is connected. The picture that is generated should be similar to that shown below.



List and Study the Program

List the program named "GPLE01". Study the listing of GPLE01 and try to understand the meaning of the GPL calls from mnemonic names and comments. All GPL calls start with the prefix "Gpl__". Try to understand how the plot was generated. These calls will be explained in the following chapters.

GPL is divided into several modules. The desired subprogram must be loaded before it can be used.

```
DEL 7000,9999
LOAD SUB "GPL252",7000,1      ! for device initialization
DEL 7000,9999
LOAD SUB "GPL253",7000,1      ! for setting plotting coordinates

DEL 7000,9999
LOAD SUB "GPL254",7000,1      ! for plotting

DEL 7000,9999
LOAD SUB "GPL257",7000,1      ! for termination
```

The library is deleted at the end of execution by the statement

```
DEL 7000,9999
```

The LOAD SUB statements require either that files "GPL252", "GPL253", "GPL254" and "GPL257" reside on the MSI disc or that the LOAD SUB statements contain a device specifier. In addition, execution requires a 64K partition.

TROUBLESHOOTING

Nothing Happens When Program Is Run

a. Check to see that the device is on and that status lights are correct.

b. Run GPLCFT to make sure that the plotter is properly connected. If this fails, follow the troubleshooting steps for GPLCFT.

The above checks should turn up most possible problems.

Program Halts on Error Condition

No ON ERROR control is supplied in this simple example. If an error message comes up with a program halt, consult an "Operating and Managing" or "Syntax Reference Guide". Some errors may be difficult to interpret. Typical errors include:

Error #	Probable Cause
2	Memory Overflow (occurs in a 32K partition, but should not happen on a 64K partition)
56	The GPL library was not found on the MSI disc.
130	Invalid device address for your hardware configuration.
131	The device REQUEST'd was not available because somebody else reserved the device.
132	The wrong type of device was REQUEST'ed, such as a workstation.
133	Selected device is down.

Picture Different Than Expected

a. Run the program to verify that the error is repeatable.
b. Run program GPLCFT to check for system and device problems.

If test (a) shows that the error is not repeatable, then it may be an intermittent problem. Check the entire configuration. If the problem still exists, then you may have to contact your Hewlett-Packard service representative for further help.

If test (a) shows a repeatable problem and GPLCFT uncovers no problem, there may be a problem somewhere in the system software or hardware. Recheck the configuration.

SUMMARY

In this chapter you have been exposed to a typical graphics application program. Operations performed include loading the GPL library with the application and plotting on one of the supported plotters. After completing these tasks, you are ready to learn how to program a graphics application.

PLOT INITIALIZATION AND TERMINATION

SECTION 4

In this chapter, you will learn the basic concepts of plotting with GPL, as well as the routines necessary to start and stop plotting. Several subprograms are available to initialize the plotter and to terminate plotting. A plot may terminate in one of three ways: normal completion of the program, an error condition, or a user initiated exit.

These subprograms are introduced in this chapter:

Gpl_plotteris	Initializes a specified plotter and the data stored in Gpl$.
Gpl_terminate	Terminates plotting and processes all data in the software and plotter buffers.
FNGpl_errm$	Displays error message for GPL errors.
Gpl_abortplot	Terminates plotting immediately and flushes all data in the buffers.
Gpl_clear	Advances paper on chart advance plotters and moves pen stable to the upper right position.
Gpl_transmit	Processes all data in the GPL buffer and brings plot to its current state.

GPL_PLOTTERIS

This is the first call to make when actually plotting. The subprogram initializes the device and the parameters in the Gpl$ string. The syntax is:

```
Gpl_plotteris(Gpl$,Model$,Dev_add,Mess_add,Error)
```

The parameters are:

Model$: Model number of graphics device stored in a string. Model$ can also contain the paper size appended to the model name (as is returned by a call to the Gpl_devident subprogram). For example ,the parameter Model$ might now contain a string similar to "7550A/A3" when calling Gpl_plotteris. If a paper code is explicitly specified, it must match the actual plotter setting. If only the model name is given, the Gpl$ string is initialized with the actual plotter setting.

The following are allowed values:

7220A	7221A*	7225A/3	7225B/3	7470A	7550A
	7221B	7225A/3V	7225B/3V	7440A	
7220C	7221C				
		7225A/4	7225B/4	7475A	
7220S	7221S	7225A/4V	7225B/4V		
7220T	7221T				

* The HP 7221A plotter is supported on all HP 250 and HP 260 models except Model 35. A hardware incompatibility prevents the HP 7221A from being initialized on the Model 35.

NOTE

"7225A/3" means that you have an HP 7225A Graphics Plotter equipped with option 17603A Personality Module. "7225A/4" means option 17604A Personality Module. The "V" is associated with plotters that have velocity select capability. A similar set of conventions holds for the 7225B Graphic Plotters.

Dev_add: This is the system address to which the graphics device is connected. Allowed values are in the ranges 8-9 or 11-20. Send the data to your CRT (address 8) or the bit bucket (address 9) when you first test a program to verify that the program will execute with no errors. A real device is assumed to be there only when 11-20 is specified . Gpl_plotteris checks to see if the device that you specified is the device that is there.

Mess_add: This is the system address for operator messages for a plotter. Usually this is 8, the workstation from which your application is running. Allowed values are in the range 8-9.

Error: This is an error return parameter to signal whether the initialization was successful. The currently supported error returns include:

0	-No hard error encountered
1-1099	-ERRN for system errors not explicitly handled by Gpl_plotteris.
1151	-Invalid Gpl$ (probably insufficient length)
1152	-Invalid Model$ (not one of the supported devices)
1153	-Invalid Device_address (Must be 8-9 or 11-20)
1154	-Invalid Message_address (must be in range 8-20)
1161	-Plotter ownership error - must do REQUEST before calling Gpl_plotteris
1162	-No response from plotter (when the devices address ranges from 11 to 20)
1163	-Device is not the specified model (for device address ranging from 11 to 20)
1164	-Plotter could no be initialized (for a variety of reasons)
1165	-Unsupported plotter response during identify sequence
1166	-Invalid plotter response during identify sequence
1181	-Plotter responded initially but later did not. All further sending of data to plotter stopped.

This error parameter should be checked in all applications. If non-zero, then unpredictable results can occur with further program execution. The function FNGpl_errm$, described in a later section and shown in the above example, may be used to display the error message.

To use Gpl_plotteris, follow this sequence:

```
DIM Gpl$[1200]
.
! Assign values to Model$,Dev_add,Mess_add
.
ON ERROR GOTO Request_error
REQUEST Dev_add
OFF ERROR
CALL Gpl_plotteris(Gpl$,Model$,Dev_add,Mess_add,Error)
IF Error<>0 THEN Init_error
.
.
! Case where REQUEST fails
!  ERRN is typically in the range from 130 to 139
Request_error:OFF ERROR
BEEP
DISP "Error in attempting to REQUEST plotter."
DISP ERRM$
.
.
! Initialization error
Init_error: DISP FNGpl_errm$(Error)
.
.
```

GPL_TERMINATE

This is the last call to make when actually plotting. It processes data in the buffers, leaves the device in a friendly state, and cleans up any picture files. The syntax is:

```
CALL Gpl_terminate(Gpl$,Error)
```

The Error return parameter gives a code for the last GPL call in which invalid parameters were encountered. 0 means no such errors.

1101 - Gpl_clear
1102 - Gpl_defldc
1103 - Gpl_draw
1104 - Gpl_frame
1105 - Gpl_idraw
1106 - Gpl_imove
1107 - Gpl_linestyle
1108 - Gpl_lorg
1109 - Gpl_message
1110 - Gpl_move
1111 - Gpl_penspeed
1112 - Gpl_physarea
1113 - Gpl_devorigin
1114 - Gpl_pen
1115 - Gpl_physrotate
1116 - Gpl_physviewp
1117 - Gpl_terminate
1118 - Gpl_text
1119 - Gpl_transmit
1120 - Gpl_textrotate
1121 - Gpl_tsize
1122 - Gpl_useldc
1123 - Gpl_usewc
1124 - Gpl_viewport
1125 - Gpl_where
1126 - Gpl_window
1127 - Gpl_cset

In addition, the following error return parameters signify that data was lost during plotting. These errors take precedence over the "soft errors" reported above.

1181 - Plotter responded initially but later stopped.
Further plotting to device terminated.
1182 - Invalid plotter response.
Further plotting to device terminated.
1183 - Picture file could not be opened during plotting.
Sending plot data to file terminated.

FNGPL_ERRM$

As an aid to handling errors and debugging simple programs, an error reporting function, FNGpl_errm$, has been included in the GPL library.

Whenever an error is returned by:

```
Gpl_plotteris
Gpl_terminate
```

and any other GPL subprograms with an error to return parameter, an error message may be displayed. Use the statement

```
DISP FNGpl_errm$(Error)
```

to display it on the screen.

GPL_ABORTPLOT

```
CALL Gpl_abortplot(Gpl$)
```

aborts the current plot. This call flushes the software and plotter buffers of all data and terminates plotting immediately. Use Gpl_terminate after Gpl_abortplot to leave the plotter in a friendly state.

A common way to use the abort feature is to allow the user to push a softkey labeled "ABORT PLOT" or "STOP PLOTTING" when actually plotting.

```
DIM Gpl$[1200]

ON ERROR GOTO Request_error
REQUEST 12
OFF ERROR
DEL 7001,9999
LOAD SUB "GPL252",7001,1
CALL Gpl_plotteris(Gpl$,"7220A",12,8,Error)
IF Error>0 THEN Init_error
ON KEY #8,#16,#24 :"STOP PLOTTING" GOTO Abort_plot
DEL 7001,9999
LOAD SUB "GPL254",7001,1
.
<code for plotting>
.
Abort=0
GOTO Terminate
Abort_plot:  DEL 7001,9999
             LOAD SUB "GPL257",7001,1
             CALL Gpl_abortplot(Gpl$)
DISP "Plot aborted."
Abort=1
Terminate:   DEL 7001,9999
             LOAD SUB "GPL257",7001,1
             CALL Gpl_terminate(Gpl$,Error)
RELEASE 12
IF Abort=0 THEN DISP "Plot terminated normally."
IF Error>0 THEN DISP FNGpl_errm$(Error)
GOTO End_message
.
Request_error: BEEP
OFF ERROR
IF ERRN=130 THEN DISP "Device address out of allowed range."
IF ERRN=131 THEN DISP "Device is already reserved."
IF ERRN=132 THEN DISP "Invalid device type."
IF ERRN<130 OR ERRN>132 THEN DISP ERRM$
GOTO End_message
.
Init_error:DISP FNGpl_errm$(Error)
End_message: DISP "Program terminated."
DEL 7001,9999
END
```

The HALT key should not be programmed to abort the plot because a HALT is immediate. The application may be in one of the GPL subprograms when HALT is pushed. This latter case would cause the application program to stop executing. Any data left in the device buffer is NOT flushed and will be plotted.

GPL_CLEAR

Gpl_clear performs a variety of functions; it should be called at the end of every plot.

a) Advances the paper for chart advance plotters when roll paper is loaded in the plotter.

b) Puts the pen away (for 7220 and 7221 series plotters).

c) Moves the pen stable to the upper right position so that paper may be easily changed.

d) Places a special code in picture files for controlling the clear function and the aborting of plots. (This last function is used only when Gpl_fileis has been called).

To execute the clear function

```
CALL Gpl_clear (Gpl$,Crt,Hardcopy)
```

Two parameters (besides Gpl$) must be supplied:

Crt - Reserved for future use. 0 must be used at this time. Hardcopy - Allowed values are 0, 1, 3, and 4.

0 No operation (NOP)
1 Moves pen out of way
3* Causes half page (ISO A4) paper advance
4* Causes full page (ISO A3) paper advance

For example, CALL Gpl_clear(Gpl$,0,3) causes an ISO A4 (or English A) size paper advance. The actual advance distance is controlled by the ENGLISH/METRIC switch setting on the plotter.

* When used with the new HP 7550A plotter, this parameter can also be used to feed a full page of paper from the plotter's paper magazine. Setting the hardcopy parameter to either "3" or "4" causes a full page of paper to be fed to the HP 7550A plotter from the plotter's paper magazine.

GPL_TRANSMIT

Graphic data is buffered internally in the GPL software. This data is stored in the Gpl$ string, and is cleared automatically when it gets full. The amount of buffer space allows about 33 separate or 65 contiguous straight lines to be stored. About 300 characters of text can also be stored in the buffer. Usually a mixture has been stored.

The Gpl_transmit function may be used to clear this buffer. The function is useful for performance optimization and for synchronizing a plot with user interactions. The syntax for Gpl_transmit is:

```
Gpl_transmit (Gpl$,Option)
```

For example, you may wish to ask a question in the middle of a plot. The Gpl_transmit function will cause the plot to be updated so that the user can see the current state of the plot at the same time that the question is asked.

In another case, you may notice that there are pauses in the plotting due to the time it takes to compute points (Gpl does consume significant computation time). You can minimize such delays by calling Gpl_transmit at judiciously selected places in your program. A good place to call Gpl_transmit is after CALL Gpl_text with a long string of text supplied.

This function is called automatically when the following are called:

```
Gpl_clear
Gpl_terminate
```

The option parameter is reserved for future use. It should be set to 0 at all times.

SUMMARY

In this chapter you have learned how to initialize and terminate plotting. Special care must be taken to terminate a plot with one of the termination calls defined by GPL, as an abnormal exit may leave the plotter in an unfriendly state. GPL includes an error routine to handle those errors that occur most frequently in graphics plotting.

DEVICE SELECTION

SECTION 5

To enable the programmer to write device-independent code, GPL includes two subprograms which can be called to determine what devices are available and the characteristics of each device. It is also possible to maintain a configuration file specifying which plotters are available on a particular system.

The following subprograms are covered in this chapter:

Gpl_config — Reads the device configuration from a data file and returns the model numbers and addresses of devices configured into your system.

Gpl_devident — Attempts to identify the device at a specified address and returns information about any device (if any) found at that address.

Gpl_config

This subprogram reads the device configuration file GPL%CF (or an alternate file name if you supply it) and returns the model numbers and system addresses of graphics devices connected to your system.

To read the default file GPL%CF, use the following call sequence:

```
DIM Model$(10),Dev_add(10),Mess_add(10)
CALL Gpl_config("","",No_devices,Model$(*),Dev_add(*),
  Mess_add(*),Error)
```

The number of devices in your system will be returned in the parameter No_devices. If 0, an error occurred, and the Error parameter should be checked. The above dimension statements allow up to 10 devices to be returned. If the file contains more than 10 devices, adjust the dimension statements accordingly.

To read from another configuration file, for example "ABCDEF", just supply the file name as the first parameter in the call.

```
CALL Gpl_config("ABCDEF"," ",No_of_devices,Model$(*),
  Dev_add(*),Mess_add(*),Error)
```

The second parameter is the protect code for the configuration file.

The array parameters (Model$(*), Dev_add(*), and Mess_add(*)) are arrays containing the model number, graphics device address, and message device address.

The Error parameter is 0 only if no errors occurred. Typical errors include:

1191 - ASSIGN Return=1: Configuration file not found.
1192 - ASSIGN Return=2: Configuration file protected.
1193 - ASSIGN Return=3: Specified file is of wrong type.
1194 - ASSIGN Return=4: Access violation.
1195 - ASSIGN Return=5: Other error.

1196 - Invalid configuration file data format
1197 - Invalid model number in configuration file
1198 - Invalid device address in configuration file
1199 - Invalid message address in configuration file

The GPL error reporting function FNGpl_errm$ may be used to display the error. Gpl_config is in file "GPL251". To load Gpl_config in a program module, use:

```
LOAD SUB "GPL251", line_start, line_increment
```

Gpl_devident

This subprogram attempts to identify the device at the supplied device address. If the identification is unsuccessful, or the device is not a supported plotter, then an error is returned. During the identification process, the plotter RS-232-C handshake is configured to communicate with your computer system.

Gpl_devident is contained in files "GPL251" and "GPL252". It is also used by Gpl_plotteris during the initialization sequence.

To use the function, you must first REQUEST the device address, then supply the device address to Gpl_devident.

```
REQUEST Device_address
CALL Gpl_devident(Device_address,Model$,Advance,Buffer,Pens,Error)
```

The returned parameters include:

Model$ - Model number of the plotter at this address. The returned value is not always exact, but is close enough for GPL to be aware of the plotter's capabilities.

