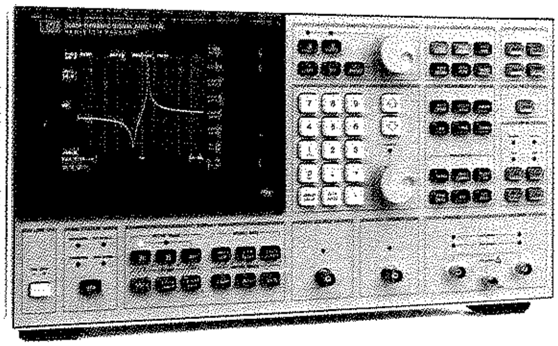
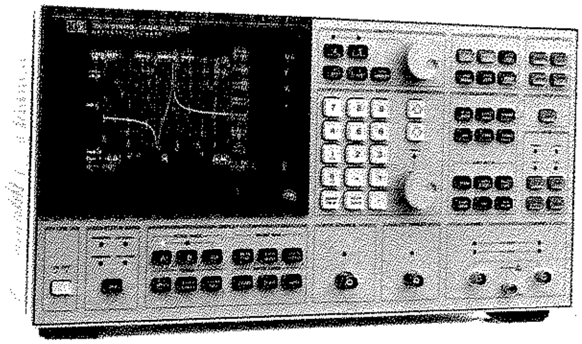
HEWLETT PACKARD

3562A DYNAMIC SIGNAL ANALYZER
PROGRAMMING INSTRUCTIONS



DOCUMENT DIRECTORY



OBJECTIVE OF THIS MANUAL

The objective of this manual is to show you how to program the HP 3562A over the Hewlett-Packard Interface Bus (HP-IB) using an external controller. This manual assumes you are familiar with operating the instrument from the front panel (refer to the *HP 3562A Operating Manual* for information on front panel features).

HOW THIS MANUAL IS ORGANIZED

This manual has three major parts:

1. **Programming Instructions** This section (Chapters 1—6) provides the HP-IB mnemonic commands for all front panel operations and explains how to use the bus-only features not available from the front panel.

2. **Introductory Programming Guide (IPG)** This programming note (Appendix A) shows you how to get started with HP-IB if you have never programmed an instrument over the bus.

3. **Quick Reference Guide (QRG)** This section (Appendix B) lists the front panel and bus-only commands alphabetically by command name and provides limits and syntax.

If you want an introduction to programming the HP 3562A over the HP-IB, first use the IPG. This explains how to get started and provides several example programs written for HP 9000 Series 200 computers with Basic 3.0. You may also want to use Chapter 2 of the programming instructions; this shows the keys and softkeys in the instrument and their HP-IB commands.

If you want to learn the instrument's HP-IB capabilities, its response to bus management commands, and its status byte/service request organization, use Chapter 1 of the programming instructions.

If you are familiar with the operation of the HP 3562A and the HP-IB in general, you can start programming using just the QRG to learn the syntax of individual commands.

If you want to use the bus-only commands to transfer data, perform special signal processing, program the display or utilize the special control commands, use Chapters 3 through 6 of the programming instructions.

How the Programming Instructions Are Organized

The programming instructions comprise Chapters 1 through 6 of this manual.

Chapter 1 "The HP 3562A and the HP-IB," describes the instrument's general HP-IB capabilities, including bus management commands and service requests (SRQs). For detailed information on programming SRQs, refer to Chapter 6.

Chapter 2 "The Front Panel Group," shows the key and softkey menu structure with all the HP-IB mnemonic commands for emulating front panel operation.

Chapter 3 "The Data Transfer Group," shows how to use the bus-only commands that transfer data traces, instrument states, coordinate transform blocks, and synthesis and curve fit tables.

Chapter 4 "The Signal Processing Group," shows how to use the bus-only commands that provide signal processing primitives.

Chapter 5 "The Display Control Group," shows how to create custom displays by directly programming the HP 3562A's vector display.

Chapter 6 "The Control and Command Group," primarily shows how to program the status byte, instrument status register, and activity status register to generate SRQs.

Notes On Using This Manual

In an effort to describe the syntax of the analyzer's commands, sample program lines are provided throughput this manual. While these were written in HP BASIC 3.0, their objective is to show you how commands are implemented in general. If you are programming in another version of BASIC or another language, study the example lines and apply their solutions to your case.

If you would like complete example programs, rather than just sample lines, the IPG in Appendix A offers twelve programs written in BASIC 3.0. In addition, Appendix C provides a number of programs written in BASIC 2.0 and 3.0 that you may be able to adapt to your needs.

PROGRAMMING MANUAL

Model 3562A
Dynamic Signal Analyzer

WARNING

To prevent potential fire or shock hazard, do not expose equipment to rain or moisture.

Manual Part No. 03562-90031
Microfiche Part No. 03562-90231

Printed Oct: 1985

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard product is warranted against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Hewlett-Packard Company will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by -hp-. Buyer shall prepay shipping charges to -hp- and -hp- shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to -hp- from another country.

HP software and firmware products which are designated by HP for use with a hardware product, when properly installed on that hardware product, are warranted not to fail to execute their programming instructions due to defects in materials and workmanship. If HP receives notice of such defects during their warranty period, HP shall repair or replace software media and firmware which do not execute their programming instructions due to such defects. HP does not warrant that the operation of the software, firmware or hardware shall be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HEWLETT-PACKARD SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office. Addresses are provided at the back of this manual.

HEWLETT PACKARD

SAFETY SUMMARY

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements. This is a Safety Class 1 instrument.

GROUND THE INSTRUMENT

To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The instrument is equipped with a three conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the instrument in the presence of flammable gases or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY INSTRUMENT

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the instrument. Return the instrument to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

Dangerous voltages, capable of causing death, are present in this instrument. Use extreme caution when handling, testing, and adjusting.

SAFETY SYMBOLS

General Definitions of Safety Symbols Used On Equipment or In Manuals.

⚠	Instruction manual symbol: the product will be marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the instrument.
ϟ	Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts must be so marked).
⏚ OR (⏚)	Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.
(⏚)	Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. A terminal marked with this symbol must be connected to ground in the manner described in the installation (operating) manual, and before operating the equipment.
⏛ OR ⊥	Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.
	Alternating current (power line).
⎓	Direct current (power line).
≂	Alternating or direct current (power line).
WARNING	The WARNING sign denotes a hazard. It calls attention to a procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.
CAUTION	The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product.
N O T E:	The NOTE sign denotes important information. It calls attention to procedure, practice, condition or the like, which is essential to highlight.

Chapter 1

3562A AND THE HP-IB

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to describe the HP-IB capabilities of the HP 3562A and explain how it interacts with the HP-IB in general. It assumes you are familiar with the operation of the HP 3562A and with HP-IB programming. The topics covered in this chapter are:

1. The HP-IB capabilities of the HP 3562A

    —Interface capabilities
    —Controller capabilities
    —Interrupts and instrument status

2. The HP 3562A's response to bus management commands

3. Overview of the HP 3562A's command set

    —Front panel group
    —Data transfer group
    —Signal processing group
    —Display group
    —Control and communication group

4. Programming hints

If you are new to HP-IB, use the Introductory Programming Guide in Appendix A to get started. This shows you how to connect an HP-IB system and provides example programs in BASIC 3.0 for HP 9000 Series 200 computers.

For general information on the HP-IB, contact your HP Sales Representative for copies of the following documents:

Tutorial Description of the Hewlett-Packard Interface Bus
Part Number 5952-0156

Condensed Description of the Hewlett-Packard Interface Bus
Part Number 59401-90030

HP-IB CAPABILITIES OF THE HP 3562A

The HP 3562A can be operated via HP-IB in two modes: as system controller or as an addressable-only device. When it is the system controller, the analyzer directs the flow of commands and data on the bus. When it is addressable-only, it responds to commands and data from the system controller. The HP 3562A also has the ability to interrupt the system controller and provide information about its internal status. This section starts by listing the interface capabilities of the analyzer, then describes its controller capabilities. The last part in this section describes the interrupt and instrument status features.

Interface Capabilities

The HP 3562A has the following interface capabilities, as defined by IEEE Standard 488-1978:

SH1	complete Source handshake
AH1	complete Acceptor handshake
T6	basic Talker; serial poll; unaddress if MLA; no Talk Only
TE0	no Extended Talker capability
L4	basic Listener; unaddress if MTA; no Listen Only
SR1	complete Service Request capability
RL1	complete Remote/Local capability
PP0	no Parallel Poll capability
DC1	complete Device Clear capability
DT1	complete Device Trigger capability
C1	system Controller capability
C2	send IFC and Take Charge Controller capability
C3	send REN Controller capability
C12	send IF messages; receive control; pass control capability
E1	open-collector drivers (250 kBytes/s maximum)

Refer to IEEE Standard 488-1978 if you need more detailed information.

Controller Capabilities

The HP 3562A's system controller capability allows it to directly control digital plotters, access disc drives, and output HP-IB command strings. When it is the only controller in an HP-IB system, the analyzer is usually operated as the system controller. (The SYSTEM CNTRLR softkey in the **HP-IB FCTN** menu is active.)

When operated on the bus with another controller (a desktop computer, for example), the analyzer generally operates in addressable-only mode. (ADDRES ONLY in the **HP-IB FCTN** menu is active.) When the HP 3562A needs to be in control of the bus, it can accept control from the system controller, then automatically pass control back when finished.

Refer to "Passing Control" in Chapter 6 for more information, including use of the Controller Address (CTAD) command.

Interrupt and Instrument Status Features

When the HP 3562A is in addressable-only mode, it can generate service requests (SRQs) to the system controller for two general reasons: it needs control of the bus to perform an operation or there is a change in its status that the controller might want to know about.

The HP 3562A communicates interrupt and status information primarily with its **status byte.** This 8-bit byte is sent in response to a serial poll and is encoded to provide a number of status indications. One of these indications is that there has been a change in the instrument's status. Specific information about this change is contained in the **instrument status register.** Finally, the system controller can monitor the analyzer's current activity by reading its **activity status register.** Together, these indicators can provide a great deal of information to the system controller. Table 1-1 shows all the status indications offered by the HP 3562A.

Table 1-1 Status Indications in the HP 3562A

Indication	Status Byte	Instrument Status Register	Activity Status Register
Requested service	*		
Error generated	*		
Ready for HP-IB commands	*		
User SRQs	*		
End of disc action	*		
End of plot action	*		
Power up	*		
Key pressed	*		
Various plotter & disc requests	*		
Instrument status change	*		
Measurement pause		*	
Auto sequence pause		*	
End of measurement		*	
Sweep point ready		*	
Channel 1 over range		*	
Channel 2 over range		*	
Channel 1 half scale		*	
Channel 2 half scale		*	
Source fault		*	
Reference locked		*	
Marker knob turned		*	
Entry knob turned		*	
Activity status change		*	
System failure			*
Filling time record			*
Filters settling			*
Curve fit in progress			*
Missed external sample			*
Timed preview active			*
Data accepted			*
Waiting for trigger			*
Waiting for arm			*
Calibration in progress			*

The status byte is read by performing a serial poll of the analyzer. The instrument status register is read by sending the IS? command. The activity status register is read by sending the AS? command. Complete information on using these, including masking, is provided in Chapter 6.

BUS MANAGEMENT COMMANDS

When the bus is in the command mode (the ATN line is true), bus management commands can be used to control interface hardware connected to the bus. This section describes the HP 3562A's response to the primary bus commands. Your controller's programming or interfacing manual should contain information on these commands from the controller's perspective.

Abort I/O

This command instructs the HP 3562A to abort input or output. It is an unconditional assumption of control of the bus by the system controller. All bus activity halts and the HP 3562A becomes unaddressed. This does not, however, clear the analyzer's HP-IB command buffer or clear any pending data input or output. The HP 3562A does not relinquish bus control when it receives this command.

BASIC example: ABORT 7

Clear Lockout & Set Local

This command instructs all instruments on the specified port to clear the local-lockout mode and return to local (front panel) operation. This command differs from the LOCAL command in that the LOCAL command addresses a specific device and does not clear the lockout mode.

BASIC example: LOCAL 7

Device Clear

The CLEAR command can affect a specific device (addressed clear) or all devices on a specified port (universal clear). This command causes the HP 3562A to clear its HP-IB command buffer; reset the SRQ bus management line (if it had been activated by the instrument); reset all status byte, instrument status and activity status masks; and abort any data input or output. This command unconditionally interrupts bus activity and gains control of the instrument. It does not, however, reset any HP 3562A parameters.

BASIC examples: CLEAR 720 (addressed clear)
CLEAR 7 (universal clear)

Local

The LOCAL command returns local (front panel) control to the HP 3562A. (When the instrument is under local control, the REMOTE front panel indicator is off, and the keyboard is enabled.) The HP-IB command buffer is not cleared by issuing this command. Any load operation in progress continues but the HP 3562A aborts dump operations in progress. (It does this if it receives any command over the bus.)

BASIC example: LOCAL 720

Local Lockout

This command disables the **LOCAL** front panel key of the HP 3562A. It does not change the remote/local status of the instrument; it does prevent the operator from using the **LOCAL** key to enable the front panel keyboard when the REMOTE command is in effect. When in remote control, LOCAL LOCKOUT secures the system from operator interference. While this command is in effect *and* the instrument is in remote control, the only way to return to front panel operation is by issuing the LOCAL command on the bus. If an unaddressed (universal) LOCAL command is used (e.g., LOCAL 7) LOCAL LOCKOUT is disabled and subsequent remote commands can be overridden from the front panel. If an addressed LOCAL command is used (e.g., LOCAL 720), local lockout will still be in effect when the device is later returned to remote control.

BASIC example: LOCAL LOCKOUT 7

Parallel Poll

This command and its accompanying PARALLEL POLL CONFIGURE are ignored by the HP 3562A. See SERIAL POLL.

Parallel Poll Configure

This command and its accompanying PARALLEL POLL are ignored by the HP 3562A. See SERIAL POLL.

Pass Control

This command shifts control of the bus from one controller to another. The Controller Address command, CTAD, (default = 21) should be sent prior to passing control. Not all controllers have the ability to pass control. Consult the operation manual of your controller to determine its capabilities in this respect.

If control is passed to the HP 3562A before it has a need for it, the analyzer immediately passes the control to the address specified by the Controller Address command. Refer to "Passing Control" in Chapter 6.

BASIC examples: PASS CONTROL 720
SEND 7; UNL UNT TALK 20 CMD 9

Remote

When this command is issued the front panel LED annunciator labeled "REMOTE" illuminates and the front panel keys are disabled (except the **LOCAL** key *if* local lockout is not active; if local lockout is active, even the **LOCAL** key is disabled). This command can be used to address the HP 3562A to listen.

BASIC examples: REMOTE 7 (universal)
REMOTE 720 (addressed)

Serial Poll

The SPOLL command instructs the HP 3562A to send its status byte to the controller. This action is usually taken in response to a service request (SRQ). Upon receiving the status byte, the controller should examine it to determine what type of service the analyzer requires. If your program sends multiple serial polls, pause for at least 5 ms between them.

BASIC example: Status—byte = SPOLL(720)

Trigger

This command triggers measurements in the HP 3562A in the same manner as its other trigger modes. TRIGGER must first be enabled in the analyzer by sending the "HPT" command to select HP-IB triggering.

BASIC example: TRIGGER 7 (universal)
TRIGGER 720 (addressed)

THE HP 3562A'S HP-IB COMMAND SET

The HP 3562A's command set includes the front panel keys and softkeys (with a few exceptions) plus a number of commands available only via HP-IB. The command set is divided into five groups:

Front panel group	keys & softkeys
Data transfer group Signal processing group Display control group Control/communication group	bus-only commands

Front Panel Group

As it name implies, this group emulates the keys and softkeys on the analyzer's front panel. A few are not programmable, however: the **LINE** key and the editing softkeys in the alpha mode (SPACE FORWRD, etc.). The alpha editing softkeys are not needed on the bus because you simply send the alpha string after entering it on your controller's keyboard.

Chapter 2 provides mnemonics for the front panel group. It is alphabetized by key, with the softkeys associated with each key listed in order of appearance. Chapter 2 is designed to help you easily emulate front panel operation via HP-IB.

Data Transfer Group

This group allows you to transfer data traces and instrument states in and out of the instrument. Both traces and states can be transferred in ASCII, ASNI binary, and a fast binary mode used internally by the analyzer.

Chapter 3 explains how to use these commands, including interpreting data headers and converting data traces.

Signal Processing Group

This group provides access to the analyzer's signal processing primitives. It allows you to set up data blocks in memory, operate on these blocks (using FFT, averaging, etc.), then transfer the blocks back to the controller or display them on the analyzer.

Chapter 4 explains how to set up blocks, get data into them, use the signal processing primitives, then get the processed data back out.

Display Control Group

This group provides control of the HP 3562A's vector display. The display can be controlled at three levels: using individual HP-GL (Hewlett-Packard Graphics Language) commands, loading an entire display from a controller, or defining the display as the plotter and using HP BASIC 3.0 graphics commands.

Chapter 5 shows how to program the display using these three approaches.

Command/Communication Group

This final group provides control and communications functions, including service requests, instrument and activity status, reading marker values, and communicating with the front panel.

Chapter 6 explains how to use these commands. It also describes the status byte, instrument status, and activity status parameters.

PROGRAMMING HINTS

1. See the beginning of Chapter 2 for emulating front panel commands.

2. Pause the controller for several seconds after sending resets or special presets if you want to send marker or math commands.

3. When programming anything on the display—especially markers—make sure that there is a valid data display first.

4. The HP 3562A can buffer up to 3 lines of 80 characters each.

5. If you request information from the analyzer (query, data transfer, etc.), allow for the information to be input to the controller immediately.

6. If AUTO CAL is ON, you will encounter long delays when the cal routine is run. This could affect your program if it contains time outs. To avoid this, it is suggested that you send the following sequence of commands:

   ```
   AUTO 0
   RST
   SNGL
   ```

   This deactivates auto cal, then runs a single cal routine.

7. When activating external sampling (ESMP1), pause the program briefly to allow the HP 3562A to measure the external sample clock.

Chapter 2

FRONT PANEL COMMANDS

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to show you the HP-IB commands for the HP 3562A's keys and softkeys. In addition, this chapter explains special considerations for some of the front panel commands. For syntax, entry ranges and suffixes, refer to the Quick Reference Guide in Appendix B. The rest of the commands—the "bus-only" commands—are covered in Chapters 3 through 6.

GETTING STARTED

A major difference between operating the instrument from the front panel and programming it over the HP-IB is that you do not always have to follow the softkey menu structure with the HP-IB. For example, to select the FFT math function from the front panel, you press MATH followed by NEXT, followed by NEXT, followed by FFT. Over the bus, however, you simply send the FFT command. In a few cases, a particular menu must be displayed before a command can be used. An example is CLEAR TABLE in synthesis. You need to first display the POLE ZERO, POLE RESIDU or POLYNOMIAL menu before telling the instrument to clear the table. These special cases are identified and explained in this chapter.

Another consideration when programming over the HP-IB: Several pairs of softkeys have the same name but different mnemonics. For example, to select the frequency response measurement and the frequency response display from the front panel, you press FREQ RESP in the **SELECT MEAS** menu and FREQ RESP in the **MEAS DISP** menu. Over the bus, however, you send FRSP and FRQR. The organization of this chapter by key avoids this problem entirely, and such cases in the QRG listing are explained as well.

Softkeys which toggle between two states (e.g, TIM AV ON OFF, AVG AU FIX) can be toggled using the basic mnemonic. However, to guarantee the resultant state, these commands allow you to send 0 or 1 after the mnemonic to explicitly choose one state or the other. Sending "TIAV1" explicitly activates time averaging, while "TIAV" merely toggles the existing state. The results of sending 0 and 1 for each toggle softkey are explained in the Quick Reference Guide.

This chapter is organized alphabetically by the keys on the front panel. Under each key, the softkeys it accesses are displayed in order of appearance. The HP-IB mnemonic is shown beside every key and softkey. This organization lets you leverage your knowledge of front panel operation into writing controller programs. If you know the particular functions you want to program, use the Quick Reference Guide (QRG) in Appendix B, which lists all keys and softkeys in alphabetical order. The QRG also explains the syntax for all commands.

There are two more ways of learning HP-IB mnemonics for individual commands: the HELP displays and the command echo. The mnemonic is shown at the top of all HELP displays, and the letters that make up the mnemonic are underlined in the command echo field on the display.

Parameter Queries

You can learn the current value of any variable parameter in the analyzer by sending the appropriate command followed by a question mark. For example, to learn the current frequency span, you could send the following BASIC statements:

```
OUTPUT 720; "FRS?"
ENTER 720; Freq—span
```

where 720 is the analyzer's address
FRS is the mnemonic for the FREQ SPAN softkey
Freq—span is the variable the value is entered into

The Alpha Menu

The softkeys in the alpha menu (SPACE FORWRD, SPACE BACKWD, INSERT ON OFF, DELETE CHAR, CLEAR LINE, AT POINTR, OVER WRITE, and CANCEL ALPHA) are not programmable over the HP-IB. When you need to send alpha characters, simply include them with the commands. The Quick Reference Guide (Appendix B) shows the syntax for every command requiring alpha entries.

A	(A)
A&B	(B)
ARM	(ARM)
AUTO MATH	(AMTH)
EDIT MATH	(EDMA)
VIEW MATH	(VWMA)
START MATH	(STMA)
LABEL MATH	(LBLM)[1]
EDIT LINE#	(LINE)[1]
DELETE LINE	(DLTL)[1]
CHANGE LINE	(CHGL)[1,2]
ADD LINE	(ADDL)[1,2]
CLEAR MATH	(CLMA)[1]
END EDIT	(ENED)[1]

[1]The EDIT MATH menu must be displayed before these commands can be used.

[2]When ADDL or CHGL is sent, the analyzer stays in the add line or change line mode, respectively. All subsequent commands until ENED (END EDIT) are added or changed.

AUTO SEQ	(ASEQ)
START ASEQ1	(ASQ1)[3]
START ASEQ2	(ASQ2)[3]
START ASEQ3	(ASQ3)[3]
START ASEQ4	(ASQ4)[3]
START ASEQ5	(ASQ5)[3]
PAUSE ASEQ	(PSAS)
CONT ASEQ	(CNAS)
SELECT ASEQ#	(SASQ)
EDIT	(EDIT)
VIEW	(VIEW)
LABEL ASEQ	(LBLA)[1]
EDIT LINE#	(LINE)[1]
DELETE LINE	(DLTL)[1]
CHANGE LINE	(CHGL)[1,2]
ADD LINE	(ADDL)[1,2]
CLEAR ASEQ	(CLAS)[1]
ASEQ FCTN	(ASFN)[1]
END EDIT	(ENED)[1]
LOOP TO	(LPTO)
GO TO	(GOTO)
ASEQ MESSGE	(ASMS)
TIMED PAUSE	(TIPS)
TIMED START	(TIST)
DSPLAY ON OFF	(DSPL)
RETURN	(RTN)

[1]The EDIT menu must be displayed before these commands can be used.

[2]When ADDL or CHGL is sent, the analyzer stays in the add line or change line mode, respectively. All subsequent commands until ENED (END EDIT) are added or changed.

[3]The mneumonics for these are always ASQ1-5, even when the labels are replaced by user-defined labels.

AVG

(linear res mode)

NUMBER AVGS	(NAVG)
AVG OFF	(AVOF)
STABLE (MEAN)	(STBL)
EXPON	(EXP)
PEAK HOLD	(PHLD)
CONT PEAK	(CNPK)
TIM AV ON OFF	(TIAV)
NEXT	(NX)
OVRLP%	(OVLP)
OV REJ ON OFF	(OVRJ)
FST AVG ON OFF	(FSAV)
PRVIEW OFF	(PROF)
MANUAL PRVIEW	(MAPR)[1]
TIMED PRVIEW	(TIPR)[1]
RETURN	(RTN)

(log res mode)

NUMBER AVGS	(NAVG)
AVG OFF	(AVOF)
STABLE (MEAN)	(STBL)
EXPON	(EXP)
PEAK HOLD	(PHLD)
CONT PEAK	(CNPK)
NEXT	(NX)
OVRLP%	(OVLP)
OV REJ ON OFF	(OVRJ)
FST AVG ON OFF	(FSAV)
RETURN	(RTN)

(swept sine mode)

NUMBER AVGS	(NAVG)
AUTO INTGRT	(AUIN)
FIXED INTGRT	(FXIN)
INTGRT TIME	(INTM)

(time capture mode)

NUMBER AVGS	(NAVG)
AVG OFF	(AVOF)
STABLE (MEAN)	(STBL)
EXPON	(EXP)
PEAK HOLD	(PHLD)
CONT PEAK	(CNPK)
TIM AV ON OFF	(TIAV)
OVRLP%	(OVLP)

[1]Use ACPT for YES and REJT for NO when previewing over the bus.

B (B)

CAL (CAL)

AUTO ON OFF	(AUTO)
SINGLE CAL	(SNGC)

COORD (CORD)

MAG (dB)	(MGDB)
MAG (dBm)	(MDBM)
MAG (LOG)	(MGLG)
MAG (LIN)	(MAG)
PHASE	(PHSE)
REAL	(REAL)
IMAG	(IMAG)
NEXT	(NEXT)

NYQUST	(NYQT)
NICHOL	(NICL)
LOG X	(LOGX)
LIN X	(LINX)
RETURN	(RTN)

CURVE FIT (CVFT)

CREATE FIT	(CRFT)
STOP FIT	(SPFT)
NUMBER POLES	(NPOL)
NUMBER ZEROS	(NZER)
LAST MEAS	(LSMS)
A & B TRACES	(ABTR)
EDIT TABLE	(EDTB)
FIT FCTN	(FTFN)
EDIT POLES	(EPOL)
EDIT ZEROS	(EZER)
FIX LINE#	(FXLN)
UNFIX LINE#	(UFLN)
ADD LINE	(ADLN)
DELETE LINE#	(DLLN)
TABLE FCTN	(TBFN)
RETURN	(RTN)
TIME DELAY	(TMDL)
SCALE FREQ	(SCLF)
CLEAR TABLE	(CLTA)
RETURN	(RTN)
USER WEIGHT	(USWT)
AUTO WEIGHT	(AUWT)
USER ORDER	(USOR)
AUTO ORDER	(AUOR)
FIT → SYNTH	(FTSN)
SYNTH → FIT	(SNFT)
EDIT WEIGHT	(EDWT)
RETURN	(RTN)
VIEW WEIGHT	(VWWT)
WEIGHT REGION	(WTRG)
WEIGHT VALUE	(WTVL)
STORE WEIGHT	(STWT)
RETURN	(RTN)

DISC	(DISC)
SAVE FILE	(SAVF)
RECALL FILE	(RCLF)
DELETE FILE	(DLTF)
VIEW CATLOG	(CAT)
NEXT PAGE	(NXTP)
PREV PAGE	(PRVP)
CATLOG POINTR	(CTPT)
DISC FCTN	(DIFN)
SERVCE FCTNS	(SVFN)
DISC COPY	(DICO)
FORMAT	(FORM)
PACK DISC	(PKDI)
THRUPT SIZE	(THSZ)
CREATE THRUPT	(CRTH)
ABORT HP-IB	(ABIB)
RETURN	(RTN)
FORMAT OPTION	(FOOP)
INIT DISC	(INDI)
INIT CATLOG	(INCT)
RETURN	(RTN)
DESTN ADDRES	(DEAD)
DESTN UNIT	(DEUN)
COPY FILES	(COFI)
OVERWR AU MAN	(OVAU)
RESUME OVERWR	(RSOV)
RESUME COPY	(RSCO)
IMAGE BACKUP	(IMBK)
RETURN	(RTN)
RESTOR CATLOG	(RSCT)
RO ERT TEST	(RERT)
OUTPUT LOG	(OULG)
NEXT PAGE	(NXPG)
CLEAR LOGS	(CLLG)
DISC STATUS	(DIST)
SPARE BLOCK	(SPBL)
RETURN	(RTN)
FAULT LOG	(FTLG)
ERT LOG	(ERLG)
RUN TM LOG	(RULG)
RETURN	(RTN)

ENGR UNITS (ENGR)

EU VAL CHAN1	(EUV1)
VOLTS CHAN1	(VLT1)
EU LBL CHAN1	(EUL1)
EU VAL CHAN2	(EUV2)
VOLTS CHAN2	(VLT2)
EU LBL CHAN2	(EUL2)

FREQ (FREQ)

(linear res & time capture modes)

FREQ SPAN	(FRS)
START FREQ	(SF)
CENTER FREQ	(CF)
ZERO START	(ZST)
MAX SPAN	(MAXS)
TIME LENGTH	(TLN)
E SMPL ON OFF	(ESMP)
SAMPLE FREQ	(SMPF)

(log res mode)

FREQ SPAN	(FRS)
START FREQ	(SF)

(swept sine mode)

FREQ SPAN	(FRS)
CENTER FREQ	(CF)
START FREQ	(SF)
STOP FREQ	(SPF)
RESLTN	(RES)
RESLTN AU FIX	(RSAU)
SWEEP RATE	(SWRT)[1]

[1]Same as SWEEP RATE in **SOURCE** menu.

FRONT BACK (FRBK)

HP-IB FCTN (IBFN)

SYSTEM CNTRLR	(SYSC)
ADDRES ONLY	(ADRS)
SELECT ADDRES	(SADR)
USER SRQ	(USRQ)
OUTPUT STRING	(OUT)[1]
ABORT HP-IB	(ABIB)[2]

[1] This is programmable over the bus only when entering output strings into an auto sequence. OUTPUT STRING cannot be executed immediately because the HP 3562A must be the system controller to use this function.
[2] Same as ABORT HP-IB in the USER LIMITS menu.

USER SRQ1	(SRQ1)
USER SRQ2	(SRQ2)
USER SRQ3	(SRQ3)
USER SRQ4	(SRQ4)
USER SRQ5	(SRQ5)
USER SRQ6	(SRQ6)
USER SRQ7	(SRQ7)
USER SRQ8	(SRQ8)
HP-IB ADDRES	(IBAD)
PLOT ADDRES	(PLAD)
DISC ADDRES	(DIAD)
DISC UNIT	(DIUN)
RETURN	(RTN)

INPUT COUPLE (ICPL)

CHAN1 AC DC	(C1AC)
CHAN2 AC DC	(C2AC)
FLOAT CHAN1	(FLT1)
GROUND CHAN1	(GND2)
FLOAT CHAN2	(FLT2)
GROUND CHAN2	(GND2)

LOCAL (LCL)

MATH (MATH)

ADD	(ADD)
SUB	(SUB)
MPY	(MPY)
DIV	(DIV)
SQUARE ROOT	(SQRT)
RECIP	(RCIP)
NEGATE	(NEG)
NEXT	(NXT)
DIFF	(DIFF)
jw	(JW)
INTGRT	(INGR)
INTGRT INIT = 0	(INGI)
jw^{-1}	(JW1)
T/1—T	(TT)
NEXT	(NEX)
RETURN	(RTN)
REAL PART	(RLPT)
COMPLX CONJ	(CMPC)
LN OF DATA	(LN)
LN^{-1} OF DATA	(LN1)
FFT	(FFT)
FFT^{-1}	(FFT1)
RETURN	(RTN)

MEAS DISP (MDSP)

(linear res mode
freq resp measurement)

FREQ RESP	(FRQR)
COHER	(COHR)
POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
CROSS SPEC	(CSPC)
IMPLS RESP	(IRSP)
AUTO MATH	(AUMT)
FILTRD INPUT	(FILT)

(linear res mode
power spec measurement)

POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
AUTO MATH	(AUMT)
FILTRD INPUT	(FILT)

(linear res mode
cross corr measurement)

CROSS CORR	(CRCR)
AUTO CORR1	(AUC1)
AUTO CORR2	(AUC2)
AUTO MATH	(AUMT)
FILTRD INPUT	(FILT)

(linear res mode
auto corr measurement)

AUTO CORR1	(AUC1)
AUTO CORR2	(AUC2)
AUTO MATH	(AUMT)
FILTRD INPUT	(FILT)

(linear res mode
histogram measurement)

HIST1	(HIS1)
HIST2	(HIS2)
PDF1	(PDF1)
PDF2	(PDF2)
CDF1	(CDF1)
CDF2	(CDF2)
AUTO MATH	(AUMT)
FILTRD INPUT	(FILT)

(all linear res mode)

TIME REC 1	(TMR1)
TIME REC 2	(TMR2)
LINEAR SPEC1	(LSP1)
LINEAR SPEC2	(LSP2)
ORBITS T1vsT2	(ORBT)
DEMOD POLAR	(POLR)
INST	(INST)
INST WNDOWD	(IWND)
AVRG	(AVRG)
RETURN	(RTN)

(log res mode
freq resp measurement

FREQ RESP	(FRQR)
COHER	(COHR)
POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
CROSS SPEC	(CSPC)
AUTO MATH	(AUMT)

(log res mode
(power spec measurement)

POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
AUTO MATH	(AUMT)

(swept sine mode)

FREQ RESP	(FRQR)
COHER	(COHR)
POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
CROSS SPEC	(CSPC)
AUTO MATH	(AUMT)

(time capture mode
power spec measurement)

POWER SPEC1	(PSP1)
POWER SPEC2	(PSP2)
FILTRD INPUT	(FILT)

(time capture mode
(histogram measurement)

HIST1	(HIS1)
HIST2	(HIS2)
PDF1	(PDF1)
PDF2	(PDF2)
CDF1	(CDF1)
CDF2	(CDF2)
FILTRD INPUT	(FILT)

(time capture mode
auto corr measurement)

AUTO CORR1	(AUC1)
AUTO CORR2	(AUC2)
FILTRD INPUT	(FILT)

(all time capture)

TIME REC1	(TMR1)
TIME REC2	(TMR2)
LINEAR SPEC1	(LSP1)
LINEAR SPEC2	(LSP2)
INST	(INST)
AVRG	(AVRG)
RETURN	(RTN)

MEAS MODE	(MSDS)
LINEAR RES	(LNRS)
LOG RES	(LGRS)
SWEPT SINE	(SSIN)
TIME CAPTUR	(CPTR)
CAPTUR SELECT	(CPSE)
THRUPT ON OFF	(THRU)
THRUPT SELECT	(THSE)
DEMOD ON OFF	(DMOD)
DEMOD SELECT	(DMSE)
LINEAR SWEEP	(LNSW)
LOG SWEEP	(LGSW)
A GAIN ON OFF	(AGON)
A GAIN SELECT	(AGSE)
REF CHAN1	(RFC1)
REF CHAN2	(RFC2)
REF LEVEL	(RFLV)
SOURCE LIMIT	(SRLM)
RETURN	(RTN)
START CAPTUR	(STCP)
ABORT CAPTUR	(ABCP)
CAPTUR POINTR	(CPNT)
POINTR INCRMT	(PTIN)
CAPTUR LENGTH	(CLEN)
CAPTUR HEADER	(CHED)
RETURN	(RTN)
START THRUPT	(STHR)
ABORT THRUPT	(ABTH)
ACTIVE FILE	(ACFL)
THRUPT LENGTH	(THLN)
THRUPT HEADER	(THED)
RETURN	(RTN)
DEMOD CHAN1	(DM1)
DEMOD CHAN2	(DM2)
DEMOD BOTH	(DMB)
PRVIEW ON OFF	(PRON)
PM/FM CRRIER	(PFCR)
DELETE FREQ	(DLFR)
DELETE ON OFF	(DLON)
RETURN	(RTN)

AM CHAN1	(AM1)
FM CHAN1	(FM1)
PM CHAN1	(PM1)
AM CHAN2	(AM2)
FM CHAN2	(FM2)
PM CHAN2	(PM2)
RETURN	(RTN)
AUTO CRRIER	(ACRR)
USER CRRIER	(UCRR)
EDIT LINE#	(EDLN)[1]
DELETE REGION	(DLRG)[1]
CHANGE REGION	(CHRG)[1]
ADD REGION	(ADRG)[1]
CLEAR TABLE	(CLRT)[1]
RETURN	(RTN)

[1]The DELETE FREQ menu must be displayed before this command can be used.

PAUSE CONT (PSCN)[1]

[1]PSCN switches back and forth between pause and continue. PAUS explicitly pauses, and CONT explicitly continues, regardless of the key's previous state. To be certain of the resulting state, use PAUS or CONT over the bus.

PLOT (PLOT)

START PLOT	(STPL)
SELECT DATA	(SDAT)
SELECT PENS	(SPEN)
SPEED F S	(SPED)
LINE TYPES	(LNTP)
PAGING CONTRL	(PCTL)
PLOT LIMITS	(PLIM)
PLOT PRESET	(PLPR)
DATA ONLY	(DATA)
DATA & ANNOT	(DAAN)
DFAULT GRIDS	(DFGR)
SOLID GRIDS	(SLGR)
TICK MARKS	(TKMK)
RETURN	(RTN)
GRID PEN	(GRDP)
TRACE A PEN	(TRAP)
TRACE B PEN	(TRBP)
ANNOT A PEN	(ANAP)
ANNOT B PEN	(ANBP)
MARKER PEN	(MKRP)
RETURN	(RTN)

SOLID LINES	(SLDL)
DASHED LINES	(DSHL)
DOTS	(DOTS)
SOLIDA DASH B	(SLDA)
USER LINES	(ULIN)
LINE A TYPE#	(LINA)
LINE B TYPE#	(LINB)
RETURN	(RTN)
PAGE FORWRD	(PGFW)
PAGE BACKWD	(PGBK)
NO PAGING	(NOPG)
CUT PG ON OFF	(CTPG)
RETURN	(RTN)
PLOT AREA	(PLAR)
GRID AREA	(GRAR)
DFAULT LIMITS	(DLIM)
USER LIMITS	(ULIM)
ROT 90 ON OFF	(ROT)
SET P1 LWR LF	(SEP1)
SET P2 UPR RT	(SEP2)
READ PEN→P1	(RDP1)
READ PEN→P2	(RDP2)
ABORT HP-IB	(ABIB)[1]
RETURN	(RTN)

[1]Same as ABORT HP-IB in the **HP-IB FCTN** menus.

PRESET (PRST)

F RESP LINRES	(FRLN)
F RESP LOGRES	(FRLG)
F RESP SWEPT	(FRSW)
P SPEC LINRES	(PSLN)
TIME CAPTUR	(TMCP)
TIME THRUPT	(THTH)
RESET	(RST)

RANGE (RNG)

CHAN 1 RANGE	(CIRG)
AUTO 1 RNG UP	(AU1U)
AUTO 1 UP&DWN	(AU1)
CHAN 2 RANGE	(C2RG)
AUTO 2 RNG UP	(AU2U)
AUTO 2 UP&DWN	(AU2)

SAVE RECALL (SAVR)

RECALL PWR DN	(RCLP)
RECALL STATE#	(RCLS)
SAVE STATE#	(SAVS)
RECALL DATA#	(RCLD)
SAVE DATA#	(SAVD)

SCALE (SCAL)

X FIXD SCALE	(XSCL)
X MRKR SCALE	(XMKR)[1]
X AUTO SCALE	(XASC)[1]
Y FIXD SCALE	(YSCL)
Y MRKR SCALE	(YMKR)[2]
Y AUTO SCALE	(YASC)[2]
Y DFLT SCALE	(YDSC)[2]

[1]Same as corresponding softkey in the X menu.
[2]Same as corresponding softkey in the Y menu.

SELECT MEAS (SMES)

(linear res mode)

FREQ RESP	(FRSP)
POWER SPEC	(PSPC)
AUTO CORR	(AUCR)
CROSS CORR	(CCOR)
HIST	(HIST)
CH 1&2 ACTIVE	(CH12)
CH 1 ACTIVE	(CH1)
CH 2 ACTIVE	(CH2)

(log res mode)

FREQ RESP	(FRSP)
POWER SPEC	(PSPC)
CH 1&2 ACTIVE	(CH12)
CH 1 ACTIVE	(CH1)
CH 2 ACTIVE	(CH2)

(swept sine mode)

FREQ RESP	(FRSP)

(time capture mode)

POWER SPEC	(PSPC)
AUTO CORR	(AUCR)
HIST	(HIST)
CH 1 ACTIVE	(CH1)
CH 2 ACTIVE	(CH2)

SELECT TRIG (SELT)

TRIG LEVEL	(TRLV)
ARM AU MAN	(ARMA)
FREE RUN	(FREE)
CHAN 1 INPUT	(C1IN)
CHAN 2 INPUT	(C2IN)
SOURCE TRIG	(SRTG)
EXT	(EXT)
SLOPE + −	(SLOP)

SINGLE (SNGL)

SOURCE (SRCE)

(linear res & time capture modes)

SOURCE LEVEL	(SRLV)
DC OFFSET	(DCOF)
SOURCE OFF	(SROF)
RANDOM NOISE	(RND)
BURST RANDOM	(BRND)
PRIODC CHIRP	(PCRP)
BURST CHIRP	(BCPR)
FIXED SINE	(FSIN)

(log res mode)

SOURCE LEVEL	(SRLV)
DC OFFSET	(DCOF)
SOURCE OFF	(SROF)
RANDOM NOISE	(RND)
FIXED SINE	(FSIN)

(swept sine mode)

SOURCE LEVEL	(SRLV)
DC OFFSET	(DCOF)
SOURCE ON OFF	(SRON)
SWEEP UP	(SWUP)
SWEEP DOWN	(SWDN)
SWEEP HOLD	(SWHD)
MANUAL SWEEP	(MNSW)
SWEEP RATE	(SWRT)[1]

[1]Same as SWEEP RATE in the **FREQ** menu.

