GERI RICKERT

HP 70000 Modular Measurement System

Display Interface

Design Guide

HEWLETT PACKARD

Manual Part Number 5958-6653

Printed in U.S.A. December 1, 1988

Contents

Chapter 1
Introduction 5
Design Guides 5

Chapter 2
The Display in the HP 70000 System 7
Definition of a Display 7
Language Design Philosophy 7
System Protocols 8
Controllers 8
Instruments 8

Chapter 3
Use of HP-MSIB Protocols 11
Power-Up 11
Key Press 11
RPG (knob) 12
LOCAL (LCL) 12
I-P (Instrument Preset) 12
Manual Operation 12
DISPLAY (DSP) 12
Select Instrument 13
Display Preset 14
Address Map 14
Report Errors 15

Chapter 4
Use of Display Resources 17
Windowing 17
Graphics Capabilities 18
Objects and Attributes 18
Groups and Items (Referenced Graphics) 20
Memory 21
Multi-Instrument Systems 22
Defaults and Power-Up State 23
HP-IB and Control Links 23

Chapter 5
COMMAND REFERENCE 25

HP-GL . . . 25
Links . . . 25
Syntax . . . 26
Response . . . 28
Example . . . 28
Comments . . . 29
Firmware Revision Dependencies . . . 29
Keyword Dictionary . . . 29

Appendix A
Command Cross References 173

List of Commands (Alphabetical by Mnemonic) . . . 173
List of Commands (Alphabetical by Description) . . . 174
Link Requirments for Commands . . . 175
HP-GL Commands . . . 176
Non-HP-GL Commands . . . 176
List of Commands by Function . . . 177
- Character Sets . . . 177
- Display Configuration . . . 177
- Error Handling . . . 177
- Hardcopy Output . . . 177
- Informational Displays . . . 177
- Labeling the Screen . . . 178
- Markers . . . 178
- Referenced Graphics . . . 178
- Remotely-Controlled Display . . . 178
- Simple Graphics . . . 179
- Utility Commands . . . 179

Appendix B
Screen Formats 181

Resolution . . . 181
Screen Layout . . . 182

Appendix C

Errors, Masks, and Status Bytes **185**
Error Mask 185
Status Mask 186
Parallel Mask 187
Clearing Errors 187

Appendix D
Character Sets **189**

Appendix E
Scaling and Origins **193**

Appendix F
Special Characters **208**

1 Introduction

The HP 70000 Modular Measurement System is a modular architecture optimized for rf and microwave instrumentation, providing downsized modular instrumentation that can share common system components. The HP 70000 system specifies a complete modular environment by defining a mainframe, display, and communication protocols that serve as the common components of the system.

The HP 70000 system operates under well-defined rules of communication protocol. These rules assure orderly display-to-module and inter-module communications. The communication protocol is described briefly in succeeding chapters of this manual. It is more thoroughly documented in the *Communication Protocol Design Guide.*

The *module* is the basic "building block" of the HP 70000 system. A module may be part of an instrument and each module is designed to perform a specific set of functions. In a system there may be one or more mainframes that contain modules, or modules may be in self-contained, stand-alone housing. Modules in the HP 70000 system communicate with the display and with each other using the HP-MSIB interface.

The display provides the human interface and information presentation ability for one or more modules. This manual documents the internal language which is implemented in the display.

Design Guides

There is extensive documentation to assist designers in developing HP 70000 modules and instruments. Published design guides provide engineering how-to information based upon many years of module-development experience. These guides illustrate the engineering principles required to design into the HP 70000 modular format. By using the design guides, the designer can learn from Hewlett-Packard's experience and avoid unnecessary experimentation.

- HP 70595A Module Development Design Guides
  1. Electrical Design Guide (5958-6626). Power supply and magnetic design.
  2. Mechanical Design Guide (5958-6627). Parts design and thermal management.

3. Electromagnetic Compatibility Design Guide (5958-6628). EMC requirements and testing.

- HP 70596A Module Communication Design Guides
  1. HP-MSIB Interface Design Guide (5958-6629). HP-MSIB interface device design specifications.
  2. Communication Protocol Design Guide (5958-6631). The protocol required for communication and the operational requirements for module types.
  3. Display Interface Design Guide (5958-6653). The manual instrument interface for the HP 70000 system.

2
The Display in the HP 70000 System

Definition of a Display

A "display" is a module of the HP 70000 System which has graphics, text, and human interface resources which can be shared among other modules in the system. It serves as the central human/machine interface for the system. Its resources can be thought of as a graphics system ("screen" or "window") and a keyboard. A screen is the entire surface used to display information. A window is the part of that screen allocated to a given module. A keyboard is any input device to the display, such as a numeric keypad, a mouse, a knob, an ASCII keyboard, softkeys, and so forth.

Any module which is allocated any portion of these resources sends commands and receives responses or status bytes from the display either over the Hewlett-Packard Modular System Interface Bus (HP-MSIB) or over HP-IB. For example, if allocated a portion of the screen, a module or a computer can send commands to draw axes, to put up a graph, to label that graph, and so on. If the module or the computer is allocated the keyboard, it can put up softkey labels and read the keyboard in either a polled or interrupt driven fashion. The minimum keyboard consists of 14 soft keys (see Figure 5.1 on page 100 for a diagram of the softkey numbers), a numeric entry pad (0-9, minus sign, decimal point, up arrow, down arrow, left arrow, and knob), the special system keys DISPLAY (DSP on the HP 70205A), USER (USR on the HP 70205A), and MENU (MNU on the HP 70205A), as well as I-P (instrument preset), HOLD, and LOCAL keys.

Language Design Philosophy

Modules use the display resources by establishing three types of links (graphics, keyboard, and control) over HP-MSIB. Computers can use these same resources by controlling the display over HP-IB. These processes can take place simultaneously. To optimize performance in the intended environment, there have been several philosophies applied in the design of this language:

Communications over the two buses are as equivalent as possible. For example, the HP-MSIB ***END*** command means the same thing as ***END*** asserted on HP-IB (using ***EOI***).

The language resembles HP-GL (the Hewlett-Packard Graphics Language) and complies as far as possible with IEEE-728.

Certain characteristics of the language have been optimized for speed with the result that the action taken upon receipt of some commands depends on which commands have previously been received.

Most of the communications in the display language are stimulus/response communications. A module or computer sends a command and an action is taken by the display. The other type of communication involves information transfer. For example, a command is sent to the display and the display sends data in response, or its "status byte". On HP-IB or HP-MSIB, a single 8-bit status byte is defined which is made available to a controller or module either through serial poll on HP-IB, the ***STATUS*** HP-MSIB command, or the **OS** (Output Status) display command.

System Protocols

Display resources can be used by any controller on HP-IB or any module on HP-MSIB. The arbitration and passing of control is strictly defined in the *Communication Protocol Design Guide*[1] for the HP 70000 System. Communication on HP-MSIB is defined in terms of links. State diagrams are defined to arbitrate control on both the HP-MSIB bus and on HP-IB.

Controllers

Throughout this document the term "controller" will be used to mean either an HP-IB controller (IEEE-488) or an HP-MSIB controller (Control Link Initiator). The only differences between the two are related to the hardware differences between the two buses. For example, addresses are from 0 to 31 on HP-IB, whereas there are 8 bits of addressing on HP-MSIB, with the upper 3 bits (0-7) being a row address and the lower 5 bits (0-31) being a column address. All of the commands in this language can be sent by a controller, that is, a module which which has established a "control link". Commands from a computer on HP-IB cause a control link to be established to HP-IB, if HP-IB is in the remote state.

Instruments

Modules on HP-MSIB share the display resources. The display is the initiator of keyboard links and of graphics links. (An instrument with a keyboard link can use the keyboard. One with a graphics link can use the screen.) The display offers these links to instruments on HP-MSIB under user control; those who respond are known as keyboard

[1]See page 6

link responders or graphics link responders. This means that an instrument can not give itself any of the display's resources; they must be assigned *by the display* either manually or through a controller.

When assigned a keyboard link, the instrument can use the keyboard (numeric pad, knob, softkeys, left arrow, and so on). A portion of the screen called a graphics *window* is assigned to the instrument when a graphics link is established. A subset of the language is used to manipulate these resources. See Appendix A for a cross reference of all commands and who can send them.

As mentioned above, instruments and controllers can be assigned graphics windows and the keyboard manually by the system user, or assignment can be made by a controller, using the **BW** (Build Window) command. A controller can also inject itself between an instrument and its keyboard by using the pre-process mode (see the **PP** command). Flexibility is provided by this latter method but it is hidden from the casual programmer of the display since it is invoked only by the **PP** (Pre-Process Mode) command.

If an instrument does not have either a graphics link, a keyboard link, or a control link, it can not send any display language commands or receive responses or status bytes. See the *Communications Protocol Design Guide* for further information on links.

3

Use of HP-MSIB Protocols

As discussed in chapter 2, display resources can be used by any controller on HP-IB or any module on HP-MSIB. A display shares its resources among instruments on HP-MSIB. The purpose of this chapter is to describe the display's use of HP-MSIB protocols for communication and module interaction in the HP 70000 System (that is, establishing and using links).

Power-Up

The display memory is such that upon power-up the display will attempt to extablish graphics links and/or a keyboard link to those modules with which it had links defined on power-down. It will do so even if those links had been temporarily broken during use of some of the manual functions of the display.

Key Press

Whenever the display has a keyboard link established it is in USER or MENU mode. These modes are entered by pressing the USER or MENU keys on the HP 70206A, or the USR or MNU keys on the HP 70205A. If a keyboard link is established, any keyboard press, other than DISPLAY (DSP on the 70205A), LOCAL (LCL on the 70205A), or the PLOT or PRINT hardkeys on the 70206A, will cause the display to attempt to communicate with the module with which it has the keyboard link.

Communication of key codes follows a specific sequence of HP-MSIB protocol events. The display will first read the key and identify its key code[1] then it will send a ***STATUS*** HP-MSIB command to the module which currently has control of the keyboard[2]. The ***STATUS*** HP-MSIB command sends a status byte with the "key pressed" bit set. It does *not* send the key code. When the module receives the HP-MSIB command it must note that the key press bit is set (see appendix C) and respond by sending the ***KY*** (Send Keyboard Data) command to the display to find out which key was pressed. See the ***KY*** command for more on this interaction.

[1]See page 99 for a list of key codes

[2]If the link is over HP-IB, the SRQ status byte is used rather than the ***STATUS*** HP-MSIB command. In either case the ***STATUS*** message will only be sent if enabled by the ***IM*** command.

RPG (knob)

Whenever the display is in USER or MENU mode, any turning of the knob will cause the display to send a ***STATUS*** HP-MSIB command to the module with the keyboard link[3]. With the knob bit of the status byte set, the ***STATUS*** command shows that the RPG count has been updated. The module should respond with the ***RP*** (Send RPG Data) command to learn the knob count since the last reading. The display will respond with the knob count and direction (positive is clockwise). See the ***RP*** (Send RPG Data) and the ***RG*** (Simulate RPG Turned) command descriptions for more about the knob.

LOCAL (LCL)

Whenever the LOCAL hardkey (LCL on the HP 70205A) is pressed, the display will send the HP-MSIB command ***RETURN TO LOCAL*** to every module on row zero (including itself). It will then be up to each module to leave HP-IB REMOTE mode. No response or acknowledgment by the module to the display is made.

I-P (Instrument Preset)

When this key is pressed the ***STATUS*** HP-MSIB command is sent to whichever module currently owns the keyboard[3]. The status byte includes a bit specifically for instrument preset pressed as well as the key pressed bit (see appendix C). Both bits will be set so the module can but does not have to query the display to find out what button was pressed as it does with other keys. The reaction by the module to having the I-P key pressed is module dependent.

Manual Operation

This section will describe how the display uses HP-MSIB protocols to perform some of the manual functions which are accessed through softkeys by first pressing the DISPLAY hardkey (DSP on the HP 70205A).

DISPLAY (DSP)

If the display has a keyboard link with any module when the DSP hardkey is pressed, the display will send the ***BREAK LINK: KEYBOARD*** HP-MSIB command to that module. It will

[3]If the link is over HP-IB, the SRQ status byte is used rather than the ***STATUS*** HP-MSIB command. In either case the ***STATUS*** message will only be sent if enabled by the ***IM*** command.

not send a ***BREAK LINK: GRAPHICS*** and the module may continue to send graphics to the display if it has a graphics link. Even though the graphics may not be seen on the screen, they still go into the vector list (an internal list of on screen strokes) and remain available for later viewing. The display will retain in its memory which module has the graphics link and which did have the keyboard link when the DISPLAY key was pressed. If no new graphics links or keyboard links are established after the DISPLAY key has been pressed, when MENU or USER is pressed the display will re-establish its keyboard link with the module by sending an ***ESTABLISH LINK: KEYBOARD*** to that same module.

Select Instrument

When the SELECT INSTR key is pressed, the display immediately begins a search to find an instrument so that it can establish a graphics link. It does so in the following manner:

The HP-MSIB command ***SEND MODULE ID*** is sent to row zero of the column address one greater than the column address address to which a graphics link is currently established (if no link currently exists, it is sent to column address 0). If there is no module at this column then the HP-MSIB interface device will cause a No Module At Address (NMAA) condition to occur and the display will move on to the next column, row zero address. If there *is* a module at the address then the module must send an identification string as its response (defined in the *Communication Protocol Design Guide*).

Once the display has received a response to its request for identification, it will then offer a graphics link to the module by sending an ***ESTABLISH LINK: GRAPHICS*** HP-MSIB HP-MSIB command. The display then waits for the instrument to respond by either sending an ***ACCEPT LINK: GRAPHICS*** or ***REJECT LINK: GRAPHICS*** HP-MSIB command. If the instrument accepts the link then the display allocates the entire screen to the module. If the module rejects the graphics link, then the display moves on to the next column. If the display reaches column 30 without establishing the graphics link then it "wraps"[4] around and starts with column 0 and continues until either a graphics link has been established or it is back where it began.

Once the graphics link has been established the menu will still be the display's display's menu because SELECT INSTR is part of the display's menu system. If the MENU or USER hardkey (MNU or USR on the HP 70205A) is now pressed then the display sends an ***ESTABLISH LINK: KEYBOARD*** HP-MSIB command, the module responds with an ***ACCEPT LINK: KEYBOARD***, and the module may label the softkeys (see ***ML*** command).

The value of SELECT INSTR lies in the fact that it provides a way to easily give the whole screen and keyboard to an instrument. It also provides the way to move the display to the next instrument on HP-MSIB whenever the system is already in operation.

[4]ROM version 5.0 does not "wrap" around.

SELECT INSTR clears all errors, clears SRQ and the serial poll status byte, clears the HP-IB output buffer, resets the error and status masks, and resets the label terminator to its default value.

Display Preset

DISPLAY PRESET[5] clears the screen and breaks all links it has with any modules and then it offers the screen and a keyboard link to the last module which had the keyboard link.

For example, suppose a display had three windows, each showing a graphics link with a different module, and it had a keyboard link with one of those modules. If DISPLAY PRESET is pressed then the display will send a ***BREAK LINK: GRAPHICS*** HP-MSIB command to each of those instruments. It will wait until each instrument has responded with an ***ACCEPT LINK: GRAPHICS*** HP-MSIB command. Once all have responded, the display will then send an ***ESTABLISH LINK: GRAPHICS*** to the instrument which had the keyboard and when the module has responded with an ***ACCEPT LINK: GRAPHICS*** HP-MSIB command then the display will give the entire screen and the keyboard to that instrument.

DISPLAY PRESET thus provides a method for changing the display screen format from multiple windows to a single window. DISPLAY PRESET also provides a way to clear HP-IB graphics from the screen, since there is no "clear screen" button. It also clears all errors, clears SRQ SRQ and the serial poll status byte, clears the HP-IB output buffer, resets the error and status masks, establishes the default character sets, size, and direction, turns off ***PP*** (Pre-process mode), re-initializes the graphics system (***DX***, ***LT***, ***BA***, and so on) and resets the label terminator to its default value. Furthermore, it resets the hardcopy parameters (set by the define hardcpy menus or the ***CY, PI, PL, EJ, HR*** and ***KC*** commands) to their default values.

Address Map

address map is a function key which will cause a matrix showing all the modules and their HP-IB and HP-MSIB addresses to appear on the screen. At any given time, four column addresses and eight row addresses are seen on screen, or 32 total. When the function is initiated, or whenever the map is shifted left or right, the entire on screen space is scanned. The scan is performed by sending an HP-MSIB ***SEND MODULE ID*** command to each address (the scan goes across a row, column by column, then up to the next row and across until all modules on screen have been scanned). At each address, either an NMAA or an ID will send the scan to the next address. All the column numbers are updated before beginning this scan but old blocks are not shifted; the new information simply overwrites the old as it is sent. (Hence a slow-to-respond module can make all the column numbers look wrong until it responds.)

[5]DISPLAY PRESET is not implemented in ROM version 5.0

As mentioned, the HP-MSIB command ***SEND MODULE ID*** is sent to each address on the screen to get the information to put up in the address map. Six softkeys (ADJUST COLUMN, ADJUST ROW, SET HP-IB, ALLOC DISPLAY, ALLOC KEYBD, ALLOC SCREEN) will also appear on the screen. The ADJUST COLUMN and ADJUST ROW keys are used with the knob to place the cursor of the display on a particular address. Once a module has been located then pressing the other softkeys will cause the display to perform according to a specified protocol.

If ALLOC SCREEN is pressed, the display will immediately break *all* of its graphics links by sending the HP-MSIB command ***BREAK LINK: GRAPHICS***. It will entirely purge its screen memory and then send ***ESTABLISH LINK: GRAPHICS*** to whatever module is highlighted on the address map and give that module the entire screen.

If ALLOC KEYBD is pressed, the display, which has already broken its keyboard link when DISPLAY was pressed, will offer a keyboard link to the module which is highlighted on the address map. Actually, the display memory remembers which module was highlighted when ALLOC KEYBD was pressed but does not send the ***ESTABLISH LINK: KEYBOARD*** HP-MSIB command until the display menu is exited by pressing the MENU or USER hardkey.

If ALLOC DISPLAY is pressed, the display will perform the same function as ALLOC SCREEN and in addition will send the ***ESTABLISH LINK: KEYBOARD*** HP-MSIB command when the display menu is left by pressing the MENU or USER hardkey.

If SET HP-IB is pressed, the softkeys will be blanked and the softkey ENTER will appear and the HP-IB address of the row zero module which is highlighted on the address map grid can be set. When the number is entered the display will send the HP-MSIB command ***SET IEEE488 ADDRESS*** to the module. The module will then either set its soft-settable HP-IB address or it will report an error.

Report Errors

Whenever a module has an error, it must inform every row 0 module in the system, by sending the HP-MSIB command ***ERROR OCCURRED***. This will cause an E to appear in the status box on the screen of all displays in the system. When this occurs, if the REPORT ERRORS softkey is pressed, the display will send the HP-MSIB command ***SEND ALL ERRORS*** to every module that notified it that an error had occurred. The module will respond with a ***COMMAND RESPONSE*** packet containing the description of the errors (ASCII characters with line feeds to separate each line). The ***END COMMAND RESPONSE*** is sent as a terminator. The display will list the description of all of the errors on the screen. If more than one module has reported errors then a MORE ERRORS softkey will appear and when pressed the screen will list the next module's errors. This is a one time transfer of information; there is no updating, and listing the description of an error on the screen will clear the error from the module if it is not hardware related.

4

Use of Display Resources

Windowing

The screen of a display is divided into windows (see Appendix B). The four kinds of windows are:

1. menu windows,
2. the character window,
3. graphics windows, and
4. the status window.

The *menu window* and *character window* are inseparable and are assigned to the controller or to the keyboard link responder. The menu windows are used to annotate the 14 softkeys of the display (see Figure 5.1 on page 100). The menu windows provide softkey labels to show the action accomplished when each key is pressed. The ***ML*** command is used to write to the menu windows.

The *character window* (or *character line*) is a single line used to provide feedback for manual or remote control. The ***FC*** command is used to write to the character window. The *status window* is used by the display to show the status of HP-IB and various system functions.

The *graphics window* of a display may be assigned to one controller or graphics link responder or may be broken into smaller rectangular windows, each of which may be assigned to a controller or graphics link responder. A graphics window is built manually by the system user or by a controller using the ***BW*** command. Each graphics window, whether one or more occur on a screen, is characterized by a P1 and a P2. These are x,y coordinate pairs which define the lower left and upper right corners of the window. A module has no control over the size or location of its graphics window. To the module, the display's window is simply a graphics device with a specified P1,P2. The language is defined such that one graphics window is entirely independent of any other graphics windows (see the Multi-Instrument Systems section on page 22).

The actual numerical values which correspond to any x,y pair are in the smallest addressable units (pixels, or dots). These are designated as "display units". The HP 70205A and the HP 70206A have screens which are 1024 × 384 display units. If a computer or instrument wishes to send graphics commands scaled to its *own units*, the display language has this capability. These units are called "user units". For example, if

a window's P1,P2 are (325,234) and (599,383) the computer or instrument can define a more useful coordinate space using the **SC** command. Perhaps the x axis can range between 0 and 100 and the y axis can range between 0 and 50 (`SC0,100,0,50`). An **SC** command without parameters sets user units equal to display units. This is the default when a window is built.

Graphics Capabilities

Objects and Attributes

Although the display can be used as a simple HP-GL plotting device, more sophisticated graphics which aid in the display of measurements are supported. These are based on identifying each thing drawn on the screen as a discrete "object", and associating the objects objects into "groups" and "items". There are nine types of objects:

1. Text: Single and multiple line labels of ASCII text. The text can be of various sizes and rotations and enhancements.
2. Absolute Plots: Plots of X,Y pairs.
3. Relative Plots: Plots of X,Y increment pairs.
4. Absolute Graphs: Plots of successive Y values. The display will auto-increment the X value by a specified amount.
5. Relative Graphs: Plots of successive Y increments. The display will auto-increment the X value by a specified amount.
6. Graticules: 2 dimensional grids.
7. Axes: 2 dimensional axes with major and minor tic marks.
8. Markers: A single character which can be placed anywhere on the window.
9. User Defined Characters: Characters drawn by the **UC** command (not to be confused with characters defined by the **SU** command).

Objects can have certain attributes assigned to them. These attributes apply to the entire object no matter how complex. Because of the nature of the object types, certain attributes apply only to one or more object types. An attempt to apply an incorrect attribute to an already-defined object yields an error. On the other hand, some attributes apply to all object types. A complete discussion of these attributes can be found under the **IT** command description in chapter 5.

Objects can be classed into text objects (text, markers, and user defined characters) and graphics objects (graphs, plots, graticules, and axes). Throughout this document for the sake of brevity, plots and graphs are referred to as "traces".

Text Objects The display language text command is the label command ***LB***. When sent text, the display will respond to the cursor control and character enhancement sequences described in appendix F. When sent text, each label can be thought of as a common terminal screen with the number of lines and characters per line determined by the configure label command ***CI*** for each label . See the ***LB*** command for more on this.

Text can be a single line or multiple lines. Whenever a multiple line label is to be created, it is required that the configure label command ***CL*** be used first. This command assigns a fixed number of lines with a fixed number of characters to the label. The display will assign a single line with a default number of characters per line if this is not done.

Displays have two character sets which can be selected with ***CA***, ***CS***, ***SA***, and ***SS*** commands. One set comes with the display (shown on page 189), the other may be defined by the user. The user set is created by the user with the "save user defined character" command ***SU***. A character is built and saved in character set 30 with this command. When usage of this character is desired, character set 30 is chosen and subsequent characters are taken from that set. Note that each window may have its own unique user defined character set 30. A blank will be printed if a character is not defined for the window in which it is being printed. The user can switch between the display set and the user set using the ***CA*** and ***CS*** commands as shown in the program example on page 46.

The characters drawn with the ***UC*** command are also text objects, as are markers. Markers are single characters which can be positioned on the window at any x,y coordinate using the ***MK*** command. They can be normal brightness or intensified (two intensified markers maximum). The character is chosen from the current character set.

Graphics Objects There are four types of graphics objects: graphs, plots, graticules, and axes. Drawing on the screen is accomplished by specifying x,y coordinates in either display units or user units. The pen is positioned by using these coordinates.

The most basic objects are graphs and plots (traces). They consist of a series of points connected by straight lines. The lines are drawn with the current line type, origin, intensity and pen number. Plots are specified by giving successive x,y coordinate pairs. Each of these points is called an endpoint. The display numbers the endpoints internally as they are drawn, starting at 0. Thus, any endpoint can be referenced for updating by the user using the trace pointer command ***TP***. Graphs are specified by successive y-coordinate values and the x-coordinate is calculated by adding the current x-coordinate of the pen location to a "delta-x" as specified by the ***DX*** command. Graphs allow faster drawing than plots. The endpoints of graphs are numbered just as plots are to allow referencing single points for updating. Graphs can also be panned left or right (see the ***PN*** command). Both graphs and plots can be interspersed with the pen up/pen

down commands ***PU*** and ***PD***. When the pen is up and its position moved, the subsequent endpoints and the lines between them are not drawn until the next pen down. See the ***GA***, ***GR***, ***PA***, ***PR***, and ***TP*** commands in chapter 5 for a complete explanation and examples of the drawing of traces on the screen.

The axis command ***AX*** draws two straight lines which intersect at right angles at a specified origin. The user can specify the length of the lines, the location of their intersection, and the position and length of major and minor tic marks.

A graticule (grid) can also be drawn on the window with the ***GT*** command. The user specifies the origin and size of the small boxes and how many are present in the x and y dimensions.

Markers are single characters which can be positioned on the window at any x,y coordinate using the ***MK*** command. They can be normal brightness or intensified (two intensified markers maximum). The character is chosen from the current character set.

Groups and Items (Referenced Graphics)

Objects can be grouped and the entire group or a single item can be manipulated. This allows an entire trace to be moved, or a single point to be changed, and so on. Information can be manipulated *after* it has been drawn, so the display requires a means of identifying and referring to particular objects. Individual objects may be referred to as *items* (see the ***GP*** and ***IT*** commands) where a single item can be any object. Once set to a certain object, the item cannot be set to another object type without being deleted first. However, if an object is not given a group and item number, that object will be *non-referenced* and cannot be changed later. Each item that is drawn on the screen has its own set of attributes (such as line type, pen number, rotation, character size, and and so on). The typical sequence of events for drawing something on the screen is:

1. identify the item using ***GP*** and ***IT***,
2. set the appropriate attributes, and
3. draw the object.

The procedure for modifying items that are already drawn on the screen is the same except for step 3, as follows:

1. identify the item using ***GP*** and ***IT***
2. set the appropriate attributes, and
3. re-draw the object.

Step 2 is partially or completely optional. That is, if the attributes of an existing item are already set as desired, it is not necessary to re-specify them. Similarly, step 3 may be optional if the object is already drawn as desired and only the attributes are to be modified. An attempt to redefine the object (change its type, for example from "graph" to "plot") without first deleting it (see ***DL***) will result in an error.

Occasionally it is desirable to manipulate many items at once (for example, switching between different screen presentations) without having to re-send all the data. In the display, this is accomplished by sorting items into groups (***GP*** command). Each group has its own set of attributes which are blinking, blanking, and origin. When a new group is created (by referencing a non-existing group) it will assume the default values for blinking (***BL***), blanking (***VW***) and origin (***OR***). All items exist in some group (even non-referenced items, as they are in group 0). If no group has been created but an item is referenced with ***IT***, group 1 is used. If a group *has* been created, the item goes into the most recently referenced group unless otherwise specified. Thus, this is one of the "history-dependent" command sets in the display language, as the most recently defined group or item is considered the "current" or "active" group and item.

To completely identify a given object requires a group number and an item number. The sequence:

identify group (***GP***), set origin (***OR***) ...

would position all the items within that group, so the effective screen position would be group origin plus item origin. The sequence:

identify group (***GP***), identify item (***IT***), set origin (***OR***) ...

would position only the specified item within the specified group. Origin setting and scaling operations can interact and produce unexpected results. See the ***OR*** and ***SC*** command descriptions and appendix E for a discussion of this.

If no group or item is specified, or if `GP0` or `IT0` is sent, the display is in "group 0" or non-referenced mode. In this mode, no object can be manipulated once it has been drawn. Also, display enhancements are not available. In this mode the display functions as a basic HP-GL plotter (see ***PA*** and ***GP***).

Memory

A display cannot draw an unlimited number of items. It is limited to 101 blocks of display list memory, which corresponds to about 12000 strokes. (See the ***RM*** command description). If this number is exceeded a "memory full" error message will result.

The display's display list (or "vector list") consists of 16K 16-bit words. The vector list is split into blocks, each of which is 128 words long. Thus there is a total of 128 blocks of memory available for graphics. The display uses 27 of these blocks for its own overhead in building various data structures and building the main vector list program itself.

This leaves 101 blocks of vector list available to do all desired graphics. This is important to know in dealing with groups and items. Each group takes at least one block of memory to set up all the item calls (also, there are only 16 groups). Each item takes at least one block of memory just to set up the line type, pen number, origin, and so forth. For this reason, the user should use groups sparingly and only when there are advantages gained by being able to manipulate the entire group with one command. For example, if all annunciators are in the same group then they can all be blanked at once.

Multi-Instrument Systems

To support multi-instrument systems, the display can simultaneously process data from up to five different *tasks*. To the user of the system, each of these tasks usually appears as a *window* on the screen (the exception is a keyboard task that has no corresponding graphics task). Four of these tasks are available to graphics and/or keyboard link responders, and can be viewed in the `show config` menu. These are tasks number 1-4, corresponding to windows number 1-4 (as in the `show config` screen and the ***BW*** and ***AW*** commands). The fifth task (corresponding to window number 5) is only available to a module (or computer) which has a *control link* to the display. Typically this task services HP-IB commands to the display, as HP-IB is serviced as though the HP-IB controller had a control link to the display.

Each graphics window and its window variables are independent of all other graphics windows. Any memory retained by the display concerning that window is not affected by other windows. In terms of referencing, group and item numbers apply only to that one window and no other. For example, if two spectrum analyzers each have half the screen, both can be updating their own group #1, item #4 which are separate objects with separate attributes. One of the items could be a graph while the other could be a label. Attributes of the items are also independent of other windows. For example, item #1 for one analyzer could be an intensified diamond shape marker while item #1 for the other analyzer could be a normal numeral 1 marker.

Subwindows assigned (***AW*** command) to one element are independent of subwindows assigned to another element. The first spectrum analyzer could assign subwindow #2 and the second analyzer could also assign subwindow #2 to some other module. These are entirely separate windows with separate P1,P2s and separate owners and they can be used separately.

Status bytes and masks are handled in a similar way by the display. The display display maintains a status byte for each element which has access to its resources. For example, the "key press" bit of the status byte is set only in the status byte going to the element which has the keyboard. Also, each status byte has its own mask as set by the input mask command ***IM***.

It is helpful to think of each window as representing a separate process, or task, within the display. The display is able to service several of these tasks simultaneously with no interaction between them.

Defaults and Power-Up State

When a display powers up or a window is initialized, certain parameters go to a default state. These are given in the description of the defaults command ***DF*** in chapter 5. These parameters are not stored in non-volatile memory, are not part of the display's learn string, and are not saved as part of any save state.

Certain parameters are stored in non-volatile memory. They are updated when they change and preserved during power-down.

These parameters are:

1. picture brightness[1] (***IA***),
2. hardcopy parameters: plotter P1/P2 (***PL***), printer and plotter addresses and "COPY IS" device (***PI***), key copy on or off (***KC***), hi res dumps on or off (***HR***), page eject on or off (***EJ***),
3. hard or soft HP-IB address setting,
4. HP-IB address (if soft address is set), and
5. screen save registers (***SV***).

HP-IB and Control Links

As mentioned elsewhere, HP-IB is usually serviced as though the HP-IB controller had the control link to the display. The serial poll status byte substitutes for the HP-MSIB ***STATUS*** command and **SRQ** is used to signal status changes but otherwise operation is very similar to HP-MSIB. The major differences between control links and the graphics or keyboard links established to modules are that more commands can be sent over a

[1] Except if the display powers up with brightness less than 9, it increases it to 9 so that it is guaranteed that there will be a visible picture on the screen.

control link than over graphics or keyboard links, and that the display is a control link responder, whereas it is a graphics and keyboard link initiator.

Unless the controller sends a ***BW*** command to assign itself to one of windows 1 through 4, the control link uses window 5. Window 5 does not show up in any screen configuration display and always has P1=(112,16) and P2=(911,383).

HP-IB gets the control link whenever data is sent to the display over HP-IB, *unless:*

- a module on HP-MSIB already has the control link *and* the display is not in remote, or
- One of windows 1-4 has been assigned to HP-IB using ***BW*** and the display is not in remote.

In the former case, if no window has been assigned HP-IB, the commands are ignored. In the latter case, HP-IB is assigned a graphics link (and a keyboard link if the ***BW*** command so specified). In this case, commands which require a control link will generate a Protocol Error when sent to the display.

If HP-IB is in local but no other control link exists, one will be assigned to HP-IB. Commands can then be sent to the display over HP-IB and will be obeyed as though the display were in remote (with some exceptions which are noted by command in chapter 5 *Command Reference*)[2].

The HP-IB **DEVICE CLEAR**[3] command aborts the command in progress (if any) and resets the parser. In ROM version 6.0 and later, it also clears the HP-IB output buffer and resets the label terminator (***DT***), the status and error masks, and the character sets, turns off pre-process mode, and clears **SRQ** and the serial poll status byte.

When the display enters remote mode on HP-IB, its menus are blanked if they were on the screen. This shows that the keyboard is disabled while in remote. If another module had the keyboard before the `DISPLAY` key was pressed, it will again be offered the keyboard when the display goes into remote.

---

[2]In ROM version 5.0, commands are *not* obeyed while in local.

[3]In BASIC `CLEAR @Display`.

5

COMMAND REFERENCE

This chapter contains an alphabetical listing of all commands available within the display. At the top of each page is listed a mnemonic, what the mnemonic stands for, and a brief description of the command. Note that all commands are two-letter commands, and that *only upper case characters are recognized* in commands.

The body of each command description contains detailed information for the command. This includes:

- whether it is from HP-GL or not,
- what links must be established to use it,
- syntax (pictorial representations and tabular description) for use of the command,
- the response (if any) to the command,
- examples showing possible uses of the command, and
- comments concerning the use of the command.