Returned	Possible
7220A	7220A,7220S single sheet mode
7220C	7220C,7220T (single sheet mode)
7220S	7220S (auto advance mode)
7220T	7220T (auto advance mode)
7221A	7221A
7221B	7221B,7221S (single sheet mode)
7221C	7221C,7221T (single sheet mode)
7221S	7221S (auto advance mode)
7221T	7221T (auto advance mode)
7225A/3	7225A/3V,7225A/3,7225B/3V,7225B/3
7225A/4	7225A/4V,7225B/4,7225B/4V,7225B/4
7470A/size	7470A
7475A/size	7475A
7550A/size	7550A
7440A/size	7440A

For the 7470A, HP ColorPro(7440A), 7475A and 7550A plotters, a call to Gpl_devident returns the paper size for which the plotter is set. Model$ contains information of the form: model_number/paper_size. For example, a call to Gpl_devident might return:

"7470A/A" (an HP 7470A Plotter with ANSI A size paper)
"7475A/B" (an HP 7475A Plotter with ANSI B size paper)
"7550A/A3" (an HP 7550A Plotter with ISO A3 size paper)
"7440A/A" (an HP 7440A plotter with ANSI A size paper)
"7440A/A4" (an HP 7440A plotter with ISO A4 size paper)

NOTE

The source of the paper size information varies with the plotting device. With the HP 7470A, HP ColorPro (7440A) and HP 7475A plotters, the maximum plot-surface (paper size) depends on the position of the plotter's paper size switch. With the HP 7550A the paper size is determined automatically by the plotter.

Advance - Parameter (redundant) specifying whether automatic chart advance capability is present.

0 - No advance
1 - Auto advance present and enabled

Buffer - Buffer size for the plotter is in bytes

Buffer Size	Plotter
0	Device address 8 or 9
600	7225A/4,7225A/4V,7225B/4,7225B/4V
630	7225A/3,7225A/3V,7225B/3,7225B/3V
928 or 2976	7220A/C/S/T
1110 or 3038	7221C/T
1128 or 3056	7221B/S
1158 or 3086	7221A
255	7470A
60	HP ColorPro (7440A)
1024	7475A
1024 (programmable to 1974)	HP ColorPro (7440A) with GEC
1024 (programmable to 12000)	7550A

In the above table "GEC" stands for "Graphics Enhancement Cartridge".

Pens - Number of pens on the plotter (1, 2, 4, 6, or 8).

Error - Error return parameter of the initialization or the identification is not successful.

Typical values for the error include:

1153 - Invalid device address
1161 - Device address must be REQUEST'd
1162 - No response from plotter
1164 - Plotter responded, but attempts to initialize failed
1165 - Unsupported plotter

1166 - Invalid response from plotter
1181 - Plotter responded then stopped responding

Again, the function FNGpl_errm$ may be used to display the error.

SUMMARY

In determining the device to be used for a particular plot, you can either rely on the default configuration file, or inquire about which devices are at addresses 11-20. If your plotter configuration is relatively stable, it is much faster to simply read the configuration file.

ESTABLISHING YOUR COORDINATE SYSTEM

SECTION 6

This chapter deals with the characteristics of individual plotters, and the logical coordinate system used by GPL. GPL allows the programmer to write device-independent code by specifying the logical units to be mapped onto the physical plotting surface.

The following subprogram calls are introduced in this chapter:

Gpl_physarea	Specifies the physical dimensions of the area within which the plot will be drawn.
Gpl_devorigin	Defines the origin point of the physical plotting area on the available plotting surface.
Gpl_defldc	Defines the logical coordinate system in user units.
Gpl_physviewp	Specifies the location of the logical device coordinate (LDC) space on the physical area.
Gpl_physrotate	Determines the orientation of the plot on the plotter. The default value allows GPL to rotate the plot to best fit the device being used.

LOGICAL UNITS VERSUS PHYSICAL UNITS

To plot on a device, you need to have the coordinates of a specific plotter. In GPL, you specify your own coordinates. You may choose any units that are convenient to your application. These coordinates may be independent of any particular plotter, which allows you to write device independent graphics programs.

SCALING AND PLACEMENT ON PHYSICAL DEVICE

In order to take full advantage of GPL's capabilities, you need to take into account some important device differences, such as different plotting areas and aspect ratios.

PHYSICAL PLOTTER CHARACTERISTICS

The supported plotters have the following plotting surface capabilities:

	Origin Offset		Maximum Plotting Area	
Series	Width mm	Height mm	X mm	Y mm
7220A/C 7221A/B/C	10	10	400	285
7220S/T 7221S/T	10	27	400	285
7225A/B	10	10	285.5	203.5
7470A(US)	1	10.2	259.1	190.2
7470A(A4)	9.5	7	274.2	190.2
7440A(US)	7.0	8.0	191.5	257.3
7440A(A4)	9.2	5.6	191.5	272.3
7475A(US-A)	5	14.5	198.2	257.9
7475A(US-B)	3.5	4.9	257.9	414.2
7475A(A4)	13.5	11.5	192.2	274.7
7475A(A3)	0.6	13.1	274.7	402.2
7550A(US-A)	2.2	15.3	196	254.2
7550A(US-B)	0.8	1.9	254.2	411.2
7550A(A4)	10.7	9.8	190	271.7
7550A(A3)	0	10.2	271.7	399.2

The origin offset denotes how far the plotting origin differs from the lower left corner of the paper. (See DEVORIGIN in Appendix D.)

Several calls, including Gpl__physarea, Gpl__devorigin, Gpl__physrotate, and Gpl__physviewp, exist to control the placement of the plot on your particular plotter.

PAPER SIZE

Typical paper sizes used in plotting include the following:

Sheets

ISO A3 - 297 mm by 420 mm
ISO A4 - 210 mm by 297 mm
US/English "B size" - 11 inches (279 mm) by 17 inches (432 mm)
US/English "A size" - 8-1/2 inches (216 mm) by 11 inches (279 mm)

Rolls (for chart advantage plotters)

Metric (317 mm high, 10mm perforation widths)
US/English (315 mm high, 18 mm perforation widths)

Acetate (Hewlett-Packard supplied)

8.5 in (216 mm) by 10.5 in (267 mm)
8.5 in (216 mm) by 11 in (279 mm) for 7470A paper-backed acetate.

GPL_PHYSAREA

```
CALL Gpl_physarea(Gpl$,Width_pamm,Height_pamm)
```

Gpl_physarea is used to specify the WIDTH and HEIGHT of your plot in mm. The defaults are:

Width=250mm
Height=180mm

Other values may be set as in the following example:

```
CALL Gpl_physarea(Gpl$,160,220)
```

sets the physical area to a WIDTH of 160mm and a HEIGHT of 220mm.

```
CALL Gpl_physarea(Gpl$,254,177.8)
```

sets the physical plotting area to a WIDTH of 254 mm (10 inches) and a HEIGHT of 177.8 mm (exactly 7 inches).

This physical plotting area may or may not be rotated to fit on the selected plotter, depending on the physical characteristics of the plotter. The optional rotation may be made explicit by using Gpl_physrotate, as described later in this chapter. It is important that the paper be oriented according to the paper loading instructions given in "Operating and Managing".

GPL_DEVORIGIN

```
CALL Gpl_devorigin(Gpl$,X_lower_left_plmm,Y_lower_left_plmm)
```

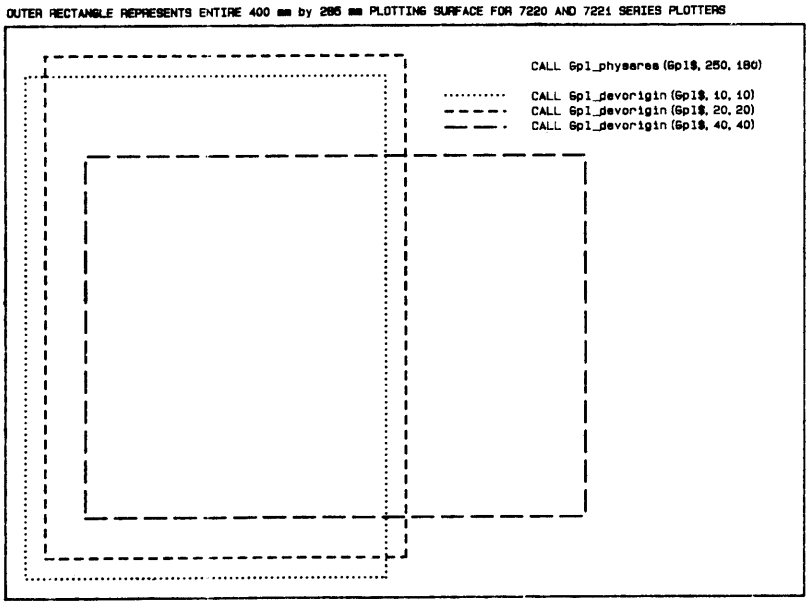
The physical plotting area is placed on the plotting surface with the Gpl_devorigin subprogram. The parameters are the X and Y position in mm of the bottom and left edges of the plot on the plotting surface relative to the plotter's addressable plotting origin.

For example,

```
CALL Gpl_devorigin(Gpl$,0,0)
```

starts the plotting area at the plotting surface origin (regardless of whether or not the plot is rotated).

It is important to note that the devorigin parameters refer to the lower left of the plotting surface (which is never rotated) and not the plot (which may or may not be rotated).



The following default values are used if Gpl_devorigin is not explicitly called.

Plotter Series	X_origin mm	Y_origin mm
7220A/C 7221A/B/C	10	10
7220S/T 7221S/T	10	27
7225A/B	10	10

7470A(US)	1	10.2
7470A(A4)	9.5	7
7440A(US)	7.0	8.0
7440A(A4)	9.2	5.6
7475A(US-A)	5	14.5
7475A(US-B)	3.5	4.9
7475A(A4)	13.5	11.5
7475A(A3)	0.6	13.1
7550A(US-A)	2.2	15.3
7550A(US-B)	0.8	1.9
7550A(A4)	10.7	9.8
7550A(A3)	0	10.2

PAPER PLACEMENT

See "Operating and Managing" for uniform paper loading instructions. Most of your plotting will be done on either US/English A size paper (8.5 in by 11 in) or ISO A4 size paper (210 mm by 297 mm).

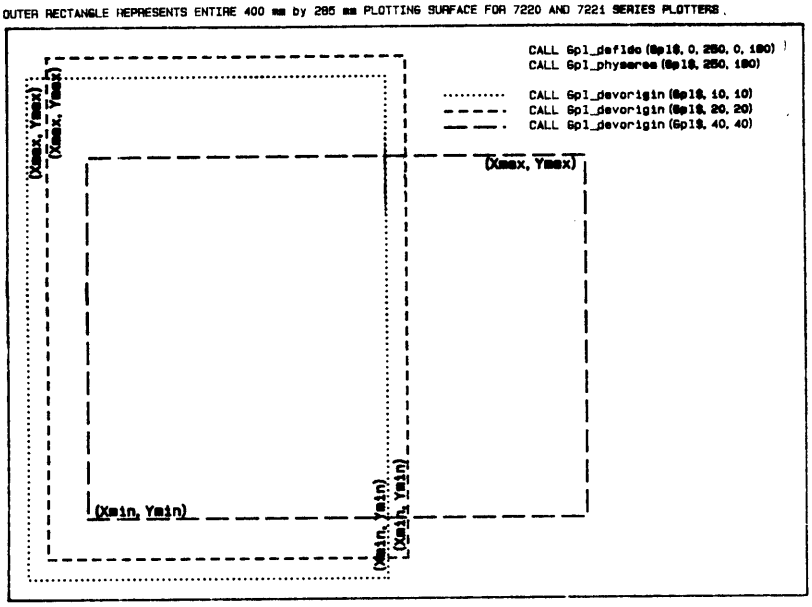
In general, binder holes should be placed to the left for 7220 and 7221 series plotters, on the top for 7225, 7475A, and 7550A plotters, and on the right for 7470A and 7440A plotters.

The defaults were chosen so that the plotting area attempts to cover the grid on 10x10 to the CM graph paper. You may use this paper (or the metric paper equivalent) for measuring the placement of items on a graph. The device origin may have to be adjusted slightly to compensate for individual plotter differences and/or the use of metric paper sizes.

LOGICAL DEVICE COORDINATES (LDC UNITS)

Your plotter has a rectangular plotting area. You are limited to plotting within this rectangle at all times.

To specify the logical units you merely specify the coordinate range to be applied to your rectangular plotting area. Four parameters are specified, the minimum and maximum X (horizontal direction) coordinates, and the minimum and maximum Y (vertical direction) coordinates. All plotting should have (X,Y) coordinates within these limits, and the limits define the "logical device coordinates", or "LDC units". See the following figure for a diagram of a logical device coordinate rectangle. Note that the point (Xmin,Ymin) corresponds to the lower left corner of your plot. The point (Xmax,Ymax) corresponds to the upper right corner. Increasing X values causes the pen to move RIGHT. Increasing Y values causes the pen to move UP.



GPL__DEFLDC

```
CALL Gpl_defldc(Gpl$,X_min,X_max,Y_min,Y_max)
```

Gpl__defldc defines logical device coordinates. The parameters X__min and X__max are the minimum and maximum X coordinates, and Y__min and Y__max are the minimum and maximum Y coordinates. X__min must be less that X__max, and Y__min must be less than Y__max.

Note: In all GPL calls in which a rectangle is specified, the boundaries are specified in the order X__min, X__max, Y__min, Y__max.

The default values are:

Xmin = 0
Xmax = 250
Ymin = 0
Ymax = 180

which is the same as would be set by the subprogram call:

```
CALL Gpl_defldc(Gpl$,0,250,0,180)
```

Helpful Hints

You should call Gpl__defldc soon after Gpl__plotteris and keep the logical device coordinate limits fixed throughout the entire plot.

The best "logical device coordinates" are in units that reflect the final physical size of your plot. For example, convenient physical size for charts and graphs are:

WIDTH	HEIGHT	FOR
250 mm	180 mm	Metric Unit Charts and Graphs
10 in	7 in	U.S. Unit Charts and Graphs

The placement of the plot in LDC coordinates is controlled by the other subprograms described in this chapter (Gpl__physarea, Gpl__devorigin, Gpl__physviewp, Gpl__physrotate). The main advantage to the Gpl__defldc subprogram is that it allows you to define any convenient coordinates; you are not restricted to using millimeters.

GPL__PHYSVIEWP

```
CALL Gpl_physviewp(Gpl$,X_min_pamm,X_max_pamm,Y_min_pamm,Y_max_pamm)
```

Specifies the placement of the LDC (logical device coordinate) limits in the physical area (Gpl__physarea) coordinate system. The default is a value which centers the LDC space in the physical area coordinate system with maximum area. The default is set whenever Gpl__physarea or Gpl__defldc is called. Thus, Gpl__physviewp should be called only after any calls to Gpl__defldc and Gpl__physarea.

This function is used whenever you wish to do one or more of the following:

(1) Stretch the plot in one direction, usually when you want to cover the entire physical area. Care must be used, as circles on the logical plotter will be deformed into ellipses.

(2) Move the plot from the center of the physical area.

(3) Change the size of the plot to something other than determined by the LDC and physical area coordinate limits.

X__min__pamm - Minimum x on physical area in millimeters

X__max__pamm - Maximum x on physical area in millimeters

Y__min__pamm - Minimum y on physical area in millimeters

Y__max__pamm - Maximum y on physical area in millimeters

GPL_PHYSROTATE

```
CALL Gpl_Physrotate(Gpl$,Rotation)
```

This function is used to control the rotation of a plot on all plotters. The plotting area specified by Gpl_physarea will be rotated according to the rotation parameter, which has 5 values: It is possible to rotate a plot from 0 to 270 degrees, in 90 degree increments (relative to the plotter axes).

0 No rotation on 7720, 7221, 7475A and 7550A plotters. 90 degree clockwise (CW) rotation on 7225, 7470A and HP ColorPro(7440A) plotters. Used most often for ISO A4 (English A or US-A) size plots that are oriented vertically.

1 90 degree counterclockwise (CCW) rotation on 7220, 7221, HP ColorPro(7440A), 7470A, and 7550A plotters. No rotation on the 7225 plotter. Used most often for ISO A4 (English A or US-A) size plots that are oriented horizontally.

2 (default) Software looks at Gpl_physarea parameters and attempts to fit on ISO A4 (English A or US-A) size sheet.

-1 90 degree clock-wise rotation (CW) on series HP ColorPro(7440A), 7470A, 7475A and 7550A.

-2 180 degree rotation on series HP ColorPro(7440A), 7470A, 7475A and 7550A.

In most cases it is best to let the rotation default to option 2, which lets GPL select the optimal plot rotation based on the plotter being used and the dimension of the plot being drawn.

SUMMARY

GPL has several subprograms to define the size and location of a plot. Default values exist for each of these specifications, allowing the programmer to either rely on these default values or to set up specifications of his own. Less complicated plots will require only the use of Gpl_defldc and Gpl_physarea. More advanced plots will also make use of the Gpl_devorigin, Gpl_physrotate and Gpl_physviewp subprograms to define the exact size and layout of the coordinate system.

DRAWING STRAIGHT LINES

SECTION 7

All GPL calls to change the position of the pen or to draw a line reference either a point or a distance on the logical coordinate system. A logical pen position is maintained, but does not always coincide with the physical position of the pen at any point in time. The pen physically moves only when there is some drawing to be done, whether text or straight lines.