SPCL FCTN (SPFN)

SELF TEST	(TST)
SERVIC TEST	(SVTS)
TIME H,M,S	(TIME)
DATE M,D,Y	(DATE)
BEEPER ON OFF	(BEEP)
SOURCE PROTCT	(SRPT)
PwrSRQ ON OFF	(PSRQ)

PROTCT ON OFF	(PTON)
RAMP TIME	(RAMP)
RETURN	(RTN)

SPCL MARKER		(SPMK)
	X FCTN OFF	(XFOF)
	HMNC ON	(HMNC)
	SBAND ON	(SBND)
	SLOPE	(SLP)
	FREQ & DAMP	(FRDA)
	POWER	(PWR)
	MRKR→PEAK	(MKPK)
	AVG VALUE	(AVGV)
	CRRIER FREQ	(CRFR)
	SBAND INCRMT	(SBIN)
	MOD INDEX	(MIND)
	SBAND POWER	(SPWR)
	CALC OFF	(CLOF)
	RETURN	(RTN)
	FNDMTL FREQ	(FNFR)
	HMNC POWER	(HPWR)
	THD	(THD)
	CALC OFF	(CAOF)
	RETURN	(RTN)
START		(STRT)
STATE TRACE		(STTR)[1]

[1]STTR switches back and forth between state and trace. STAT explicitly displays the state, and TRAC explicitly displays the trace(s). To be certain of the resulting condition, use STAT and TRAC over the bus.

SYNTH	(SNTH)
POLE ZERO	(PZRO)
POLE RESIDU	(PRSD)
POLY NOMIAL	(POLY)
CONVRT TABLE	(CVTB)
CREATE CONST	(CCON)
CREATE TRACE	(CTRC)
TO→POL ZERO	(TOPZ)
TO→POL RESIDU	(TOPR)
TO →POLY	(TOPY)
EDIT POLE#	(EDPL)[1,2]
EDIT ZERO#	(EDZR)[1]
EDIT RESDU#	(EDRS)[2]
EDIT NUMER#	(EDNM)[3]
EDIT DENOM#	(EDDN)[3]
DELETE VALUE	(DLTV)
CHANGE VALUE	(CHGV)
ADD VALUE	(ADDV)
SYNTH FCTN	(SNFN)
CLEAR TABLE	(CLTB)
RETURN	(RTN)
GAIN FACTOR	(GAIN)
TIME DELAY	(TDLY)
SCALE FREQ	(SCFR)
RETURN	(RTN)

[1]POLE ZERO menu must be displayed before these commands can be used.
[2]POLE RESIDU menu must be displayed before these commands can be used.
[3]POLY NOMIAL menu must be displayed before these commands can be used.

TRIG DELAY	(TRGD)
CHAN1 DELAY	(C1DL)
CHAN2 DELAY	(C2DL)

UNITS	(UNIT)
L SPEC UNITS	(LSUN)
P SPEC UNITS	(PSUN)
SWEPT UNITS	(SWUN)
Hz (Sec)	(HZS)
RPM (Sec)	(RPMS)
Orders (Revs)	(ORDR)
Orders CAL	(ORCL)
TRACE TITLE	(TITL)
VOLTS PEAK	(VTPK)[1]
VOLTS RMS	(VTRM)[1]
VOLTS	(VLTS)[1]
$VOLTS^2$	(VT2)[1]
RETURN	(RTN)
VOLTS PEAK	(VTPK)[1]
VOLTS RMS	(VTRM)[1]
VOLTS	(VLTS)[1]
$VOLTS^2$	(VT2)[1]
$V/\sqrt{Hz}$ ($\sqrt{PSD}$)	(VHZ)[1]
V^2/Hz (PSD)	(V2HZ)
V^2s/Hz (ESD)	(V2SH)
RETURN	(RTN)
VOLTS PEAK	(VTPK)[1]
VOLTS RMS	(VTRM)[1]
RETURN	(RTN)

[1]Appropriate menu (L SPEC UNITS, P SPEC UNITS or SWEPT UNITS) must be displayed before these commands can be used. It is recommended that you always send the first level menu command before the second level command. For example, PSUN;VTPK for VOLTS PEAK in the P SPEC UNITS menu.

UPPER LOWER	(UPLO)

VIEW INPUT (VWIN)

(linear res, log res & swept sine modes)

INPUT TIME 1	(ITM1)
INPUT TIME 2	(ITM2)
INPUT SPEC 1	(ISP1)
INPUT SPEC 2	(ISP2)
VIEW OFF	(VWOF)

(time capture mode)

INPUT TIME 1	(ITM1)
INPUT TIME 2	(ITM2)
INPUT SPEC 1	(ISP1)
INPUT SPEC 2	(ISP2)
TIME RECORD	(TMRC)
LINEAR SPEC	(LSPC)
TIME BUFFER	(TMBF)
VIEW OFF	(VWOF)

(time throughput active)

INPUT TIME 1	(ITM1)
INPUT TIME 2	(ITM2)
INPUT SPEC 1	(ISP1)
INPUT SPEC 2	(ISP2)
THRUPT TIME 1	(THT1)
THRUPT TIME 2	(THT2)
NEXT RECORD	(NXRC)
VIEW OFF	(VWOF)

WINDOW (WNDO)

HANN	(HANN)
FLAT TOP	(FLAT)
UNIFRM (NONE)	(UNIF)
FORCE EXPON	(FOXP)
USER SAVD1	(USD1)
FORCE CHAN1	(FRC1)
EXPON CHAN1	(XPN1)
FORCE CHAN2	(FRC2)
EXPON CHAN2	(XPN2)

X (X)

X VALUE	(XVAL)
X MRKR SCALE	(XMKR)[1]
X AUTO SCALE	(XASC)[1]
SCROLL ON OFF	(SCRL)
HOLD X CENTER	(HXCT)
HOLD X RIGHT	(HXRT)
HOLD X LEFT	(HXLF)
HOLD X OFF	(HXOF)

[1]Same as corresponding softkey in the **SCALE** menu.

X OFF		(XOFF)
Y		(Y)
	Y VALUE	(YVAL)
	Y MRKR SCALE	(YMKR)[1]
	Y AUTO SCALE	(YASC)[1]
	Y DFAULT SCALE	(YDSC)[1]
	HOLD Y CENTER	(HYCT)
	HOLD Y UPPER	(HYUP)
	HOLD Y LOWER	(HYLW)
	HOLD Y OFF	(HYOF)

[1]Same as corresponding softkey in the **SCALE** menu.

Y OFF		(YOFF)

Chapter 3

THE DATA TRANSFER GROUP

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to show you how to perform data block transfers between a controller and the HP 3562A. The following topics are addressed:

1. Data formats offered by the HP 3562A

2. Loading/dumping data traces

3. Loading/dumping instrument states

4. Dumping the coordinate transform block

5. Loading/dumping the synthesis table

6. Accessing capture and throughput files on disc

This chapter deals only with these data block transfers. For display buffer transfers, see Chapter 5. For signal processing primitive block transfers, see Chapter 4. Note: ANSI and ASCII transfers cannot be performed while the HP 3562A is in auto sequence edit.

THREE DATA FORMATS

The HP 3562A offers three data formats for transferring data via HP-IB: ASCII, ANSI floating point binary, and a non-standard binary used internally by the instrument. All three formats are provided to better address the needs of specific instrument/controller operations.

Every data transfer requires a format specifier, a length word, a header, and data. In some cases there is no header, and in others there is no data. But in general, these four items are required. The format specifier and length word depend on the data format; these are discussed separately in the following descriptions of data formats. The header and data depend on the type of data (e.g., data trace) being transferred; these are discussed throughput this chapter with each type of transfer.

NOTE

This section compares relative speeds of the three data transfer formats. This is done for comparison purposes only. No guarantees of actual transfer rates are expressed or implied. Transfer rates are highly dependent on system configuration, instrument state, controller, language, and programming.

Each data format offers unique advantages; the choice depends on speed requirements, data being transferred, and ability to handle each format. The ASCII format is the slowest of the three, but it is a commonly used standard format. The ANSI floating point format is relatively fast and is a standard format compatible with many controllers. Internal binary is the fastest data transfer format offered by the HP 3562A.

ASCII Data Format

The ASCII (American national Standard Code for Information Interchange) format is a common data communication code which uses 8-bit bytes to represent single characters. The transfer rate for this format is the slowest of the three because ASCII requires many more bytes per number, as compared to ANSI and internal binary. Also, the HP 3562A's internal processor must take the time to convert data between the binary format it uses to the ASCII data format before dumping and after loading.

The format specifier for ASCII data is #I, and the length word following this shows the number of *variables* to be transferred. Refer to "HP 3562A Internal Binary Format" for an explanation of the conversion of header information to ASCII.

ANSI Floating Point Format

The ANSI binary data format is the 64-bit floating point binary data format specified by IEEE draft standard P754 and used by HP Series 200 and other computer/controllers. This format is faster than the ASCII format because fewer bytes are required to specify a number to a given number of decimal places. It is a standard format used by many controllers and, therefore, saves controller time if the data block is to be processed outside the instrument. Table 3-1 shows the bit arrangement of ANSI floating point.

Table 3-1 ANSI Floating Point Format

	Bit 15 … Bit 0
Word 0:	S E E E E E E E E E E E M M M M
Word 1:	M M M M M M M M M M M M M M M M
Word 2:	M M M M M M M M M M M M M M M M
Word 3:	M M M M M M M M M M M M M M M M

where S is sign bit
E is biased exponent (+1023)
M is mantissa

The format specifier for ANSI data is #A, and the length word following this shows the number of *bytes* to be transferred. Refer to "HP 3562A Internal Binary Format" for an explanation of the conversion of header information to ASCII.

HP 3562A Internal Binary Format

The internal binary format consists of several data types combined in each transfer to maximize transfer rates. This format is faster than ASCII or ANSI because no conversion is required in the analyzer and fewer bytes are transferred. The following data types are used with internal binary:

64-bit floating point ("long real")

32-bit floating point ("real")

32-bit integer ("long integer")

16-bit integer ("integer")

String

Data transfers are always in 32-bit floating point when using internal binary format. Header information, including instrument state, is a combination of all five types. The explanations of each transfer later in this chapter list the data type for each variable. The following paragraphs describe these five internal binary data types.

64-bit Floating Point Data Type ("Long Real")

FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF EEEEEEEE

where: F is an intermediate fractional bit of the mantissa in two's-complement form with a binary point between the most significant (sign) bit and the next bit and the mantissa is normalized.

E is the exponent part in two's-complement integer format.

This format does not allow the first two bits to be identical (i.e., both 0 or both 1) unless the number represented is zero. A non zero mantissa always describes a fraction F, where $0.5 \leq F < 1$ or $-0.5 \geq F \geq -1$.

The 64-bit floating point data type is used infrequently in headers. For ASCII and ANSI transfers, each internal 64-bit value is converted to one 64-bit ANSI value.

32-bit Floating Point Data Type ("Real")

FFFFFFFF FFFFFFFF FFFFFFFF EEEEEEEE

Refer to the 64-bit description for explanations of E and F. The 32-bit floating point data type is used for both data and header transfers. For ASCII and ANSI transfers, each internal 32-bit value is converted to one 64-bit value in ANSI format.

32-Bit Integer Data Type ("Long Integer")

This data type consists of 32-bit integers in two's complement format. This type is used infrequently in headers. For ASCII and ANSI transfers, each 32-bit internal integer is converted to two 16-bit integers (high word, low word), each of which is then converted to 64-bit ANSI floating point format.

16-Bit Integer Data Type ("Integer")

This data type consists of 16-bit integers in two's complement format. This type is used extensively in headers for Boolean and enumerated values. For ASCII and ANSI transfers, each internal 16-bit integer is converted to one 64-bit ANSI floating point value.

String Data Type

The string data type consists of ASCII-encoded bytes representing alphanumeric data. Each string is preceded by one byte indicating the number of data bytes in the string. Each data byte represents one alphanumeric character. For ASCII and ANSI transfers, two string bytes are converted to to one 64-bit floating point element. First, two successive string bytes are concatenated to form one 16-bit integer, then that integer is floated.

LOADING/DUMPING DATA TRACES

The active trace on the display can be dumped in any of the three data formats. When a trace is dumped, the data points are preceded by the format specifier, the length word, and a group of bytes called the "header." The header defines the conditions under which the trace is being displayed. The contents of the header are described in the next section.

Trace data is dumped either in real/imaginary pairs or real numbers and represents the measured data, not what is necessarily on the screen. For example, if you attempt to dump a phase trace, you will get the entire frequency response function from which the phase trace was derived. If you want strictly what is on the display, refer to "Dumping the Coordinate Transform Block" later in this chapter. All data trace dumps are calibrated (i.e., no conversion is needed); the data points are scaled in the current units.

The Data Header

The header dumped with data traces is the same for all three data formats. The only difference is in the variable count. Table 3-2 shows the contents of the data header. For data types listed in the table as "E-type" (enumerated type) the value of that variable can be decoded by referring to table 3-3. The range of values for each E-type is shown in parentheses. The (+1) beside the byte count for strings is a reminder that each string is preceded by one length byte.

The index shown for each format indicates the position in the header that each item is located for each data format. The binary index indicates 16-bit words, and the ASCII/ANSI index indicates elements.

Table 3-2 Contents of the Data Header

Item	Data Type	Size (bytes)	Binary Index	ASCII/ANSI Index
Display function	E-type (0-49)	2	1	1
Number of elements	Integer	2	2	2
Displayed elements	Integer	2	3	3
Number of averages	Integer	2	4	4
Channel selection	E-type (0-3)	2	5	5
Overflow status	E-type (0-3)	2	6	6
Overlap percentage	Integer	2	7	7
Domain	Integer	2	8	8
Volts peak/rms	E-type (0-2)	2	9	9
Amplitude Units	E-type (0-7)	2	10	10
X Axis Units	E-type (0-35)	2	11	11
Auto Math Label	String	13 (+1)	12	12
Trace Label	String	21 (+1)	19	19
EU Label 1	String	5 (+1)	30	30
EU Label 2	String	5 (+1)	33	33
Float/Integer	Boolean	2	36	36
Complex/Real	Boolean	2	37	37
Live/Recalled	Boolean	2	38	38
Math result	Boolean	2	39	39
Real/Complex input	Boolean	2	40	40
Log/Linear data	Boolean	2	41	41
Auto math	Boolean	2	42	42
Real time status	Boolean	2	43	43
Measurement Mode	E-type (0-4)	2	44	44
Window	E-type (0-8)	2	45	45
Demod type chan 1	E-type (45-47)	2	46	46
Demod type chan 2	E-type (45-47)	2	47	47
Demod active chan 1	Boolean	2	48	48
Demod active chan 2	Boolean	2	49	49
Average status	E-type (0-2)	2	50	50
Not used	Integers (2)	4	51	51
Samp freq/2 (real)	Real	4	53	53
Samp freq/2 (imag)	Real	4	55	54
Not used	Real	4	57	55
Delta X-axis[2]	Real	4	59	56
Max range (for scaling)	Real	4	61	57
Start time value	Real	4	63	58
Expon wind const 1	Real	4	65	59
Expon wind const 2	Real	4	67	60
EU value chan 1	Real	4	69	61
EU value chan 2	Real	4	71	62
Trig delay chan 1	Real	4	73	63
Trig delay chan 2	Real	4	75	64
Start freq value	Long Real	8	77	65
Start data value	Long Real	8	81[1]	66

[1]The last word for binary is number 84.
[2]$10^{\Delta X}$ for log-x axis.

Table 3-3 Enumerated Types for Data Header Variables

Display Function

0 No data

1 Frequency response
2 Power spectrum 1
3 Power spectrum 2
4 Coherence
5 Cross spectrum

6 Input time 1
7 Input time 2
8 Input linear spectrum 1
9 Input linear spectrum 2
10 Impulse response

11 Cross correlation
12 Auto correlation 1
13 Auto correlation 2
14 Histogram 1
15 Histogram 2

16 Cumulative density function 1
17 Cumulative density function 2
18 Probability density function 1
19 Probability density function 2
20 Average linear spectrum 1

21 Average linear spectrum 2
22 Average time record 1
23 Average time record 2
24 Synthesis pole-zero
25 Synthesis pole-residue

26 Synthesis polynomial
27 Synthesis constant
28 Windowed time record 1
29 Windowed time record 2
30 Windowed linear spectrum 1

31 Windowed linear spectrum 2
32 Filtered time record 1
33 Filtered time record 2
34 Filtered linear spectrum 1
35 Filtered linear spectrum 2

36 Time capture buffer
37 Captured linear spectrum
38 Captured time record
39 Throughput time record 1
40 Throughput time record 2

41 Curve fit
42 Weighting function
43 Not used
44 Orbits
45 Demodulation polar

46 Preview demod record 1
47 Preview demod record 2
48 Preview demod linear spectrum 1
49 Preview demod linear spectrum 2

Channel Selection

0 Channel 1
1 Channel 2
2 Channels 1 & 2
3 No channel

Overload Status

0 Channel 1
1 Channel 2
2 Channels 1 & 2
3 No channel

Domain Type

0 Time
1 Frequency
2 Voltage (amplitude)

Volts peak/rms

0 Peak
1 RMS
2 Volts (indicates peak only)

Amplitude Units

0 Volts
1 Volts squared
2 PSD (V^2/Hz)
3 ESD (V^2s/Hz)
4 $\sqrt{}$PSD (V/$\sqrt{}$Hz)

5 No amplitude units
6 Unit volts
7 Unit volts^2

X Axis Units

0 No units
1 Hertz
2 RPM
3 Orders
4 Seconds

5 Revs
6 Degrees
7 dB
8 dBV
9 Volts

10 V$\sqrt{}$Hz ($\sqrt{}$PSD)
11 Hertz/second
12 Volts/EU
13 Vrms
14 V^2/Hz (PSD)

15 Percent
16 Points
17 Records
18 Ohms
19 Hertz/Octave

20 Pulse/Rev
21 Decades
22 Minutes
23 V^2s/Hz (ESD)
24 Octave

25 Seconds/Decade
26 Seconds/Octave
27 Hz/Point
28 Points/Sweep
29 Points/Decade

30 Points/Octave
31 V/Vrms
32 V^2
33 EU referenced to Chan 1
34 EU referenced to Chan 2

35 EU value

Measurement Mode

0 Linear resolution
1 Log resolution
2 Swept sine
3 Time capture
4 Linear resolution throughput

Demod Type Chan 1

45 AM
46 FM
47 PM

Demod Type Chan 2

45 AM
46 FM
47 PM

Average Status

0 No data (any data received are invalid)
1 Not averaged
2 Averaged

Window

0 Window not applicable
1 Hann
2 Flat top
3 Uniform
4 Exponential

5 Force
6 Force chan 1/expon chan 2
7 Expon chan. 1/force chan 2
8 User

ASCII Format

To dump the active trace in ASCII, use the DDAS (Dump Data in AScii) command; to load data in ASCII, use LDAS (Load Data in AScii). The format specifier is #I, and the two bytes (one word) following that indicate the number of variables to be transferred. For example, the following BASIC statements dump then load a frequency response trace:

```
OPTION BASE 1                      ! Set array base to 1
DIM Data__buffer (1668)            ! Create array for data
OUTPUT @Dsa; "DDAS"                ! Dump ASCII command
ENTER @Dsa USING "2A,K"; F$,L      ! Read format & length
PRINT F$,L                         ! Verify format & length
ENTER @Dsa; Data__buffer(*)        ! Read header & trace data

OUTPUT @Dsa; "LDAS"                ! Load ASCII command
OUTPUT @Dsa USING "2A,K"; F$,L     ! Output format & length
OUTPUT @Dsa; Data__buffer(*)       ! Output header & trace data
```

Because this is an ASCII transfer, the format specifier read into F$ is #I. For the specific case of a frequency response trace, the length word read into L indicates 1668 variables (1601 data points (801 real/imaginary pairs) plus the 66-element header). To make this a general program, you should redimension the array Data__buffer to L after reading L.

ANSI Format

To dump the active trace in ANSI floating point, use the DDAN (Dump Data in ANsi) command; to load data in ANSI, use LDAN (Load Data in ANsi). The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump then load a frequency response trace:

```
OPTION BASE 1                      ! Set array base to 1
DIM Data__buffer (1668)            ! Create array for data
OUTPUT @Dsa; "DDAN"                ! Dump ANSI command
ENTER @Dsa USING "%,2A,W"; F$,L    ! Read format & length
PRINT F$,L                         ! Verify format & length
ASSIGN @Dsa; FORMAT OFF            ! Allow binary data
ENTER @Dsa; Data__buffer(*)        ! Read header & trace data
ASSIGN @Dsa; FORMAT ON             ! Allow ASCII data

OUTPUT @Dsa; "LDAN"                ! Load ANSI command
OUTPUT @Dsa USING "#,2A,W"; F$,L   ! Output format & length
ASSIGN @Dsa; FORMAT OFF            ! Allow binary data
OUTPUT @Dsa; Data__buffer(*)       ! Output header & trace data
ASSIGN @Dsa; FORMAT ON             ! Allow ASCII data
```

Notice that Data__buffer is dimensioned to the number of bytes divided by 8 (13344/8 = 1668); this is a 64-bit floating point transfer wherein every 8 bytes coming from the analyzer represent one element of data. Because this is an ANSI transfer, the format specifier read into F$ is #A. For the specific case of a frequency response trace, the length word read into L indicates 13344 bytes. To make this a general program, you should redimension the array Data__buffer after reading L. Remember to divide the length word value by 8 for all ANSI transfers.

Internal Binary Format

To dump the active trace in internal binary, use the DDBN (Dump Data in internal BiNary) command; to load data in internal binary use LDBN (Load Data in internal BiNary). The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump then load a frequency response trace:

```
OPTION BASE 1                          ! Set array base to 1
INTEGER Data__buffer (3288)            ! Create array for data
OUTPUT @Dsa; "DDBN"                    ! Dump binary command
ENTER @Dsa USING "%,2A,W"; F$,L        ! Read format & length
PRINT F$,L                             ! Verify format & length
ASSIGN @Dsa; FORMAT OFF                ! Allow binary data
ENTER @Dsa; Data__buffer(*)            ! Read header & trace data
ASSIGN @Dsa; FORMAT ON                 ! Allow ASCII data

OUTPUT @Dsa; "LDBN"                    ! Load binary command
OUTPUT @Dsa USING "#,2A,W"; F$,L       ! Output format & length
ASSIGN @Dsa; FORMAT OFF                ! Allow binary data
OUTPUT @Dsa; Data__buffer(*)           ! Output header & trace data
ASSIGN @Dsa; FORMAT ON                 ! Allow ASCII data
```

Notice that Data__buffer is dimensioned to the number of bytes divided by 2 ($6576 \div 2 = 3288$). The header contains 168 bytes (84 16-bit integers), and the data trace contains 6408 bytes (1602 32-bit floating point values).

Because this is a binary transfer, the format specifier read into F$ is #A. For the specific case of a frequency response trace, the length word read into L indicates 6576 bytes. To make this a general program, you should redimension the array Data__buffer after reading L.

LOADING/DUMPING THE INSTRUMENT STATE

The instrument state can be dumped and loaded in any of the three data formats. When you dump the instrument state, you get what is shown in the state display (you don't need to display the state to dump it, however).

Contents of the Instrument State

Table 3-4 shows the contents of the instrument state. For those data types listed as "E-type" (enumerated type), refer to table 3-5 to decode the value. The range of values for each E-type is shown in parentheses. The (+1) beside the byte count for strings is a reminder that each string is preceded by one length byte.

The index shown for each format indicates the position in the header that each item is located for each data format. The binary index indicates 16-bit words, and the ASCII/ANSI index indicates elements.

Table 3-4 Contents of the Instrument State

Item	Data Type	Size (bytes)	Binary Index	ASCII/ANSI Index
Measurement mode	E-type (0-3)	2	1	1
Measurement 1	E-type (0-5)	2	2	2
Measurement 2	E-type (0-5)	2	3	3
Window type	E-type (11-15)	2	4	4
Force/Expon window 1	E-type (0-1)	2	5	5
Force/Expon window 2	E-type (0-1)	2	6	6
Average type	E-type (6-10)	2	7	7
Overlap percentage	Integer	2	8	8
Number of averages	Integer	2	9	9
Sweep # of averages	Integer	2	10	10
Trigger type	E-type (18-23)	2	11	11
Trigger slope	E-type (16-17)	2	12	12
Preview type	E-type (0-2)	2	13	13
Sample type	E-type (24-25)	2	14	14
Range units chan 1	E-type (8-35)	2	15	15
Range units chan 2	E-type (8-35)	2	16	16
Range type 1	E-type (26-28)	2	17	17
Range type 2	E-type (26-28)	2	18	18
Input coupling 1	E-type (29-30)	2	19	19
Input coupling 2	E-type (29-30)	2	20	20
Source type	E-Type (31-36)	2	21	21
Chirp percent	Integer	2	22	22
Burst percent	Integer	2	23	23
Sweep direction	E-type (0-1)	2	24	24
Sweep mode	E-Type (38-39)	2	25	25
Ext sample freq units	E-Type (1-20)	2	26	26
Bandwidth units	E-Type (1-3)	2	27	27
Log span index	Integer	2	28	28
Log start index	Integer	2	29	29
Sweep rate units	E-Type (11-26)	2	30	30
Auto gain ref chan	E-Type (0-3)	2	31	31
Demod channels	E-type (0-3)	2	32	32
Demod type chan 1	E-type (45-47)	2	33	33
Demod type chan 2	E-type (45-47)	2	34	34
Source level units	E-type (8-13)	2	35	35
Source offset units	E-type (9)	2	36	36
Trigger level units	E-type (9-34)	2	37	37
Capt/thru length units	E-type (4-17)	2	38	38
EU label 1	String	5(+ 1)	39	39
EU Label 2	String	5(+ 1)	42	42
Auto carrier on/off	Boolean	2	45	45
Time average on/off	Boolean	2	46	46
Auto/fixed resolution	Boolean	2	47	47
Auto gain on/off	Boolean	2	48	48
Auto/fixed integrate	Boolean	2	49	49

Table 3-4 Contents of the Instrument State cont.

Item	Data Type	Size (bytes)	Binary Index	ASCII/ANSI Index
Fast average on/off	Boolean	2	50	50
Overload reject on/off	Boolean	2	51	51
Chan 1 float/ground	Boolean	2	52	52
Chan 2 float/ground	Boolean	2	53	53
Time throughput on/off	Boolean	2	54	54
Demodulation on/off	Boolean	2	55	55
EU or volts chan 1	Boolean	2	56	56
EU or volts chan 2	Boolean	2	57	57
Manual/auto arm	Boolean	2	58	58
Demod preview on/off	Boolean	2	59	59
Delete freq on/off	Boolean	2	60	60
Lin res Fstart pegged	Boolean	2	61	61
Swept Fstart pegged	Boolean	2	62	62
Force length chan 1	Real	4	63	63
Force length chan 2	Real	4	65	64
Expon time constant 1	Real	4	67	65
Expon time constant 2	Real	4	69	66
Sweep time	Real	4	71	67
Sweep rate	Real	4	73	68
Sweep resolution	Real	4	75	69
Sweep integrate time	Real	4	77	70
Auto gain level	Real	4	79	71
Auto gain limit	Real	4	81	72
Source level	Real	4	83	73
EU value chan 1	Real	4	85	74
EU value chan 2	Real	4	87	75
Trigger delay chan 1	Real	4	89	76
Trigger delay chan 2	Real	4	91	77
Integrate var thresh	Real	4	93	78
Capt/thru length	Real	4	95	79
Frequency span	Real	4	97	80
Time record length	Real	4	99	81
Frequency resolution	Real	4	101	82
Time resolution	Real	4	103	83
External sample rate	Real	4	105	84
Sample rate (actual)	Real	4	107	85
Range channel 1	Real	4	109	86
Range channel 2	Real	4	111	87
Preview time	Real	4	113	88
Trigger level	Real	4	115	89
Source dc offset	Real	4	117	90
Fixed sine frequency	Long Real	8	119	91
Start frequency	Long Real	8	123	92
Center frequency	Long Real	8	127	93
Sweep start	Long Real	8	131	94
Sweep end	Long Real	8	135	95
Carrier frequency	Long Real	8	139*	96

*Last word for binary is number 142.

Table 3-5 Enumerated Types for Instrument State Values

Measurement Mode

0 Linear resolution
1 Log resolution
2 Swept sine
3 Time capture

Measurement Type

0 Frequency response
1 Cross correlation
2 Power spectrum
3 Auto correlation
4 Histogram
5 No measurement

Window Type

11 Hanning
12 Flat top
13 Uniform
14 User window
15 Force/Exponential

Force/Exponential Window

0 Force
1 Exponential

Average Type

6 Stable
7 Exponential
8 Peak
9 Continuous Peak
10 Averaging Off

Trigger Type

18 Free Run
19 Channel 1
20 Channel 2
21 External
22 Source Trigger
23 HP-IB Trigger

Trigger Slope

16 Positive
17 Negative

Preview Type

0 Manual Preview
1 Timed Preview
2 Preview Off

Sample Type

24 Internal Sample
25 External Sample

Range Units

8 dBV
9 Volts
13 Vrms
35 EU

Auto Range Type

26 Auto Range On
27 Auto Range Off
28 Auto Range Set

Input Coupling

29 AC
30 DC

Source Type

31 Source off
32 Random noise
33 Burst random
34 Periodic chirp
35 Burst chirp
36 Swept sine
37 Fixed sine

Sweep Direction

41 Up
42 Sweep hold
43 Manual sweep
44 Down

Sweep Mode

39 Linear sweep
40 Log sweep

External Sample Frequency Units

1 Hertz
2 RPM
20 Pulses/rev

Bandwidth Units

1 Hertz
2 RPM
3 Orders

Sweep Rate Units

11 Hertz/second
25 Seconds/decade
26 Seconds/octave

Auto Gain Reference Channel

0 Channel 1
1 Channel 2
2 Not used
3 No channel

Demod Channel Number

0 Channel 1
1 Channel 2
2 Both channels
3 No channels

Demod Type 1/2

45 AM
46 FM
47 PM

Source Level Units

8 dBV
9 Volts
13 Vrms

Source DC Offset Units

9 Volts

Trigger level units

9 Volts
33 EU channel 1
34 EU channel 2

Capt/Thru Session Length Units

4 Seconds
5 Revs
16 Points
17 Records

ASCII Format

To dump the state in ASCII, use the DSAS (Dump State in AScii) command; to load the state in ASCII, use LSAS (Load State in AScii). The format specifier is #I, and the two bytes (one word) following that indicate the number of variables to be transferred. For example, the following BASIC statements dump then load the instrument state:

```
OPTION BASE 1                        ! Set array base to 1
DIM State__buf (96)                  ! Create array for state
OUTPUT @Dsa; "DSAS"                  ! Dump ASCII command
ENTER @Dsa USING "2A,K"; F$,L        ! Read format & length
PRINT F$,L                           ! Verify format & length
ENTER @Dsa; State__buf(*)            ! Read state

OUTPUT @Dsa; "LSAS"                  ! Load ASCII command
OUTPUT @Dsa USING "2A,K"; F$,L       ! Output format & length
OUTPUT @Dsa; State__buf(*)           ! Output state
```

Because this is an ASCII transfer, the format specifier read into F$ is #I. The length word for dumping the state always indicates 96 elements in ASCII format.

ANSI Format

To dump the state in ANSI, use the DSAN (Dump State in ANsi) command; to load the state in ANSI, use LSAN (Load State in ANsi). The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump then load the instrument state:

```
OPTION BASE 1
DIM State__buf (96)                  ! Create array for state
OUTPUT @Dsa; "DSAN"                  ! Dump ANSI command
ENTER @Dsa USING "%,2A,W"; F$,L      ! Read format & length
PRINT F$,L                           ! Verify format & length
ASSIGN @Dsa; FORMAT OFF              ! Allow binary data
ENTER @Dsa; State__buf(*)            ! Read state
ASSIGN @Dsa; FORMAT ON               ! Allow ASCII data

OUTPUT @Dsa; "LSAN"                  ! Load ANSI command
OUTPUT @Dsa USING "#,2A,W"; F$,L     ! Output format & length
ASSIGN @Dsa; FORMAT OFF              ! Allow binary data
OUTPUT @Dsa; State__buf(*)           ! Output state
ASSIGN @Dsa; FORMAT ON               ! Allow ASCII data
```

Because this is an ANSI transfer, the format specifier read into F$ is #A. The length word always indicates 768 bytes for dumping the state in ANSI format, but the array was dimensioned for 96 values (768 ÷ 8).

Internal Binary Format

To dump the state in binary, use the DSBN (Dump State in internal BiNary) command; to load the state in binary, use LSBN (Load State in internal BiNary). The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump then load the instrument state:

```
OPTION BASE 1
INTEGER State__buf (142)                    ! Create array for state
OUTPUT @Dsa; "DSBN"                         ! Dump binary command
ENTER @Dsa USING "%,2A,W"; F$,L             ! Read format & length
PRINT F$,L                                  ! Verify format & length
ASSIGN @Dsa; FORMAT OFF                     ! Allow binary data
ENTER @Dsa; State__buf(*)                   ! Read state
ASSIGN @Dsa; FORMAT ON                      ! Allow ASCII data

OUTPUT @Dsa; "LSBN"                         ! Load binary command
OUTPUT @Dsa USING "#,2A,W"; F$,L            ! Output format & length
ASSIGN @Dsa; FORMAT OFF                     ! Allow binary data
OUTPUT @Dsa; State__buf(*)                  ! Output state
ASSIGN @Dsa; FORMAT ON                      ! Allow ASCII data
```

Because this is a binary transfer, the format specifier read into F$ is #A. The length word always indicates 284 bytes for dumping the state in binary format, but the array was dimensioned for 142 values (284 ÷ 2).

DUMPING THE COORDINATE TRANSFORM BLOCK

The coordinate transform block contains three groups of data: the display parameters, the data header for the active trace, and the displayed data in the active trace. As with other block transfers, this can be dumped in any of the three data formats. Note, however, the coordinate transform block can be dumped only; it has no load command. The coordinate transform data block contains exactly what you see on the display; if you have a phase trace active, the coordinate block contains phase data. The coordinate block is not as accurate as the data trace.

This section describes the contents of the coordinate transform block and the coordinate transform header. The last three parts in this section explain how to dump it in each of the three data formats.

Contents of the Coordinate Transform Block

Table 3-6 shows the organization of the data received after dumping the coordinate transform block:

Table 3-6 The Coordinate Transform Block

Part 1:	Coordinate transform header
Part 2:	Data header for active trace
Part 3:	Displayed trace data

Table 3-7 shows the contents of the coordinate transform header. For data types listed in the table as "E-type" (enumerated type) the value of that variable can be decoded by referring to table 3-8. The range of values for each E-type is shown in parentheses. The (+1) beside the byte count for strings is a reminder that each string is preceded by one length byte.

The index shown for each format indicates the position in the header that each item is located for each data format. The binary index indicates 16-bit words, and the ASCII/ANSI index indicates elements.

Table 3-7 The Coordinate Transform Header

Item	Data Type	Size (bytes)	Binary Index	ASCII/ANSI Index
Y coordinates	E-type (0-10)	2	1	1
# of disp elements	Integer	2	2	2
First element	Integer	2	3	3
Total elements	Integer	2	4	4
Display sampling	E-type (0-2)	2	5	5
Scaling	E-type (0-3)	2	6	6
Data Pointer	Long Integer	4	7	7
In Data	Long Integer	4	8	9
Log/Linear x-axis	Boolean	2	9	11
Sampled display data	Boolean	2	10	12
Plot/Graph mode	Boolean	2	11	13
Phase wrap	Boolean	2	12	14
Not used	Integers (18)	36	13	15
X scale factor	Real	4	31	33
Grid min Y scale	Real	4	33	34
Grid max Y scale	Real	4	35	35
/ Div	Real	4	37	36
Min value of data	Real	4	39	37
Max value of data	Real	4	41	38
Y cumulative Min	Real	4	43	39
Y cumulative Max	Real	4	45	40
Y scale factor[1]	Real	4	47	41
Not used	Reals (4)	16	49	42
Stop value	Long Real	8	57	46
Left grid	Long Real	8	63	47
Right grid	Long Real	8	67	48
Left data	Long Real	8	71	49
Right data	Long Real	8	75[2]	50

[1]Multiply by data to calibrate trace data
[2]Last word for binary is number 78.

Table 3-8 Enumerated (E-type) Values for Coordinate Transform Block

Y Coordinate	
1	Real
2	Imaginary
3	Linear magnitude
4	Log magnitude
5	dB
6	Nyquist
7	Not used
8	Phase
9	Nichols
10	dBm
Display Sampling	
0	not sampled (# of displayed elements = total elements)
1	half sampled (# of displayed elements = total elements/2)
2	sampled (# of displayed elements < total elements)
Scaling	
0	X and Y auto scale
1	X fixed scale, Y auto scale
2	X auto scale, Y fixed scale
3	X and Y fixed scale

ASCII Format

To dump the coordinate transform block in ASCII, use the DCAS (Dump Coordinate block in AScii) command. The format specifier is #I, and the two bytes (one word) following that indicate the number of variables to be transferred. For example, the following BASIC statements dump the coordinate transform block when a frequency response is displayed:

```
OPTION BASE 1
DIM Coord__buf (917)                  ! Create array for data
OUTPUT @Dsa; "DCAS"                   ! Dump ASCII command
ENTER @Dsa USING "2A,K"; F$,L         ! Read format & length
PRINT F$,L                            ! Verify format & length
ENTER @Dsa; Coord__buf(*)             ! Read headers & trace data
```

Because this is an ASCII transfer, the format specifier read into F$ is #I. For the specific case of a frequency response trace with full X scale, the length word read into L indicates 917 elements (50 in coordinate transform header, 66 in the data header, and 801 from the display). To make this a general program, you should redimension the array Coord__buf to L after reading L.

ANSI Format

To dump the coordinate transform block in ANSI, use the DCAN (Dump Coordinate block in ANSI) command. The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump the coordinate transform block when a frequency response is displayed:

```
OPTION BASE 1
DIM Coord__buf (917)                  ! Create array for data
OUTPUT @Dsa; "DCAN"                   ! Dump ANSI command
ENTER @Dsa USING "#,2A,W"; F$,L       ! Read format & length
PRINT F$,L                            ! Verify format & length
ASSIGN @Dsa; FORMAT OFF               ! Allow binary data
ENTER @Dsa; Coord__buf(*)             ! Read headers & trace data
ASSIGN @Dsa; FORMAT ON                ! Allow ASCII data
```

Because this is a binary transfer, the format specifier read into F$ is #A. For the specific case of a frequency response trace with full X scale, the length word read into L indicates 7336 bytes. This make the element count equal 917 (7336 ÷ 8), with 50 elements in the coordinate transform header, 66 in the data header, and 801 from the display. To make this a general program, you should redimension the array Coord__buf after reading L.

Internal Binary Format

To dump the coordinate transform block in internal binary, use the DCBN (Dump Coordinate block in Binary) command. The format specifier is #A, and the two bytes (one word) following that indicate the number of bytes to be transferred. For example, the following BASIC statements dump the coordinate transform block when a frequency response is displayed:

```
OPTION BASE 1
INTEGER C__hdr (78)                          ! Array for coord header
INTEGER D__hdr (84)                          ! Array for data header
INTEGER T__data (1602)                       ! Array for trace data
OUTPUT @Dsa; "DCBN"                          ! Dump binary command
ENTER @Dsa USING "#,2A,W"; F$,L              ! Read format & length
PRINT F$,L                                   ! Verify format & length
ASSIGN @Dsa; FORMAT OFF                      ! Allow binary data
ENTER @Dsa; C__hdr(*),D__hdr(*),T__data(*)   ! Read 3 parts of transfer
ASSIGN @Dsa; FORMAT ON                       ! Allow ASCII data
```

Because this is an binary transfer, the format specifier read into F$ is #A. For the specific case of a frequency response trace wit full X scale, the length word read into L indicates 3528 bytes: 156 in the coordinate header, 168 in the data header, and 3204 from the display. (Trace data are dumped in 32-bit floating point, so 4 bytes are required for each of the 801 displayed points.) To make this a general program, you should redimension the array T__data after reading L.

DUMPING/LOADING THE SYNTHESIS AND CURVE FIT TABLES

The synthesis and curve fit tables can be dumped and loaded in each of the three data formats. Transfer the table to pole-zero synth format first. For descriptions of these formats, please refer to "Three Data Formats" earlier in this chapter. The commands used are:

DTAN	Dump Table in ANsi
DTBN	Dump Table in internal BiNary
DTAS	Dump Table in AScii
LTAN	Load Table in ANsi
LTBN	Load Table in internal BiNary
LTAS	Load Table in AScii

Contents of the Synthesis and Curve Fit Tables

Table 3-9 shows the contents of the synthesis and curve fit tables dumped via HP-IB. For enumerated (E-type) values, refer to table 3-10.