HP-GL

Many commands in the display language are taken from the Hewlett-Packard Graphics Language (HP-GL) for plotters. These commands are listed in a table in Appendix A. For each command in the command reference an indication is given as to whether the command is from HP-GL or not. In most cases these commands work the same for the display as they would for a plotter (although certain commands, like ***IN***, ***OO*** and ***OS***, interact with the system in more complex ways than would be true for a plotter).

Note that even if a display command is non-HP-GL, there *may* be an HP-GL command which uses the same two-letter mnemonic. Hence, when running a program which outputs HP-GL to the display, care should be taken to use only those display commands listed as also being HP-GL commands.

Links

As described in Chapter 3, the display communicates with HP-IB and with system elements by means of *links* established as defined in the *Communication Protocol Design*

Guide[1] for the HP 70000 System. For each command in the command reference, the link required to send that command is shown.

Syntax

The syntax is defined with a diagram and a description of the parameters. The display supports numeric parameters (*integer* only) sent in either ASCII or binary form (floating point numbers are not supported). Binary entry mode is selected by sending a `#I` after the command, as in the example below, and then sending the appropriate number of bytes to the display.

```
20 ! GA 0,260 in binary mode:  GA      256*0   +   0 , 256*1   +   4
30  OUTPUT @Display;"IN;PD;IT1;GA #I"&CHR$(0)&CHR$(0)&CHR$(1)&CHR$(4);
40  SEND 7;DATA 0 END     !     Assert EOI after last byte
```

Due to a quirk in the display's operation, the END condition, when sent over HP-IB, must come *after* the last byte, rather than *with* it. Hence, in the above example, line 30 could *not* have been sent as

```
OUTPUT @Display;"IN;PD;IT1;GA #I"&CHR$(0)&CHR$(0)&CHR$(1)&CHR$(4);END
```

or the last data point would have been ignored.

Once the display enters binary mode, it stays there through the end of the current command. Any numeric parameter may be sent to the display in binary form.

The **Bits** column in the syntax tables refers to the number of bits expected for the parameter if sent in binary form. The parameter input will complete after that many bits are read in, but a `Missing terminator` error will be generated if an ***END*** or an ASCII terminator (semicolon or linefeed) is not sent after the last byte of the last parameter (if there is a maximum number of parameters). Also, if no range is specified for a parameter, and the parameter is sent in ASCII (rather than binary) mode, but the parameter sent would not fit in the number of bits specified, the parameter will be read modulo $2^{number\ of\ bits}$. Imagine, for example, that the number 525 is sent but the parameter is specified as 8 bits, and no range is specified for the parameter. 2^8 is 256, and 525 MOD 256 is 13, so the parameter would be read as 13. No error would be generated. If a range is specified and an out of range number is received, the display sends a `Parameter out of range` error and sets the parameter to either the upper or lower limit, depending on whether the value sent was too high or too low (for example, if a 6 is sent and the upper limit is 4, the parameter is set to 4).

The syntax diagrams show the form for ASCII mode parameters. The binary form is the same except that in binary mode, no separators are used, and the terminator is an

[1] See page 6.

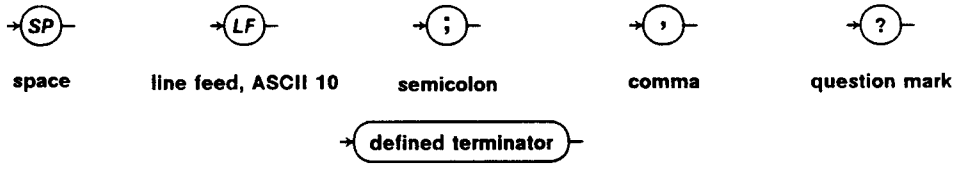
HP-IB ***END*** (***EOI*** asserted) or HP-MSIB ***END*** command. In ASCII mode, commas are used as parameter separators, and the command terminator is a semicolon or a line feed. Spaces are *definitely not* to be used as separators (see below). If two commas in a row are sent, the parser will supply a default for the missing parameter (for example, `BW 1,,,,0;`). The supplied defaults are specified for each command in the **Default** column of its syntax table. If no default is specified, 0 is assumed. The specified default will also be used if less than the total allowable number of parameters are sent for a given command (for example, `BW 1;`). Note that some parameters have no default, so those parameters *must* be sent (see ***AW***, ***DT***, and ***MA***, for example).

All numeric parameters are integers (rather than floating point numbers). Spaces and ***CR*** (ASCII 13) characters are ignored, except that a space *will* terminate the parsing of a number. Spaces should *not* be used as parameter separators, however, because once a space is encountered, the display reads the input stream until it finds the next non-blank character, and uses that character as the separator. Hence, if the sequence `GT 100 100;` is sent, the display will absorb the first 1 of the second 100 and use it as the separator, thus reading the second parameter as a 0!

Label-type commands have their own terminator, as explained in the ***DT***, ***LB***, ***ML***, and ***FC*** command descriptions. For all others, semicolon and linefeed are the terminators. These will reset the parser, so a command `A;` will be ignored, since the parser will be reset before it had two characters. A bad terminator will also reset the parser, so for the command `DE100;` for example, the 1 will be taken as the terminator and the 00; will be parsed as a command (which will generate an error).

The format for the syntax diagrams is as follows: commands and characters which should be sent exactly as shown are depicted in bold, uppercase letters, such as

Certain special symbols are used frequently in syntax diagrams. These are:



Parameters described in syntax diagrams are shown in normal-face, lower case letters, such as:

In such cases the user is expected to send some number or string of characters as that parameter. A description, defaults, and ranges for the parameters are shown in the table below the syntax diagram.

Response

If sending the command generates a response message from the display, the category labelled **Response** describes what that will be. It will be sent to the I/O channel which generated the message (for HP-IB, it will be sent the next time the display is addressed to Talk). In general, responses are terminated with a ***CR LF*** and an ***END*** condition. For HP-MSIB, the ***END*** condition is the HP-MSIB ***END*** message (see the *Communication Protocol Design Guide* for details). For HP-IB, the ***END*** condition consists of asserting EOI with the last byte of the transmission. There are a few exceptions to this rule, and they are noted in the text.

Example

For each command, an example is shown. The examples are typically program lines or commands from HP Series 200/300 computers running the BASIC language system. They illustrate usage of the command over HP-IB. Over HP-MSIB, the usage may differ slightly, but the examples serve to demonstrate salient features of each command.

Many of the example programs are included on the disc (HP Part number 5010-1564) which accompanies this manual. The files are stored on this disc in ASCII form so they must be loaded with the BASIC "GET" command. The filename is included on the first line of each example.

Note that the HP BASIC statement `OUTPUT` is defined to implicitly send a ***CR LF*** at the end of the statement, unless suppressed by a trailing semicolon. Hence, the statement

```
20     OUTPUT @Display;"IN"
```

Will send four bytes to the display: I, N, ***CR*** and ***LF***. Since the ***LF*** is a terminator, the display will accept and act on the command, just as though a semicolon had been sent with it, as below:

```
20     OUTPUT @Display;"IN;"
```

It is recommended, however, that the latter form *always* be used, with the semicolon *explicitly* included, to avoid problems with `OUTPUT USING` statements, and also because commands sent on HP-MSIB will not have the benefit of the implicit ***LF*** from HP BASIC, and hence will not have any terminator without the semicolon (unless an HP-MSIB ***END*** is also sent). You may wish to suppress the ***CR LF*** added on by HP BASIC by putting another semicolon at the end of the entire `OUTPUT` statement, as:

```
20     OUTPUT @Display;"IN;";
```

In many of the examples, the ***DE*** and ***IN*** commands are used to set up the example and guarantee a predictable result. In actual practice, these would only be used to set up the display, not every time a command sequence must be sent.

In some examples, a line will appear such as

```
OUTPUT @Display;"GP1;IT2;IS50;"
```

which is outside a program. Sending such a line will only work if the last BASIC program to run is in a PAUSEd state; otherwise the I/O path `@Display` will be undefined. If that is the case, simply substitute the display's HP-IB address (for example, 704) for `@Display`, as

```
OUTPUT 704;"GP1;IT2;IS50;"
```

Comments

Further detail about each command is found in the **Comments** section. See also appendix A for a listing of commands organized by function and group (for example, HP-GL, non-HP-GL, and so on).

Firmware Revision Dependencies

The firmware revision history of the display is:

Version	Datecode
Rom Version 5.0	(no datecode - issued March, 1985)
Rom Version 6.0	860625

Where applicable, key differences between the firmware versions are noted.

Keyword Dictionary

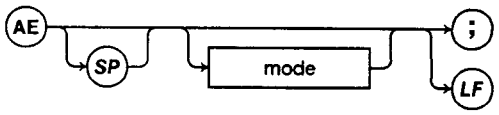
The alphabetical keyword dictionary follows:

AE

ALPHA ENTRY allows a display to simulate a full ASCII keyboard.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0 = off, otherwise on	0	-	8

Response: None

Example:

```
OUTPUT @Display;"AE1;"
```

Alphanumeric entry is enabled. Softkey 7 (lower right) and the knob no longer go to the owner of the keyboard, but instead are used by the display for simulating ASCII keyboard entry.

Comments:

When ***AE*** is required, the display will re-label softkey 7 (see Figure 5.1 on page 100) to ENTER. It will also display a portion of its character set in a window at the bottom of the screen. The knob is then enabled to underline one character at a time. The manual user selects a character to be sent to the active element by turning the knob until the correct character is underlined (turning the knob will cause the remainder of the character set to scroll into the window). Then the user pushes ENTER and the desired key code is sent to the active element. The key code sent is $256 + n$ where n represents the number of the selected character (see the table in appendix D). For characters 0 through 127, n represented the USASCII code for the character. For example, if "A" were selected, the code 256 + 65 would be sent, since 65 is the ASCII code for "A". This allows simulation of a full ASCII keyboard. All key presses except softkey 7 perform their standard functions.

There are two ways to exit the ***AE*** mode:

(1) by the keyboard owner or controller sending `"AE0"`, and

(2) by breaking the keyboard link.

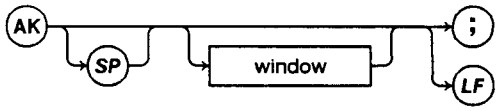
When the mode is exited, the display will blank softkey 7 and restore the portion of the graphics window which was blanked. It is the responsibility of the module program to relabel softkey 7 (but not until ***AE*** mode is exited).

AK

ASSIGN KEYBOARD assigns the keyboard to the instrument which owns the specified window.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
window	window number	window which currently owns the keyboard (1 if no window owns the keyboard)	1 through 4	8

Response: None

Example:

```
OUTPUT @Display;"AK3;"
```

The keyboard is assigned to the instrument which has graphics window number 3.

Comments:

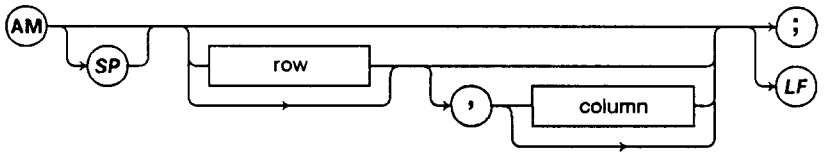
See page 22 and page 23 for a discussion of the display's graphics windows. Window number 5 (the controller window) may not be explicitly assigned the keyboard, but it implicitly owns it anyway, since *any* command may be sent across a control link (see the *Communication Protocol Design Guide*[2] for more on links).

[2]See page 6

ADDRESS MAP initiates the display's address map function.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
row	row to place active block	-1	-1 through 7	8
column	column to place active block	-1	-1 through 31	8

Response: None

Example:

```
10     ASSIGN @Display to 704
20     OUTPUT @Display;"AM 4,18;"
30     END
```

The address map function of the display is initiated and the screen is filled with a map of elements in the HP-MSIB system. The cursor (highlighted block) is placed at row 4, column 18. Other addresses may be viewed by turning the knob.

Comments:

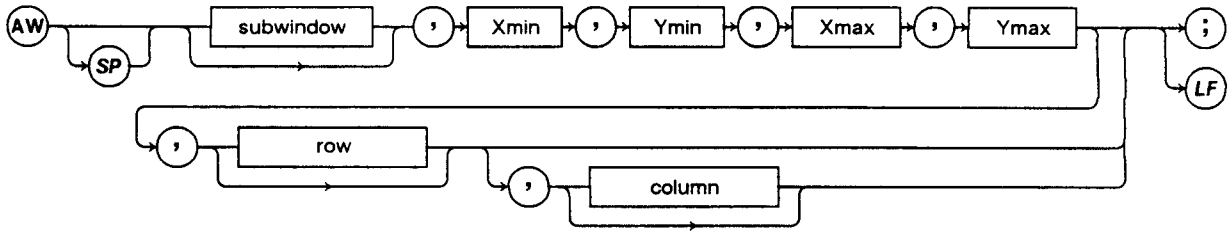
If either row or column is -1, the address map function is ended. The windows which were on the screen when ***AM*** was invoked will return to the screen when the address map function is ended. The ***AM*** function must be ended before new functions can be entered. See also page 14.

AW

ASSIGN SUBWINDOW defines a subsection of an already defined window to be used by a slave.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
subwindow	number of subwindow	1	1 or 2	8
Xmin	x-coordinate of lower left corner of subwindow in user units	-	after un-scaling, in display units, x-coordinate of lower left corner of window to x-coordinate of upper right corner of window −201	16
Ymin	y-coordinate of lower left corner of subwindow in user units	-	after un-scaling, in display units, y-coordinate of lower left corner of window to y-coordinate of upper right corner of window −101	16
Xmax	x-coordinate of upper right corner of subwindow in user units	-	after un-scaling, in display units, x-coordinate of lower left corner of window + 200 to x-coordinate of upper right corner of window	16

(continued)

Item	Description	Default	Range	Bits
Ymax	y-coordinate of upper right corner of subwindow in user units	-	after un-scaling, in display units, y-coordinate of lower left corner of window + 100 to y-coordinate of upper right corner of window	16
row	row address on HP-MSIB of slave to get subwindow	0	0 through 7	8
column	column address on HP-MSIB of slave to get subwindow	31	0 through 31	8

Note: For Xmin, Ymin, Xmax, and Ymax above, the values in the **Description** field are in user units as scaled for the requestor's window, while the values in the **Range** field are in physical display units. Thus, the parameters coming in must first be un-scaled (converted to actual display coordinates in display units) according to the scale factors set up by the most recent **SC** command, then tested against the limits shown in the **Range** column.

Response: None

Example:

```
10     ASSIGN @Display to 704
20     OUTPUT @Display;"AW1,100,100,600,200,0,18;"
30     END
```

This command defines a rectangular subwindow (all windows are rectangular) numbered 1 (of a possible 2) within the window belonging to the sender of the ***AW*** command, whose lower left vertex is at (100,100) in user units (units set for the main window with the **SC** command), and whose upper right vertex is at (600,200) in user units. It is assigned to the element at address 0,18.

Comments:

See the discussion of the ***BW*** command for more on the display's windows. The ***AW*** command is used to set up a "window-within-a-window". When the display receives an ***AW***, if there is enough memory and no more than three tasks (plus HP-IB) running, and the requested window lies within the requestor's window, a new window is set up and

assigned to the module at the address indicated, as a subwindow to the requestor's window. A column address of 31 (default) purges the subwindow (thus `AW1;` deletes subwindow 1).

The control task (usually window 5) may be assigned a subwindow. Note that subwindows are not seen in the display's `show config` screen (nor is window 5, for that matter).

Note that although there are no defaults for Xmin, Xmax, Ymin, or Ymax, if they are not sent (by using commas with nothing between, as `AW 1,,,,,0,18;` then the parser will report an error but will also provide parameters equal to the lower limits for each parameter, namely P1X=window P1X (lower left corner), P2X=P1X+200 (in display units), P1Y=window P1Y, and P2Y=window P1Y+100. Thus, if `AW 1,,,,,0,18;` is sent to the default window (X=112 to 911, Y=16 to 383) the subwindow will be built from X=112 to 312, Y=16 to 116, and an error will be reported. Note that the default window corners which result from not sending *any* of these parameters (for example, `AW 1;`) will be irrelevant, since the column address will default to 31, which purges the subwindow.

The subwindow being defined must be contained within the window of the instrument doing the defining (subwindow), and it must not be smaller than 200 × 100 in display units as shown by the **Range** above. If the subwindow indicated is already defined, it will be re-defined by this command.

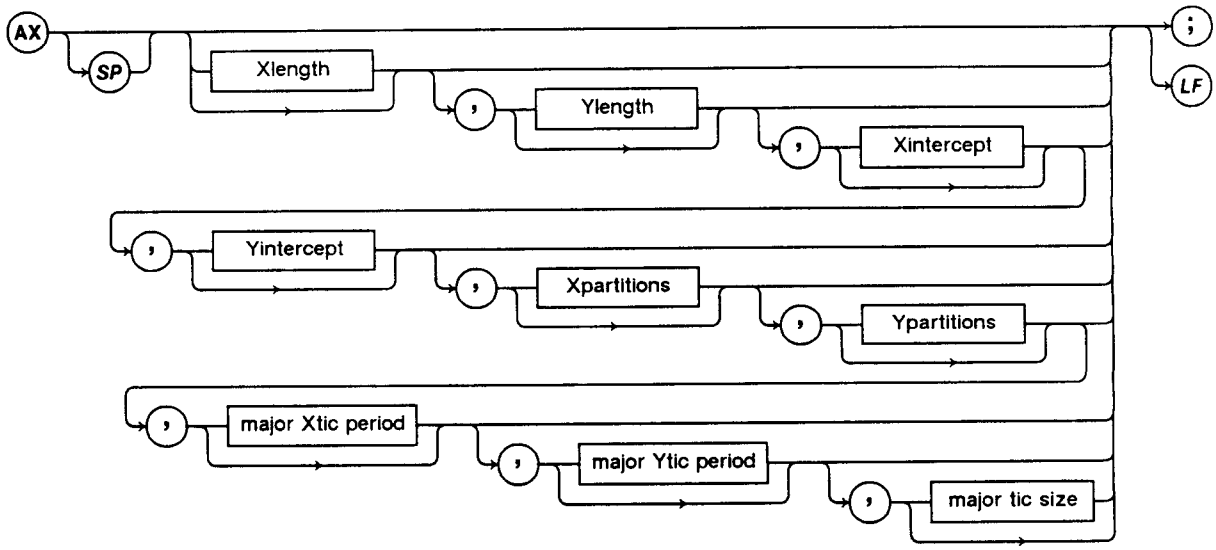
Firmware Revison Dependencies:

In ROM Version 5.0, if the column parameter defaults because two separators with nothing between them was sent, as opposed to having the command terminate before this parameter was sent, then it defaults to 0 instead of 31.

AXIS draws axes.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
Xlength	length of x-axis	800	0 through 32767	16
Ylength	length of y-axis	320	0 through 32767	16
Xintercept	where y-axis crosses x-axis	0	0 through Xlength	16
Yintercept	where x-axis crosses y-axis	0	0 through Ylength	16
Xpartitions	number of partitions on the x-axis	0	0 through 32767	16
Ypartitions	number of partitions on the y-axis	0	0 through 32767	16
major Xtic period	number of minor tics per major tic	0	0 through 32767	16
major Ytic period	number of minor tics per major tic	0	0 through 32767	16
major Xtic size	length of major Xtic	16	0 through 32767	16

Xintercept, above, means "how many points along the x-axis does the y-axis cross it"; likewise, Yintercept means "how many points up the y-axis does the x-axis cross it".

Response: None

AX

Example:

```
1    ! Program "AX"
10     ASSIGN @Display to 704
20     OUTPUT @Display;"DE;IN;GP1;IT1;"
30     OUTPUT @Display;"OR500,200;"
40     OUTPUT @Display;"AX500,250,250,125,20,10,10,5,20;"
50     END
```

This program provides that item 1 in group 1 will be axes such that the x-axis shall have a length 500 and the y-axis shall have a length of 250. The axes will intersect each other at a point which is 250 units to the right on the x-axis and 125 units up on the y-axis. This intersection will be placed on the screen at the point of the currently defined origin, in this case the point 500,200 (see the ***OR*** command). There will be twenty partitions on the x-axis and ten partitions on the y-axis. Every tenth minor tic on the x-axis will be replaced with a major tic and every fifth minor tic on the y-axis will be replaced with a major tic. Major tics will be twenty x-axis units long and minor tics will be half that long. (Minor tics are always 1/2 as long as major tics.)

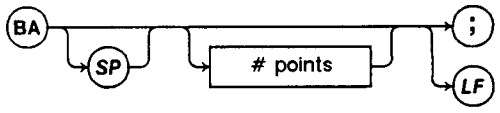
Comments:

The active item type *must* be an axis or a *new* item or an **illegal parameter** error is generated (except for non-referenced graphics). All parameters are optional, and in user units. The partitions on the x and y axes are separated by major and minor tics. The major tics on the y-axis will be the same apparent size as the major tics on the x-axis regardless of the y-axis scale-factor. The x-axis scale factor is used to calculate their size. The intersection of the axes will be at the origin of the item (referenced) or of the window (non-referenced).

BLANK AHEAD sets the number of points in the trace which will be blanked to the right of the last point plotted.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
# points	number of points to blank after last point plotted	0	≥ 0	16

Response: None

Example:

```
1    ! Program "BA1"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;IT1;OR 150,100;DX50;"
30     OUTPUT @Display;"GA0,40,20,60,50,150,40,60,30,20,50;"
40     OUTPUT @Display;"BA2;TP4;GA50;" ! Re-send point 4 to blank next 2
50     END
```

This example draws a trace in the middle of the screen and then blanks two segments in the middle of it. The result is identical to that generated by the following lines:

```
1    ! Program "BA2"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;IT1;OR 150,100;DX50;"
30     OUTPUT @Display;"GA0,40,20,60,50;PU;GA 150,40;PD;GA60,30,20,50;"
40     END
```

Comments:

A "trace" is a collection of interconnected points, all of which make up a single graphics item. Traces are drawn with the ***GA***, ***GR***, ***PA*** and ***PR*** commands. The ***BA*** command sets the number of points to be blanked in a trace, beyond the point most recently plotted. It is usually used in conjunction with the ***TP*** command. Its value is that it allows any point on an already drawn trace to be updated, and to have the update emphasized by blanking several points on the trace after the updated point. The

blanking is accomplished by "lifting" the pen for the next n points[3] of trace data (unless the trace ends after the current endpoint $+n$). Then, as further graph or plot commands relating to that trace come in, the blanked area moves one point farther along the trace, and the new points are plotted. Note that this *can* slow down the trace processing time as the data is read in.

This command does *not* depend on the pen being down, that is, the blanking works even if the pen is up at the endpoint where the **BA** command comes in. Thus, the lines below are equivalent to the examples above.

```
1    ! Program "BA3"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;IT1;OR 150,100;DX50;"
30     OUTPUT @Display;"GA0,40,20,60,50,150,40,60,30,20,50;"
40     OUTPUT @Display;"BA1;TP5;PU;GA0"  ! Send new point 5 with pen up
50     END
```

The **BA** command sets the blank-ahead for the active item. It is ignored in non-referenced graphics mode (group 0) or if no group is active. This command also has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;BA10;` will have no effect).

Because the range of parameters includes only positive numbers, small negative numbers are interpreted as large positive numbers, and hence negative parameters should be avoided. They will not, however, generate errors.

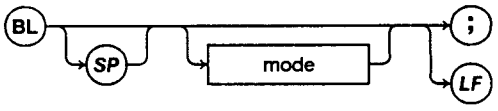
The default blank-ahead used when an item is created is 0.

[3]where n is the parameter sent

BLINK ON/OFF turns the blink mode on or off.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=blink off, else on	1	-	8

Response: None

Example:

```
OUTPUT @Display;"GP1;IT1;BL1;"
```

This will cause item 1 of group 1 to start blinking. The blink will remain active until a `IT1;BL0` command is received (or just a `BL0` if item 1 is still the active item).

```
OUTPUT @Display;"GP1;BL1;"
```

This will cause the entire group 1 to start blinking regardless of whether individual objects are activated to blink or not.

Comments:

This command turns blink mode on or off for the currently active group or item (see *Groups and Items* section one page 20 in Chapter 4). It is meaningless for non-referenced graphics.

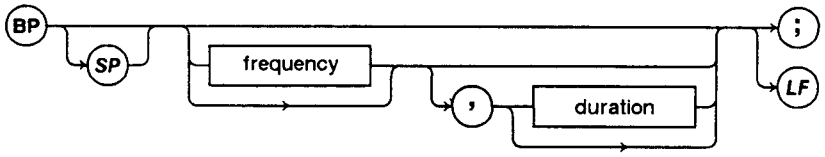
The blinking attributes of groups or item "OR" together, that is, if either an item or the group it is in are set to blink, it will blink.

BP

BEEP causes the display to beep with the specified frequency and duration.

HP-GL Yes
Link Required Any Link

Syntax:



Item	Description	Default	Range	Bits
frequency	beep tone as described below	2000	500 through 2000	16
duration	beep duration as described below	100	0 through 1000	16

Response: None

Example:

```
OUTPUT @Display;"BP1000,30;"
```

This will cause a beep of 15 ms in length at a frequency of 1040 Hz.

Comments:

No error is generated if the parameters are out of range, they are simply set to the appropriate upper or lower limit depending on whether the parameter was high or low. Individual parameters will produce a beep tone for a duration of time according to the following definition:

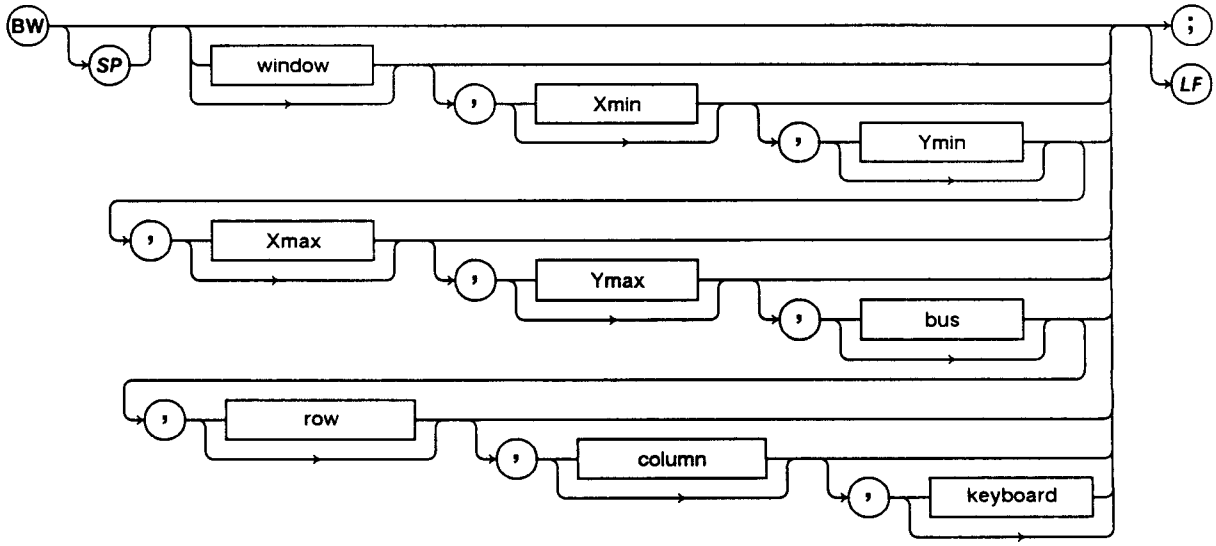
Parameters		Results	
frequency	duration	approximate tone	length of time
500	0-15	520 Hz	8 ms
501 - 571	16 - 31	595 Hz	15 ms
572 - 666	32 - 63	695 Hz	31 ms
667 - 800	64 - 127	830 Hz	61 ms (*)
801 - 1000	128 - 255	1040 Hz	123 ms
1001 - 1333	256 - 511	1400 Hz	246 ms
1334 - 2000	512 - 1000	2080 Hz (*)	492 ms

(*) default value

BUILD WINDOW builds a window on the screen, assigns an HP-MSIB or HP-IB element to that window, and determines whether to also assign the keyboard to that element.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
window	number of window	1	1 through 4	8
Xmin	x-coordinate of lower left corner of window in display units	112	0 to 823	16
Ymin	y-coordinate of lower left corner of window in display units	16	0 to 283	16

(continued)

BW

Item	Description	Default	Range	Bits
Xmax	x-coordinate of upper right corner of subwindow in display units	911	Xmin + 200 to x-coordinate of upper right corner of window	16
Ymax	y-coordinate of upper right corner of subwindow in display units	383	Ymin + 100 to y-coordinate of upper right corner of window	16
bus	0=HP-IB, 1=HP-MSIB	1	0 or 1	8
row	row address on HP-MSIB of module to get window	0	0 through 7	8
column	column address on HP-MSIB of module to get window	0	0 through 31	8
keyboard	1 = assign keyboard to module	0	0 or 1	8

Response: None

Example:

```
OUTPUT @Display;"BW4,20,30,320,330,1,5,27,1;"
```

This command builds window 4, whose lower left vertex is at (20,30) and whose upper right vertex is at (320,330). The window is assigned to whatever element is at the HP-MSIB address 5,27. The element is assigned the display's keyboard.

Comments:

There are 5 windows total in the display: 1 through 4 can be assigned with ***BW***; window 5 (the controller window) is used for the control link and is assigned to the owner of that link. HP-IB gets the control link whenever data is sent to the display over HP-IB, *unless:*

- a module on HP-MSIB already has the control link *and* the display is not in remote.
- One of windows 1-4 has been assigned to HP-IB and the display is not in remote.

Window 5 may not be explicitly built via ***BW***, or assigned the keyboard, but *any* command may be sent across a control link (see the *Communication Protocol Design Guide* for more on links). See also page 22 and page 23 for a further discussion of the display's graphics windows and control links..

All windows are rectangular. Xmin, Ymin are the coordinates of the lower left corner (P1) and Xmax, Ymax are the coordinates of the upper right corner (P2). Besides the ranges shown above for these coordinates, the window being defined must not be smaller than 200 × 100 in display units (the coordinate values sent are in display units) and can not go beyond 1023 in the x-direction and 383 in the y-direction (see appendix B). If the window indicated is already defined, it will be re-defined by this command, the window cleared, and all task variables for that window re-initialized just as though an ***IN*** command had been sent.

If the keyboard parameter is 1, the module assigned the window will also be assigned the keyboard. If there is more than one window, the display will draw borders around the windows and the window with the keyboard will get a solid border while the other windows will get a dotted border. If the display menus are on the screen, then no window will have a solid border because the display has the keyboard. (To actually draw a border around the window using graphics commands, use the method shown in the ***CA*** example or the ***GT*** example).

Graphics done in the window will be in display units (dots) relative to the lower left corner (0,0) of the *screen* until an ***SC*** command is sent (see ***SC*** and appendix E).

The bus, row, and column parameters determine which I/O channel will be used to send commands to the window. If HP-MSIB, the element at the HP-MSIB row address, HP-MSIB column address is assigned to that window. If HP-IB, the row and column addresses are ignored.

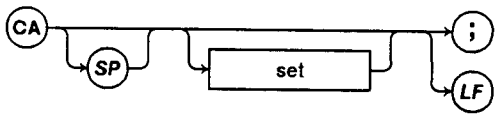
The P1,P2 coordinates and link owners for all current windows are remembered through a power-down by the CMOS RAM.

CA

DESIGNATE ALTERNATE CHARACTER SET designates the character set that is to be used as the alternate character set.

HP-GL Yes
Link Required Any Link

Syntax:



Item	Description	Default	Range	Bits
set	character set number	0	-	8

Response: None

Example:

```
1    ! Program "CA"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"BW1,112,16,911,383,0;" ! Build window to draw in
40     OUTPUT @Display;"SC0,100,0,100;"        ! Scale window to 100 x 100
50     OUTPUT @Display;"GP2;IT1;PA0,0,100,0,100,100,0,100,0,0;"   !Draw frame
60     OUTPUT @Display;"SUE,8,6,99,0,2,-8,0,4,-4,-4,-4,8,0,0,2;"  !(See SU)
70     OUTPUT @Display;"CS0;"   !DESIGNATE STANDARD CHARACTER SET = SET 0
80     OUTPUT @Display;"CA30;" !DESIGNATE ALTERNATE CHARACTER SET = SET 30
90     OUTPUT @Display;"SS;"    !SELECT STANDARD CHARACTER SET;START USING IT
100    OUTPUT @Display;"GP1;IT1;OR 10,50;DT@;LBEEEEEE@;"
110    OUTPUT @Display;"SA;"    !SELECT ALTERNATE CHARACTER SET;START USING IT
120    OUTPUT @Display;"GP1;IT2;OR 30,50;DT@;LBEEEEEE@;"
130    OUTPUT @Display;"CS30;CA0;" !SWITCH THE STANDARD/ALTERNATE SETS
140    OUTPUT @Display;"SS;"    !SELECT NEW STANDARD SET AND USE IT
150    OUTPUT @Display;"GP1;IT3;OR 50,50;DT@;LBEEEEEE@;"
160    OUTPUT @Display;"SA;"    !SELECT NEW ALTERNATE SET AND USE IT
170    OUTPUT @Display;"GP1;IT4;OR 70,50;DT@;LBEEEEEE@;"
180    END
```

The ***CA***, ***CS***, ***SA***, and ***SS*** commands are demonstrated in the program above. Four labels are drawn, one by one across the screen. The first is drawn in the standard set, the second in the alternate set, then the two sets are swapped and two more labels are drawn, the first in the new alternate set and the second in the new standard set.

Comments:

The **CS** command is used to designate the character set that is to be used as the standard character set and the **CA** command is used to designate which character set is to be used as the alternate character set. The controller or instrument selects between the standard and the alternate character set by sending either the **SS** command or the **SA** command.

The character sets available are listed as follows:

number	Character set
0	US ASCII with MMS extensions (See appendix D)
30	Saved User defined Character Set (See **SU** command)

Note that the sequence

```
IN;CS30;DT@;LBFRED@;
```

will *not* cause the word FRED to be written using character set 30, because until either an **SS** or **SA** command is sent, the current character set remains in force (which would be set 0 in this case because of the IN command). The sequence

```
IN;CS30;SS;DT@;LBFRED@;
```

which includes an **SS**, *will* select character set 30 for labeling.