The following subprograms are introduced in this chapter:

Gpl_move	Moves the logical pen to a location specified by an (X,Y) coordinate position.
Gpl_imove	Moves the logical pen an incremental distance in the X and Y directions.
Gpl_where	Returns the current position of the logical pen.
Gpl_frame	Draws a frame around the current window rectangle (windows are defined in Chapter 10).
Gpl_draw	Draws from the current logical pen position to the specified (X,Y) coordinate position.
Gpl_idraw	Draws an incremental distance in the X and Y directions from the current logical pen position.
Gpl_linestyle	Defines the style of line to be used for all drawing until another linestyle is specified. Various linestyles are available, including solid, dotted, and dashed.

CURRENT POSITION (CP)

All graphics are drawn relative to a "logical pen" called "Current Position", or "CP". The CP is an (X,Y) position that is used as the starting point of all straight lines and text. The term "CP" was chosen to conform with terms commonly used in computer graphics.

GPL_MOVE, GPL_IMOVE

```
CALL Gpl_move(Gpl$,X_wc,Y_wc)

CALL Gpl_imove(Gpl$,Dx_wc,Dy_wc)
```

To move the CP to some desired location, either Gpl_move or Gpl_imove may be used. Gpl_move moves the CP to some absolute position (X,Y). Gpl_imove moves the CP incrementally from the previous CP value.

For example,

```
CALL Gpl_move(Gpl$,10,20)
CALL Gpl_imove(Gpl$,2,0)
```

first moves the CP to (10,20) and then moves the CP to (12,20).

The plotter pen only moves when some actual drawing is to occur. Thus, the calls to Gpl_move and Gpl_imove in the above example do not result in any physical plotter pen movement unless they are followed by a "draw" or "text" call.

There is no default or initial value for the CP. Therefore, set the value with Gpl_move before any drawing is done.

GPL_WHERE

Sometimes, you need to know where the CP is located after drawing straight lines or text. To find out the CP, use Gpl_where.

For example,

```
CALL Gpl_where(Gpl$,X,Y)
```

results in the CP coordinates being stored in (X,Y).

STRAIGHT LINES

All plotting is done with straight lines or text. To get curves, symbols, and shaded regions, you build the figure out of a sequence of straight lines.

GPL_FRAME

```
CALL Gpl_frame(Gpl$)
```

Gpl_frame may be used to draw a box around the current plotting region. Four straight lines are used, and the CP is restored to its value before the call to Gpl_frame.

The meaning of the words "current plotting region" will be sharpened in the more advanced sections below. For the moment, the current plotting region will be defined to be the LDC (logical device coordinate) limits specified by Gpl_defldc.

GPL_DRAW, GPL_IDRAW

```
CALL Gpl_draw(Gpl$,X_wc,Y_wc)

Gpl_idraw(Gpl$,Dx_wc,Dy_wc)
```

Gpl_draw draws a straight line from the CP to the specified (X,Y) location. Gpl_idraw draws from the CP incrementally by the specified (Dx,Dy) incremental units. In both cases, the CP is updated to the end point of the line.

For example,

```
CALL Gpl_move(Gpl$,10,20)
CALL Gpl_draw(Gpl$,30,40)
CALL Gpl_idraw(Gpl$,0,10)
```

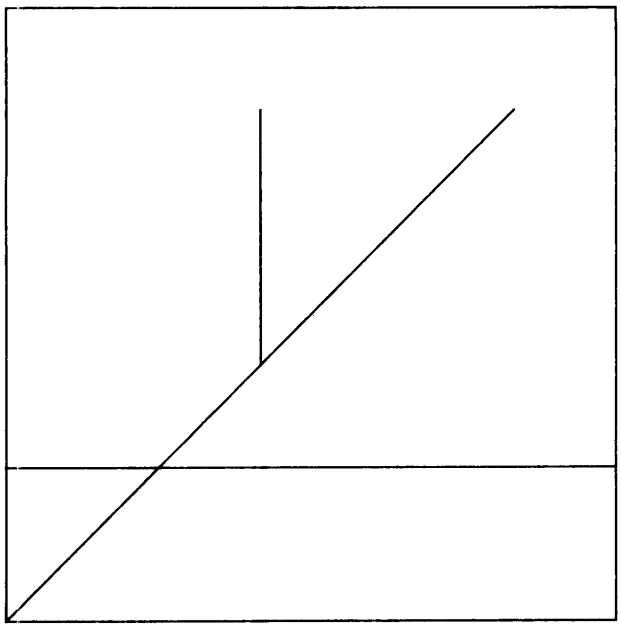
first draws a straight line from (10,20) to (30,40). Then another straight line is drawn from (30,40) to (30,50). After these calls, the CP is located at (30,50).

LIMITS OF CP

The CP can be placed anywhere, on or off your plotting area. If a line is drawn partly on and partly off the plotting area, the software will "clip" away the unplottable portions of the line.

Example:

```
1000 CALL Gpl_frame(Gpl$)
1010 CALL Gpl_move(Gpl$,0,)
1020 CALL Gpl_draw(Gpl$,100,100)
1030 CALL Gpl_imove(Gpl$,-50,0)
1040 CALL Gpl_idraw(Gpl$,0 -50)
1050 CALL Gpl_move(Gpl$,-20,30)
1060 CALL Gpl_draw(Gpl$,120,30)
```



GPL_LINESTYLE

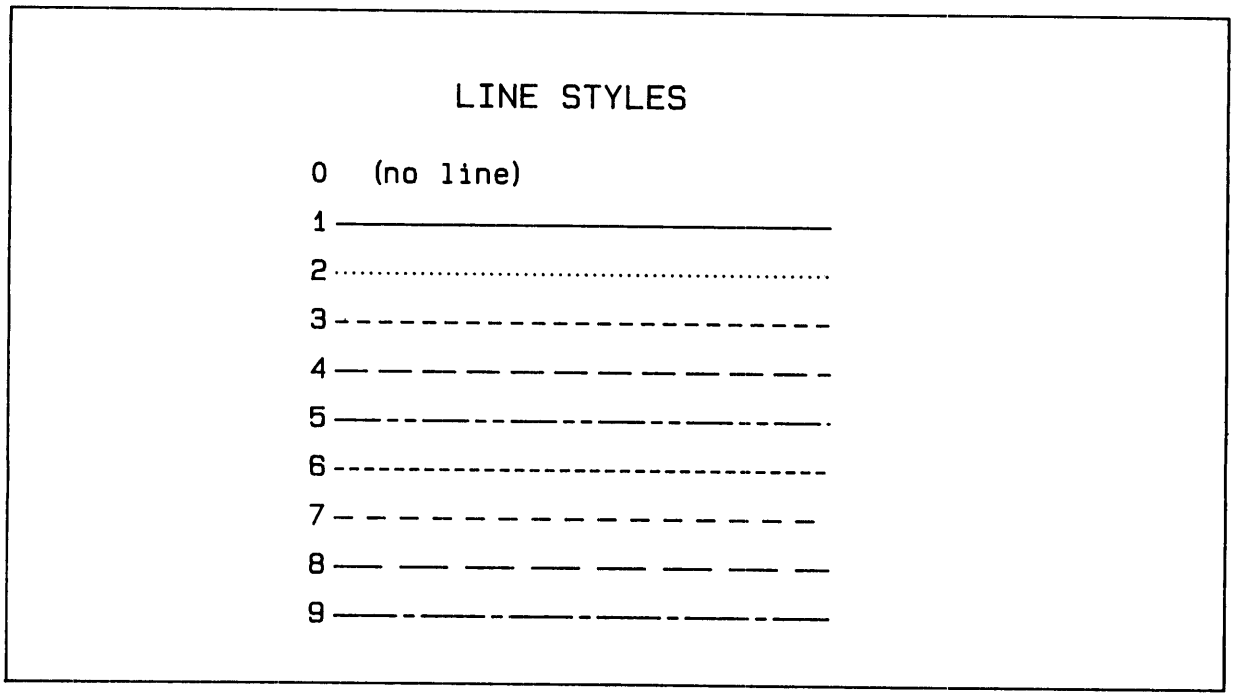
```
CALL Gpl_linestyle(Gpl$,Linestyle)
```

Up to this point we have been drawing straight lines with solid lines. Actually, there are 9 different linestyles available for producing visual differences, and these are numbered from 1 to 9. To select a particular linestyle, use Gpl_linestyle. For example,

```
CALL Gpl_linestyle(Gpl$,5)
```

selects linestyle 5.

The next figure shows the various linestyles. The linestyles are also shown on the confidence test plot that is created by program GPLCFT.



SUMMARY

In this chapter we have covered the subprograms necessary to draw lines and move the logical pen. All plotting to be done by GPL is done as a series of straight lines. All points on the plotting area are referenced by an (X,Y) coordinate position. If a point outside of the current plotting region is specified, actual drawing outside of the plotting limits will be clipped, although the CP is allowed to move outside of this area.

PEN CONTROL

SECTION 8

Plotters compatible with your computer systems have from one to eight pens. Some have variable pen velocity capability. GPL can detect the model number of each plotter and send appropriate signals to it. Programmed calls that cannot be understood by a plotter are ignored.

This chapter discusses the following subprogram calls:

Gpl_pen	Selects a pen on multiple-pen plotters, and can also return to pen to its stall.
Gpl_penspeed	Specifies the speed of the plotting in cm/sec.

GPL_PEN

```
CALL Gpl_pen(Gpl$,Pen_number)
```

The plotter can have from one to eight pens. A particular pen may be selected under program control with the Gpl_pen function.

Gpl_pen accepts integer values from 0 to 8. (The parameter Pen_number must be a REAL variable.) 0 puts the pen in the stall on 7220 and 7221 plotters. 1(default) selects pen 1,2 selects pen 2,etc.

The Gpl_pen call is ignored for single-pen plotters.

On the two-pen plotter, even values select the right pen; odd values select the left pen. On the four-pen plotter, Gpl_pen values 5 through 8 will select pens 1 through 4, respectively. On the six-pen plotter Gpl_pen values 7 and 8 will select pens 1 and 2 respectively. On the eight-pen plotters, Gpl_pen values 1 through 8 will select pens 1 through 8.

```
CALL Gpl_pen(Gpl$,7)
```

selects pen 7 on the eight-pen plotter, selects pen 1 on the six-pen plotter, selects pen 3 on the four-pen plotter, selects pen 1 on the two-pen plotter, and is ignored for the one-pen plotter. Parameter value 0 also suppresses text and straight lines even if the pen cannot be put away. GPL behaves as if the pen had been put away in this case.

GPL_PENSPEED

```
CALL Gpl_penspeed(Gpl$,Velocity_cps)
```

Many applications require that the pen velocity be changed, depending on the types of pens and paper being used in the plotter. For example, when plotting on overhead transparency acetate, it is advisable to slow down the pen velocity to 10cm/sec or less. This permits easier transfer of the ink to the acetate plotting surface, which is nonabsorbent.

The parameter to be supplied is a value which represents the pen velocity in cm/sec. The value should be in the range from 1 to 36 in order for the call to be valid. Any value outside that range causes the call to be ignored.

For example:

```
CALL Gpl_penspeed(Gpl$,10)
```

sets the penspeed to 10 cm/sec.

If your plotter does not have programmable velocity select, this call will be ignored.

NOTE

Some models of the HP 7225A have velocity select, and some do not. If your plotter does not have velocity select, use "7225A/3" or "7225A/4" for the model number. If your plotter does have velocity-select, use 7225A/3V" or "7225A/4V" for the model number in the Gpl_plotteris and in GPL%CF. If you specify the "V" when no "V" should have been specified, the plotter will not operate correctly. A similar problem exists for the HP 7225B series plotters. The four possible model numbers are "7225B/3", "7225B/3V", "7225B/4" and "7225B/4V".

NOTE

On the 7470A and 7475A plotters, there is a dead range of pen speed between 0.38 and 3.8 cm/sec. A Gpl_penspeed value of 1 yields a speed of 0.38 cm/sec. Values 2 and 3 yield 3.8 cm/sec. Any value above 3 yields the accurate speed.

DRAWING TEXT STRINGS

SECTION 9

Graphs and charts may be annotated with text strings using the text subprogram calls available in GPL. Both the height and width of the text cells may be specified, as well as the rotation (horizontal, vertical, etc.).

The subprogram calls introduced in this chapter are as follows:

Gpl_text	Draws a text string at the current pen position.
Gpl_tsize	Defines the height and width of the text character cell.
Gpl_lorg	Specifies the label origin of the text, or the origin point of the text string with respect to the current pen position.
Gpl_textrotate	Determines the rotation of text horizontal, or rotated in increments of 90 degrees.
Gpl_cset	Specifies the character set from which text characters will be drawn. Characters may be drawn from the plotter firmware or from GPL's character files.
Gpl_itextwidth	Returns the width of a specified character string in terms of logical device coordinates, used in computing text formats.

PLOTTER FIRMWARE CHARACTERS AND SOFTWARE-GENERATED CHARACTERS

PLOTTER FIRMWARE CHARACTERS — the firmware of each plotter contains the code to plot the US ASCII character set. The plotter-resident character sets can vary between plotters.

SOFTWARE-GENERATED CHARACTERS — GPL includes data files containing alternate character sets which can be specified to override the plotter firmware characters.

Of the software-generated fonts, some are of standard width, and some are of proportional width. ROMAN, STICK, and ITALIC are proportional fonts.

example of standard spacing
example of proportional spacing

This is a plottir
This is a plotting exar

GPL_TEXT

```
CALL Gpl_Text(Gpl$,Text$)
```

To draw text on your graphics plotter, simply pass the information as a string.

For example,

```
CALL Gpl_text(Gpl$,"HELLO!")
```

draws HELLO! on your plotter.

The entire 96 character US ASCII character set may be displayed. The default character set is the plotter firmware set. To override the default, or to draw other non-US ASCII characters, see "SOFTWARE-GENERATED CHARACTERS" in this chapter.

In addition, Roman Extension characters appearing on European keyboards may also be displayed (See "Drawing European Characters", below.)

The placement of the text depends on the position of the CP before the call to Gpl_text. Normally (lower left justification case), the CP is at the lower left corner of the string before the call. After the call, the CP is at the lower right corner of the string. This allows multiple calls to Gpl_text to be used to produce a long concatenated string.

These justification rules are modified by use of the Gpl_lorg subprogram described in a later section.

A call to Gpl_text is usually preceded by a call to Gpl_move to ensure proper text placement.

Use of CR and LF Control Characters with Gpl_text

ASCII control characters (0-31) are normally ignored in the Gpl_text call. The exceptions are Carriage Return (CR -ASCII 13) and Line Feed (LF -ASCII 10).

The CR and LF functions produce the same results as normally encountered in printing: CR returns the CP to the beginning of the string of text, and LF moves the CP down one line.

The CR and LF characters are not automatically inserted at the end of a string of text; you have to put them there. If omitted, the CP will be left at the end of the text, ready to concatenate additional text. These control characters should only be used at the end of a string.

Text Clipping

In the HP 7221 series plotter firmware, text is not properly clipped if the beginning of the text string is outside the current window. If your application requires clipped text, software-generated character sets (described later) will give proper text clipping.

Drawing European Characters with Gpl_text

The HP Computer System is capable of using a full 8 bit ASCII code (0-255). Within the system, ASCII codes from 128 to 159 are used for 32 additional control characters. The ASCII codes from 160 to 255 are used for additional 96 character printable character sets, including Roman Extensions for European applications.

The GPL software produces the Roman Extension characters when the appropriate code from 160 to 255 is included in the string. Katakana, Math symbols, and HP 250/260 special characters cannot be drawn.

NOTE

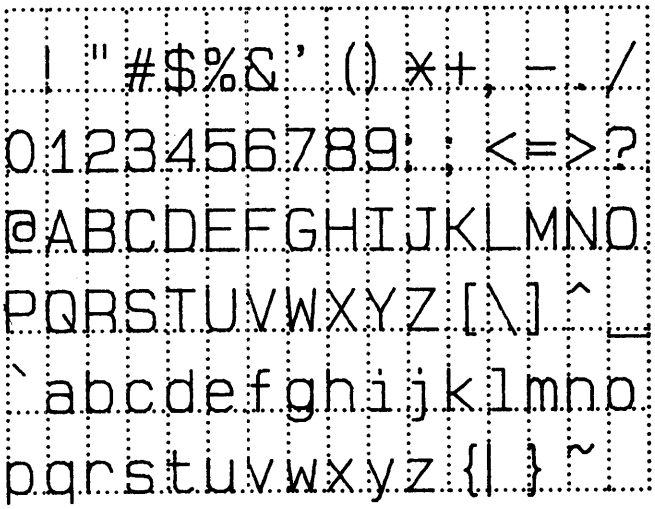
You should remove unused control characters before passing a string to Gpl_text.

GPL_TSIZE

```
CALL Gpl_tsize(Gpl$,Cell_height_ldc,Cell_width_dev_ptc)
```

The text character height and width is controlled by Gpl_tsize.