Table 3-9 Contents of Synthesis & Curve Fit Tables

Item	Data Type	Size (bytes)	Binary Index	ASCII/ANSI Index
Table type	E-type (0-4)	2	1	1
Number in left side	Integer	2	2	2
Number in right side	Integer	2	3	3
Left side values[1]	Complex [1:22]	176	4	4
Right side values[1]	Complex [1:22]	176	92	48
Left constraints[2]	Boolean [1:22]	22	180	92
Right constraints[2]	Boolean [1:22]	22	202	114
Time delay	Real	4	224	136
Gain factor	Real	4	226	137
Scale frequency	Real	4	228	138
Current line	Integer	2	230	139
Current half	E-type (0-1)	2	231	140
Zero total	Integer	2	232	141
Pole total	Integer	2	233	142

[1]Each complex value is a pair of 32-bit floating point values representing a complex conjugate pair. These arrays are arranged as line 1 real, line 1 imaginary, line 2 real, etc. If a value is real-only, the imaginary part is zero.

[2]This is an array of 22 boolean elements, one flag for each line in the table. A 1 indicates that value in the table is constrained (user-created or fixed), and a 0 indicates that value is unconstrained.

Table 3-10 E-types in Synthesis & Curve Fit Tables

Table Type		Current Half	
0	Pole zero synth	0	Left
1	Pole residue synth	1	Right
2	Polynomial synth		
3	Constant trace		
4	Curve fit		

ACCESSING DISC FILES

This section explains the arrangement of HP 3562A disc files. These files may be stored data traces, time throughput sessions, and time capture buffers. The next section explains how to read trace files, and then throughput and capture files are discussed. All files on disc are stored in internal binary format.

Accessing Data Trace Files

Reading data trace disc files is very similar to dumping traces directly out of the analyzer. The only difference is that the data portion of the file (following the header) always starts on a sector boundary. Since the HP 3562A uses only 256-byte sectors, you simply need to ignore the bytes between the end of the header and byte #256, the beginning of the data.

The data header is 168 bytes long, so ignore bytes 169 through 255 (words 85 through 127). Please refer to "Dumping/Loading Data Traces" earlier in this chapter for the rest of the information you need. Note that data trace files are scaled and do not need to be multiplied by a calibration factor when read off disc.

Accessing Throughput and Capture Files

There are three types of disc storage for throughput and capture files: one-channel, two-channel without delay, and two-channel with delay. Capture files are treated as one-channel throughput files. "Delay" in this case indicates differential delay between the two channels, not the delay at the beginning of the session. The following sections explain:

1. How data records are arranged for each of the three storage types

2. How to handle skipped tracks

3. How to scale data

4. How to use the calibration table

5. How to interpret the throughput/capture header.

Data Record Arrangement

Throughput/capture files are composed of the throughput/capture header followed by data. The header is composed of one BDAT file sector followed by three sectors of HP 3562A header information. Throughput and capture data are arranged by time record. Records are composed of 2048 16-bit integer words (4096 bytes).

Record arrangement depends on the number of channels and whether no cross-channel trigger delay was used to start the session (throughput only). In capture and single-channel throughput files, records are arranged sequentially started with the first record stored, as shown in figure 3-1. The header contains one BDAT file sector and three sectors of header information. Because these three sectors contain 768 bytes (256 x 3) and the header is only 648 bytes long, there are 120 unused bytes at the end of the third sector (the fourth sector counting the BDAT sector).

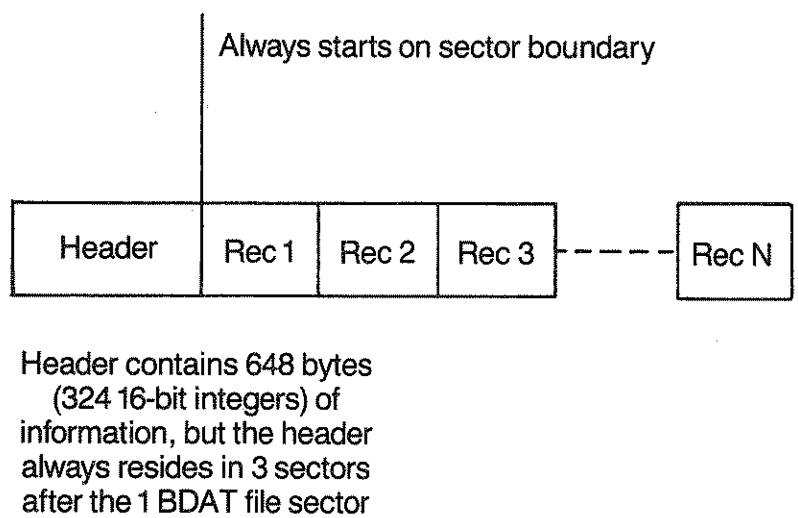


Figure 3-1 Disc Storage of One-Channel Files

For two-channel throughput with no trigger delay, records alternate for each channel (1st record on Channel 1, 1st record on Channel 2, 2nd record on Channel 1, etc.). Figures 3-3a, b and c show an example of a throughput with 5 records on each channel and a 2.5 record delay on Channel 2. Once again, the header contains one BDAT file sector and three sectors of header information. Because these three sectors contain 768 bytes (256 x 3) and the header is only 648 bytes long, there are 120 unused bytes at the end of the third sector (the fourth sector counting the BDAT sector).

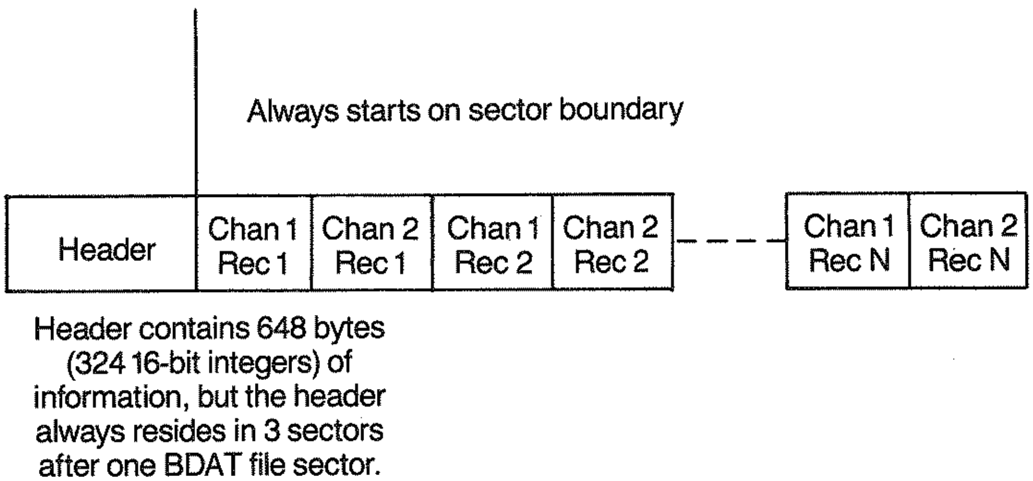


Figure 3-2 Disc Storage of Two-Channel Files without Delay

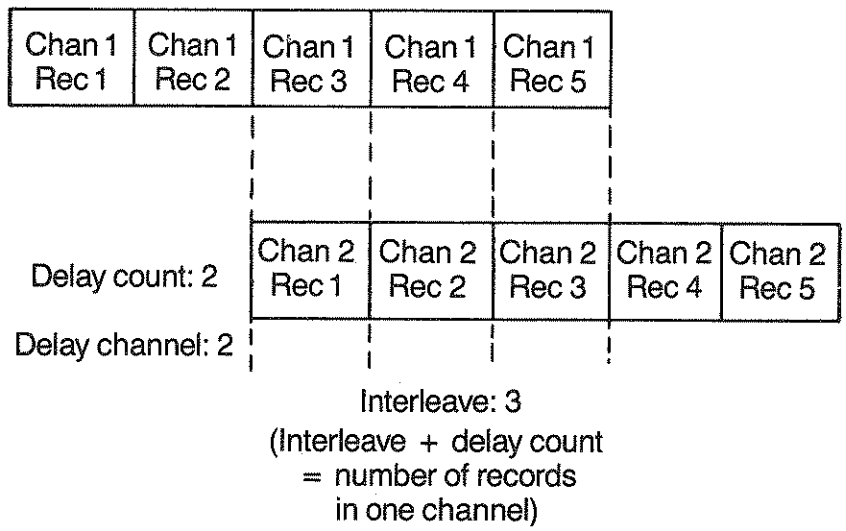
For two-channel throughputs with trigger delay, records are interleaved according to the amount of differential delay between the two channels, as shown in figure 3-3. Once again, the header contains one BDAT file sector and three sectors of header information. Because these three sectors contain 768 bytes (256 x 3) and the header is only 648 bytes long, there are 120 unused bytes at the end of the third sector (the fourth sector if you count the BDAT sector).

Figure 3-3a shows how what the interleave, delay count and delay channel indicator in the throughput/capture header mean. Interleave is the number of pairs of Channel 1/Channel 2 records between the Channel 1 records and the Channel 2 records. Delay count is the number of whole records of delay between the two channels. In this example, the delay is 2.5 records, but just the 2 records are indicated by the delay count variable. (The remaining partial record is explained in figure 3-3b.) The delay channel just indicates which channel is delayed past the other, Channel 2 in this example.

Figure 3-3b shows how the remaining 1/2 record delay is handled. The partial record count shows the number of data points in the remaining partial delay record. If the data are real-only (baseband), the number of data points equals the number of words in the record. If the data are complex (zoom), the number of data points is 1/2 the number of words. Figure 3-3b also shows where the valid data records actually reside in relation to the records created by the disc. Remember that in this example, the delay is 2.5 records, and interpret your data file according to the delay you actually have.

Figure 3-3c shows how the records are actually arranged on the disc and how you need to re-assemble them to get valid records for the delayed channel. In this example, the first half of Chan 2 Rec 1 and the last half of Chan 2 Rec 6 contains irrelevant data. Note that the partial record count shows both the number of invalid data points at the beginning of Rec 1 and the number of valid data points at the beginning of Rec 6.

3-3a Interleave, Delay Count and Delay Channel



3-3b Partial Record and Location of Valid Data Records

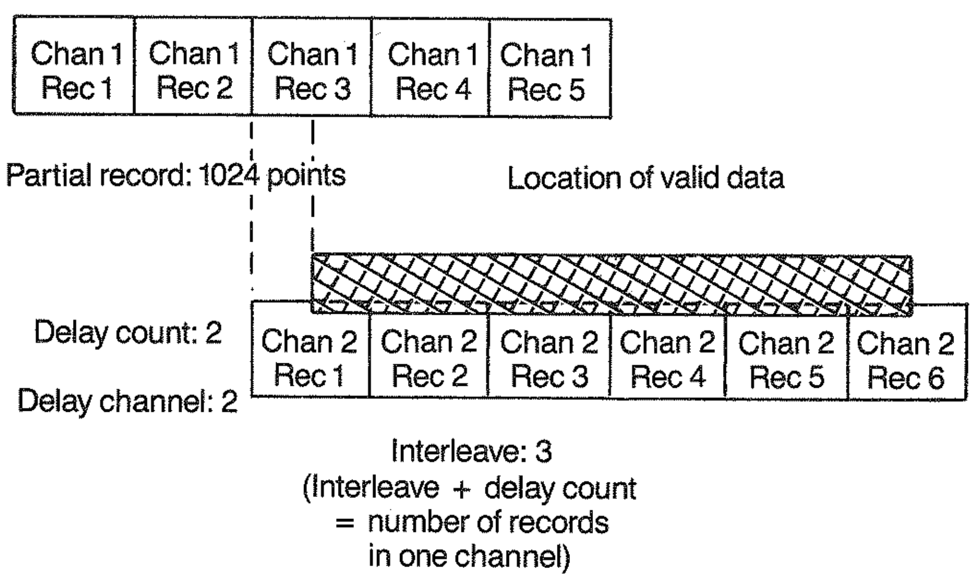


Figure 3-3 Disc Storage of Two-Channel Files with Delay

3-3c Arrangement of Delayed Records on Disc and Re-assembly

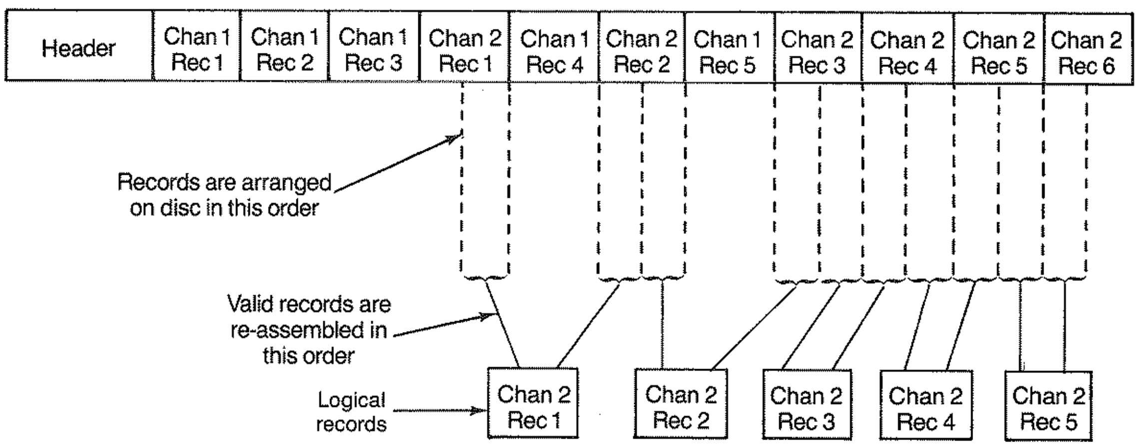


Figure 3-3 Disc Storage of Two-Channel Files with Delay cont.

Skipped Tracks

When the HP 3562A throughputs to Hewlett-Packard Command Set/80 (CS/80) disc drives, it skips over tracks which have been previously spared. (Refer to Chapter 11 in the HP 3562A Operating Manual for information on sparing tracks.) Before reading data from a CS/80 disc file, you should read the number of skipped tracks indicator in the header and see if there are any spared tracks in the file area. If there are, you need to pass over these areas as you read the data.

The skipped track offset table shows the location of up to 9 spared tracks. These are address offsets from the beginning of the entire file, not absolute addresses. You can use the sectors/track indicator to determine where in the data file the good disc area resumes. Remember that the HP 3562A always uses 256 byte sectors and there are 2048 data points per record (each point is one word, so there are 4096 bytes required to store one record). Consequently, each record requires 16 sectors (4096/256) of disc area.

Data Scaling

Data points read out of a disc file must be scaled to obtain calibrated values. Here is the formula to scale data:

Scaled data = (−4/3)(disc data)(range)(32,768)/26028.55

where: disc data is the data portion of the file
range is the range setting for
that channel

Calibration Tables

Two calibration tables are stored in the header. Cal table #1 is used for Channel 1, and table #2 is used for Channel 2/Channel 1. The curves should be reconstructed over the desired frequency range using linear interpolation.

Each table is composed of 56 complex values. Each complex value is composed of two 16-bit integers representing a real/imaginary pair. The span from 0 to 90 kHz is covered in 2 kHz steps; the span from 91 to 100 kHz is covered in 1 kHz steps. The two cal tables are scaled by the "Mag cal scale factors" 1 and 2, respectively.

The Throughput/Capture Header

Table 3-11 shows the throughput/capture header. Refer to table 3-12 for enumerated (E-type) values.

Table 3-11 Throughput/Capture Header

Item	Data Type	Size (bytes)	Binary Index
Complex data flag	Boolean (1 = yes)	2	1
Bytes per point	Integer	2	2
Points per record	Integer	2	3
Channel type	E-type (0-1)	2	4
Bandwidth units	E-type (1-3)	2	5
X units	E-type (0-35)	2	6
Delay channel[1]	E-type (0-3)	2	7
Delay count[1]	Integer	2	8
Partial record[1]	Integer	2	9
Interleave[1]	Integer	2	10
# of realtime records	Integer	2	11
Sectors/track	Integer	2	12
Skip track offsets[2]	Long Integers (9)	36	13
Digit Revision	Integer	2	31
Not used	Integer	2	32
# of skip tracks	Integer	2	33
Cal failure	Boolean (1 = yes)	2	34
Start frequency	Long Real	8	35
Center frequency	Long Real	8	39
Frequency span	Real	4	43
Δt	Real	4	45
Mag cal cspc scale factor	Real	4	47
Mag cal fr sp scale factor	Real	4	49
Digitized pt len 1	Long Integer	4	51
Range units 1	E-type (0-35)	2	53
Trig delay 1	Long Integer	4	54
Coupling 1	E-type (29-30)	2	56
Input float 1	Boolean (1 = float)	2	57
Overflow status 1	Boolean (1 = overrange)	2	58
EU Label 1	String	5 (+1)	59
Range 1	Real	4	62
Delay 1	Real	4	64

Table 3-11 Throughput/Capture Header cont.

Item	Data Type	Size (bytes)	Binary Index
EU value 1	Real	4	66
Digitized pt len 2	Long Integer	4	68
Range units 2	E-type (0-35)	2	70
Trig delay 2	Long Integer	4	71
Coupling 2	E-type (29-30)	2	73
Input float 2	Boolean (1 = float)	2	74
Overflow status 2	Boolean (1 = overrange)	2	75
EU Label 2	String	5	76
Range 2	Real	4	79
Delay 2	Real	4	81
EU value 2	Real	4	83
Cal table 1	Int array [2,56]	224	85
Cal table 2	Int array [2,56]	224	197
Sec att corr 1[3]	Complex (2 reals)	8	309
Sec att corr 2[3]	Complex (2 reals)	8	313
Trigger phase corr[5]	Long Integer	4	317
Trigger path delay[5]	Real	4	319
Dig filter word 1	Integer	2	321[6]
Dig filter word 2	Integer	2	323[6]
Not used	Integers (2)	4	323[4]

[1]Relevant only in two channel throughputs; refer to "Data Record Arrangement" earlier in this chapter.
[2]Table contains 10 address offsets.
[3]This is the correction factor at 100 kHz for the secondary attenuators.
[4]Last word in header is 324.
[5]These two variables are not used to calibrate throughput data, but they are available for your information.
[6]Valid only if Digit Revision ≥ 1.

Table 3-12 E-types in Throughput/Capture Header

Channel Type and Delay Channel

0	Channel 1
1	Channel 2
2	Both channels
3	No channels

Bandwidth Units and X Units

0	Null
1	Hz
2	RPM
3	Orders
4	Seconds
5	Revs
6	Degrees
7	dB
8	dBV
9	Volts
10	Volts/Hz
11	Hz/second
12	Volts/EU
13	Volts rms
14	Volts²/Hz
15	Percent
16	Points
17	Records
18	Ohms
19	Hz/octave
20	Pulses/rev
21	Decades
22	Minutes
23	Oct/minute
24	Octaves
25	Sec/decade
26	Sec/octave
27	Hz/point
28	Points/sweep
29	Points/decade
30	Points/octave
31	V/Vrms
32	Volts²
33	Channel 1 EU
34	Channel 2 EU
35	EU

Range Units 1/2

8	dBV
9	Volts
13	Volts rms
35	EU

Coupling 1/2

29	AC
30	DC

SIGNAL PROCESSING GROUP

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to explain the use of the signal processing command group. These commands allow you to set up data blocks in the HP 3562A's memory then perform a number of signal processing operations on these blocks. The topics covered here are:

1. Overview of signal processing steps
2. General block operations
3. Transferring blocks
4. Math operations
5. Averaging operations
6. Measurement operations
7. Plotting and graphing results

NOTE 1

Most of the signal processing operations described in this chapter can be performed using waveform math. Please refer to chapter 9 in the operating manual, which describes the math operations, before using the primitives in this chapter. If waveform math can meet your needs, it presents a much simpler programming task than the signal processing primitives.

NOTE 2

The HP 3562A must be paused before you use signal processing primitives (the data blocks will be erased otherwise).

OVERVIEW OF SIGNAL PROCESSING STEPS

There are five general steps to perform signal processing primitive operations in the HP 3562A:

1. Set up primitive blocks
2. Input data
3. Perform operations
4. Output results
5. Display results

The first step, setting up the blocks you need, is covered in the next section, "General Block Operations." For step two, you have two choices for input data: digital input via HP-IB or analog signals from the input channels. Transferring primitive data blocks via HP-IB is covered in "Transferring Blocks" later in this chapter, and analog input is covered in "General Block Operations" (the ANIN command). For step three, choose the desired operation from "Math Operations," "Averaging Operations" or "Measurement Operations" later in this chapter. To output results via HP-IB, refer to "Transferring Blocks." Finally, if you want to display result on the analyzer's screen, refer to "Plotting and Graphing Results" at the end of this chapter.

PARTIAL MEMORY MAP

The following memory map shows the location of important data blocks. Signal processing blocks start overlaying RAM at "TRACE A DATA."

Partial Memory Map
(Typical linear resolution state)

TRACE A DATA	(4k)
TRACE B DATA	(4k)
DATA ON CHANNEL 1 (integer)	(4k)
DATA ON CHANNEL 2 (integer)	(4k)
FFT OUTPUT—CHANNEL 1 (integer)	(2k)
FFT OUTPUT—CHANNEL 2 (integer)	(2k)
MEASUREMENT WORKING BLOCK—CHANNEL 1	(4k)
MEASUREMENT WORKING BLOCK—CHANNEL 2	(4k)
MEASUREMENT AVERAGING BLOCKS	(8k)

GENERAL BLOCK OPERATIONS

This section describes the commands used to create and handle data blocks. You should familiarize yourself with the commands in this section before attempting to use the other commands in this chapter. The commands covered in this section are:

Block size	(BLSZ)
Point count	(PTCT)
Float block	(FLTB)
Unfloat block	(UFLB)
Move block	(MOVB)
Move complex constant	(MOVX)
Move real constant	(MOVC)
Partial block clear	(PCLR)
Analog input	(ANIN)

Block Size (BLSZ)

The block size command (BLSZ) allocates memory for signal processing operations by creating individual blocks. When creating blocks, you specify the size of the block(s), the number of the first block, and how many blocks you want to create. The syntax of BLSZ is:

BLSZs,n1,[n]

where s is the size of the block(s) in words
n1 is the number of the first block
n is the number of blocks to be created (optional)

There are approximately 37.9 kwords of RAM reserved for signal processing blocks. Individual block size is limited to 32kwords. If you want to create multiple blocks, make sure that their combined size does not exceed 37.9 kwords. The number of the first block, n1, must be between 0 and 15, inclusive. The number of blocks to be created, n, can be from 1 to 16, provided that the combination of n1 and n used does not attempt to create a block numbered higher than 15. Note that n is optional; if you do not specify it, one block is created.

As an example, the BASIC statement:

```
OUTPUT 720; "BLSZ100,0,2
```

creates 2 100-word blocks, numbered 0 and 1

If more than one block exists in memory, changing the of lower-numbered blocks affects higher-numbered blocks. For example, if you have 10 500-word blocks, then recreate block 1 only at 400 words, blocks 2 through 10 will be shifted down 100 words. This can effectively erase data, so recreate blocks carefully.

If you intend to use the FFT operations, some constraints apply to primitive blocks. First, all blocks used in FFT operations must be 2048 words long (2 kwords). Second all blocks used for FFTs must reside on 2 kwords boundaries (i.e., the size of all lower-numbered blocks must be a multiple of 2kwords).

Point Count (PTCT)

The point count command (PTCT) allows you to specify a portion of an existing block for use in subsequent operations. Its syntax is:

PTCTn,p

where n is the block
p is the number of points

The block number, n, must be between 0 and 15 and must represent an active block. The number of points, p, specifies that the first p points (words) in block n will be used. Of course, p cannot be greater than the size of the block. As an example, the BASIC statement:

OUTPUT 720; "PTCT1,50"

specifies that the first 50 points in block 1 will be used any time in the future that block is used. To respecify the point count for a block, send PTCT again.

Float Block (FLTB)

The float block command (FLTB) is used to convert integer data to floating point format. It syntax is:

FLTBn1,n2 [,count]

where n1 is the integer block (source)
n2 is the destination block
count is point count (optional)

FLTB floats n1 and puts the result in n2. Of course, n1 and n2 must be valid block numbers.

Unfloat Block (UFLB)

The unfloat block command (UFLB) is used to convert floating point data to integer format. Its syntax is:

UFLBn1,n2 [,count]

where n1 is the floating point block (source)
n2 is the destination
count is point count (optional)

UFLB unfloats n1 and puts the result in n2. Of course, n1 and n2 must be valid block numbers.

Move Block (MOVB)

The move block command (MOVB) is used to move the contents of one block into another block. Its syntax is:

MOVBn1,n2,p [,count]

where n1 is the source block
n2 is the destination block
count is the number of points to be moved (optional)

The block numbers n1 and n2 specify the source and destination, respectively. The point count, p, specifies how many points (words) from n1 are to be moved into n2. If p is not specified, all of n1 is moved. As an example, the BASIC statement:

OUTPUT 720; "MOVB9,3,50

moves the first 50 points of block 9 into block 3.

Move Complex Constant (MOVX)

The move complex constant command (MOVX) moves a complex constant into a complex block. Its syntax is:

MOVXn1,n2,n3[,count]

where n1 is the real part of source constant
n2 is the imaginary part of source constant
n3 is the complex destination block
count is destination point count (optional)

The block number n3 must represent a valid block. As an example, the BASIC statement:

OUTPUT 720; "MOVX1,2,3

moves blocks 1 and 2 into block 3. Block 3 now = (4,7,5,8,6,9).

Move Real Constant (MOVC)

Moves a real constant into a block. Its syntax is as follows:

MOVCn1,n2,[,count]

where n1 is real source constant
n2 is real destination block
count is destination point count (optional)

Partial Block Clear (PCLR)

The partial block clear command (PCLR) allows you to clear points at the beginning of a block. Its syntax is:

PCLRn1,p

where n1 is the block to be partially cleared
p is the number of points to be cleared

For example, the BASIC statement:

OUTPUT 720; "PCLR1,5"

clears the first 5 points in block 1.

Analog Input (ANIN)

The analog input command (ANIN) allows you to take data from the input channels for use in signal processing primitives. Its syntax is:

ANINn1,n2,c1,c2

where n1 is the destination block for Channel 1
n2 is the destination block for Channel 2
c1 is the number of points to take on Ch 1
c2 is the number of points to take on Ch 2

For example, the BASIC statement:

OUTPUT 720; "ANIN1,2,1024,1024"

inputs 1024-point blocks on both channels into primitive blocks 1 (Channel 1) and 2 (Channel 2). If no data is wanted on a channel, set the number of points for that channel to zero. This command was the current triggering setup. See "Accessing Throughput and Capture Files" in Chapter 3 for information on scaling the data.

TRANSFERRING BLOCKS

This section explains how to transfer signal processing between the controller and the HP 3562A. Each dump or load requires two steps: first identify the block to be transferred, then send the dump or load command specifying the data format. The topics covered in this section are:

1. The primitive block header
2. The block pointer (PBLK)
3. Dumping blocks in ASCII (DBAS)
   in ANSI (DBAN)
   in internal binary (DBBN)
4. Loading blocks in ASCII (LBAS)
   in ANSI (LBAN
   in internal binary (LBBN)

The Primitive Block Header

Every primitive block has a 3-word header located at the beginning of the block. These 3 word are transparent to any size specifications. If you dump a block, make sure to allow for the 3 non-data words at the beginning.

Table 4-1 shows the primitive block header. Note that the header has this format regardless of the data format of the block.

Table 4-1 Primitive Block Header

Word	Description	Range
1	Block type	0 = real floating point 1 = complex floating point 2 = real integer 3 = complex integer
2	Block exponent	see text
3	Point count	equal to PTCT value

The value of word 1 depends on the data format in which the block was filled and any subsequent operations performed on it. The block exponent value in word 2 is used to calculate amplitude values for real integer and complex integer data blocks (types 2 and 3). The equation is:

$$\text{amplitude} = \text{value}(2^{\text{block exponent}})$$

Finally, the value of word 3, the point count, is equal to point count specified for the block. If you have previously specified this with the PTCT command, word 3 will be equal to the value of PTCT. If you have not used PTCT on this block, word 3 is equal to the dynamic length of the block in points.

Primitive Block Pointer (PBLK)

The primitive block pointer command (PBLK) specifies the active block for dumping and loading. Its syntax is:

PBLKn

where n is the number of the block

The number of the block, n, must be between 0 and 15 and must represent an existing block.

Dumping Blocks

Primitive data blocks can be dumped in each of the three data formats (refer to Chapter 3 for descriptions of data types). When the HP 3562A receives the dump command, it outputs six elements:

elements 1-2: #I or #A (to specify format)

element 3: length variable

elements 4-6: header (described earlier in this section)

The length variable differs from the point count in the header in that the length variable includes the three header elements, while the point count does not.

Dump Block in ASCII (DBAS)

The dump block in ASCII command (DBAS) dumps a block (specified by the block pointer) in ASCII format. (For a description of the ASCII format, plese refer to Chapter 3.) The format specifier is #I, and the length word specifies the number of elements to be transferred.

Dump Block in ANSI (DBAN)

The dump block in ANSI format (DBAN) dumps a block (specified by the block pointer) in ANSI floating point format. (For a description of this format, please refer to Chapter 3.) The format specifier is #A, and the length word specifies the number of bytes to be transferred. Only blocks shorter than 32,768 bytes (including header) can be transferred this way.

When using the ANSI transfer, remember that these are 8-byte floating point values. Also, if your computer has an ASCII formatter, you need to disable it for ANSI transfers.

Dump Block in Internal Binary (DBBN)

The dump block in internal binary (DBBN) dumps a block (specified by the block pointer) in the internal 32-bit floating point format. (For a description of this format, please refer to Chapter 3.) The format specifier is #A, and the length word specifies the number of bytes to be transferred. Only blocks shorter than 32,768 bytes (including header) can be transferred this way.

When using the binary transfer, remember that these are 32-bit floating point values. Also, if your computer has an ASCII formatter, you need to disable it for binary transfers.

Loading Blocks

When primitive data blocks are loaded into the HP 3562A, it expects the following six elements:

elements 1-2:	#I or #A to specify format
elements 3:	length variable
elements 4-6:	header (described earlier in this section)

Load Block in ASCII (LBAS)

The load block in ASCII command (LBAS) loads a block (specified by the block pointer) in ASCII format. (For a description of the ASCII format, plese refer to Chapter 3.) The format specifier is #I, and the length word specifies the number of elements to be transferred.

Load Block in ANSI (LBAN)

The load block in ANSI format (LBAN) loads a block (specified by the block pointer) in ANSI floating point format. (For a description of this format, please refer to Chapter 3.) The format specifier is #A, and the length word specifies the number of bytes to be transferred. Only blocks shorter than 32,768 bytes (including header) can be transferred this way.

When using the ANSI transfer, remember that these are 8-byte floating point values. Also, if your computer has an ASCII formatter, you need to disable it for ANSI transfers.

Load Block in Internal Binary (LBBN)

The load block in internal binary (LBBN) loads a block (specified by the block pointer) in the internal 32-bit floating point format. (For a description of this format, please refer to Chapter 3.) The format specifier is #A, and the length word specifies the number of bytes to be transferred. Only blocks shorter than 32,768 bytes (including header) can be transferred this way.

When using the binary transfer, remember that these are 32-bit floating point values. Also, if your computer has an ASCII formatter, you need to disable it for binary transfers.

MATH OPERATIONS

The commands in this section perform math operations on data blocks. If you have not yet created and filled the blocks needed for your math operation, refer to the previous section, "General Block Operations." The commands covered in this section are:

Add blocks	(ADDB)
Add complex constant	(ADDX)
Add real constant	(ADDC)
Subtract blocks	(SUBB)
Subtract complex constant	(SUBX)
Subtract real constant	(SUBC)
Multiply blocks	(MPYB)
Multipy by complex constant	(MPYX)
Multiply by real constant	(MPYC)
Multiply by jω	(MPJW)
Multiply by self conjugate	(MPSC)
Multiply by magnitude squared	(MPMG)
Divide by block	(DIVB)
Divide by complex constant	(DIVX)
Divide by real constant	(DIVC)
Divide by jω	(DVJW)
Divide imaginary part	(DIVI)
Divide real part	(DIVR)
Divide into real constant	(DVIR)
Negate block	(NEGB)
Conjugate block	(CNJB)
Differentiate block	(DIFB)
Integrate block	(INGB)
Power spectrum summation	(PSPS)
Cross spectrum summation	(CSPS)

Add Blocks (ADDB)

The add blocks command (ADDB) allows you to add two data blocks. Its syntax is:

ADDBn1,n2[,n3]

where n1 is the first addend
n2 is the second addend
n3 is the destination of the result (optional)

ADDB adds n1 to n2 and puts the result in n3. n3 is an optional parameter; if it is not specified, the result in put in n2.

Add Complex Constant to Block (ADDX)

The add complex constant command (ADDX) allows you to add a complex constant to a complex block. Its syntax is:

ADDXn1,n2,n3[,n4]

where n1 is the real part of the source constant
n2 is the imaginary part of the source constant
n3 is the complex second addend block
n4 is the optional destination for the result

If n4 is not specified, the result is put in n3. As an example, the BASIC statement:

OUTPUT 720; "ADDX1,2,3,4"

adds 1+, 2 to 3 and puts the result in block 4.

Add Real Constant to Block (ADDC)

The add real constant to block command (ADDC) adds a real constant to the contents of a second block. Its syntax is:

ADDCn1,n2[,n3]

where n1 is the source constant
n2 is the real second addend block
n3 is the optional destination for the result

If n3 is not specified, the result is put in n2.

Subtract Blocks (SUBB)

The subtract block command (SUBB) allows you to subtract one block from another. Its syntax is:

SUBBn1,n2[,n3]

where n1 is the minuend
n2 is the subtrahend
n3 is the optional destination block

SUBB subtracts n2 from n1 and puts the result in n3. If n3 is not specified, the result is put in n2.

Subtract Complex Block From Complex Constant (SUBX)

The subtract complex constant command (SUBX) allows you to subtract a complex block from a complex constant. Its syntax is:

SUBXn1,n2,n3[,n4]

where n1 is the real part of the minuend
n2 is the imaginary part of the minuend
n3 is the complex subtrahend
n4 is the optional destination for the result

If n4 is not specified, the result is put in n3. As an example, the BASIC statement:

OUTPUT 720; "SUBX1,2,3,4"

subtracts block 3 from 1 + , 2 and puts the result in 4.

Subtract Real Constant from Block (SUBC)

The subtract real constant from block command (SUBC) subtracts a block from a real constant. Its syntax is:

SUBCn1,n2[,n3]

where n1 is the constant minuend
n2 is the subtrahend block
n3 is the optional destination for the result

If n3 is not specified, the result is put in n2.

Multiply Blocks (MPYB)

The multiply blocks command (MPYB) allows you to multiply two blocks. It syntax is:

MPYBn1,n2[,n3]

where n1 is the first factor
n2 is the second factor
n3 is the optional destination for the result

MPYB multiplies n1 by n2 and puts the result in n3. n3 is an optional parameter; if it is not specified, the result is put in n2.

Multiply Block by Complex Constant (MPYX)

The multiply complex constant command (MPYX) allows you to multiply a complex constant by a complex block. Its syntax is:

MPYXn1,n2,n3[,n4]

where n1 is the real part of the source constant
n2 is the imaginary part of the source constant
n3 is the complex block
n4 is the optional destination for the result

If n4 is not specified, the result is put in n3. As an example, assume block 1 = (1,3,5), block 2 = (2,4,6) and block 3 = (1,2,3,4,5,6). The BASIC statement:

```
OUTPUT 720; "MPYX1,2,3,4"
```

multiplies 1 and 2 by 3 and puts the result in block 4.

Multiply Block by Real Constant (MPYC)

The multiply real constant block command (MPYC) multiplies a real constant by a real block. Its syntax is:

MPYCn1,n2[,n3]

where n1 is the source constant
n2 is the real block
n3 is the optional destination for the result

If n3 is not specified, the result is put in n2.

Multiply Block by jω (MPJW)

The multiply by jω command (MPJW) command allows you to multiply a block by jω to perform artificial differentiation. Its syntax is:

MPJWω^{start},Δω,n1[,n2]

where ω^{start} is the starting value of ω
Δω is the ω increment
n1 is the block to be differentiated
n2 is the optional destination block for the result

Multiply Block by Self Conjugate (MPSC)

The multiply by self-conjugate command (MPSC) allows you to multiply a complex block by its complex conjugate. Its syntax is:

MPSCn1[,n2]

where n1 is the complex block
n2 is the optional destination for the result

Multiply Block by Magnitude Squared (MPMG)

The multiply by magnitude squared command allows you to multiply a real block by the magnitude squared of a complex block. Its syntax is:

MPMGn1,n2[,n3]

where n1 is the real block
n2 is the complex block
n3 is the optional destination of the result

If n3 is not specified, the result is put in n1.

Divide Block by Block (DIVB)

The divide block command (DIVB) allows you to divide one block by another. It syntax is:

DIVBn1,n2[,n3]

where n1 is the dividend
n2 is the divisor
n3 is the optional destination for the result

DIVB divides n1 by n2 and puts the result in n3. n3 is an optional parameter; if it is not specified, the result is put in n2.

Divide Block by Complex Constant (DIVX)

The divide block by complex constant command (DIVX) allows you to divide a block by a complex constant. Its syntax is:

DIVXn1,n2,n3[,n4]

where n1 is the real part of the divisor
n2 is the imaginary part of the divisor
n3 is the complex dividend block
n4 is the optional destination for the result

If n4 is not specified, the result is put in n3. As an example, the BASIC statement:

OUTPUT 720; "DIVX1,2,3,4"

divides block 3 by 1 + , 2 and puts the result in block 4.

Divide Block by Real Constant (DIVC)

The divide block by real constant command (DIVC) divides a block by a real constant. Its syntax is:

DIVCn1,n2[,n3]

where n1 is the constant divisor
n2 is the dividend block
n3 is the optional destination for the result

If n3 is not specified, the result is put in n2.

Divide Block by jω (DVJW)

The divide by jω command (DVJW) command allows you to divide a block by jω to perform artificial integration. Its syntax is:

DVJWω_{start},$\Delta\omega$,n1[,n2]

where ω_{start} is the starting value of ω
$\Delta\omega$ is the ω increment
n1 is the block to be integrated
n2 is the optional destination block for the result

If n2 is not specified, the result is put in n1.

Divide Imaginary Part of Block (DIVI)

The divide imaginary part of block command (DIVI) allows you to divide the imaginary part of a complex block by a real constant. Its syntax is:

DIVIn1,n2[,n3]

where n1 is the complex block
n2 contains the real value
n3 is the optional destination for the result

If n3 is not specified, the result is put in n1.

Divide Real Part of Block (DIVR)

The divide real part of block command (DIVR) allows you to divide the real part of a complex block by a real constant. Its syntax is:

DIVRn1,n2[,n3]

where n1 is the complex block
n2 contains the real value
n3 is the optional destination for the result

If n3 is not specified, the result is put in n1.

Divide Block into Real Constant (DVIC)

The divide block into real constant command (DVIC) allows you to divide a real block into a real constant. Its syntax is:

DVICn1,n2[,n3]

where n1 is the real divisor block
n2 is the real dividend constant
n3 is the optional destination for the result

If n3 is not specified, the result is put in n1.

Negate Block (NEGB)

The negate block allows you to negate the contents of a block. Its syntax is:

NEGBn1[,n2]

where n1 is the block to be negated
n2 is the optional destination for the result

If n2 is not specified, the result is put in n1.

Conjugate Block (CNJB)

The conjugate block command (CNJB) computes the complex conjugate of a data block. Its syntax is:

CNJBn1[,n2]

where n1 is the block to be conjugated
n2 is the optional destination of the result

If n2 is not specified, the result is put in n1.

Differentiate Block (DIFB)

The differentiate block comand (DIFB) computes the differential of a data block. Its syntax is:

DIFBn1[,n2]

where n1 is the block to be differentiated
n2 is the optional destination for the result

If n2 is not specified, the result is put in n1.

Integrate Block (INGB)

The integrate block comand (INGB) computes the integral of a data block. Its syntax is:

INGBn1[,n2]

where n1 is the block to be integrated
n2 is the optional destination for the result

If n2 is not specified, the result is put in n1.

Power Spectrum Summation (PSPS)

The power spectrum summation command (PSPS) computes the power spectrum of a complex floating point block and sums it with the contents of a second block. Its syntax is:

PSPSn1,n2

where n1 is the block to be summed
n2 is contains the cumulative result

Cross Spectrum Summation (CSPS)

The cross spectrum summation command (CSPS) computes the cross spectrum of two complex floating point blocks and sums the result with the contents of a third block. Its syntax is:

CSPSn1,n2,n3

where n1 is the first complex block
n2 is the second complex block
n3 contains the cumulative result

AVERAGING OPERATIONS

The HP 3562A offers the following averaging primitives:

Exponential averaging	(XAVG)
Power spectrum exponential averaging	(PXAV)
Cross spectrum exponential averaging	(CXAV)
Peak hold	(PKHD)
Power spectrum peak hold	(PPEK)
Cross spectrum peak hold	(CPEK)

Exponential Averaging (XAVG)

The exponential average command (XAVG) averages data blocks using an exponentially weighted averaging formula. Its syntax is:

XAVGn1,n2,awf

where n1 is the block to be averaged
n2 is the cumulative average
awf is the exponential weighting factor

The weighting factor, awf, is interpreted as a power of 2. The formula used in exponential averaging is:

$$A_n = (1-2^{-n})a_n + 2^{-n}D_n$$

where A_n is cumulative average (in n2)
D_n is new block (in n1)
awf is exponential weighting factor

Power Spectrum Exponential Averaging (PXAV)

The power spectrum exponential averaging command (PXAV) computes the power spectrum from a complex block then exponentially averages that with a cumulative average in another block. Its syntax is:

PXAVn1,n2,awf

where n1 is the complex data block to be averaged
n2 is the cumulative average
awf is the exponential weighting factor

Refer to the exponential averaging command (XAVG) for the formula used.