Firmware Revision Dependencies:

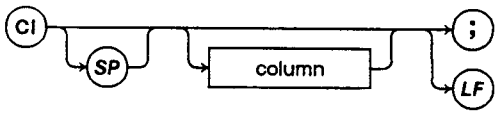
In ROM Version 5.0, the default parameter is random.

CI

SELECT INSTRUMENT establishes a link with an instrument.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
column	column to start search in	0	0 to 30	16

Response: None

Example:

```
OUTPUT @Display;"CI9;"
```

A search begins for a module which can take over the display screen. The search is begun at column 9.

Comments:

This command allows a SELECT INSTR (see page 13) to be initiated remotely, just as when the SELECT INSTR key is pressed. A search of row 0 is begun, starting at the specified column. The first module which accepts it is given the entire graphics screen of the display. The search wraps around at 30 and starts again at 0 until all locations have been searched. If no parameter is sent, and any window on the display is already assigned to a module, the search begins at the next address above that module's. If no window is assigned and no parameter is sent, the search starts at 0.

It is not advisable to try to transfer data to or from the display via HP-IB while this search is in progress, since at each unsuccessful address, all errors are cleared and all HP-IB task variables are re-initialized, which makes parsing unreliable. Bit 4 of the status byte can be monitored (although **SRQ** cannot be looked for since every time HP-IB restarts the status mask is reset). This bit is cleared when the ***CI*** begins and set when it ends (see appendix C). The **OG** command can then be used to find out which module, if any, was selected. The program below can be used for this purpose:

```
1    ! Program "CI"
10     ASSIGN @Display TO 704
20     Display_ready=4
30     OUTPUT @Display;"CI;"
40     DISP "Searching for module..."
50     REPEAT
60     UNTIL BIT(SPOLL(@Display),Display_ready)
70     OUTPUT @Display;"OG;"
80     ENTER @Display;Row,Column
90     BEEP
100    IF Row=-1 THEN
110      DISP "No module found."
120    ELSE
130      DISP "Module found at Row ";VAL$(Row);", Column ";VAL$(Column)
140    END IF
150    END
```

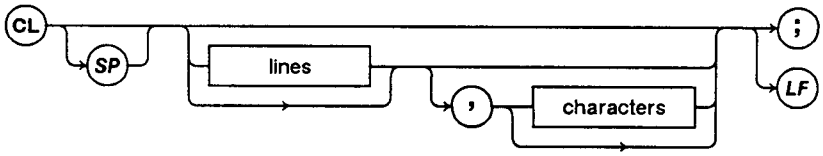
Firmware Revision Dependencies: Revision 6.0 and later only.

CL

CONFIGURE LABEL sets up a single line or multi-line label.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
lines	number of lines in label	1	1 through 24	8
characters	number of characters per line	35	1 through 68	8

Response: None

Example:

```
OUTPUT @Display;"GP1;IT1;CL2,8;OR200,100;"
```

This command defines group 1, item 1 as a text "label" with two lines of 8 characters each to be placed at the point 200,100. A label is the basic unit of text in the display. The label will remain blank until filled by the ***LB*** command. (See ***LB*** command and appendix D).

Comments:

This instruction is used to set up a new label. Since many different items of text may appear anywhere on the screen, the user needs to configure their x,y dimensions, which this command allows. The instruction specifies the number of lines in the label and the number of characters in each line. The label is initialized to all blanks. The label is then written or modified with the ***LB*** command. Scrolling of single and multiple line labels is discussed under the ***LB*** command.

Note that the default origin of (0,0) places the lower left corner of the first character cell of the top line of the label at 0,0 on the screen. Thus, if no location is given for a label of two or more lines, only the first line will be seen, as the second line will be off the bottom of the screen.

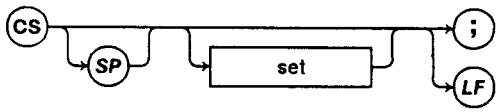
An attempt to configure a non-text item (for example, ***PA***) with ***CL*** will generate an **illegal parameter** error. This will also occur if a ***CL*** is received in non-referenced graphics (GP0).

CS

DESIGNATE STANDARD CHARACTER SET designates the character set that is to be used as the standard character set.

HP-GL Yes
Link Required Any Link

Syntax:



Item	Description	Default	Range	Bits
set	character set number	0	-	8

Response: None

Example:

See the ***CA*** command for an example program that demonstrates the use of the ***CA***, ***CS***, ***SA*** and ***SS*** commands.

Comments:

The ***CS*** command is used to designate the character set that is to be used as the standard character set and the ***CA*** command is used to designate which character set is to be used as the alternate character set. The controller or instrument selects between the standard and the alternate character set by sending either the ***SS*** command or the ***SA*** command. The standard character set is also used when default conditions exist (power-on, ***DF***, ***IN***, ***CI***, SELECT INSTR, DISPLAY PRESET, DEVICE CLEAR).

The character sets available are listed as follows:

number	Character set
0	US ASCII with MMS extensions (See appendix D)
30	Saved User defined Character Set (See ***SU*** command)

See the discussion of ***CA*** for the proper command sequence to use to select character sets.

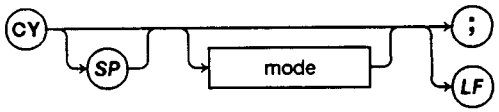
Firmware Revision Dependencies:

In ROM Version 5.0, the default parameter is random.

COPY Initiates a hardcopy print or plot as if the PRINT or PLOT key was pressed.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=use plotter's limits, 1=use stored P1,P2	0	-	8

Response:

The display dumps its screen to the printer or plotter on HP-IB or HP-MSIB as established by the ***PI*** command or the COPY IS PRT/PLT softkey. Commands sent to a printer are compatible with the HP Thinkjet printers, and commands sent to a plotter are in HP-GL.

Comments:

The command ***CY*** copies the entire graphics window to the selected device. If KEYCOPY is ON in the display's define hardcpy menu, or a KC 1 command has been sent, the menu windows, status window, and character line are also copied.

If the copy device is a plotter, and the mode parameter is 0, the display asks the plot device for its plot limits and adjusts the output data to put the plot within those limits. If the copy device is a plotter, and the mode parameter is 1, the display uses the P1,P2 it has stored in its memory (set with the ***PL*** command or the define hardcpy menu). If the copy device is a printer, the mode parameter is ignored.

The mode parameter is overridden, and the stored P1,P2 are used, if:

- the destination device established by the ***PI*** command or the COPY IS PRT/PLT softkey is defined to be Listen Only
- the display's SYSTEM CONTROLLER switch is off (6.0 only)
- the destination device is on HP-MSIB

Pressing any key aborts the dump.

Firmware Revision Dependencies, dumps over HP-IB:

ROM Version 5.0 :

SYSTEM CONTROLLER off: the display checks the bus after 1/2 second for a controller by looking for ATN or REN true, and if it finds one, aborts. If not, addresses plotter or printer as though display were a controller.

SYSTEM CONTROLLER on: the dump will not work.

ROM Version 6.0 and later:

SYSTEM CONTROLLER on: the display waits for REN and ATN to go false, then addresses printer or plotter and executes dump. If the plotter or printer is defined to be Listen Only, the display will not try to address anything but will function as a Talk Only device and the dump will proceed (this will not work if there is any other active controller on HP-IB).

SYSTEM CONTROLLER off: the display must be addressed to talk by the controller and the printer or plotter addressed to listen. The exception is the case where the dump device is a Listen Only plotter or printer, and is so defined via the ***PI*** command or the COPY IS PRT/PLT softkey. In that case, the display will function as a Talk Only device and the dump will proceed (this will not work if there is any other active controller on HP-IB). The stored P1,P2 are always used. Also, any characters sent to the display over HP-IB, other than semicolon, line feed, carriage return, or space, will abort the dump, assuming that the dump was invoked over HP-IB.

In general, software which worked on Version 5.0 with the SYSTEM CONTROLLER switch off will work on Version 6.0 with the SYSTEM CONTROLLER switch *on*.

In Versions 5.0 and 6.0, DEVICE CLEAR should not be sent during a dump, as it may lock out further keyboard interaction, requiring a power cycle. Also in Versions 5.0 and 6.0, no ***END*** is sent at the end of the dump, so only by checking the status byte can one determine whether a dump has concluded.

Examples:

These examples are for ROM Version 6.0 only.

The program below works when the SYSTEM CONTROLLER switch is not set:

```
1    ! Program "CY1"
10     ASSIGN @Plotter TO 705
20     ASSIGN @Display TO 704
30     REMOTE 7
40     CLEAR @Display
50     OUTPUT @Display;"IM 255,16;"     ! Set display to assert SRQ on READY
60     OUTPUT @Display;"PI 1,0,5;"      ! Specify dump to plotter, HP-IB 5.
70     OUTPUT @Display;"CY;";           ! Invoke copy.
80     SEND 7;UNL TALK 4 LISTEN 5 DATA  ! Address devices
90   Continue:  !
100    ON INTR 7 GOTO Srq_int           !
110    ENABLE INTR 7;2                  ! ENABLE SRQ INTERRUPTS
120    DISP "Waiting for plot to finish"
130  Idle:GOTO Idle                     ! Wait for interrupt
140  !
150  Srq_int:!
160    A=SPOLL(@Display)                      ! See if plot finished
170    IF NOT (BIT(A,4)) THEN GOTO Continue ! No, continue
180    REMOTE 7                               ! Yes, wrap up
190    DISP "All done."
200    END
```

The program below works when the SYSTEM CONTROLLER switch is set:

```
1    ! Program "CY2"
10     ASSIGN @Plotter TO 705
20     ASSIGN @Display TO 704
30     REMOTE 7
40     CLEAR @Display
50     OUTPUT @Display;"IM 255,16;"     ! Set display to assert SRQ on READY
60     OUTPUT @Display;"PI 1,0,5;"      ! Specify dump to plotter, HP-IB 5.
70     OUTPUT @Display;"CY;";           ! Invoke copy.
80     LOCAL 7                          ! Get computer...
90     SEND 7;UNT UNL DATA              !            ...off HP-IB
100  Continue:  !
110    ON INTR 7 GOTO Srq_int           !
120    ENABLE INTR 7;2                  ! ENABLE SRQ INTERRUPTS
130    DISP "Waiting for plot to finish"
140  Idle:GOTO Idle                     ! Wait for interrupt
150    !
160  Srq_int:!
170    A=SPOLL(@Display)                      ! See if plot finished
180    IF NOT (BIT(A,4)) THEN GOTO Continue ! No, continue
190    REMOTE 7                               ! Yes, wrap up
200    DISP "All done."
210    END
```

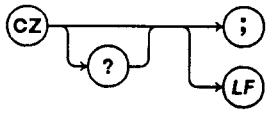
The SYSTEM CONTROLLER switch is only checked at power-up, so the statement "SYSTEM CONTROLLER switch set" means "SYSTEM CONTROLLER switch set at power-up."

CZ

OUTPUT CHARACTER SIZE asks the display for the size of the characters in the currently active item or group.

HP-GL	No
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response:

Xpercent,Ypercent: two unsigned ASCII integers separated by commas and terminated by ***CR LF*** and ***END*** (see page 28). They represent the current character size for the window from which the ***CZ*** was sent, in a percent of the x and y dimensions (width and height) of the window. See the ***SI*** and ***SR*** command descriptions and appendix D for more on character sizes.

Example:

```
1    ! Program "CZ"
10     ASSIGN @Display to 704
20     OUTPUT @Display;"DE;IN;CZ;"
30     ENTER @Display;Xpercent,Ypercent
40     DISP Xpercent,Ypercent
50     END
```

The values **Xpercent, Ypercent** will appear on the screen of the controller.

Comment:

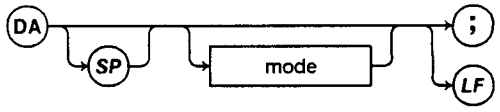
For the normal, default window, with character size 0 selected, the values returned for Xpercent and Ypercent respectively will be 1.87 and 4.34, because the size 0 character width and height are 15 dots and 16 dots respectively, and the default window width and height are 801 dots and 369 dots respectively, and

$15/801 = 1.87\%$ and $16/369 = 4.34\%$.

DELETE ALL NON-REF/REF OBJECTS deletes all referenced items and groups or all non-referenced objects within a window.

HP-GL No
Link Required Graphics Link and window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	1=referenced, 0=non-referenced	0	0 or 1	8

Response: None

Example:

```
OUTPUT @Display;"DA0;"
```

This will delete all of the non-referenced objects within a window.

```
OUTPUT @Display;"DA1;"
```

This will delete all of the referenced objects within a window.

Comments:

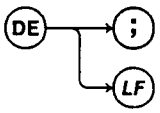
An explanation of items and groups and their referencing can be found in Chapter 4 and under the descriptions of the commands ***IT*** and ***GP***. A `DA0` followed by a `DA1` will delete all of the objects in a window, thus clearing the window.

DE

DELETE SCREEN deletes everything showing on the screen.

HP-GL	No
Link Required	Control Link

Syntax:



Response: None

Example:

```
OUTPUT @Display;"DE;"
```

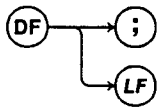
The screen (including softkey labels) will immediately become completely blank except for the annunciator block and softkey labels (see appendix B), and no instrument will be assigned to any part of the screen. All windows and their contents will be deleted. All links but the control link will be broken.

If desired, a *completely* blank screen can be achieved by also sending `DS0` and `ML0`. To clear a *window* see the ***IN***, ***PG***, or ***DA*** commands.

SET DEFAULT VALUES sets certain parameters to a predefined state.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response: None

Example:

```
OUTPUT @Display;"DF;"
```

Comments:

The ***DF*** command provides the means to set most parameters of a window to a default state. After a ***DF***, the state shown in Table 5.1 exists.

Firmware Revision Dependencies:

In ROM Version 5.0, ***DF*** clears the error bit in the status byte (see appendix C). In ROM Version 6.0, it does not.

75

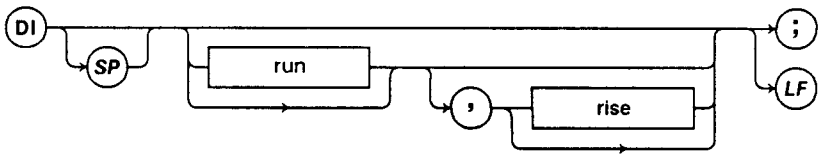
Graphics	non-referenced (***GP0***)
Line type (***LT***)	solid
Pen (***SP***)	1
Blinking (***BL***)	off
View (***VW***)	on
Marker char (***MA***)	#255, diamond
origin (***OR***)	0,0
Mapping (***MP***)	anisotropic
Masks	default (see ***IM***)
Label terminator (***DT***)	(ASCII 3)
Character sets (***CA*** and ***CS***)	Set 0
Selected character set	Standard (***SS***)
Pre-process mode (***PP***)	off
Pen status	up
Delta X (***DX***)	1
Character size (***SI, SR***)	0
Lettering direction (***DI, DR***)	0

Table 5.1: Default Parameters

SET CHARACTER DIRECTION ABSOLUTE specifies the direction in which characters are lettered.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
run	width of box defining direction	1	-	16
rise	height of box defining direction	0	-	16

Response:

None

Example:

```
OUTPUT @Display;"DI10,5;"
```

This ***DI*** command specifies that the direction of the characters in the current label will be in a line which is slanted with a vertical increase of five display units for every horizontal increase of ten display units (see the ***SC*** command description for an explanation of display units).

For such a label, the lettering direction will actually appear to be 45°, due to the compressed horizontal resolution of the display (see appendix B).

Comments:

Run and rise specify the direction according to the relationship rise/run where rise measures the number of units of vertical change and run measures the number of units of horizontal change. The character rotation is rounded to the nearest 90° since only four rotations are possible. A change in P1 or P2 (the vertices of the window) will not affect the direction of lettering. (See the example given for the ***DR*** command.) A ***DI***

command with a rise parameter of zero will produce horizontal labeling. A ***DI*** command with a run parameter of zero will produce vertical labeling with characters rotated 90°. Characters rotated in this fashion appear elongated horizontally, due to the compressed horizontal resolution (see appendix B).

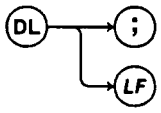
The ***DI*** command sets the lettering direction for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a label is drawn, its direction cannot be changed. The direction of a label will be that in effect at the time the label is built. For referenced graphics, changing the direction at any time will change the label. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;DI1,0;` will have no effect).

The default lettering direction used when an item is created is *horizontal.*

DELETE deletes the item or group which is currently selected.

HP-GL	No
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response: None

Example:

```
OUTPUT @Display;"GP1;IT2;DL;"
```

Group 1, Item 2 is deleted.

Comments:

Deletes the currently active group or item (see the ***IT*** and ***GP*** commands). The object(s) will disappear from the screen. If sent in non-referenced mode (`GP0`), nothing happens.

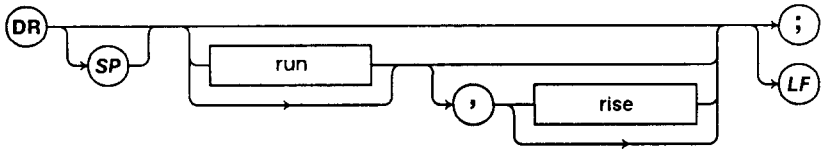
After a ***DL*** is sent, the display is in Group 0 (non-referenced graphics) until another ***GP*** or ***IT*** is sent.

DR

SET CHARACTER DIRECTION RELATIVE specifies the direction in which characters are lettered relative to the width and length of the window in which they appear.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
run	horizontal direction defined as a % of window width	100	(see text)	16
rise	vertical direction defined as a % of window height	0	(see text)	16

Response: None

Example:

```
1    ! Program "DR"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;BW1,200,100,400,300,0,,,1;"
30     OUTPUT @Display;"BW2,0,0,1023,383" ! So borders will show
40     OUTPUT @Display;"GP1;IT1;CL1,8;DT@;LBRELATIVE@"
50     OUTPUT @Display;"OR250,200;"
60     OUTPUT @Display;"DR20,10;"
70     OUTPUT @Display;"IT2;CL1,8;DT@;LBABSOLUTE@"
80     OUTPUT @Display;"OR250,125;"
90     OUTPUT @Display;"DI20,10"
100    END
```

This program will build two windows; window 1 is the window of interest, and will have a solid border because it is assigned to the keyboard (see ***BW***). (Window 2 is only built so that the border on window 1 will be visible). Two label items are created: item 1 ("RELATIVE") with lettering direction set by ***DR***, and item 2 ("ABSOLUTE") with lettering direction set by ***DI***. The ***DR*** command provides a vertical increase of 10% of the vertical length of the window for every horizontal increase of 20% of the horizontal width of the window. Thus, for P1 = (200,100) and P2 = (400,300), the label will have

a vertical change of 20 units for every 40 units of horizontal change. For such a label, the lettering direction will actually appear to be 45°, due to the compressed horizontal resolution of the display (see appendix B).

The direction would be different for different values of P1 and P2. For example, change the `400` in line 20 to `800` and re-run the program. Note that the *absolute* label does not change direction but the *relative* label does.

Comments:

Run and rise are integers and specify a percentage of the algebraic distance between P1 and P2. Run is the desired percentage of P2x - P1x and rise is the desired percentage of P2y - P1y (P1 and P2 are the lower left and upper right vertex points of the window). Thus (as shown in the example above), the window's P1 and P2 affect the direction of lettering when using ***DR*** (the ***DI*** command is used for selecting a lettering direction not affected by P1 and P2).

Range limitations dictate that $run \times window\ \ width < 32768$ and $rise \times window\ \ height < 32768$. In other words, if the run parameter is 50 (meaning 50%) and the window width is 800, then $50 \times 800 = 40,000$, so the range will be exceeded. No error will be generated, but the results will be unpredictable.

A ***DR*** command with a rise parameter of zero will produce horizontal labeling. A ***DR*** command with a run parameter of zero will produce vertical labeling with characters rotated 90°. Characters rotated in this fashion appear elongated horizontally, due to the compressed horizontal resolution (see appendix B). Note that the character rotation is rounded to the nearest 90° since only four rotations are possible.

The ***DR*** command sets the lettering direction for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a label is drawn, its direction cannot be changed. The direction of a label will be that in effect at the time the label is built. For referenced graphics, changing the direction at any time will change the label. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;DR50,30;` will have no effect).

The default lettering direction used when an item is created is *horizontal.*

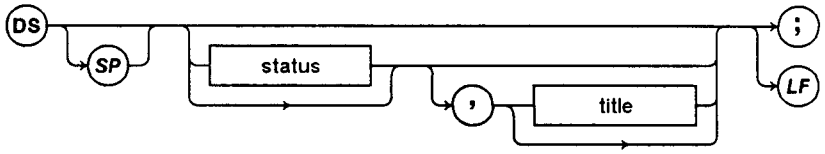
Firmware Revision Dependencies: In ROM Version 5.0, ***DR*** sent with no parameters generated an error.

DS

DISPLAY STATUS ON/OFF selects whether or not the block with the RLTSEA status annunciators, and/or the character line, appear.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
status	0=annunciator block off	1	-	8
title	0=character line box off	0	-	8

Response: None

Example:

```
OUTPUT @Display;"DS0,1;"
```

The annunciator block (box with the RLTSEA annunciators and the "USER/MENU/DISP" label) is turned off, and the box around the character line (see the ***FC*** command and appendix B) is turned on.

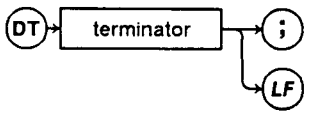
Firmware Revision Dependencies:

In ROM Version 5.0, the default for the title parameter is random.

DEFINE TERMINATOR defines the character to be used as the terminator of strings. Parameters of the commands ***FC***, ***LB***, and ***ML*** are affected.

HP-GL Yes
Link Required Any Link

Syntax:



Item	Description	Default	Range	Bits
terminator	character to be used to terminate the strings in ***LB***, ***ML***, and ***FC*** commands	none	any ASCII character	-

Response: None

Example:

```
OUTPUT @Display;"DT@;"
```

From this point on, strings are terminated by the character @.

Comments:

This defines the character which is to be used as the terminator for labels for the window. Each I/O channel (for example, HP-IB and up to four HP-MSIB addresses) has its own terminator. Whatever byte immediately follows the two letters DT will be used as the new terminator for that window (hence, no space should separate the DT from the parameter). The terminator at window initialization is ASCII 3, ***ETX*** (end of text), which can be typed in as CTRL-C.

Firmware Revision dependencies:

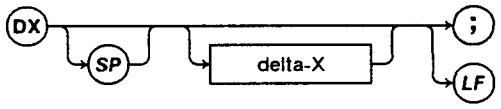
In 5.0, the power-up terminator for an I/O channel is reset at power-up, on an HP-MSIB reset, on a ***DF***, or whenever that channel's window is initialized. In 6.0, the HP-IB channel's terminator is reset as well on a DEVICE CLEAR or when the DISPLAY PRESET softkey is pressed.

DX

SET DELTA-X sets the X increment to be used in the graph instructions ***GA*** and ***GR***.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
delta-X	delta-X in user units for graphs	1	-	8

Response: None

Example:

```
OUTPUT @Display;"GP1;IT3;DX30;"
```

The X increment to be used for group 1, item 3 in any subsequent graph command (***GR*** or ***GA***) will be 30 user units.

Comments:

This command sets the X increment to be used in the graph commands ***GA*** and ***GR***. The X-coordinate of the current pen position will be added to the specified delta-X to determine the new X-coordinate for every point.

The delta-X is characteristic of an entire item. Hence, changing it while drawing a referenced graphics item changes the spacing between every point in the item. In non-referenced graphics (`GP0`), the delta-X can be changed as a trace is drawn and it will change only for succeeding points.

The ***DX*** command sets the delta-x for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a trace is drawn, it cannot be changed. For referenced graphics, changing the delta-x at any time will change the trace. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;DX1;` will have no effect) or if the active item is not a graph.

Since the delta-X is specified in user units, it is dependent on the current scale factor (**SC**). Changing the scale factor and then sending another **DX** will cause the point spacing of an already drawn (referenced) trace to change to reflect the new scale factor and the new delta-X. Thus, although the **DX** parameter must be an *integer*, it is nonetheless possible to specify a delta-x which is a non-integral number of *display units*, since the delta-x is a function of both the **DX** parameter and the scale factor.

The default delta-x used when an item is created is 1.

See also the **GA**, **GR**, **PA**, and **PR** command descriptions.

Examples:

```
1    ! Program "DX"
10     ASSIGN @Display to 704
20     OUTPUT @Display;"DE;SC 0,100,0,100"
30     OUTPUT @Display;"IT1;DX 20;GA 20,80,30,10;"
40     OUTPUT @Display;"SC 0,200,0,100"
50     OUTPUT @Display;"IT2;OR 0,100;DX 10;GA 20,80,30,10;"
60     END
```

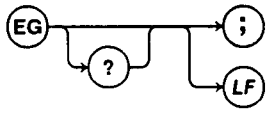
Lines 30 and 50 will cause identical traces to be drawn, with the second trace 100 user units above the first.

EG

OUTPUT ERROR MESSAGE enables the user to get a text description of an error.

HP-GL	No
Link Required	Any Link

Syntax:



Response:

ROM Version 5.0: a string containing a single error message terminated by ***CR LF***. The message is that of the most recently occurring usage error. The message is identical to that appearing when the REPORT ERRORS softkey is pressed. ***EG*** does not clear the error. If there is no error, the string "2000,No errors" is sent.

ROM Version 6.0 and later: Error message list, with numbers. The numbers are separated from the message by commas (for example, "2001,Illegal command") and each message ends with a ***CR LF***. The entire message is terminated by ***END*** (see page 28). The list contains all usage errors which have occurred since the last time ***EG*** or ***OE*** was received or REPORT ERRORS was pressed, in no particular order. "0,No errors" is sent if there are no errors. ***EG*** clears the errors.

A list of errors can be found in appendix C.

Example:

This asks for a string of error descriptions:

```
1    ! Program "EG1"
10     ASSIGN @Display to 704
20     DIM Error_string$[25]
30     OUTPUT @Display;"EG;"
40     ENTER @Display; Error_string$
50     DISP Error_string$
60     END
```

This prints all the current error messages:

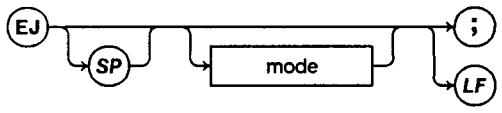
```
1    ! Program "EG2"
10     ASSIGN @Display to 704
30     DIM Errmsg$(7)[32]
40     OUTPUT @Display;"EG;"                        ! Ask for errors
50     ENTER @Display USING "%,K";Errmsg$(*)  ! Read w/end on EOI
60     FOR I=0 TO 7
70       IF Errmsg$(I)<>"" THEN PRINT Errmsg$(I)
80     NEXT I
90     END
```

EJ

EJECT ON/OFF enables/disables the eject after a plotter or printer dump.

HP-GL	No
Link Required	Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=off, 1=on	1	0 or 1	8

Response: None

Example:

```
OUTPUT @Display;"EJ0;"
```

The display will now not send form feeds after printer dumps or ***PG*** commands after plotter dumps.

Comments:

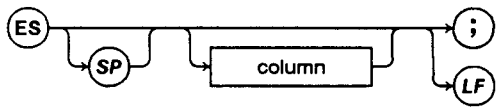
The ***PG*** command causes a page eject on some plotters but can cause an error on older plotters, so it is sometimes worthwhile to defeat it.

Firmware Revision dependencies: Revision 6.0 and later only

ERROR SCREEN is equivalent to selecting the REPORT ERRORS utility on a display.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
column	column to report first	-1	-1 through 31	8

Response: None

Example:

```
OUTPUT @Display;"ES 18;"
```

The REPORT ERRORS utility will be selected and the scan for modules with errors will start at HP-MSIB column 18. (Only reports from modules with HP-MSIB row address 0, column address greater than 18, with errors, will appear).

Comments:

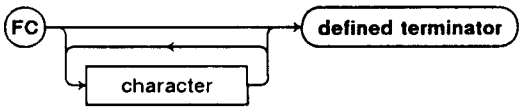
The search for errors starts at the column given as the parameter and ends with column 31; the search does *not* wrap around to 0 and search the columns whose value is less than that of the parameter. If the parameter is -1, the REPORT ERRORS utility will be aborted. The ***ES*** command clears non-hardware related errors after they have been reported.

FC

FILL CHARACTER LINE fills the character line at the bottom of the display with text.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range
character	character to be added to character line		any ASCII character
defined terminator	defined by ***DT***		

Response: None

Example:

```
OUTPUT @Display;"DT@;FCCenter Frequency 123 MHz@;"
```

The character string Center Frequency 123 MHz will appear in the character line of the display. The terminator @ is defined by the ***DT*** command.

Comments:

The display maintains a line at the bottom of the display called the *character line* for use by the keyboard owner to display character data (see appendix B). The ***FC*** command is used by an element with access to the keyboard to fill the character line with text. If more than 40 characters are sent, the text will scroll left so that the last character sent is always on the far right. For this purpose the character line is identical to a single-line label. Note that the "text" message sent must end with the previously defined label terminator (see the ***DT*** command.)

Certain control characters are useful while using ***FC***. ***LF*** (ASCII 10) should not be sent in ***FC*** text. The ***FF*** (ASCII 12) character blanks the character line. Also, some control characters and sequences cause special actions. These are listed in appendix F. These

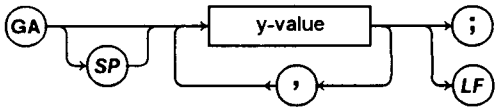
actions include display enhancements and special cursor motions among other things, as well as functions like display functions on and off. Also, the ***LB*** command description provides more information on using special label features.

GA

GRAPH ABSOLUTE moves the pen to the location specified by the y-values.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
y-value	y-coordinate of point to which the pen will be moved	none		16

Response: None

Example:

```
1    ! Program "GA"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;PD;"
30     OUTPUT @Display;"DX100;"
40     OUTPUT @Display;"GA40,50;"
50     OUTPUT @Display;"GP1;IT1;DX450;GA50,225;"
60     OUTPUT @Display;"IT2;DX50;GA 100,125;PU;GA150;PD;GA175;"
70     END
```

With delta-X set at 100, the ***GA*** command on line 40 will move the pen from its initial location (0,0) to the point (100,40) and then to the point (200,50), resulting in *two* straight lines being drawn. The ***GA*** command on line 50 is in referenced graphics mode, with delta-X set to 450, and thus the pen will be put down at the point (0,50) and moved to the point (450,225) resulting in *one* straight line being drawn. The ***GA*** command on line 60 sets delta-X to 50 and so will put the pen down at the point (0,100) and move it to the point (50,125), then lift it and move it to the point (100,150), then put it down again and move it to the point (150,175), resulting in *two* straight lines being drawn with a gap between them.

This example generates the identical screen as the example under the ***PA*** command.

Binary mode example (see also page 26):

```
1    ! Program "GAB"
10     ASSIGN @Display TO 704
20   ! This is equivalent to "GA 0,300":
30     Y1$=CHR$(0)&CHR$(0)    ! (256*0)+0=0
40     Y2$=CHR$(1)&CHR$(44)   ! (256*1)+44=300
50     OUTPUT @Display;"DE;IN;PD;IT1;DX600;"
60     OUTPUT @Display;"GA #I"&Y1$&Y2$;
70     SEND 7;DATA 0 END      ! Send end AFTER last byte
80     END
```

This example generates the identical screen (one straight line) as the binary example under the ***PA*** command.

Comments:

This instruction causes the pen to move to the Y-location specified by **y-value** in *user units*. The X value of the location to be moved to will be the current X-coordinate plus delta-X. If the pen is down a straight line will be drawn between the current location and the new location. (See the ***DX*** and the ***PD*** commands.) As with all commands, the data may be sent to the display in binary (#I) format. (See page 26.)

In referenced graphics, there is no starting pen position for lines. The first ***GA*** command sets the pen y-axis start position (the x-axis start position is 0) and lowers the pen. Thus it requires at least two such commands to draw one line.

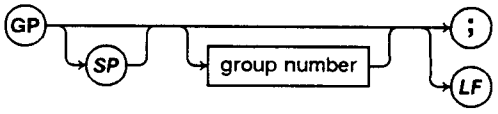
See the discussion of plotting under the ***PA*** command to understand the differences between graphing in referenced and non-referenced graphics. See Chapter 4 and the ***GP*** and ***IT*** command descriptions for more on referenced graphics.

GP

IDENTIFY GROUP references a group of objects on the screen. (See chapter 4).

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
group number	group to make active	0	0, and 1 through 16	8

Response: None

Example:

```
OUTPUT @Display;"GP2;OR 100,100;"
```

The origin of group 2 is set to (100,100). References to subsequent items will apply to group 2 objects until a new group is selected.

```
OUTPUT @Display;"GP1;VW1;BL1;GP2;VW0;"
```

Group 1 is set to be viewable. All objects in group 1 will blink. Group 2 is blanked.

```
OUTPUT @Display;"GP0;"
```

All subsequent graphics will be non-referenced until the next ***GP*** or ***IT*** command (an ***IT*** command during non-referenced graphics will automatically select Group 1).

Comments:

The display supports the grouping of graphics objects (called "items") into numbered "groups" (a full discussion of groups and items can be found in Chapter 4). Once a group (other than 0) has been selected, that group is said to be "active" and the display is in "referenced graphics mode". Grouping objects allows them to be collectively moved (by setting the group origin), blanked, or set to blink together.

A group number of 0 invokes "non-referenced graphics mode", which is designed to operate like a pen plotter (for HP-GL compatibility). In this mode, objects are drawn

and essentially forgotten; they cannot then be moved, or set to blink, or blanked; it is as though they had been drawn with a pen on a piece of plotter paper. Once drawn they will not change until the window is re-initialized or deleted or a ***DA*** command is sent. When group 0 is being used, we say that there is no active group.