A character cell is the region around a character that determines the amount of white space between lines of text and between characters. In other words, the character cell height is the distance between the bases of two consecutive printed lines of text. The cell width is the distance between the left edges of the cells of two characters. See the next figure.



..... Indicates cell boundaries

Some software-generated fonts have varying cell widths. (See "SOFTWARE-GENERATED CHARACTERS" in this chapter.)

In a call to Gpl_tsize, the cell height is specified in LDC units. The cell width is specified as a percentage deviation from default, which is 0. 5 times the cell height.

Thus,

```
CALL Gpl_tsize(Gpl$,10,0)
```

results in a cell height of 10 LDC units and a cell width of 0. 5x10=5 LDC units.

To change the character cell width:

```
CALL Gpl_tsize(Gpl$,10,-20)
```

results again in a cell height of 10 units but a 20% narrower cell width of 4 LDC units.

The default height is nine units and the default width deviation is 0. With the default LDC coordinate system and mapping to the device, these defaults result in character cells that are 9 mm high.

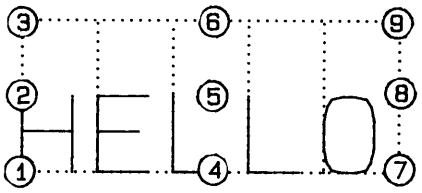
Specifying Character Size and Placement with Gpl_tsize

The Gpl_tsize call refers to character cells, not the characters themselves. On HP Graphics Plotters, the characters are placed against the lower left edge of the cells. The height of a capital letter in the plotter firmware is 1/2 the cell height. The height of a software proportional letter is 21/32 the cell height. Most capital letter widths are 2/3 the cell width. Lower case firmware letters have heights which depend on the particular plotter model number.

GPL_LORG

```
CALL Gpl_lorg(Gpl$,Label_origin)
```

Gpl_lorg (standing for Label Origin) determines the positioning of the output in relation to a reference point. Values 1 to 9 indicate different orientations in reference to the current pen (CP) position.



Remember that the parameter refers to a point on the string of CELLS, not the string of characters.

NOTE

When the font in use is a SOFTWARE-GENERATED PROPORTIONAL font, like Serif or Decorative, Gpl_lorg must be 1.

Slight adjustments may have to be made in the CP when you wish to justify on the characters.

```
CALL Gpl_lorg(Gpl$,1)
```

is the default. The CP before the call to Gpl_text is at the lower left corner of a line of text. The justification is "LEFT" horizontally and "BOTTOM" vertically.

```
CALL Gpl_lorg(Gpl$,4)
```

is commonly used to center text horizontally. The CP before a call to Gpl_text is in the center of a line of text horizontally and on the base line of the characters. The justification is "CENTER" horizontally and "BOTTOM" vertically.

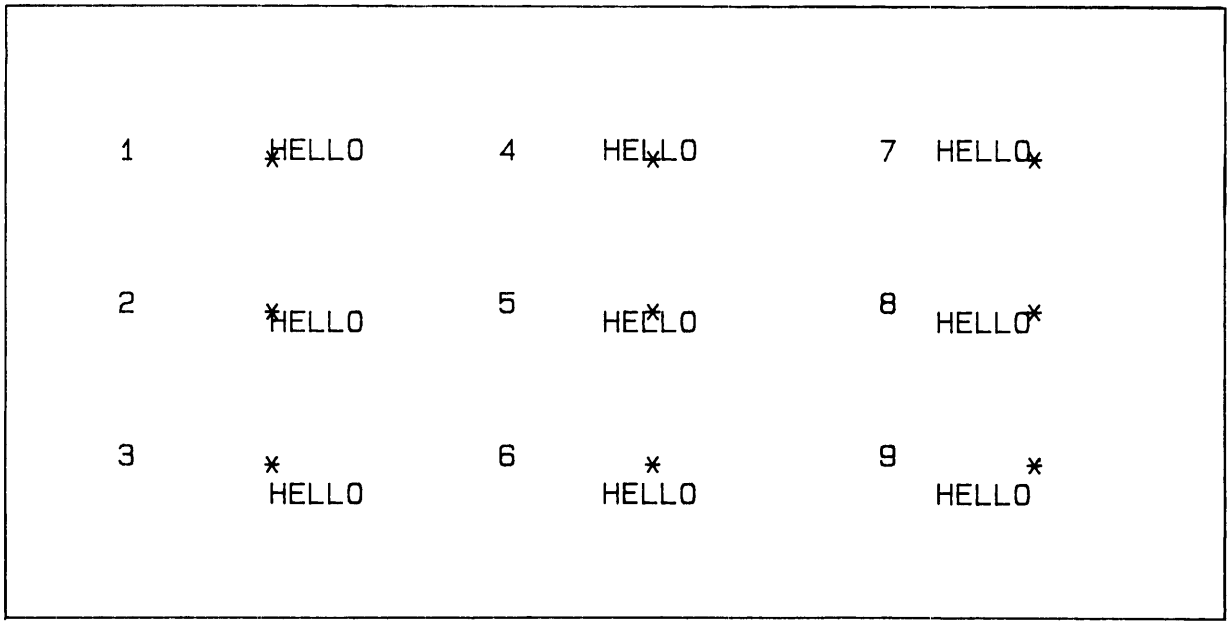
```
CALL Gpl_lorg(Gpl$,7)
```

means to "RIGHT" justify horizontally and "BOTTOM" justify vertically.

```
CALL Gpl_lorg(Gpl$,2)
```

means to "LEFT" justify horizontally and "CENTER" justify vertically.

The meanings of the Gpl_lorg parameter are summarized in the next figure, which shows the CP before calling `Gpl_text(Gpl$,"HELLO")`. In all cases, the CP after the call is at position 7.



GPL_TEXTROTATE

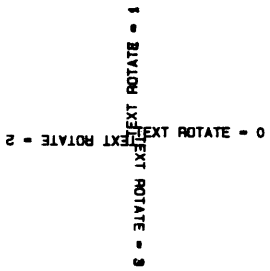
```
CALL Gpl_textrotate(Gpl$,Rotation)
```

For most business applications, it is recommended by many graphics designers that all text be printed in the normal horizontal manner. For example, the labels for the Y axis (title and tic mark labels) should be printed horizontally. A good place to put the Y axis title is just above the Y axis, as was done in Chapter 3 example, GPLEO1.

However, if your application requires text rotation, you may use the Gpl_textrotate function. The parameter supplied is an integer with the following 4 values:

0 - (default) normal horizontal text
1 - Rotate text 90 degree counter clockwise (CCW) from horizontal
2 - Rotate text 180 degrees CCW from horizontal
3 - Rotate text 270 degrees CCW from horizontal

The text rotation chosen will be in effect until another call to Gpl_textrotate is made. See next figure.



GPL_ITEXTWIDTH

```
CALL Gpl_itextwidth(Gpl$,Text$,Width)
```

Function Gpl_itextwidth stands for Inquire Text Width. It determines the width of a proportional or non-proportional string for your use in computing formats.

```
CALL Gpl_itextwidth(Gpl$,"ABC",Width)
```

The width of Text$ is returned by Width, and is given in LDC units. This example will center a string horizontally about the point 50,70.

```
CALL Gpl_lorg(Gpl$,1)
CALL Gpl_itextwidth(Gpl$,"ABC",Width)
```

```
CALL Gpl_move(Gpl$,50-Width/2,70)
CALL Gpl_text(Gpl$,"ABC")
```

Gpl_itextwidth calls Gpl_transmit which is not present in Gpl_251 or Gpl_252; therefore, always load one of the files Gpl_253 through Gpl_257 with file Gpl_258. Remember that the width of a particular string depends on the last value specified in the Gpl_tsize statement (which specifies the character heights and widths).

SOFTWARE-GENERATED CHARACTERS

Gpl_cset

```
CALL Gpl_cset(Gpl$,Char_set,Character_file$)
```

The Roman Extension characters used with European keyboards (and all software fonts described in this manual) are produced from software "digitizations." The standard "plotter fonts" (used by default) are produced from firmware in the plotter. Plotter firmware fonts have also been digitized and stored in GPL files, but files GPL271 through GPL276 are not normally used because GPL produces these characters by default.

Font Name	US ASCII File name	Roman Extensions File Name
Serif Roman Light	GPLRL0	GPLRL1
Serif Roman Medium	GPLCR0	GPLCR1
Serif Roman Bold	GPLTR0	GPLTR1
Serif Italic Light	GPLLI0	GPLLI1
Serif Italic Medium	GPLCI0	GPLCI1
Serif Italic Bold	GPLTI0	GPLTI1
Stick Light	GPLSR0	GPLSR1
Stick Medium	GPLDR0	GPLDR1
Sans Serif Italic Light	GPLLJ0	GPLLJ1
Sans Serif Italic Medium	GPLMJ0	GPLMJ1
Gothic English	GPLGE0	GPLGE1
7220A/S, 7221A/B/S, 7225A/B	GPL271	GPL272
7220C/T, 7221C/T Plotters	GPL273	GPL274
7470A/, HP ColorPro(7440A) plotters	GPL275	GPL276
7475A/, 7550A Plotters	GPL275	GPL276

Depending on the plotter in use, one of GPL272, 274, or 276 is used if Gpl_cset is not called.

Use Gpl_cset to specify a software character set or to return to the hardware default.

```
CALL Gpl_cset(Gpl$,Char_set,Character_file$)
```

Char_set - Specifies whether to use the lower 96 characters (US ASCII) or the upper 96 characters (Roman Extensions) with this file.

0 - Lower (ASCII 32-127)
1 - Upper (ASCII 160-255)

Character file$ - String containing the name of a character file, including volume specifier if necessary. See the tables above for software character file names.

A null string may be supplied if Char_set=0. This means to use the default hardware character set for the plotter.

A typical usage is to use the "best" plotter characters (as produced by default on the HP 7220C, 7220T, 7221C, and 7221T) on all plotters. These best characters look better than the characters produced on other plotters. To do this, make calls of the form:

```
IF Model$[5;1]<>"C" AND Model$[5;1]<>"T" THEN
CALL Gpl_cset(Gpl$,0,"GPL273")
CALL Gpl_cset(Gpl$,1,"GPL274")
END IF
```

The first line calls the software characters only if the plotter has the old character sets. This allows usage of the hardware generated characters, which are drawn much faster, whenever possible.

Software character sets typically take 2 or 3 times longer to plot than firmware generated characters.

SUMMARY

The firmware of each plotter contains code to create the default character font. The default plotter firmware fonts can be overridden by a software-generated font in a GPL file. Default values exist for all text definition subprograms. Once these definitions are changed, however, the new values stay in effect until another call to the subprogram is made. This means that once a text size and rotation are specified, that size and rotation will be in effect until a new text size and rotation are specified.

USING SOFTWARE CHARACTERS

SECTION 10

This chapter supplements Chapter 9, and should be read by the programmer who plans to use software-generated characters with HP plotters.

WIDTH OF PLOTTER FIRMWARE FONTS

Plotter firmware characters are of fixed width -- two-thirds the width of the cell and left-justified within the cell.

WIDTH OF SOFTWARE-GENERATED FONTS

Some software characters are of fixed width like plotter firmware fonts, and some are proportional. The proportional software characters are centered within their cells. When underlining or drawing boxes around characters, plot the lines with respect to cell edges.

UNDERLINING PLOTTER FIRMWARE CHARACTERS

To underline the string "ABCDE", use this algorithm:

```
CALL Gpl_cset(Gpl$,0,"GPL273")
CALL Gpl_cset(Gpl$,1,"GPL274")
  •
  •
  •
Text_height=7
CALL Gpl_tsize(Gpl$,Text_height,0)
CALL Gpl_move(Gpl$,X,Y)
CALL Gpl_text(Gpl$,"ABCDE")
CALL Gpl_move(Gpl$,X,Y-Text_height*3/32)
CALL Gpl_idraw(Gpl$,(5-1/3)*Text_height/2,0)
```

The factor in Gpl_idraw is designed to prevent underline overhang at the end of the string. Each character occupies the left two thirds of its cell. The last one-third of the last cell in the string is vacant, and should not be underlined.

The next algorithm performs the same function, but is generalized to allow variable values, including a variable width deviation of the characters in the string. This next algorithm calls for all characters to be 17 percent narrower (Width_dev_ptc = -17) than normal.

```
CALL Gpl_cset(Gpl$,0,"GPL273")
CALL Gpl_cset(Gpl$,1,"GPL274")
  •
  •
  •
```

```
Text_height=7
Width_dev_ptc=-17
CALL Gpl_tsize(Gpl$,Text_height,Width_dev_ptc)
CALL Gpl_move(Gpl$,X,Y)
Text$="ABCDE"
L=LEN(Text$)
CALL Gpl_text(Gpl$,Text$,)
CALL Gpl_itextwidth(Gpl$,Text$,Width)
CALL Gpl_move(Gpl$,X,Y-Text_height*3/32)
CALL Gpl_idraw(Gpl$,(L-1/3)*Width/L,0)
```

UNDERLINING PROPORTIONAL SOFTWARE CHARACTERS

Proportional characters are centered within their cells, so underlining is simple.

```
CALL Gpl_cset(Gpl$,0,"GPLTR0")
CALL Gpl_cset(Gpl$,1,GPLTR1")
  •
  •
  •
CALL Gpl_tsize(Gpl$,T_height,Width_dev_pct)
CALL Gpl_lorg(Gpl$,1)
CALL Gpl_move(Gpl$,X,Y)
CALL Gpl_text(Gpl$,Text$)
CALL Gpl_itextwidth(Gpl$,Text$,Width)
CALL Gpl_move(Gpl$,X,Y-T_height*7/32)
CALL Gpl_idraw(Gpl$,Width,0)
CALL Gpl_idraw(Gpl$,0,-T_height/32)
CALL Gpl_idraw(Gpl$,-Width,0)
CALL Gpl_idraw(Gpl$,0,T_height/32)
```

Similar algorithms can be developed to draw boxes around one or more lines of text.

CREATING A CHARACTER SET FILE

Example program GPLE06, included with the GPL library and loaded whenever GPLLOD is used to load GPL, gives the algorithm and instructions for packing the data into strings. The PACK DROM must be configured to create character set files.

CHARACTER FILE FORMAT

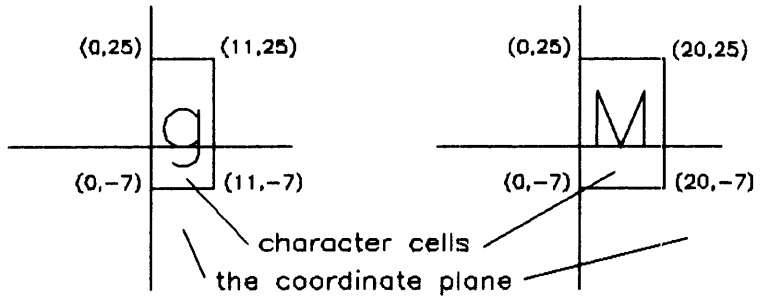
Each character set file contains exactly 97 records. When called, Gpl_cset checks that the file contains 97 records.

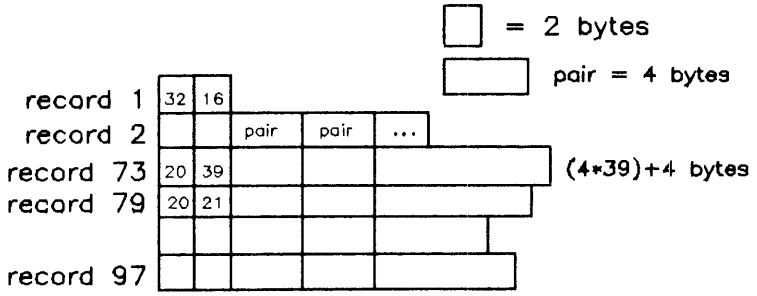
The first record contains two integers packed into a four-character string. Both integers are in the range 1 to 998. The first specifies cell height normalization, which is the height of the symbol cells in this file, and corresponds to the Cell_height_ldc parameter of Gpl_tsize. The second is cell width normalization,

which is usually one half the height normalization. When called, Gpl_cset checks that the first record contains these two integers packed into a string.

Records 2 through 97 each contain the digitization data for one of the 96 different characters. The digitization of a character, contained in one record, consists of coordinate pairs packed into a string. Each pair occupies four bytes (four characters). The record is organized as follows:

Cell width (in LDC units)
Number (N) of coordinates pairs which follow
First X coordinate
First Y coordinate
Second X coordinate
Second Y coordinate
•
•
•
Nth X coordinate
Nth Y coordinate





The number of bytes in each string (or record) is (4*N)+4. Determine which string has the greatest number of bytes, and let Max_bytes equal that length. When creating the file in which to keep this data, let it be 97 records of Max_bytes+4 bytes per record.

```
CREATE "SYMFIL", 97, Max_bytes+4
```

Your computer's file system uses the additional four bytes. The X and Y values of the pairs are in the range -998 to +998. The special pair (999,999) means to lift the pen and move it to the point specified by the next pair. GPL is in use, the pen moves to the point specified by the first X,Y coordinate, draws to the second coordinate, then to the next, and continues drawing until the (999,999) pair. The pen raises, moves

to the next point, lowers, and draws until the next (999,999) point. The last pair in the record does not have to be (999,999); the pen raises automatically.