Cross Spectrum Exponential Averaging (CXAV)

The cross spectrum exponential averaging command (CXAV) computes the cross spectrum of two complex blocks then exponentially averages that with a cumulative average in another block. Its syntax is:

CXAVn1,n2,n3,awf

where n1 is the first complex block
n2 is the second complex block
n3 is the cumulative average
awf is the exponential weighting factor

Refer to the exponential averaging command (XAVG) for the formula used.

Peak Hold (PKHD)

The peak hold command (PKHD) compares the magnitudes of two blocks on a point-to-point basis and holds the larger values. Its syntax is:

PXHDn1,n2

where n1 is the new block
n2 contains the peak values

Power Spectrum Peak Hold (PPEK)

The power spectrum peak hold command (PPEK) computes the power spectrum of a complex block then compares its magnitudes to a second power spectrum block and holds the larger values. Its syntax is:

PPEKn1,n2

where n1 is the new complex block
n2 contains the peak values

Cross Spectrum Peak Hold (CPEK)

The cross spectrum peak hold command (CPEK) computes the cross spectrum of two complex blocks then compares those magnitudes to a third cross spectrum block and holds the larger values. Its syntax is:

CPEKn1,n2,n3

where n1 is the first complex block
n2 is the second complex block
n3 contains the peak values

MEASUREMENT OPERATIONS

The HP 3562A offers the following measurement primitives:

Histogram	(HST)
Real FFT	(RFFT)
Complex FFT	(CFFT)
Real inverse FFT	(RFT1)
Complex inverse FFT	(CFT1)

Histogram (HST)

The histogram command (HST) computes the histogram of a block and records the histogram count in a second block. Its syntax is as follows:

HSTn1,n2,vmax

where n1 is the block to be computed (cannot be complex)
n2 is the destination block
vmax is the maximum absolute amplitude range for block n1

The number of histogram bins equals the number of points in the destination block (must be greater than zero). Vmax should be greater than the magnitude of any element in n1 to allow for rounding.

Real FFT (RFFT)

The real FFT command (RFFT) computes the FFT of a real integer data block and stores the result in a second block. Its syntax is as follows:

RFFTn1,n2

where n1 is the block to be transformed
n2 is the destination for the result

The result is a 1k complex block. RFFT can be performed only on block sizes of 2048 that reside on 2k boundaries in memory. The imaginary part of the DC bin contains the Fs/2 point (used by the inverse FFT). To place a block on a 2k boundary, make sure that all data blocks up to the block to be transformed are multiples of 2 kwords long. Also, blocks for FFT and inverse FFT operations must reside in the first 32 kwords of the 37.9 kwords available for signal processing primitives. The FFT commands use the window currently selected, unless the force or exponential is active, in which case the uniform window is used.

NOTE 1

To obtain the correct 2-sided linear spectra from the FFT commands, multiply by the appropriate window correction factor:

Uniform → 1.414242555
Hann → 2.828485107
Flat top → 7.403524615

NOTE 2

Forward FFTs use a coefficient of $\frac{1}{\sqrt{2N}}$.

Complex FFT (CFFT)

The complex FFT command (CFFT) computes the FFT of a complex integer data block and stores the result in a second block. Its syntax is as follows:

CFFTn1,n2

where n1 is the block to be transformed
n2 is the destination for the result

The result is a 1k complex block. CFFT can be performed only on block sizes of 1024 complex points that reside on 2k boundaries in memory. To place a block on a 2k boundary, make sure that all data blocks up to the block to be transformed are multiples of 2 kwords long. Also, blocks for FFT and inverse FFT operations must reside in the first 32 kwords of the 37.9 kwords available for signal processing primitives. The FFT commands use the window currently selected, unless the force or exponential is active, in which case the uniform window is used.

Real Inverse FFT (RFT1)

The real inverse FFT command (RFT1) computes the inverse FFT of a complex integer data block and stores the result in a second block. Its syntax is as follows:

RFT1n1,n2

where n1 is the block to be transformed
n2 is the destination for the result

The result is a 2k real block. RFT1 can be performed only on block sizes of 1024 complex points that reside on 2k boundaries in memory. To place a block on a 2k boundary, make sure that all data blocks up to the block to be transformed are multiples of 2 kwords long. Also, blocks for FFT and inverse FFT operations must reside in the first 32 kwords of the 37.9 kwords available for signal processing primitives. The FFT commands use the window currently selected, unless the force or exponential is active, in which case the uniform window is used. n1 is destroyed by the inverse FFT.

Complex Inverse FFT (CFT1)

The complex inverse FFT command (CFT1) computes the inverse FFT of a complex integer data block and stores the result in a second block. Its syntax is as follows:

CFT1n1,n2

where n1 is the block to be transformed
n2 is the destination for the result

The result is a 1k complex block. CFT1 can be performed only on block sizes of 1024 complex points that reside on 2k boundaries in memory. To place a block on a 2k boundary, make sure that all data blocks up to the block to be transformed are multiples of 2 kwords long. Also, blocks for FFT and inverse FFT operations must reside in the first 32 kwords of the 37.9 kwords available for signal processing primitives. The FFT commands use the window currently selected.

PLOTTING AND GRAPHING DATA BLOCKS

The HP 3562A's plotting and graphing primitives allow you to display data blocks on the analyzer's screen. The plotting operations plot data versus data to create traces. The graphing operations create displays given a data block and an X-axis increment. The commands covered in this section are:

Plot complex block	(PCBL)
Plot real block	(PRBL)
Graph block	(GRBL)
Graph imaginary part	(GRIM)
Graph real part	(GRRE)

Plotting Complex Blocks (PCBL)

The plot complex block command (PCBL) plots the real part of a complex block versus the imaginary part of that block. Its syntax is:

PCBLn1

where n1 is the complex block to be plotted.

Plotting Real Blocks (PRBL)

The plot real block command (PRBL) allows you to create a display by plotting one real floating point data block against another. Its syntax is:

PRBLn1,n2

where n1 is the first real block
n2 is the second real block

Both blocks must be real and their point counts must be set the same.

Graphing Real Blocks (GRBL)

The graph real block command (GRBL) creates a trace from a real block and an X-axis increment. Its syntax is:

GRBLn1,x,Δx

where n1 is the block to be graphed
x is the X-axis starting point
Δ is the X-axis increment

Before using this command, you need to create and activate a display buffer that is at least as big as the primitive block you want to graph. Refer to Chapter 5 for handling display buffers. The primitive block n1 is transferred to the active display buffer when GRBL is executed.

Graphing Imaginary Part of Blocks (GRIM)

The graph imaginary part command (GRIM) is similar to the graph block (GRBL), except that GRIM uses just the imaginary part of a complex block to create the trace. Its syntax is:

GRIMn1,x,Δx

where n1 is the block to be graphed
x is the X-axis starting point
Δ is the X-axis increment

Refer to GRBL if you need more information.

Graphing Real Parts of Blocks (GRRE)

The graph real part command (GRRE) operates in the same manner as GRIM, except that GRRE graphs the real part of a complex block. Its syntax is:

GRREn1,x,Δx

where n1 is the block to be graphed
x is the X-axis starting point
Δ is the X-axis increment

Refer to GRIM and GRBL if you need more information.

DISPLAY CONTROL GROUP

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to explain the use of the display control group of bus-only commands. There are three approaches to programming the display: the Hewlett Packard Graphics Langauge (HP-GL), the binary language used by the display, or defining the display as a plotter for HP BASIC 3.0 graphics commands. This chapter addresses the following topics:

1. Description of the vector display
   — methods of display programming

2. Handling buffers

3. Programming with HP-GL
   — moving the pen
   — writing into buffers
   — drawing into buffers

4. Direct binary programming
   — the 1345A programming langauge
   — loading binary display buffers

5. Defining the display as a BASIC 3.0 plotter

6. Dumping display buffers

To get started, read the description of the display and the instructions on handling buffers, then select the method best suited to your application.

DESCRIPTION OF THE VECTOR DISPLAY

The HP 3562A's display produces images by combining vectors and text characters. There are 2048 points on each axis, for a total of over 4 million addressable points on the display. The lower left corner of the display is address 0,0 and the upper right corner is 2047,2047. The display's aspect ratio is 4.7:3.9 (X,Y).

Methods of Programming the Display

As stated at the beginning of the chapter, there are three methods you can use to program images on the display:

1. HP-GL commands
2. Direct binary programming
3. Defining the display as a BASIC 3.0 plotter

HP-GL is the language used by Hewlett-Packard plotters, and the HP 3562A implements a subset of that langauge. Using HP-GL is a simple way to create custom graphics. You can create up to 16 display buffers, which you then fill with commands and put on the display as needed. The commands are straightforward; each performs just one function. For example, to select line type 1 (solid lines), you simply send the Line Type command "LT1" to the appropriate buffer. The Introductory Programming Guide in Appendix A provides an example of HP-GL programming.

Direct binary programming is one level closer to the display hardware and software. Instead of many simple commands, this method has just four commands, each of which can perform multiple tasks. Each command is a 16-bit word, and you configure each bit in the command. For example, the Set Conditions command selects line types as well, but it can also select brightness and writing speed. While the direct binary commands are more complicated, they provide faster display control because fewer individual commands are required. (In fact, the HP-GL commands are used internally to select the binary commands; HP-GL isolates you from the bit-by-bit programming.)

Identifying the display as a plotter for BASIC 3.0 graphics commands allows you to program the display in a high level langauge. This method is the easiest for BASIC 3.0 users, but it the slowest.

In summary, use the display as a BASIC 3.0 plotter when you want programming that is easy to learn and easy to use, and when speed is not a concern. Use the direct binary method for more serious graphics work when both program size and execution time are critical. Finally, use HP-GL when you need faster execution than BASIC 3.0 and friendlier programming than direct binary.

A two-step procedure that gives you the ease of HP-GL and the speed of direct binary is to load a buffer with HP-GL commands, dump it back to the analyzer, then reload it as a binary command buffer. Once you convert a set of HP-GL commands to binary, which is done automatically as you fill the buffer, you can then take advantage of direct binary's speed. "Dumping Display Buffers," later in this chapter, explains how to do this.

Overview of Display Programming Steps

Regardless of the method you use, there are four general steps to programming user displays:

1. Create display buffers in the analyzer's memory
2. Activate a particular buffer
3. Load the buffer (with HP-GL, binary or BASIC 3.0)
4. Display the buffer

You must follow this sequence to get anything on the display. Steps 1, 2 and 4 are independent of the method used and are covered in the next section, "Handling Display Buffers." Step 3 is dependent on the method used; the three methods are discussed individually later in this chapter.

HANDLING DISPLAY BUFFERS

A display buffer is simply an area you reserve in the HP 3562A's memory for display programming. You can create up to sixteen display buffers. There are six commands for handling buffers:

DBSZ (display buffer size)—creates and sizes buffers

DBAC (display buffer activate & clear)—clears and activates a particular buffer

DBAA (display buffer activate & append)—activates a buffer and allows commands to be added to it

DBUP (display buffer up)—puts a buffer up on the display

DBDN (display buffer down)—takes a buffer down off the display

DBSW (display buffer switch)—replaces the buffer on the display with another buffer.

These commands are discussed in the following paragraphs. Keep in mind that the general sequence used with buffers is to create a buffer, activate it, fill it with commands, then put it up on the display.

Creating Buffers

Buffers are created with the DBSZ (display buffer size) command. This sets the size, identifies each buffer with a unique number, and determines how many buffers are created. Its syntax is:

DBSZs,n1,n

where s is size of buffer in words
n1 is number of first buffer
n is number of buffers created

For example, the BASIC statement:

OUTPUT 720; "DBSZ100,0,4"

creates 4 buffers, numbered 0, 1, 2 and 3, each 100 words long. There are approximately 11 kwords of memory available for all display buffers, and the combined size of all buffers you create cannot exceed this. The number of the first buffer, n1, must be between 0 and 15, inclusive. The number of buffers, n, cannot cause buffers numbered higher than 15. For example, if n1 is 10, n cannot be greater than 6.

Clearing and Activating a Buffer

Before a buffer can be filled, it must be activated. You have two choices: clear and activate or append and activate (discussed next). One buffer can be active at any time; it is the active buffer that receives the graphics commands sent to the analyzer. The syntax for clearing and activating is:

DBACn

where n is the number of the buffer

The buffer specified must already exist, and n must be between 0 and 15, inclusive. For example, the command:

DBAC1

clears buffer number 1 and then activates it. If the specified buffer is already on the display, DBAC takes it down and clears it.

Clearing Buffers

To clear a buffer without activating it, use the clear buffer command (CLBFn, where n is the buffer to be cleared).

Appending and Activating a Buffer

If you need to add commands to a buffer that has some commands already it but is not currently active, you need to append and activate, rather than clear and activate. The syntax is:

DBAAn

where n is the buffer to be activated

For example, the BASIC statement:

OUTPUT 720; "DBAA5"

activates buffer number 5 without clearing it. As with DBAC, the buffer must already exist, and n must be between 0 and 15, inclusive. If the n is already on the display, it is taken down and activated.

Putting Buffers Up and Down

After you have filled a buffer with the desired commands, the next step is to put it up on the display. This is done with DBUPn, where n is the buffer to be displayed. The command is ignored if n is already up.

To take a buffer down, use DBDNn, were n is the buffer to be taken down. For both DBUP and DBDN, the buffer must already exist, and n must be between 0 and 15,inclusive.

Display Buffer Switch

For fast buffer switching, the DBSW (display buffer switch) command is provided. Its syntax is:

DBSWn1,n2

where n1 is the buffer to go up
n2 is the buffer to come down

If n1 is already on the display, the command has no effect. Both buffers must already exist, and n1 and n2 must be between 0 and 15, inclusive.

PROGRAMMING WITH HP-GL

The Hewlett Packard Graphics Langauge (HP-GL) provides a simple method of programming the analyzer's display. Here is the general sequence of steps used with HP-GL:

1. Set up necessary buffer(s)
2. Activate one buffer
3. Move pen to desired location
4. Write text or draw vector
5. Repeat steps 3 and 4 as needed
6. Put the buffer up on the display

Modify this sequence as needed to produce your display. Steps 1, 2 and 6 are discussed earlier in this chapter under "Handling Display Buffers." Remember that the screen does not change until the buffer is put up on the display. The following sections shows you how to move the pen, write text, and draw vectors.

MOVING THE PEN

The "pen" is the beam used to produce images on the display. The nomenclature is carried over from the original use of HP-GL, where the pen is an actual pen in a plotter. This section explains how to control and move the pen. This is needed in two areas: positioning the pen to start writing or drawing, and to actually draw vectors.

Turning the "Pen" On and Off

Two commands determine whether the pen is up or down. PU (pen up) lifts the pen (turns the beam off). PD (pen down) sets the pen down (turns the beam on). To move from one point to another without drawing on the display, as when positioning the pen to start drawing, turn the beam off. To move while drawing, as when drawing a vector or writing text, turn the beam back on. In many cases you cannot be certain of the beam's current status, so it is a good idea to explicitly turn it on or off before moving it. Note that, unlike a plotter, dropping the pen on a display does not produce a dot; you need to move it a short distance to produce a mark.

Absolute and Relative Plotting

There are two ways of moving the pen: absolute plotting and relative plotting. Absolute plotting moves to an address relative to the origin (0,0—the lower left corner). The command is PA (Plot Absolute). For example, the BASIC statement:

```
OUTPUT 720; "PA1000,1000"
```

moves the beam to approximately the center of the display. The first number if the X-axis location, and the second is the Y-axis location. Remember this will draw or not draw to address, depending on whether the beam is on or off.

Relative plotting moves to an address relative to the current position of the beam. The command is PR (Plot Relative). For example, if the pen had not been moved since the PA1000,1000 command, sending the basic statement:

```
OUTPUT 720; "PR0,-500"
```

moves the beam 500 Y-axis units down from the center of the display. The X-axis location is not changed because its relative address was specified as 0. Note that negative X values move the beam to the left, and negative Y values move the beam down.

WRITING INTO BUFFERS

Once you have the pen positioned, you can write text into the buffer. You can control character size, brightness, and rotation when writing text.

Setting Character Size

Character size is set with CHSZn, where n is 0-3:

0 = 24 × 36 points (default)
1 = 36 × 54 points
2 = 48 × 72 points
3 = 60 × 90 points

Setting Brightness

There are four levels of brightness you can select, using BRITn, where n is 0-3:

0 = off
1 = dim
2 = half bright
3 = full bright (default)

Rotating Characters

Characters can be rotated at four angles, using CHROn, where n is 0-3:

0 = 0° (default)
1 = 90°
2 = 180°
3 = 270°

Writing on the Display

When you have positioned the beam and set size, brightness and rotation, you are ready to write text. The command is WRIT, and the alpha string must be enclosed either in single quote marks or a pair of double quote marks. For example, the BASIC statements:

OUTPUT 720; "WRIT'MESSAGE'"

and OUTPUT 720; "WRIT""MESSAGE"""

both write MESSAGE at the current beam position. Because of the obvious complexity of the second format, the first is recommended.

As an example of combining the four text commands, the BASIC statements:

```
OUTPUT 720; "CHSZ2"
OUTPUT 720; "BRIT3"
OUTPUT 720; "CHRO1"
OUTPUT 720; "WRIT'XXXXXXXXXX'"
```

write XXXXXXXXXX on the display at a 90 degree angle, with character size 2 and brightness 3.

DRAWING INTO BUFFERS

Drawing vectors is merely a special application of moving the beam. Send the PD command to turn the beam on, then PA (Plot Absolute) and PR (Plot Relative) can draw vectors for you. For example, the BASIC statements:

```
OUTPUT 720;"PU"
OUTUPT 720;"PA1000,1000"
OUTPUT 720;"PD"
OUTPUT 720;"PR0,-800"
```

draw a vector from the center of the screen to 800 units down the Y-axis to 1000,200. The brightness selection (BRITn) explained in the last section applies to vectors as well. There is one more selection for vectors only, selecting the line type.

Selecting Line Types

Lines types can be selected with LTn, where n is 0-4:

0 = solid lines (default)
1 = solid lines with intensified endpoints
2 = long dashed lines
3 = short dashed lines
4 = endpoints only

If an optional second parameter is sent, it is ignored (for HP-GL compatibility).

Figure 5-1 shows the five line types available.

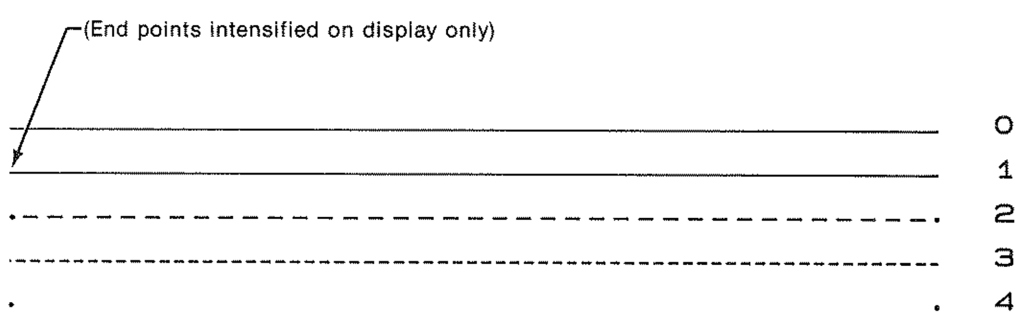


Figure 5-1 Display Line Types

DIRECT BINARY PROGRAMMING

This is the fastest method of programming user displays. As explained earlier, it uses the commands that the display processor itself uses. This saves time by bypassing the conversion from HP-GL to the display's binary language.

The overall programming sequence is the same: create a buffer, activate it, load it, then display it. This method provides commands to dump and load the user display buffers in ANSI floating point, ASCII, or internal binary format. (See Chapter 3 for descriptions of these data formats.)

The Display's Binary Language

The HP 3562A uses the HP 1345A Digital Display. The 1345A receives 16-bit words that are decoded into its four commands:

PLOT—moves the beam on the screen

GRAPH—creates a graph given a set of data

SET CONDITION—defines vector attributes (brightness, etc.)

TEXT—writes alphanumeric text

Section V of the 1345A Designer's Manual and the 1345A Quick Reference Guide have been included at the end this chapter to give you all the details of direct binary programming.

When you want to fill the active buffer, you send a data array with the load user display commands: LUAS (ASCII), LUAN (ANSI), or LUBN (internal binary). The following sections explain how to transfer binary display data to the analyzer. Please refer to Chapter 3 for descriptions of the three data formats.

Loading User Buffers in ASCII (LUAS)

The LUAS command loads the active display buffer with ASCII integer values. Here is a sample listing:

```
OUTPUT @Dsa; "DBSZ100,1"
OUTPUT @Dsa; "DBAC1"
OUTPUT @Dsa; "LUAS"
OUTPUT @Dsa USING "2A,K";"#I",5
OUTPUT @Dsa; Array(*)
OUTPUT @Dsa; "DBUP1"
```

After receiving LUAS, the analyzer expects #I to specify ASCII data, then a variable containing the number of ASCII variables to be sent (5 in this example). After receiving these first four bytes, the analyzer is ready for data, which is in "Array" in this example. After the active buffer (#1) is filled with the contents of "Array," the buffer is put up on the display with DBUP.

Loading User Buffers in ANSI Floating Point (LUAN)

The LUAN command loads the active display buffer with 64-bit ANSI floating point values, which are converted to integers. Here is a sample listing:

```
OUTPUT @Dsa; "DBSZ100,1"
OUTPUT @Dsa; "DBAC1"
OUTPUT @Dsa; "LUAN"
OUTPUT @Dsa USING "#,2A,W";"#A",40
OUTPUT @Dsa; FORMAT OFF
OUTPUT @Dsa; Array(*)
OUTPUT @Dsa; FORMAT ON
OUTPUT @Dsa; "DBUP1"
```

After receiving LUAN, the analyzer expects to receive #A to specify ANSI data followed by the length word specifying the number of bytes to be output (40 in this example). After receiving these first four bytes, the analyzer is ready for data, which is in "Array" in this example. (The ASCII formatter was deactivated for this computer to prevent it from converting ANSI to ASCII. Your computer/language may handle this differently; if it automatically formats output data to ASCII, you need to disable this feature before sending ANSI data.) Finally, buffer #1 is put up on the display.

Loading User Buffers in Internal Binary (LUBN)

The LUBN command loads the active display buffer in the HP 3562A's internal binary format. This load command can be used only with data that have been dumped from the analyzer in (or externally converted to) the internal binary format. Here is a sample listing:

```
OUTPUT @Dsa; "DBSZ100,1"
OUTPUT @Dsa; "DBAC1"
OUTPUT @Dsa; "LUBN"
OUTPUT @Dsa USING "#,2A,W";"#A",10
OUTPUT @Dsa; FORMAT OFF
OUTPUT @Dsa; Array(*)
OUTPUT @Dsa; FORMAT ON
OUTPUT @Dsa; "DBUP1"
```

After receiving LUBN, the analyzer expects to receive #A to specify binary data followed by the length word specifying the number of bytes to be output (10 in this example). After receiving these first four bytes, the analyzer is ready for data, which is in "Array" in this example. (The ASCII formatter was deactivated for this computer to prevent it from converting binary to ASCII. Your computer/language may handle this differently; if it automatically formats output data to ASCII, you need to disable this feature before sending binary data.) Finally, buffer #1 is put up on the display.

DISPLAY PROGRAMMING WITH BASIC 3.0

As explained at the beginning of this chapter, the third method of display programming is defining the analyzer's display as the plotter for BASIC 3.0 graphics. The use of this technique is described in BASIC 3.0 Graphics Techniques. The command used to specify the display is:

```
PLOTTER IS 720, "HPGL"
```

where 720 is the analyzer's address

For example, the following BASIC 3.0 statements plot a box and some large text on the display:

```
PLOTTER IS 720, "HPGL"
OUTPUT 720; "DBSZ250,1"
OUTPUT 720; "DBAC1"
VIEWPORT 0,88,5,99
WINDOW -130,130,-100,100
FRAME
LORG 5
CSIZE 17
MOVE 0,0
LABEL "BIG TEXT"
OUTPUT 720; "DBUP1"
```

This example also demonstrates the ability of this technique to draw larger text than is possible with the HP-GL technique.

DUMPING DISPLAY BUFFERS

Every display buffer in the HP 3562A, both user buffers and the analyzer's own internal buffers, can be dumped via HP-IB. The internal buffers hold data traces, marker readouts, etc. This section shows you how to select the buffer to be dumped, describes the internal display buffers, and shows how to dump the selected buffer.

Dumping buffers takes two steps: first, use the vector block pointer (VBLK) to identify the buffer to be dumped. Second, select the data format in which you want the data dumped, then send the appropriate command. Buffers can be dumped in ASCII, ANSI floating point, and the internal binary formats. (For general information on these formats, please refer to Chapter 3.)

Display buffers contain 1345A binary commands (see "Direct Binary Programming" earlier in this chapter). When HP-GL or BASIC 3.0 commands are loaded, they are converted to 1345A commands by the analyzer. Because of this conversion, you can program a display initially with HP-GL or BASIC 3.0, load it into the analyzer, then dump out the direct binary equivalent. If you then store these binary commands, you can have the speed advantage of direct binary any time in the future that this display is needed.

The Vector Display Buffer Pointer (VBLK)

The buffer to be dumped is selected with the vector buffer pointer command (VBLK). Its syntax is:

VBLKn

where n is the buffer number

The number you specify with n depends on whether or not user buffers are being used. Table 5-1 shows the value of n to be used for dumping all user and internal display buffers. Note that to dump user buffers, their numbers are offset by +4 from the number used to identify them for other graphics commands.

Table 5-1 Identifying Buffer Pointer Values

Value of n (VBLKn)	User buffer	Internal buffer
0	—	Softkey underlining
1	—	Softkey menu
2	—	Command echo
3	—	Message
4	0	Special markers, trace A
5	1	Special markers, trace B
6	2	X marker readout
7	3	Y marker readout
8	4	Trace A
9	5	Trace B
10	6	Grid
11	7	—
12	8	Ya readout
13	9	Yb readout
14	10	Xa readout
15	11	Xb readout
16	12	A label
17	13	B label
18	14	—
19	15	—

If any user buffer has been created, the user buffer corresponding to n is dumped. Otherwise, the internal buffer corresponding to n is dumped. For example, if you set up a user buffer with the DBSZ command then send VBLK10, you will get user buffer 6 if you send a dump command. However, if you had not created a user buffer and you sent VBLK10, you would get the internal grid buffer in response to a dump command.

Dumping Buffers in ASCII (DVAS)

The display buffer identified with the vector buffer pointer (VBLK) can be dumped in ASCII format with the DVAS command. There is no header with this transfer, just #I and the length variable. The following BASIC statements dump the internal buffer that contains the softkey labels:

```
OPTION BASE 1
OUTPUT 720; "DVAS"
ENTER 720 USING "2A,K";A$,Length
REDIM Buffer(Length)
ENTER 720 Buffer(*)
```

This dumps the #I format specifier into A$, the length variable into "Length," and the ASCII variables into integer array "Buffer."

Dumping Buffers in ANSI Floating Point (DVAN)

The display buffer identified with the vector buffer pointer (VBLK) can be dumped in ANSI floating point format with the DVAN command. There is no header with this transfer, just #A and the length word indicating the number of bytes to be transferred. The following BASIC statements dump the internal buffer that contains the softkey labels:

```
OPTION BASE 1
ASSIGN @Dsa to 720
OUTPUT @Dsa; "DVAN"
ENTER @Dsa USING "%,2A,W";A$,Length
REDIM Buffer (Length DIV 8)
ASSIGN @; Dsa FORMAT OFF
ENTER @Dsa Buffer(*)
```

This dumps the #A format specifier into A$, the length word into "Length," then redimensions the array to Length/8 (8-byte floating point values).

Dumping Buffers in Internal Binary (DVBN)

The display buffer identified with the vector buffer pointer (VBLK) can be dumped in the analyzer's internal binary format with the DVBN command. There is no header with this transfer, just #A and the length word indicating the number of bytes to be transferred. The following BASIC statements dump the internal buffer that contains the softkey labels:

```
OPTION BASE 1
ASSIGN @Dsa to 720
OUTPUT @Dsa; "DVBN"
ENTER @Dsa USING "%,2A,W";A$,Length
REDIM Buffer (Length DIV 8)
ASSIGN @Dsa; FORMAT OFF
ENTER @Dsa Buffer(*)
```

This dumps the #A format specifier into A$, the length word into "Length," then redimensions the array to Length/2 (2-byte values).

1345A Quick Reference Guide

1345A COMMANDS.

NOTE: Bit D15 is used only for vector memory board commands. For standard 1345A commands, D15 should be Ø.

1345A 16 Bit Data Word.

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
COMMAND		← COMMAND MODIFIERS →												

1345A COMMANDS

	Bit 14	Bit 13
Set Condition	1	1
Plot Command	Ø	Ø
Graph Command	Ø	1
Text Command	1	Ø

Set Condition Command.

Set Condition Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
1	1	I_1	$I_Ø$	X	X	L_1	$L_Ø$	Ø	X	W_1	$W_Ø$	X	X	X

Note: Bit 6 (D6) must be zero.

Command Modifiers:

a. To Set Line Intensity:

I_1	$I_Ø$	Intensity
Ø	Ø	Blank
Ø	1	Dim
1	Ø	Half Brightness
1	1	Full Brightness

b. To Set Line Type:

L_1	$L_Ø$	Type
Ø	Ø	Solid Line
Ø	1	Intensified End Points on Solid Line
1	Ø	Long Dashes
1	1	Short Dashes

c. To Set Writing Speed:

W_1	$W_Ø$	Speed
1	1	0.05 in. per μs
1	Ø	0.10 in. per μs
Ø	1	0.15 in. per μs
Ø	Ø	0.20 in. per μs

Plot Command.

Plot Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
Ø	Ø	XY	PC	$D_{1Ø}$	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
				← DATA →										

Command Modifiers:

a. XY Information (D12)

Ø = X coordinate (Ø-2Ø47), specified by $D_Ø$-$D_{1Ø}$
1 = Y coordinate (Ø-2Ø47), specified by $D_Ø$-$D_{1Ø}$

b. PC Beam Control Information (D11)

Ø = Beam OFF (move)
1 = Beam ON (draw)

Programming Command Ranges.

PROGRAMMING COMMAND RANGES OF THE 1345A

1345A Command	Octal Range	Hexadecimal Range
a. Plot		
X	00000-07777	0000-0FFF
Y (beam off)	10000-13777	1000-17FF
Y (beam on)	14000-17777	1800-1FFF
b. Graph		
Set Delta-X	20000-27777	2000-2FFF
Y (beam off)	30000-33777	3000-37FF
Y (bem on)	34000-37777	3800-3FFF
c. Text	40000-57777	4000-5FFF
d. Set Condition	60000-77777	6000-7FFF

Graph Command.

Graph Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
Ø	1	XY	PC	$D_{1Ø}$	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
				← DATA →										

Command Modifiers:

a. XY Information (D12)

Ø = Set Delta-X increment, specified by $D_Ø$-$D_{1Ø}$ for all subsequent Y coordinates
1 = Set Y coordinate, specified by $D_Ø$-$D_{1Ø}$. The beam is to be moved to this Y in conjunction with the Delta X increment.

b. PC Beam Control Information (D11)

Ø = Beam OFF (move)
1 = Beam ON (draw)

MEMORY BOARD COMMANDS.

Vector Memory Word.

M15	M14	M13	M12	M11	M1Ø	M9	M8	M7	M6	M5	M4	M3	M2	M1	MØ
Ø	B_{14}	B_{13}	B_{12}	B_{11}	$B_{1Ø}$	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1	$B_Ø$

(SEE DATA BIT DEFINITIONS FOR 1345A COMMANDS)

Internal Jump.

An internal jump does not affect the Vector Memory address pointer.

M15	M14	M13	M12	M11	M1Ø	M9	M8	M7	M6	M5	M4	M3	M2	M1	MØ
1	Ø	X	X	A_{11}	$A_{1Ø}$	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	$A_Ø$

X = DON'T CARE
M15 = 1, M14 = Ø Internal jump to Vector Memory address specified by A_{11} thru $A_Ø$ during refresh.

Address Pointer.

M15	M14	M13	M12	M11	M1Ø	M9	M8	M7	M6	M5	M4	M3	M2	M1	MØ
X	X	X	X	A_{11}	$A_{1Ø}$	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	$A_Ø$

X = DON'T CARE
$A_Ø$ = Ø: Set pointer register to the Vector Memory address value specified by A_{11} thru $A_Ø$.

1345A Modified ASCII Character Set.

1345A MODIFIED ASCII CODE CONVERSION TABLE

LEAST SIGNIFICANT CHARACTER	MOST SIGNIFICANT CHARACTER Ø	1	2	3	4	5	6	7
Ø		centered *	SP	Ø	@	P	`	p
1	HP logo	centered o	!	1	A	Q	a	q
2	β	↑	"	2	B	R	b	r
3		←	#	3	C	S	c	s
4	upper-half tic	↓	$	4	D	T	d	t
5	lower-half tic	→	%	5	E	U	e	u
6	left-half tic	√	&	6	F	V	f	v
7	right-half tic	π	'	7	G	W	g	w
8	back space	Δ	(	8	H	X	h	x
9	1/2 shift down	μ	)	9	I	Y	i	y
A	line feed	° (degree)	*	:	J	Z	j	z
B	inv. line feed	Ω	+	;	K	[	k	{
C	1/2 shift up	ρ	,	<	L	\	l	|
D	carriage return	Γ	-	=	M	]	m	}
E	horizontal tic	θ	.	>	N	^	n	□
F	vertical tic	λ	/	?	O	—	o	►

EXAMPLES:

HP logo	=	Ø1
A	=	41
i	=	69
√	=	16
►	=	7F
line feed	=	Ø9

Capabilities for Character and Vector Combinations.

Conditions.

Average character drawing time: 16 μsec
Recommended refresh rate: 60 Hz ~ 16.6 msec
1345A writing speed: 0.1 in./μsec
Vector dead time: 1 μsec

	NUMBER OF CHARACTERS TO BE DRAWN			
	0	100	200	300
Total frame time (msec)	16.67	16.67	16.67	16.67
Character writing time (msec)	0	1.60	3.20	4.80
Time left to draw vectors (msec)	16.67	15.07	13.47	11.87
AVERAGE VECTOR LENGTH	APPROXIMATE NUMBER OF VECTORS DRAWN			
0.1 in.	8330	7530	6730	5930
0.5 in.	2770	2510	2240	1970
2.0 in.	790	710	640	560
6.0 in.	270	240	220	190

Text Command.

Text Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
1	Ø	S_1	$S_Ø$	R_1	$R_Ø$	ES	C_7	C_6	C_5	C_4	C_3	C_2	C_1	$C_Ø$

CHARACTER ——→

Command Modifiers:

For $C_Ø$-C_7, see modified ASCII conversion table

a ES Establish Size of Character

Ø = Use previous size and rotation
1 = Establish new size and rotation according to S_1, $S_Ø$, R_1 and $R_Ø$

b Rotate Character CCW

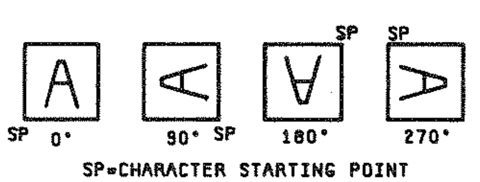
R_1	$R_Ø$	Rotation
Ø	Ø	0 degrees
Ø	1	90 degrees
1	Ø	180 degrees
1	1	270 degrees

S_1	$S_Ø$	Size	W × H (in addressable points)
Ø	Ø	1×	24 × 36
Ø	1	1.5×	36 × 54
1	Ø	2×	48 × 72
1	1	2.5×	60 × 90

4 PROGRAMMABLE CHARACTER SIZES:

1.0 × 56 characters per line, 29 horizontal lines possible.
1.5 × 37 characters per line, 19 horizontal lines possible.
2.0 × 28 characters per line, 14 horizontal lines possible.
2.5 × 22 characters per line, 11 horizontal lines possible.

Character Rotation.



Vector Drawing Time Calculations.

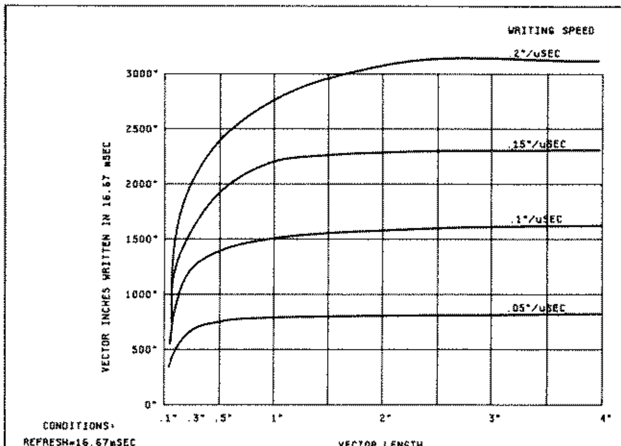
$$\text{VECTOR DRAWING TIME} = \frac{\text{VECTOR LENGTH}}{\text{WRITING SPEED}} + \frac{1\ \mu s}{\text{VECTOR}}$$

$$\text{PICTURE DRAWING TIME} = \sum_{1}^{N} \frac{\text{VECTOR LENGTH}}{\text{WRITING SPEED}} + \frac{1\ \mu s}{\text{VECTOR}} + \sum_{1}^{M} \frac{15\ \mu s}{\text{CHARACTER}}$$

N = TOTAL NUMBER OF VECTORS

M = TOTAL NUMBER OF CHARACTERS

Vector Length vs. Writing Speed for 60 Hz Refresh Rate.



SECTION V
1345A PROGRAMMING

INTRODUCTION.

This section of the DESIGNERS MANUAL will describe the programmable functions of the 1345A Digital Display Module. Proper understanding of the capabilities and limitations of the 1345A will enable the user to obtain optimum performance. This section of the manual will be divided into three parts. These three parts will address the areas of 1345A Programming Commands, 1345A Display Requirements, and Performance Optimization. It is recommended that the user read through Section 4, Interfacing the 1345A, prior to reading this section. Please read the complete text once to gain a firm foundation of the total 1345A operating environment.

The 1345A Digital Display has 4 commands. These are PLOT, GRAPH, SET CONDITION, and TEXT. These four commands provide complete programmable vector and text generation with a minimum of command overhead. Most vector and text operations can be handled with only one 16 bit command word.

The 1345A receives 16 bit data words over the 26 pin interface connector. These 16 bit data words are decoded by the 1345A into one of four distinct commands. Each 16 bit data word sent to the 1345A can be separated into two distict data fields. The 1345A 16 bit data word is shown in figure 5-1.

Each of the commands that the 1345A can recognize is selected by the state of data bits D14 and D13. Data bit D15 is used only for memory board operations and is discussed later. The 1345A without memory uses only data bits D0-D14. The lower 13 data bits D0-D12 are used as command modifiers.

MSB														LSB
D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	DØ
COMMAND		← COMMAND MODIFIERS →												

1345A COMMANDS

	Bit 14	Bit 13
Set Condition	1	1
Plot Command	Ø	Ø
Graph Command	Ø	1
Text Command	1	Ø

Figure 5-1. 1345A 16 Bit Data Word

These modifiers allow each command to have several selectable attributes. Vector drawing operations are directly dependent on the status of these data bits in every 1345A command. Each of these commands and their modifiers will be dicussed using programming examples. The 16 bit data for the examples will be in HEXADECIMAL or HEX format. This format is easier to follow than 16 bit binary data words. Each HEX data word sent to the 1345A will be equivalent to a 16 bit binary word.

HEX Format Generation. Each 16 bit data word can be separated into four, four bit binary numbers. This allows each four bit binary number to have sixteen distinct combinations. Each of these combinations is assigned a HEX equivalence. The conversion from binary to HEX is contained in figure 5-2.

Each data word in the following command examples will use this HEX format. These HEX representations will correspond to the required bit patterns recognized by the 1345A.

Hexadecimal Code	Binary Code b4	b3	b2	b1
Ø	Ø	Ø	Ø	Ø
1	Ø	Ø	Ø	1
2	Ø	Ø	1	Ø
3	Ø	Ø	1	1
4	Ø	1	Ø	Ø
5	Ø	1	Ø	1
6	Ø	1	1	Ø
7	Ø	1	1	1
8	1	Ø	Ø	Ø
9	1	Ø	Ø	1
A	1	Ø	1	Ø
B	1	Ø	1	1
C	1	1	Ø	Ø
D	1	1	Ø	1
E	1	1	1	Ø
F	1	1	1	1

Hex Code	C	F	Ø	8
Binary Code	11ØØ	1111	ØØØØ	1ØØØ

Figure 5-2. Binary to HEX Conversion

PROGRAMMING EXAMPLES.

Vector Plotting.