The ***GP*** instruction is used to select the active group. If no previous reference has been made to that group, the group will be created. When a group is created, it will assume the default values for each attribute. These are:

attribute	state
blinking	off
blanked	off (viewable)
origin	0,0 (user units)

After a group is referenced and thus selected (or created), and before any items are referenced (see ***IT***), any blinking, origin, or blanking instructions *will* apply to all objects within the group, but *the display will be set for non-referenced graphics.* This means that graphics objects with no item reference will be placed in Group 0, as the display does not yet know what item number to use (see example under ***IT***). After the first item is referenced, the display will do referenced graphics using that item, and any blinking, origin, or blanking instructions apply to that item until another item is referenced. All items referenced after a group instruction and before the next are considered part of that group.

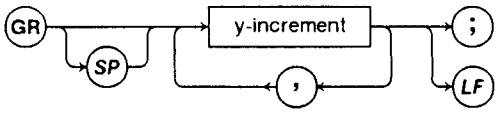
The blinking and blanking attributes of groups or item "OR" together, that is, if either an item or the group it is in are set to blink, it will blink, and if either an item or the group it is in are blanked, it will be blanked. The origins add, and add to the window origin (see ***OR*** and appendix E).

GR

GRAPH RELATIVE moves the pen to the location specified by the Y-increments.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
y-increment	value added to current y-coordinate to derive the y-coordinate of the point to which the pen will be moved	none		16

Response: None

Example:

```
1    ! Program "GR"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;PD;"
30     OUTPUT @Display;"DX40;"
40     OUTPUT @Display;"GR20,80,30,10;"
50     END
```

With delta-X set at 40, this program will move the pen from its starting point (0,0) to the point (0+40,0+20)=(40,20) to the point (40+40,20+80)=(80,100) to the point (80+40,100+30)=(120,130) to the point (120+40,130+10)=(160,140). Since the pen is down ("PD;") four straight lines will be drawn.

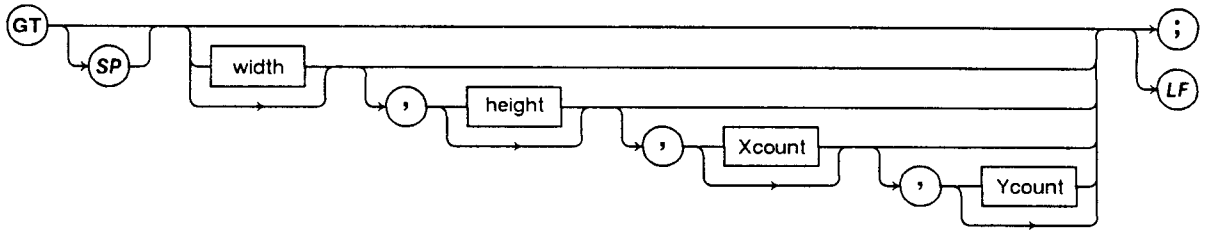
Comments:

This instruction causes the pen to move to the Y-location specified by **y-increment** in *user units*. The X value of the location to be moved to will be the current X-coordinate plus delta-X. The Y value of the new point will be the current Y plus **y-increment**. If the pen is down a straight line will be drawn between the current location and the new location. (See the ***DX*** and the ***PD*** commands.) As with all commands, the data may be sent to the display in binary (#I) format. See also the discussion of plotting under the ***PA*** and ***GA*** commands.

GRATICULE draws a grid.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
width	width of a box	80	-	16
height	height of a box	40	-	16
Xcount	number of boxes across	10	-	16
Ycount	number of boxes vertically	8	-	16

Response: None

Example:

```
OUTPUT @Display;"DE;GP1;IT1;GT 100,50,10,5;"
```

A grid will be drawn, using the current line type, which will consist of 50 rectangles whose dimensions are 100 horizontal units by 50 vertical units (all units are user units). They will be arranged in 10 columns and 5 rows resulting in a large rectangle whose dimensions are 1000 horizontal units by 250 vertical units.

Comments:

This instruction draws a grid which consists of Xcount boxes horizontally and Ycount boxes vertically. Each box's dimensions are width by height. The lower left corner of the graticule is drawn at the current pen position if non-referenced, at the item origin otherwise. All parameters are in *user units*.

The line type is adjusted for the compressed horizontal resolution of the display (see appendix B) by stretching it on horizontal strokes, so that it appears uniform in both vertical and horizontal directions.

GT

This command can also be used to make bar charts or to frame a window, as shown below:

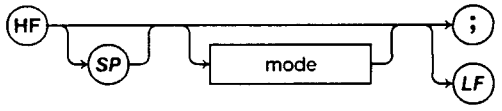
```
1    ! Program "GT"
10     ASSIGN @Display TO 704
15     OUTPUT @Display;"DE;IN;SC 0,800,0,368;"
20     OUTPUT @Display;"IT1;GT 800,368,1,1"
30     OUTPUT @Display;"IT2;OR 100,0;GT 1,200,100,1"
40     END
```

Line 10 of the above program initializes the default controller window (window 5), which extends from P1=(112,16) to P2=(911,383). It sets the scaling so that each *user unit* equals one *display unit*, and so the origin of the window is (0,0). Line 20 frames the window by drawing one 800 x 368 box around it. Line 30 draws a vertical bar at (100,0) in *user units*, by drawing a graticule with 100 tall (200 dots high), thin (one dot wide) boxes stacked next to each other.

HOLDOFF ON/OFF stops the display from updating the screen.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	1=held off, else normal	0	-	8

Response:

None

Example:

```
OUTPUT @Display;"HF1;"
```

The screen will freeze completely.

Comments:

While holdoff is enabled, the controller may manipulate traces, position, data, and so on, but no changes will show on the screen. If an `HF0` is then performed, holdoff mode is off and the screen will show the result of the manipulations which have occurred while the screen was frozen.

HF

When the holdoff is enabled (**mode** = 1) the display will finish reading the current vector list data to the screen and then stop drawing. The display will not accept any more data until this is complete.

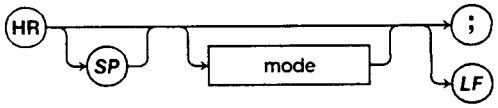
WARNING

This command will affect response time and should only be used when absolutely necessary. This command will stop drawing for *all* windows in a multi-window configuration.

HI RES ON/OFF enables/disables full resolution (1024 wide) printer dumps.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=off, 1=on	0	0 or 1	8

Response:

None

Example:

```
OUTPUT @Display;"HR1;"
```

High resolution dumps are enabled.

Comments:

In high resolution mode, the display dumps 1024 dots per screen line to the printer. It sets the printer to high density mode to do this. Otherwise, the data is compressed to 512 dots per line and the printer set to low density mode. The printer resolution is not set back to its old value after a screen dump because there is generally no way to ask the printer what it should be set back to.

The format used for setting the resolution is that of the HP Thinkjet printer. This format is not compatible with certain other HP printers (such as the HP Paintjet).

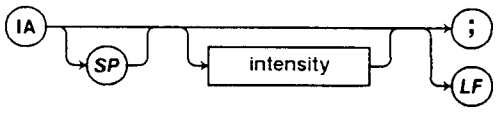
Firmware Revision dependencies: Version 6.0 and later only.

IA

INTENSITY ADJUST allows the display brightness to be set remotely.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
intensity	display brightness	15	0 to 19	

Response:

None

Example:

```
OUTPUT @Display;"IA9;"
```

The screen intensity is set to 9.

Comments:

Intensity 0 is not necessarily all the way off. To erase the screen, use ***DS*** and ***DE*** (from a controller), or see the ***DA***, ***PG***, or ***IN*** commands to erase a window.

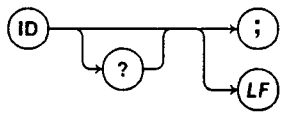
The intensity setting is preserved through a power cycle except that if it is set to less than 9 on power-up, the display sets it to 9 on power-up in order to guarantee that screen data will be visible.

Firmware Revision dependencies: Version 6.0 and later only.

OUTPUT IDENTIFICATION makes the display identifier available for output to the computer.

HP-GL	No
Link Required	Any Link

Syntax:



Response:

70205A ***CR LF*** or 70206A ***CR LF***, terminated by ***END*** (see page 28).

Example:

```
1    ! Program "ID"
10     ASSIGN @Display to 704
20     OUTPUT @Display;"ID?;"
30     ENTER @Display; M$
40     DISP M$
50     END
```

Comments:

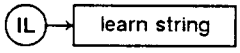
This command is used whenever the user wishes to know the display model number. Identical to the ***OI*** command.

IL

INPUT LEARN STRING prefixes a binary learn string being sent to the display (See ***OL***).

HP-GL	No
Link Required	Control Link

Syntax:



Response:

None

Example:

```
OUTPUT @Display;"OL;"
ENTER @Display USING "%,K"; Learnstring$
```

... A period of time later ...

```
OUTPUT @Display USING "#,K";Learnstring$
```

The display will return to the state it was in when the ***OL*** command was sent.

Comments:

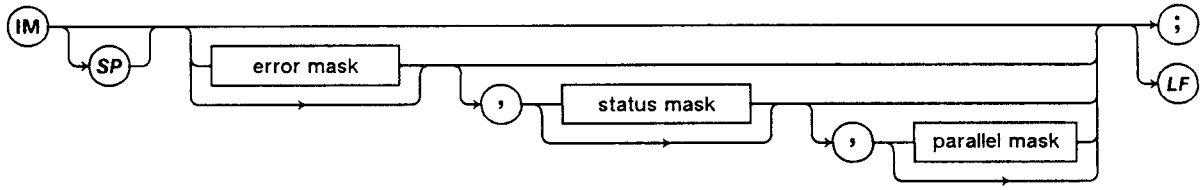
Upon receipt of the ***OL*** command, the display sends back a learn string in binary (#I) form. The first two characters of this string are IL. It thus satisfies the syntax of the ***IL*** command, so it can be sent back to the display at any time in order to return the display to the state it was in when the ***OL*** command was sent. Note that the user does *not* also send an IL since it is already included in the learn string. See the ***OL*** command for further discussion.

END must be sent with the last byte unless the learn string sent is the exact length required. For HP-IB, this means ***EOI*** asserted with the last byte. For HP-MSIB, this means the HP-MSIB ***END*** command is sent after the last byte (see the *Communication Protocol Design Guide* for details).

INPUT MASK specifies the conditions under which errors will be reported and ***STATUS*** messages sent.

HP-GL Yes
Link Required Any Link

Syntax:



Item	Description	Default	Range	Bits
error mask	(see text)	255	-	8
status mask	(see text)	32	-	8
parallel mask	(see text)	0	-	8

Response:

None

Example:

```
OUTPUT @Display;"IM 9,128,0;"
```

This specifies that a 2011, Memory overflow or a 2001, Illegal command (see appendix C) will set the error bit and that a ***STATUS*** (SRQ on HP-IB) message will be sent whenever a key press is available.

Comments:

The ***IM*** instruction specifies the conditions under which an error message, a ***STATUS*** message, and a positive parallel poll response will occur for the I/O channel which sent the message. The parallel poll response is not implemented in the display but is still a legal parameter for this command. The masks and error codes are defined in appendix C.

IM

The ***STATUS*** message for HP-IB consists of asserting SRQ and sending the status byte on a serial poll. For HP-MSIB, it is the ***STATUS*** HP-MSIB command being sent. In either case, if a condition set to generate a ***STATUS*** message occurs, the message will be sent.

Firmware Revision dependencies:

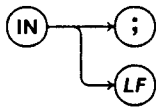
In ROM version 5.0:

- The status bit setting and clearing works differently.
- The error mask function does not work properly.
- The bit in the status byte (the byte that goes out as part of the status message) is only set if the corresponding bit in the mask is set (in 6.0, it is set regardless and the mask determines whether to pull SRQ or send the ***STATUS*** HP-MSIB command).

INITIALIZE returns the window to initial power on state.

HP-GL Yes
Link Required Any Link

Syntax:



Response:

None

Example:

```
OUTPUT @Display;"IN;"
```

Comments:

The ***IN*** command returns the window to the initial setup state. All errors are cleared and the WINDOW INITIALIZED bit of the status byte (see appendix C) is set (all other bits are cleared). SRQ is cleared. The default window values as described in the ***DF*** command are set. The window is cleared and all data is lost. The scale factor **SC** is reset so that user units are equal to display units.

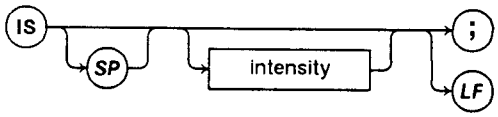
The window configuration (location and size) and links are unchanged. If there is a keyboard link to the window, the menu keys are erased.

IS

INTENSITY SELECT sets the intensity level of certain items on the window.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
intensity	Intensity of an item. For markers, 0-127=normal, 128-255=high. For all other items, ignored.	0	-	8

Response: None

Example:

```
1    ! Program "IS"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"IN;"
30     OUTPUT @Display;"GP1;IT2;MK200,100;IS -1;" ! High intensity
40     OUTPUT @Display;"GP2;IT1;MK300,100;IS 0;"  ! Normal intensity
50     END
```

This program places a marker at the point (200,100) and a marker at the point (300,100) and then sets the intensity level at *high* for the first marker. Note that a parameter of -1 is equivalent to 255, because the parameter size is 8 bits. Hence, `IS -1` intensifies a marker.

Comments:

This instruction selects the intensity level of items or groups on the screen of the display. In the 70205A and 70206A, it is *only implemented for markers* and is ignored for all other types of items. These displays have two available intensities, normal and high, but for two markers only. Thus, only two markers at a time can be intensified.

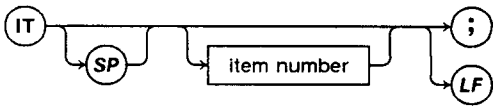
The ***IS*** command sets the intensity for the active item. Once drawn, changing the intensity at any time will change the marker. This command has no effect on non-referenced markers or if there is no active item in a selected group (that is, sending the sequence `GP1;IS0;` will have no effect).

IT

IDENTIFY ITEM references a specific object on the screen. (See chapter 4).

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
item number	item to make active	0	0, and 1 through 50	8

Response: None

Example:

```
OUTPUT @Display;"GP2;IT37;"
```

All subsequent graphics instructions will apply to the graphics item numbered 37 in group 2. If such an item does not exist then the next graphics instruction will create it and set its type.

Comments:

The ***IT*** command references objects within the display. Once an item is referenced, and until another group or item is referenced, that item is said to be the "active" item. A full discussion of items (and groups) can be found in Chapter 4 and under the ***GP*** command. The item referenced will be in the most recently referenced group, or in group 1 if no group has been referenced. If the item identified does not exist the item is created, and subsequent instructions define the item type and attributes. If the item does exist then the subsequent instructions will modify the item in the appropriate manner. If the item already exists and the subsequent instructions are not applicable to that item type then an **Illegal parameter** error is declared (see below). For example, if ***GA*** is sent after an item has already been sent an ***LB*** command, an error is declared.

The item types and applicable instructions are:

item type	applicable instructions
Text	***LB,CL,SI,SR,DI,DR***
Absolute graph	***GA,DX,BA,TP,PN***
Relative graph	***GR,DX,BA,TP,PN***
Absolute plot	***PA,BA,TP***
Relative plot	***PR,BA,TP***
Graticule	***GT***
Axis	***AX***
Marker	***MK,MA***
User defined text	***UC***

The following instructions apply to *all* item types, in that no error will be declared if they are sent:

BL, VW, IS, SP, PD, PU, OR, SS, SA, CS, CA, LT.

When an item is created, it assumes the default values for each attribute. If no attributes are then specifically set for the item, it retains these defaults. Some attributes relate to both groups and items, some to items only.

Group/Item attributes

attribute	initial state	associated command
blinking	off	***BL***
blanking	off	***VW***
origin	0,0	***OR***

The attributes in the above group pertain to groups *and* items - for example, an item may have its own origin, which is added to the group's origin when the item is plotted. If a ***GP*** command is sent, and before any ***IT*** command is sent, one of the above commands is sent, the attribute modified will be that of the *group*.

Item attributes

attribute	state	associated command
pen	1	***SP***
line type	solid	***LT***
blank ahead	0	***BA***
trace pointer	0	***TP***
delta-x	1	***DX***
intensity	low	***IS***
marker character	diamond	***MA***
character size	smallest (0)	***SI,SR***
character rotation	horizontal	***DI,DR***

The attributes in the above group pertain to *items only* - that is, an item may have a pen number, but there is no pen associated with, say, group 1. If a ***GP*** command is sent, and before any ***IT*** command is sent, one of the above commands is sent, the command will have no effect. *After a* ***GP*** *and before* ***IT****, the display is still in non-referenced graphics mode, and all graphics data goes to group 0.*

Just as for the ***GP*** command, a 0 parameter (for example `IT0`) allows information to be placed on the screen, but that information cannot be referenced again. In other words "group 0" and "item 0" mean the same thing: non-referenced graphics. Since group 0 has no items, it alone of all the groups keeps a record of the "current" value of each of the *item* attributes above.

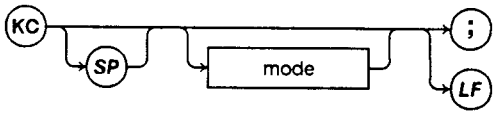
Caution should be exercised when creating items. Each item uses at least one block of memory from the vector list, which is finite in size. The ***RM*** command can be used to determine how much memory remains in the display at any time. An excessive number of items can quickly run the display out of memory (see Chapter 4).

See also the ***SC*** and ***OR*** command descriptions and appendix E for more on how items are placed and moved around in a window.

KEY COPY ON/OFF enables/disables the ability of a display to copy the softkey labels.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=off, 1=on	0	0 or 1	8

Response:

None

Example:

```
OUTPUT @Display;"KC1;"
```

The display is now able to copy the softkey labels.

Comments:

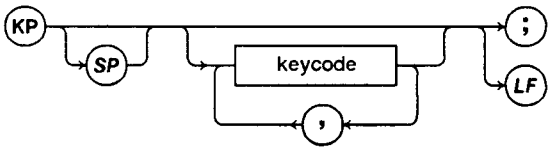
This command enables and disables the ability of a display to copy the softkey labels, status block and character line in plotter or printer dumps (see appendix B).

KP

SIMULATE KEY PRESSED accomplishes a phantom key press from the controller.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
key code	key to simulate (see list of key codes on page 99)	64	-	16

Response: None

Example:

```
OUTPUT @Display;"KP33;"
```

The minus sign key is pressed.

Comments:

Using this command, the controller may remotely press any key on the display. If the display menus are active and this command is sent over HP-IB, HP-IB must be in LOCAL (LCL) for the display to obey the key. If the display menus are *not* active, the "key press" is simply passed on to the owner of the keyboard just as though a key had actually been pressed on the keyboard, on an external keyboard, or by using ***AE***. The keyboard owner may then use the ***KY*** command to receive the key code. This command along with the ***RG*** and ***PP*** commands provide the means for semi-automatic operation.

The key codes can be found in Table 5.2. See Figure 5.1 for a key to the softkey numbers (the softkeys are labeled using the ***ML*** command).

Further key codes are 256 + n where n represents the codes of the USASCII character set. For example, the character "A" is represented by the code 256+65 (since 65 is the ASCII code for A). These kinds of key codes are useful for simulating what the user could send using the ***AE*** command or an external ASCII keyboard.

Table 5.2: Key Codes

Key pressed	Key code	Key pressed	Key code
Numeric 0	0	Softkey 10	26
Numeric 1	1	Softkey 11	27
Numeric 2	2	Softkey 12	28
Numeric 3	3	Softkey 13	29
Numeric 4	4	Softkey 14 (lower left)	30
Numeric 5	5	Decimal Point	32
Numeric 6	6	Minus Sign	33
Numeric 7	7	Left Arrow	34
Numeric 8	8	Down Key	35
Numeric 9	9	Up Key	36
Softkey 1 (upper right)	17	Menu (MNU)	37
Softkey 2	18	User (USR)	39
Softkey 3	19	Local	40
Softkey 4	20	Hold	41
Softkey 5	21	Print	42
Softkey 6	22	Display (DSP)	43
Softkey 7 (lower right)	23	Instrument Preset (I/P)	46
Softkey 8 (upper left)	24	Plot	47
Softkey 9	25	Null Key (no key pressed)	64

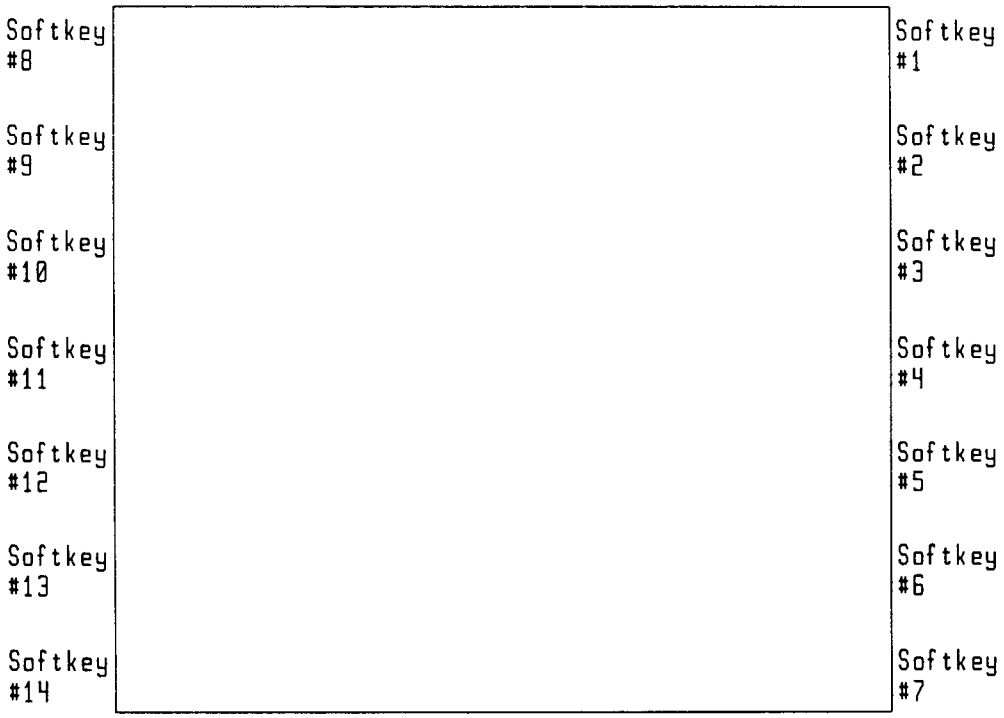


Figure 5.1: Display Softkey Numbers

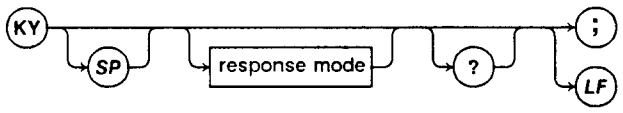
Firmware Revision Dependencies: In ROM version 5.0, ***KP*** commands to press keys in the display menus are not obeyed in LOCAL.

KY

SEND KEYBOARD DATA is sent by an instrument in order to find out what key was pressed.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
response mode	1=binary, else ASCII	0	-	8

Response:

The key code of the key which was pressed, as shown in the table on page 99, terminated by ***CR LF*** and ***END*** (see page 28).

Example:

```
OUTPUT @Display;"KY;"
ENTER @Display; Key
```

The key code of the key most recently pressed will be sent out.

The program below shows how to implement an interrupt-driven key reading program on HP-IB:

```
10   ! Program "KY"
20     REMOTE 7
30     ASSIGN @Display TO 704
40   ! Establish keyboard link with display so key presses will
50   ! be sent to the computer:
60     OUTPUT @Display;"DE;BW1,,,,,0,,,1;"
70     OUTPUT @Display;"IM 255,160;" ! Pull SRQ on key or error
80     ON INTR 7 GOSUB Key_read      ! Set up interrupt vector
90 Idle:ENABLE INTR 7;2                  ! Interrupt on SRQ
100    GOTO Idle                         ! Loop
110    !
```

(continued on next page)

```
120 Key_read:    !
130   IF BIT(SPOLL(@Display),7) THEN       ! Key pressed?
140     OUTPUT @Display;"KY"               ! Get key number
150     ENTER @Display;Key_number
160     PRINT "Key ";Key_number;" pressed."
170     BEEP
180   END IF
190   RETURN
200   !
210   END
```

Comments:

This command is sent by the module with the keyboard link or by the controller whenever it wants to find out which key was pressed (see the ***KP*** command). The response may be either in ASCII or binary (#I) format depending on the parameter. Key codes are in the table on page 99.

KY clears the stored key code, so pressing a key and then sending ***KY*** twice will cause the key press to be lost. In other words, the key code sent in response to a ***KY*** is the code of the key most recently pressed since the last ***KY*** was sent.

Note that pressing the DISPLAY (DSP) key, LOCAL (LCL) key, or PRINT or PLOT hardkeys does not generate a key code; these keys are intercepted by the display system and not passed on.

The discussion on page 11 shows the normal sequence of events surrounding the use of ***KY*** by modules. ***KY*** can be sent at any time, whether or not a ***STATUS*** message has been received; if no key has been pressed, the display will simply return key code 64. It is, in fact, recommended that ***KY*** be sent after a keyboard link has been established to make sure that any pending key presses are serviced.

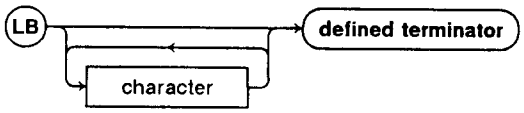
Sending ***KY*** clears the Key Pressed bit of the status byte (see appendix C).

LB

LABEL puts text on the display.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
character	character to be added to label	-	ASCII character	
defined terminator	terminator defined by DT	-	-	

Response: None

Example:

```
OUTPUT @Display;"DT@;GP1;IT1;LBNAME@"
```

The label NAME is placed on the screen of the display. Note that the symbol @ is being used as the label terminator as defined in the ***DT*** command.

```
OUTPUT @Display;"GP1;IT1;CL2,8;LBSPECTRUMANALYZER@;OR100,200;"
```

By using the command ***CL*** in conjunction with the ***LB*** command, the two-line eight-character label placed on the screen of the display is:

```
SPECTRUM
ANALYZER
```

The lower left corner of the cell containing the S will be at point 100,200 (see appendix D).

Note that if the command GP1;IT1;CL2,8;LBSPECTRUM ANALYZER@ had been used, three eight-character lines would be needed due to the extra space. Due to the scrolling procedure described below, this would have resulted in the following label.

```
ANALYZE
R
```

Comments:

This command puts text in the window. The dimensions (number of characters by number of lines) of the label can be set with the ***CL*** command. The size of the text can be set with the ***SI*** or ***SR*** commands. For multi-line labels the next line is written to by sending ***CR LF*** between lines.

With a multiple line label, if the user sends more characters in a line than are configured, the display will write subsequent characters on the next line. The user can also move to the next line by sending carriage return and/or line feed characters. When the last line of the label is written, further text will be written onto the last line with all the lines scrolling up one line. The top line scrolls off the top of the label[4]. Thus, in situations where you might expect extra text to be truncated, the extra text is printed and the earlier text is lost. The easiest way to think of a label is as a small computer terminal screen, as wide and high as you have configured the label.

Some control characters and sequences cause special actions. These are listed in appendix F. These actions include display enhancements and special cursor motions among other things, as well as functions like DISPLAY FUNCTIONS on and off.

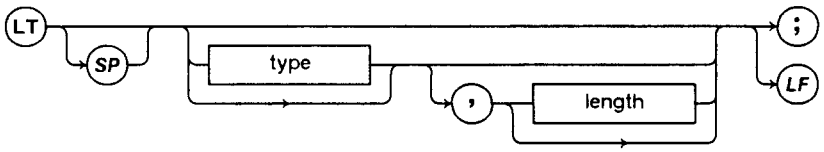
[4]With a single line label, if more characters are written than assigned, the single characters will scroll off to the left.

LT

SET LINE TYPE sets the line style to be used by the instructions ***GA,GR,PR,PA,GT***.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
type	(see table below)	Solid	-	16
length	ignored			16

Response: None

Example:

```
OUTPUT @Display;"GP1;IT3;LT;"
```

Sets default line type (solid) for group 1, item 3.

Comments:

This command sets the line style to be used by line drawing instructions (***GA, PA, GR, PR, GT, AX***). The pattern is set as follows:

Type	Line	Equivalent parameter
no parm	solid	
0	end points only	
1	. . (one dot on in twelve)	2048
2	.... (four dots on in twelve)	15
3	.... (six dots on in twelve)	63
4	. ...	575
5	.. ..	831
6	.. .. (two dots on in four)	819
7	. . . . . . (one dot on in four)	2184
8	. . . . . (every other dot on)	2730

If the parameter is outside the range 0 - 8, then the parameter (MOD 32768 if it is too big for 16-bits) is treated as a binary pattern which becomes the line type. Only the bottom 12 bits are used, and they repeat on the line. For example, if the parameter is 13653, we take the bottom 12 bits which are 010101010101 and that becomes the line pattern (every other dot on).

The line type is not sent out on plotter dumps. All line types are the same on plotter dumps.

The ***LT*** command sets the line type for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a trace is drawn, it cannot be changed. For referenced graphics, changing the line type at any time will change the trace. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;LT;` will have no effect).

The line types used are not necessarily compatible with HP-GL. The second parameter "length" is ignored if sent, but is included for HP-GL compatibility.

The default line type used when an item is created is *solid.*

NOTE

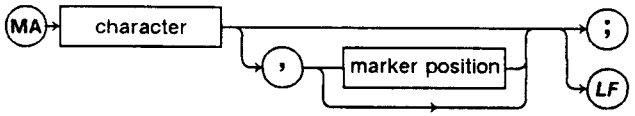
Because the display "stretches" each dot horizontally to take up two dot positions, line type 8 appears solid for horizontal lines. This phenomenon also affects the other line types to some degree. (See appendix B).

MA

MARKER ATTRIBUTES defines the character and orientation attributes of the current marker (see ***MK***).

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
character	marker character	none	any ASCII character	-
marker position	(see text)	centered (0)	see text	8

Response: None

Example:

```
OUTPUT @Display;"MAA,1;"
```

The current marker will be the ASCII character A and the top of the character will be on the point.

Comments:

If in non-referenced mode, this command defines which character of the display's character set (set 0) will be used as the marker in all subsequent non-referenced ***MK*** commands in the window (referenced markers will be the usual default marker). If the ***MA*** command is used as part of a referenced marker then it will define which character will be used for that marker only. The parameter specifies the ASCII character. The initial character for a window is CHR$(255) (a diamond). Position numbers of 1-4 place the marker with its top(1), bottom(2), left(3), or right(4) edge on the point specified by the ***MK*** command. Otherwise the marker is centered on the point specified by the ***MK*** command. The size of the marker character is not adjustable; it is always the smallest character size (size 0).

The ***MA*** command sets the marker attributes for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a marker is drawn, its

attributes cannot be changed. For referenced graphics, changing the attributes at any time will change the marker. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;MAE,0;` will have no effect).

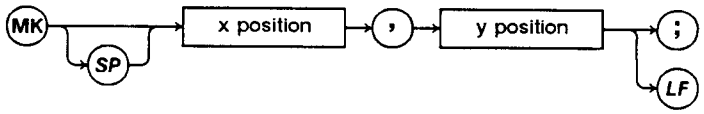
The default attributes used when an item is created are character number 255 (a diamond) and position 0.

MK

MARKER places a marker at the specified coordinates.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
x position	x position in user units of marker	0	-	16
y position	y position in user units of marker	0	-	16

Response:

None

Example:

```
1    ! Program "MK"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"PD;"
40     OUTPUT @Display;"MAA,1;"
50     OUTPUT @Display;"MK200,100;"
90     OUTPUT @Display;"GP1;IT2;MK300,100;"
100    END
```

The character A is placed as a non-referenced marker at the coordinates (200,100) and the default marker (a diamond) is placed as a referenced marker at the coordinates (300,100).

Comments:

This command creates a marker, or references an existing one, and places it on the window. The marker will have the characteristics specified by the ***MA*** command. For non-referenced markers, the marker is placed relative to the window origin. A ***PD*** may need to be done to see the marker if it is non-referenced. For non-referenced graphics, the marker will be placed X units to the right and Y units above the current pen position.

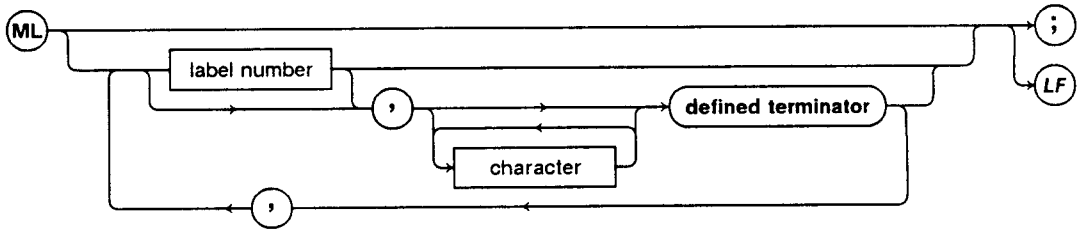
Firmware Revision Dependencies:

In ROM Version 5.0, the defaults are random.

MENU LOAD loads text into the specified menu key.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
label number	softkey to label, 0=clear all	0	0-14	8
character	ASCII character to add to label	-	-	8
defined terminator	set by ***DT*** command	-	-	8

Response: None

Example:

```
10    ASSIGN @Display TO 704
20    OUTPUT @Display;"IN;DT@;"
30    OUTPUT @Display;"ML3,Freq@,4,LONGLABEL@;"
40    END
```

This command will load the text `Freq` into the softkey number 3 label area. Note the terminator @ as defined with the ***DT*** command. The softkey 4 label area will be loaded with the text

```
LONGLAB
EL
```

ML

Comments:

The ***ML*** command loads text into the specified menu key. See Figure 5.1 on page 100 for a guide to the menu key numbers. Each key label is effectively a two line, 7 character per line label, and the rules for filling labels presented under ***LB*** apply here, too, except that a ***CR*** is automatically inserted if an ***LF*** is sent. A key number of 0 causes the display to erase all of the softkeys. The command `ML2;` will blank key 2.

Each key may be labeled with two lines, seven characters each. If the text sent will fit on one line of the label, the label is vertically centered in the softkey window.

To display a two line label, whose first line contains seven characters, no ***LF*** should be sent, as after the seventh character the display will start filling the second line, and a line feed received at the start of the second line will erase the first line. For consistency, *always* sending 14 characters, padding unused spaces with blanks, is the best approach.