VIEWING TRANSFORMATIONS

SECTION 11

In this chapter, we embark on one of the hardest, and most useful, concepts in GPL: viewing transformations.

The following subprograms are discussed in this chapter:

Gpl_window	Defines the 'World Coordinates' which will apply to the viewport area.
Gpl_viewport	Specifies that subset of the LDC area which will serve as a viewport. The world coordinates apply to this area.
Gpl_useldc	Defines the current plotting region to be the LDC area and the current units to be LDC units.
Gpl_usewc	Defines the current plotting region to be the viewport and the current units to be WC units.

The viewing transformation allows you to map any set of rectangular coordinates onto your logical plotter. This is useful when plotting units such as "dollars per month" or percentages on a chart or graph.

You will find it worth your time to be patient and learn the concepts of WINDOW and VIEWPORT, which collectively make up the viewing transformation.

DEFINITIONS

Some useful concepts will be defined in this section. The terminology chosen is not very mnemonic, but it coincides with state-of-the-art terminology as used by computer graphics experts. In particular, "world coordinates", "window", and "viewport" are used frequently in computer graphics standardization efforts.

World Coordinates

"World Coordinates" are the particular coordinates that are natural for your data, such as MILLIONS OF DOLLARS vs YEAR or UNIT SALES plotted as a function of WEEKS.

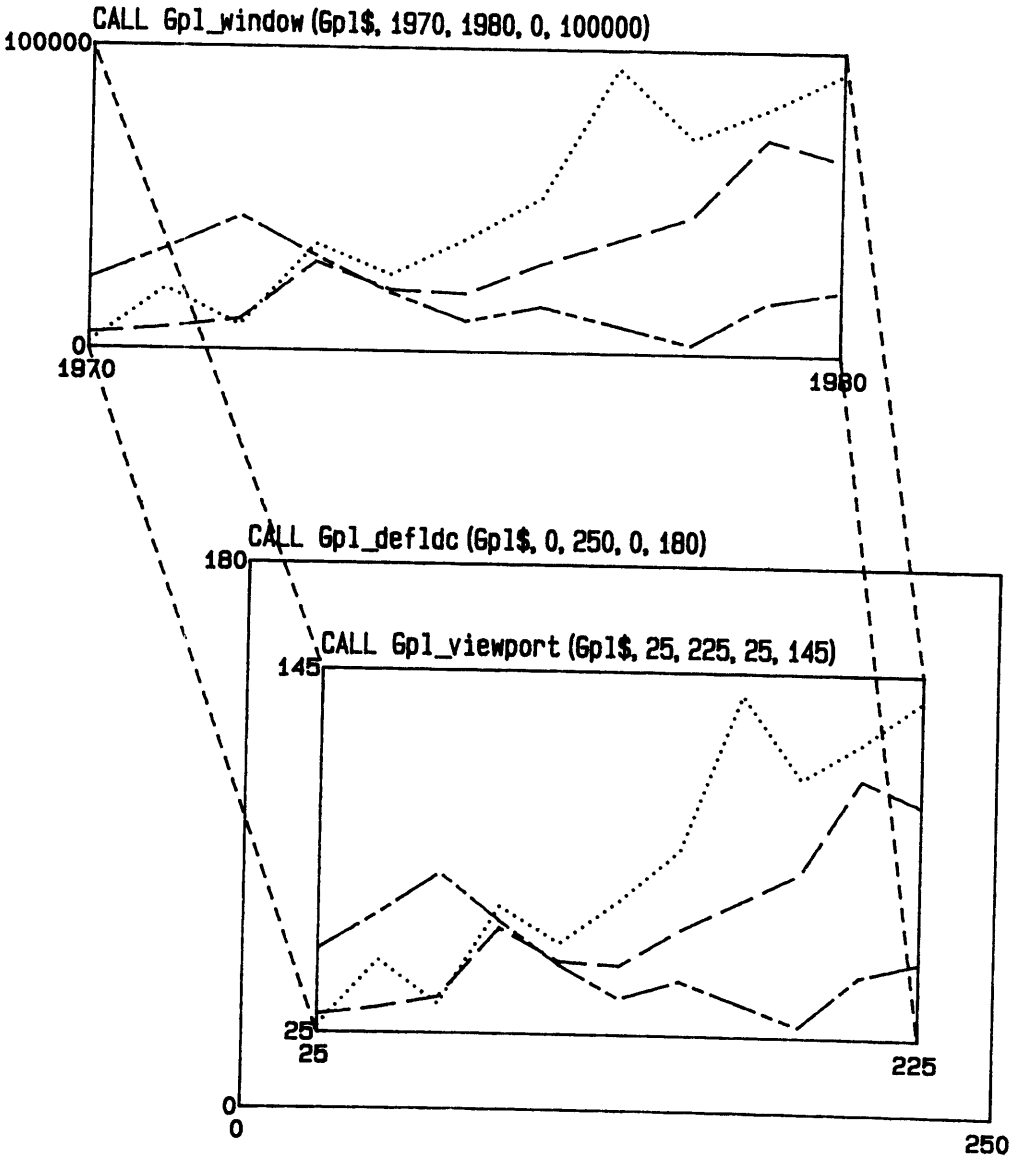
Logical Device Coordinates

To refresh your memory, "Logical Device Coordinates" are the coordinates to represent your (device independent) plotter. This coordinate system remains fixed throughout your problem, and allows you to place text annotation along with plotted curves in one convenient coordinate system.

"Logical Device Coordinates" are a generalization of "normalized device coordinates" or "graphical display units" that are used in typical graphics subroutine liberals. They allow you to use any convenient coordinate system that you desire as the model for the plotting surface.

Viewing Transformation

A "Viewing Transformation" is a mapping between your problem coordinates (world coordinates) and the plotter (logical device coordinates). It is specified by a "window" rectangle in world coordinates that is mapped onto a "viewport" rectangle in logical device coordinates.



Window

All graphic lines and text within the window rectangle will be mapped onto the viewport rectangle in plotter coordinates. When a window is specified, using Gpl_window, all MOVE and DRAW functions will be in the world coordinates defined by the window.

Viewport

The "viewport" rectangle may be viewed as a subset of the logical device coordinate area and is defined in LDC units. All graphic lines and text within the window rectangle will be mapped onto the viewport rectangle. The lower left corner of the window maps to the lower left corner of the viewport, the lower right corner of the window maps to the right corner of the viewport, and so on.

The window and viewport may each be independently specified. The default window and viewport are the logical device coordinate system, and the defaults are used whenever Gpl_defldc is called.

Clipping (or Scissoring)

All graphic lines and text with world coordinates outside the window are "clipped" or "scissored". That is, plotting only occurs within the window, just as if a pair of scissors were used to cut out a picture defined by the window.

GPL_WINDOW

```
CALL Gpl_window(Gpl$,Xmin_ldc,Xmax_ldc,Ymin_ldc,Ymax_ldc)
```

The subprogram Gpl_window is used to define the window rectangle. Like all rectangles in GPL, it is defined by the following four parameters in sequence:

Xmin_wc - Minimum X coordinate
Xmax_wc - Maximum X coordinate
Ymin_wc - Minimum Y coordinate
Ymax_wc - Maximum Y coordinate

The default is the logical device coordinate space, and the default is reset each time Gpl_defldc is called. Thus, Gpl_window should be called after calling Gpl_defldc.

Suppose, for example, that you want the X units range to be from the years 1970 to 1980 and the Y units range to be from 10,000 to 1000,000 dollars. Then use

```
CALL Gpl_window(Gpl$,1970,1980,10000,100000)
```

GPL_VIEWPORT

```
CALL Gpl_viewport(Gpl$,Xmin_ldc,Xmax_ldc,Ymin_ldc,Ymax_ldc)
```

The subprogram Gpl_viewport is used to define the viewport rectangle. Again, four rectangle parameters are specified in the following sequence:

Xmin_ldc - Minimum logical device coordinate
Xmax_ldc - Maximum logical device coordinate
Ymin_ldc - Minimum logical device coordinate
Ymax_ldc - Maximum logical device coordinate

For example,

```
CALL Gpl_viewport(Gpl$,30,40,50,60)
```

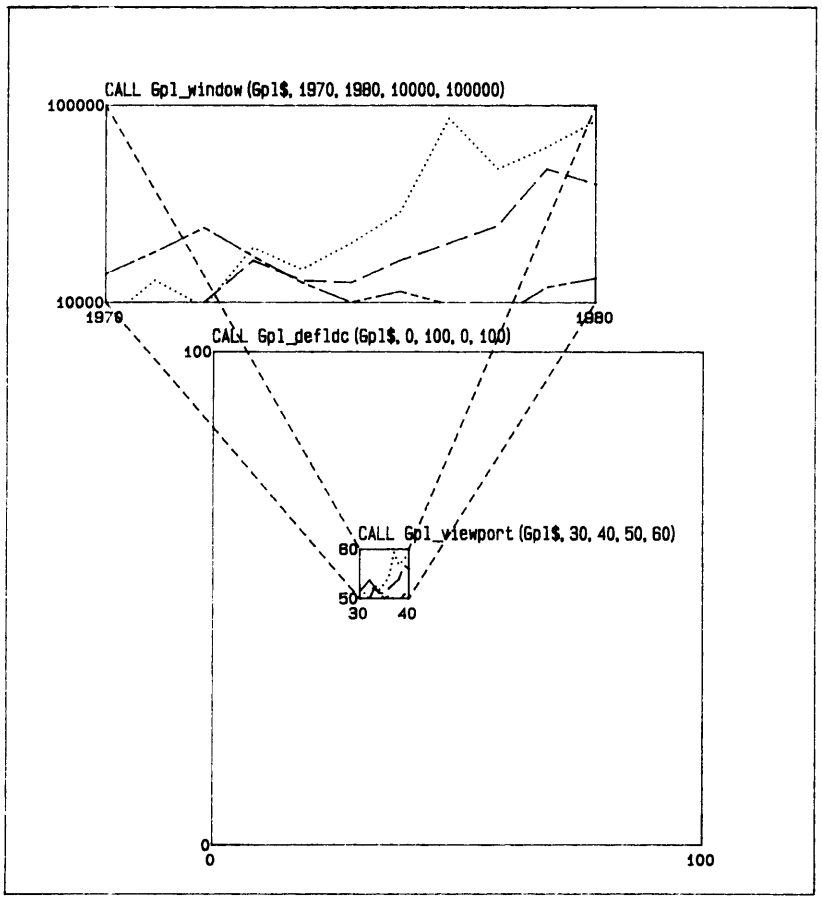
will place the graphical data in a rectangle with

Xmin_ldc=30
Xmax_ldc=40
Ymin_ldc=50
Ymax_ldc=60

on the logical plotter.

The default is the logical device coordinate space, and the default is reset each time Gpl_defldc is called. Thus, Gpl_viewport should be called after calling Gpl_defldc.

Normally, Gpl_window and Gpl_viewport are called paired together to completely define a viewing transformation. You can, if you choose, call them individually and in any order.



GPL_USELDC

```
CALL Gpl_useldc(Gpl$)
```

This call allows the current viewing transformation to be temporarily suspended and to make the window and viewport temporarily equal to the logical device coordinates. The normal viewing transformation is restored when Gpl_usewc, described in the next section, is called.

The CP is not converted to a meaningful value. Thus Gpl_move should be used before drawing a line (Gpl_draw, etc.) or text (Gpl_text). The call syntax is simply

```
CALL Gpl_useldc(Gpl$)
```

GPL_USEWC

```
CALL Gpl_usewc(Gpl$)
```

This call restores the viewing transformation to the value before a call to Gpl_useldc. This mode is also restored by a call to Gpl_window, Gpl_viewport, or Gpl_defldc.

The pair of calls, Gpl_useldc and Gpl_usewc, are useful when plotting several sets of data on a graph, and you wish to annotate outside the graph between plots. They are also useful when you wish to do rectangle shading or to create plotting symbols, as shown in later chapters.

Again, the CP is not transformed to a meaningful value. The call syntax is

```
CALL Gpl_usewc(Gpl$)
```

SUMMARY

In this chapter, you learned how to use viewing transformations. These allow you to map your problem coordinates (world coordinates) onto your logical plotter coordinates. If no viewport or world coordinates are defined, the viewport will be equal to the logical device coordinate space and the world coordinates will be equal to the logical device coordinates.

It is recommended that you use the viewing transformation whenever you change the coordinate system within an application.

It is also recommended that you not change the following parameters within a single plot (after defining them at the start):

Gpl_defldc parameters
Gpl_physarea parameters
Gpl_devorigin parameters
Gpl_physrotate parameters

This preserves device independence of the plots, and the code is easier to maintain.

PICTURE FILES

SECTION 12

GPL allows device dependent picture files to be created for future use. These files are useful when a plotter is unavailable or when multiple copies of the same plot are to be generated, or when plotters need to be used with maximum efficiency.

The following subprograms and utilities are introduced in this chapter:

Gpl_fileis	Sets up a data file into which plotter protocol and messages will be written.
Gpl_message	Writes a message into the data file defined using Gpl_fileis. Optionally, allows a GPLFIL program pause after displaying the message.
GPLFIL	Utility program used to recreate a plot from a GPL data file which was created using Gpl_fileis. The device used to recreate such a plot must be identical to the device originally used to draw the plot.

GPL picture files contain device protocol for a particular plotter. The first record contains the model number of the plotter to protect GPLFIL from sending the picture to the wrong device.

GPL_FILEIS

Subprogram Gpl_fileis is used to create a picture file. Either a new picture file may be created or a previously created file may be replaced with a new picture.

The call syntax is

```
CALL Gpl_fileis(Gpl$,Pic_file$,Protect$,Option,Error)
```

with the following parameter meanings:

Pic_file$ - Name of picture file to be created or replaced. The GPL software CREATE's the file **and** any scratch files that it needs. A file of 500 records times 80 data bytes is first created. This is extended by 500 records whenever the end of the file is encountered when writing into it. When Gpl_terminate is called, the file is reduced to the actual number of records used. Gpl_abortplot causes the file to have incorrect or incomplete data, and the file should be purged by the application programmer.

Protect$ - Protect code for this file. Optional. If no protect is used, just supply a null string.

Option - Used to decide whether to purge a previously existing file with the same name and protect code.

-1 - PURGE any previously existing file with the same name and protect code. Then CREATE the new picture file. Essentially a replace option.

0 - CREATE a picture file only if a previous does not exist with the same name. This prevents accidental destruction of a previously created file.

Error - Error return parameter. If no errors are encountered, then 0 is returned. To display an error message for non-zero parameter values, use DISP FNGpl_errm$(Error).

Subprogram Gpl_fileis is contained in LOAD SUB file "GPL252", which also contains Gpl_plotteris. If the initialization call to Gpl_plotteris is successful (Error=0), then the next GPL call should be Gpl_fileis. This prevents plotter initialization data from being lost; it is still stored in internal buffers after the call to Gpl_plotteris.

GPL_MESSAGE

Operator messages, such as "CHANGE PENS" or "CHANGE PAPER", may be embedded in picture files through the use of a call to Gpl_message. The syntax is

```
CALL Gpl_message(Gpl$,Message$,Wait)
```

The parameters are:

Message$ - Message to be stored in file.

Wait - Parameter used to specify a pause after the message is displayed.

0 - No pause
1 - Pause and prompt for a continue.

When your application program is run , the operator message will appear with an LDISP command if you specified a Message_address of 8 when calling Gpl_plotteris. In addition, a pause prompt will be displayed of the form "PRESS ENTER TO CONTINUE".

If the Message_address in Gpl_plotteris is 9, then neither the message nor a pause prompt will appear when your application program is run. When GPLFIL is used to display the picture, however, the message and/or pause prompt will appear.

GPLFIL UTILITY PROGRAM

```
RUN "GPLFIL [device-spec]"
```

Utility program GPLFIL may be used to display pictures stored in files on a plotter. Just run the program, and it will prompt you for the name of the file.

PLOTTING AN ACETATE

SECTION 13

To produce multi-color overhead transparencies, order the appropriate HP Overhead Transparency Kit for your plotter. Note that all plotters with 'Grit Wheel Drive' require a special paper-backed acetate sheet that is different from the acetate used by other plotters.

When plotting on acetate, the programmer must design certain precautions into the software.

OVERHEAD TRANSPARENCY KIT

The HP Overhead Transparency Kits, which are different according to the plotter you own, contain plotter pens and sheets of acetate. When the kit supplies are used up, individual items can be ordered separately.