An explanation of vector drawing will help clarify the process. In figure 5-3, there are three vectors defined by four endpoints. Each vector requires two endpoints. The vector from point 1 to point 2 requires two endpoint declarations. The vector from point 2 to point 3 requires only point 3 be declared as an endpoint, because point 2 is already established. The vector drawn from point 2 to point 3 is a vector with the beam off. This allows the beam to be moved to new vector starting points without affecting existing displayed vectors. The vector from point 3 to point 4 is drawn with the beam on. The

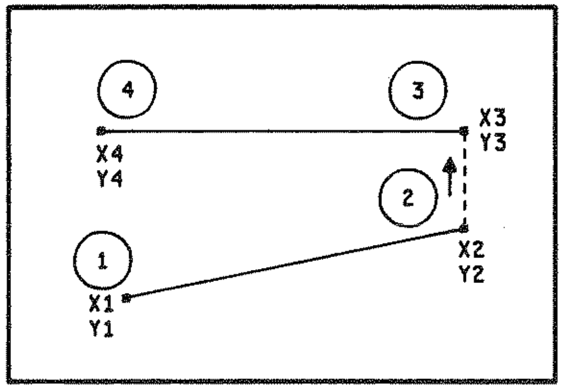


Figure 5-3. Vector Plotting

correct sequence for constructing vectors in PLOT mode is ALWAYS X first, Y next, X, Y, X, . . . , Y, until the vector sequence is complete. A vector is plotted according to the last SET CONDITION command sent to the 1345A.

The "BEAM ON" bit in the PLOT command is ignored if the coordinate being specified is an X value. The beam status only has effect if the Y coordinate is being entered. The CRT beam will move to the location specified by the last X and Y coordinate values specified in the PLOT commands.

Graph Plotting.

An example of the graph command is contained in figure 5-4. In this example 15 vectors are drawn with only 20 commands. The sequence is described below.

Step 1 — Set Condition to define line type.

Step 2 — Plot command to set X location at lower left corner of graph.

Step 3 — Plot command to set Y location at lower left corner of graph.

Step 4 — Graph command to set X increment value. This value is referenced to the X axis of the graph.

Step 5 — Graph command with beam off and Y value set to 0. This will not plot anything, but is used to initiate the X increment to point 1.

Step 6 — Graph command with beam on and Y value set to point 1.

Step 7 — Graph command with beam on and Y value set to point 2.

.

. (send only Y values of points 3 through 14)

.

Step 20 — Graph command with beam on and Y value set to point 15.

Normal X,Y plot mode would require 33 commands to construct the same graph. Note that the above command sequence does not include generation of the graph axis, only construction of the graph itself.

The construction of a graph can have two forms. The vectors may start at either the origin or somewhere along the Y axis of the graph. If the origin is the starting point, then the user needs to set the first Y value to zero. This will not plot anything but will start the graph at the origin and increment the X value by one. When the next Y value is sent, a vector will be drawn from the origin to the new Y value. If the Y axis is the starting point then the user needs to send the first Y value with the beam off. This will insure that the axis of the graph is not altered by the line type set for the graph trace. For the next Y value the beam should be turned on.

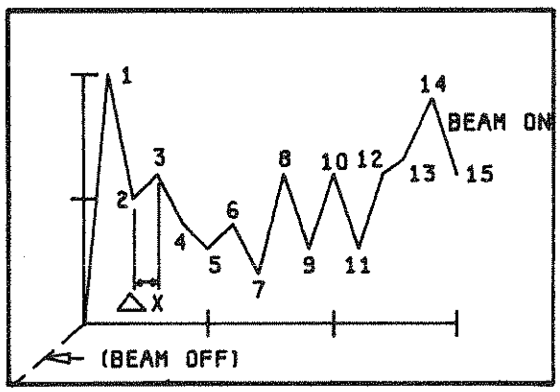


Figure 5-4. Graph Mode Example

1345A COMMANDS.

Set Condition Command.

When D14 and D13 are both in the High TTL state, the 1345A will interpret the data word as a SET CONDITION command. This command is used to set vector attributes. The attributes affected are line type, speed, and intensity. The required bit patterns for this command and its command modifiers are contained in figure 5-5.

By combining line intensity and writing speed parameters, up to twelve levels of discernible intensities can be generated. Figure 5-6 contains several example combinations. This allows the user to create displays with background graticules and intensify important trace data. The beam will be brightest with the intensity set at full bright at the slowest writing speed. The beam will be dimmest with the intensity set at dim at the fastest writing speed. The SET CONDITION command may be executed at any time and the vector attributes will remain in effect until another SET CONDITION command is executed. Data bit 6 in this command is defined to be TTL low. This MUST occur when the Set Condition command is executed or the display may respond in an undefined fashion.

Set Condition Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
1	1	I_1	$I_Ø$	X	X	L_1	$L_Ø$	Ø	X	W_1	$W_Ø$	X	X	X

Note: Bit 6 (D6) must be zero.

Command Modifiers:

a. To Set Line Intensity:

I_1	$I_Ø$	Intensity
Ø	Ø	Blank
Ø	1	Dim
1	Ø	Half Brightness
1	1	Full Brightness

b. To Set Line Type:

L_1	$L_Ø$	Type
Ø	Ø	Solid Line
Ø	1	Intensified End Points on Solid Line
1	Ø	Long Dashes
1	1	Short Dashes

c. To Set Writing Speed:

W_1	$W_Ø$	Speed
1	1	0.05 in. per μs
1	Ø	0.10 in. per μs
Ø	1	0.15 in. per μs
Ø	Ø	0.20 in. per μs

Figure 5-5. Set Condition Command

6998_h	Dim, Short Dash, Speed 0.05
7800_h	Bright, Solid, Speed 0.2
7000_h	Half Bright, Solid, Speed 0.2
7100_h	Half Bright, Long Dash, Speed 0.2

Figure 5-6. Set Condition Examples In Hex

Plot Command.

When the two most significant bits of the data word, D14 and D13 are in a low TTL state, the 1345A will recognize the data word to be a PLOT command. Figure 5-7 contains the correct bit pattern for this command.

Plot Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
Ø	Ø	XY	PC	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
				← DATA →										

Command Modifiers:

a. XY Information (D12)

Ø = X coordinate (Ø-2Ø47), specified by D_0-D_{10}
1 = Y coordinate (Ø-2Ø47), specified by D_0-D_{10}

b. PC Beam Control Information (D11)

Ø = Beam OFF (move)
1 = Beam ON (draw)

Figure 5-7. Plot Command Bit Pattern

This command moves the beam to a specific X-Y location in the defined cartesian coordinate plane each time an X-Y coordinate pair is received. The values of the X and Y coordinates range from 0 to 2047. The origin of the cartesian plane is located in the lower left corner and has an X-Y value of (0,0). This command also turns the beam on or off for each vector. The beam may be moved in either mode. The vector is drawn from the previous beam location to the current location specified by the last two X,Y coordinate values in the PLOT commands. The vector is drawn in accordance with the last SET CONDITION command received by the 1345A.

The diagram in figure 5-8 is a single vector defined by its endpoints in the vector drawing area. To draw this line the 1345A would need to receive two sets of X and Y coordinates. The 1345A receives the coordinates in the specified order X1,Y1,X2,Y2. The beam is moved only when the Y coordinate is received. The status of the beam is only affected by the beam status bit in the Y coordinate command.

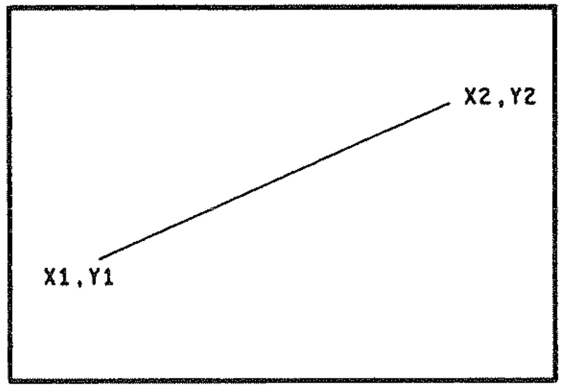


Figure 5-8. Vector Defined By Endpoints

An example of vector plotting is contained in figure 5-9. This example contains vectors drawn with the beam on and with the beam off. The steps to draw these figures are given in the required sequence with equivalent HEX code for the 16 bit data words.

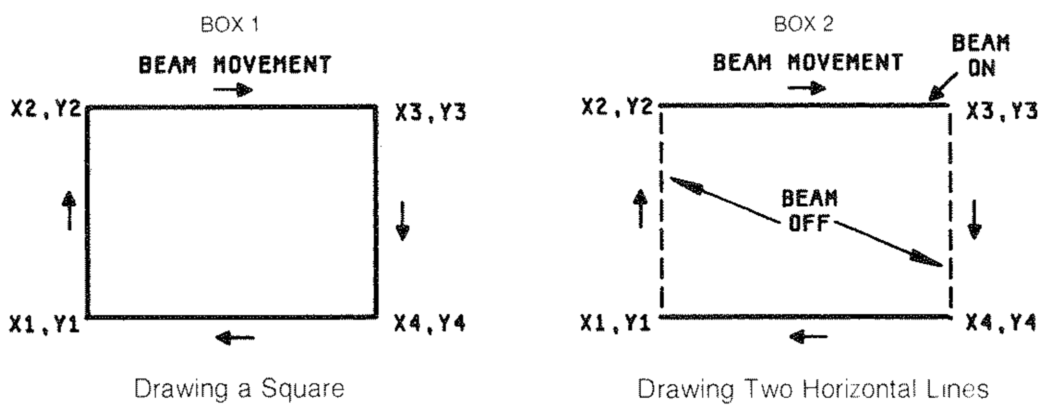


Figure 5-9. Plot Vector Example

To draw the figures, send the following 16 bit data words in sequence to the 1345A.

	Command Step	Box 1 data	Box 2 data	1345A
1.	Set Condition	7818h	7818h	Sets Vector type (Solid Full Bright, .05)
2.	Plot X1	0200h	0200h	X1=512
3.	Plot Y1 (beam off)	1200h	1200h	move to Y1=512
4.	PLot Y2 (beam on)	1F00h	(beam off) 1700h	move to Y2=1792
5.	Plot X3	8F00h	8F00h	X3=1792
6.	Plot Y3 (beam on)	1F00h	1F00h	move to Y3=1792
7.	Plot Y4 (beam on)	1A00h	(beam off) 1200h	move to Y4=512
8.	Plot X1	0200h	0200h	X1=512
9.	Plot Y1 (beam on)	1A00h	1A00h	move to Y1=512

A description of these two examples will help the user understand the vector plotting process. Step 1 defines the vector attributes for the vectors to be plotted by the 1345A. Definition of a starting point is crucial when plotting vectors. Steps 2 and 3 initialize the starting point of the box. Next a new Y value is received indicating that the beam be turned on. Since the X value didn't change, only a new Y value need be sent. The beam will move to the location specified by the X-Y location when the Y value is received. The vector is drawn according to the status of the last SET CONDITION command.

When a new horizontal location is required, both the X and Y coordinates need to be sent to the 1345A. The beam is only moved and the vector drawn when a Y coordinate is received. The Y value doesn't change going from step 4 to step 5, but the X value does. This requires that a new X-Y coordinate pair be sent to the 1345A as in steps 5 and 6. In step 7, the X value doesn't require a change so only a new Y value is sent in step 7. The beam is turned on to draw the vector. In steps 8 and 9 a new X-Y pair is required so both values must be sent. To draw box 2, only steps 5 and 7 need to be changed. The beam status bit tells the 1345A to turn the beam off during the movement. A vector is still drawn, but with the beam turned off.

The user should notice that when a vector is to be drawn vertically, only a Y value is sent for the second vector endpoint. The 1345A has a "last X" register that stores the value of the last X location. This feature allows vertical vectors with the same X values to be drawn with one less endpoint requirement.

When plotting vectors in the vector drawing area, the user should take into account the difference in CRT screen height and width. The 1345A vector drawing area is 9.5 cm high by 12.5 cm wide and has 2048 addressable points in either direction. If this difference is not taken into account, boxes will appear as rectangles. To plot vectors correctly, the user may need to apply a scaling factor to vector endpoint calculations. The scaling factors for the 1345A are approximately 215.58 addressable points/cm in the Y direction and 163.84 addressable points/cm in the X direction. These figures are used when calculating the actual length of vectors in cm.

Graph Command.

The GRAPH command is very similar to the PLOT command. The purpose of the GRAPH command is to allow plotting of vectors that have equal incremental X coordinates. When data word bits D14 and D13 are TTL low and TTL high respectively the 1345A will interpret the data word to be a GRAPH command as shown in figure 5-10. In the GRAPH mode, the 1345A will automatically increment the X coordinate after each Y-coordinate is received. This allows single valued functions to be plotted in graph form with fewer endpoints than would be possible using X,Y coordinates for each data point.

Graph Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
Ø	1	XY	PC	D1ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	Dø
				← DATA →										

Command Modifiers:

a. XY Information (D12)

Ø = Set Delta-X increment, specified by Dø-D1ø for all subsequent Y coordinates
1 = Set Y coordinate, specified by Dø-D1ø. The beam is to be moved to this Y in conjunction with the Delta X increment.

b. PC Beam Control Information (D11)

Ø = Beam OFF (move)
1 = Beam ON (draw)

Figure 5-10. Bit Definition For Graph Command

There are three command modifiers in the GRAPH command. These modifiers control the X increment, Y coordinate data value, and the beam status. When D12 is 0, the data in bits D0-D10 define the value of the X increment. This is the amount the X coordinate will increase after each Y coordinate is plotted. The range of the X increment is 0 to 2047. It should be noted that X increases relative to present X,Y coordinate values on the screen. Figure 5-11 contains an example of the graph mode commands. The beam moves when the Y coordinate value is received.

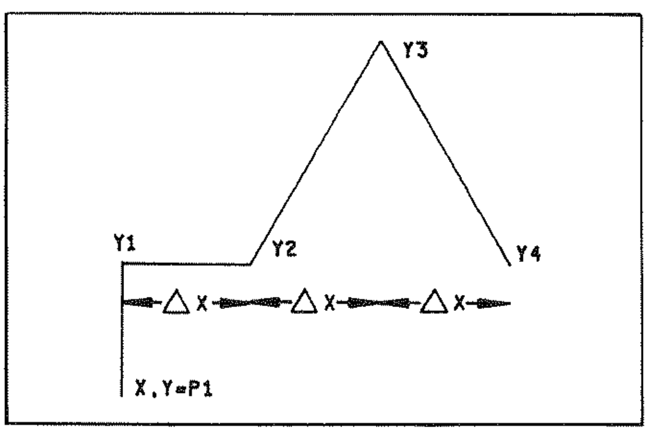


Figure 5-11. Graph Mode Example

To create the output in figure 5-10 the following steps were executed with the given 16 bit HEX data.

Command Step	16 Bit Data	1345A
1. Set Condition	7818h	Set Vector Attributes (Solid Full Bright, .05)
2. Plot X1	0200h	X=512
3. Plot Y1 (beam off)	1200h	move to Y=512
4. Graph command Set Delta X	2040h	set X increment to 64
5. Graph command Y1	3280h	Y=640
6. Graph command Y2	3280h	Y=640
7. Graph command Y3	3300h	Y=768
8. Graph command Y4	3280h	Y=640

Step 1 defines the line type, speed, and intensity. Steps 2 and 3 determine the starting point of the graph. The delta X increment is established in step 4. The (4) Y values are sent in steps 5-8. The value of X is incremented AFTER each Y value is received.

If the graph is to start at the axis origin, then execute a graph command with a first Y value set to zero. This will not plot anything, but will increment the X value by delta X. The next vector will be drawn from the origin to the Y value for the first X increment. If the graph is to start at the Y axis, then execute a Y value command. The next vector will be drawn from the Y value on the Y axis to the Y value of the first X increment.

Text.

The 1345A comes complete with an internal character generator. This internal character data is a modified ASCII character set for graphics use. The data for commanding the 1345A to enter the text mode is in figure 5-12. Data bits D14 must be TTL high and D13 must be TTL low. When this command is executed the 1345A will interpret the lower eight data bits, D0-D7 as an equivalence for an ASCII or special character. Each vector of the character is drawn on the CRT screen according to the vector characteristics of the last SET CONDITION command. The characters are always drawn at the slowest writing speed. The line type has no visible effect except on the largest character size, (2.5X). The position is defined by the last X and Y coordinates received by the 1345A.

When generating characters, the 1345A automatically provides character spacing to the right of each character.The TEXT command has command modifiers for size and rotation information. New size and rotation information is controlled by the status of data word bit D8. To initiate new character attributes, bit D8 must be set high as a new information indicator. If this data bit is "0", the size and rotation bits are ignored.

The 1345A has 4 character sizes. These 4 sizes are defined by the status of bits D11 and D12. The amount of space needed to draw the characters is contained in figure 5-12. This is the required space needed out of 2048 × 2048 possible points. The number of characters that can be drawn across the screen at the different sizes is in figure 5-13. An example of 1x character spacing is contained in figure 5-14.

Text Command:

MSB D14	D13	D12	D11	D1Ø	D9	D8	D7	D6	D5	D4	D3	D2	D1	LSB DØ
1	Ø	S_1	$S_ø$	R_1	$R_ø$	ES	C_7	C_6	C_5	C_4	C_3	C_2	C_1	$C_ø$

CHARACTER ⟶ (D7–DØ)

Command Modifiers:

For $C_ø$-C_7, see figure 5-14

a. ES Establish Size of Character

Ø = Use previous size and rotation
1 = Establish new size and rotation according to S_1, $S_ø$, R_1 and $R_ø$

b. Rotate Character CCW

R_1	$R_ø$	Rotation
Ø	Ø	0 degrees
Ø	1	90 degrees
1	Ø	180 degrees
1	1	270 degrees

c. Character Size

S_1	$S_ø$	Size	W × H (in addressable points)
Ø	Ø	1×	24 × 36
Ø	1	1.5×	36 × 54
1	Ø	2×	48 × 72
1	1	2.5×	60 × 90

Figure 5-12. Text Command Bit Pattern

4 PROGRAMMABLE CHARACTER SIZES:

1.0 × 56 characters per line, 29 horizontal lines possible.
1.5 × 37 characters per line, 19 horizontal lines possible.
2.0 × 28 characters per line, 14 horizontal lines possible.
2.5 × 22 characters per line, 11 horizontal lines possible.

Figure 5-13. 1345A Character Display Capabilities

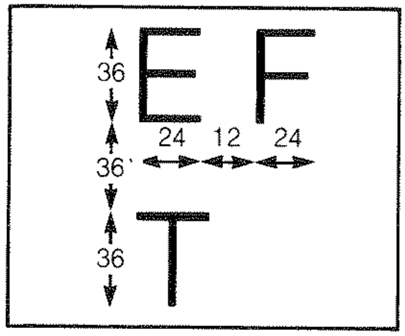


Figure 5-14. Example of 1× Character Spacing

The starting position of each character is the lower left corner of the defined character cell. After drawing a character, the 1345A advances to the starting point of the next character much like a typewriter would operate. The 1345A also contains many special characters that facilitate graphics and display annotation. Figure 5-15 contains the modified 1345A ASCII character set in HEX format. This HEX code is sent to the 1345A in the lower 8 bits of each text command.

1345A MODIFIED ASCII CODE CONVERSION TABLE

LEAST SIGNIFICANT CHARACTER	MOST SIGNIFICANT CHARACTER Ø	1	2	3	4	5	6	7
Ø		centered *	SP	Ø	@	P	`	p
1	HP logo	centered o	!	1	A	Q	a	q
2	β	↑	"	2	B	R	b	r
3		←	#	3	C	S	c	s
4	upper-half tic	↓	$	4	D	T	d	t
5	lower-half tic	→	%	5	E	U	e	u
6	left-half tic	√‾	&	6	F	V	f	v
7	right-half tic	π	'	7	G	W	g	w
8	back space	Δ	(	8	H	X	h	x
9	1/2 shift down	μ	)	9	I	Y	i	y
A	line feed	° (degree)	*	:	J	Z	j	z
B	inv. line feed	Ω	+	;	K	[	k	{
C	1/2 shift up	ρ	,	<	L	\	l	¦
D	carriage return	Γ	-	=	M	]	m	}
E	horizontal tic	θ	.	>	N	^	n	□
F	vertical tic	λ	/	?	O	—	o	►

EXAMPLES:

HP logo	=	Ø1
A	=	41
i	=	69
√‾	=	16
►	=	7F
line feed	=	Ø9

Figure 5-15. 1345A Modified ASCII Character Set

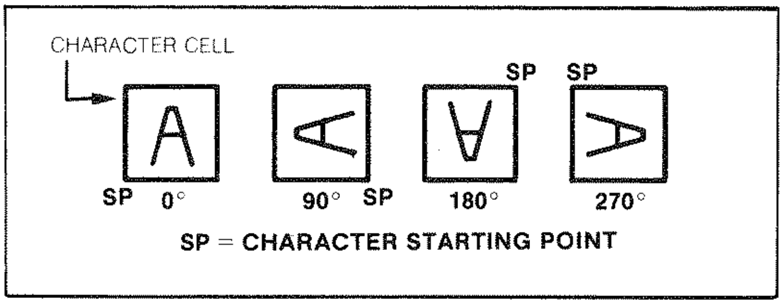


Figure 5-16. Character Rotation

Character rotation is an additional feature of the 1345A. The 1345A can be programmed to rotate any character at 0, 90, 180, or 270 degrees rotation measured counter clockwise from horizontal. This can be done for any character at any size. The starting point of the character is always the lower left corner relative to any rotation. For character rotation, the entire character area is rotated the specified number of degrees and the starting point moves around in a counter clockwise fashion. For example the starting point of a character rotated 180 degrees would be the upper right corner. This technique is illustrated in figure 5-16.

Since the starting point of the character changes with rotation, so does the direction of character spacing. If the rotation is 180 degrees, the characters will be written upside down from right to left. If the rotation mode is 270 degrees, the characters will advance from top to bottom. Rotation spacing examples are contained in figure 5-17.

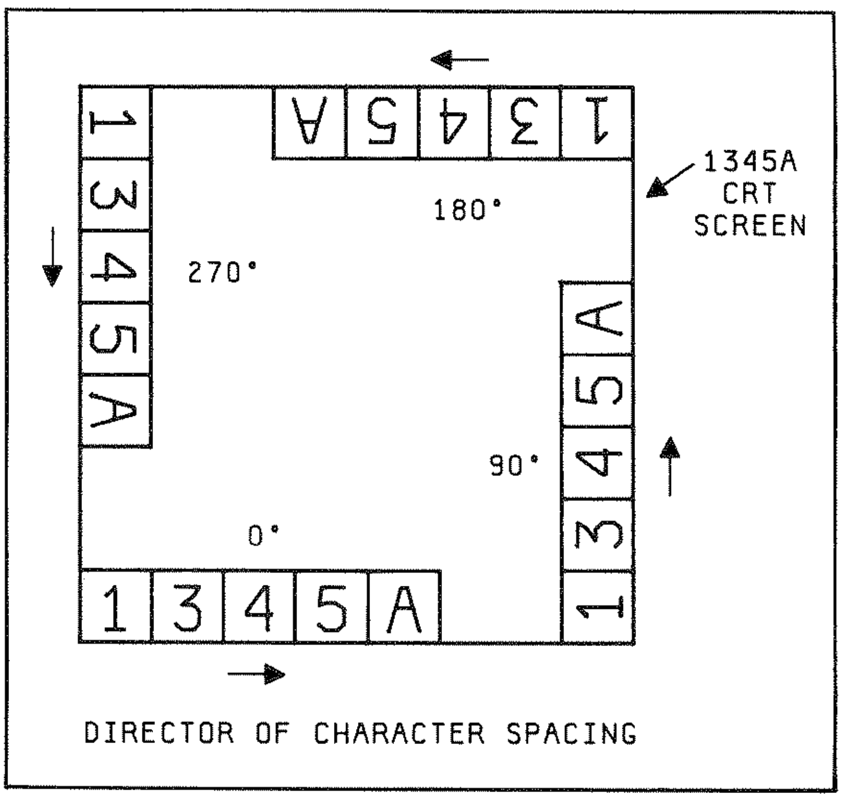


Figure 5-17. Character Rotation Spacing

Character generation on the 1345A has several capabilities that the user need be aware of. Certain characters that the 1345A is capable of drawing CANNOT be written within a certain distances of certain CRT screen boundaries. These characters are listed in Figure 5-18. The characters are referenced to the to the screen boundary at which the limitation occurs.

It is important to observe the recommended character boundary specifications, to avoid problems which might be encountered by writing at the screen edges. Figure 5-19 contains recommended limits for each character size at each screen edge. Failure to observe these limits may result in undefined results particularly when writing characters listed in figure 5-18. The user should plot all characters within these specified borders.

The user should not attempt to write any character along a screen edge. The character spacing guidelines in figure 5-19 allow ample spacing for characters of all specified sizes. Characters NOT specified in figure 5-18 may be written closer to the screen borders but it is not recommended.

BOUNDARY CHARACTERS

Left Boundary:	Ø7 "right-half tic"; Ø8 "back space"; ØE "horizontal tic"; ØF "vertical tic"; 1Ø "centered *"; 11 "centered o"; 41 "A"; 57 "W"; 5F "_"; 77 "w"
Bottom Boundary:	Ø2 "β"; Ø5 "lower-half tic"; Ø9 "1/2 shift down"; ØA "line feed"; ØF "vertical tic"; 1Ø "centered *"; 11 "centered o"; 19 "μ"; 1C "ρ"; 24 "$"; 28 "("; 29 ")"; 2C ","; 3B ";"; 51 "Q"; 5B "["; 5D "]"; 5F "_"; 67 "g"; 6A "j"; 7Ø "p"; 71 "q"; 79 "y"; 7B "{"; 7D "}"
Top Boundary:	Ø1 "HP logo"; ØB "inv. line feed"; ØC "1/2 shift up"; 16 "√‾"; 1A "° (degree)"; 24 "$"; 28 "("; 29 ")"; 38 "8"; 5B "["; 5D "]"; 7B "{"; 7D "}"; 7E "□"
Right Boundary:	Ø1 "HP logo"; 16 "√‾"; 41 "A"; 51 "Q"; 57 "W"; 61 "a"; 71 "q"; 77 "w"; 7E "□"

NOTE: HEX character equivalents appear in quotation marks.

Figure 5-18. Boundary Characters

Wrap Around. The user needs to be aware of a phenomenon called "wrap around". If one or more vectors are drawn outside the vector drawing area, the display will draw vectors on opposite sides of the CRT. One part of the vector will be at one side of the screen while the other part of the vector will be drawn on the opposite side of the CRT. The picture will appear distorted with visible vectors connecting ends of the vectors This can be corrected by plotting inside the 1345A vector drawing area.

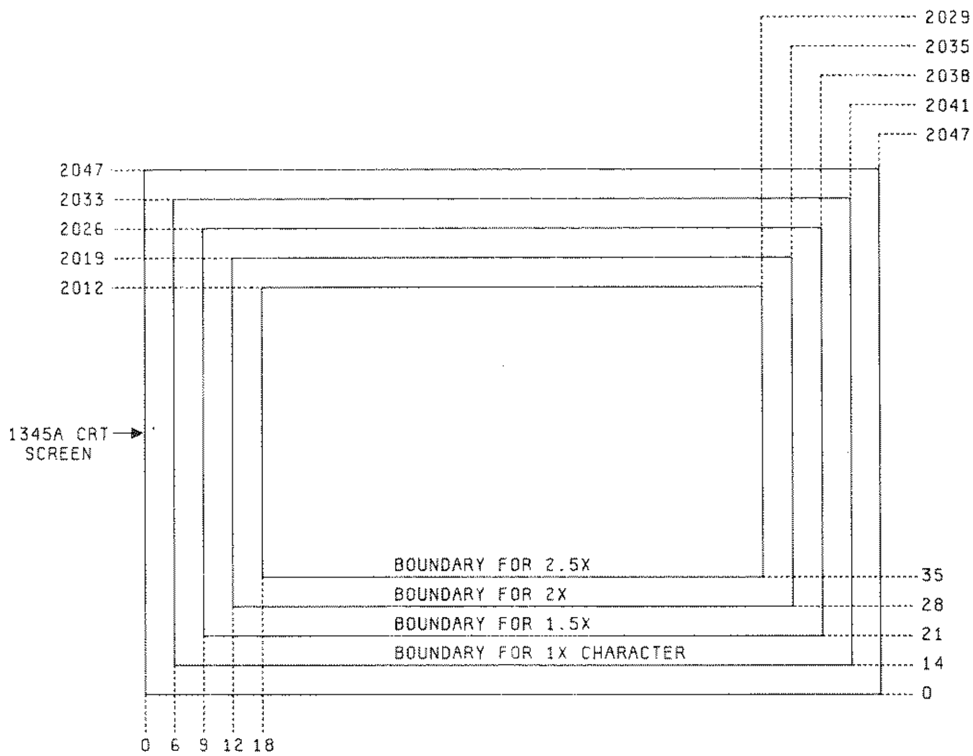


Figure 5-19. Character Borders

PROGRAMMING THE MEMORY OPTION (704).

The 1345A Memory Option stores up to 4k, 16 bit commands and refreshes the CRT thus relieving the user processor of data storage and CRT refresh requirements. The vector memory will appear to the user processor as a single memory location. The memory option recognizes two commands for programming. These commands are for data transfer and memory address pointer manipulation. A data transfer is either a read from or a write to the vector memory. Address pointer operations are used for positioning the data in the vector memory list and selecting a desired memory read address.

The vector memory contains a 4k by 16 bit memory, a 60 hz refresh timer, and two address pointers for accessing the memory. The timer is used to generate a refresh cycle of approximately 60 hz. This timer, when enabled will display the contents of the vector memory approximately once every 16.67 ms. There is a jumper on the memory board that allows the user to initiate the refresh cycle from an external source. This would be used to synchronize the refresh cycle with the user instrument data transfers or to refresh the display at a frequency other than 60 hz.

There are two pointers used to control access of data to and from the vector memory. One of these pointers is called the refresh pointer. It is enabled at the start of a refresh cycle and starts sequencing through vector memory until the end of memory is reached or an internal jump to 4095 is encountered. This is an internal memory address pointer that the user cannot access. The other pointer is called the Vector Memory Address pointer. This pointer is used to control data access to the vector memory. This pointer may be positioned by user commands for data transfer into and out of the vector memory list. In either case, an important fact is, that after a read or write operation the address of this pointer will increment by one.

Chapter 6

COMMAND/COMMUNICATION GROUP

PURPOSE OF THIS CHAPTER

The purpose of this chapter is to explain the bus-only control and communication group of commands. The topics covered here are:

Service Requests & Instrument Status	The status byte The instrument status register The activity status register Labeling user SRQs Power-on SRQ Reading sweep points Ready status Source fault status Reference locked status Measurement done status Missed sample status Overflow status Identify query Revision query Serial number query Setup state transfer HP-IB trigger enable Passing control Error code query
Reading Marker Values	X marker Individual special markers Grouped special markers
Communicating with the Front Panel	Key presses Reading Entry knob movement Reading Markers knob movement Writing to the message field Controlling display updating Reading auto carrier values Controlling HP logo for plotting

Most of the topics in this chapter are also discussed in condensed format in Appendix B, "Quick Reference Guide."

SERVICE REQUESTS AND INSTRUMENT STATUS

The service request (SRQ) is sent by the HP 3562A to gain the attention of the system controller. The SRQ is generated by conditions in the status byte (see the next section). When the HP 3562A issues an SRQ (activates the SRQ line), it also sets bit #6 in the status byte. This is the Require Service (RQS) bit, sometimes referred to as the "status bit" in connection with a poll.

An SRQ is sent for two general reasons: either the analyzer needs control of the bus, or there is some change in its internal status that the controller may be interested in.

The HP 3562A generates SRQs at three levels. First, true conditions in the status byte directly send the SRQ. Second, true conditions in the instrument status register (IS) indirectly generate SRQs through the status byte. Third, conditions in the activity status (AS) register indirectly generate SRQs through the IS then through the status byte.

Your controller's program doesn't necessarily have to be interrupt-driven. Any status condition or event capable of sending an SRQ can also be read directly. Table 6-1 summarizes the available status checks and how you can read them with a controller. The "Command" column shows the checks that have dedicated HP-IB commands.

Programming for Service Requests

In many applications, the controller program will be written so that it stops execution and polls all instruments on the bus when it receives an SRQ. A program written to perform serial polls dumps an entire status byte from each instrument and checks the status bit to detect which instrument requires service. When the instrument requesting service is identified, the reason for the SRQ can be found by decoding the status byte. Any unmasked status bits and conditions can initiate an SRQ. RESET and DEVICE CLEAR reset all masks in the status byte, instrument status and activity status registers.

As mentioned earlier, your program does not have to be interrupt-driven: every condition/event listed in table 6-1 can be read without waiting for an SRQ. The scheme you should take, waiting for interrupts or reading status checks, depends on your application.

Table 6-1 Summary of Status Checks in the HP 3562A

	Where/how to read it			
Condition/Event	**Status Byte**	**IS**	**AS**	**Command**
Requested service	*			
Error generated	*			ERR?
Ready for HP-IB commands	*			RDY?
User SRQs	*			
End of disc action	*			
End of plot action	*			
Power up	*			
Key pressed	*		KEY?	
Various plotter & disc requests	*			
Instrument status change	*		IS?	
Measurement pause		*		
Auto sequence pause		*		
End of measurement, capture or throughput		*		SMSD
Sweep point ready		*		SSWP
Channel 1 over range		*		SOV1
Channel 2 over range		*		SOV2
Channel 1 half scale		*		
Channel 2 half scale		*		
Source fault		*		SFLT
Reference locked		*		RLOK
Marker knob turned		*		
Entry knob turned		*		
Activity status change		*		AS?
System failure			*	
Filling time record			*	
Filters settling			*	
Curve fit in progress			*	
Missed external sample			*	SMSP
Timed preview active			*	
Data accepted			*	
Waiting for trigger			*	
Waiting for arm			*	
Ramping source			*	
Diagnostic in progress			*	
Marker calc in progress			*	
Identify				ID?
Revision				REV?
Send setup state				SET?

The Status Byte

The status byte is an 8-bit byte that provides information about the analyzer's current interaction with the bus. It provides 35 conditions, each with a unique code. All conditions are capable of generating SRQs. Some of the conditions can be masked, which prevents them from sending an SRQ, regardless of their current state. For example, if the "key pressed" condition is enabled (unmasked) and a key is pressed on the front panel, reading the status byte indicates that it was indeed the key pressed condition that generated the SRQ.

Table 6-2 shows the eight bits in the HP 3562A's status byte. The status byte is read by serial polling the analyzer (which also clears the status byte). Five of these bits are encoded; refer to table 6-3 for the condition codes.

Table 6-2 The HP 3562A's Status Byte

Bit	Value	Description
7	128	see table 6-3
6	64	RQS (HP 3562A requested service)
5	32	ERR (HP-IB error)
4	16	RDY (ready to accept HP-IB commands)
3	8	see table 6-3
2	4	see table 6-3
1	2	see table 6-3
0	1	see table 6-3

RDY (bit 4) is set when the instrument is ready to receive commands over the bus. This occurs when the command buffer is empty. The HP-IB command buffer has a capacity of three 80-byte command lines where a byte represents one character, and a line is defined to be terminated by a line-feed or activation of the EOI (End Or Identify) bus management line (carriage returns are ignored).

ERR (bit 5) is set when the instrument encounters an error condition and is cleared when the error register is read by the controller with the ERR? query command. Refer to "Error Codes" later in this section to decode the number returned with ERR?.

RQS (bit 6) is set when the analyzer activates the SRQ bus management line and is cleared when the controller serial polls the HP 3562A for its status byte.

Table 6-3 shows the condition codes represented by bits 7, 3, 2, 1 and 0 in the status byte.

Table 6-3 Status Byte Condition Codes

Status bit Numbers 7 3 2 1 0	Status Byte Value	Description
0 0 0 0 0	0	No service requested
0 0 0 0 1	1	User SRQ #1
0 0 0 1 0	2	User SRQ #2
0 0 0 1 1	3	User SRQ #3
0 0 1 0 0	4	User SRQ #4
0 0 1 0 1	5	User SRQ #5
0 0 1 1 0	6	User SRQ #6
0 0 1 1 1	7	User SRQ #7
0 1 0 0 0	8	User SRQ #8
0 1 0 0 1	9	End of disc action
0 1 0 1 0	10	End of plot action
0 1 0 1 1	11	Instrument status change
0 1 1 0 0	12	Power up
0 1 1 0 1	13	Key pressed
0 1 1 1 0	14	Device Clear Plotter, Listen HP 3562A
0 1 1 1 1	15	Unaddress Bus, Listen HP 3562A
1 0 0 0 0	128	Talk plotter, Listen HP 3562A
1 0 0 0 1	129	Talk disc execution, Listen HP 3562A
1 0 0 1 0	130	Talk disc report, Listen HP 3562A
1 0 0 1 1	131	Talk Amigo disc command, Listen HP 3562A
1 0 1 0 0	132	Talk Amigo disc data, Listen HP 3562A
1 0 1 0 1	133	Talk Amigo short status, Listen HP 3562A
1 0 1 1 0	134	Talk disc identify, Listen HP 3562A
1 0 1 1 1	135	Talk Amigo parallel poll, Listen HP 3562A
1 1 0 0 0	136	Listen Plotter, Talk HP 3562A
1 1 0 0 1	137	Listen disc command, Talk HP 3562A
1 1 0 1 0	138	Listen disc execution, Talk HP 3562A
1 1 0 1 1	139	Listen Amigo disc command, Talk HP 3562A
1 1 1 0 0	140	Listen Amigo disc data, Talk HP 3562A
1 1 1 0 1	141	Listen Amigo disc read, Talk HP 3562A
1 1 1 1 0	142	Listen Amigo disc write, Talk HP 3562A
1 1 1 1 1	143	Listen Amigo disc format, Talk HP 3562A

Condition 0 indicates that no service was requested and it was not the HP 3562A that sent the SRQ. Conditions 1—8 are the eight USER SRQ softkeys (see "Labeling User SRQs" later in this section). Condition 9 indicates that disc action under the analyzer's control is finished; 10 shows the same thing for a plotter. Condition 11 is the "window" into the instrument status (IS) register; any change in the IS register sets this condition. Condition 12 is set if the PwrSRQ ON OFF softkey (in the **SPCL FCTN** menu) is ON and power is applied to the analyzer. Condition 13 is set if key code monitoring is enabled and a key on the analyzer's front panel is pressed. Conditions 14, 15 and 128-143 are provided for controllers incapable of passing control; refer to "Passing Control" later in this chapter.

The status byte can indicate up to three conditions simultaneously:

1. Occurrence of an error with ERR (bit 5)
2. Readiness to accept more commands with RDY (bit 4)
3. One of the 32 other conditions (bits 7,3,2,1,0)

The analyzer remembers one status condition beyond the one shown in the status byte. For example, assume the power-on and key pressed conditions are both enabled, and you power on and press a key. If you then read the status byte, it indicates the power-on SRQ (which occurred first). Since reading the status byte this time clears it, reading it again shows the key pressed condition. This queuing applies only to conditions 1-12. When conditions 13-143 are set, they must be serviced and cleared before the analyzer can continue.

Masking the Status Byte

When a condition is "masked," it is prevented from generating an SRQ when it becomes true. At power-on, all conditions except the power-on SRQ are masked (disabled), but it is a good idea to explicitly mask and unmask conditions as needed. Masking a condition does not prevent it from occurring, nor does it prevent the condition code from being set. Table 6-4 summarizes status byte masking.

Table 6-4 Masking Status Byte Conditions

Condition	How to Mask
0	not maskable (never generates an SRQ)
1-8	not maskable
9-10	masked with SRQD; unmasked with SRQE
11	masked with ISMn, where n is decimal equivalent of the bits in the IS register to be unmasked. This bit is *completely* masked by sending ISM0.
12	masked with PSRQ0; unmasked with PSRQ1
13	masked with KEYD; unmasked with KEYE
14-15	not maskable
16 (RDY)	masked with RDYD; unmasked with RDYE
32 (ERR)	masked with ERRD; unmasked with ERRE
64 (RQS)	not maskable (never generates an SRQ)
128-143	not maskable

Conditions 9 and 10 are unmasked with SRQE (optional service request enable) and masked with SRQD (optional service request disable). Condition 11 is masked/unmasked indirectly with ISMn (instrument status mask). Refer to "Masking the IS Register" later in this section for details. The point here is that unmasking at least one bit in the IS register automatically unmasks condition 11 in the status byte. Condition 12 is masked by pressing PwrSRQ ON OFF to OFF (or sending PSRQ0 over the bus) and unmasked by pressing it ON (or sending PSRQ1 over the bus). Condition 13 is masked with KEYD (key code disable) and unmasked with KEYE (key code enable). Condition 16 (the RDY bit) is masked with RDYD (ready SRQ disable) and unmasked with RDYE (ready SRQ enable). Condition 32 is masked with ERRD (error SRQ disable) and unmasked with ERRE (error SRQ enable). Remember, to enable a condition, unmask it. To disable it and prevent it from sending an SRQ, mask it.

SRQs are generated only by the status byte; the instrument status (IS) and activity status (AS) registers must generate SRQs indirectly through the status byte. The IS register can generate an SRQ if condition 11 in the status byte is enabled. The AS register is twice removed: bit 13 of the IS register and condition 11 of the status byte must be enabled for the AS to generate an SRQ.

The Instrument Status Register

Unlike the status byte, which shows the analyzer's current interaction with the bus, the instrument status (IS) register shows various conditions of the analyzer's internal status. The IS register does not generate SRQs (at least not directly). True conditions in the IS set condition 11 in the status byte, which in turn sends the SRQ.