Menu labels are always the smallest character size, size 0 (see ***SI***). The alternate character set will be used (if active) in menu labels, but there is no means of switching character sets *within* a label.

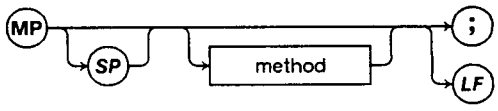
Some control characters and sequences cause special actions. These are listed in appendix F. These actions include display enhancements and special cursor motions among other things, as well as functions like display funcitons on and off.

It may be necessary to leave the `DISPLAY` menus to see the menu labels.

SET MAPPING METHOD sets the mapping method to be used by the **SC** command.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
method	0 = anisotropic 1 = isotropic	0	0 or 1	8

Response:

None

Example:

```
OUTPUT @Display;"MP0;"
```

This selects anisotropic mapping.

Comments:

This command selects either anisotropic or isotropic mapping.

If the window is mapped anisotropically, the user's P1 and P2 will correspond to the lower left and upper right corners of the window. This means that if the user's aspect ratio (ratio of height to width) is not the same as the window's aspect ratio, then a square plotted in the user units will not appear square on the display.

If the window is isotropically mapped, the user's P1 and P2 may not correspond to the corners of the window, but a square plotted in the user's units will always appear square on screen. The corners of the user's units will always lay on either the horizontal or vertical edges of the window. If the aspect ratio of the user's units exactly matches the aspect ratio of the window, P1 and P2 will fall in the corners of the window.

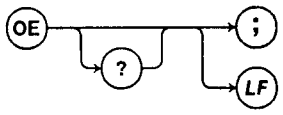
Note that this means that when MP1 is used, squares which appear square on the screen may not appear square in hardcopy dumps, and further, that two squares plotted on displays with different aspect ratios may generate hardcopy dumps that do not match.

The aspect ratio of the graphics on the screen is 1024 x 384 or 2.67 (8 x 3). Because of the compressed horizontal resolution, this would map perfectly on a screen whose aspect ratio was 1.33 (4 x 3). The aspect ratio of the 70205A screen is 4.16 x 3, or 1.39, so the factor to stretch for isotropism is 1.04 or 4%. The aspect ratio of the 70206A screen is 4.36 x 3, or 1.45, so the factor to stretch for isotropism is 1.09 or 9%.

OUTPUT ERROR makes the error number list available for output.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response:

ROM Version 5.0: error number ***CR LF***

The error number of the most recent usage error as an unsigned ASCII integer. 2000 is sent if there are no errors.

ROM Version 6.0 and later: error number list ***CR LF***. The numbers are separated by commas. The entire message is terminated by ***CR LF*** and ***END*** (see page 28). The list is of all usage errors which have occurred since the last time ***OE*** or ***EG*** was received or REPORT ERRORS was pressed, in no particular order. 0 is sent if there are no errors.

A list of errors can be found in appendix C.

Example:

```
1    ! Program "OE"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"OE;"
30     ENTER @Display;Error_number$
40     DISP Error_number$
50     END
```

The response is a list of numbers, for example "2002,2007,2001" or "0" if there are no errors being reported.

Comments:

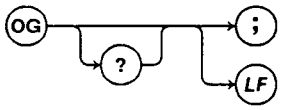
Hardware errors are not output in response to ***OE***, only usage errors. See the ***EG*** command for an example showing how to input the error list. The Error bit in the status byte is cleared when ***OE*** is performed.

OG

OUTPUT GRAPHICS LINK requests the address of a graphics link owner.

HP-GL No
Link Required Any Link

Syntax:



Response:

HP-MSIB Row,HP-MSIB Column ***CR LF***, terminated by ***END*** (see page 28) **Example:**

```
1    ! Program "OG"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"OG?;"
30     ENTER @Display;Row,Column
40     DISP Row,Column
50     END
```

Comments:

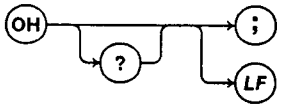
This command is intended for use by a controller whenever the user wishes to know if any HP-MSIB module has a graphics link. If more than one module has a graphics link, only one (not predictable) will be reported. If none, the response will be $-1,-1$. Its primary function is to determine the result of a ***CI*** command (see example under ***CI***).

Firmware Revision dependencies: Version 6.0 or later.

OUTPUT HARD LIMITS requests the hard screen limits from the Display.

HP-GL Yes
Link Required Any Link

Syntax:



Response:

0, 0, 1023, 383 ***CR LF***, terminated by ***END*** (see page 28).

Example:

```
1    ! Program "OH"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"OH?;"
30     ENTER @Display;Limits$
40     DISP Limits$
50     END
```

The response shown above will be returned with the numbers representing the values X1,Y1,X2,Y2 - the horizontal and vertical limits of the screen of the display (P1, P2).

Comments:

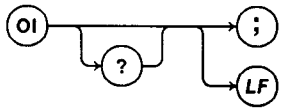
This command is used whenever the user wishes to learn the hard screen limits of the display. The display will respond with its limits expressed in display units.

OI

OUTPUT IDENTIFICATION makes the display identifier available for output to the computer.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response:

70205A ***CR LF*** or 70206A ***CR LF***, terminated by ***END*** (see page 28).

Example:

```
1    ! Program "OI"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"OI?;"
30     ENTER @Display;A$
40     DISP A$
50     END
```

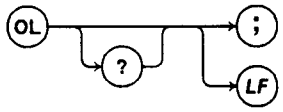
Comments:

This command is used whenever the user wishes to know the display model number. Identical to the ***ID*** command.

OUTPUT LEARN STRING requests that a display output the information required to return it to the state it is presently in.

HP-GL	No
Link Required	Control Link

Syntax:



Response:

A learnstring: IL plus data in "#I" notation.

Example:

See the ***IL*** command.

Comments:

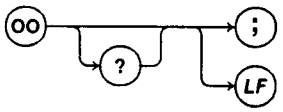
When this command is received, the device will output information required to return it to the state (see ***SV*** command) it is currently in, including all the saved states. When this message is sent back to the display with no additional headers or trailers, the display will return to the state it was in when the learn string was sent. See the ***IL*** command for further discussion.

OO

OUTPUT OPTIONS outputs the display options.

HP-GL Yes
Link Required Any Link

Syntax:



Response:

0,1,0,0,0,0,0,0 ***CR LF***, terminated by ***END*** (see page 28)

Example:

```
1    ! Program "OO"
10     OPTION BASE 1
20     DIM Options(8)
30     ASSIGN @Display TO 704
40     OUTPUT @Display;"OO?;"
50     FOR I=1 TO 8
60       ENTER @Display USING "#,K";Options(I)
70       PRINT "Option ";I;" = ";Options(I)
80     NEXT I
90     END
```

The following array will be shown on the computer:

```
Option 1 = 0
Option 2 = 1
Option 3 = 0
Option 4 = 0
Option 5 = 0
Option 6 = 0
Option 7 = 0
Option 8 = 0
```

Comments:

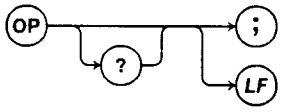
The ***OO*** command is designed to accommodate extensions to the display language at such time as capability sets change. The field definitions do not necessarily correspond to those defined for HP-GL.

OP

OUTPUT P1,P2 requests the location of the lower left (P1) and upper right (P2) vertices of the window.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response:

P1x,P1y,P2x,P2y in display units for the window, as signed ASCII integers, separated by commas, terminated by ***CR LF*** and ***END*** (see page 28).

Example:

```
1    ! Program "OP"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"OP?;"
30     ENTER @Display;P1x,P1y,P2x,P2y
40     DISP P1x,P1y,P2x,P2y
50     END
```

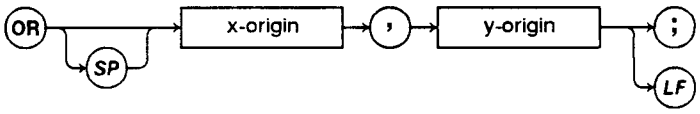
Comments:

This command is used whenever a module or controller wishes to determine the coordinates of the lower left and upper right vertices of its window.

SET ORIGIN sets an offset that will be added to all objects before they are drawn.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
x-origin	x-coordinate of window, group, or item	-	-	16
y-origin	y-coordinate of window, group, or item	-	-	16

Response:

None

Example:

```
OUTPUT @Display;"GP1;IT1;OR30,20;"
```

The display is commanded to add the values 30,20 to the coordinates of item 1 in group 1. Thus, if the display is instructed to plot point 35,53 it will actually plot point 65,73.

```
OUTPUT @Display;"GP1;OR10,30;IT1;OR30,20;"
```

The display will place the object referenced by item 1 at an offset of 40 x-units and 50 y-units.

```
OUTPUT @Display;"GP0;OR10,30;"
```

The display will add 10 and 30 respectively to the x and y coordinates of any coordinates sent it in non-referenced graphics.

OR

Comments:

The ***OR*** command allows the setting of the origin of the active group or item, or of the non-referenced graphics origin. This allows groups and items to be independently moved around within a window (see Chapter 4 and the ***GP*** and ***IT*** commands for a discussion of referenced graphics and groups and items).

For non-referenced graphics (GP0), when an ***OR*** command is sent to set x-origin and y-origin, the (non-referenced) pen is effectively moved to that point (without drawing any line). Thus, all subsequent non-referenced objects will have the values x-origin, y-origin *in user units* added to their x and y values before they are drawn; if the display is directed to plot (x,y) it will plot (x + x-origin, y + y-origin) instead.

For referenced graphics, each group has its own origin, as does each item. When a group or item is created, the origin defaults to (0,0) in user units. An ***OR*** command can then be sent to set the group or item's origin. When the referenced object is drawn, to the x and y coordinates of the item are added the origin of the group and the origin of the item (and the origin of the window, if it has been set by the ***SC*** command) before the item is drawn.

Once the origin of a group or item has been specified, it can be changed at any time. The effect of changing the origin is to immediately shift the position of the group or item on the screen. The various origins may be thought of as relative positioning points: the origin of a group is relative to the origin of the window, and the origin of an item is relative to the origin of the group that contains the item. The origin of the window is set by the ***SC*** command, or is the lower left corner of the screen, if no ***SC*** command has been sent (see appendix E).

WARNING

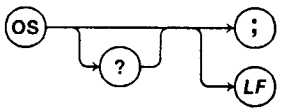
The physical location of a group's reference point on the screen (its origin) is determined by the scale factor in effect at the time the group's origin is *specified* (that is, when the group or item is defined or when the ***OR*** command is received). Changing the window's scale factor (with ***SC***) will *not* move the object (although it may change its x or y size) until the origin is re-specified. See the ***SC*** command description and appendix E for more on this.

The ***OR*** command requires that parameters be sent. No default values exist for x-origin and y-origin.

OUTPUT STATUS makes the status byte available for output to the computer or instrument.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response:

An unsigned ASCII integer, terminated by ***CR LF*** and ***END*** (see page 28) which is the display's status byte. See appendix C for a discussion of the display's status byte.

Example:

```
OUTPUT @Display;"OS"
ENTER @Display; Status
```

Comments:

The **OS** command makes the status byte available for output to the device or element. For HP-IB this is the same byte that is sent in response to a serial poll, except that bit 6 will never be on, but *will* be on in the serial poll status byte if SRQ is being asserted. For HP-MSIB, this is the same byte that goes out as the ***STATUS*** HP-MSIB command (see the *Communication Protocol Design Guide*). In Rom Version 6.0, the status byte is cleared after an **OS**. SRQ is not cleared on an **OS**. Appendix C lists the conditions under which SRQ and the status byte are cleared.

Note

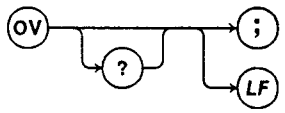
The display has a separate status byte for each window (See Chapter 4).

OV

OUTPUT VERSION asks for the firmware version and the date code.

HP-GL No
Link Required Any Link

Syntax:



Response:

A string identifying the firmware version and the date, terminated by ***CR LF*** and ***END*** (see page 28).

For 5.0, the string will be

```
ROM Version 5.0  Feb, 1985
```

For 6.0, it is

```
860625   ROM Version 6.0
```

Example:

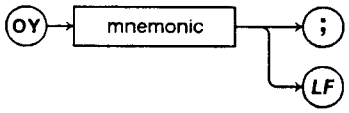
```
1    ! Program "OV"
10     ASSIGN @Display TO 704
20     DIM Version$[30]
30     OUTPUT @Display;"OV?;"
40     ENTER @Display;Version$
50     DISP Version$
60     END
```

Comments: The form used in Rom Version 6.0 is the form to be used in all future firmware versions.

OUTPUT CAPABILITY provides the means to determine if a particular display recognizes a given command.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description
mnemonic	a two-letter mnemonic to test

Response:

The unsigned ASCII integer 0, 1, or 2, terminated by ***CR LF*** and ***END*** (see page 28). Their meaning is explained below.

Example:

```
10    ASSIGN @Display TO 704
20    OUTPUT @Display;"OYAX;"
30    ENTER @Display;Recognize
40    DISP Recognize
50    END
```

The display is being asked if it has capability to recognize and respond to the ***AX*** command. In Rom Version 6.0, the value 2 will be returned.

Comments:

The ***OY*** command provides the means to determine if a particular display recognizes a given command. The parameter is the two letter mnemonic for the command of interest. The display will respond with an integer output which indicates the display's capability as follows:

response	meaning
0	Unrecognized Command.
1	Recognized Command, Not Implemented.
2	Recognized Command, Implemented.

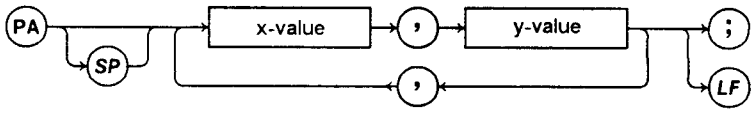
OY

Firmware Revision dependencies: A bug in Version 5.0 caused the display to respond 1 if it simply recognized a command, even if it was not implemented.

PLOT ABSOLUTE moves the pen to the position specified.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
x-value	x-coordinate of point to which pen will be moved	none	-	8
y-value	y-coordinate of point to which pen will be moved	none	-	8

Response: None

Example:

```
1    ! Program "PA1"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;PD;"
30     OUTPUT @Display;"PA100,40,200,50;"
40     OUTPUT @Display;"GP1;IT1;PA0,50,450,225;"
50     OUTPUT @Display;"IT2;PA 0,100,50,125;PU;PA100,150;PD;PA150,175"
60     END
```

The ***PA*** command on line 30 will move the pen from its initial location (0,0) to the point (100,40) and then to the point (200,50), resulting in *two* straight lines being drawn. The ***PA*** command on line 40 is in referenced graphics mode (see below) and thus the pen will be put down at the point (0,50) and moved to the point (450,225) resulting in *one* straight line being drawn. The ***PA*** command on line 50 will put the pen down at the point (0,100) and move it to the point (50,125), then lift it and move it to the point (100,150), then put it down again and move it to the point (150,175), resulting in *two* straight lines being drawn with a gap between them.

This example generates the identical screen as the example under the ***GA*** command.

PA

Binary mode example (see also page 26):

```
1    ! Program "PA2"
10     ASSIGN @Display TO 704
20   ! This is equivalent to "PA 0,0,600,300":
30     X1$=CHR$(0)&CHR$(0)    ! (256*0)+0=0
40     Y1$=CHR$(0)&CHR$(0)    ! (256*0)+0=0
50     X2$=CHR$(2)&CHR$(88)   ! (256*2)+88=600
60     Y2$=CHR$(1)&CHR$(44)   ! (256*1)+44=300
70     OUTPUT @Display;"DE;IN;PD;IT1;PA #I"&X1$&Y1$&X2$&Y2$;
80     SEND 7;DATA 0 END      ! Send end AFTER last byte
90     END
```

This example generates the identical screen (one straight line) as the binary example under the ***GA*** command.

Comments:

The ***PA*** command is one of the basic line drawing commands in the display (the others are ***PR, GA*** and ***GR***). These commands are used to draw *traces*, which are interconnected series of dots on the screen. As described in Chapter 4 and under the ***GP*** and ***IT*** commands, there are two basic types of graphics in the display: *referenced* and *non-referenced* graphics.

In non-referenced graphics (`GP0` or `IT0`), lines are always drawn from the *current pen position*. The pen starts in the lower left corner of the screen, in the UP position.[5] If an ***OR*** command is sent during non-referenced graphics, it moves the pen to the new origin but does not draw a line. The pen state (up or down) is not changed. Non-referenced graphics mode is designed to operate like a pen plotter, for HP-GL compatibility. Thus, in this mode, objects are drawn and essentially forgotten; they can not be moved, or set to blink, or blanked; it is as though they had been drawn with a pen onto a piece of plotter paper. Once drawn they will not change until the window is initialized or deleted or a ***DA*** command is sent. When group 0 is being used, we say that there is no active group.

In referenced graphics, there is no starting pen position for lines. The first plotting command (***PA***, ***GA***, and so on) sets the pen start position and lowers the pen. Thus it requires at least two such commands to draw one line. This eliminates the need for a ***PU*** to set the initial pen position, but it does make referenced graphics different from non-referenced graphics. Objects once drawn can be moved around singly (as items) or in collections of objects (as groups). Thus, setting the scaling and the origins of groups and items implicitly moves the pen around the screen.

[5]This means that a ***PD*** command must always be issued in non-referenced graphics (Group 0) to see any data on the screen.

In non-referenced graphics, however, sending **SC** and **OR** commands will not move the pen; it has to be moved as desired with a **PA** or other plotting command. This can be confusing unless you remember that non-referenced graphics are included for HP-GL compatibility, and that is exactly the way pen plotters work; the only way to move the pen is with a plotting command, not a scaling or origin-setting command. That is why when you initialize a window, set the scaling and the origin, and drop the pen, then send your first **PA** command, the display draws a line from the lower left corner of the screen, which is where the pen starts, to your first point; even if your first point is (0,0) in your scaled user units. The pen does not move until you move it. The proper way to start a plot in non-referenced graphics therefore, is with a `PA x,y;PD;` where (x,y) are the desired starting coordinates in user units.

Note that graphics may be plotted anywhere on the screen, and are not necessarily restricted by window size. To scale a window so that subsequent graphics will relate to the window, the **SC** command is used. An **SC** command has no impact on the pen position or on graphics already on the screen; it will impact subsequent plotting commands only (see the **SC** command description and appendix E).

As with all commands, the data may be sent to the display in binary (#I) format (see page 26). Note that you cannot mix **PA** and **PR** commands (or **GA** and **GR** commands) in a single item as they are distinct item types (see **IT**).

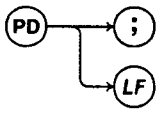
The display does not clip graphics done outside a window; they appear on the screen whether physically within the window or not.

PD

PEN DOWN lowers the pen without moving it to a new location.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response:

None

Example:

```
OUTPUT @Display;"PD;"
```

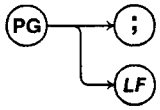
Comments:

This command lowers the sending window's "pen" (or that of the active group or item) without moving it to a new location. The "pen down" bit in the status byte is set (see appendix C). After ***PD***, lines drawn on the screen will be seen, until a ***PU*** command is sent. In referenced graphics (see ***GP*** and ***IT*** and chapter 4), items are created with the pen down. In non-referenced graphics a ***PD*** must be sent before any lines will be seen.

PAGE erases all information presented on the window.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response: None

Example:

```
OUTPUT @Display;"PG;"
```

All information presented on the window will be erased and all of its associated data will be lost. The configuration of the window will be unchanged. All groups and items become undefined, and non-referenced graphics is switched on.

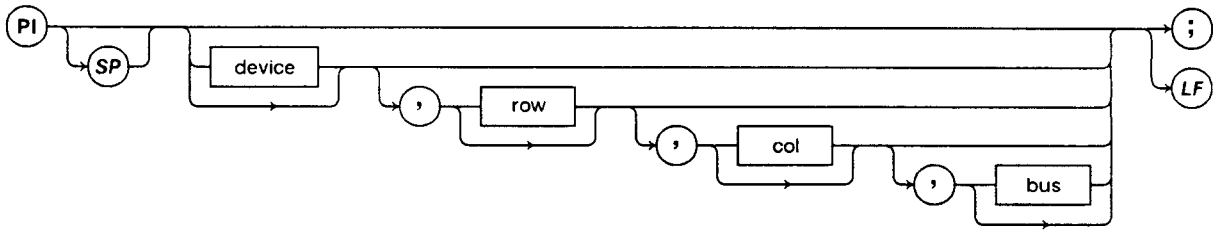
This command is the equivalent of a page eject on a plotter.

PI

PRINTER/PLOTTER IS specifies the printer or plotter for use with the **CY** function.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
device	0=printer,1=plotter	0	0-12	8
row	device HP-MSIB row address	current *	0-7	8
col	device HP-MSIB column address	current *	0-31	8
bus	0=HPIB, 1=HP-MSIB	current *	0,1	8

* The term "current" refers to the current address of the printer or plotter, depending on which device is selected by the first parm.

Response: None

Example:

```
OUTPUT @Display;"PI0,,31,0;"
```

The display is informed that the dump device is a listen-only printer on HP-IB. Future **CY** commands will trigger printing to that device.

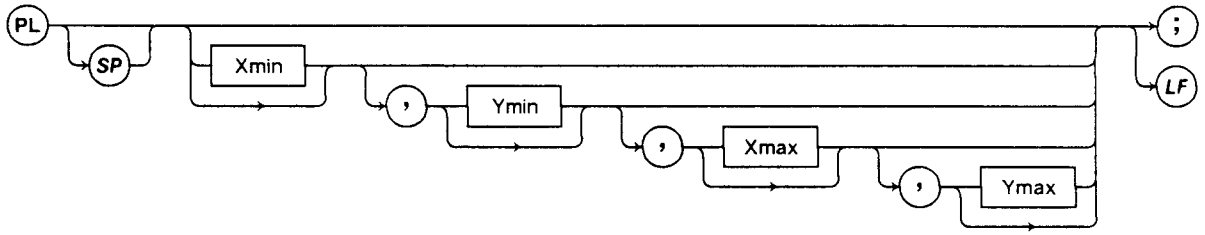
Comments:

This command specifies the dump device by address, bus, and type for the display to use whenever the **CY** command is invoked. On HP-IB, the HP-MSIB column address is used as the HP-IB address. If the bus is HP-IB and the column address is 31, address is HP-IB Listen Only.

PLOTTER LIMITS sets the points P1,P2 in memory to be used for "listen-only" type plotter dumps.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
Xmin	x-coord of P1	value currently stored in display	-	16
Ymin	y-coord of P1	value currently stored in display	-	16
Xmax	x-coord of P2	value currently stored in display	-	16
Ymax	y-coord of P2	value currently stored in display	-	16

Response:

None

Example:

```
OUTPUT @Display;"PL 100,500,100,1000"
```

The display will scale its plot data to the rectangle formed by the points (100,100) and (500,1000) on the plotter, during plotter dumps.

Comments:

This command sets the points P1,P2 to be used instead of the plotter's P1,P2 for plotter dumps. The rectangle formed by the points P1 and P2 is the portion of the plotter to which the entire display is plotted.

Under most circumstances, these values will not be used, because the display will ask the plotter to send its P1,P2 (the ones the plotter user has set up) over the bus so it can

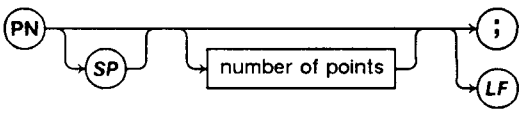
plot to the desired area. The times when this will not happen, and the ***PL*** parameters will be used instead, are: Plotter is on HP-MSIB; Plotter is configured in "`define hardcopy`" as a Listen Only device; ***CY*** has been sent with parameter 1; ***CY*** has been sent and display is not a system controller (6.0 and later only).

The parameters are initialized (when a window is opened) to the values: Xmin, 100; Ymin, 100; Xmax, 10100; Ymax, 7600. These plot an 8 1/2 x 11 inch picture on an HP 9872 plotter.

PAN allows panning of a graph.

HP-GL	No
Link Required	Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
number of points	amount to pan trace	0	-	16, signed

Response: None

Comments:

The ***PN*** command is used to shift an entire graph horizontally on the screen by the number of endpoints specified. The shift is to the right for positive parameters and to the left for negative parameters. The points that are panned past the last endpoint in the trace are lost, while the points "panned in" are blanked since there is no information to present.

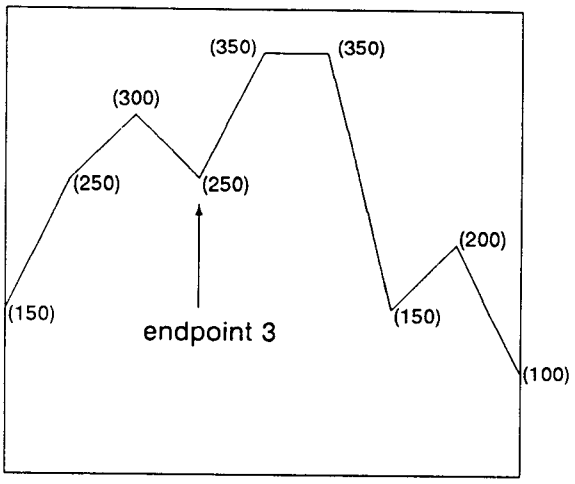
PN only works with traces drawn with ***GA*** and ***GR***, not with ***PA*** or ***PR***.

Example:

This example demonstrates the ***PN*** command, and how the ***PN*** and ***TP*** commands interact. Let's say a graph has been drawn with a delta-x (***DX***) of 100 and that the current endpoint (see ***TP***) is endpoint 3. This will in fact be the case if the following program is run:

```
1    ! Program "PN"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;SC 0,800,0,400;"                ! Scale window
30     OUTPUT @Display;"IT1;PA0,0,0,400,800,400,800,0,0,0;" ! Draw frame
40     OUTPUT @Display;"IT2;DX 100;"
50     OUTPUT @Display;"GA 150,250,300,250,350,350,150,200,100;"
60     OUTPUT @Display;"TP3;"
70     PAUSE
80     END
```

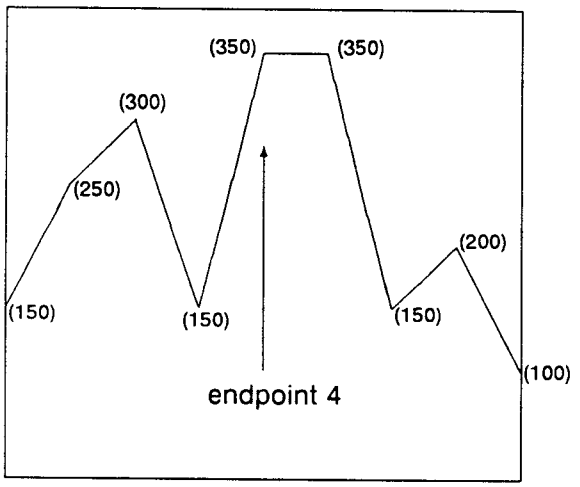
The resultant graph will appear as shown in the figure below (the numbers by the points show the y-coordinates of each point):



Now let's say that

```
OUTPUT @Display;"GA 150;"
```

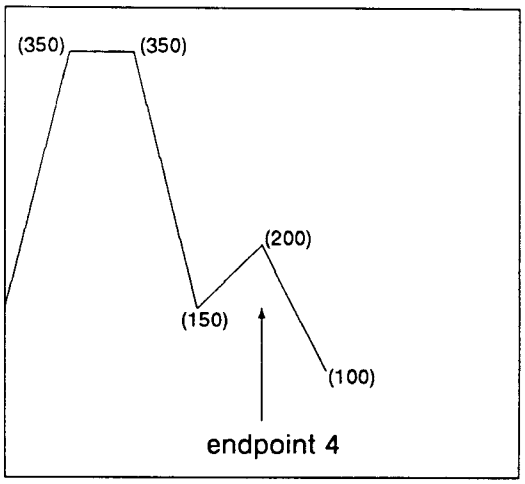
is executed. This causes the segment from endpoint 2 to endpoint 3 to change so that it goes to the point with y-coordinate 150 (the x-coordinate is as before). It also causes the current endpoint to change to endpoint 4. The graph will now appear as shown below:



Now let's say that

```
OUTPUT @Display;"PN -3;"
```

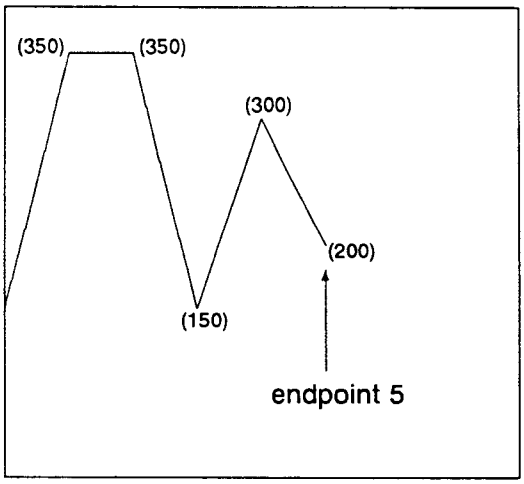
is executed. The graph will then appear as shown below, shifted three endpoints to the left:



Note that the current endpoint is still endpoint 4, but endpoint 4 is at a different point on the trace because the trace has shifted. Now suppose that

```
OUTPUT @Display;"GA 300;"
```

is executed. The current endpoint will become endpoint 5, and the graph will appear as shown below:



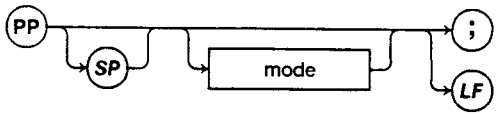
This demonstrates that if, after panning, it is desired that graphing will continue at the same point in the trace as before, the trace pointer must be updated to reflect the pan. This means that one has to keep track of the trace pointer in those cases.

PP

PRE-PROCESS MODE allows the controller to receive all keyboard inputs, no matter which element owns the keyboard.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=0ff, 1=0n	0	0 or 1	8

Response: None

Example:

```
OUTPUT @Display;"PP1;"
```

All key presses go to the controller (except as noted in the text, below).

The program below shows how to implement an interrupt-driven program which will insert an HP-IB controller between a module and its keyboard:

```
1    ! Program "PP"
10     ASSIGN @Display TO 704
20     REMOTE 7                              ! Must be in REMOTE to use PP
30     OUTPUT @Display;"IM 255,160;"  ! Pull SRQ on key or error
40  Idle:ENABLE INTR 7;2                     ! Interrupt on SRQ
50     OUTPUT @Display;"PP1;"                ! PP on
60     ON INTR 7 GOSUB Key_read              ! Set up interrupt vector
70     GOTO Idle                             ! Loop
80     !
90  Key_read:     !
100    IF BIT(SPOLL(@Display),7) THEN        ! Key pressed?
110      OUTPUT @Display;"KY"                ! Get key number
120      ENTER @Display;Key_number
130      PRINT "Key ";Key_number;" pressed."
140      OUTPUT @Display;"KP ";Key_number  ! "Press the key"
150      BEEP
160    END IF
170    RETURN
180    !
190    END
```

Comments:

Key presses and knob turns normally go to the keyboard owner (***PP*** off). When in ***PP*** mode and when in remote on HP-IB or when the responder of an HP-MSIB control link (see the *Communication Protocol Design Guide*), all input goes to the controller whether or not it owns the keyboard. The controller can then do what it wishes with the input. By using the ***KP*** command it can send these inputs on to the keyboard owner.

In this mode, a module which owns the keyboard and a graphics window may label the softkeys and update the graphics but the physical key presses get sent to the controller. The controller can then "pre-process" these inputs by either ignoring them, performing its own processing, or echoing them to the active element. The controller echoes keys to the active element by sending the ***KP*** (Simulate Key Pressed) command to the display. When the display receives these commands, it performs the same actions as if the key were pressed and it was not in the pre-process mode.

As an example, with an HP-IB controller and key press interrupts enabled, the sequence of events can be:

1. user presses a key
2. Status sent to controller, using **SRQ** status byte sequence on HP-IB, or HP-MSIB ***STATUS*** command on HP-MSIB.
3. controller sends ***KY*** command asking "which key" was pressed.
4. display sends key number to controller.
5. controller decides to take some action, then echo key press. Sends ***KP*** command with the same integer key number back to the display.
6. display then sends the HP-MSIB ***STATUS*** command to the keyboard owner, with the "key pressed" bit set.
7. module sends ***KY*** command asking "which key"?
8. display sends key number to module
9. module performs normal manual action.

The instrument is unaware of steps 2 through 5. Steps 1 and 6 through 9 would still occur if the display were not in pre-process mode.

This sequence assumes an applications program in the controller which needs to know about and participate in the key transactions between module and display.

PP

Note that pressing the DISPLAY (DSP) key, LOCAL (LCL) key, or PRINT or PLOT hardkeys does not generate a keycode; these keys are intercepted by the display system and not passed on.

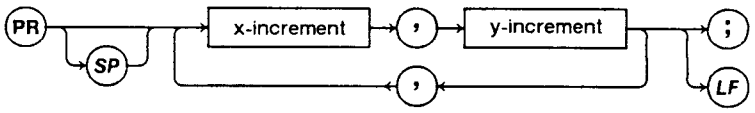
Modules on HP-MSIB would not use ***PP*** to service the keyboard in the display; they would use ***KY*** as shown in the description of ***KY***. ***PP*** is intended for use by controllers who want to come between a module and its keyboard.

PP has no effect on the knob or knob count communications.

PLOT RELATIVE moves the pen to a new location relative to its present location.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
x-increment	change in x-coordinate value	none	-	16
y-increment	change in y-coordinate value	none	-	16

Response: None

Example:

```
1    ! Program "PR"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"PD;"
40     OUTPUT @Display;"OR50,50;"
50     OUTPUT @Display;"PR300,100,50,75;"
60     OUTPUT @Display;"GP1;IT1;MK400,225;"
70     END
```

Two lines will be drawn connecting the points (50,50), (350,150), and (400,225). A marker will be placed at the endpoint of the second line.

Comments:

This command causes the pen to move to new locations determined by the x-increment and the y-increment values. If the current location is (x,y) then the new location will be (x + x-increment,y + y-increment). If the pen is down a straight line will be drawn between the current location and the new location. However, if the ***PR*** command is the first part of a referenced graph then the pen will be put down at the new location.