SPECIAL CONSIDERATIONS

Ink Transfer and Drying

The plotting occurs directly on acetate, which is non-porous and does not absorb ink. This characteristic is different from paper, which readily absorbs ink. Special pens, containing a fast drying ink, must be used to plot on this acetate.

The paper pen normally in a plotter must be replaced with the special acetate pens whenever plotting on acetate. The pens should remain capped whenever not being used for a long time to prevent the tip from drying up. If a tip does become dry, it may often be restored by using the special solvent supplied with the Overhead Transparency Kit.

The acetate must be carefully handled to avoid transferring oils from your hand to the acetate prior to plotting. After plotting, care must be used to avoid touching the surface for about 10 minutes while the ink dries. In addition to these handling precautions, the programmer should perform some other operations described in the programming section below.

Viewing Area

Many overhead acetate slide projectors have a useful viewing area that is limited to a square about 8 to 9 inches (200 to 225 mm) to a side. Thus the physical plotting area must have to be reduced to fit into this area. In addition the origin of the plot may have to be adjusted to center on the acetate. Use Gpl_physarea and Gpl_devorigin to perform these adjustments.

The film and paper provided with the Transparency Plotter Film Kits and Paper and Pen Kits is supplied in both English/US 'A'size (8.5" X 11") and Metric 'A4'size (210 X 297 mm).

PROGRAMMING GUIDELINES

Viewing Area

You can use Gpl_physarea to cut down the size of your plot to fit the limits of the overhead transparency projector. For example, for a plot that is normally 250 mm wide by 180 mm high, you would use:

```
CALL Gpl_physarea(Gpl$,250,180)
```

To cut this down to say 200 mm wide (80% or original size), just use:

```
CALL Gpl_physarea(Gpl$,200,144)
```

Centering

After getting the proper viewing area, adjust the Gpl_devorigin parameters to give a centered plot on the acetate.

Pen Velocity

The pen velocity needs to be slowed down for maximum ink transfer. A velocity of 10 cm/sec seems to work well for most situations:

```
CALL Gpl_penspeed(Gpl$,10)
```

Multiple Passes

After drawing the entire plot, you may want to repeat it again at least once. The lines will have much better definition.

Avoiding Ink Run

You should be careful when lines of different color cross or when shaded regions of different color touch. Ink run can occur because of the slow drying time.

Suggestions:

1. For shaded regions, separate them slightly so that the regions do not touch.

2. For lines that cross, adjust the order of plotting so that one line has a chance to dry somewhat before a crossing line of a different color intersects. One way to do this is to plot all pen 1 lines and text. Then plot all pen 2 lines and text, etc.

PROGRAM DESIGN TIPS

SECTION 14

The GPL library imposes a natural structure on an application program.

SIMPLE APPLICATIONS

The graphics portions of simple applications should proceed in the following sequence:

```
DIM Gpl$[1200]

        REQUEST device

        LOAD "GPL252" library

                DEL 5001,9999
                LOAD SUB "GPL252",5001,1
```

Initialize plotter
Gpl_plotteris

```
        LOAD "GPL253" library

                DEL 5001,9999
                LOAD SUB "GPL253",5001,1
```

Define LDC space

Gpl_defldc

Define physical area and origin
Gpl_physarea
Gpl_devorigin

```
        LOAD "GPL254" library

                DEL 5001,9999
                LOAD SUB "GPL254", 5001,1
```

Define the window and viewport

Gpl_window
Gpl_viewport

Gpl_useldc
Gpl_usewc

Draw text and straight lines

Gpl_move	Gpl_imove	
Gpl_draw	Gpl_idraw	
Gpl_text		
Gpl_pen	Gpl_linestyle	Gpl_penspeed
Gpl_tsize	Gpl_textrotate	Gpl_lorg

Repeat the previous two steps as often as needed

Abort plot, if needed

Gpl_abortplot

Clear the plot (should always be called)

Gpl_clear

Go back to any step after plotter initialization if additional plots are desired

```
LOAD "GPL257" library

        DEL 5001,9999
        LOAD SUB "GPL257",5001,1
```

Terminate plotting

Gpl_terminate

```
DELETE GPL library

        DEL 5001,9999

RELEASE device
```

COMPLEX APPLICATIONS

The graphics portions of more complex or larger applications should proceed in the following sequence:

```
DIM Gpl$[1200]

LOAD "GPL251" library

        DEL 5001,9999
        LOAD SUB "GPL251", 5001,1
```

Read configuration file

Gpl_config

Select device

```
REQUEST device
```

Check device

Gpl_devident

```
LOAD "GPL252" library

        DEL 5001,9999
        LOAD SUB "GPL252", 5001,1
```

Initialize plotter

Gpl_plotteris

Initialize picture file (if used)

Gpl_fileis

```
LOAD "GPL253", library

        DEL 5001,999
        LOAD SUB "GPL253", 5001,1
```

Define LDC space

Gpl_defldc

Define physical area and origin

Gpl_physarea
Gpl_devorigin
Gpl_physrotate
Gpl_physviewp

Pick character set options

Gpl_cset

```
LOAD "GPL254" library

        DEL 5001,9999
        LOAD SUB "GPL254", 5001,1
```

Define the window and viewport

Gpl_window
Gpl_viewport
Gpl_useldc
Gpl_usewc

Draw text and straight lines

Gpl_move
Gpl_draw
Gpl_text
Gpl_imove
Gpl_idraw

Gpl_pen	Gpl_linestyle	Gpl_penspeed
Gpl_tsize	Gpl_textrotate	Gpl_lorg

Repeat the previous two steps as often as needed

Abort plot, if needed

Gpl_abortplot

Clear the plot (should always be called)
Gpl_clear

Go back to any step after plotter initialization if additional plots are desired

```
LOAD "GPL267" library

        DEL 5001,999
        LOAD SUB "GPL257", 5001,1
```

Terminate plotting

Gpl_terminate

```
DELETE GPL library

        DEL 5001,9999

RELEASE device
```

MEMORY USAGE

The GPL library was implemented in BASIC and takes a lot of user memory space. It was broken into the files "GPL251" through "GPL258" in order to conserve memory space as much as possible. The largest file, "GPL254", uses approximately 34K bytes of memory.

CHART DESIGN

APPENDIX
A

Charts offer a convenient way to summarize and analyze data. Visual aids allow fast interpretation of data, and allow trends to be spotted. Modern businesses are realizing the value of business charts, and are producing them with computer graphics products like GPL.

This appendix contains helpful hints on developing graphics programs. The four types of charts most commonly developed by programmers, and discussed here, are line charts, column charts, bar charts, and pie charts.

GRAPHICS CONSIDERATIONS

An often neglected part of computer generated charts is the graphic design. In this section, we have adapted commonly accepted principles that graphic designers and artists use to the capabilities of the HP plotters and the GPL software.

Simplicity

The first commandment in the art of charts is KEEP IT SIMPLE. A chart simplifies the analysis of data, as long as there is not too much data. If the chart starts to become very cluttered, or too many different trends are displayed, then break the chart down into several charts.

Too much or unnecessary text distracts the viewer from the data. Thus, the chart should contain only text that is necessary for the viewer to understand which specific data he is viewing. Conclusions from that data, strategy, recommendations, etc. should be printed elsewhere if the data is to have any impact.

You should keep your charts as simple as possible.

Select the Right Chart Type

This appendix discusses four common chart types. Select a chart type according to how you wish to analyze your data. For example, you may wish to compare several different sets of data. Or you may wish to see the trend of a single set of data as a function of some parameter. Or you may wish to see how the components of data compare.

Line charts offer a convenient means of comparing several sets of data or of looking at the trend for a single set of data. They are most often used when:

(1) there are many points in each data set

(2) points in a data set are spaced irregularly

(3) there are four or less sets of data

Column charts allow one to see more easily the trend of totals while breaking up data into components. They are not good for comparing several sets of absolute data.

Bar charts are used in much the same way as column charts, and the decision as to which to use depends on the number of sets of data and the magnitudes of the data.

Pie charts are used when the component breakdown is of interest. They should not be used for showing absolute magnitudes.

In choosing which chart to use, be sure to ask

(1) What am I trying to see? The trend of a single set of data, comparing the trends of several sets, or just showing the component breakdown.

(2) Is the data irregularly or regularly spaced?

(3) How many points in each data set? How many data sets?

(4) What is the range of data? If the range is very wide, perhaps logarithmic axes are required.

Plot Sizes

It is assumed that the plotting is to be done on ISO A4 size (210 mm by 297 mm) or English A size (8.5 in by 11 in) paper. For other sizes of paper, scaling may be done but one must be careful in handling the text.

A convenient aspect ratio for plots is for the Width/Height ratio to be the square root of 2 or the inverse. In this appendix, we will always consider the case where the plot is wider than its height. The square root of 2 is convenient when putting 2,4, or even 8 plots into the same plotting region as for a single plot. A convenient size is 254.6 mm (10.02 inches) wide by 180 mm high (7.09 inches), which has an aspect ratio very close (within .02%) of the square root of 2. The default LDC space was chosen to have an aspect ratio close to the square root of 2 while rounding to the easily remembered 250 by 180 region. When you know that you need to scale the plot to fit several on a page, you should use more exact values.

Text

Most text will be generated with the narrow line pens on a pen plotter (0.3 mm line width is typical for a new pen). The ideal aspect ratio for the height of a capital letter to the width of a line is in the range of 6 to 10. By picking capital letter heights of 3 mm, the aspect ratio begins at 10 for a new pen and goes down in value as the pen wears down. The text sizes in these recommended guidelines have been chosen to range from 2.5 to 4 mm, within the ideal range.

When fitting two or more plots on a single page, the application programmer should adjust the size and placement of the characters so that the characters stay at least 2.5 mm high, for purposes of readability as pens wear down.

Most text on a chart is used to generate a (1) title, (2) subtitle, (3) axis titles, (4) axis data point labels, (5) legends (data set labels), (6) footnotes, and (7) annotation.

Titles should be lettered with 4mm high characters, and a title should be limited to about 45 characters of width (180 mm max width). Subtitles should use 3.5 mm high characters, and should be limited to about 50 characters of width.

Three mm high characters are used for most other data: axis titles, axis data point labels, legend names, footnotes, and most annotation. Some annotation that is not part of the data, such as initials and date, may be drawn with 2.5 mm high characters.

When making the charts smaller than a full page, such as when fitting two or more on a single page, most of the text should still be in 3 mm high characters, and the width of lines and the number of text entries will have to be cut down to fit the smaller region.

GLOSSARY

APPENDIX B

Coordinate Systems: Four different coordinate systems control GPL plotting. The WC and LDC systems are used for device-independent plotting. The PLMM and PAMM systems are used for placing the plot on a specific physical plotter.

CP: Abbreviation for "Current Position".

Current Position: (X,Y) location in the world coordinate system that is used as a reference for drawing straight lines and graphics text. For straight lines, it represents the starting point for the line. Graphics text will be justified against the Current Position (abbreviated CP) depending on the current Label origin (Gpl__lorg) attribute value.

LDC: Abbreviation for "Logical Device Coordinates".

Logical Device Coordinates: Coordinate system for specifying placement on the (device independent) logical plotter. This logical plotter is defined in the application program (Gpl__defldc) before actually doing any plotting with straight lines and text.

PAMM: Abbreviation for "Physical Area Millimeters".

Physical Area Millimeters: Coordinates which are used to specify the actual plotting area on a physical device. The plotting area boundaries are specified by Gpl__physarea, and the placement of the LDC (Logical Device Coordinate) Limits is specified by a call to Gpl__physviewp.

PLMM: Abbreviation for "Plotter Millimeters".

Plotter Millimeters: Coordinates used to specify the actual placement of a plot on a plotting surface. The coordinates are in millimeters and are positive or 0. The point (0,0) refers to the minimum possible X and Y plotting coordinates; not the lower left corner of the paper. On most plotters, it is not possible to plot to the lower and left edges of the paper. The placement of a plot is specified by a call to Gpl__devorigin.

Viewing Transformation: Mapping between world coordinates and the logical device coordinates. It is specified by a window rectangle (in world coordinates) that is mapped onto a viewport rectangle (in logical device coordinates). All lines and text drawn within the window will be mapped to the viewport. Lines and text outside the window will be "clipped" (not plotted).

Viewport: The viewport is a rectangle that specifies the placement of graphical data on the logical plotter (logical device coordinates). It is specified by a call to Gpl__viewport. The viewport and window (Gpl__window) are usually specified together and are collectively called the "viewing transformation".

WC: abbreviation for "World Coordinates"

Window: The window is a rectangle in world coordinates that is to be mapped onto the logical plotter (logical device coordinates). It is specified by a call to Gpl__window. The window and viewport (Gpl__viewport) are usually specified together and are collectively called the "viewing transformation". All graphical data outside the window will not be plotted.

World Coordinates: Also called problem coordinates, user coordinates, or application coordinates. These may be defined for your application by specifying a viewing transformation.

GPL SUBPROGRAM FILES AND FONTS

APPENDIX C

SUBPROGRAM FILES

GPL uses these files during execution.

GPL251 - Device Selection
GPL252 - Plotter initialization and Plot Setup
GPL253 - Plot Setup and Specifications
GPL254 - Text and Straight Lines
GPL255 - Straight Lines
GPL256 - Text
GPL257 - Plot Termination
GPL258 - Text Width Inquiry function

FONT FILES

Font Name	Lower 96 File name	Roman Extensions File name
Serif Roman Light	GPLRL0	GPLRL1
Serif Roman Medium	GPLCR0	GPLCR1
Serif Roman Bold	GPLTR0	GPLTR1
Serif Italic Light	GPLLI0	GPLLI1
Serif Italic Medium	GPLCI0	GPLCI1
Serif Italic Bold	GPLTI0	GPLTI1
Stick Light	GPLSR0	GPLSR1
Stick Medium	GPLDR0	GPLDR1
Sans Serif Italic Light	GPLLJ0	GPLLJ1
Sans Serif Italic Medium	GPLMJ0	GPLMJ1
Gothic English	GPLGE0	GPLGE1
7220A/S, 7221A/B/S, 7225A/B	GPL271	GPL272
7220C/T, 7221C/T Plotters	GPL273	GPL274
7470A/HP ColorPro(7440A) plotters	GPL275	GPL276
7475A/ 7550A plotters	GPL275	GPL276

OTHER GPL FILES

GPL242 Used by GPLCFT
GPL243 Used by GPLCFT
GPL261 - Used by GPLFIL
GPLEnn - A series of example and reference programs

Summary of HP 250 Sans Serif Fonts

Sans Serif Light

GPLSR0, GPLSR1

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
  ° ¨ çÑñ¡¿¤£ §
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß
```

Sans Serif Medium

GPLDR0, GPLDR1

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
  ° ¨ çÑñ¡¿¤£ §
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß
```

Sans Serif Light Italic

GPLLJ0, GPLLJ1

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
  ° ¨ çÑñ¡¿¤£ §
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß
```

Sans Serif Medium Italic

GPLMJ0, GPLMJ1

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
  ° ¨ çÑñ¡¿¤£ §
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß
```

Summary of HP 250 Plotter Fonts

7220A/S, 7221A/B/S, 7225A/B

GPL271, GPL272

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
-  ···çÑñ ¡¿¤£¥§
äëöüáéóúàèòùâêôû
Åı0Æåíøæ¤ÄìÖÜÉÿß
```

7220C/I, 7221C/I

GPL273, GPL274

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
-  ··çÑñ ¡¿¤£¥§
âêôûáéóúàèòùäëöü
Åî0Æåíøæ¤ÄìÖÜÉïß
```

747OA

GPL275, GPL276

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~⊢
                £
-  ···çÑñ ¡¿¤£¥§
äëöüáéóúàèòùâêôû
Åı0Æåíøæ¤ÄìÖÜÉÿß⊢
```

Summary of HP 250 Decorative Fonts

Gothic English

GPL270, GPL271

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
                £
  ¨ çÑñ¡¿¤£ §
äëöüáéóúàèòùâêôû
ÅıØÆåíøæ¤ÄìÖÜÉïß
```

Summary of HP 250 Serif Fonts

Serif Light

GPLRL0, GPLRL1
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

Serif Medium

GPLCR0, GPLCR1
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

Serif Bold

GPLTR0, GPLTR1
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

Serif Light Italic

GPLLI0, GPLLI1
!"#$%&'()+,-./*
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

Serif Medium Italic

GPLCI0, GPLCI1
!"#$%&'()+,-./*
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

Serif Bold Italic

GPLTI0, GPLTI1
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
£
•¨çÑñ¡¿¤£§
âêôûáéóúàèòùäëöü
ÅîØÆåíøæÄìÖÜÉïß

GPL ERRORS

APPENDIX D

GENERAL PHILOSOPHY

Four types of errors commonly appear, categorized as follows:

- Soft errors - Caused typically by invalid parameter values in GPL subprogram calls that are non-catastrophic in nature.
- Hard errors - Caused typically by invalid parameter values in GPL subprogram calls that are catastrophic in nature.
- Parameter Reference Errors - Occur during program development. Caused typically by:

  a) Incorrectly typing a variable name

  b) Using an invalid variable type (such as INTEGER when REAL should have been used)

  c) Using an incorrect dimension (such as forgetting to dimension the GPL$ string)

- Miscellaneous - Usually occur during program development. Caused by:

  a) Memory overflow

  b) Improper subprogram loading

Each of these error categories is handled differently by GPL and is covered in more detail in the following sections.