Table 6-5 shows the instrument status (IS) register. The contents of the IS are read by sending the IS? command (which also clears the register). Unlike the status byte, the IS is not encoded: each bit represents a single condition/event. Remember that condition 11 in the status byte must be enabled (unmasked) before the IS can indirectly generate an SRQ.

Table 6-5 Instrument Status Register

Bit	Value	Condition/Event	
0	1	Measurement pause	
1	2	Auto sequence pause	
2	4	End of measurement, capture or throughput	
3	8	End of auto sequence	
4	16	Sweep point ready	
5	32	Channel 1 over range	
6	64	Channel 2 over range	1 = yes
7	128	Channel 1 half range	0 = no
8	256	Channel 2 half range	
9	512	Source fault	
10	1024	Reference unlocked	
11	2048	Remote marker knob turn	
12	4096	Remote entry knob turn	
13	8192	activity status register change	
14	16384	Power-on test failed	

Bit 0 is set when the measurement has been paused, either from the front panel or via HP-IB. Bit 1 is set when an auto sequence has been paused. Bit 2 is set when a measurement, capture, or throughput ends. For averaged measurements, this is at the completion of the last average. When averaging is off, it is set after each measurement. Bit 3 is set when an auto sequence is finished. Bit 4 is set when the analyzer is in the swept sine mode and a sweep point is ready. Bits 5—6 can be set only when a measurement, capture, or throughput is in progress. Bits 7—8 are set if the signal reaches half-range at least once during the measurement. Bit 9 indicates when a source fault occurs that causes the source to supply more than 12 volts. Bit 10 indicates whether the analyzer is locked to the external reference signal (at the EXT REF IN rear panel connector). Bits 11 and 12 indicate that the Markers and Entry knobs, respectively, have been moved. Bit 13 indicates a change in the activity status register. Bit 14 is set if the power-on self test fails.

Most of these bits have corresponding HP-IB commands. Bits 0 and 2 works with SMSD; refer to "Measurement Done Status" later in this section. Bit 4 works with SSWP; refer to "Sending Sweep Points" later in this section. Bits 5—6 work with SOV1 and SOV2I; refer to "Overflow Status" later in this section. Bit 9 works with SFLT; refer to "Source Fault Status" later in this chapter. Bit 10 works with RLOK; refer to "Reference Lock Status" later in this section. Bits 11 and 12 work with the remote knob commands; refer to "Communicating with the Front Panel" later in this chapter.

Masking the Instrument Status Register

Bits in the IS are masked with the ISMn command, where n is the decimal equivalent of the sum of the values of the bits to be unmasked. For example, the BASIC statement

```
OUTPUT 720;"ISM20"
```

unmasks bit 2 (value = 4) and bit 4 (value = 16), and masks all other bits. Remember that at least one bit in the IS must be unmasked to unmask condition 11 in the status byte. At power-on, the IS mask defaults to all bits masked. You can read the current masking of the IS register with the ISM? query:

```
OUTPUT 720;"ISM?"
ENTER 720;IS_mask
PRINT IS_mask
```

Bit 4 (sweep point ready) can also be masked with DSWQ (disable sweep SRQ) and unmasked with ESWQ (enable sweep SRQ). Bit 11 (remote marker knob turn) can be masked with RMKD (remote marker knob disable) and unmasked with RMKE (remote marker knob enable). Bit 12 can be masked with REND (remote entry knob disable) and unmasked with remote RENE (remote entry knob enable).

The Status Query (STA?)

The status query command (STA?) provides some information from both the status byte and the instrument status register. Sending STA? causes the HP 3562A to return the 16-bit word shown in table 6-6. Note that STA? does not clear the information shown in these bits.

Table 6-6 The STA? Word

Bit	Value	Condition/Event
0	1	Not used
1	2	Not used
2	4	Key pressed
3	8	Not used
4	16	RDY
5	32	ERR
6	64	RQS
7	128	Message on screen
8	256	Measurement pause
9	512	Auto sequence pause
10	1024	End of measurement
11	2048	End of auto sequence
12	4096	Sweep point ready
13	8192	Channel 1 over range
14	16384	Channel 2 over range
15	32768	Math overflow

The only unique information provided by STA? is the message on screen indicator (bit 7). This is set when a message is displayed in the message field on the screen. This field is the second line from the bottom on the right side. Messages appear in half-bright upper and lower case. To read the message, send the display message query command (DSP?), which returns up to 24 characters. See page 6-23. Here is a sample listing:

```
OUTPUT 720;"STA?"
ENTER 720;Status
```

The Activity Status Register

The activity status (AS) register indicates several aspects of the HP 3562A's current activity. It generates SRQs through the IS register, then through the the status byte. Unlike the status byte and IS, reading the AS register with AS? does not erase it. The AS register indicates events, as opposed to conditions. Consequently, it is possible to receive an SRQ caused by the AS, then find the register empty when you read it with AS?. Keep this in mind when programming for AS-based interrupts.

Table 6-7 shows the activity status (AS) register. The contents of the AS are read by sending the AS? command (which also clears the register). Unlike the status byte and like the IS, the AS is not encoded: each bit represents a single condition. Remember that *both* bit 13 of the IS and condition 11 of the status byte must be enabled before the AS can indirectly generate an SRQ.

Table 6-7 Activity Status Register

Bit	Value	Event
0	1	Check fault log
1	2	Filling time record
2	4	Filters settling
3	8	Curve fit in progress
4	16	Missed sample (when in external sample)
5	32	Timed preview
6	64	Accept data
7	128	Waiting for trigger
8	256	Waiting for arm
9	512	not used
10	1024	Ramping source
11	2048	Diagnostic in progress
12	4096	Marker calc in progress

Use these event indicators to monitor the analyzer's activity after assigning tasks to it. Bit 0 indicates that a system error inside the HP 3562A has been entered into the fault log. The fault log is intended for use by trained service people only; refer to the *HP 3562A Service Manual* for details. Bit 1 indicates that the time record is being filled, which becomes more noticeable as the frequency span decreases (increasing the time record length). Bit 3 indicates that a curve fit is in progress. Bit 4 indicates that a sample was missed while in external sampling because the external sampling frequency is too high. Bits 5 and 6 are used with previewing in the linear resolution mode. Bit 5 indicates that the analyzer is paused for a time preview, and bit 6 tells whether or not the last time record was accepted. Bits 7 and 8 indicate that the analyzer is waiting for the trigger signal or manual arming, respectively. Bit 9 indicates that the calibration routine is in progress. Bit 10 indicates that the source is being ramped. Bit 11 indicates that a service diagnostic is in progress. Finally, bit 12 indicates that a special marker calculation is in progress.

Masking the AS Register

Because it monitors events, the AS must be masked for the positive-going or the negative-going transition of each bit. Two commands are used to mask the AS register. ASMHn unmasks the bits equal to n as they change from low to high (0 to 1). ASMLn unmasks the bits as they change from high to low (1 to 0). The current masking of the AS can be read with the ASML? and ASMH? queries:

```
OUTPUT 720;"ASML?"
ENTER 720;ASM—low
OUTPUT 720;"ASMH?"
ENTER 720;ASM—high
PRINT ASM—low,ASM—high
```

As an example of AS masking, the BASIC statements

```
OUTPUT 720;"ISM8192"
OUTPUT 720;"ASML8"
```

detect when a curve fit currently in progress finishes. The ISM8192 unmasks two conditions simultaneously: by unmasking at least one bit in the IS, it unmasks condition 11 in the status byte (instrument status change); and by unmasking bit 13 in the IS, it allows changes in the AS register to be communicated to the IS register. The second statement, ASML8, unmasks bit 8 in the AS (curve fit in progress) for its transition from high to low. While the curve fit is in progress, bit 8 is high; as soon as the fit ends, bit 8 drops low. This in turn sets bit 13 in the IS, which then sets condition 11 in the status byte and sends the SRQ. The flowchart in figure 6-1 summarizes these actions.

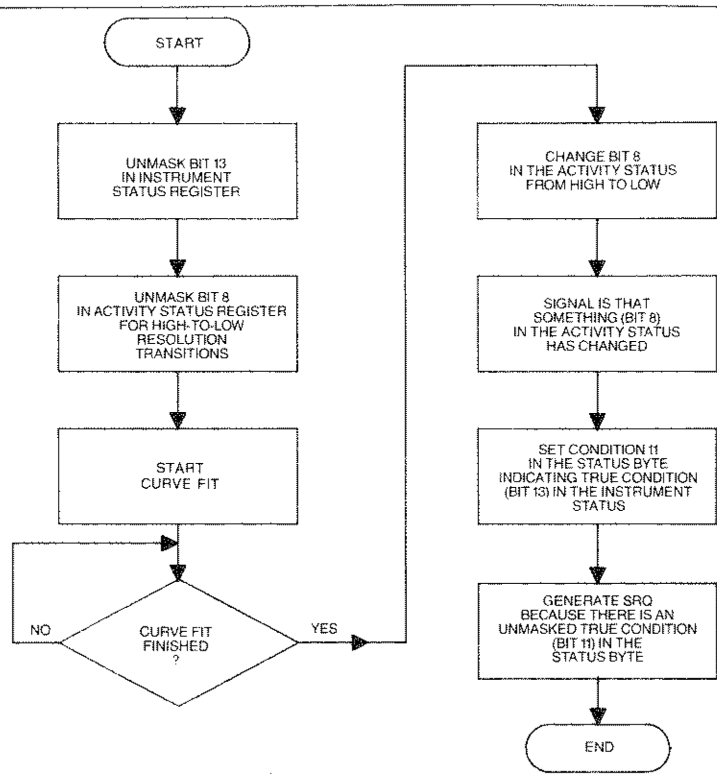


Figure 6-1 Example of Activity Status Masking

Labeling User SRQs

The HP 3562A offers a special class of interrupts called user SRQs. These allow you to initiate the SRQ whenever you want to, rather than depending on the device to issue one when it needs to. Under the HP-IB FCTN key, there is a softkey labeled USER SRQ. This softkey displays a menu containing the USER SRQ1 through USER SRQ8 softkeys. You can label each of these softkeys and individually detect the eight user SRQs. This feature has extensive implications: by utilizing the user SRQ softkeys, you can run the controller in the "background" while operating the analyzer from its front panel softkeys. You can create an entire menu structure by redefining the USER SRQ menu with the controller program. Labels are saved in nonvolatile memory and are not affected by power-down or preset.

To label the USER SRQ softkeys, use the LBS1—LBS8 commands. Labels can be one or two lines, with a maximum of six characters per line. The label must be enclosed in single or double quote marks, and if two lines are labeled, they must be separated by a comma. For example, the BASIC statement:

```
OUTPUT 720;"LBS4'TWO,LINES'"
```

labels the USER SRQ4 softkey as

```
TWO
LINES
```

Labels can contain letters, numbers, and any punctuation that does not affect command syntax. Lines with fewer than six characters are automatically centered. Refer to "The Status Byte" earlier in this chapter for handling the SRQs generated by user SRQs. An example program written in BASIC 3.0 that labels and handles all eight user SRQs is provided in the Introductory Programming Guide in Appendix A.

The Power-On SRQ

By setting the PwrSRQ ON OFF softkey in the SPCL FCTN menu to ON, you can command the HP 3562A to send an SRQ when it is powered on. The state of PwrSRQ ON OFF is saved in nonvolatile memory in the analyzer, so it is not affected by power-down or reset. The power-on SRQ is detected as condition 12 in the status byte; see "The Status Byte" earlier in the chapter for information on decoding the status byte.

Reading Sweep Points (SSWP)

When the HP 3562A is measuring in the swept sine mode, you can read each sweep point via HP-IB. This allows you to write your own auto adjustments programs, for example. The send sweep point command (SSWP) is used in conjunction with the sweep point ready condition (bit 4) in the instrument status register.

The general procedure for reading the sweep points is:

1. Enable the sweep point ready bit in the instrument status register using ESWQ or enable the sweep point ready SRQ with ISM.
2. Start the sweep.
3. Wait for the SRQ interrupt.
4. Decode the status byte and instrument status register to verify that the sweep is indeed ready.
5. Tell the analyzer to send the sweep point using SSWP.
6. Go back to step 3 and wait for the next point.
7. Disable sweep point with DSWQ when finished.

An alternative to the interrupt-driven method is reading the IS register in a loop, which would replace steps 3 and 4. SSWP returns five variables, in the following order:

Input power

Output Power

Cross spectrum real part

Cross spectrum imaginary part

Frequency

The first four are floating point real variables, and frequency is long floating point. An example program written in HP BASIC 3.0 that reads sweep points and displays them on the controller's CRT is provided in the Introductory Programming Guide in Appendix A.

Ready Status Query (RDY?)

The ready status query (RDY?) indicates whether or the not the analyzer's HP-IB command buffer is full. It returns either a 1 (buffer is empty) or a 0 (buffer has some commands). The HP 3562A always returns a 1 in response to the RDY? query. Use the RDY bit in the status byte if you need to monitor the command buffer.

The HP-IB command buffer can store three lines of 80 bytes each. A line is defined to be terminated by a line feed command or activation of the EOI bus management line. Commands can be queued in the buffer, and they are processed as soon as they are received. When the buffer is full, the HP-IB handshaking sequence forces the controller to wait.

Source Fault Status (SFLT)

The source fault status query (SFLT) returns a 1 if a failure in the source is causing it to supply over 12 volts. A 0 is returned when the source level is in its normal operating range.

Reference Locked Status (RLOK)

The reference locked status command (RLOK) indicates whether or not the analyzer is locked to an external reference signal (applied to the EXT REF IN rear panel connector). A 1 is returned if it is locked, a 0 if not. This command provides the same information as bit 10 in the instrument status register.

Measurement Done Status (SMSD)

The measurement done status command (SMSD) indicates if a measurement, capture or throughput is in progress. SMSD returns a 1 if the measurement, capture or throughput is done and a 0 if it is still in progress.

Missed Sample Status (SMSP)

The missed sample status command (SMSP) indicates if the analyzer missed a sample while in external sampling. This is caused by an external sample rate greater than 256 kHz.

Overflow Status (SOV1, SOV2)

The send overflow status commands (SOV1 and SOV2) return a 1 if an overrange occurred in the last measurement, and a 0 if not. These flags are set only during a measurement and are cleared only by reading.

Identify Query (ID?)

This query (ID?) is used to identify devices on the bus. The HP 3562A responds to ID? by returning the 7-character string "HP3562A."

Revision Query (REV?)

This query (REV?) identifies the revision code of the software contained and the instrument and code and format convention revision to which the software is written. For example, the BASIC statements:

```
OUTPUT 720; "REV?"
ENTER 720; Software, Format
PRINT Software, Format
```

Provide the software and format codes.

Serial Number Query (SER?)

This command is a partial implementation of the serial number query. The HP 3562A responds to it returning a 10-character string: prefix (4 numbers indicating the date of the analyzer's introduction), country of manufacture (A for USA), and 5 zeros. Individual instrument serial numbers are not provided (the 5 zeros are returned instead).

Setup State Transfer (SET, SET?)

The SET? command dumps the current instrument state in the ANSI floating point format. The SET command loads a state that has been previously dumped with SET? back into the analyzer. SET? is interchangeable with the DSAN (Dump State in ANsi) command, and SET is interchangeable with the LSAN (Load State in ANsi) command. Please refer to Chapter 3 for information on using DSAN and LSAN.

HP-IB Trigger Enable (HPT)

In addition to the triggering modes selectable from the front panel, the analyzer can also be triggered via HP-IB. To do this, you first need to select HP-IB triggering by sending the HPT command. Once HPT is sent, the analyzer can respond to the HP-IB bus management command "TRIGGER."

Passing Control

The HP 3562A is capable of controlling the bus so that it can control plotters, access disc drives and output command strings. When it needs control of the bus, the most efficient method is to:

1. Send the CTAD command (controller address) to the HP 3562A telling it where to pass control back to when it is finished.
2. Send the command that requires the analyzer to have control of the bus; STPL (START PLOT), for example.
3. Wait for the analyzer to issue an SRQ saying its needs control of the bus.
4. Pass control to the analyzer. This is a controller-dependent operation; HP BASIC 3.0 provides the PASS CONTROL command for this purpose.
5. Wait for the HP 3562A to send a second SRQ saying it is finished with the bus. You can, of course, have the controller continue its program without waiting for the analyzer to release control of the bus if regaining control, is not important. In any case, the HP 3562A automatically passes control back to the controller specified by CTAD when it no longer needs it.

The Introductory Programming Guide in Appendix A provides two examples of passing control, one for plotter control and one for sharing a disc drive with a controller.

If your controller is incapable of passing control, use status byte conditions 14, 15 and 128-143 to detect when each device on the bus needs to talk and listen. Then explicitly address and unaddress each device as needed to complete the data transfer. Your controller's HP-IB documentation should explain its use of the HP-IB secondary commands needed to do this.

A troubleshooting hint: if your controller grabs control of the bus before the HP 3562A is finished, see if some other device on the bus is sending an inadvertent SRQ. Such an SRQ causes the controller to immediately retake control of the bus.

Time-Out Control

To enable time-out control, send TMOE. This causes the HP 3562A to abort bus activity if it has control and a device under its control does not respond to a command after ~5s. To disable time-out, send TMOD. TMOE is the default.

Error Codes

The Error query (ERR?) returns the error code of the last HP-IB error. Each error code has a corresponding description in table 6-9. Note that these are the same errors as those encountered in front panel operation. For complete descriptions, with suggested corrective actions, refer to Appendix B of the *HP 3562A Operating Manual.*

Table 6-9 Error Codes

Code	Error	Code	Error
100	No Peak Avg in HIST Meas	200	Not Active Softkey
101	No Peak Avg in CORR Meas	201	Unknown Mnemonic
102	Freq Resp, No 1 Ch Demod	202	Line Too Long
103	Cross Corr, No 1 Ch Demod	203	Command Too Long
104	No fundamental	204	Alpha Delimiter Expected
105	X Marker Must Be Active	205	Not A Valid Terminator
106	Buffer Overflow	206	Extra Chars In Command
107	No Coord Change Allowed	207	Function Inactive
108	Not In Frequency Domain	300	Missing Input
109	No Data		
		301	Not Valid Units
110	Measurement In Progress	302	Not A Valid Number
111	Trace Not Compatible	303	Alpha Too Long
112	Data Type Incompatible	304	Number Too Long
113	Data Blocks Incompatible	305	Out Of Range
114	Source Block Empty		
		306	Unable To Curve Fit
115	User Display Not Enabled	307	Bad # Of Parameters
116	No Active Display Buffer	308	Auto Carrier Selected
117	Recursive Call	309	ENTRY Not Enabled
118	Not A Valid Auto Math		
119	Bad Setup State	400	Not A Valid Block Length
		401	Not A Valid Block Mode
120	Bad Auto Sequence Table	402	Not HP-IB Controller
121	Bad Synth Table	403	HP-IB Time Out
122	Bad Non-Volatile State	500	Bad Plotter Data Read
123	Bad Data Block	600	Cannot Recall Throughput
124	Bad Data Header		
		601	Not A Valid Catalog
125	Marker Not On	602	Unformatted Disc
126	No Valid Marker Units	603	Catalog Full
127	No Capture Data	604	Not A Valid Name
128	No Thruput Data	605	Not A Valid Display
129	Thruput Data Too Long		
		606	File Not Found
130	Bad Curve Fit Table	607	Disc Full
131	Bad Capture	608	Disc Reject
132	Bad Thruput	609	Recall Active Auto Seq
133	Not A Valid User Window	610	Unknown Disc Command Set
134	Bad Primitive Block		
		611	No Disc In Drive
135	View Input Disabled	612	Disc Write Protected
136	Cannot Use Zoom Data	613	Disc Fault
137	Already Running	614	Disc Transfer Error
138	May Be Inaccurate	615	No Spares Or Fault Areas
139	Cannot Be Complex		
		616	No Thruput File
140	Bad Delete Freq Table	617	Catalog Not In Memory
141	Loops Nested Too Deep	618	File Size Not Specified
142	Demod In Zoom Only	619	Select Capture To Recall
143	Numeric Overflow	620	Source = Destination
144	Invalid: Nyquist/Nichols		
		621	Sector Size < > 256 Bytes
145	Invalid: Log Data	622	Not Valid Format Option
146	No Carrier	623	Not Valid For This Disc
147	No Peak Hold In Time Avg	624	Destination Too Small
148	Calibration In Progess		
149	No Avg in Demod Hist		

READING MARKER VALUES

The HP 3562A allows you to read the X marker and the slope and power special marker functions via HP-IB. (The Y marker is not tied to display data, so there is little value in reading it over the bus.) This section explains the commands used for these functions and the data they provide. The Introductory Programming Guide in Appendix A has a example program in HP BASIC 3.0 that reads all three marker values. Note that before reading marker values, you should explicitly set the units and coordinates in which you want the trace to be calculated.

Reading the X Marker (RDMK)

The read marker command (RDMK) returns two long floating-point numbers: the x-axis ("X = ")and y-axis ("Ya = " or "Yb = ") values of the X marker. The following BASIC statements read the X marker:

```
OUTPUT 720; "RDMK"
ENTER 720; X,Ya
PRINT "X = ";X,"Ya = ";Ya
```

Reading the Special Marker Once (RSMO)

The read special marker once command (RSMO) returns the value of the POWER, FREQ & DAMP or AVG VALUE special marker function, whichever one was pressed last for each trace. This is a long floating point value and is scaled in the current display coordinates and units. The following BASIC statements read whichever of these marker functions is active:

```
OUTPUT 720; "RSMO"
ENTER 720; Marker__Vala, Marker__Valb
PRINT Marker__Val
```

Reading the Special Marker Group (RSMG)

The read special marker group command (RSMG) returns the value of the SLOPE, HMNC POWER, THD or SBAND POWER special marker function, whichever one is active for each trace. This is a long floating point value and is scaled in the current units and coordinates. The following BASIC statements read the SLOPE marker:

```
OUTPUT 720; "RSMG"
ENTER 720; Slope a, Slope b
PRINT Slope
```

COMMUNICATING WITH THE FRONT PANEL

The rest of this chapter shows you how to communicate with the analyzer's front panel: keys, eight softkeys, and two knobs. The end of this section shows you how to write messages to the message field and control display updating.

Key Codes

Each key and the eight generic softkeys are assigned key code. You can use these codes in two ways: monitor key presses by interpreting key codes, and simulate key presses by sending key codes to the analyzer.

There are four commands used with this feature. KEY? is a query that returns the key code of the last key pressed since power-up or reset (if KEYE has been sent previously). KEYn sends a key code, where n is the code from 1 to 70, to the analyzer. And there are two commands used for masking/unmasking the key pressed condition in the status byte. KEYD masks (disables) the condition, and KEYE unmasks it.

Table 6-10 lists the HP 3562A's key codes. Note that the eight softkey buttons have unique codes, but individual softkey labels do not. The code of the last key pressed (since power-up or reset) is returned by the KEY? command. Key presses are simulated by sending the analyzer the KEYn command, where n is the code of the key to be simulated. The key buffer holds the last three key presses. COM? returns the HP-IB command of the last key pressed (this is useful for detecting softkeys).

Table 6-10 Key Codes

Key Name	Code	Key Name	Code
No Key Pressed	0		
ENGR UNITS	1	Softkey 4	36
INPUT COUPLE	2	Softkey 5	37
TRIG DELAY	3	Softkey 2	38
HP-IB FCTN	4	Softkey 1 (top)	39
DISC	5	Softkey 3	40
SELECT TRIG	6	5	41
CAL	7	6	42
RANGE	8	4	43
AVG	9	Softkey 7	44
SELECT MEAS	10	Softkey 6	45
WINDOW	11	1	46
LOCAL	12	3	47
PLOT	13	2	48
SOURCE	14	MARKER VALUE	49
FREQ	15	– (negative sign)	50
MEAS MODE	16	BACKSPACE	51
START	17	Softkey 8 (bottom)	52
SPCL FCTN	18	VIEW INPUT	53
PRESET	19	0	54
MATH	20	, (comma)	55
SYNTH	21	. (decimal point)	56
AUTO SEQ	22	A	57

Table 6-10 (Continued)

PAUSE CONT	23	B	58
SAVE RECALL	24	A&B	59
Y	25	COORD	60
SPCL MARKER	26	MEAS DISP	61
HELP	27	ARM	62
AUTO MATH	28	SINGLE	63
CURVE FIT	29	UPPER LOWER	64
X OFF	30	STATE TRACE	65
X	31	UNITS	66
Y OFF	32	FRONT BACK	67
8	33	SCALE	68
9	34	UP arrow	69
7	35	DOWN arrow	70

Reading Entry Knob Movement

The rotary pulse generator (RPG) knob in the Entry group can be addressed via HP-IB. You can use the knob to generate SRQs or use it to send numeric values to the controller.

To set up the knob in the Entry group to generate SRQs as it is rotated, you need use the instrument status register. Bit 12 in this register is used to indirectly generate the SRQ; refer to "The Instrument Status Register" earlier in this chapter.

The knob has a numeric range of -32 768 to +32 767. To program its value, use the RENV command (remote entry knob value). To read its current value, use the RENV? query. The Entry knob has variable acceleration, which you set with the RENS (Remote Entry Knob Speed) command. RENS0, 32767, specifies fixed acceleration, and RENS1, 32767 specifies variable acceleration. Use RENE to enable remote entry, or REND to disable it.

Reading Markers Knob Movement

The Markers group knob can also be addressed via HP-IB. This knob uses bit 11 in the instrument status register. Addressing the Markers knob is similar to addressing the Entry knob; the difference is that the acceleration of the Markers knob is fixed. To program its value, use the RMKV command (remote markers knob value). To read its current value, use the RMKV? query. Send RMKE to enable remote markers, RMKD to disable. When remote markers are enabled, the X and Y marker values should not be set.

Writing to the Message Field

You can write messages up to 24 characters long to the displays message field. Use the DSP command and put the message string in single quotes. For example, the BASIC statement:

```
OUTPUT 720;"DSP'Hi Mom'."
```

Display "Hi Mom" (without quotes) in the message field. To read the message currently in the field, use the DSP? query, which returns an alphanumeric string up to 24 characters long. For example:

```
OUTPUT 720; "DSP?"
ENTER 720; Message$
PRINT Message$
```

Reads and prints the current message. When a measurement is started a "blank" message is displayed, which sets bit 7 of the STA? word.

Controlling Display Updating

Two commands are provided to enable/disable updating on the display. To disable updating, send the DSPD (display disable) command. To enable it, send DSPE (display enable). Note that once you send DSPD, updating is disabled until you re-enable it by sending DSPE or resetting the analyzer.

Reading Auto Carrier Values

The values calculated by the demodulation algorithm's auto carrier feature can be read via HP-IB. The command SACR (Send Auto Carrier) returns four values:

Auto carrier calculated for Channel 1
Auto carrier calculated for Channel 2
Phase offset removed from Channel 1
Phase offset removed from Channel 2

For example, the BASIC statements:

```
OUTPUT 720; "SACR"
ENTER 720; Carrier1, Carrier2, Phase1, Phase2
```

return the four values. These values are in floating point format.

Controlling the HP Logo for Plotting

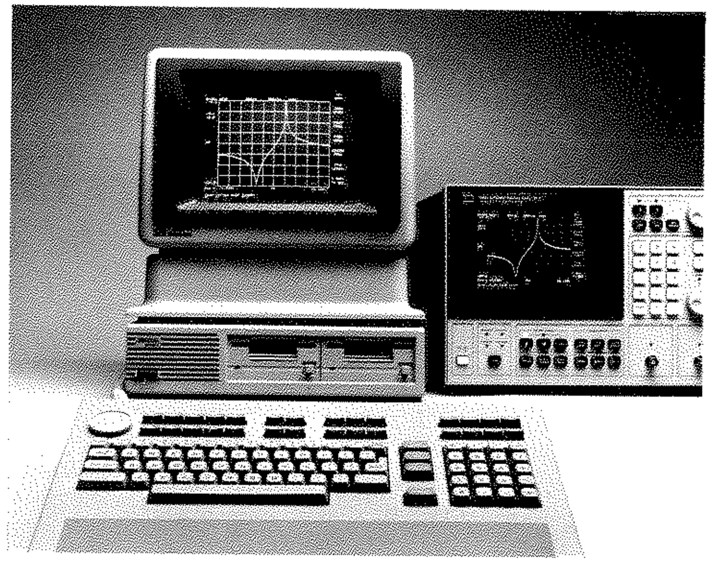
The HP logo that appears at the top of table displays is not normally plotted, but you can specify it to be plotted if desired. Send the command "LOGO0" to disable it or "LOGO1" to enable it.

Programming Guide

3562A/90000 SERIES 200-1

MAY 1985

Supersedes: None



INTRODUCTORY PROGRAMMING GUIDE
for the HP 3562A Dynamic Signal Analyzer with the HP 9000 Series 200 Desktop Computer (Basic)

INTRODUCTION

This is an introductory guide to programming the HP 3562A Dynamic Signal Analyzer using an HP 9000 Series 200 Computer with BASIC 3.0. This note has two purposes: showing you how to start programming the HP 3562A over the Hewlett-Packard Interface Bus (HP-IB) and providing simple example programs. It assumes you are familiar with the general features of the HP 3562A and some version of the BASIC language.

The topics discussed in this guide are:

The HP-IB concept—explains the advantages of remotely programming your analyzer with a computer

Reference information—where to turn for more information

Connecting the HP-IB system—how to get the computer and the analyzer to work together

The HP 3562A's command set—an overview of the analyzer's flexibility and power when programmed via HP-IB

Example programs—provides twelve short example programs to help you get started with programs of your own

HEWLETT PACKARD

THE HP-IB CONCEPT

In a general sense, programming an instrument via HP-IB simply replaces the instrument's front panel with commands sent by a computer or calculator. For example, instead of pressing the **START** key on the analyzer's front panel, you send the STRT mnemonic command via HP-IB. The immediate advantage is automation: your computer now controls the analyzer.

In most cases, HP-IB programmed operation emulates front panel operation. However, the HP 3562A provides additional functions available only over the bus. These allow you to transfer data in and out of the analyzer, write custom signal processing routines, control the vector display and communicate with various parts of the instrument.

This programming note shows you both aspects, front panel emulation and the special bus-only commands. Each step in the example programs is documented and explained, so use these programs to help understand the concept of HP-IB programming. And then try modifying the programs and view the results. All the HP-IB command mnemonics, with limits, terminators and syntax, are provided in the HP-IB Quick Reference Guide in Appendix B of the *HP 3562A Programming Manual.*

Reference Information

Complete operating and programming information for the HP 3562A is offered in these documents:

- HP 3562A Operating Manual (P/N 03562-90000)
- HP 3562A Programming Manual (P/N 03562-90030) This includes the HP-IB Quick Reference Guide for the HP 3562A.

The Hewlett-Packard BASIC 3.0 programming language for HP 9000 Series 200 Computers is covered in these documents:

- BASIC 3.0 User's Guide (P/N 98613-90040)
- BASIC 3.0 Programming Techniques (P/N 98613-90010)
- BASIC 3.0 Language Reference (P/N 98613-90050)
- BASIC 3.0 Interfacing Techniques (P/N 98613-90020)
- BASIC 3.0 Graphics Techniques (P/N 98613-90030)

The entire documentation package (consisting of eleven manuals) for BASIC 3.0 has the following part number:

- Manual Kit for BASIC 3.0 (P/N 98613-87901)

The HP 9000 Series 200 Computers have individual operating manuals.

CONNECTING THE HP-IB SYSTEM

This section shows you to connect the HP 3562A to the computer and verify that the system works.

Equipment Required

To run the example programs in this note, you need the following equipment:

- HP 3562A Dynamic Signal Analyzer
- HP 9000 Series 200 Computer
- HP 10833A/B/C/D HP-IB Cable

Setup

Begin by removing power from the HP 3562A and the computer.

1. As shown in figure 1, connect the HP 3562A to the computer using the HP-IB cable.

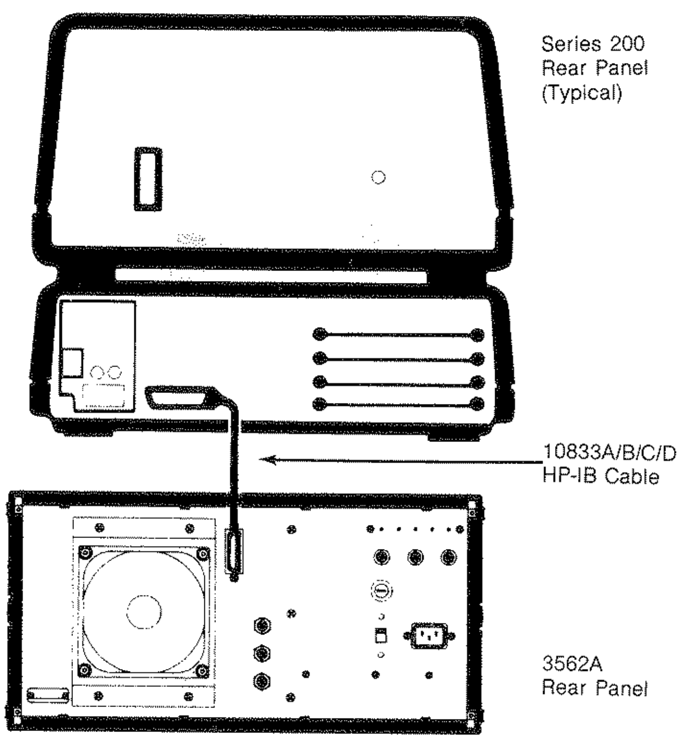


Figure 1 HP-IB System Connection

CAUTION

The HP 3562A has metric threaded HP-IB cable mounting studs, as opposed to English threads. Metric threaded HP 10833A/B/C/D cables must be used. Metric fasteners are colored black, while English fasteners are are colored silver. DO NOT attempt to connect black and silver fasteners, or damage to cable and instrument will result.

2. Apply power to the computer. If required, load the BASIC language operating system, following the instructions in the computer's operating manual. Note: with BASIC 3.0, you need to load at least the "HPIB" driver BIN and the "IO" language extensions BIN. You may also need the "DISC" and "FHPIB" drivers; consult the BASIC 3.0 *User's Guide* to see what your system needs. Note that you will probably need more language extensions as your programs get more complex than these simple examples.

3. Apply power to the HP 3562A.

4. The HP-IB address of the 3562A is stored in nonvolatile memory. It is a good idea to view the address after you power-up to see its current value. Press the **HP-IB FCTN** key in the HP 3562A's HP-IB group. When its softkey menu appears, press the SELECT ADDRES softkey. This displays a menu containing the HP-IB ADDRES softkey. Press this softkey to display the analyzer's current address in the lower left corner of the display. If you want to enter a new address, use the **0-9** numeric keypad and the ENTER softkey. The examples in this note use address 20 for the HP 3562A.

Check-Out

To verify proper system connection, type in the following command on the computer:

```
REMOTE 720
```

and press the computer's [Return], [EXECUTE], or [EXEC] key (depending on the type of keyboard you have). After you have issued the remote command, the green REMOTE indicator LED on the HP 3562A's front panel should light up. This indicates that the analyzer is now under the remote control of the computer. Try pressing some keys on the HP 3562A to verify that the keyboard is disabled. If you want to return local (front panel) control to the analyzer, press the **LOCAL** key in the HP-IB group. You can also issue the "LOCAL 7" command from the computer.

If the REMOTE indicator does not light up, recheck the system connection as explained earlier in "Setup." If this does not solve the problem, the computer, the analyzer or the cable may be at fault. Once you have successfully verified the system connection, you are ready to run some example programs.

THE HP 3562A'S HP-IB COMMAND SET

The HP 3562A provides HP-IB codes to emulate front panel operation, as well as a number of "bus-only" commands to perform special operations via HP-IB. The HP-IB command set is divided into five groups:

1. **Front panel**—provides HP-IB commands for the keys and softkeys.
2. **Data Transfer**—allows you to dump and load data traces and instrument states.
3. **Signal Processing**—provides signal processing primitives that allow you to operate on the blocks of data inside the analyzer.
4. **Display Control**—gives you complete access to the vector display for programming custom text and graphics displays.
5. **Command/Communication**—provides service requests, instrument status, special queries and communication with the front panel.

In addition to front panel emulation examples, this programming note provides examples from the data transfer, display control, and command/communication groups. Instructions for using all the commands in these groups are provided in the *HP 3562A Programming Manual.*

PROGRAMMING EXAMPLES

This section provides twelve example programs that show you the power and flexibility provided by the HP 3562A combined with an external controller. These programs are written in BASIC 3.0 for HP 9000 Series 200 computers.

Example #1: Setting Up a Measurement

This program shows you how to set up several basic measurement parameters. It sets up and starts a frequency response measurement in the linear resolution mode using commands accessible from the front panel. This is the type of programming used to emulate front panel operation.

```
10    ! Example 1
20    ! Setting up a linear resolution measurement
30    !
40    Dsa=720                      ! Address of 3562A
50    !
60    OUTPUT Dsa;"RST"             ! Reset 3562A
70    OUTPUT Dsa;"LNRS"            ! Linear resolution mode
80    OUTPUT Dsa;"FRSP"            ! Frequency response meas
90    OUTPUT Dsa;"FRS10KHZ"        ! Span at 0-10 kHz
100   OUTPUT Dsa;"RND"             ! Random noise source
110   OUTPUT Dsa;"SRLV5V"          ! Source level at 5V
120   OUTPUT Dsa;"HANN"            ! Hanning window
130   OUTPUT Dsa;"NAVG10ENT"       ! 10 averages
140   OUTPUT Dsa;"STBL"            ! Stable averaging
150   OUTPUT Dsa;"RNG5V"           ! Range 5V both channels
160   !
170   ! Measurement is set up and ready to start
180   !
190   OUTPUT Dsa;"STRT"            ! Start the measurement
200   OUTPUT Dsa;"FRQR"            ! Frequency response display
210   END
```

Line 40 sets the analyzer's HP-IB address (700 for the HP-IB interface plus 20 for the HP 3562A) equal to the variable "Dsa." Lines 60 through 150 set up individual parameters: reset, mode, measurement, frequency span, noise source, source level, number of averages, stable averaging and input range. (To set up any measurement over the HP-IB, you merely program each variable, just as you do from the front panel.) When the setup is ready, line 190 starts the measurement. Note that this program could have been compressed to a few lines, but it was written this way to illustrate each step.

Example #2: Interactive Swept Sine

This program makes a swept sine measurement that sends each measurement point to the computer. The HP 3562A sends five variables at each sweep point: frequency, input power, output power, cross spectrum real, and cross spectrum imaginary. You could use a program like this one to monitor a sweep and operate on each point as it becomes available. This program merely displays the result at each point, but you could use it for the data collection portion of another program. (You may need to modify the data display portion to use this with small screens.)

```
10    ! Example 2
20    ! Setting up a swept sine measurement and
30    ! sending each sweep point to the controller
40    !
50    Dsa=720                    ! Address of 3562A
60    !
70    OUTPUT Dsa;"ESWQ"          ! Enable sweep SRQ
80    OUTPUT Dsa;"FRSW"          ! Swept sine preset
90    OUTPUT Dsa;"SRLV1V"        ! Source level at 1V
100   OUTPUT Dsa;"SRON"          ! Source on
110   OUTPUT Dsa;"RNG1V"         ! Range 1V; both channels
120   OUTPUT Dsa;"RES25P/DC"     ! Resolution 25 pts/dec
130   !
140   ! Sweep is set up; ready to start
150   !
160   PRINT USING "5X,4A,12X,5A,5X,6A,4X,6A,4X,6A";"Freq","PwrIn","PwrOut","CrSpRe","CrSpIm"
170                              ! Set up table on CRT
180   OUTPUT Dsa;"STRT"          ! Start the sweep
190   !
200   OUTPUT Dsa;"IS?"           ! Read IS register
210   ENTER Dsa;Is_byte          !
220   IF BIT(Is_byte,2) THEN 390
230                              ! Quit if sweep done
240   IF BIT(Is_byte,4) THEN 310
250   GOTO 200
260                              ! If sweep point ready,
270                              ! go on to send point.
280                              ! If not ready, go back
290                              ! and read IS register.
300   !
310   OUTPUT Dsa;"SSWP"          ! Send each sweep point
320   ENTER Dsa;I,O,R,Im,F       ! I = input power
330                              ! O = output power
340                              ! R = cross spec real
350                              ! Im = cross spec imag
360                              ! F = frequency
370   PRINT F,I,O,R,Im           ! Print the 5 values
380   GOTO 200
390   END
```

Lines 10 through 190 set up and start the sweep. Line 50 sets the HP 3562A's address to "Dsa." Line 70 enables the sweep service request (SRQ). This allows the analyzer to request service from the controller when each point is ready. Line 80 sends the F RESP SWEPT command to activate the swept sine preset. Lines 90 through 120 set up several sweep variables. Line 160 displays the table heading on the CRT. The analyzer and the computer are now ready to start, and line 180 sends the start command (equivalent to pressing the yellow **START** key.)

Lines 200 through 250 monitor the analyzer's instrument status register to detect when sweep points are ready. Line 200 requests the value of the HP 3562A's instrument status (IS) register (the contents of this register are fully described in programming manual). Line 210 enters the IS contents into the variable "Is_byte." Line 220 checks to see if bit 2 in the IS is set, indicating that the measurement is over; if yes, the program ends. If no, line 240 checks to see if bit 4 of the IS is set. This bit indicates when the sweep point is ready. If a point is ready, program control moves to line 310. If neither bit 2 nor bit 4 is set, line 250 loops the program back to line 200 to read the IS again. The program stays in the 200-250 loop until the sweep is finished or a point is ready.