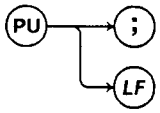
As with all commands, the data may be sent to the display in binary (#I) format (see page 26). See also the discussion of plotting under the ***PA*** command.

PU

PEN UP raises the pen without moving it to a new location.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Response:

None

Example:

```
OUTPUT @Display;"PU;"
```

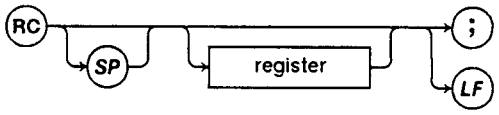
Comments:

This command raises the sending window's "pen" (or that of the active group or item) without moving it to a new location. The "pen down" bit in the status byte is cleared (see appendix C). After ***PU***, lines drawn on the screen will not be seen, until a ***PD*** command is sent. In referenced graphics (see ***GP*** and ***IT*** and chapter 4), items are created with the pen down. In non-referenced graphics a ***PD*** must be sent before any lines will be seen.

RECALL STATE recalls a setup from non-volatile memory and has the display go to that state.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
register	0 = off, otherwise on	1	1-4	8

Response: None

Example:

See the **SV** command for a complete example.

Comments:

When ***RC*** is received the display recalls the setup in non-volatile memory. See the ***SV*** command description for details on what makes up this state.

Firmware Revision dependencies:

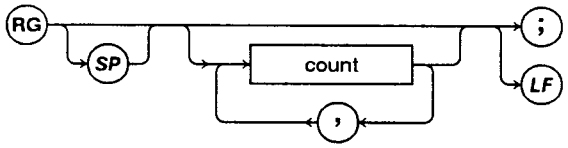
In ROM Version 5.0, sending ***RC*** with no parameters generates an error.

RG

SIMULATE RPG TURNED simulates turning of the knob.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
count	number of counts to send	0	-	16

Response: None

Example:

```
OUTPUT @Display;"RG30;"
```

The display will react as if the knob had been turned in the clockwise direction by a count of 30.

Comments:

If the status mask is set appropriately (see ***IM***), then when the knob is turned, a status byte is sent to the keyboard owner alerting it to that fact. If the keyboard is assigned to HP-IB, this will be in the form of an **SRQ**, requiring a serial poll or an **OS** to read the status byte. If a module on HP-MSIB owns the keyboard, the status byte will come in the form of ***STATUS*** HP-MSIB command.

If the module does not send back an ***RP*** command after the display sends the ***STATUS*** command, and the knob continues to turn, the display will send ***STATUS*** messages at about 200 ms intervals. If the module responds instantly to the ***STATUS*** message, and the knob continues turning, the display will send another ***STATUS*** message right away. However, there will always be at least 15 ms between ***STATUS*** messages (see also page 12).

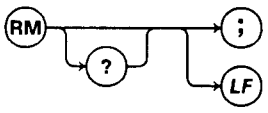
See the ***KP*** command for a discussion of what the display does with the count.

Firmware Revision dependencies: In Revision 6.0, a bug prevents this command from working.

REMAINING MEMORY determines memory available in the display.

HP-GL No
Link Required Any Link

Syntax:



Response:

A string containing two ASCII integers, separated by a comma and terminated by ***CR LF*** and ***END*** (see page 28).

Example:

```
1    ! Program "RM"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"RM;"
30     ENTER @Display;Numblocks,Blocksize
40     DISP Numblock,Blocksize
50     END
```

Comments:

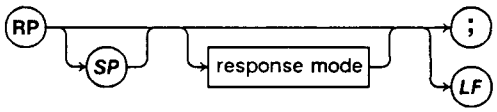
This command is used to determine the amount of available memory in the vector list. **Numblocks** is the number of blocks of memory left in the vector list. **Blocksize** is the number of 16-bit words in each block (always 128). See Chapter 4 for more on the display's memory.

RP

SEND RPG DATA asks the display for the accumulated knob count since the last reading of the knob count.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
response mode	1=binary, else ASCII	0	-	8

Response:

knobcount, a signed 16 bit binary integer or a signed ASCII integer, terminated by ***CR LF*** and ***END*** (see page 28).

Example:

```
1    ! Program "RP"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"RP;"
30     ENTER @Display;Countnum
40     DISP Countnum
50     END
```

The response will be an integer indicating the accumulated count in ASCII since the last ***RP*** was sent.

Comments:

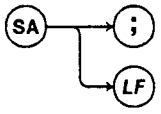
The display notifies the controller or keyboard owner that the knob has been turned by setting the knob bit of the status byte. If the controller then uses the ***RP*** command it can learn by how much and in which direction (positive is clockwise) the knob has been turned. The response may be either in ASCII or binary (#I) format.

RP clears the "knob count available" bit of the status byte (see appendix C).

SELECT ALTERNATE CHARACTER SET selects the alternate character set as the character set to be used for all subsequent characters.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response:

None

Example:

See the ***CA*** command for an example program that demonstrates the use of the ***CA, CS, SA*** and ***SS*** commands.

Comments:

The ***SA*** command causes the display to begin writing all labels and markers for the sending window using the alternate character set specified by the ***CA*** command. It will continue to use this set until the power is turned off, the display is initialized (DISPLAY PRESET, SELECT INSTR, DEVICE CLEAR, ***DF***, ***IN***), or the ***SS*** command is used. Note that the alternate character set is used only to create labels and markers, and all commands (including the label terminator) must still be sent in the standard US ASCII character set.

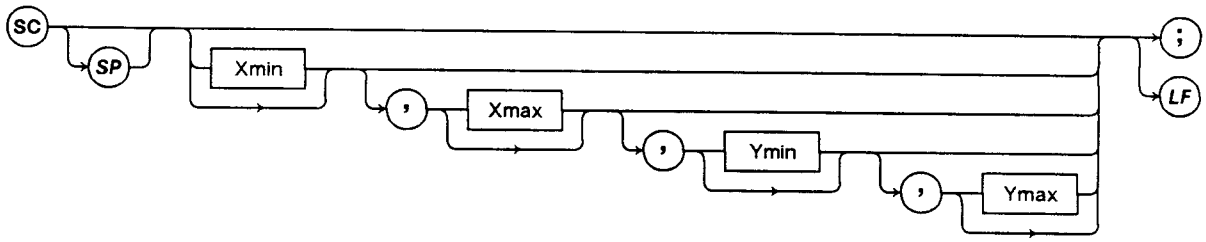
Once drawn, switching character sets will not change a label. It has to be redrawn under the new character set.

SC

SCALE TO USER UNITS establishes a user-unit coordinate system by mapping the P1 and P2 (corner) points of a window onto the scaling points P1 and P2.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
Xmin	user's lower-left X-coordinate (P1X)	current	-	16
Xmax	user's upper-right X-coordinate (P2X)	current	-	16
Ymin	user's lower-left Y-coordinate (P1Y)	current	-	16
Ymax	user's upper-right X-coordinate (P2Y)	current	-	16

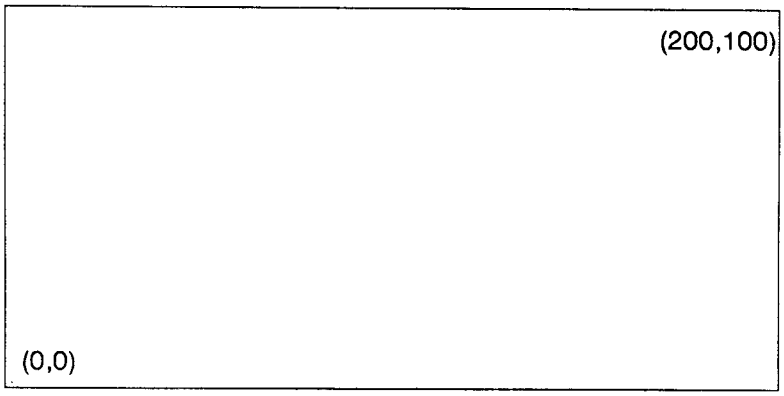
Response:

None

Example:

```
OUTPUT @Display;"SC 0,200,0,100;"
```

The window, no matter where it is or what its dimensions are, may now be thought of by the programmer as being a 200 x 100 window with the lower left corner at (0,0), as seen below.



Comments:

This command allows scaling of the window to units convenient to an application. It basically allows the programmer to define the corner points of the window to be whatever is desired, thus establishing new dimensions for the window in what are called *user units*. The parameters Xmin and Ymin define the user-unit coordinates of P1 (lower left corner of window) and the parameters Xmax and Ymax define the user-unit coordinates of P2 (upper right corner of window). P1 and P2 may be any two opposite corners of a rectangle. The **SC** command affects *window* parameters and applies to all objects, groups, and items in the window sent *while it is in effect.*

Until an **SC** command is sent to a window, any graphics done to the window will be in *display units*, that is, in the physical coordinates of the display screen (see appendix B, where these are defined). In other words, if a line is drawn to point (0,0) it will go to the lower left corner of the *screen*, which is the point (0,0) in display units. This will be true *even if the window is up in the middle of the screen* and doesn't even *include* point (0,0) in display units within it. Until an **SC** is sent, the lower left corner of the *window* is *not* (0,0) even though there is an inclination on the part of the programmer to think of it that way. What this means is that, no matter where you build a window, your graphics will use the whole display screen and be plotted in display units until you send an **SC**. Even if you build a small window in the upper left corner of the screen, a line drawn from (0,0) to (1023,383) will go from the lower left corner of the screen to the upper right corner.

Note that since the default parameters for **SC** are the current P1 and P2 for the window on whose I/O channel the command was sent, sending **SC** with no parameters establishes user units equal to the display units. Thus the default that exists when a window is built can be returned to at any time by sending an `SC;`.

SC

Once an **SC** has been sent, the window will be scaled to the coordinates of the **SC** command. All graphics will now be in *user units*, the lower left corner of the window will take on the P1 value set by the **SC** command (Xmin,Ymin), and the upper right corner of the window will take on the P2 value set by the **SC** command (Xmax,Ymax). If another, different **SC** is later sent, it will affect all subsequent graphics *but will not affect lines already drawn on the screen.*

NOTE

No pen movement is implied by the **SC** command. The physical point on the screen where the pen is currently situated does not change when the window is re-scaled - merely where it will go on the *next* graphics command.

In referenced graphics, once a scale factor and item origin have been set, sending graphics to that item will cause it to be drawn where its origin is specified, thus providing implicit pen movement. It is as though there is a pen per item. But in non-referenced graphics (GP0), there is only one "pen", and sending **SC** and **OR** commands will not move it: it has to be moved as desired with a **PA** or other plotting command. This can be confusing unless you remember that non-referenced graphics are included for HP-GL compatibility, and that is exactly the way pen plotters work: the only way to move the pen is with a plotting command, not a scaling or origin-setting command.

It is important to understand the order in which the display performs graphics operations related to the **SC**, **OR** and **BW** commands, and graphics commands like **GA, PA, AX,** and so on. Only by gaining such an understanding will the proper order in which to send these commands become clear. This is explained in appendix E. The programmer should read, execute and understand the examples in appendix E to gain a clear understanding of how these commands interact. The basic rule of thumb is:

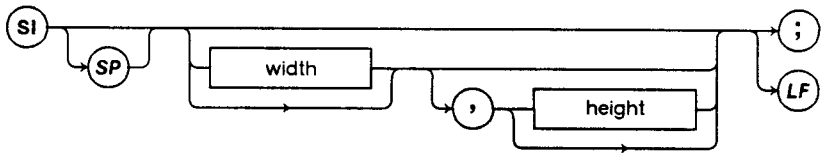
RULE

Send the **SC** command *before* you define or send data to any group or item to which you wish that **SC** command to apply.

SET ABSOLUTE CHARACTER SIZE specifies the size of characters and symbols in display units.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
width	character width in display units (dots)	15	0-246	16
height	ignored			

Response:

None

Example:

```
OUTPUT @Display;"GP1;IT1;SI30;"
```

The characters in group 1, item 1 will be 30 units wide.

Comments:

This command sets the absolute character size for use in ***LB*** commands. For HP-GL compatibility, this command's parameters specify size in terms of the width in dots of the characters (the **height** parameter is ignored but is included for HP-GL compatibility). The display stores character size as a value from 0 to 15, where size 0 is the smallest size available. The display will choose the closest character size that fits the **width** parameter, according to the table below. See appendix D for more on character sizes.

width parameter	size	character cell width	character cell height
0 - 21	**0**	15	16
22 - 36	**1**	30	32
37 - 51	**2**	45	48
52 - 66	**3**	60	64
67 - 81	**4**	75	80
82 - 96	**5**	90	96
97 - 111	**6**	105	112
112 - 126	**7**	120	128
127 - 141	**8**	135	144
142 - 156	**9**	150	160
157 - 171	**10**	165	176
172 - 186	**11**	180	192
187 - 201	**12**	195	208
202 - 216	**13**	210	224
217 - 231	**14**	225	240
232 - 246	**15**	240	256

Note that because the default width parameter is 15 dots, the default character size is size 0.

The ***SI*** command sets the character size for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a label is drawn, its size cannot be changed. The size of a label will be the size in effect at the time the label is built. For referenced graphics, changing the size at any time will change the label size. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;SI30;` will have no effect).

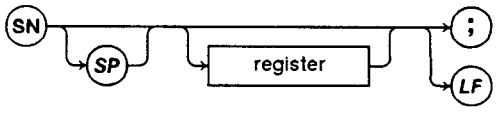
See Chapter 4 and the ***GP*** and ***IT*** commands for more on referenced graphics and groups and items.

The default character size used when an item is created is size 0.

SHOW CONFIGURATION remotely invokes the display's `show config` function.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
register	storage register to display	-1	-1 to 4	8

Response: None

Example:

```
OUTPUT @Display;"SN1;"
```

The display will show the first screen storage register.

Comments:

The display maintains five internal registers to store window configurations (see ***SV*** and ***RC***). The `show config` function allows these registers to be viewed.

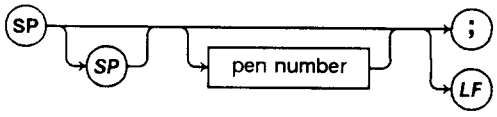
A parameter of -1 terminates this function and returns the screen to the graphics windows.

SP

SELECT PEN selects a pen for drawing subsequent objects.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
pen number	pen to use	1	-	8

Response:

None

Example:

```
OUTPUT @Display;"IT3;SP2;"
```

The pen number for item 3 will be pen 2.

Comments:

The ***SP*** command sets the pen number for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once an object is drawn, its pen number cannot be changed, but will be the pen number that was in effect at the time the object was drawn. For referenced graphics, changing the pen at any time will change the item's pen number. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;SP1;` will have no effect).

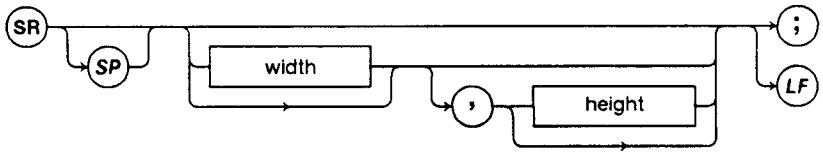
Although the display does not have multiple colors, the pen information is retained and used for plotter dumps.

The default pen used when an item is created is pen 1.

SET RELATIVE CHARACTER SIZE specifies the size of characters and symbols in user units.

HP-GL Yes
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
width	character width as a percent of window width	whatever will yield 15 dot wide characters	see text	16
height	ignored			

Response:

None

Example:

```
OUTPUT @Display;"GP1;IT1;SR3,3;"
```

Both parameters are set to 3% so all subsequent characters in Group 1, Item 1 will have width equal to as close to 3% of the width of the window in which they appear as the display can come.

Comments:

This command sets the absolute character size for use in ***LB*** commands. For HP-GL compatibility, this command's parameters specify size in terms of the width of the window (the **height** parameter is ignored but is included for HP-GL compatibility). The display stores character size as a value from 0 to 15, where size 0 is the smallest size available. The display will choose the closest character size that fits the **width** parameter, according to the table below. See appendix D for more on character sizes.

SR

In the table below, multiply the **width** parameter by the width of the window in display units to get the value for the first column. If this result is greater than 246, a **Parameter range** error will be declared.

width parameter × window width in display units	size	character cell width	character cell height
0 - 21	**0**	15	16
22 - 36	**1**	30	32
37 - 51	**2**	45	48
52 - 66	**3**	60	64
67 - 81	**4**	75	80
82 - 96	**5**	90	96
97 - 111	**6**	105	112
112 - 126	**7**	120	128
127 - 141	**8**	135	144
142 - 156	**9**	150	160
157 - 171	**10**	165	176
172 - 186	**11**	180	192
187 - 201	**12**	195	208
202 - 216	**13**	210	224
217 - 231	**14**	225	240
232 - 246	**15**	240	256

Note that because the default width parameter is equivalent to 15 dots, the default character size is size 0.

The ***SR*** command sets the character size for the active item, or if no group is active, for non-referenced graphics. For non-referenced graphics, once a label is drawn, its size cannot be changed. The size of a label will be the size in effect at the time the label is built. For referenced graphics, changing the size at any time will change the label size. This command has no effect if there is no active item in a selected group (that is, sending the sequence `GP1;SR10;` will have no effect). The default character size used when an item is created is size 0.

See Chapter 4 and the ***GP*** and ***IT*** commands for more on referenced graphics and groups and items.

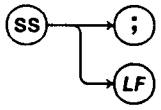
Firmware Revision dependencies:

In ROM Version 5.0, sending ***SR*** with no parameters generates an error.

SELECT STANDARD CHARACTER SET selects the standard character set as the character set to be used for all subsequent characters.

HP-GL	Yes
Link Required	Any Link

Syntax:



Response:

None

Example:

See the ***CA*** command for an example program that demonstrates the use of the ***CA, CS, SA*** and ***SS*** commands.

Comments:

The ***SS*** command causes the display to write all subsequent labels and markers for the sending window using the standard character set specified by the ***CS*** command. It will continue to use this set until the ***SA*** command is sent. The standard character set is also activated upon power-up and by the ***DF***, ***IN***, DISPLAY PRESET, SELECT INSTR, and DEVICE CLEAR commands.

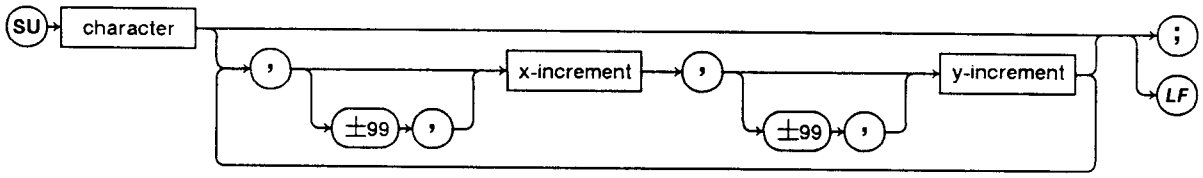
Once drawn, switching character sets will not change a label. It has to be redrawn under the new character set.

SU

SAVE USER DEFINED CHARACTER creates symbols to be added to character set 30.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
character	character to define	none	any ASCII character	-
x-increment	x direction of pen	none	-14 through 14	8
y-increment	y direction of pen	none	-14 through 14	8

Response:

None

Example:

```
1    ! Program "SU"
10     ASSIGN @Display TO 704
20     CLEAR @Display
30     OUTPUT @Display;"DE;IN;"
40     OUTPUT @Display;"SUE,99,0,5,-30,0,10,-1,-10,-1,30,0,0,5;"
50     OUTPUT @Display;"CS30;SS;PD;"
60     OUTPUT @Display;"GP1;IT1;DT@;LBE@"
70     OUTPUT @Display;"OR500,200;"
80     END
```

Line 40 saves the character symbol for summation (Sigma) in character set 30 and character position E. Line 50 designates and selects the character set 30 as the standard character set (see **CS** and ***SS***) with the pen down. Lines 60 and 70 name the summation symbol as a label and print it on the screen.

Comments:

The user defined character command provides the means to save characters of your own design into character set 30. It can be used to create symbols not included in the display's character sets, to store logos, or to create your own character fonts. The command is similar to the HP-GL command ***UC*** except the character is not drawn but is saved and character strokes *must* be specified at multiples of 45 degrees (although, due to the compressed horizontal resolution, they will not display that way; see appendix B). Each defined window may have its own unique set of user defined characters.

The character parameter denotes which position of character set 30 the character will be saved in. For example, if saved as the character E, whenever set 30 is the current set and E is sent in a label, the symbol is drawn.

The x- and y- increments should appear in pairs to define a character stroke of the pen. Character strokes must be at multiples of 45 degrees. The way this is achieved is that if one of the increments is zero, the direction and distance of the character stroke will be that of the non-zero increment. Positive x-increment parameters move the pen in the direction of labeling, that is, to the right with default label direction, and positive y-increment parameters move the pen up with default label direction. Negative parameters move the pen in the opposite direction. If both strokes are nonzero, the direction will be set by the signs of the strokes and will be either 45 degrees, 135 degrees, -45 degrees or -135 degrees. The length of the stroke will be that of the x-increment. The maximum length of a stroke is 14. No error is generated if the absolute value of a stroke length exceeds 14, it is merely set to 14. Only 15 strokes can be stored for a given character; any extra stroke pairs are ignored.

A +99 value for the x- or y- increment lowers the pen; a -99 value for the x- or y-increment raises the pen. If the x-increment is either of these, the next parameter read will be considered to be a new x-increment. If the y-increment is either of these, the next parameter read will be considered to be a new y-increment.

Upon entry into an ***SU*** command the pen is raised. Each ***SU*** command must have at least one pen down parameter in order to draw anything. An ***SU*** command without a pen down will result in a blank character. When an ***SU*** command is complete, the pen returns to its up/down status as set by ***PU*** or ***PD***. The initial x,y increment is relative to the character origin point, and each subsequent move is relative to the last commanded pen position.

Note that the ***SU*** command does *not* draw the character on the screen. To do this, create a label using character set #30 and the ***LB, CA***, and ***SA*** commands. (The above example demonstrates one method for drawing the saved character on the screen.) The

size of the drawn character will be a function of the character size in effect for that label (see ***SI*** and ***SR***).

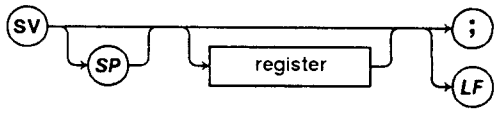
Each defined window may have its own unique character set of user defined characters (see ***SU*** command.) Thus, if the ***CA*** and ***SA*** commands are used for a specific window then the alternate character set has to have been defined for that specific window. (that is, the sequence of commands
`BW1<parameters>,SU<parameters>,BW2<parameters>` will mean that the character defined by the `SU` command *will not* appear in window two since it was defined for window one.)

Saved defined characters need not fit into a single character-space field, but the horizontal spacing between characters will remain constant and a function of the character size no matter what is in the user-defined character's strokes.

SAVE STATE saves a display state in non-volatile memory.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
register	0 = off, otherwise on	1	1-4	8

Response: None

Example:

```
1    ! Program "SV"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"BW 1,120,20,400,120,0,,,1;"
40     OUTPUT @Display;"BW 2,520,30,720,150,1,0,6,0;"
50     OUTPUT @Display;"SV3;"
60     PAUSE
70     END
```

Two windows will be seen on the screen and their definitions will be saved in register 3. The screen can then be changed and/or other functions can be dealt with in such a way that the screen has a different appearance. For example, press LOCAL, DISPLAY (DSP on the 70205A), and SELECT INSTR. When the following line is then sent, the display is directed to recall its state from save register 3:

```
OUTPUT @Display;"RC3;"
```

The screen will be restored to appear as it did with the two windows defined above.

SV

Comments:

When ***SV*** is received by the display it saves its screen state in non-volatile memory. A display's screen state includes the window sizes and all information needed to re-allocate the resources (windows, keyboard) to the addresses which had access to those resources at the time the ***SV*** command was sent or SAVE CONFIG was executed. This includes:

window sizes - P1,P2 for each of the 5 windows (4 + HP-IB).

window assignments - The address each window is assigned to.

link types - The links established for each window, including the address the keyboard is assigned to and whether any address has a control link to the display.

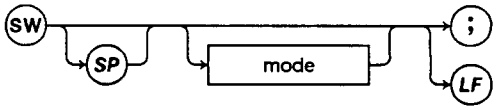
Firmware Revision dependencies:

In ROM Version 5.0, sending ***SV*** with no parameters generates an error.

SWEEP ON/OFF provides the means to turn off the sweep circuits in the display, thus blanking the picture.

HP-GL No
Link Required Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=sweep off, else normal	1	-	8

Response:

None

Example:

```
OUTPUT @Display;"SW0"
```

Turns off the sweep circuits.

Comments:

This command turns off the CRT sweep circuits in the display. This allows testing to be done in sensitive EMC environments. It is not actually intended for use in blanking the picture (see the ***DE***, ***DA***, ***PG***, and ***IN*** commands instead).

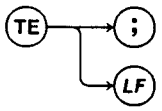
Firmware Revision dependencies: Version 6.0 and later.

TE

SELF TEST initiates the display's internal self test.

HP-GL	No
Link Required	Control Link

Syntax:



Response: None

Example:

```
OUTPUT @Display;"TE;"
```

The Display will do its self test.

Comments:

This command initiates the display's self test. The display will not respond to any further commands until the self-test is complete. At the completion of the self test, the display will set its status byte with the result of the test (see appendix C). The character set (from the final test) remains on the screen after the test. The following example HP-IB program can be used to determine the result of the test:

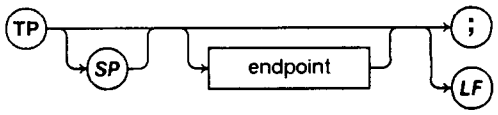
```
1    ! Program "TE"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"0E;0S;"     ! Clear errors and status byte
30     OUTPUT @Display;"TE"         ! Initiate self test
40     OUTPUT @Display;"0S"         ! Get status
50     ENTER @Display;A
70     IF BIT(A,5) THEN
80       PRINT "Display confidence test failed."
90     ELSE
100      PRINT "Display confidence test passed."
110    END IF
111  ! Clear char set off screen by entering then leaving REPORT ERRORS:
112    OUTPUT @Display;"ES0"        ! Clear char set off screen
113    OUTPUT @Display;"ES-1"       ! Restore original screen
114    OUTPUT @Display;"IM 255,32" ! Restore masks to power-up values
120    END
```

Firmware Revision dependencies: In Version 5.0, the display does not leave the character set on the screen after the test.

TRACE POINTER specifies the starting point in a graph or plot that the next graph command (***GA*** or ***GR***) or plot command (***PA*** or ***PR***) will modify.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0 = off, otherwise on	0 (first point)	-	16

Response:

None

Example:

```
1    ! Program "TP1"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"PD;"
40     OUTPUT @Display;"GP1;IT1;DX40;GA150,30,200,50,100,20;"
50     OUTPUT @Display;"GA200,150,250,80;"
60     END
```

The trace defined in line 40 will be drawn on the screen followed by the trace defined in line 50 resulting in a trace with 10 endpoints whose y-coordinates are 150, 30, 200, 50, 100, 20, 200, 150, 250, 80.

Now run the following program with the ***TP*** command between the two ***GA*** commands.

```
1    ! Program "TP2"
10     ASSIGN @Display TO 704
20     OUTPUT @Display;"DE;IN;"
30     OUTPUT @Display;"PD;"
40     OUTPUT @Display;"GP1;IT1;DX40;GA150,30,200,50,100,20;"
50     OUTPUT @Display;"TP3;"
60     OUTPUT @Display;"GA200,150,250,80;"
70     END
```

The trace defined in line 40 will be drawn on the screen and then the trace defined in line 60 will be drawn on the screen starting at the fourth point of the first trace, resulting in a trace with 7 endpoints whose y-coordinates are 150, 30, 200, 200, 150, 250, 80.

Comments:

A trace is a group of points being plotted or graphed by ***PA, PR, GA*** or ***GR*** commands. Every point in a trace may be called an "endpoint", because it is the endpoint of one or two segments which are part of the trace. The points are connected together and form one graphics item. As the command is processed, the "current endpoint" proceeds down the trace, with the first endpoint being referenced as endpoint 0 followed by endpoint 1, 2, and so on. The ***TP*** command selects which endpoint of a graph or plot is to be plotted (or modified) when the *next* ***GA*** (or ***PA***, or ***GR***, or ***PR***) command is sent. The parameter indicates the endpoint to position the refresh pointer on, meaning the next point will get graphed at that position. Thus, if TP2 is sent, this indicates that when the next graphing or plotting command comes across, the endpoint to start with will be endpoint 2 of the previously graphed trace.

The default value is 0, meaning the first point. If a point is asked for that hasn't been graphed yet, the command defaults to the next point to be graphed.

The ***TP*** command sets the trace pointer for the active item. This command has no effect if there is no active item in a selected group (that is, sending the sequence GP1;TP0; will have no effect). A ***TP*** command in non-referenced graphics will be ignored.

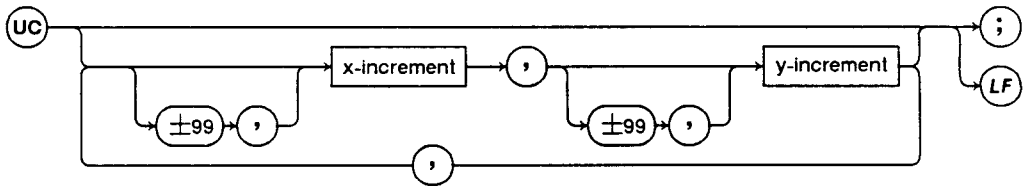
The default trace pointer when an item is created is 0.

See also the ***BA*** and ***PN*** command descriptions for more examples using ***TP***.

USER DEFINED CHARACTER draws symbols not included in the display's character sets.

HP-GL	Yes
Link Required	Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
x-increment	x direction of pen	none	-	8
y-increment	y direction of pen	none	-	8

Response:

None

Example:

```
1    ! Program "UC"
10     ASSIGN @Display TO 704
20     CLEAR @Display
30     OUTPUT @Display;"DE;IN;"
40     OUTPUT @Display;"BW1,150,100,450,300,0,0,0,1;"
50     OUTPUT @Display;"SC0,1000,0,1000;"
60     OUTPUT @Display;"PU;PA0,0,;PD;PA1000,0,1000,1000,0,1000,0,0;"
70     OUTPUT @Display;"OR500,500;"
80     OUTPUT @Display;"UC10,5,99,0,10,-20,0,10,-15,-10,-15,20,0,0,10;"
90     END
```

This program will draw a window and place the character symbol for a summation sign (Sigma) in the window.

Comments:

The ***UC*** command provides the means to draw characters of your own design. Unlike ***SU***, it draws the character as it is received and does not save it.

UC

The handling of pen up and down (using ± 99) is exactly as described for the ***SU*** command.

The x- and y- increments should appear in pairs. They specify the number of x or y units that the pen will move horizontally or vertically from the current pen position. Positive x-increment parameters move the pen in the direction of labeling, that is, to the right with default label direction, and positive y-increment parameters move the pen up with default label direction. Negative parameters move the pen in the opposite direction. The ***UC*** command is unlike the ***SU*** command in that the pen stroke is actually defined by the x- and y- increments. The stroke's length *may* be greater than 14 and its direction does *not* have to be a multiple of 45 degrees.

A ***UC*** command without a stroke list will result in a pen movement of one character-space field horizontally. When a ***UC*** command is complete, the pen returns to its up/down status as set by PU or PD. The position of the pen when the ***UC*** command is executed becomes the character origin point. The initial x,y increment is relative to the character origin point, and each subsequent move is relative to the last commanded pen position. Upon completion of the user defined character, the pen is automatically moved one character-space field to the right of the character origin point. This point becomes the current pen position and hence, the character origin point for the next character (if any).

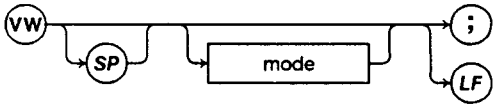
The size of the drawn character will be a function of the character size in effect when it is drawn (see ***SI*** and ***SR***). To change the size of an already drawn (referenced) user character, you must change the item size and then re-send the strokes. See Chapter 4 for more on referenced graphics and groups and items.

User defined characters need not fit into a single character-space field. Note that the user-defined character is a specific item type (see ***IT***) and hence cannot be mixed with regular labels or text in a referenced item (the display will report an **Illegal parameter** error if you try).

VIEW ON/OFF is used to blank or un-blank the current item or group.

HP-GL No
Link Required Graphics Link and a window, or Control Link

Syntax:



Item	Description	Default	Range	Bits
mode	0=group or item off, else on	1	-	8

Response:

None

Example:

```
OUTPUT @Display;"GP1;IT1;VW0;"
```

Item 1 of group 1 is blanked from the window.

```
OUTPUT @Display;"GP2;VW0;"
```

Group 2 is blanked from the window

Comments:

Each item (see ***IT*** command) or group (see ***GP*** command) can be individually blanked from the window or un-blanked by sending this command. If the window is in non-referenced mode then this command will have no effect.

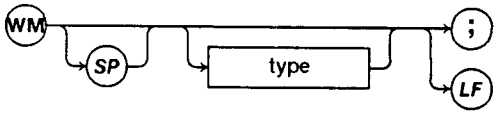
The blanking attributes of groups or item "OR" together, that is, if either an item or the group it is in are set to blank, it will be blanked.

WM

WHICH MENU loads the active menu window with the desired text.

HP-GL No
Link Required Keyboard Link or Control Link

Syntax:



Item	Description	Default	Range	Bits
type	0: menu window empty 1: MENU 2: USER	0	0-2	8

Response: None

Example:

```
OUTPUT @Display;"WM2;"
```

USER will be shown in the status block window.

Comments:

When received, the display will assign text to the status block window according to the type indicated. Note that the keyboard owner has this responsibility; the display does not do it automatically.