SOFT ERRORS

The most common errors are caused by parameters out of range, which result in a GPL subprogram call being ignored. These errors are called "Soft Errors". Soft errors tend to cause part of a picture to be incorrect, such as a missing line, or an incorrect linestyle. A code specifying the routine in which the LAST soft error occurred is returned with GPL_terminate (unless a hard error occurred).

1101 - Gpl_clear
1102 - Gpl_defldc
1103 - Gpl_draw
1104 - Gpl_frame
1105 - Gpl_idraw
1106 - Gpl_imove
1107 - Gpl_linestyle

1108 - Gpl_lorg
1109 - Gpl_message
1110 - Gpl_move
1111 - Gpl_penspeed
1112 - Gpl_physarea
1113 - Gpl_devorigin
1114 - Gpl_pen
1115 - Gpl_physrotate
1116 - Gpl_physviewp
1117 - Gpl_terminate
1118 - Gpl_text
1119 - Gpl_transmit
1120 - Gpl_textrotate
1121 - Gpl_tsize
1122 - Gpl_useldc
1123 - Gpl_usewc
1124 - Gpl_viewport
1125 - Gpl_where
1126 - Gpl_window
1127 - Gpl_cset

HARD ERRORS

Certain errors, called "Hard Errors", can cause disastrous run time results. These can occur in two areas:

1. Invalid parameters during initialization of a device or file. This occurs when using Gpl_plotteris of Gpl_fileis. An error return parameter is included with each of these calls. A non-zero error return should always cause the application program to take corrective action before proceeding with a plot.

0 - No hard error encountered
1-1099 - ERRN for system errors not explicitly handled by Gpl_plotteris.
1151 - Invalid Gpl$ (probably insufficient length)
1152 - Invalid Model$ (not one of the supported devices)
1153 - Invalid Device address (Must be 8-9 or 11-20)
1154 - Invalid Message address (must be in range 8-20)
1155 - Invalid file name syntax
1156 - Invalid protect code syntax
1157 - Invalid Option in Gpl_fileis
1161 - Plotter ownership error - must do REQUEST before calling Gpl_plotteris.
1162 - No response from plotter (when the devices address ranges from 11 to 20)
1163 - Device is not the specified model (for device address ranging from 11 to 20)
1164 - Plotter could not be initialized (for a variety of reasons)
1165 - Unsupported plotter at Device address
1166 - Invalid plotter response during identify sequence
1167 - Incorrectable device error
1170 - File already exists (Gpl_fileis Option=0)
1176 - Old file could not be purged (Option=-1)

1181 - Plotter responded initially but later did not.
All sending of further data to plotter stopped.
1182 - Invalid plotter response during plotting
1183 - Picture file unavailable during plotting
1191 - Configuration file not found
1192 - Configuration file protected
1193 - Wrong file type for configuration file
1194 - Configuration file access error
1195 - Configuration file could not be opened
1196 - Invalid configuration file data format
1196 - Invalid model in configuration file
1198 - Invalid device address in configuration file
1199 - Invalid message address in configuration file

2. Lost data during output. This can occur due to a device being removed from on line status or due to system/device problems during data transfer. Some of these errors are detected and some are not. If such an error is detected, it will be signalled in the Gpl_terminate error parameter.

1181 - Plotter responded initially but later stopped.
Further plotting to device terminated.
1182 - Invalid plotter response.
Further plotting to device terminated.
1183 - Picture file could not be opened during plotting.
Sending plot data to file terminated.

PARAMETER REFERENCE ERRORS

The "Parameter Reference Error" causes the program to halt. It is usually caused by passing incorrect variable names, types, or dimensions to the subprogram parameter lists. For example, it is common for the programmer to supply an incorrect name for Gpl$. Such an error will cause error 202 to occur in most cases with a program halt. GPL does not detect or warn the programmer of such errors. The application programmer must thoroughly debug the program to prevent parameter reference errors.

Common errors:

8 - Improper parameter matching
9 - Incorrect # of parameters
in subprogram call

18 - Subscript out of range. Most common
cause: Improperly dimensioned Gpl$.
Be sure to use

```
DIM Gpl$[1200]
```

131, 132, 133 - Plotter is not available, of wrong type, or plotter
is down

202 - Insufficient dimension length in PACK statement. Probable cause:
Improperly passed Gpl$ array name in subprogram call.

MISCELLANEOUS ERRORS

2 - Memory overflow.

7 - Undefined function or subprogram

56 - File name is undefined.

314 - Device must be requested before using Gpl__plotteris or Gpl__devident.

SUMMARY OF GPL SUBPROGRAMS

APPENDIX
E

List of GPL Subprograms and the files in which they are found

A file containing the subprogram you need must be loaded at the time you call the subprogram.

	GPL 251	GPL 252	GPL 253	GPL 254	GPL 255	GPL 256	GPL 257	GPL 258
Gpl_abortplot				x	x	x	x	
Gpl_clear				x	x	x	x	
Gpl_config	x							
Gpl_cset		x	x			x		
Gpl_defldc		x	x					
Gpl_devident	x	x						
Gpl_devorigin		x	x					
Gpl_draw				x	x			
FNGpl_errm$	x	x						
Gpl_fileis		x					x	
Gpl_frame				x	x			
Gpl_idraw				x	x			
Gpl_itextwidth								x
Gpl_imove				x	x	x		
Gpl_linestyle		x	x	x	x			
Gpl_lorg		x	x	x		x		
Gpl_message				x	x	x		
Gpl_move		x		x	x	x		
Gpl_pen		x	x	x	x	x	x	
Gpl_penspeed		x		x	x	x		
Gpl_physarea		x	x					
Gpl_physrotate		x	x					
Gpl_physviewp		x	x					
Gpl_plotteris		x						
Gpl_terminate					x		x	
Gpl_text				x		x		
Gpl_textrotate		x	x	x	x	x		
Gpl_transmit			x	x	x	x	x	
Gpl_tsize		x	x	x	x	x		
Gpl_useldc				x	x	x		
Gpl_usewc				x	x	x		
Gpl_viewport				x		x		
Gpl_where				x		x		
Gpl_window				x		x		

NOTE

If Gpl_cset is called from GPL252 an error 7 might occur; therefore, always call Gpl_cset from files Gpl_253 or Gpl_256.

GPL_ABORTPLOT

```
Gpl_abortplot(Gpl$)
```

Used to abort a plot. Typically, this is used when there is a problem with paper or pens. Clears internal Gpl$ buffer. Clears device buffer.

GPL_CLEAR

```
Gpl_clear(Gpl$,Crt,Hardcopy)
```

Clears the plotter display, if possible. Sets up the device for paper change if the change is not programmable. Stores a special code in picture file to indicate the separation between plots. Moves pen stable out of way and/or puts pen away as a convenience when changing paper.

When doing any plotting this command should always be called at the end of a plot. It is especially useful when doing multiple plots.

Crt - Unused parameter reserved for future use. Intended to be used in clearing graphics Crt displays if and when such devices are supported by GPL. As Hewlett-Packard will make no guarantee about future device support, you should always set this parameter to 0.

Hardcopy - Used to clear a plotter. Currently allowed values include 0, 1, 3 and 4. No other values should be used at this time, as other values are reserved for other possible devices that might be supported in the future.

0 NOP (no operation)
1 Put pen away and/or move pen put of way
3* Advance paper a half page on chart advance plotters. Put pen away and/or move pen out of way for non-chart advance.
4* Advance paper a full page (ISO A3 or English 11x17) and cut the left edge for chart advance plotter. Put pen away and/or move pen out of way for non-chart advance plotters.

*When used with the new HP 7550A plotter, this parameter can also be used to feed a full page of paper from the plotter's paper magazine; setting the hardcopy parameter to either "3" or "4" causes a full page of paper to be fed to the HP 7550 plotter from the plotter's paper magazine.

Note: Chart advance commands are sent to chart advance plotters only when roll paper is loaded.

GPL_CONFIG

```
Gpl_config(Config$,Protect$,No_devices,Model$(*),Dev_add(*),
Mess_add(*),Error)
```

Interrogates the graphics device configuration file (GPL%CF or a user supplied file name) and returns the device addresses and model numbers for graphics devices connected to the system.

Config$ - Name of the configuration file. If left blank, GPL%CF in the current MSI disc is assumed.

Protect$ - Protect code for the configuration file name.

No_devices - Number of model numbers returned.

Model$(*) - String array containing the models connected to the system.

Dev_add(*) - Array of device addresses

Mess_add(*) - Array of message addresses

GPL_CSET

```
Gpl_cset(Gpl$,Char_set,Character_file$)
```

Used to select a software character set or to specify that the default hardware set be used.

Char_set - Specifies whether Character_file$ applies to ASCII codes 32-127 or to 160-255.

0 - 32-127 (usually used for US ASCII codes)
1 - 160-255 (usually used for Roman Extensions)

Character_file$ - Name of character set file. See Appendix A for a list of available character file names. When Char_set=0, use "" to specify default plotter set.

GPL_DEFLDC

```
Gpl_defldc(Gpl$,Xmin_ldc,Xmax_ldc,Ymin_ldc,Ymax_ldc)
```

Defines the logical device coordinate system. When called, the window and viewport are set equal to the specified LDC coordinate limits. The physical viewport is reset to the default of centered, isotopic, maximum size mapping onto the physical area. The units mode is reset to WC mode (Gpl_usewc).

Xmin_ldc - Minimum LDC x coordinate. Default is 0.

Xmax_ldc - Maximum LDC x coordinate. Default is 250. Should be greater than Xmin_ldc.

Ymin_ldc - Minimum LDC y coordinate. Default is 0.

Ymax_ldc - Maximum LDC y coordinate. Default is 180. Should be greater than Ymin_ldc.

The defaults are (0,250,0,180).

A soft error occurs if either Xmin_ldc >= Xmax_ldc or Ymin_ldc >=Ymax_ldc.

GPL_DEVIDENT

```
Gpl_devident(Dev_add,Model$,Advance,Buffer,Pens,Error)
```

Used to interrogate the plotter and important parameters at the specified plotter address.

Dev_add - Specified device address. Only input to subprogram. Note that Gpl$ is NOT a parameter for this call.

Model$ - Model number at this address. Important: Because of ambiguities in the identification process, the exact model number may not be returned. In that case, the model number returned will be sufficient for an input to Gpl_plotteris (because Gpl_plotteris also uses Gpl_devident). For example, if the device is a 7220T and the chart advance is not operational, the model number that is returned is "7220C".

NOTE

For the plotters 7470A, HP ColorPro(7440A), 7475A and 7550A a call to Gpl_devident returns the paper size for which the plotter is set. Model$ contains information of the form: model_number/paper size. The source of the paper size information varies with the plotting device. With the 7470A, HP ColorPro(7440A) and 7475A plotters the maximum plot-surface (paper size) depends on the position of the plotter's paper size switch. With the HP 7550A the paper size is determined automatically by the plotter. Thus, a user program can automatically adjust plotting parameters, such as the actual size of the plot.

Advance - Parameter specifying whether automatic chart advance is present.

0 - No auto advance is present
1 - Auto advance is present

Buffer - Number of bytes in the device buffer.

Pens - Number of pens on the plotter.

Error - Error return parameter. Zero if no error. Use FNGpl_errm$ to display the error message.

GPL_DEVORIGIN

```
Gpl_devorigin(Gpl$,X_lower_left_plmm,Y_lower_left_plmm)
```

Places the physical plotting area (as specified by Gpl_physarea) on the plotter. The origin gives the minimum x and y coordinates for plotting in the PLOTTER's coordinate system, regardless of whether the physical area is rotated.

X_lower_left_plmm - Minimum plotter x coordinate for any plotting to occur.

Y_lower_left_plmm - Minimum plotter y coordinate for any plotting to occur.

The defaults and minimum and maximum values depend on the plotter.

	Defaults		Minimum		Maximum	
Model$	X	Y	X	Y	X	Y
7220A/C 7221A/B/C	10	10	0	0	400	285
7220S/T 7221S/T	10	27	0	0	400	285
7225A/B	10	10	0	0	285.5	203.5
7470A(US)	1	10.2	0	0	259.1	190.2
7470A(A4)	9.5	7	0	0	274.2	190.2
7440A(US)	7.0	8.0	0	0	191.5	257.3
7440A(A4)	9.2	5.6	0	0	191.5	272.3
7475A(US-A)	5	14.5	0	0	198.2	257.9
7475A(US-B)	3.5	4.9	0	0	257.9	414.2
7475A(A4)	13.5	11.5	0	0	192.2	274.7
7475A(A3)	0.6	13.1	0	0	274.7	402.2
7550A(US-A)	2.2	15.3	0	0	196	254.2
7550A(US-B)	0.8	1.9	0	0	254.2	411.2
7550A(A4)	10.7	9.8	0	0	190	271.7
7550A(A3)	0	10.2	0	0	271.7	399.2

US, US-A, US-B, A4, and A3 refer to rocker switch settings on the back panel of each plotter.

GPL_DRAW

```
Gpl_draw(Gpl$,X_wc,Y_wc)
```

Draws a straight line from the CP to (X_wc,Y_wc). After the draw, the CP is updated to (X_wc,Y_wc).

X_wc - X world coordinate

Y_wc - Y world coordinate

Either X_wc or Y_wc may take on any real value.

FNGPL_ERRM$

```
DISP FNGpl_errm$(Error)
```

Used to display error messages corresponding to an error returned in call to one of the following:

Gpl_plotteris
Gpl_fileis
Gpl_terminate
Gpl_devident
Gpl_config

The error parameter is the one returned by the subprogram with the error.

GPL_FILEIS

```
Gpl_fileis(Gpl$,Picture_file$,Protect$,Option,Error)
```

Used to create and open a picture file in which plotter data will be stored. The data is specific to the model number declared by Model$ in Gpl_plotteris. After terminating the application (after Gpl_terminate is encountered and zero Error is returned), the picture may be sent to the device with program GPLFIL.

Picture_file$ - file spec for the file. It must satisfy the naming conventions for files. It must not have been created if Option=0 and must have been created if Option=-1.

Protect$ - Protect code for Picture_file$. If a file is to be appended, this is necessary for access. If a new file is created, then this code will be assigned to the file.

Option - Specifies whether a new picture file is to be created (Option=0) or replaced (Option=-1). Option=-1 creates a file after purging any previously existing file with the same name.

Error - Error return parameter for all errors that may be caused. Returned values include:

```
   0 - No error encountered
1151 - Invalid Gpl$
1155 - Invalid picture file name syntax
1156 - Invalid protect code for picture file
1157 - Invalid value for Option
1170 - File already exists (Option=0)
1176 - Old file could not be purged (Option=-1)
```

GPL_FRAME

```
Gpl_frame(Gpl$)
```

Draws a frame around the current window rectangle (or equivalently, around the current viewport rectangle). Uses current physical pen (Gpl_pen) and linestyle (Gpl_linestyle). The CP is unchanged from the value prior to the call.

GPL_IDRAW

```
Gpl_idraw(Gpl$,Dx_wc,Dy_wc)
```

Draws a straight line incrementally from the CP to CP+(Dx_wc,Dy_wc). The CP is updated to the end point after the draw.

Dx_wc - X incremental world coordinate

Dy_wc - Y incremental world coordinate

Either Dx_wc or Dy_wc may take on any real value.

GPL_IMOVE

```
Gpl_imove(Gpl$,Dx_wc,Dy_wc)
```

Moves the CP incrementally by (Dx_wc,Dy_wc).

Dx_wc - X incremental world coordinate

Dy_wc - Y incremental world coordinate

Either Dx_wc or Dy_wc may take on any real value.

GPL_ITEXTWIDTH

```
CALL Gpl_itextwidth(Gpl$,"ABC",Width)
```

Function GPL_itextwidth stands for Inquire Text Width. It determines the width of a proportional or non-proportional string for your use in computing formats.

GPL_LINESTYLE

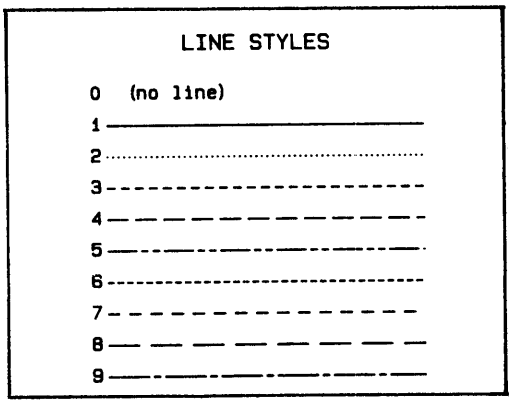
```
Gpl_linestyle(Gpl$,Linestyle)
```

Linestyle - Integer from 0 to 9 specifying the current linestyle.