Line 310 commands the analyzer to send the sweep point. One sweep point (consistig of five values) is sent each time bit 4 in the IS register indicates a point is ready. Line 320 enters the five variables passed at each point: input power (I), output power (O), cross spectrum real part (R), cross spectrum imaginary part (Im), and frequency (F). These values are printed by line 370. Line 380 then sends the program back to line 200 to wait for the next point. The END statement in line 390 is reached whenever bit 2 of the IS is set, indicating the measurement is finished (detected in line 220).

Example #3: User Service Requests (SRQs)

Service requests (SRQs) allow a device on the bus to request service from the system controller. When the controller receives an SRQ, it can then "poll" each device on the bus to see which one requested service. After identifying the device, the controller can then read the device's status byte to determine why it requested service. Reasons for requesting service can be anything from a printer being out of paper to an analyzer ending a measurement.

The HP 3562A offers a special class of interrupts called user SRQs. These allow you to initiate the SRQ whenever you want to, rather than depending on the device to issue one when it needs to. Under the **HP-IB FCTN** key, there is a softkey labeled USER SRQ. This softkey displays a menu containing the USER SRQ1 through USER SRQ8 softkeys. You can label each of these softkeys and individually detect the eight user SRQs. This feature has extensive implications: by utilizing the user SRQ softkeys, you can run the controller in the "background" while operating the analyzer from its front panel softkeys. You can create an entire menu structure by redefining the USER SRQ menu with the controller program.

This program shows you how to label the user SRQ softkeys and handle the SRQ interrupts. In a real program, you would direct the SRQ from a softkey press to perform a particular action. In this program, each softkey merely displays a CRT message.

```
10     ! Example 3
20     ! Labeling and handling user SRQs
30     !
40     Dsa=720                   ! Address of 3562A
50     !
60     OUTPUT Dsa;"LBS1""SOFT,KEY 1"""
70     OUTPUT Dsa;"LBS2""SOFT,KEY 2"""
80     OUTPUT Dsa;"LBS3""SOFT,KEY 3"""
90     OUTPUT Dsa;"LBS4""SOFT,KEY 4"""
100    OUTPUT Dsa;"LBS5""SOFT,KEY 5"""
110    OUTPUT Dsa;"LBS6""SOFT,KEY 6"""
120    OUTPUT Dsa;"LBS7""SOFT,KEY 7"""
130    OUTPUT Dsa;"LBS8""SOFT,KEY 8"""
140    !
150    ! The user SRQ softkeys are labeled
160    !
170    OUTPUT Dsa;"USRQ;LCL"     ! Display USER SRQ menu
180    ON INTR 7 GOTO 220        ! Look for interrupt
190    ENABLE INTR 7;2           ! Enable SRQ interrupts
200 Idle:   GOTO Idle            ! Wait for interrupt
210    !
220   Status_byte=SPOLL(Dsa)     ! Serial poll the 3562A &
230                              ! read its status byte
240   Status_byte=BINAND(Status_byte,15)
250                              ! Delete bits 4 - 7
260   !
270   ! Following lines identify the softkey pressed
280   !
290   IF Status_byte=1 THEN PRINT "SOFT KEY 1"
300   IF Status_byte=2 THEN PRINT "SOFT KEY 2"
310   IF Status_byte=3 THEN PRINT "SOFT KEY 3"
320   IF Status_byte=4 THEN PRINT "SOFT KEY 4"
330   IF Status_byte=5 THEN PRINT "SOFT KEY 5"
340   IF Status_byte=6 THEN PRINT "SOFT KEY 6"
350   IF Status_byte=7 THEN PRINT "SOFT KEY 7"
360   IF Status_byte=8 THEN PRINT "SOFT KEY 8"
370   GOTO 180                   ! Go back and wait
380                              ! for next interrupt
390    END
```

Lines 10 through 160 write custom labels in the eight USER SRQ softkeys. Line 40 sets the analyzer's address to the variable "Dsa." Lines 60 through 130 label the eight softkeys (SOFT KEY 1 through 8 in this case; you can use any combination of two six-character lines, upper or lower case).

Lines 170 through 210 display the USER SRQ menu and wait for you to press one of the eight softkeys. Line 170 tells the HP 3562A to display the USER SRQ menu then return local control. Line 180 instructs the computer to go to line 220 if it receives an interrupt over the HP-IB (interface 7). Line 190 enables SRQ interrupts, and line 200 forces the program to wait until it receives an interrupt. The BASIC statements ON INTR and ENABLE INTR simply tell the computer what kind of interrupts to accept and where to go if it receives them.

When the interrupt occurs, the program goes to line 220. This line serial polls the HP 3562A, requesting its status byte so the computer can determine why the analyzer requested service. The status byte is read into the variable "Status_byte." To make decoding easier, line 240 erases the bits in Status_byte that don't concern us here. (For user SRQs, we are interested only in the lower four bits of the status byte.) Lines 290 through 360 determine which softkey was pressed and displays an appropriate message. (At this point, you can the replace the individual messages with program routines that perform appropriate action for each softkey.) Line 370 sends the program back to line 180 to wait for the next softkey press. In this program, the END statement in line 390 is never reached.

Example #4: Reading Marker Values

The HP 3562A allows an external computer to query many aspects of its operation and status. One of these capabilities is sending current marker values (X- and Y-axis values of the X marker and the special marker values). This program shows how to read these values and enter them into the controller.

```
10    ! Example 4
20    ! Reading marker values and printing them
30    ! on the computer CRT
40    !
50    Dsa=720                   ! Address of 3562A
60    !
70    OUTPUT Dsa;"X"            ! Activate X marker
80    OUTPUT Dsa;"RDMK"         ! Read marker command
90    ENTER Dsa;X,Y             ! Read X- & Y-axis values
100   PRINT "X =";X,"Y =";Y     ! Display X and Y
110   !
120   OUTPUT Dsa;"PWR"          ! Special power marker
130   OUTPUT Dsa;"RSMO"         ! Read special marker once
140   ENTER Dsa;Power           ! Read power
150   PRINT "Power =";Power     ! Print power value
160   !
170   OUTPUT Dsa;"SLP"          ! Special slope marker
180   OUTPUT Dsa;"RSMG"         ! Read slope command
190   ENTER Dsa;Slope           ! Read slope
200   PRINT "Slope =";Slope     ! Print slope value
210   END
```

Lines 50 sets the analyzer's address equal to the variable "Dsa." Lines 70 through 110 read the X- and Y-axis marker values. Both these values are transmitted when the analyzer receives the Read Marker command (RDMK). Line 70 activates the X marker, then line 80 sends the RDMK command. Line 90 tells the computer to input two variables; the first is the X-axis value, and the second is the Y-axis value. Line 100 prints the two values.

Lines 120 through 160 read the power value. This function shows the total power in the active trace. Line 120 activates the function, then line 130 sends the read special marker once command (RSMO). This command reads the value of the last softkey pressed in the group of FREQ & DAMP, POWER and AVG VALUE. The value enters the computer at line 140 and is printed by line 150.

Lines 170 through 200 read the slope value. This function shows the slope at the marker position. This segment operates in the same manner as lines 120 through 160, except it uses the read special marker group command (RSMG). This command reads the value of the active softkey in the group of SLOPE, HMNC POWER, THD, MOD INDEX and SBAND POWER. Finally, line 200 ends the program.

This program shows marker reading in general. For your applications, you should explicitly set the display units before reading markers so that you know the units in which the marker values will be transmitted.

Example #5: Custom Graphics

The display group of bus-only commands allows you to write and draw on the analyzer's vector display. This is basically a three-step process: create and activate a display buffer, fill the buffer, then put the buffer on the display. A display "buffer" is simply an area in memory reserved for your display commands. Uses of custom graphics include providing operator messages and showing equipment connections.

This program shows two simple steps: first writing a message on the display, then drawing some simple boxes. You can expand these techniques to create almost any display.

```
10    ! Example 5
20    ! Drawing and writing on the 3562A display
30    !
40    Dsa=720                   ! Address of 3562A
50    !
60    OUTPUT Dsa;"COMD"         ! Disable command echo
70    OUTPUT Dsa;"DBSZ500,1,2"! Set up 2 500-word
80                              ! display buffers,
90                              ! #1 and #2
100   OUTPUT Dsa;"DBAC1"        ! Activate buffer #1
110   OUTPUT Dsa;"CHSZ2"        ! Character size=2
120   OUTPUT Dsa;"PU"           ! Lift pen
130   OUTPUT Dsa;"PA100,1000"   ! Move pen to left
140                             ! center of display
150   OUTPUT Dsa;"PD"           ! Put pen down
160   OUTPUT Dsa;"WRIT'Custom Graphics Display'"
170                             ! Write text into
180                             ! buffer
190   OUTPUT Dsa;"DBUP1"        ! Put up buffer #1
200   WAIT 1                    ! Pause for 1 second
210   !
220   ! Rest of this program draws on the display
230   !
240   OUTPUT Dsa;"DBAC2"        ! Activate buffer #2
250   !
260   T=1900                    ! Top of box
270   B=300                     ! Bottom
280   L=100                     ! Left side
290   R=1700                    ! Right side
300   !
310   FOR Box=1 TO 15           ! Draw 15 boxes
320     OUTPUT Dsa;"PU"         ! Move pen to
330     OUTPUT Dsa;"PA";L,B     ! bottom
340     OUTPUT Dsa;"PD"         ! left corner
350     !
360     OUTPUT Dsa;"PA";L,T     ! These 4 lines
370     OUTPUT Dsa;"PA";R,T     ! draw the box
380     OUTPUT Dsa;"PA";R,B     !
390     OUTPUT Dsa;"PA";L,B     !
400     !
410     OUTPUT Dsa;"DBSW2,1"    ! Switch buffers
420     OUTPUT Dsa;"DBAA2"      ! Lets us add to #2
430     !
440     T=T-50                  ! These four lines
450     B=B+50                  ! decrease size for
460     L=L+50                  ! the next box
470     R=R-50                  !
480     !
490   NEXT Box                  ! Draw next box
500   END
```

Line 40 sets the analyzer's address equal to the variable "Dsa." Line 60 disables command echoing, which significantly increases program speed. Line 70 creates two display buffers, #1 and #2, each 500 words long. Line 100 clears and activates buffer #1, which will be used to write the message. (Buffer #2 will be used to draw the boxes.) Line 110 sets the character size to 2 (your choices are 0 to 3). Line 120 lifts the "pen," allowing us to move the beam without drawing anything. (These commands—PU, PD, etc.—are the ones used by Hewlett-Packard plotters, hence the pen analogies.) Line 130 moves the pen (using the Plot Absolute command) to location 100,1000. The lower left corner is 0,0; the upper right corner is 2048,2048. Line 150 puts the pen back down so that line 160 can write the message "Custom Graphics Display ."

We have now written the message we want into display buffer #1; the next step is to put the buffer up on the display. This is accomplished with line 190. Line 200 waits for a second before going on to the second part of the program. To review the steps in this segment: create buffers, activate a buffer, fill it with graphics commands, then put it on the display. Figure 2 shows the display after the first segment is finished.

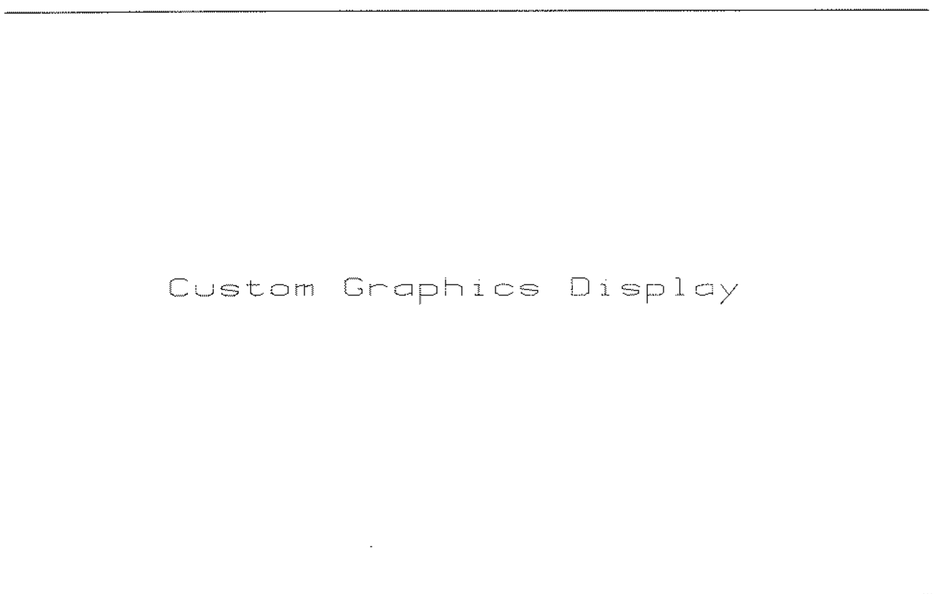


Figure 2 Message Display

Line 240 starts the drawing segment of the program by clearing and activating buffer #2. Buffer #1 is still displayed, however, because we haven't yet pulled it down from the display. Lines 260 through 290 initialize the four variables that define the top (T), bottom (B), left side (L), and right side (R) of the box. This is roughly the size of a single measurement display.

The FOR/NEXT loop from line 310 to line 490 draws fifteen boxes, each slightly smaller than the last. Lines 320 through 340 move the pen to the lower left corner of the box. Lines 360 through 390 draw the four sides of the box. After the box is complete, line 410 switches buffer #1 for #2. This takes buffer #1 (the message) down from the display and puts buffer #2 (the box) up on the display. Line 420 allows us to add more commands to buffer #2 without taking it down from the display. Lines 440 through 470 decrease the size of the box to draw another one inside the last one. Line 490 sends the program back to draw the next box.

Each time the program makes one loop a box is drawn, slightly smaller than the last box. The buffer switch command (DBSW2,1) in line 410 tells the analyzer to put buffer #2 back up on the display after each new box is drawn. Figure 3 shows the display after the fifteen boxes are drawn.

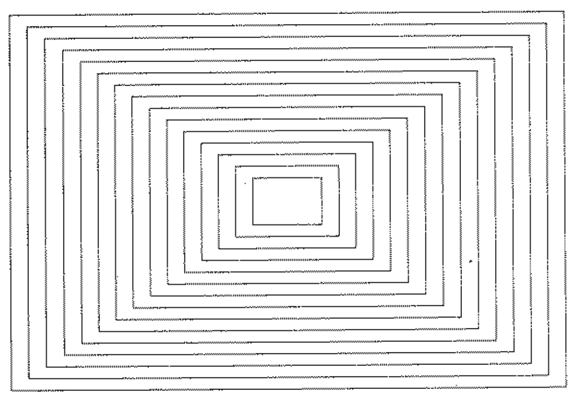


Figure 3 The Graphics Display

To get a good feel for graphics on the HP 3562A's display, try modifying lines in this program. For example, deleting the Pen Up command in line 120 causes the screen to draw a vector from the current position to 100,1000 when the Plot Absolute command in line 130 is executed. In the drawing segment, try changing such variables as the decrease factors in lines 440 through 470.

Example #6: Plotting with a Controller on the Bus

This program shows you how to pass control from the computer to the analyzer so it can plot the display. Because only one device can be in control of the bus at a time, you need to pass control to the HP 3562A when you want to plot. After the plot is finished, control is passed back to the computer. A program of this type is necessary when a controller is on the bus, and the HP 3562A needs to be the active controller: plotting, accessing a disc drive, or outputting command strings. This program uses service request (SRQ) interrupts to pass control back and forth.

```
10    ! Example 6
20    ! Plotting with a controller on the bus
30    !
40    Dsa=720                   ! Address of 3562A
50    OUTPUT Dsa;"CTAD21"       ! Controller address
60    ON INTR 7 GOTO Got_srq    ! When SRQ received,
70    ENABLE INTR 7;2           ! go to "Got_srq1"
75    OUTPUT Dsa;"SRQE"         ! Enables SRQs
80    OUTPUT Dsa;"STPL"         ! START PLOT command
90 Idle1:  GOTO Idle1           ! Wait for SRQ
100   !
110 Got_srq:                    ! Plotting has started
120   Erase=SPOLL(Dsa)          ! Erase SRQ flag
130   PRINT "Control passed to HP 3562A"
140   PASS CONTROL Dsa          ! Pass control to 3562A
150   !
160   ON INTR 7 GOTO Plot_done! Wait for plot to finish
170   ENABLE INTR 7;2           ! Enable SRQ interrupts
180 Idle2:  GOTO Idle2          ! Wait for interrupt
190   !
200 Plot_done:                  ! Plot is finished
210   Erase=SPOLL(Dsa)          ! Erase SRQ flag
220   PRINT "Control passed back to computer"
230   REMOTE Dsa                ! Put 3562A in remote
240   END
```

Line 40 sets the analyzer's address to the variable "Dsa." Line 50 sends the computer's address (21) to the HP 3562A so it knows where to pass control back to when it is finished plotting. If you want to pass control to another device, simply modify line 50 accordingly.

Lines 60 through 100 monitor for SRQ interrupts and tell the HP 3562A to start plotting. When the analyzer receives the START PLOT command from the computer, it generates an SRQ telling the computer it needs control of the bus before it can plot. Line 60 sends the program to "Got_srq" if an interrupt is detected on the bus. Line 70 enables only SRQ interrupts. Line 80 sends the START PLOT (STPL) command. Line 90 forces the program to wait until the SRQ is received. In this case, the wait is very short because the SRQ is received as soon as the HP 3562A reacts to the STPL command in line 80.

Lines 110 through 150 pass control to the analyzer after it tells the computer it needs control of the bus. Line 120 serial polls the HP 3562A to clear the SRQ. This is the only function of the serial poll in this example (the analyzer can't receive control until its SRQ is serviced). In a real program, however, you should interpret the status byte (which is read into the variable "Erase" here) to make sure that it was actually plotting that sent the SRQ. To keep this example simple, we assume it was. Line 130 displays a message on the CRT, and line 140 uses the BASIC 3.0 command PASS CONTROL to pass control to the analyzer.

In order to know when the analyzer is through with the bus, the computer must monitor for the "end of plot SRQ." The HP 3562A automatically passes control when it no longer needs it. This is done in lines 160 through 190. Line 160 sends the program to "Plot_done" when the plot done SRQ is received. Line 170 enables only SRQ interrupts (they must be re-enabled each time they are anticipated). Line 180 forces the program to wait until the SRQ is received. If the program wasn't forced to wait here, it would continue on without the computer having control of the system (which may or may not be desirable).

Lines 200 through 240 are optional but allow the program to exit with both the computer and the analyzer ready for more action. Line 210 clears the SRQ, as explained earlier. Line 220 lets you know that the computer is back in control of the bus. Line 230 puts the HP 3562A back into the remote mode.

This program can be modified to fit your particular application. For example, you can delete lines 50 and 80, and the computer will idle until you manually press the START PLOT softkey. This allows you to use the front panel and plot without physically removing the controller from the bus. Similarly, if you end the program at line 200, the HP 3562A stays in local mode after it is finished plotting. Complete information on using SRQ and the status byte is provided in Chapter 6 of the *HP 3562A Programming Manual.* In addition, the CTAD command (controller address) should be sent to specify the address to which the analyzer should pass control when it is finished. This is also explained in Chapter 6.

Example #7: Sharing a Disc Drive with a Controller

This program is a modified version of example #6 that allows you to share a disc drive with the system controller. The program passes control to the analyzer, has it save the display to disc, then passes control back to the computer.

```
10    ! Example 7
20    ! Disc accessing with a controller on the bus
30    !
40    Dsa=720                    ! Address of 3562A
50    !
60    ON INTR 7 GOTO Got_srq     ! When SRQ received,
70    ENABLE INTR 7;2            ! go to "Got_srq!"
75    OUTPUT Dsa;"SRQE"          ! Enables SRQs
80    INPUT "File name?",N$      ! Input file name
90    OUTPUT Dsa;"SAVF'";N$;"'"
100                              ! SAVE FILE command
110 Idle1: GOTO Idle1            ! Wait for SRQ
120   !
130 Got_srq:                     ! Disc access has started
140   Erase=SPOLL(Dsa)           ! Erase SRQ flag
150   PRINT "Control passed to HP 3562A"
160   PASS CONTROL Dsa           ! Pass control to 3562A
170   !
180   ON INTR 7 GOTO Disc_done!  Wait for disc to finish
190   ENABLE INTR 7;2            ! Enable SRQ interrupts
200 Idle2:  GOTO Idle2           ! Wait for interrupt
210   !
220 Disc_done:                   ! Disc is finished
230   Erase=SPOLL(Dsa)           ! Erase SRQ flag
240   PRINT "Control passed back to computer"
250   PRINT
260   PRINT "File '";N$;"' saved"
270   REMOTE Dsa                 ! Put 3562A in remote
280   END
```

The only differences between this program and example #6 are the commands needed to save files on disc. Line 80 asks you to enter the desired file name. Line 90 combines the SAVE FILE (SAVF) command with the file name you enter and sends this to the analyzer. Note the syntax of this command: it is necessary to combine the mnemonic (SAVF), the opening single quote ('), the alpha name (N$), and the closing single quote (') in exactly this arrangement. The semicolons (;) must be used to concatenate the various pieces of the command. The other difference is in the message displayed on the CRT by lines 240 through 260, which confirms that the file was saved under the name you entered.

Example #8: Dumping/Loading Data Traces

This program shows you how to dump a data trace from the analyzer. This dump is done in ANSI floating point format. Detailed instructions on all the HP 3562A's data transfer capabilities are provided into Chapter 3 of the *HP 3562A Programming Manual.* The last part of the program loads the data trace back into the analyzer so that you can verify both transfers.

```
10    ! Example 8
20    ! Dumping/loading the active trace
30    !
40    OPTION BASE 1             ! Set array base to 1
50    ASSIGN @Dsa TO 720        ! Address of 3562A
60    DIM Block(2500)           ! Set initial array
70    !
80    OUTPUT @Dsa;"DDAN"        ! Dump data command
90    ENTER @Dsa USING "#,2A,W";A$,Length
100                             ! Determine length
110   Float=Length DIV 8        ! Floating point data
120   REDIM Block(Float)        ! Resize array
130   !
140   ASSIGN @Dsa;FORMAT OFF    ! Disable ASCII format
150   ENTER @Dsa;Block(*)       ! Enter data
160   ASSIGN @Dsa;FORMAT ON     ! Enable ASCII format
170   !
180   PRINT "Data trace dumped to controller"
190   PRINT "Elements received:";Float
200   PAUSE                     ! Change trace now
210                             ! to verify transfer
220   !
230   OUTPUT @Dsa;"LDAN"        ! Load data command
240   OUTPUT @Dsa USING "#,2A,W";"#A",Length
250                             ! Data format & length
260   ASSIGN @Dsa;FORMAT OFF    ! Disable ASCII format
270   OUTPUT @Dsa;Block(*)      ! Output data
280   ASSIGN @Dsa;FORMAT ON     ! Enable ASCII format
290   !
300   PRINT "Data trace loaded back to 3562A"
310   END
```

Line 40 sets the option base for the array at 1. This was done to make the number of variables in the array more obvious. Line 50 assigns an I/O path to the analyzer at address 720. Line 60 dimensions a real array with a maximum size of 2500 values. (The number of variables dumped by the program varies according to the type of trace dumped; the program redimensions the array later.) Line 80 outputs the dump data command (DDAN = Dump Data in ANsi binary.) You can also dump data in ASCII and a special binary format; the programming manual has the details.

The data are dumped in two stages. The first stage consists of four bytes: two byte specifying the data format and two bytes specifying the amount of data that will be transferred. The format specifier (#A for ANSI data) is ignored in this case, and the length word is read into "Length." This stage is accomplished by lines 80 and 90.

The second stage starts with line 110. Since this transfer is in floating point format, eight bytes will be coming over for each value. We want each value in the array to represent one complete value from the analyzer, so line 110 divides "Length" by 8 to produce "Float." (Floating point values are 8 bytes long.) The value of "Float" is then used to redimension (line 120) the array "Block" into which the data will be read.

Line 140 disables the ASCII formatter in the computer. If this is not done, the computer takes every two byte and converts them to one ASCII character, rather than waiting for eight bytes. Line 150 completes the second stage of the data transfer. The (*) after the array name simply tells the computer to use the entire array; remember we just resized the array (line 120) to fit our data length. Line 160 re-enables the ASCII formatter.

The PAUSE in line 200 was inserted so you can change the trace on the display and then load the trace just dumped back into the analyzer. This is quick indication that both transfers worked.

The load section of the program starts at line 230, which outputs the load data command. Line 240 then outputs the #A and the length word. Lines 260 through 280 output the data stored in the array "Block."

Example #9: Dumping/Loading Instrument States

This program is similar in concept to example #8: it dumps the instrument state to the controller then allows you to load into back into the analyzer later. The "instrument state" is the collection of setup parameters that define the analyzer's current operating state. If you want to view the instrument state display, press the **STATE TRACE** key until the state is displayed.

The method in this program is the same as that in example #8. The differences are in the dump and load commands used and the size of the array. Note that the array size is fixed for the instrument state.

```
10    ! Example 9
20    ! Dumping/loading the instrument state
30    !
40    OPTION BASE 1              ! Set array base to 1
50    ASSIGN @Dsa TO 720         ! Address of 3562A
60    DIM Block(100)             ! Set initial array
70    !
80    OUTPUT @Dsa;"DSAN"         ! Dump data command
90    ENTER @Dsa USING "#,2A,W";A$,Length
100                              ! Determine length
110   Float=Length DIV 8         ! Floating point data
120   REDIM Block(Float)         ! Resize array
130   !
140   ASSIGN @Dsa;FORMAT OFF     ! Disable ASCII format
150   ENTER @Dsa;Block(*)        ! Enter state
160   ASSIGN @Dsa;FORMAT ON      ! Enable ASCII format
170   !
180   PRINT "Instrument state dumped to controller"
190   PRINT "Elements received:";Float
200   PAUSE                      ! Change state now
210                              ! to verify transfer
220   !
230   OUTPUT @Dsa;"LSAN"         ! Load state command
240   OUTPUT @Dsa USING "#,2A,W";"#A",Length
250                              ! Data format & length
260   ASSIGN @Dsa;FORMAT OFF     ! Disable ASCII format
270   OUTPUT @Dsa;Block(*)       ! Output state
280   ASSIGN @Dsa;FORMAT ON      ! Enable ASCII format
290   !
300   PRINT "Instrument state loaded back to 3562A"
310   END
```

Example #10: Power-on Service Requests

As explained in example #3, instruments (and sometimes operators) can send service requests (SRQs) to the system controller. One of the SRQs provided by the HP 3562A is the power-on SRQ. As its name implies, this requests service when the analyzer is powered on.

This program forces the computer to wait until the analyzer is turned on, then displays a message indicating so. In a real program, you would substitute the the message with the appropriate service routine.

```
10     ! Example 10
20     ! Handling a power-on SRQ
30     !
40     Dsa=720                    ! Address of 3562A
50     !
60     ON INTR 7 GOTO Got_srq     ! Look for interrupts
70     ENABLE INTR 7;2            ! Enable SRQ interrupts
80     PRINT "Waiting for power-on ..."
90 Idle:   GOTO Idle              ! Wait for interrupt
100    !
110 Got_srq:                      ! Received SRQ
120    Status_byte=SPOLL(Dsa)     ! Serial poll 3562A and
130                               ! read its status byte
140    Status_byte=BINAND(Status_byte,15)
150                               ! Mask bits 4-7
160    IF Status_byte=12 THEN Go_on
170                               ! Continue if power-on
180                               ! SRQ falg is set
190    GOTO 60                    ! Interrupt was not
200    !
210 Go_on:   PRINT "HARK!     The HP 3562A is awake!"
220    END
```

Lines 60 through 100 enables the interrupt then wait for the SRQ to occur. Line 60 tells the program to go to "Got_srq" when it receives the SRQ. Line 70 enables the SRQ interrupt. Line 80 displays a message, and line 90 forces the program to wait until the SRQ is received.

After the SRQ is received, lines 110 through 190 determine if it was actually the HP 3562A's power-up that generated the SRQ. (In contrast to examples 6 and 7, which did not verify the cause of the interrupt. The reason it was done here is that the time between when the computer starts looking for an interrupt and when it actually receives one is indeterminate. Another device could generate an SRQ in the meantime, and we need away to separate the interrupts.) Line 120 serial polls the HP 3562A and reads its status byte. Line 140 masks out bits 4—7 of the status byte (the power-on SRQ is encoded in bits 0—3, so erasing the other bits make decoding easier). Line 160 checks to see if the variable "Status_byte" equals 12. If yes, the SRQ was caused by the analyzer's power-up, and the program prints a message with line 210 then ends. If not, the SRQ had another cause, and line 190 sends the program back to line 60 to wait for the next interrupt.

To demonstrate this program, turn the HP 3562A off then run the program. When the message "Waiting for power-on . . . " is displayed, turn the analyzer back on. This generates the power-on SRQ and causes the program to continue with line 110.

Example #11: External Control without the Controller

This example shows a very powerful HP-IB feature of the HP 3562A: providing "external" control without the controller. The analyzer's system controller capability allows it to output HP-IB command strings. These commands can be used to program other devices on the bus. When combined with the auto sequence feature, command strings allow you to create entire programs in the HP 3562A that control other devices.

```
             Auto Sequence 1   16 Keys Left
Display ON   Label: EXMPLE 11
  1  HP-IB FCTN: OUTPUT STRING --DBSZ50, 1
  2  HP-IB FCTN: OUTPUT STRING --DBAC1; PU
  3  IBFN: OUTPUT STRING --PA600, 1000; PD
  4  HP-IB FCTN: OUTPUT STRING --CHSZ3
  5  IBFN: OUTPUT STRING --WAIT'CONTROL'
  6  HP-IB FCTN: OUTPUT STRING --DBUP1
```

Let's start with a one-line example. Press the **HP-IB FCTN** key (the one used earlier to set addresses). One of the softkeys in its menu is OUTPUT STRING. After pressing this, you can enter an HP-IB address followed by a comma, then the desired command string. Here is an example that resets a device at address 7:

OUTPUT STRING 7,RST

Assume that the device at address 7 is a synthesizer you are using to stimulate a device the HP 3562A is testing. You could program the synthesizer's setup state in an auto sequence, then with one key press set up the synthesizer under HP-IB control.

To make the OUTPUT STRING feature even more useful, the analyzer allows you to send HP-IB commands back to itself. Just replace the address and comma with two dashes (– –). Let's modify the last example to reset the HP 3562A:

OUTPUT STRING—RST

Try this one and verify that the analyzer is indeed reset. Obviously, we haven't accomplished much if we merely emulate front panel commands. However, the real power of the OUTPUT STRING – – is using the bus-only commands. (Remember these are the commands available only via HP-IB and not from the front panel.) The following example uses an auto sequence programmed with some of the graphics commands used in example #5:

Line 1 sets up one 50-character display buffer. Line 2 activates this buffer (#1) then lifts the "pen." Line 3 move the pen to left center of the screen then puts the pen back down. Line 4 sets the character size to 3 (the largest size available). Line 5 writes the message "CONTROL." Line 6 puts buffer #1 up on the display. Figure 4 shows the result of running this auto sequence.

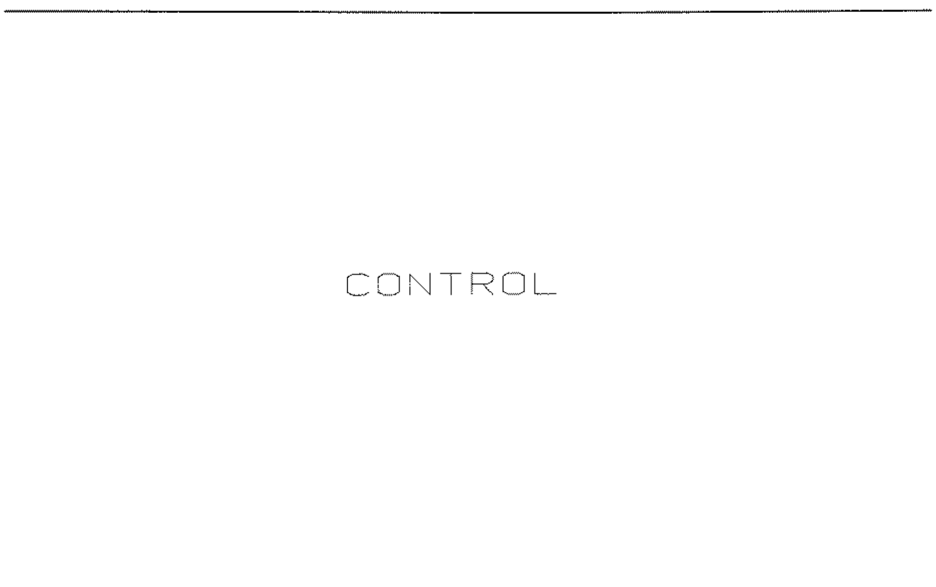


Figure 4 The Result of Example #11

By combining auto sequences with graphics commands, you can display operator messages and simple diagrams without using a computer. (Chapter 10 in the *HP 3562A Operating Manual* explains the use of auto sequences.)

Example #12: User-Defined Windows

This program shows you how to load a user-defined window into the HP 3562A. This feature allows you to create a function in your computer then have the analyzer use that function as its window. Creating a function in a controller and transferring to the analyzer presents a special problem: you need to construct a header to go along with the data you want to send. To provide a simple example, this program modifies example #8, which dumped a trace and its header then turned around and loaded them back into the analyzer. To avoid the problem of creating a header, we will use the header dumped from the analyzer and simply replace the original data with our created windowing function.

```
10    ! Example 12
20    ! Loading a user-defined window
30    !
40    OPTION BASE 1             ! Set array base to 1
50    ASSIGN @Dsa TO 720        ! Address of 3562A
60    OUTPUT @Dsa;"FRLN"        ! Freq resp preset
65    WAIT 3                    ! Pause to allow preset
70    OUTPUT @Dsa;"LSP1"        ! Select linear spectrum
80    DIM Header(66)            ! Create header array
90    DIM Trace(1602)           ! Create trace array
100   !
110   OUTPUT @Dsa;"DDAN"        ! Dump data command
120   ENTER @Dsa USING "#,2A,W";A$,Length
130   !
140   ASSIGN @Dsa;FORMAT OFF    ! Disable ASCII format
150   ENTER @Dsa;Header(*),Trace(*)
160                             ! Enter header and trace
170   ASSIGN @Dsa;FORMAT ON     ! Enable ASCII format
180   !
190   MAT Trace= (0)            ! Erase trace data
200   Trace(1)=.999028          !
210   Trace(3)=.957252          ! Enter window data
220   Trace(5)=.351960          !
230   !
240   OUTPUT @Dsa;"LDAN"        ! Load data command
250   OUTPUT @Dsa USING "#,2A,W";"#A",Length
260                             ! Data format and length
270   ASSIGN @Dsa;FORMAT OFF    ! Disable ASCII format
280   OUTPUT @Dsa;Header(*),Trace(*)
290                             ! Output header and trace
300   ASSIGN @Dsa;FORMAT ON     ! Enable ASCII format
310   !
320   OUTPUT @Dsa;"FFT1"        ! Inverse FFT
330   OUTPUT @Dsa;"SAVD1"       ! Save function
340   OUTPUT @Dsa;"USD1"        ! Select user window
350   END
```

Most of this program was taken from example #8, so this explanation covers only those lines unique to this example. Lines 60 and 70 select the frequency response preset and the linear spectrum display. This causes the analyzer to expect a linear spectrum in return when the window function is loaded. Lines 80 and 90 create separate arrays for the header and data that will be dumped from the analyzer. They are put into separate arrays to make replacing the data while keeping the header easier.

Lines 110 through 170 are similar to the dump commands used in example #8, with the exception of using two arrays. Line 190 is a BASIC function to set the Trace array to zero; this erases the trace data just dumped from the analyzer. Lines 200 through 220 insert several values into the beginning of the trace array, while leaving the rest equal to zero. Note that this example is not intended to produce a usable window, but only to demonstrate how a window can be created. After the trace data have been replaced by the window function, the header can be combined with the new trace data and sent back to the analyzer. This is accomplished by lines 240 through 300.

The user-defined window must be a time domain function, so line 320 performs an inverse FFT on the trace jump loaded into the analyzer. Line 330 then saves the result in the saved data #1 memory location. Finally, line 340 selects the user-defined window.

HEWLETT PACKARD

FOR MORE INFORMATION: Call your HP Sales Office listed in the telephone directory white pages. Ask for the Electronic Instruments Department. Or write to Hewlett-Packard: **U.S.A.** P.O. Box 10301, Palo Alto, CA 94303-0890. **Europe:** P.O. Box 999, 1180 AZ Amstelveen, The Netherlands. **Canada:** 6877 Goreway Drive, Mississauga, L4V 1M8, Ontario. **In Japan:** Yokogawa-Hewlett-Packard Ltd., 3-29-21, Takaido-Higashi, Suginami-Ku, Tokyo 168. Elsewhere in the world, write Hewlett-Packard Intercontinental, 3495 Deer Creek Road, Palo Alto, CA 94304 U.S.A.

Printed in U.S.A. Data subject to change 5953-5183

QUICK REFERENCE GUIDE

INTRODUCTION

This appendix provides condensed HP-IB programming information for the HP 3562A Dynamic Signal Analyzer. It contains the following information in quick reference format:

General command syntax

Response to bus management commands

Command mnemonics, including syntax, limits & terminators

Service requests

Status byte description, including masking

Instrument status register description

Activity status register description

Error codes

Key codes

For complete information, please refer to Chapters 1 through 6. This appendix is intended for reference use by programmers familiar with both the HP 3562A and the computer/controller being used.

The mnemonic list is divided in two parts. The first part contains the front panel (key and softkey) mnemonics listed alphabetically. The second part contains the bus-only commands listed alphabetically.

GENERAL COMMAND SYNTAX

The general syntax for sending commands to the HP 3562A is:

<mnem> <opt sp> <para> <sep> <para> <opt sp> <suff> <term>

where <mnem> is the command mnemonic
<opt sp> is ignored optional space
<para> is first command-dependent parameter
<sep> is required comma (,) for multi-parameter commands
<para> is second command-dependent parameter
<opt sp> is ignored optional space
<suff> is command-dependent suffix
<term> is command terminator (semicolon)

For example, to set up a frequency span from 10 to 60 kHz, you would send the command:

FRS 10,60 KHZ;

where: FRS is the mnemonic
10 is the first command-dependent parameter
, is the parameter separator
60 is the second command-dependent parameter
KHZ is the command-dependent suffix
; is the command terminator

Note that the front panel mnemonics usually emulate the respective key or softkey. In some cases, suffixes (terminators, delimiters) are not required. The syntax required for every command is described in the mnemonic table. You should consult this whenever there is a question about a particular command's syntax.

Parameter Queries

To query the current value of any variable parameter, send the appropriate mnemonic followed by a question mark. For example, to learn the current frequency span, send FRS?.

RESPONSE TO BUS MANAGEMENT COMMANDS

Table 1 summarizes the HP 3562A's response to the HP-IB primary bus management commands.

Table 1 Response to Bus Management Commands

Command	**Response**
ABORT I/0	Aborts data input or output and unaddresses the analyzer. Does not clear the HP-IB command buffer.
CLEAR LOCKOUT & SET LOCAL	Clears local lockout and returns to local control.
DEVICE CLEAR	Unconditionally interrupts bus activity: clears the HP-IB command buffer, resets the SRQ line, aborts data input/output, and enters REMOTE mode.
LOCAL	Returns to local (front panel) control and aborts load operations in progress, but does not abort dump operations or clear the HP-IB command buffer.
LOCAL LOCKOUT	Disables the front panel LOCAL key, but does affect local/remote status.
PARALLEL POLL	Does not respond.
PARALLEL POLL CONFIGURE	Does not respond.
PASS CONTROL	Accepts control if needed; passes control back when finished to address specified by the CTAD command. Immediately passes control back is it receives control when it does not need it.
REMOTE	Forces the HP 3562A into the REMOTE mode.
SERIAL POLL	Responds by sending its status byte, a 8-bit integer.
TRIGGER	Accepts HP-IB triggering if it is first enabled by sending the analyzer the HPT command.