A

Command Cross References

List of Commands (Alphabetical by Mnemonic)

AE	Alpha Entry
AK	Assign Keyboard
AM	Address Map
AW	Assign Subwindow
AX	Axis
BA	Blank Ahead
BL	Blink On/Off
BP	Beep
BW	Build Window
CA	Designate Alternate Character Set
CI	Select Instrument
CL	Configure Label
CS	Designate Standard Character Set
CY	Copy
CZ	Output Character Size
DA	Delete All Non-Ref/Ref Objects
DE	Delete Screen
DF	Set Default Values
DI	Set Character Direction Absolute
DL	Delete
DR	Set Character Direction Relative
DS	Display Status On/Off
DT	Define Terminator
DX	Set Delta-X
EG	Output Error Message
EJ	Eject On/Off
ES	Error Screen
FC	Fill Character Line
GA	Graph Absolute
GP	Identify Group
GR	Graph Relative
GT	Graticule
HF	Holdoff On/Off
HR	Hi Res On/Off
ID	Output Identification
IL	Input Learn String
IM	Input Mask
IN	Initialize
IS	Intensity Select
IT	Identify Item
KC	Key Copy On/Off
KP	Simulate Key Pressed
KY	Send Keyboard Data
LB	Label
LT	Set Line Type
MA	Marker Attributes
MK	Marker
ML	Menu Load
MP	Set Mapping Method
OE	Output Error
OG	Output Graphics Link
OH	Output Hard Limits
OI	Output Identification
OL	Output Learn String
OO	Output Options
OP	Output P1,P2
OR	Set Origin
OS	Output Status
OV	Output Version
OY	Output Capability
PA	Plot Absolute
PD	Pen Down
PG	Page
PI	Printer/Plotter Is
PL	Plotter Limits
PN	Pan
PP	Pre-Process Mode
PR	Plot Relative
PU	Pen Up
RC	Recall State
RG	Simulate RPG Turned
RM	Remaining Memory
RP	Send RPG Data
SA	Select Alternate Character Set
SC	Scale to User Units
SI	Set Absolute Character Size
SN	Show Configuration
SP	Select Pen
SR	Set Relative Character Size
SS	Select Standard Character Set
SU	Save User Defined Character
SV	Save State
SW	Sweep On/Off
TE	Self Test
TP	Trace Pointer
UC	User Defined Character
VW	View On/Off
WM	Which Menu

List of Commands (Alphabetical by Description)

Address Map	AM	Output Identification	OI
Alpha Entry	AE	Output Learn String	OL
Assign Keyboard	AK	Output Options	OO
Assign Subwindow	AW	Output P1,P2	OP
Axis	AX	Output Status	OS
Beep	BP	Output Version	OV
Blank Ahead	BA	Page	PG
Blink On/Off	BL	Pan	PN
Build Window	BW	Pen Down	PD
Configure Label	CL	Pen Up	PU
Copy	CY	Plot Absolute	PA
Define Terminator	DT	Plot Relative	PR
Delete	DL	Plotter Limits	PL
Delete All Non-Ref/Ref Objects	DA	Pre-Process Mode	PP
Delete Screen	DE	Printer/Plotter Is	PI
Designate Alternate Character Set	CA	Recall State	RC
Designate Standard Character Set	CS	Remaining Memory	RM
Display Status On/Off	DS	Save State	SV
Eject On/Off	EJ	Save User Defined Character	SU
Error Screen	ES	Scale to User Units	SC
Fill Character Line	FC	Select Alternate Character Set	SA
Graph Absolute	GA	Select Instrument	CI
Graph Relative	GR	Select Pen	SP
Graticule	GT	Select Standard Character Set	SS
Hi Res On/Off	HR	Self Test	TE
Holdoff On/Off	HF	Send Keyboard Data	KY
Identify Group	GP	Send RPG Data	RP
Identify Item	IT	Set Absolute Character Size	SI
Initialize	IN	Set Character Direction Absolute	DI
Input Learn String	IL	Set Character Direction Relative	DR
Input Mask	IM	Set Default Values	DF
Intensity Select	IS	Set Delta-X	DX
Key Copy On/Off	KC	Set Line Type	LT
Label	LB	Set Mapping Method	MP
Marker	MK	Set Origin	OR
Marker Attributes	MA	Set Relative Character Size	SR
Menu Load	ML	Show Configuration	SN
Output Capability	OY	Simulate Key Pressed	KP
Output Character Size	CZ	Simulate RPG Turned	RG
Output Error	OE	Sweep On/Off	SW
Output Error Message	EG	Trace Pointer	TP
Output Graphics Link	OG	User Defined Character	UC
Output Hard Limits	OH	View On/Off	VW
Output Identification	ID	Which Menu	WM

Link Requirments for Commands

In the table below,if *any* of the indicated links exists, the command will be honored.

Command	Link Required			Command	Link Required		
	Graphics	Control	Keyboard		Graphics	Control	Keyboard
AE		X	X	LT	X	X	
AK		X		MA	X	X	
AM		X		MK	X	X	
AW	X	X		ML		X	X
AX	X	X		MP	X	X	
BA	X	X		OE	X	X	X
BL	X	X		OG	X	X	X
BP	X	X	X	OH	X	X	X
BW		X		OI	X	X	X
CA	X	X	X	OL		X	
CI		X		OO	X	X	X
CL	X	X		OP	X	X	
CS	X	X	X	OR	X	X	
CY		X	X	OS	X	X	X
CZ	X	X		OV	X	X	X
DA	X	X		OY	X	X	
DE		X		PA	X	X	
DF	X	X	X	PD	X	X	
DI	X	X		PG	X	X	
DL	X	X		PI		X	X
DR	X	X		PL		X	X
DS		X	X	PN	X	X	
DT	X	X	X	PP		X	
DX	X	X		PR	X	X	
EG	X	X	X	PU	X	X	
EJ		X	X	RC		X	
ES		X		RG		X	X
FC		X	X	RM	X	X	X
GA	X	X		RP		X	X
GP	X	X		SA	X	X	X
GR	X	X		SC	X	X	
GT	X	X		SI	X	X	
HF		X	X	SN		X	
HR		X	X	SP	X	X	
ID	X	X	X	SR	X	X	
IL		X		SS	X	X	X
IM	X	X	X	SU	X	X	
IN	X	X	X	SV		X	
IS	X	X		SW		X	
IT	X	X		TE		X	
KC		X	X	TP	X	X	
KP		X	X	UC	X	X	
KY		X	X	VW	X	X	
LB	X	X		WM		X	X

HP-GL Commands

BP	Beep	OO	Output Options
CA	Designate Alternate Character Set	OP	Output P1,P2
CS	Designate Standard Character Set	OS	Output Status
DF	Set Default Values	PA	Plot Absolute
DI	Set Character Direction Absolute	PD	Pen Down
DR	Set Character Direction Relative	PG	Page
DT	Define Terminator	PR	Plot Relative
IM	Input Mask	PU	Pen Up
IN	Initialize	SA	Select Alternate Character Set
LB	Label	SC	Scale to User Units
LT	Set Line Type	SI	Set Absolute Character Size
OE	Ouput Error	SP	Select Pen
OH	Output Hard Limits	SR	Set Relative Character Size
OI	Output Identification	SS	Select Standard Character Set
		UC	User Defined Character

Non-HP-GL Commands

AE	Alpha Entry	IT	Identify Item
AK	Assign Keyboard	KC	Key Copy On/Off
AM	Address Map	KP	Simulate Key Pressed
AW	Assign Subwindow	KY	Send Keyboard Data
AX	Axis	MA	Marker Attributes
BA	Blank Ahead	MK	Marker
BL	Blink On/Off	ML	Menu Load
BW	Build Window	MP	Set Mapping Method
CI	Select Instrument	OG	Output Graphics Link
CL	Configure Label	OL	Output Learn String
CY	Copy	OR	Set Origin
CZ	Output Character Size	OV	Output Version
DA	Delete All Non-Ref/Ref Objects	OY	Output Capability
DE	Delete Screen	PI	Printer/Plotter Is
DL	Delete	PL	Plotter Limits
DS	Display Status On/Off	PN	Pan
DX	Set Delta-X	PP	Pre-Process Mode
EG	Output Error Message	RC	Recall State
EJ	Eject On/Off	RG	Simulate RPG Turned
ES	Error Screen	RM	Remaining Memory
FC	Fill Character Line	RP	Send RPG Data
GA	Graph Absolute	SN	Show Configuration
GP	Identify Group	SU	Save User Defined Character
GR	Graph Relative	SV	Save State
GT	Graticule	SW	Sweep On/Off
HF	Holdoff On/Off	TE	Self Test
HR	Hi Res On/Off	TP	Trace Pointer
ID	Output Identification	VW	View On/Off
IL	Input Learn String	WM	Which Menu
IS	Intensity Select		

List of Commands by Function

Character Sets

CA	Designate Alternate Character Set
CS	Designate Standard Character Set
SA	Select Alternate Character Set
SS	Select Standard Character Set
SU	Save User Defined Character
UC	User Defined Character

Display Configuration

AK	Assign Keyboard
BW	Build Window
CI	Select Instrument
DE	Delete Screen
DF	Set Default Values
IN	Initialize
OG	Output Graphics Link
OP	Output P1,P2
RC	Recall State
SN	Show Configuration
SV	Save State

Error Handling

EG	Output Error Message
ES	Error Screen
IM	Input Mask
OE	Ouput Error
OS	Output Status

Hardcopy Output

CY	Copy
EJ	Eject On/Off
HR	Hi Res On/Off
KC	Key Copy On/Off
PI	Printer/Plotter Is
PL	Plotter Limits

Informational Displays

AM	Address Map
DS	Display Status On/Off
ES	Error Screen
OI	Output Identification
OO	Output Options
OV	Output Version
SN	Show Configuration

Labeling the Screen

CL	Configure Label
DI	Set Character Direction Absolute
DR	Set Character Direction Relative
DT	Define Terminator
LB	Label
SI	Set Absolute Character Size
SR	Set Relative Character Size

Markers

IS	Intensity Select
MA	Marker Attributes
MK	Marker

Referenced Graphics

BA	Blank Ahead
BL	Blink On/Off
DA	Delete All Non-Ref/Ref Objects
DL	Delete
GP	Identify Group
IT	Identify Item
OR	Set Origin
PN	Pan
TP	Trace Pointer
VW	View On/Off

Remotely-Controlled Display

AE	Alpha Entry
FC	Fill Character Line
KP	Simulate Key Pressed
KY	Send Keyboard Data
ML	Menu Load
PP	Pre-Process Mode
RG	Simulate RPG Turned
RP	Send RPG Data

Simple Graphics

AX	Axis
DX	Set Delta-X
GA	Graph Absolute
GR	Graph Relative
GT	Graticule
LT	Set Line Type
MP	Set Mapping Method
PA	Plot Absolute
PD	Pen Down
PR	Plot Relative
PU	Pen Up
SC	Scale to User Units
SP	Select Pen

Utility Commands

BP	Beep
HF	Holdoff On/Off
OL	Output Learn String
PG	Page
RM	Remaining Memory
SW	Sweep On/Off
TE	Self Test
WM	Which Menu

B

Screen Formats

Resolution

The 70205A and 70206A displays are mono-chromatic displays with a resolution of 1024 X 400 dots. The two displays are virtually identical from a programming point of view, the only exception being a slight difference in aspect ratios (see the ***MP*** command). Both graphics systems use *dot stretching*. Dot stretching is used to create square pixels, because given the essentially 4 x 3 display aspect ratio, and the 8 x 3 graphics aspect ratio, each pixel is about twice as high as it is wide. Therefore, to achieve square pixels, provide for smoother vertical transitions, and equalize the brightness of vertical and horizontal lines, all dots are stretched to be two pixels wide. The figure below shows how this helps make traces overlap and smooth the transitions.

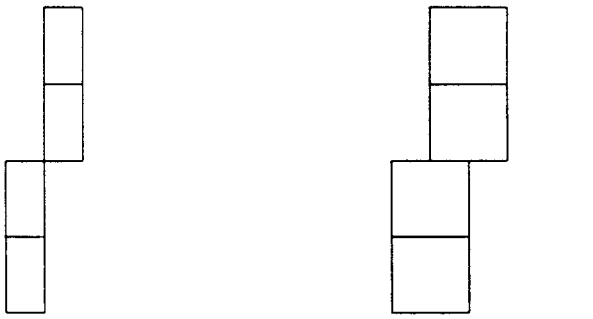
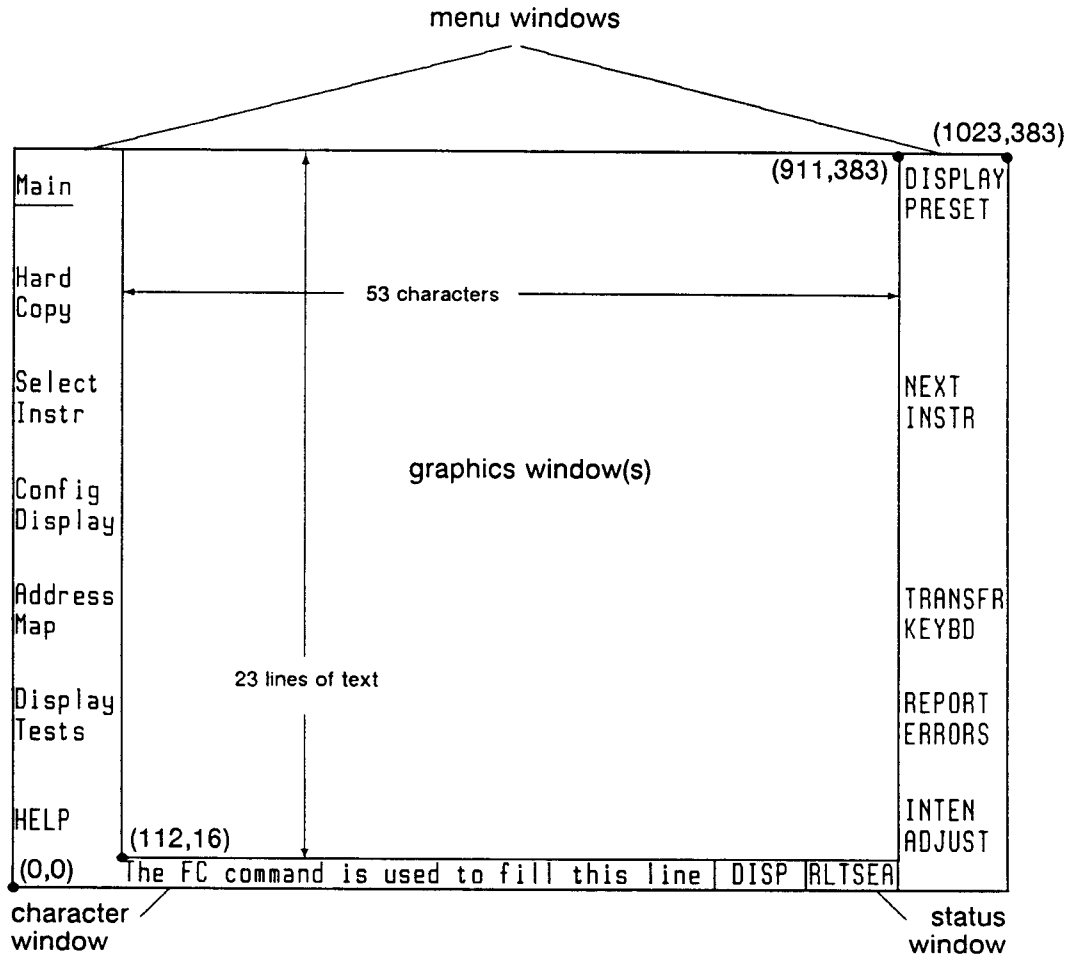


Figure B.1: Effect of Dot Stretching

Other side effects of dot stretching relate to the appearance of certain line types. For example, line type 7 must be used to draw dotted horizontal lines, because if line type 8 is used, the dots will stretch together to show what appears to be a solid line (see the ***LT*** command). This is not a problem when drawing dotted lines in graticules, because the ***GT*** command compensates for the compressed horizontal resolution and widens the spacing of the dots in the horizontal lines.

Screen Layout

The 70205A and 70206A have identical screen layouts, as shown in the figure below:



The ***menu windows*** are used to display either the display's menus or the menus of the module which owns the ***keyboard link***. The ***graphics window***, which can be subdivided into four other arbitrary windows, is used for the display of graphics and text by modules which own ***graphics links*** (actually, the entire screen may be used for this purpose, but the area shown as the ***graphics window***, above, is the recommended area for graphics). The graphics window, as shown, will hold 53 characters across (of the smallest size characters). The ***character window*** (or "character line") is simply one

line of text (40 characters long) which can be displayed by the module with the ***keyboard link*** using the ***FC*** indexFC command. It is useful for giving prompts and other human interface information to the user. The ***DS*** command can be used to turn on and off a box which surrounds the character window. The ***status window*** is for the use of the display, to communicate system status and messages to the user (in fact, the status window overlays and sometimes overrides the character window, in the 70205A and 70206A). The ***DS*** command can be used to turn it on or off.

Part of the status window is the ***annunciator block*** which contains the letters RLTSEA, not all of which are always on. These letters stand for:

R	HP-IB Remote
L	HP-IB Listen
T	HP-IB Talk
S	HP-IB SRQ, on when the display is pulling on SRQ
E	Error, on when there is an error in *any* Row 0 module. Also, blinks if a fault is detected in the HP-MSIB interface at power-up.
A	Active, on when the Display owns the keyboard or when it is being selected in a configuration function.

C

Errors, Masks, and Status Bytes

The ***IM*** (input mask) command specifies the conditions under which an error message or a status message will occur for the I/O channel which sent the message. See the ***IM*** command description for syntax. The masks are defined as follows:

Error Mask

The error mask specifies the conditions that will cause an error to be reported by the display[1]. Its value is the sum of any combination of the bit values shown in the table below. When an error occurs, the bit in the error mask corresponding to the error is tested to determine if the error is to be reported. If a bit is not set, the error is not reported, and there is no way to ever determine if that error occurred.

error mask bit	bit value	associated error message
0	1	2001,Illegal command
1	2	2006,Parm out of range
2	4	2002,Illegal parameter
3	8	2011,Memory overflow
4	16	2005,Illegal character set
5	32	2007,Missing terminator
6	64	2009,Protocol error
7	128	6008,Confidence test failed

The error mask for a channel will be reset to the default whenever that channel's window is initialized. For HP-IB, this occurs at power-up, DISPLAY PRESET, SELECT INSTR, DEVICE CLEAR, or when an ***IN*** command is sent.

A description of each of the errors follows.

2001,Illegal command The display has been sent a command it does not recognize (for example, `OUTPUT @Display;"XX"` would generate this error).

2006,Parm out of range A parameter that violates the range specification for a given command has been sent.

[1]The error mask does not work properly in Rom Version 5.0

2002,Illegal parameter An item has been sent a command that does not match its type (for example, sending ***LB*** to a ***PA*** type item), a ***CL*** command has been sent in group 0, or a bad learnstring has been sent.

2011,Memory overflow An attempt has been made to allocate more vector list memory than the display contains. Usually this means that the user is trying to display more traces than the display's memory can support, or that too many strokes have been sent in non-referenced graphics.

2005,Illegal character set An attempt has been made to specify a character set (using ***CA*** or ***CS***) that is not available in the display.

2007,Missing terminator A command has been sent to the display without a valid terminator (see the **Syntax** section in Chapter 5).

2009,Protocol error A command has been sent to the display that requires a link type that is not currently established between the sender and the display.

6008,Confidence test failed A self test (probably invoked with the ***TE*** command) has failed.

Note that there are a set of *hardware* errors that can generate error messages on the display; these errors are not reported through the status byte or over the bus at all. If the display's E light will not go out, it may because one of the hardware errors is present. In that case, SRQ will not be asserted even if the error mask bit is on. Consult your service manual in that case.

Status Mask

The status mask specifies the conditions that will send a status message across that channel. For HP-IB, this means asserting SRQ, after which the status byte can be read via serial poll. For HP-MSIB, it means sending the ***STATUS*** HP-MSIB command.

The bits in the status byte are associated with the conditions described below. When one of these conditions occurs, the bit in the status byte will be set regardless of the state of the status mask and for HP-IB[2], SRQ will be asserted if the mask bit for that condition is set, or for HP-MSIB, the status message will be sent if the mask bit for that condition is set. The status mask value specified is the sum of any combination of the bit values shown in the table below.

[2]In ROM Version 5.0, the status mask and SRQ do not work properly

status mask bit	bit value	set when:	cleared when:
0	1	Pen down	Pen up
1	2	IP key pressed	***KY*** is performed
2	4	knob count available	***RP*** is performed
3	8	Window initialized	***OS*** is performed
4	16	Hardcopy dump finished	Hardcopy dump started
		CI finished	***CI*** started
5	32	Error occurs in display	***OE*** or ***EG*** is done
6	64	-	-
7	128	Key is pressed	***KY*** is performed

Bit 6, above, is reserved by HP-IB as the "Require Service" bit. It is set, in the HP-IB SRQ status byte only, when SRQ is asserted, and cleared when a Serial Poll is performed or SRQ is un-asserted. It is not maskable.

For HP-IB, a serial poll clears SRQ, but not the status byte. Conversely, if a bit in the status byte is cleared as shown in the table above, SRQ remains asserted until a serial poll is performed. Also, the status byte is cleared completely when an ***OS*** is received, but again, if SRQ is being asserted, it will remain so until a serial poll is performed.

The default status mask value of 32 specifies that if an error occurs then a service request will be sent and bit 6 of the status byte will be set. The status mask for a channel will be reset to the default whenever that channel's window is initialized. For HP-IB, this occurs at power-up, DISPLAY PRESET, SELECT INSTR, DEVICE CLEAR, or when an ***IN*** command is sent. When this occurs the status byte is cleared and SRQ is un-asserted.

Parallel Mask

The parallel mask is ignored but is included for HP-GL compatibility.

Clearing Errors

Errors are cleared when a window is re-initialized (***IN***), when ***OE*** or ***OG*** is sent, or when a REPORT ERRORS is performed.

Firmware Revision dependencies:

See the ***IM*** command description.

D

Character Sets

The **CS** command is used to designate the character set that is to be used as the standard character set and the **CA** command is used to designate which character set is to be used as the alternate character set. The controller or instrument selects between the standard and the alternate character set by sending either the **SS** command or the **SA** command. Both the standard and alternate set are Set 0 (see page 191), and the standard character set is selected, when default conditions exist (power-up, **DF**, **IN**, **CI**, SELECT INSTR, DEVICE CLEAR).

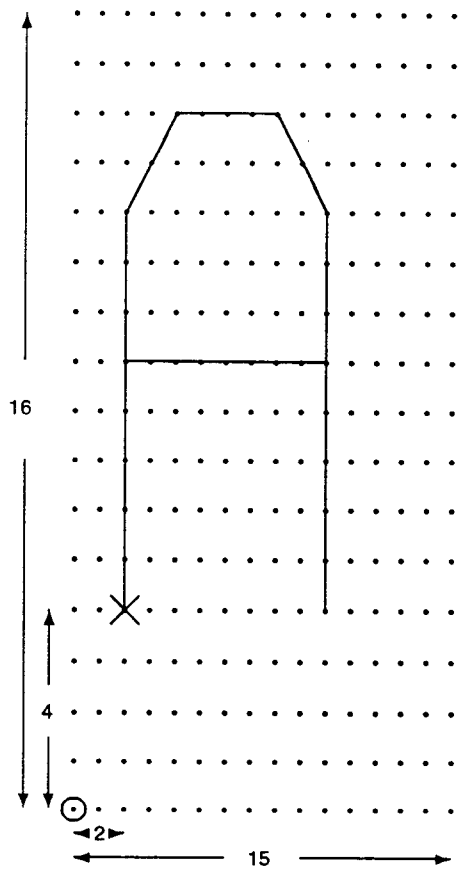
The basic character cell for size 0 characters is 15 by 16 dots, as shown on page 190. The origin of the character is circled - it is the lower left dot in the cell. It is that point in the cell which will get placed at the origin of a label character. The base point of the character itself is up 4 points and over 2 points from the origin, marked by an X in the figure below. This means that to place a *character* origin at a specific point, you must place the *cell* 4 points down and 2 points to the left. This scales as the size increases; for example, instead of 2 and 4, for size 2 characters it is 6 and 12. In general, the offset of the character from the cell origin is:

$$x_{offset} = (size + 1) \times 2, y_{offset} = (size + 1) \times 4$$

The character sets available are listed as follows:

number	Character set
0	US ASCII with MMS extensions (see below)
30	Saved User defined Character Set (See **SU** command)

See the **SI** command description for more on character sizes, and the **CA** command for more on character sets.



0 = character cell origin

X = character origin

Basic Character Cell (Size 0)

Characters are placed using the character cell origin.

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
0	NU	32		64	@	96	`	128	0	160	ÿ	192	ϕ	224	É
1	SH	33	!	65	A	97	a	129	1	161	α	193	ϕ	225	Í
2	SX	34	"	66	B	98	b	130	2	162	β	194	ϕ	226	Ó
3	EX	35	#	67	C	99	c	131	3	163	γ	195	Δ	227	Ú
4	ET	36	$	68	D	100	d	132	4	164	δ	196	∇	228	á
5	EQ	37	%	69	E	101	e	133	5	165	ϵ	197	∠	229	é
6	AK	38	&	70	F	102	f	134	6	166	ζ	198	Ξ	230	í
7	BL	39	'	71	G	103	g	135	7	167	η	199	Υ	231	ó
8	BS	40	(	72	H	104	h	136	8	168	θ	200	Γ	232	ú
9	HT	41	)	73	I	105	i	137	9	169	ι	201	Π	233	À
10	LF	42	*	74	J	106	j	138	:	170	κ	202	Ψ	234	È
11	VT	43	+	75	K	107	k	139	;	171	λ	203	Φ	235	Ì
12	FF	44	,	76	L	108	l	140	<	172	μ	204	∫	236	Ò
13	CR	45	-	77	M	109	m	141	=	173	ν	205	∫	237	Ù
14	SO	46	.	78	N	110	n	142	>	174	ξ	206	£	238	à
15	SI	47	/	79	O	111	o	143	?	175	o	207	ℍ	239	è
16	DL	48	0	80	P	112	p	144	@	176	π	208	ℝ	240	ì
17	D1	49	1	81	Q	113	q	145	A	177	ρ	209	Ç	241	ò
18	D2	50	2	82	R	114	r	146	B	178	σ	210	ç	242	ù
19	D3	51	3	83	S	115	s	147	C	179	τ	211	Ñ	243	â
20	D4	52	4	84	T	116	t	148	D	180	υ	212	ñ	244	ê
21	NK	53	5	85	U	117	u	149	Ä	181	ϕ	213	ℚ	245	î
22	SY	54	6	86	V	118	v	150	Ë	182	χ	214	¢	246	ô
23	EB	55	7	87	W	119	w	151	Ï	183	ψ	215	√	247	û
24	CN	56	8	88	X	120	x	152	Ö	184	ω	216	Σ	248	â
25	EM	57	9	89	Y	121	y	153	Ü	185	Ω	217	≃	249	ê
26	SB	58	:	90	Z	122	z	154	ä	186	ℵ	218	∞	250	î
27	EC	59	;	91	[	123	{	155	ë	187	ℵ	219	±	251	ô
28	FS	60	<	92	\	124	|	156	ï	188	⟨	220	÷	252	û
29	GS	61	=	93	]	125	}	157	ö	189	=	221	≠	253	∴
30	RS	62	>	94	^	126	~	158	ü	190	⟩	222	¿	254	∵
31	US	63	?	95	_	127	▮	159	Ÿ	191	↑	223	Á	255	◊

Character Set 0

E

Scaling and Origins

In order to demonstrate how the **SC** command interacts with other graphics commands, execute the following program in which two windows are built on the screen: window 1, which takes up the entire screen, and window 2, a small window in the middle of the screen (see program SC1, below, for the entire program):

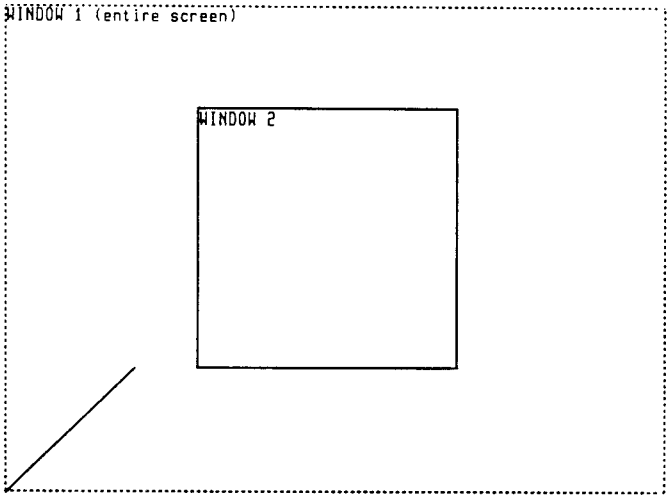
```
10    ASSIGN @Display TO 704
20    OUTPUT @Display;"DE;BW1,0,0,1023,383,1,,,0"   ! window 1, whole screen
30    OUTPUT @Display;"BW2,300,100,700,300,0,,,1"   ! window 2, with frame
40    OUTPUT @Display;"DT@;PA 10,360;PD;LBWINDOW 1 (entire screen)@"
50    OUTPUT @Display;"PU;PA 310,280;PD;LBWINDOW 2@"
100   END
```

The keyboard is assigned to window 2, thus giving it a solid frame, and window 2 is assigned to HP-IB. This means that *the window to which we are drawing is Window 2.* You might think that means that our graphics will appear there, but read on. Window 1 is given a dotted frame (see the **BW** command description to understand the assigning and framing of windows).

Now, add a line to the above program so that it draws a line on the screen (line 70, below):

```
10    ASSIGN @Display TO 704
20    OUTPUT @Display;"DE;BW1,0,0,1023,383,1,,,0"   ! window 1, whole screen
30    OUTPUT @Display;"BW2,300,100,700,300,0,,,1"   ! window 2, with frame
40    OUTPUT @Display;"DT@;PA 10,360;PD;LBWINDOW 1 (entire screen)@"
50    OUTPUT @Display;"PU;PA 310,280;PD;LBWINDOW 2@"
70    OUTPUT @Display;"GP1;IT1;PA0,0,200,100"       ! draw a line
100   END
```

The screen will look like the figure below:

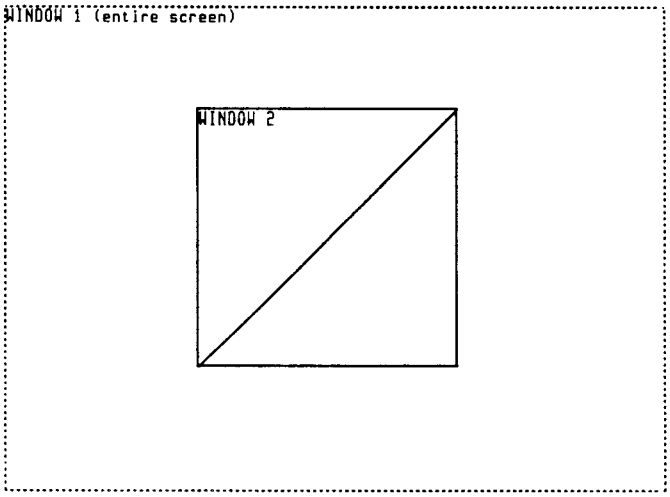


Note that the line appears in the lower left corner of the screen, outside window 2, even though we are drawing in window 2!

Now add line 60, below, to scale the window before drawing in it:

```
1     ! Program "SC1"
10      ASSIGN @Display TO 704
20      OUTPUT @Display;"DE;BW1,0,0,1023,383,1,,,0"   ! window 1, whole screen
30      OUTPUT @Display;"BW2,300,100,700,300,0,,,1"   ! window 2, with frame
40      OUTPUT @Display;"DT@;PA 10,360;PD;LBWINDOW 1 (entire screen)@"
50      OUTPUT @Display;"PU;PA 310,280;PD;LBWINDOW 2@"
60      OUTPUT @Display;"SC0,200,0,100;"
70      OUTPUT @Display;"GP1;IT1;PA0,0,200,100"
100     END
```

The screen will now look like the figure below:



The line now goes across window 2 from corner to corner. This is because window 2 is now scaled in user units, and the origin is set relative to the window, whereas before we scaled it, we were plotting in display units, relative to the whole screen. So, even though you may be drawing in a window, you first need to scale the window to relate the graphics to *it*, rather than to the full screen.

Now we come to some of the more difficult to anticipate aspects of scaling and how they interact with other graphics commands. To understand the following explanation, you need to know that the display maintains a list in its memory of all the lines and characters on its screen at any time. This list is called the *vector list.* What an object looks like and where it is are determined by the data stored in the vector list. The data is stored there in *display units* so that it can be copied directly from the vector list onto the screen. That means that for every object on the screen, both its origin and stroke data are stored in the vector list in display units, whether the object is scaled or not. Remember that until an **SC** command is sent, users units *are* display units.

Now the user need only remember three key points to understand how scaling, origins and graphics work:

- An item's origin and its stroke data are stored separately in the vector list.
- Once the stroke data for an item is sent, it is scaled and goes into the vector list and remains unchanged until it is sent again.
- Once the origin data for an item is sent, it is scaled and goes into the vector list and remains unchanged until it is sent again. It is initialized when a group or item is created, to (0,0) in user units.

The important thing to understand is when the stroke (line, trace) data is updated, and when the origin data is updated. The former will determine the precise size and appearance of the trace; the latter will determine where the trace appears on the screen. And *neither one is updated* when an **SC** command is sent. This means that the trace data can be sent and the scale factor then changed, and it will not affect the appearance of the trace at all. Similarly, once the origin has been set, changing the scale factor will not affect the origin; the trace will not move on the screen. In both cases, re-sending the same trace and origin data after the scale factor changes will cause the effect that you might have expected to see simply by changing the scale factors.

One effect of this is that the order in which commands are sent can heavily impact the data on the screen. For example,

```
IN;GP1;IT1;SC-200,200,-100,100;PA 0,0,200,100;
```

is *not* equivalent to

```
IN;SC-200,200,-100,100;GP1;IT1;PA 0,0,200,100;
```

even though the only difference is *when* the `GP1;IT1;` was sent. In the first case, the group origin was initialized to (0,0) in user units (which means in display units, since **SC** had not yet been sent) when the group was built, and since no **OR** command was sent to change it, the group origin remains at (0,0) in display units, or the lower left corner of the screen. In the second case, the scale factor is all set up by the time the group is created, so the group origin is placed at (0,0) in *user units*, which in this case is right in the center of the window.