0 - No lines. Suppresses straight lines from plot.
1 - Solid line (default)
2 - Dotted line
3 - Medium dashes
4 - Long dashes
5 - Very-long dash plus two short dashes
6 - Short dashes

7 - Medium long dashes
8 - Very-long dashes
9 - Very-long dash plus one short dash

See the following figure:



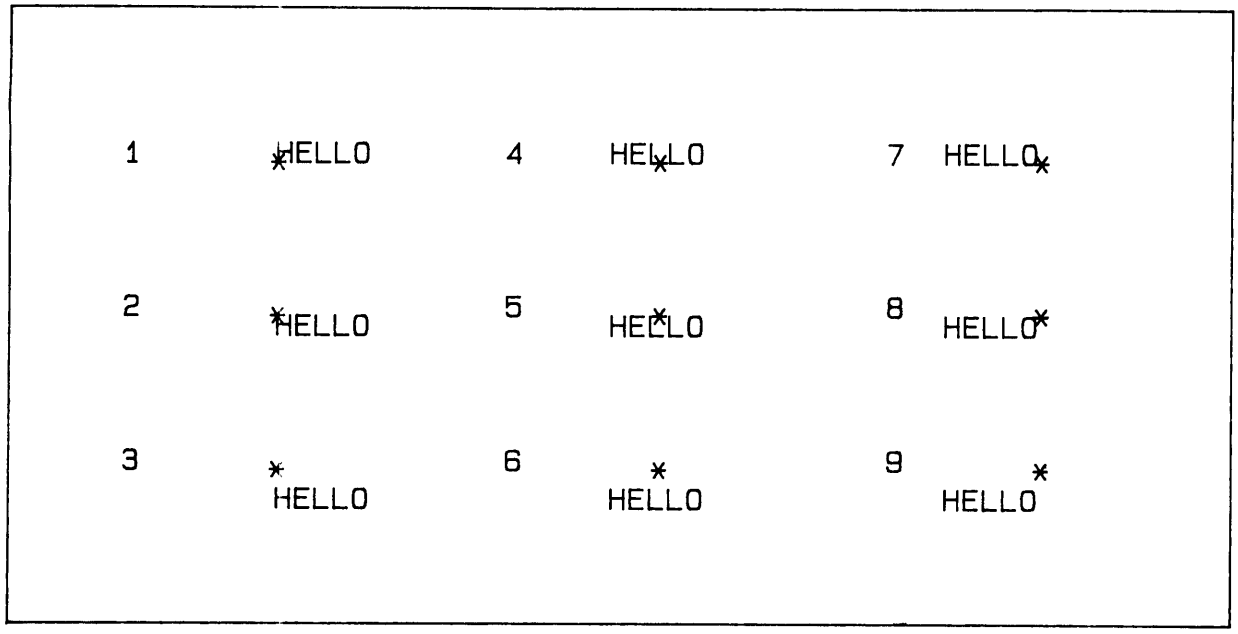
GPL_LORG

```
Gpl_lorg(Gpl$,Label_origin)
```

Selects the current text formatting option. Text lines will be justified both horizontally and vertically relative to the CP according to the current Label origin attribute value.

Label origin	Horizontal	Vertical
1 (default)	Left	Bottom
2	Left	Center
3	Left	Top
4	Center	Bottom
5	Center	Center
6	Center	Top
7	Right	Bottom
8	Right	Center
9	Right	Top

The above table shows where the CP is located relative to the string of text just prior to a call to Gpl_text. See the following figure. Note that the formatting is performed on character CELLS, not the characters themselves.



GPL_MESSAGE

```
Gpl_message(Gpl$,Message$,Wait)
```

Sends an appropriate operator message to the message address. Optionally waits for an operator response.

Message$ - String containing the operator message.

Wait - Specifies whether to wait for an operator response.

0 - No wait
1 - Wait until operator responds on message device

GPL_MOVE

```
Gpl_move(Gpl$,X_wc,Y_wc)
```

Moves the CP to (X_wc,Y_wc). The CP pen is lifted, interrupting the accumulated line patterns on dashes or dotted lines.

X_wc - X world coordinate

Y_wc - Y world coordinate

Either X_wc or Y_wc may take on any real value.

There is no guaranteed default CP, especially after changing the viewing transformation or outputting a line of text. Thus each output primitive should be preceeded by a call to Gpl_move.

GPL_PEN

```
Gpl_pen(Gpl$,Pen_number)
```

Raises physical pen. Selects new pen or puts it away if possible with selected plotter hardware.

Pen_number - New pen to be selected. Range: Integer values from 0 to 8.

0 - Put pen away. Suppresses text and lines.
1 - Select pen 1, if possible, or left pen on 7470A
2 - Select pen 2, if possible, or right pen on 7470A
3 - Select pen 3, if possible.
4 - Select pen 4, if possible.
5 - Select pen 5, if possible.
    Select pen 1 on four pen plotters.
6 - Select pen 6, if possible.
    Select pen 2 on four pen plotters.
7 - Select pen 7, if possible.
    Select pen 3 on four pen plotters.
    Select pen 1 on six pen plotters
8 - Select pen 8, if possible.
    Select pen 4 on four pen plotters.
    Select pen 2 on six pen plotters

GPL_PENSPEED

```
Gpl_penspeed(Gpl$,Velocity_cps)
```

Selects the pen velocity in centimeters per second. For devices without velocity select, this command is ignored.

Velocity_cps - Selected velocity in centimeters/sec. Range:1-40. The default is device dependent and the results depend on the device capabilities.

The device capabilities include:

Plotter	Velocity (char/sec)	actual Velocity (cm/sec)
7220 series	1-36	1-36
7221 series	1-36	1-36
7225A/3,7225A/4	any	25
7225B/3,7225B/4	any	25
7225A/3V,7225A/4V	1-25	1-25
7225B/3V,7225B/4V	1-25	1-25
7470A/7475A	1	.38
	2-3	3.8
	4-36	4-36
7550A	1-40	1-40
7440A (ColorPro)	1-40	1-40

GPL_PHYSAREA

```
Gpl_physarea(Gpl$,Width_pamm,Height_pamm)
```

Selects the area for actual plotting. The logical plotter coordinates will be mapped within this physical plotting area depending on the current physical viewport (Gpl_physviewp). The physical area may be translated by a call to Gpl_devorigin and rotated by a call to Gpl_physrotate.

Width_pamm - Width of plot in millimeters. This is the width as the plot (on paper) would be viewed. It is not the width in the plotter coordinate system. Default is 250 mm.

Height_pamm - Height of plot in millimeters. This is the height as the plot (on paper) is viewed. Default is 180 mm.

GPL_PHYSROTATE

```
Gpl_physrotate(Gpl$,Rotation)
```

Controls the rotation of the physical plotting area (as specified by a call to Gpl_physarea). See Chapter 9 for details.

Rotation - Rotation code. Takes on the following values:

0 No rotation on 7220, 7221, 7475A and 7550A series plotters. 90 degree clockwise rotation (CW) rotation on 7225, 7470A and HP ColorPro (7440A) plotters. Used most often for ISO A4 (English A) size plots that are oriented vertically.

1 Rotate 90 degrees CCW (counter-clockwise) on 7220,7221, HP ColorPro (7440A), 7470A and 7550A plotters. No rotation on the 7225 plotter. Used most often for ISO A4 (English A) size plots that are oriented horizontally.

2 (default) Software looks at Gpl_physarea parameters and attempts to fit on ISO A4 (English A) size sheet.

-1 90 degree clock-wise rotation (CW) on HP ColorPro (7440A), 7470A, 7475A and 7550A.

-2 180 degree rotation on HP ColorPro (7440A), 7470A, 7475A and 7550A.

GPL_PHYSVIEWP

```
Gpl_physviewp(Gpl$,Xmin_pamm,Xmax_pamm,Ymin_pamm,Ymax_pamm)
```

Specifies the placement of the LDC limits in the physical area (Gpl_physarea) coordinate system. The default is a value which centers the LDC space in the physical area coordinate system with maximum area. The default is set whenever Gpl_physarea or Gpl_defldc is called. Therefore, Gpl_physviewp should be called only after any calls to Gpl_defldc and Gpl_physarea.

This function is used whenever you wish to do one or more of the following:

(1) Stretch the plot in one direction, usually when you want to cover the entire physical area. Care must be used, because circles on the logical plotter will be deformed into ellipses.

(2) Move the plot from the center of the physical area.

(3) Change the size of the plot to something other than determined by the LDC and physarea coordinate limits.

X_min_pamm - Minimum x on physical area in millimeters
X_max_pamm - Maximum x on physical area in millimeters
Y_min_pamm - Minimum y on physical area in millimeters
Y_max_pamm - Maximum y on physical area in millimeters

GPL_PLOTTERIS

```
Gpl_plotteris(Gpl$,Model$,Device_address,Message_address,Error)
```

Initializes the plotter and the Gpl$ parameters for other GPL subprogram calls. Before doing this, it verifies that the specified model is the one at the specified address.

Model$ - String containing the model number of the plotter (e.g. "7220A" or "7225B/4V").
Model$ can also contain the paper size appended to the model name (as is returned by a call to the Gpl_devident subprogram). For example, the parameter Model$ might now contain a string similar to "7550A/A3" when calling Gpl_plotteris. If a paper code is explicitly specified it must match the actual plotter setting. If only the model name is given, the Gpl$ string is initialized with the actual plotter setting.

Device_address - system address to which the plotter is connected. Allowed range: 8-9,11-20.

Message_address - system address to which the operator messages are to be sent. Allowed range: 8-9.

Error - Error return parameter for the hard errors which may occur. Values to be returned include:

```
    0 - No hard error encountered
1-1099 - ERRN for system errors  not explicitly handled by Gpl_plotteris
  1151 - Invalid Gpl$ (probably insufficient length)
  1152 - Invalid Model$ (not one of the supported devices)
  1153 - Invalid Device address (Must be 8-9 or 11-20)
  1154 - Invalid Message address (must be in range 8-20)
  1161 - Plotter ownership error - must do REQUEST calling Gpl_plotteris
  1162 - No response from plotter (when the devices address range from
         11 to 20)
  1163 - Device is not the specified model (for device address
         ranging from 11 to 20)
  1164 - Plotter could not be initialized
  1165 - Unsupported plotter at Device address
  1166 - Invalid plotter response during identify sequence
  1181 - Plotter responded initially but later did not.
         All further sending of data to plotter stopped.
```

The application program should always check the error parameter for hard errors.

GPL_TERMINATE

```
Gpl_terminate(Gpl$,Error)
```

Ends the GPL application. This is the last GPL call in an application.

Error - Error return parameter for soft errors that may have occurred. May be 0 (no errors) or positive (code specifying the last subprogram call in which an error was encountered).

The codes and the associated subprograms are listed below:

1101 - Gpl__clear
1102 - Gpl__defldc
1103 - Gpl__draw
1104 - Gpl__frame
1105 - Gpl__idraw
1106 - Gpl__imove
1107 - Gpl__linestyle
1108 - Gpl__lorg
1109 - Gpl__message
1110 - Gpl__move
1111 - Gpl__penspeed
1112 - Gpl__physarea
1113 - Gpl__devorigin
1114 - Gpl__pen
1115 - Gpl__physrotate
1116 - Gpl__physviewp
1117 - Gpl__terminate
1118 - Gpl__text
1119 - Gpl__transmit
1120 - Gpl__textrotate
1121 - Gpl__tsize
1122 - Gpl__useldc
1123 - Gpl__usewc
1124 - Gpl__viewport
1125 - Gpl__where
1126 - Gpl__window
1127 - Gpl__cset
1181 - Plotter responded initially but later stopped.
Further plotting to device terminated.
1182 - Invalid plotter response.
Further plotting to device terminated.
1183 - Picture file could not be opened during plotting.
Sending plot data to file terminated.

GPL__TEXT

```
Gpl_text(Gpl$,Text$)
```

Draws a line of text on the graphics display. The placement of the text relative to the Current Position (CP) is controlled by the Text rotation (Gpl__textrotate) and Label origin (Gpl__lorg) attribute values. The size of the characters is controlled by the Text size (Gpl__tsize) attribute values. The text supplied must be in the 96 character printable set or the Roman Extensions. The only control characters that will be recognized are CR (carriage return ASCII 13) and LF (line feed - ASCII 10), at the end of the string. If no CR and LF are present, the CP will be at the end of the string of text, ready for more text to be appended as if the Label origin attribute were one. A CR moves the CP back to the starting position, and a LF moves the CP and starting position down one line.

Text$ - String to be plotted. The only control characters that may be added are CR and LF, at the end of the string.

GPL_TEXTROTATE

```
Gpl_textrotate(Gpl$,Rotation)
```

Rotates text in 90 degree increments. Rotation is the number of 90 degree increments to rotate. Allowed values include:

0 - No rotation
1 - Rotate 90 degrees CCW (counterclockwise)
2 - Rotate 180 degrees CCW
3 - Rotate 270 degrees CCW

GPL_TRANSMIT

```
Gpl_transmit(Gpl$,Option)
```

Clears GPL graphics buffer (stored on Gpl$) of graphic data and sends the data to the device. Used for optimizing performance and for synchronizing the application program with the plot. Called automatically when the following are called:

```
Gpl_clear
Gpl_terminate
```

Option - Reserved for future use with different classes of graphics devices. It should always be set to 0.

GPL_TSIZE

```
Gpl_tsize(Gpl$,Cell_height_ldc,Cell_width_dev_pct)
```

Selects the current text size. The height of the character cell is the base line to base line distance between lines of text. The width is the distance between the left edges of characters in a line. On the supported plotters, capital letters are 1/2 the cell height and 2/3 the cell width. The characters are justified against the left and lower edges of the cells.

Cell_height_ldc - Height of character cell in logical device coordinate units. Range: Any real>0. Default: 9

Cell_width_dev_pct - Percentage deviation from default. Range: From -99.99 to +200.00. Default: 0.

GPL_USELDC

```
Gpl_useldc(Gpl$)
```

Sets the current units to the LDC space. The window and viewport are temporarily reset to be equal to the LDC space. The previously specified window and viewport rectangles may be restored with a call to Gpl_usewc. The value of the CP is lost.

GPL_USEWC

```
Gpl_usewc(Gpl$)
```

Restores the current units to the last specified windows and viewport. This is used after a call to Gpl_useldc was used to plot temporarily in the LDC space. The value of the CP is lost.

GPL_VIEWPORT

```
Gpl_viewport(Gpl$,Xmin_ldc,Xmax_ldc,Ymin_ldc,Ymax_ldc)
```

Sets the viewport rectangle to the specified value. The viewport rectangle specifies the placement of window data on the logical plotter. The viewport is reset to the default of LDC limits whenever Gpl_defldc is called. In addition, Gpl_viewport restores the units mode to world coordinates (Gpl_usewc).

Xmin_ldc - Minimum x LDC coordinate
Xmax_ldc - Maximum x LDC coordinate
Ymin_ldc - Minimum y LDC coordinate
Ymax_ldc - Maximum y LDC coordinate

A soft error occurs when either Xmin_ldc >= Xmax_ldc or Ymin_ldc >= Ymax_ldc

When Gpl_plotteris is called, the window, viewport, and LDC space are set to the default of (0,250,0,180). In addition, the units mode is World Coordinates (Gpl_usewc).

GPL_WHERE

```
Gpl_where(Gpl$,X_wc,Y_wc)
```

Inquires and returns the Current Position (CP). The CP is moved whenever a straight line or text is drawn on the plotter. In addition, the CP may be set by a call to Gpl_move.

X_wc - X world coordinate
Y_wc - Y world coordinate

GPL_WINDOW

```
Gpl_window(Gpl$,Xmin_wc,Xmax_wc,Ymin_wc,Ymax_wc)
```

Sets the window rectangle, which defines the world coordinates to be mapped onto the plotter. The placement on the logical plotter is specified by the viewport rectangle. All data outside the window rectangle is "clipped" (suppressed from being plotted). All graphical data (straight lines and text) within the rectangle is mapped to the viewport. The default window is the logical device coordinates, and it is reset to the default whenever Gpl_defldc is called. Gpl_window also resets the units mode to World Coordinates (Gpl_usewc)

Xmin_wc - Minimum X world coordinate
Xmax_wc - Maximum X world coordinate
Ymin_wc - Minimum Y world coordinate
Ymax_wc - Maximum Y world coordinate

A soft error occurs when either Xmin_wc >= Xmax_wc or Ymin_wc >= Ymax_wc. When Gpl_plotteris is called, the window, viewport, and LDC space are set to the default of (0,250,0,180). In addition, the units mode is World Coordinates (Gpl_usewc).

Part No. 45261-90064 E0986

Printed in Federal Republic of Germany
September 1986

HERRENBERGER STRASSE 130
D-7030 BOEBLINGEN