This all boils down to the following general principle:

RULE

Send the **SC** command *before* you define or send data to any group or item to which you wish that **SC** command to apply.

Here are more details which may aid in gaining a thorough understanding of these processes:

When the display receives an **SC** command, it modifies two sets of window variables: the *scale factor* and *offset* of the window. For both x and y, the *scale factor* is the amount by which to multiply an incoming graphics coordinate, and the *offset* is the amount (in display units) to then add to the scaled coordinate, before putting it in the vector list and, hence, on the screen. Remember that all data in the vector list is stored in actual, screen coordinates (display units). When a window is initialized, the offset is set to 0 and the scale factor to 1. Then, when a group is created or a group origin modified, the new origin is converted to display units using the current *scale factor* then added to the current *offset* and stored as the group's origin. Thus, it is only when a *group origin* is modified (or a group created) that the current value of *offset* for the window is applied to the group. Similarly, it is only when new data is sent to the item that the current value of *scale factor* is applied to the data. That is why nothing on the screen changes simply because a new **SC** command is sent.

NOTE

Remember that the **SC** command modifies *window* variables; the scale factor and offset are window variables, not associated with each group and item. Objects are drawn according to the current scale factors and offsets *for the window.*

In order to demonstrate these subtleties, execute the following program in which the axis described in the example for the ***AX*** command is drawn on the screen of the display. In this program, window 1 has a dotted border and window 2 a solid border, because window 2 has the keyboard assigned. Window 2 is assigned to HP-IB, which means that is where we are doing our graphics.

```
1    ! Program "SC2"
10     ASSIGN @Display TO 704
20     REMOTE 7                        ! Exit DISPLAY menus to see borders
30     OUTPUT @Display;"DE;"           ! Clean the screen
40     OUTPUT @Display;"BW1,120,20,400,200,1,,,0"  ! no keyboard
50     OUTPUT @Display;"BW2,120,200,520,300,0,,,1" ! keyboard, HP-IB
70     OUTPUT @Display;"SC250,750,75,325;"
80     OUTPUT @Display;"GP1;"
90     OUTPUT @Display;"OR500,200;"
100    OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
110    END
```

Now let us picture the important variables for a group, item, and window as the following data structure for window 2, group 1, and item 1, shown as it looks after executing line 50 in the above program:

WINDOW 2

window parameters, in display units (dots), from ***BW***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from *SC*					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
X scale factor		Y scale factor		X offset	Y offset
1		1		0	0

GROUP 1

X origin	Y origin
-	-

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

The graphics parameters are the same as the window parameters because user units are the same as display units at this point: the **SC** command has not yet been sent. No groups or items exist and no data has been sent. Now send the **SC** command as in line 70, `OUTPUT @Display;"SC250,750,75,325;"`. This sets the user window width to 500 and the height to 250. The scale factors are

$$window\ width/user\ width$$

or 400/500 for x and 100/250 for y, which are 0.8 and 0.4, respectively. The offsets are

$$window\ P1 - (user\ P1 \times scale\ factor),$$

which for x is $120 - (250 \times 0.8) = -80$ and for y is $200 - (75 \times 0.4) = 170$. The data structures become:

WINDOW 2

window parameters, in display units (dots), from **BW**					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from **SC**					
P1X	P1Y	P2X	P2Y	width	height
250	75	750	325	500	250
X scale factor		Y scale factor		X offset	Y offset
.8		.4		-80	170

GROUP 1

X origin	Y origin
-	-

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

Now send line 80, `OUTPUT @Display;"GP1;"`. This will establish the group 1 origin. The group origin is initialized to (0,0) in user units, which must be converted to display units before storing, by multiplying by the scale factor and adding the offsets (which means just adding the offsets, because the value starts at 0). So the group origin is initialized to the window offset:

WINDOW 2

window parameters, in display units (dots), from ***BW***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from ***SC***					
P1X	P1Y	P2X	P2Y	width	height
250	75	750	325	500	250
X scale factor		Y scale factor		X offset	Y offset
.8		.4		-80	170

GROUP 1

X origin	Y origin
-80	170

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

This *initial* group origin[1] can be checked by determining where on the screen the corner points of the window[2] would be placed if sent in a graphics command. To do this, you need to know that when the data is plotted to the CRT by the stroke-to-dot converter, *the group origin, item origin, and data are all added together* to determine the point to plot to. Hence, since there *is* no item origin yet, the window corner points would be graphed at

$$(user\ unit\ value \times scale\ factor) + group\ origin$$

which, for P1X will be $250 \times .8 - 80 = 120$, for P1Y will be $75 \times .4 + 170 = 200$, for P2X will be $750 \times .8 - 80 = 520$, and for P2Y will be $325 \times .4 + 170 = 300$. Thus, the corner points in display units will be (120,200) and (520,300), which *are* the corner points of the window as set by the ***BW*** command.

[1](0,0) in user units, (-80,170) in display units

[2](250,75) and (750,325) in user units

Now line 90, `OUTPUT @Display;"OR500,200;"` is sent. This causes the group origin to be modified. First, each coordinate is "un-scaled", that is, converted to display units, by multiplying by the scale factor. This converts (500,200) to (400,80). Now the offsets are added to yield (320,250), and this value is stored as the new group origin. These coordinates are the coordinates, in display units, of the actual point on the screen that will be the origin for group 1. It is in the exact center of the window, whether viewed from the standpoint of display units or user units. Hence the following note:

NOTE

The current window offsets are stored as part of each group's *origin* when that origin is set or changed.

That is the only way the window offset is applied to the graphics, by becoming part of the group origin. The variables now look like:

WINDOW 2

window parameters, in display units (dots), from *BW*					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from *SC*					
P1X	P1Y	P2X	P2Y	width	height
250	75	750	325	500	250
X scale factor		Y scale factor		X offset	Y offset
.8		.4		-80	170

GROUP 1

X origin	Y origin
320	250

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

Finally, we send line 100,

```
OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
```

This creates item 1, sets its origin, and draws the axes. The item origin is set by *scaling* the origin parameters (since they are in user units) but without adding any *offsets* since the offset is already added to the *group* origin. Here, the item is being created, so the default origin of (0,0) is scaled and stored, but scaling 0 yields 0, so (0,0) is stored for the item origin. Similarly, as the trace (axis) data comes in, it is scaled but no offsets are added. Remember that when the data is plotted to the CRT, the group origin, item origin, and data are all added together to determine the point to plot to. In this case,

the axes go right in the center of window 2 (axes are always placed such that their intersection is at the item origin). The final appearance of the variables is as shown below:

WINDOW 2

window parameters, in display units (dots), from **BW**					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from **SC**					
P1X	P1Y	P2X	P2Y	width	height
250	75	750	325	500	250
X scale factor		Y scale factor		X offset	Y offset
.8		.4		-80	170

GROUP 1

X origin	Y origin
320	250

ITEM 1

X origin	Y origin	stroke data
0	0	axis strokes

Now, what if we re-send the scale factor, to change it? Suppose we send a scale factor `SC 250,1250,75,575`. This will change the scale factors and offsets, but not the group or item origins or data (until they are re-sent). Hence the screen remains unchanged. The variables become:

WINDOW 2

window parameters, in display units (dots), from **BW**					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from **SC**					
P1X	P1Y	P2X	P2Y	width	height
250	75	1250	575	1000	500
X scale factor		Y scale factor		X offset	Y offset
.4		.2		20	185

GROUP 1

X origin	Y origin
320	250

ITEM 1

X origin	Y origin	stroke data
0	0	axis strokes

Note that there has been no change to the group or item data, which is the data that actually gets copied to the screen. If the group origin is set and data is sent, the result will reflect the new scale factors and offsets, but until then, the data on the screen remains unchanged. Furthermore, if new data is sent without changing or re-sending the origin, the size and shape of the trace will reflect the new scale factors but the trace won't move! This is non-intuitive and can cause confusion. Let us look at the variables again to understand it. Suppose at this point we send the same axis command as before, in other words, the identical data, now that we have re-scaled the window:

```
OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
```

Item 1 already exists, so the origin need not be created, and no origin data is sent, so the stored item origin remains unchanged. Likewise the group origin, and hence the group offsets, are unchanged. The new window offset is not accessed, as that only happens when a group origin is modified.

The axis data is scaled according to the new scale factors, which just happen to be half their previous value, so the axis is re-drawn at half its previous size. Because the group offsets haven't changed, the axis is drawn right where it was before - in the exact center of the window - even though according to the new **SC** data, that point is no longer (500,200), the point we had previously sent as the group origin. Except for the fact that the axis strokes have been updated, the variables remain unchanged:

WINDOW 2

window parameters, in display units (dots), from *BW*					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from *SC*					
P1X	P1Y	P2X	P2Y	width	height
250	75	1250	575	1000	500
X scale factor		Y scale factor		X offset	Y offset
.4		.2		20	185

GROUP 1

X origin	Y origin
320	250

ITEM 1

X origin	Y origin	stroke data
0	0	axis strokes, updated

Now we re-send the identical origin command,

```
OUTPUT @Display;"GP1;OR500,200;"
```

The axes will (finally) move to the point that now represents (500,200) (which is probably what the programmer expected to happen when the new **SC** values were sent). This puts the axes in the lower left corner of the window but does not update the axis stroke data (so the axis size and shape remain unchanged). The variables will look like:

WINDOW 2

window parameters, in display units (dots), from *BW*					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from *SC*					
P1X	P1Y	P2X	P2Y	width	height
250	75	1250	575	1000	500
X scale factor		Y scale factor		X offset	Y offset
.4		.2		20	185

GROUP 1

X origin	Y origin
220	225

ITEM 1

X origin	Y origin	stroke data
0	0	axis strokes

But let's say we really want the axes up in the center of the window, where they were before. Let's also say we want to leave the group origin alone. We can do this by setting the *item* origin. Heretofore it has simply been (0,0), the default, but if we set it as `OUTPUT @Display;"IT1;OR 250,125;"`, the axes will move to the center of the screen. The item origin in display units will simply be that in user units times the scale factors, or $(250 \times .4, 125 \times .2) = (100, 25)$, and now the variables will be:

WINDOW 2

window parameters, in display units (dots), from *BW*					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from *SC*					
P1X	P1Y	P2X	P2Y	width	height
250	75	1250	575	1000	500
X scale factor		Y scale factor		X offset	Y offset
.4		.2		20	185

GROUP 1

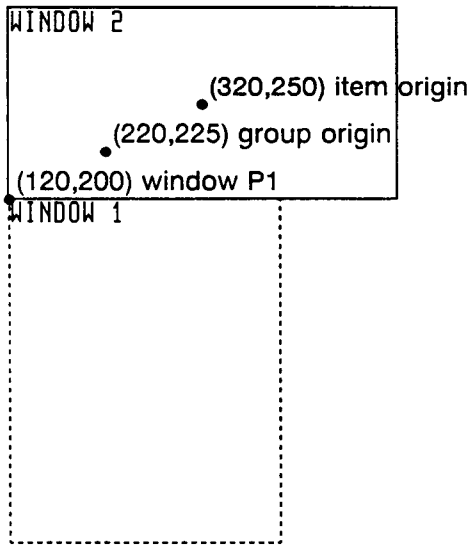
X origin	Y origin
220	225

ITEM 1

X origin	Y origin	stroke data
100	25	axis strokes

The data in the Group and Item boxes, above, is what actually gets drawn to the screen. Note that adding up the group and item origins in display units, we derive the point (320,250), which is the center of the window.

The relationship between the window and the group and item origins is shown in the figure below:



Here is another example, to illustrate the order dependence of the various parameters. Execute the following program, which is similar to the above program but has a very different result:

```
1    ! Program "SC3"
10     ASSIGN @Display TO 704
20     REMOTE 7                      ! Exit DISPLAY menus to see borders
30     OUTPUT @Display;"DE;"         ! Clean the screen
40     OUTPUT @Display;"BW1,120,20,400,200,1,,,0"   ! no keyboard
50     OUTPUT @Display;"BW2,120,200,520,300,0,,,1" ! HP-IB, keyboard
65     OUTPUT @Display;"GP1;"
66     OUTPUT @Display;"OR500,200;"
100    OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
110    END
```

After running this program, the axes are drawn in the middle of the *screen*, rather than within window 2. The only difference between these programs is that the **SC** command is left out. Thus, the graphics end up being done in display units. Hence the window boundaries are essentially ignored and the whole screen is used.

From the variable structure standpoint, the variables look as they did in the previous example after window 2 is built (line 50):

WINDOW 2

window parameters, in display units (dots), from ***BW***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from ***SC***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
X scale factor		Y scale factor		X offset	Y offset
1		1		0	0

GROUP 1

X origin	Y origin
-	-

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

Then when line 65 is sent, group 1 gets defined using display units instead of user units. The group origin that gets stored is then simply (0,0) as the scale factors are still 1 and the offsets 0. Then when line 66 is sent, `OR500,200;` the origin that gets stored is exactly as sent, (500,200):

WINDOW 2

window parameters, in display units (dots), from ***BW***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
graphics parameters, in user units, from ***SC***					
P1X	P1Y	P2X	P2Y	width	height
120	200	520	300	400	100
X scale factor		Y scale factor		X offset	Y offset
1		1		0	0

GROUP 1

X origin	Y origin
500	200

ITEM 1

X origin	Y origin	stroke data
-	-	(no data)

Thus, when the ***AX*** command finally gets sent, the scale factors are 1 and the origin is at (500,200), so it gets plotted without any scaling, in the middle of the screen, rather than in window 2. This may seem odd if the user was expecting it to go into the window to which it was sent; hence the need to understand the plotting process.

In the following example, the scale factor is sent, but only *after* the group origin is defined. This results in axes of the correct size, but again plotted outside the window. The size is right because the scale factors had been set before the *data* was sent, but not before the *origin* was set up:

```
1    ! Program "SC4"
10     ASSIGN @Display TO 704
20     REMOTE 7                        ! Exit DISPLAY menus to see borders
30     OUTPUT @Display;"DE;"           ! Clean the screen
40     OUTPUT @Display;"BW1,120,20,400,200,1,,,0"   ! no keyboard
50     OUTPUT @Display;"BW2,120,200,520,300,0,,,1" ! HP-IB, keyboard
65     OUTPUT @Display;"GP1;"
66     OUTPUT @Display;"OR500,200;"
70     OUTPUT @Display;"SC250,750,75,325;"
100    OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
110    END
```

Finally, in the example below, the axes are built in the other window, window 1. Try to predict where they will go before you run the program:

```
1    ! Program "SC5"
10     ASSIGN @Display TO 704
20     REMOTE 7                        ! Exit DISPLAY menus to see borders
30     OUTPUT @Display;"DE;"           ! Clean the screen
40     OUTPUT @Display;"BW1,120,20,400,200,0,,,1"   ! HP-IB, keyboard
50     OUTPUT @Display;"BW2,120,200,520,300,1,,,0" ! no keyboard
70     OUTPUT @Display;"SC250,750,75,325;"
80     OUTPUT @Display;"GP1;"
90     OUTPUT @Display;"OR500,200;"
100    OUTPUT @Display;"IT1;AX500,250,250,125,20,10,10,5,20;"
110    END
```

F

Special Characters

The character sequences listed in the following tables cause special actions when sent in ***LB***, ***FC***, or ***ML*** commands.

CHR$(7)	beeps.
CHR$(8)	moves cursor back one character.
CHR$(10)	***LF*** Moves cursor down one line. If in an ***ML*** command, ***LF*** is turned into ***CR LF*** before it is executed (see ***CR***, below). If on the last line of a label, the label will roll up one line. Ends active enhancement except in ***ML***. Should not be used in single line labels or ***FC***, except as part of ***CR LF***.
CHR$(11)	moves cursor up one line. If on top line of a referenced label, cursor will move to bottom line. Ends active enhancement.
CHR$(12)	***FF*** Homes cursor and clears label.
CHR$(13)	***CR*** Moves cursor to start of line.
CHR $(27)	ESC (see below)

After an ESC, the next character may cause an action, as described in the following table:

A	same as CHR$(11) above
B	moves cursor down one line. If on bottom line of a referenced label, cursor will move to top line.
C	moves cursor right one. If at end of label, goes to first char.
G	same as CR above.
H	homes cursor (upper left of label/screen).
J	clear to end of screen/label
K	clear to end of line
S	scroll up one line
T	scroll down one line
Y	Turn on DISPLAY FUNCTIONS
Z	Turn off DISPLAY FUNCTIONS
&	Initiates other functions (see below)

ESC & functions

Enhancements:

Ec&dA	blinking
Ec&dB	inverse
Ec&dC	inverse+blinking
Ec&dD	underline
Ec&dE	underline+blinking
Ec&dF	underline+inverse
Ec&dG	underline+inverse+blinking
Ec&d@	enhancements off

The above enhancements are *not* available for ***LB*** commands sent in non-referenced graphics (GP0; or IT0;). If sent, the escape sequence is simply printed to the screen.

Cursor addressing:

Ec&a*C	Absolute column addressing
Ec&a*R	Absolute row addressing
Ec&a*-C	Relative column addressing
Ec&a*-R	Relative row addressing
Ec&a*+C	Relative column addressing
Ec&a*+R	Relative row addressing

(Where * is an integer specifying a row or column, as Ec&a10R)

As shown above, the cursor can be positioned to any point within the label using absolute or relative co-ordinates. These sequences can also be combined, with the first (row or column) command being lower case and the second (column or row) upper case, as:

Ec&a*r*C row first, then column.

Ec&a*c*R column first, then row.

When using these coordinates, the top line in the label is Row 0, and the leftmost character is Column 0.

DISPLAY FUNCTIONS

As shown above, EcY invokes DISPLAY FUNCTIONS and EcZ cancels it. DISPLAY FUNCTIONS is characteristic of each individual label item; it may be on for some and off for others at any given time. It is off when the label is created. In Group 0, it is *always* on, and EcY and EcZ have no effect. When DISPLAY FUNCTIONS is on, control characters that would otherwise cause special actions are instead printed in the label.

With DISPLAY FUNCTIONS on, the ***CR*** character is printed to the screen, and then a ***CR LF*** is performed (see above), which is *not* printed to the screen. This ends the active the enhancement, even if sent in an ***ML***.

Also, even with DISPLAY FUNCTIONS on, the current label terminator is never printed. To see that terminator printed it is necessary to temporarily change terminators.

Index

absolute 64, 76
absolute graph 18
absolute plots 18
ACCEPT LINK: GRAPHICS 13, 14
ACCEPT LINK: KEYBOARD 13
active 78, 93
active group 21
address 8
Address Map 14
address map 15
ADDRESS MAP 33
address, column 8
address, row 8
ADJUST COLUMN 15
ADJUST ROW 15
AE 30, 98
AK 32
ALLOC DISPLAY 15
ALLOC KEYBD 15
ALLOC SCREEN 15
ALPHA ENTRY 30
AM 33
anisotropic 59, 113
annunciator 58, 66
annunciator block 183
annunciators 22
ASCII 27, 30
aspect ratio 113, 181
ASSIGN KEYBOARD 32
ASSIGN SUBWINDOW 34
ATN 54
attributes 18
AW 22, 27, 34
AX 20, 37, 95, 106, 152, 198
axis 18
AXIS 37
BA 14, 39, 95, 96, 168
BASIC 28
BEEP 42
binary mode 26
BL 21, 41, 59, 95
BLANK AHEAD 39
blanked 39, 79
blanking 21, 79
blink 79
BLINK ON/OFF 41
blinking 21, 59, 79
Blocksize 147
borders 45
BP 42
BREAK LINK: GRAPHICS 13, 14, 15
BREAK LINK: KEYBOARD 12
brightness 86, 92
BUILD WINDOW 43
bus 43, 134
BW 9, 17, 22, 24, 27, 36, 43, 64, 152, 162, 193

CA 19, 45, 46, 52, 59, 95, 149, 159, 162, 189
capability 127
character 160
character line 17, 53, 66, 74, 182
character rotation 61, 64, 65
character set 46, 52, 59, 149, 159
Character Set 0 191
character set 30 19
character sets 24
character size 20, 56, 59, 153, 157
character sizes 56, 153, 189
character window 17, 182
character, user defined 18, 169
CI 48, 116, 187, 189

CL 19, 50, 95, 105
clear screen 14
clearing errors 187
clearing screen 57, 58, 86, 91, 133
col 134
column 33, 34, 35, 43
column address 8, 13, 14, 15
command cross reference 173
COMMAND RESPONSE 15
commands by function 177
commands, character sets 177
commands, display configuration 177
commands, error handling 177
commands, hardcopy output 177
commands, HP-GL 176
commands, informational displays 177
commands, labeling 178
commands, link requirements 175
commands, lists 173
commands, markers 178
commands, non-HP-GL 176
commands, referenced 178
commands, remote control 178
commands, simple graphics 179
commands, utility 179
Communication Protocol Design Guide 6
configuration 155
CONFIGURE LABEL 50
control characters 104, 208
control link 7, 8, 22, 24, 32, 44
Control Links 23
controller 8
CONTROLLER, SYSTEM 54
coordinate pairs 19
COPY 53
count 146
CR 27
CS 19, 46, 52, 59, 95, 149, 159, 160, 189
current group 21
CY 14, 53, 134
CZ 56

DA 57, 58, 79, 86, 130
DE 29, 58, 86
default 59
default parameters 59
defaults 23
define hardcpy 14, 53
DEFINE TERMINATOR 67
DELETE 63
DELETE ALL NON-REF/REF OBJECTS 57
DELETE SCREEN 58
Delta X 59
delta-X 68, 76, 80
Design Guides 5
DESIGNATE ALTERNATE CHARACTER SET 46
DESIGNATE STANDARD CHARACTER SET 52
device 134
DEVICE CLEAR 24, 52, 54, 67, 149, 159, 185, 187, 189
DF 23, 52, 59, 67, 91, 149, 159, 189
DI 59, 61, 64, 95, 96
direction, label 61, 64
direction, pen 161
display 7
DISPLAY 11, 12, 13, 15, 24, 103, 112
display 125
DISPLAY 142
DISPLAY FUNCTIONS 105
display functions 112
DISPLAY FUNCTIONS 209, 210
Display Interface Design Guide 6
Display Preset 14
DISPLAY PRESET 52, 67, 149, 159, 185, 187
DISPLAY STATUS ON/OFF 66

display units 17, 35, 61, 69, 82, 117, 122, 151, 195
DL 21, 63
dot 17
DR 59, 62, 64, 95, 96
DS 58, 66, 86, 183
DSP 11, 12, 13
DT 24, 27, 59, 67, 74, 104, 111
DX 14, 19, 59, 68, 77, 80, 95, 96, 137

EG 70, 115, 187
EJ 14, 23, 72
EJECT ON/OFF 72
Electrical Design Guide 5
Electromagnetic Compatibility Design Guide 6
END 27, 28, 29, 54, 70, 88
END COMMAND RESPONSE 15
endpoint 19
ENTER 15, 30
erasing screen 57, 58, 86, 91, 133
error 115
error description 185
error mask 24, 185
error message 70
ERROR OCCURRED 15
ERROR SCREEN 73
Error, protocol 24
ES 73
ESTABLISH LINK: GRAPHICS 13, 14, 15
ESTABLISH LINK: KEYBOARD 13, 15

FC 17, 27, 66, 67, 74, 208
FILL CHARACTER LINE 74
framing windows 45, 82

GA 20, 39, 68, 76, 80, 94, 95, 106, 129, 137, 152, 167
GP 20, 21, 57, 59, 77, 78, 94, 96, 124, 130, 132, 144, 154, 158, 171, 196
GR 20, 39, 68, 80, 95, 106, 130, 137, 167
GRAPH ABSOLUTE 76
GRAPH RELATIVE 80
graph, absolute 18
graph, relative 18
graphics 59
graphics link 7, 8, 13, 15, 17, 22, 24, 116, 182
graphics objects 19
graphics screen 48
graphics window 17, 22, 32, 36, 45, 182
graphics, referenced 20
graphs 19
graticule 18, 20
GRATICULE 81
graticules 18
grid 18, 20, 81
group 18, 20, 22, 92, 94, 107, 108, 124, 130, 154, 158
groups 133
GT 20, 45, 81, 95, 106, 181

hardkeys 103
height 153, 157
HF 83
HI RES ON/OFF 85
HOLDOFF ON/OFF 83
HP-GL 7, 25, 78, 107, 130, 153, 157, 161
HP-GL commands 176
HP-IB 7, 23
HP-MSIB 7
HP-MSIB Interface Design Guide 6
HR 14, 23, 85

I-P 12
IA 23, 86
ID 87, 118
identification 118
IDENTIFY GROUP 78
IDENTIFY ITEM 94
IEEE-488 23

IL 88, 119
illegal parameter 94
IM 11, 12, 23, 59, 89, 146, 185
IN 25, 29, 45, 52, 58, 86, 91, 149, 159, 185, 187, 189
INITIALIZE 91
initialized 91
initiator 24
INPUT LEARN STRING 88
INPUT MASK 89
instrument 8
intensified marker 22
intensity 19
INTENSITY ADJUST 86
INTENSITY SELECT 92
IS 92, 95, 96
isotropic 113
IT 18, 20, 21, 57, 77, 78, 94, 96, 124, 130, 132, 144, 154, 158, 170, 171, 196
item 18, 20, 22, 92, 94, 107, 108, 124, 130, 154, 158, 196
items 133

KC 14, 23, 53, 97
key 112
key code 98, 102
KEY COPY ON/OFF 97
key labels 97
Key Press 11
key press 141
keyboard 7, 32, 43, 45, 74, 102, 142, 193
keyboard link 7, 8, 12, 13, 17, 22, 24, 182
KEYCOPY 53
knob 12, 141, 146, 148
KP 98, 103, 141, 146
KY 11, 98, 102, 141, 187

label 47, 50, 58, 59, 61
LABEL 104
label 112, 158, 161
LABEL 208
label direction 61, 64
label terminator 14, 67
labels 149
LB 19, 27, 50, 67, 75, 94, 95, 104, 112, 157, 162, 208
LCL 11, 12
learn string 119
lettering direction 59
limits 117, 135
line type 19, 20, 22, 59, 81
line types 106
line, character 182
link 25
link requirements, commands 175
link, control 7, 8, 22, 23, 32, 44
link, graphics 7, 8, 13, 15, 17, 22, 23, 116, 182
link, keyboard 7, 8, 12, 13, 17, 22, 23, 91, 182
lists, commands 173
LOCAL 11, 12
local 24
LOCAL 98, 103, 142
LT 14, 59, 95, 96, 106, 181

MA 27, 59, 95, 96, 108, 110
mapping 59, 113
marker 18, 19, 20, 22, 59, 92, 108
MARKER 110
marker 149
MARKER ATTRIBUTES 108
marker position 108
marker, non-referenced 93
mask 23, 89, 185
masks 59
Mechanical Design Guide 5
memory 21, 145, 147, 163
MENU 11, 12, 13, 15
menu 112
MENU LOAD 111

menu window 17, 53, 172, 182
menus 112
MK 19, 20, 95, 108, 110
ML 13, 17, 27, 58, 67, 98, 111, 208
MNU 11, 12
model number 118
module 5
Module Development Design Guides 5
MORE ERRORS 15
MP 59, 113, 181
Multi-Instrument Systems 22

NMAA 13, 14
non-HP-GL commands 176
non-referenced 59
non-referenced graphics 20, 68, 78, 94, 107, 108, 110, 124, 129, 132, 133, 143, 144, 151, 154, 156, 157, 158
Numblocks 147

objects 18, 19
objects, graphics 18, 19
objects, text 18, 19
OE 70, 115, 187
offset 197
OG 48, 116, 187
OH 117
OI 87, 118
OL 88, 119
OO 25, 120
OP 122
OR 21, 59, 79, 95, 96, 123, 130, 131, 152, 196
origin 19, 21, 22, 59, 123, 152
OS 125, 187
OUTPUT 28, 29
OUTPUT CAPABILITY 127
OUTPUT CHARACTER SIZE 56
OUTPUT ERROR 115
OUTPUT ERROR MESSAGE 70
OUTPUT GRAPHICS LINK 116
OUTPUT HARD LIMITS 117
OUTPUT IDENTIFICATION 87, 118
OUTPUT LEARN STRING 119
OUTPUT OPTIONS 120
OUTPUT P1,P2 122
OUTPUT STATUS 125
OUTPUT VERSION 126
OV 126
OX 25
OY 127

PA 20, 21, 39, 51, 69, 76, 80, 95, 106, 129, 152, 167
PAGE 133
pair, x,y 17
PAN 137
parallel mask 187
parallel poll 89
PD 20, 77, 80, 95, 110, 130, 132, 161
pen 59, 156
pen direction 161
PEN DOWN 132
pen number 19, 20, 22, 156
pen stroke 161
PEN UP 144
PG 72, 86, 133
PI 14, 23, 53, 134
pixel 17
PL 14, 23, 53, 135
PLOT 11
plot 18, 19
PLOT 53, 103, 142
PLOT ABSOLUTE 129
PLOT RELATIVE 143
plot, absolute 18
plot, relative 18
plotter 54, 78
plotter dumps 53, 97, 107
PLOTTER LIMITS 135
plotting 129, 134

PN 19, 95, 137, 168
pointer 167
power-up 11
PP 9, 14, 59, 98, 140
PR 20, 39, 69, 95, 106, 130, 137, 143, 167
pre-process mode 24, 59
PRE-PROCESS MODE 140
PRINT 11, 53, 103, 142
printer 54
printer dump 85
printer dumps 53, 97
PRINTER/PLOTTER IS 134
printing 134
protocol error 24
protocols, use of 11
PU 20, 95, 130, 132, 144, 161

RC 145, 155
RECALL STATE 145
reference, command 173
referenced graphics 20, 68, 77, 78, 94, 107, 108, 110, 124, 129, 132, 143, 144, 151, 154, 156, 157, 158
referencing 22
register 145, 155
REJECT LINK: GRAPHICS 13
relative 64, 80
relative graphs 18
relative plots 18
REMAINING MEMORY 147
REMOTE 12
remote 24
REN 54
REPORT ERRORS 15
report errors 70
REPORT ERRORS 73, 115, 187
resolution 181
responder 24
responders 9
response 28
response mode 148
RETURN TO LOCAL 12
RG 146
rise 61, 64
RLTSEA 66
RM 21, 96, 147
rotation 20
rotation, character 65
row 33, 34, 35, 43, 134
row address 8, 13, 14, 15
RP 12, 146, 148, 187
RPG 12
run 61, 64

SA 19, 46, 52, 95, 149, 159, 162, 189
SAVE CONFIG 164
SAVE STATE 163
SAVE USER DEFINED CHARACTER 160
SC 18, 21, 35, 45, 61, 69, 91, 96, 124, 131, 150, 193
scale 131, 135, 150
scale factor 124, 151, 197
SCALE TO USER UNITS 150
scaling 195
screen 15, 22, 131, 193
screen layout 182
SELECT ALTERNATE CHARACTER SET 149
SELECT INSTR 52, 149, 159, 185, 187, 189
Select Instrument 13
SELECT INSTRUMENT 48
SELECT PEN 156
SELECT STANDARD CHARACTER SET 159
SELF TEST 166
SEND ALL ERRORS 15
SEND KEYBOARD DATA 102
SEND MODULE ID 13, 14, 15

SEND RPG DATA 148
separator 27
serial poll 14, 24, 90
SET ABSOLUTE CHARACTER SIZE 153
SET CHARACTER DIRECTION ABSOLUTE 61
SET CHARACTER DIRECTION RELATIVE 64
SET DEFAULT VALUES 59
SET DELTA-X 68
SET HP-IB 15
SET IEEE488 ADDRESS 15
SET LINE TYPE 106
SET MAPPING METHOD 113
SET ORIGIN 123
SET RELATIVE CHARACTER SIZE 157
show config 22, 36
SHOW CONFIGURATION 155
SI 56, 59, 95, 96, 105, 112, 153, 170, 189
SIMULATE KEY PRESSED 98
SIMULATE RPG TURNED 146
SN 155
softkey 58
softkey numbers 98
softkeys 98
SP 59, 95, 96, 156
space 27
SR 56, 59, 95, 96, 105, 157, 170
SRQ 11, 12, 14, 24, 48, 89, 91, 125, 141, 186
SS 19, 46, 52, 59, 95, 149, 159, 160, 189
state 163
STATUS 11, 12
status 24, 66
STATUS 90, 103, 125, 146, 186
status byte 14, 23, 24, 48, 59, 90, 91, 115, 125, 146, 148, 166, 185
status mask 14, 48
status window 53, 183
SU 18, 19, 47, 52, 160, 169
subwindow 22, 34, 35, 36
SV 23, 119, 145, 155, 163
SW 165
SWEEP ON/OFF 165
syntax 26
SYSTEM CONTROLLER 54
System Protocols 8

TE 166
terminator 24, 27, 59, 67, 74, 104, 111
test, self 166
text 50, 74, 104, 208
text objects 19
tic size 37
TP 19, 20, 39, 95, 96, 137, 167
trace 18, 19, 39, 68, 130, 137, 196
TRACE POINTER 167

UC 18, 19, 161, 169
USER 11, 12, 13, 15
user defined character 19, 160
USER DEFINED CHARACTER 169
user defined characters 18
user units 17, 35, 38, 69, 77, 80, 81, 82, 113, 150, 157, 195
userunits 151
USR 11, 12

vector list 13, 22, 147, 195
version 126
view 59
VIEW ON/OFF 171
VW 21, 59, 95, 171

WHICH MENU 172
width 153, 157
window 7, 9, 14, 22, 43, 44, 82, 92, 113, 124, 125, 131, 136, 151, 162, 193
window 5 24, 32, 36

window borders 45
window, character 17, 182
window, controller 82
window, corners 17
window, graphics 17, 22, 32, 36, 45, 182
window, menu 17, 53, 172, 182
window, status 53, 183
WM 172

x position 110
X-increment 68
x-increment 143, 160, 169
Xcount 81
Xintercept 37
Xlength 37
Xmax 35, 43, 135, 150
Xmin 35, 43, 135, 150
Xpartition 37
Xpercent 56
Xposition 124
Xtic 37

y position 110
Y-coordinate 76
y-increment 80, 143, 160, 169
y-value 76
Ycount 81
Yintercept 37
Ylength 37
Ymax 35, 43, 135, 150
Ymin 35, 43, 135, 150
Ypartition 37
Ypercent 56
Yposition 124
Ytic 37

Manual Part Number 5958-6653