FRONT PANEL COMMANDS

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
A	A			A
A & B TRACES	ABTR			ABTR
A&B	AB			AB
A GAIN ON OFF	AGON	0 or 1		AGON1 = on AGON0 = off
A GAIN SELECT	AGSE			AGFN
ABORT CAPTUR	ABCP			ABCP
ABORT HPIB	ABIB			ABIB
ABORT THRUPT	ABTH			ABTH
ACTIVE FILE	ACFL	alpha		ACFL 'aaaaaaaa'
ADD	ADD	$10^{\pm 38}$	TRACE A (TRCA) TRACE B (TRCB) SAVED 1 (SAV1) SAVED 2 (SAV2)	ADDrrrr ADDssss
ADD LINE	ADDL	see comment		ADDL (auto sequence; all subsequent commands are entered in aseq
ADD LINE	ADLN	$10^{\pm 38}$	MHZ, HZ, KHZ	ADLNrr,rrss (curve fit table)
ADD REGION	ADRG	0-100 kHz	MHZ, HZ, KHZ	ADRGrr,rrss
ADD VALUE	ADDV	$10^{\pm 38}$	MHZ, HZ, KHZ	ADDVrr,rrss
ADDRESS ONLY	ADRS			ADRS
AM CHAN 1	AM1			AM1
AM CHAN 2	AM2			AM2
ANNOT A PEN	ANAP	0—8 +		ANAPrr (number limited by plotter)
ANNOT B PEN	ANBP	0—8 +		ANBPrr (number limited by plotter)
ARM	ARM			ARM

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
ARM AU MAN	ARMA	0 or 1		ARMA0 = manual ARMA1 = auto
ASEQ FCTN	ASFN			ASFN
ASEQ MESSGE	ASMS	alpha		ASMS'aa..a' (24 char. max)
AT POINTR	ATPT			ATPT
AUTO 1 RNG UP	AU1U			AU1U
AUTO 1 UP&DWN	AU1			AU1
AUTO 2 RNG UP	AU2U			AU2U
AUTO 2 UP&DWN	AU2			AU2
AUTO CORR	AUCR			AUCR
AUTO CORR1	AUC1			AUC1
AUTO CORR2	AUC2			AUC2
AUTO CRRIER	ACRR			ACRR
AUTO INTGRT	AUIN			AUIN
AUTO MATH	AMTH			AMTH (**AUTO MATH** key)
AUTO MATH	AUMT			AUMT (AUTO MATH softkey)
AUTO ON OFF	AUTO			AUTO0 = off AUTO1 = on
AUTO ORDER	AUOR			AUOR
AUTO SEQ	ASEQ			ASEQ
AUTO WEIGHT	AUWT			AUWT
AVG	AVG	1—32767		AVGrrrr
AVRG	AVRG			AVRG
AVG OFF	AVOF			AVOF
AVG VALUE	AVGV			AVGV
B	B			B
BEEPER ON OFF	BEEP			BEEP0 = off BEEP1 = on
BURST CHIRP	BCRP	1-99		BCRPrr

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
BURST RANDOM	BRND	1-99		BRNDrr
CAL	CAL			CAL
CALC OFF (HMNC)	CAOF			CAOF
CALC OFF (SBAND)	CLOF			CLOF
CAPTUR HEADER	CHED			CHED
CAPTUR LENGTH	CLEN	see comment	USEC, MSEC, SEC MIN, REVS, PNTS REC	CLENrrss (range depends on suffix; 10 records or equivalent limits)
CAPTUR POINTR	CPNT	(same as CAPTUR LENGTH)		CPNTrrss
CAPTUR SELECT	CPSE			CPSE
CATLOG POINTR	CTPT	1-20		CTPTrr
CDF 1	CDF1			CDF1
CDF 2	CDF2			CDF2
CENTER FREQ	CF	see comment	MHZ, HZ, KHZ ORD, RMP	CFrrss (range limited to 100 kHz – (10.24 mHz/2)
CH 1 ACTIVE	CH1			CH1
CH 1&2 ACTIVE	CH12			CH12
CH 2 ACTIVE	CH2			CH2
CHAN 1 AC DC	C1AC	0 or 1		C1AC 0 = dc C1AC 1 = ac
CHAN 1 DELAY	C1DL	see comment	USEC, MSEC, SEC, MIN, REVS, REC	C1DLrrss (range depends on suffix; – 4095 points and + 50 records are absolute limits)
CHAN 1 INPUT	C1IN			C1IN
CHAN 1 RANGE	C1RG	-51-27	V, MV, VRMS, MVRM, DBV, EU	C1RGrrss (range depends on suffix; absolute limit is – 51 to + 27 dBV)
CHAN 2 AC DC	C2AC	0 or 1		C2AC 0 = dc C2AC 1 = ac
CHAN 2 DELAY	C2DL	see comment	USEC, MSEC, SEC, MIN, REVS, REC	C2DLrrss (range depends on suffix; – 4095 points and + 50 records are absolute limits)
CHAN 2 INPUT	C2IN			C2IN

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
CHAN 2 RANGE	C2RG	-51-27	V, MV, VRMS, MVRM, DBV, EU	C2RGrrss (range depends on suffix; absolute limit is −51 to +27 dBV
CHANGE LINE	CHGL	see comment		CHGL (for auto sequences and auto math; entry is any valid commamnd)
CHANGE REGION	CHRG	0-100 kHz	MHz, Hz, KHz	CHRGrr,rrss
CHANGE VALUE	CHGV	$10^{\pm 38}$	MHZ, HZ, KHZ	CHGVrr,rrss
CLEAR ASEQ	CLAS			CLAS (auto sequence must be displayed first)
CLEAR LOGS	CLLG			CLLG
CLEAR MATH	CLMA			CLMA
CLEAR TABLE	CLTA			CLTA (curve fit)
CLEAR TABLE	CLTB			CLTB (synthesis; table must be displayed first)
CLEAR TABLE	CLRT			CLRT (delete freq)
COHER	COHR			COHR
COMPLX CONJ	CMPC			CMPC
CONT ASEQ	CNAS			CNAS
CONT PEAK	CNPK			CNPK
CONVRT TABLE	CVTB			CVTB
COORD	CORD			CORD
COPY FILES	COFI	alpha		COFI'aaaaaaaa' COFI'<,aaaaaa' COFI'aaaaaa,>' COFI'aaa,aaa'
CREATE CONST	CCON	$10^{\pm 38}$		CCONrr CCONrr,rr
CREATE FIT	CRFT			CRFT
CREATE THRUPT	CRTH	alpha		CRTH'aaaaaaa'
CREATE TRACE	CTRC			CTRC
CROSS CORR	CCOR			CCOR (measurement)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
CROSS CORR	CRCR			CRCR (display)
CROSS SPEC	CSPC			CSPC
CRRIER FREQ	CRFR	0-100 kHz	MHZ, HZ, KHZ, RPM, ORD	CRFRrrss
CURVE FIT	CVFT			CVFT
CUT PG ON OFF	CTPG	0 or 1		CTPG0 = off CTPG1 = on
DASHED LINES	DSHL			DSHL
DATA & ANNOT	DAAN			DAAN
DATA ONLY	DATA			DATA
DATE M,D,Y	DATE	mm,dd,yy		DATEmm,dd,yy
dB	DB			dB (terminator only)
dBV	DBV			dBV (terminator only)
DC OFFSET	DCOF	0—10	MV, V, VRMS, MVRM, DBV	DCOFrrss (max is 10Vpeak
Decade	DEC			DEC (terminator only)
Degree	DEG			DEG (terminator only)
DELETE FILE	DLTF	alpha	AT POINTR	DLTF'aaaaaaaa' DLTFATPT
DELETE FREQ	DLFR			DLFR
DELETE LINE	DLTL			DLTL (auto sequence or auto math; table must be displayed first)
DELETE LINE#	DLLN	1-20		DLLNrr
DELETE REGION	DLRG	1-20		DLRGrr
DELETE VALUE	DLTV			DLTV
DEMOD BOTH	DMB			DNB
DEMOD CHAN 1	DM1			DM1
DEMOD CHAN 2	DM2			DM2
DEMOD ON OFF	DMOD	0 or 1		DMOD0 = off DMOD1 = on
DEMOD POLAR	POLR			POLR
DEMOD SELECT	DMSE			DMSE

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
DESTN ADDRES	DEAD	1-7		DEADr
DESTN UNIT	DEUN	1-15		DEUNrr
DFAULT GRIDS	DFGR			DFGR
DFAULT LIMITS	DLIM			DLIM
DIFF	DIFF			DIFF
DISC	DISC			DISC
DISC ADDRES	DIAD	1-7		DIADr
DISC COPY	DICO			DICO
DISC FCTN	DIFN			DIFN
DISC STATUS	DIST			DIST
DISC UNIT	DIUN	0-15		DIUNrr
DIV	DIV	$10^{\pm 38}$	TRACE A (TRCA) TRACE B (TRCB) SAVED 1 (SAV1) SAVED 2 (SAV2)	DIVrrr DIVssss
DOWN ARROW	DOWN			DOWN
DOTS	DOTS			DOTS
DSPLAY ON OFF	DSPL	0 or 1		DSPL0 = off DSPL1 = on
E SMPL ON OFF	ESMP	0 or 1		ESMP0 = off ESMP1 = on
EDIT	EDIT			EDIT
EDIT DENOM#	EDDN	1-20		EDDNrr
EDIT LINE#	EDLN	1-20		EDLNrr
EDIT LINE#	LINE	1-20		LINErr
EDIT MATH	EDMA			EDMA
EDIT NUMER#	EDNM	1-20		EDNMrr
EDIT POLE#	EDPL	1-20		EDPLrr
EDIT POLES	EPOL			EPOL
EDIT RESDU#	EDRS	1-20		EDRSrr
EDIT TABLE	EDTB			EDTB

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
EDIT WEIGHT	EDWT			EDWT
EDIT ZERO#	EDZR	1-20		EDZRrr
EDIT ZEROS	EZER			EZER
END EDIT	ENED			ENED (auto sequence or auto math; table must be displayed first)
ENGR UNITS	ENGR			ENGR
ENTER	ENT			ENT (terminator only)
EU	EU			EU (terminator only)
EU LBL CHAN 1	EUL1	alpha		EUL1 'aaaaaa'
EU LBL CHAN 2	EUL2	alpha		EUL2 'aaaaaa'
EU VAL CHAN 1	EUV1	± 1nV to ± 1000V	VEU, MVEU, DB	EUV1rrss
EU VAL CHAN 2	EUV2	± 1nV to ± 1000V	VEU, MVEU, DB	EUV2rrss
EXPONENT	E			rrErr (exponential notation; example: 10E4 = 100 000. D or L can be used in place of E.)
EXPON	EXP			EXP
EXPON CHAN 1	XPN1	$10^{\pm 38}$	USEC, MSEC, SEC, MIN, REVS	XPN1rrss
EXPON CHAN 2	XPN2	$10^{\pm 38}$	USEC, MSEC, SEC, MIN, REVS	XPN2rrss
EXT	EXT			EXT
F RESP LINRES	FRLN			FRLN
F RESP LOGRES	FRLG			FRLG
F RESP SWEPT	FRSW			FRSW
FAULT LOG	FTLG			FTLG (disc service functions)
FFT	FFT			FFT
FFT^{-1}	FFT1			FFT1
FILTRD INPUT	FILT			FILT
FIT FCTN	FTFN			FTFN

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
FIT → SYNTH	FTSN			FTSN
FIX LINE#	FXLN	1-20	ENT	FXLNrr
FIXED INTGRT	FXIN			FXIN
FIXED SINE	FSIN	64-100000	MHZ, HZ, KHZ RPM, ORDS	FSINrrss (range is 64 µHz to 100 kHz; entry limits depend on suffix)
FLAT TOP	FLAT			FLAT
FLOAT CHAN 1	FLT1			FLT1
FLOAT CHAN 2	FLT2			FLT2
FM CHAN 1	FM1			FM1
FM CHAN 2	FM2			FM2
FNDMTL FREQ	FNFR	0-100k	MHZ, HZ, KHZ RPM, ORDS	FNFRrrrsss
FORCE CHAN 1	FRC1	$10^{\pm 38}$	USEC, MSEC, SEC, MIN, REVS	FRC1rrrsss
FORCE CHAN 2	FRC2	$10^{\pm 38}$	USEC, MSEC, SEC, MIN, REVS	FRC2rrrsss
FORCE/EXPON	FOXP			FOXP
FORMAT	FORM			FORM
FORMAT OPTION	FOOP	0-239		FOOPrr
FREE RUN	FREE			FREE
FREQ	FREQ			FREQ
FREQ & DAMP	FRDA			FRDA
FREQ RESP	FRQR			FRQR (display)
FREQ RESP	FRSP			FRSP (measurement)
FREQ SPAN	FRS			
Linear Resolution		10.24 mHz - 100 kHz	MHZ, HZ, KHZ, RPM, ORDS	FRSrrrsss
Log Resolution		1-5	DEC	FRSrDEC
Swept Sine		2 mHz—100 kHz	MHZ, HZ, KHZ, DEC, OCT	FRSrrrsss
Time Capture		same as linear resolution		

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
FRONT BACK	FRBK			FRBK
FST AV ON OFF	FSAV	0 or 1		FSAV0 = off FSAV1 = on
GO TO	GOTO	1-20		GOTOrr
GRID AREA	GRAR			GRAR
GRID PEN	GRDP	1—max		GRDPrr (max = number of pens in plotter)
GROUND CHAN1	GND1			GND1
GROUND CHAN2	GND2			GND2
HANN	HANN			HANN
HELP	HELP			HELP
HIST	HIST			HIST
HIST 1	HIS1			HIS1
HIST 2	HIS2			HIS2
HMNC ON	HMNC			HMNC
HMNC POWER	HPWR			HPWR
HOLD X CENTER	HXCT			HXCT
HOLD X LEFT	HXLF			HXLF
HOLD X OFF	HXOF			HXOF
HOLD X RIGHT	HXRT			HXRT
HOLD Y CENTER	HYCT			HYCT
HOLD Y LOWER	HYLW			HYLW
HOLD Y OFF	HYOF			HYOF
HOLD Y UPPER	HYUP			HYUP
HP-IB ADDR	IBAD	0—31		IBADrr
HP-IB FCTN	IBFN			IBFN
Hz	HZ			HZ (terminator only)
Hz (Sec)	HZS			HZS
Hz/Point	HZ/P			HZ/P (terminator only)
Hz/mSec	H/MS			H/MS (terminator only)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
Hz/Min	HZ/M			HZ/M (terminator only)
Hz/Sec	HZ/S			HZ/S (terminator only)
Hz/Order	HZ/O			HZ/O (terminator only)
IMAG	IMAG			IMAG
IMAGE BACKUP	IMBK			IMBK
IMPLS RESP	IRSP			IRSP
INIT CATLOG	INCT	alpha		INCT'aaaaaa'
INIT DISC	INDI	alpha		INDI'aaaaaa'
INPUT COUPLE	ICPL			ICPL
INPUT SPEC 1	ISP1			ISP1
INPUT SPEC 2	ISP2			ISP2
INPUT TIME 1	ITM1			ITM1
INPUT TIME 2	ITM2			ITM2
INST	INST			INST
INST WNDOWD	IWND			IWND
INTGRT	INGR			INGR
INTGRT INIT = 0	INGI			INGI
INTGRT TIME	INTM	10^{-3}–10^{38}	USEC, MSEC, SEC	INGRTrrss
$j\omega$	JW			JW
$j\omega^{-1}$	JW1			JW1
kHz	KHZ			KHZ (terminator only)
kHz/Order	KH/O			KH/O (terminator only)
L SPEC UNITS	LSUN			LSUN
LABEL ASEQ	LBLA	alpha		LBLA'aaa,aaa'
LABEL MATH	LBLM	alpha		LBLM'aaa,aaa'
LAST MEAS	LSMS			LSMS
LIN X	LINX			LINX
LINE	not progammable over the HP-IB			

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
LINE A TYPE#	LINA	1-8; see comment		LINArrr,rrr (number depends on plotter; 2nd parameter optional)
LINE B TYPE#	LINB	1-8; see comment		LINBrrr,rrr (number depends on plotter; 2nd parameter optional)
LINE TYPES	LNTP			LNTP
LINEAR RES	LNRS			LNRS
LINEAR SPEC	LSPC			LSPC
LINEAR SPEC 1	LSP1			LSP1
LINEAR SPEC 2	LSP2			LSP2
LINEAR SWEEP	LNSW			LNSW
LN OF DATA	LN			LN
LN^{-1} OF DATA	LN1			LN1
LOCAL	LCL			LCL
LOG RES	LGRS			LGRS
LOG SWEEP	LGSW			LGSW
LOG X	LOGX			LOGX
LOOP TO	LPTO	1-20 (r1) 1-32,767 (r2)		LPTOr1,r2 (1st number is end of loop; 2nd is cycle count)
MAG (LIN)	MAG			MAG
MAG (LOG)	MGLG			MGLG
MAG (dB)	MGDB			MGDB
MAG (dBm)	MDBM			MDBM
MANUAL PRVIEW	MAPR			MAPR
MANUAL SWEEP	MNSW			MNSW
MARKER VALUE	MKVL			MKVL
MATH	MATH			MATH
MAX SPAN	MAXS			MAXS
MEAS DISP	MDSP			MDSP

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
MEAS MODE	MSMD			MSMD
uSec	USEC			USEC (terminator only)
mEU	MEU			MEU (terminator only)
mHz	MHZ			MHZ (terminator only)
mHz/Order	MH/O			MH/O (terminator only)
mSec	MSEC			MSEC (terminator only)
mV	MV			MV (terminator only)
mV/EU	MVEU			MVEU (terminator only)
mVrms	MVRM			MVRM (terminator only)
Min	MIN			MIN (terminator only)
Min/Dec	M/DC			M/DC (terminator only)
Min/Oct	M/OC			M/OC (terminator only)
MPY	MPY	$10^{\pm 38}$	TRACE A (TRCA) TRACE B (TRCB) SAVED 1 (SAV1) SAVED 2 (SAV2)	MPYrrr MPYssss
MRKR → PEAK	MKPK			MKPK
NEGATE	NEG			NEG
NEXT	NXT			NXT (**MATH** menu, first level)
NEXT	NEX			NEX (**MATH** menu, second level)
NEXT	NEXT			NEXT (**COORD** menu)
NEXT	NX			NX (**AVG** menu)
NEXT PAGE	NXTP			NXTP (disc catalog)
NEXT PAGE	NXPG			NXPG (disc service logs)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
NEXT RECORD	NXRC			NXRC
NICHOL	NICL			NICL
NO	Use REJT in place of NO for previewing over the bus			
NO PAGING	NOPG			NOPG
NUMBER AVGS	NAVG	1-32767		NAVGrrr
NUMBER POLES	NPOL	1-40		NPOLrr
NUMBER ZEROS	NZER	1-40		NZERrr
NYQUST	NYQT			NYQT
Oct/Min	OC/M			OC/M (terminator only)
Oct/Sec	OC/S			OC/S (terminator only)
Octave	OCT			OCT (terminator only)
Ohm	OHM			OHM (terminator only)
ORBITS T1vsT2	ORBT			ORBT
Orders	ORD			ORD (terminator only)
Orders (Revs)	ORDR			ORDR
Orders CAL	ORCL	$10^{\pm 38}$	HZ/0, KH/0 MH	ORCL rrr sss
OUTPUT LOG	OULG			OULG
OUTPUT STRING	not programmable via HP-IB			
OV REJ ON OFF	OVRJ	0 or 1		OVRJ0 = off OVRJ1 = on
OVER WRITE	OVWR			OVWR
OVERWR AU MAN	OVAU	0 or 1		OVAU0 = off OVAU1 = on
OVRLP%	OVLP	1-90		OVLPrr
P SPEC LINRES	PSLN			PSLN
P SPEC UNITS	PSUN			PSUN
PACK DISC	PKDI			PKDI
PAGE BACK	PGBK			PGBK
PAGE FORWRD	PGFW			PGFW
PAGING CONTRL	PCTL			PCTL

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
PAUSE ASEQ	PSAS			PSAS
PAUSE CONT	PSCN			PSCN toggles PAUS—pauses CONT—continues
PDF 1	PDF1			PDF1
PDF 2	PDF2			PDF2
PEAK HOLD	PHLD			PHLD
PHASE	PHSE	±180	DEG	PHSErrrDEG
PLOT	PLOT			PLOT
PLOT ADDRES	PLAD	0-31		PLADrr
PLOT AREA	PLAR			PLAR
PLOT LIMITS	PLIM			PLIM
PLOT PRESET	PLPR			PLPR
PM CHAN 1	PM1			PM1
PM CHAN 2	PM2			PM2
PM/FM CRRIER	PFCR			PFCR
Points	PNTS			PNTS (terminator only)
Points/Dec	P/DC			P/DC (terminator only)
Points/Oct	P/OC			P/OC (terminator only)
Points/Sweep	P/SW			P/SW (terminator only)
POLAR AMvsPM	POLR			POLR
POLE RESIDU	PRSD			PRSD
POLE ZERO	PZRO			PZRO
POLY-NOMIAL	POLY			POLY
POWER	PWR			PWR
POWER SPEC	PSPC			PSPC
POWER SPEC 1	PSP1			PSP1
POWER SPEC 2	PSP2			PSP2
PRESET	PRST			PRST
PREV PAGE	PRVP			PRVP

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
PRIODC CHIRP	PCRP	1-99		PCRPrr
PROTCT ON OFF	PTON	0 or 1		PTON0 = off PTON1 = on
PRVIEW OFF	PROF			PROF
PRVIEW ON OFF	PRON			PRON0 = off PRON1 = on
Pulse/Rev	P/RV			P/RV (terminator only)
PwrSRQ ON OFF	PSRQ	0 or 1		PSRQ0 = off PSRQ1 = on
RAMP TIME	RAMP	$10^{\pm 38}$	USEC, MSEC, SEC, MIN, REC	RAMPrrrsss (limit is $10^{\pm 38}$s; entry range depends on suffix)
RANDOM NOISE	RND			RND
RANGE	RNG	–51 to +27 dBV	V, MV, VRMS, MVRM, DBVR,	RNGrrrsss (entry is EU optional; range depends on suffix)
READ PEN→P1	RDP1			RDP1
READ PEN→P2	RDP2			RDP2
REAL	REAL			REAL
REAL PART	RLPT			RLPT
RECALL DATA#	RCLD	1 or 2		RCLDr
RECALL FILE	RCFL	alpha		RCFL'aaaaaaa'
RECALL PWR DN	RCLP			RCLP
RECALL STATE#	RCLS	1—5		RCLSr
RECIP	RCIP			RECIP
Record	REC			REC (terminator only)
REF CHAN 1	RFC1			RFC1
REF CHAN 2	RFC2			RFC2
REF LEVEL	RFLV	5 mV to 31.5 Vpk	V, MV, VRMS, MVRM, DBVR, EU	RFLVrrrsss
RESLTN	RES	64 µHz— 99.99994 kHz	HZ/P, P/SW	RESrrrsss
		1—1^{10} pts/dec	P/DC, P/OC, P/SW	

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
RESLTN AU FIX	RSAU	0 or 1		RSAU0 = fix RSAU1 = au
RESTOR CATLOG	RSCT	alpha		RSCT'aaaaa'
RETURN	RTN			RTN
REVS	REVS			REVS (terminator only)
ROT 90 ON OFF	ROT	0 or 1		ROT0 = off ROT1 = on
RPM	RPM			RPM (terminator only)
RPM (Sec)	RPMS			RPMS
SAMPLE FREQ	SMPF	1-256 kHz	KHZ, HZ, MHZ RPM, P/RV	SMPFrrrsss
SAVE DATA#	SAVD	1 or 2		SAVDr
SAVE FILE	SAVF	alpha		SAVF'aaaaaaaa'
SAVE STATE#	SAVS	1—5		SAVSr
SAVE RECALL	SAVR			SAVR
SAVED 1	SAV1			SAV1
SAVED 2	SAV2			SAV2
SBAND INCRMT	SBIN	12.8 µHz – 100 kHz	KHZ, Hz, MHZ, RPM, ORD	SBINrrrsss
SBAND ON	SBND			SBND
SBAND POWER	SPWR			SPWR
SCALE	SCAL			SCAL
SCALE FREQ	SCFR	$10^{\pm 6}$	KHZ, HZ, MHZ	SCFRrrrsss (**SYNTH**)
SCALE FREQ	SCLF	$10^{\pm 6}$	KHZ, HZ, MHZ	SCLFrrrsss (**CURVE FIT**)
SCROLL ON OFF	SCRL	0 or 1		SCRL0 = off SCRL1 = on
Sec	SEC			SEC (terminator only)
Sec/Dec	S/DC			S/DC (terminator only)
Sec/Oct	S/OC			S/OC (terminator only)
SELECT DATA	SDAT			SDAT
SELECT MEAS	SMES			SMES

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
SELECT PENS	SPEN			SPEN
SELECT TRIG	SELT			SELT
SELF TEST	TST			TST
SERVCE FCTNS	SVFN			SVFN
SET P1 LWR LF	SEP1	±32767		SEP1rrr,rrr
SET P2 UPR RT	SEP2	±32767		SEP2rrr,rrr
SINGLE	SNGL			SNGL
SINGLE CAL	SNGC			SNGC
SLOPE	SLP			SLP
SLOPE + −	SLOP	0 or 1		SLOP0 = off SLOP1 = on
SOLID GRIDS	SLGR			SLGR
SOLID LINES	SLDL			SLDL
SOLIDA DASH B	SLDA			SLDA
SOURCE	SRCE			SRCE
SOURCE LEVEL	SRLV	0—5V	V, MV, VRMS, MVRM, DBV	SRLVrrrsss
SOURCE LIMIT	SRLM	5 mV—5V	V, MV, VRMS, MVRM, DBV	SRLMrrrsss
SOURCE OFF	SROF			SROF
SOURCE ON OFF	SRON	0 or 1		SRON0 = off SRON1 = on
SOURCE TRIG	STRG			STRG
SPARE BLOCK	SPBL	depends on drive see Chapter 11 of operating manual		SPBL
SPCL FCTN	SPFN			SPFN
SPCL MARKER	SPMK			SPMK
SPEED F S	SPED	0 or 1		SPED0 = slow SPED1 = fast
SQUARE ROOT	SQRT			SQRT
STABLE (MEAN)	STBL			STBL
START	STRT			STRT

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
START ASEQ 1	ASQ1			ASQ1
START ASEQ 2	ASQ2			ASQ2
START ASEQ 3	ASQ3			ASQ3
START ASEQ 4	ASQ4			ASQ4
START ASEQ 5	ASQ5			ASQ5
START CAPTUR	STCP			STCP
START FREQ	SF	0—99999.97952 Hz	KHZ, Hz MHZ, RPM, ORD	SFrrrsss (linear res, capture)
		0.1 Hz—100 kHz	KHZ, HZ MHZ	SFrrrsss (log res)
		64 µHz—99999.99988 Hz	KHZ, HZ MHZ, RPM, ORD	SFrrrsss (swept sine)
START MATH	STMA			STMA
START PLOT	STPL			STPL
START THRUPT	STHR			STHR
STATE TRACE	STTR			STTR STAT = state TRAC = trace
STOP FIT	SPFT			SPFT
STOP FREQ	SPF	120 µHz - 100 kHz	KHZ, HZ, MHZ, RPM, ORD	SPFrrrsss (swept sine)
STORE WEIGHT	STWT			STWT
SUB	SUB	$10^{\pm 38}$	 TRACE A (TRCA) TRACE B (TRCB) SAVED 1 (SAV1) SAVED 2 (SAV2)	SUBrrr SUBsss
SWEEP DOWN	SWDN			SWDN
SWEEP HOLD	SWHD			holds the sweep
SWEEP RATE	SWRT	$10^{\pm 38}$	S/DC, M/DC, S/OC, M/OC, H/MS, HZ/S, HZ/M	SWRTrrrsss (limit depends on suffix)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
SWEEP UP	SWUP			SWUP
SWEPT SINE	SSIN			SSIN
SWEPT UNITS	SWUN			SWUN
SYNTH	SNTH			SNTH
SYNTH → FIT	SNFT			SNFT
SYNTH FCTN	SNFN			SNFN
SYSTEM CNTLR	SYSC			SYSC
SYSTEM GAIN	GAIN	$10^{\pm 38}$		GAINrrr
T/1 – T	TT			TT
TABLE FCTN	TBFN			TBFN
THD	THD			THD
THRUPT HEADER	THED			THED
THRUPT LENGTH	THLN	1—32767	USEC, MSEC, SEC, MIN, REVS, REC	THLNrrrsss (limit is 32767 records; range depends on suffix)
THRUPT ON OFF	THRU	0 or 1		THRU0 = off THRU1 = on
THRUPT SELECT	THSE			THSE
THRUPT SIZE	THSZ	1—32767	USEC, MSEC, SEC, MIN, REVS, REC	THSZrrrsss (limit is 32767 records; range depends on suffix)
THRUPT TIME 1	THT1			THT1
THRUPT TIME 2	THT2			THT2
TICK MARKS	TKMK			TKMK
TIM AV ON OFF	TIAV	0 or 1		TIAV0 = off TIAV1 = on
TIME BUFFER	TMBF			TMBF
TIME CAPTUR	CPTR			CPTR (**MEAS MODE**)
TIME CAPTUR	TMCP			TMCP (**PRESET**)
TIME DELAY	TMDL	$10^{\pm 38}$	USEC, MSEC, SEC	TMDLrrrsss (**CURVE FIT**)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
TIME DELAY	TDLY	$10^{\pm 38}$	USEC, MSEC, SEC	TDLYrrrsss (**SYNTH**)
TIME H,M,S	TIME	00,00,00 - 23,59,59		TIMEhh,mm,ss
TIME LENGTH	TLN	8 ms - 78125s	USEC, MSEC, SEC MIN, REVS	TLNrrrsss (limit is 78125s; range depends on suffix)
TIME REC 1	TMR1			TMR1
TIME REC 2	TMR2			TMR2
TIME RECORD	TMRC			TMRC
TIME THRUPT	TMTH			TMTH
TIMED PAUSE	TIPS	0—32767	SEC	TIPSrrrSEC
TIMED PRVIEW	TIPR	0—10^{38}	SEC	TIPRrrSEC
TIMED START	TIST	00,00,00 24,59,59		TISTrr,rr,rr (24 hour deactivates timed start)
TO→POL RESIDU	TOPR			TOPR
TO→POLY	TOPY			TOPY
TO→POL ZERO	TOPZ			TOPZ
TRACE A	TRCA			TRCA
TRACE A PEN	TRAP	0—max		TRAPrr (max is number of pens in plotter)
TRACE B	TRCB			TRCB
TRACE B PEN	TRBP	0—max		TRBPrr (max is number of pens in plotter
TRACE TITLE	TITL	alpha		TITL'aaaaaa'
TRIG DELAY	TRGD			TRGD
TRIG LEVEL	TRLV	$10^{\pm 38}$	V, MV, EUC1, EUC2	TRLVrrrrssss (max is 10V for ext trigger)
UNFIX LINE#	UFLN	1—20		UFLNrr
UNIFRM (NONE)	UNIF			UNIF
UNITS	UNIT			UNIT
UP ARROW	UP			UP
UPPER LOWER	UPLO			UPLO

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
USER ORDER	USOR			USOR
USER LIMITS	ULIM			ULIM
USER LINES	ULIN			ULIN
USER CRRIER	UCRR	current span	MHZ, HZ, KHZ, RPM, ORD	UCRRrrrsss
USER SAVD1	USD1			USD1
USER SRQ	USRQ			USRQ
USER SRQ1	SRQ1			SRQ1
USER SRQ2	SRQ2			SRQ2
USER SRQ3	SRQ3			SRQ3
USER SRQ4	SRQ4			SRQ4
USER SRQ5	SRQ5			SRQ5
USER SRQ6	SRQ6			SRQ6
USER SRQ7	SRQ7			SRQ7
USER SRQ8	SRQ8			SRQ8
USER WEIGHT	USWT			USWT
V	V			V (terminator only)
V/EU	VEU			VEU (terminator only)
V/$\sqrt{}$Hz ($\sqrt{}$PSD)	VHZ			VHZ
VOLTS^2	VT2			VT2
V^2/HZ (PSD)	V2HZ			V2HZ
V^{2}s/HZ (ESD)	V2SH			V2SH
VIEW	VIEW			VIEW
VIEW CATLOG	CAT			CAT
VIEW INPUT	VWIN			VWIN
VIEW MATH	VWMA			VWMA
VIEW OFF	VWOF			VWOF
VIEW WEIGHT	VWWT			VWWT

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

Front Panel Commands (cont)

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
Vrms	VRMS			VRMS (terminator only)
VOLTS	VLTS			VLTS
VOLTS CHAN 1	VLT1			VLT1
VOLTS CHAN 2	VLT2			VLT2
VOLTS PEAK	VTPK			VTPK
VOLTS RMS	VTRM			VTRM
WEIGHT REGION	WTRG	0—100 kHz	KHZ, HZ, MHZ	WTRGrrrsss
WEIGHT VALUE	WTVL	$10^{\pm 38}$		WTVLrrr
WINDOW	WNDO			WNDO
X	X	see comment		Xrrrsss (entry optional; range and suffix depend on current display)
X AUTO SCALE	XASC			XASC
X FCTN OFF	XFOF			XFOF
X FIXD SCALE	XSCL	see comment		XSCLrrrsss XSCLrrr,rrrsss (range and suffix depend on current display)
X MRKR SCALE	XMKR			XMKR
X OFF	XOFF			XOFF
X VALUE	XVAL	see comment		XVALrrrsss XVALrrr,rrrsss (range and suffix depend on current display)
Y	Y	see comment		Yrrrsss (entry optional; range and suffix depend on current display)
Y AUTO SCALE	YASC			YASC
Y DFLT SCALE	YDSC			YDSC
Y FIXD SCALE	YSCL	see comment		YSCLrrrsss YSCLrrr,rrrsss (range and suffix depend on current display)

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

NAME	MNEM	RANGE	SUFFIXES	SYNTAX
Y MRKR SCALE	YMKR			YMKR
Y OFF	YOFF			YOFF
Y VALUE	YVAL	see comment		YVALrrrsss YVALrrr,rrrsss (range and suffix depend on current display)
YES	Use ACPT in place of YES when previewing over the bus.			
ZERO START	ZST			ZST

r = value within the range specified in the **RANGE** column
s = one of the suffixes from the **SUFFIX** column
a = alphanumeric character

BUS-ONLY COMMANDS

COMMAND	MNEM	SYNTAX/DATA FORMAT
Add Block	ADDB	ADDBn1,n2[,n3] Adds block n1 to n2 and puts the result in n3. If n3 is not specified the result is put in n2.
Add Complex Constant to Block	ADDX	ADDXn1,n2,n3[,n4] Adds complex constant n1, n2 to block n3 (n1 is the real part and n2 is the imaginary part). The result is put in n4 is specified, n3 if not.
Add Real Constant to Block	ADDC	ADDCn1,n2[,n3] Adds constant n1 to real block n2 and puts the result in n3. If n3 is not specified the result put in n2.
Activity Status Query	AS?	AS? Returns contents of activity status register
Activity Status Mask High	ASMH	ASMHn Where n = decimal equivalent of sum of bits to be unmasked. See table 7.
Activity Status Mask Low	ASML	ASMLn Where n = decimal equivalent of sum of bits to be unmasked. See table 7.
Brightness	BRIT	BRITn Where values for n are: 0 = trace off 1 = dim 2 = half bright 3 = full bright (default)
Block Size	BLSZ	BLSZ size,n1[,count] Size is limited to 37 900 words n1 is first buffer (0 to 15) count is number of buffers; if not specified, count = 1
Character Rotation	CHRO	CHROn Where values of n are: 0 = 0° (default) 1 = 90° 2 = 180° 3 = 270°
Character Size	CHSZ	CHSZn Where values of n are: 0 = 24 x 36 points (default) 1 = 36 x 54 2 = 48 x 72 3 = 60 x 90
Clear Buffer	CLBF	CLBFn Where n = −4 to 15

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Command Echo Disable	COMD	COMD
Command Echo Enable	COME	COME
Complex Fast Fourier Transform	CFFT	CFFTn1,n2 Performs FFT on complex block n1 and puts results in n2.
Complex Inverse Fast Fourier Transform	CFT1	CFT1n1,n2 Performs inverse FFT on complex block n1 and puts results in n2.
Conjugate Block	CNJB	CNJBn1[,n2] Computes the complex conjugate of complex block n1 and puts results in n2. If n2 is not specified the results are put in n1.
Controller Address	CTAD	CTADn Where n = 0 to 31
Cross Spectrum Exponential Average	CXAV	CXAVn1,n2,n3,awf Computes cross spectrum of complex floating point blocks n1 and n2 and exponentially averages it with complex floating point block n3. awf is the average weighting factor (a power of two); Result is put in block n3.
Cross Spectrum Peak Hold	CPEK	CPEKn1,n2,n3 Computes cross spectrum of complex floating point blocks n1 and n2 and compares magnitudes of result with complex block n3. The larger values are put in n3.
Cross Spectrum Summation	CSPS	CSPSn1,n2,n3 Computes cross spectrum of complex floating point blocks n1 and n2 and adds it to complex floating point block n3, puts results in n3.
D (exponent image specifier)	D	Used as an exponent indicator in scientific notation, as is "E" and "L".
Differentiate Block	DIFB	DIFBn1[,n2] Computes differential of block n1 and puts result in n2. If n2 is not specified result is put in n1.
Disable Sweep SRQ	DSWQ	DSWQ
Display	DSP	DSP'aaa...aaa' (max 24 characters)
Display Buffer Size	DBSZ	DBSZsize,n1[,count] Size is number of words in each buffer; n1 is the first buffer; count is the number buffers. If count is not specified, one block is configured.
Display Query	DSP?	DSP? Returns display message, up to 24 characters

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Display Buffer Active Append	DBAA	DBAAn Where n is the buffer to be activated
Display Buffer Active Clear	DBAC	DBAC n Where n is the buffer to be cleared and activated
Display Buffer Switch	DBSW	DBSWn1,n2 Where n1 is the currently displayed buffer, and n2 is the buffer to be displayed
Display Buffer Down	DBDN	DBDNn Where n is te buffer to be taken down
Display Buffer Up	DBUP	DBUPn Where n is the buffer to be put up
Display Disable	DSPD	DSPD
Display Enable	DSPE	DSPE
Divide Block	DIVB	DIVBn1,n2[,n3] Divides block n2 by n1 and puts results in n3. If n3 is not specified, result is put in n2.
Divide Block into Real Constant	DVIC	DIVCn1,n2[,n3] Divides block n2 by constant n1 and puts results in n3. If n3 is not specified, results are put in n2.
Divide Block by jω	DVJW	DVJWω_{start},$\Delta\omega$,n1[,n2] Divides complex block n1 by jω and puts results in n2. If n2 is not specified, results are put into n1. ω_{start} is the floating point starting value of ω and Δω is the incremental value of ω.
Divide Imaginary Part	DIVI	DIVIn1,n2[,n3] Divides the imaginary part of complex floating point block n1 by real constant n2 and puts result in n3. If n3 is not specified, result is put in n1.
Divide By Complex Constant	DIVX	DVIXn1,n2,n3[,n4]; Divides block n1 by complex constant n2, n3 and puts the results in n4. If n4 is not specified the results are put in n1. n1 may be a real or complex block. Dividing a real block by a complex number requires a destination block twice the size of the real (source) block.
Divide By Constant	DIVC	DVICn1,n2[,n3] Divides block n1 by constant n2 and puts results in n3. If n3 is not specified, results are put in n1.
Divide Real Part	DIVR	DIVRn1,n2[,n3] Divides real part of complex floating point block n1 by real constant n2 and puts result in n3. If n3 is not specified, result is put in n1.

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Dump Block in ANSI Binary	DBAN	DBAN Dumps primitive block PBLKn in ANSI format.
Dump Block in ASCII	DBAS	DBAS Dumps primitive block PBLKn in ASCII format.
Dump Block in Internal Binary	DBBN	DBBN Dumps primitive block PBLKn in internal binary format.
Dump Coordinate Transform Block in ANSI Binary	DCAN	DCAN Dumps coordinate transform block in ANSI format.
Dump Coordinate Transform Block in ASCII	DCAS	DCAS Dumps coordinate transform block in ASCII format.
Dump Coordinate Transform Block in Internal Binary	DCBN	DCBN Dumps coordinate transform block in internal binary format.
Dump Data in ANSI Binary	DDAN	DDAN Dumps active trace in ANSI format.
Dump Data in ASCII	DDAS	DDAS Dumps active trace in ASCII format.
Dump Data in Internal Binary	DDBN	DDBN Dumps active trace in internal binary format.
Dump State in ANSI Binary	DSAN	DSAN Dumps state in ANSI format.
Dump State in ASCII	DSAS	DSAS Dumps state in ASCII format.
Dump State in Internal Binary	DSBN	DSBN Dumps state in internal binary format.
Dump Table in ANSI Binary	DTAN	DTAN Dumps synth/curve fit table in ANSI.
Dump Table in ASCII	DTAS	DTAS Dumps synth/curve fit table in ASCII.
Dump Table in Internal Binary	DTBN	DTBN Dumps synth/curve fit table in internal binary.
Dump Vector Display Buffer in ANSI Binary	DVAN	DVAN Dumps vector display buffer VBLKn in ANSI format.
Dump Vector Display Buffer in ASCII	DVAS	DVAS Dumps vector display buffer VBLKn in ASCII format.
Dump Vector Display Buffer in Internal Binary	DVBN	DVBN Dumps vector display buffer VBLKn in internal binary format.

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Enable Sweep SRQ	ESWQ	ESWQ
Error Code Query	ERR?	ERR? Returns error code; refer to table 8 for description.
Error SRQ Disable	ERRD	ERRD
Error SRQ Enable	ERRE	ERRE
Exponential Average	XAVG	XAVGn1,n2,awf Exponentially averages n1 with n2 and puts the results in n2. awf is the average weighting factor (a power of two).
Float Block	FLTB	FLTBn1,n2 [, count] Converts integers in block n1 to floating point (real) and puts results in n2. Count is optional point count.
Graph Real Block	GRBL	GRBLn,x,Δx Where n is active buffer x is starting location Δx is increment.
Graph Imaginary Part of Complex Block	GRIM	GRIMn,x,Δx Where n is active buffer x is starting location Δx is increment.
Graph Real Part of Complex Block	GRRE	GRREn,x,Δx Where n is active buffer x is starting location Δx is increment.
Histogram	HST	HSTn1,n2,vmax n1 contains the new input data, n2 is the histogram count block, and vmax in the maximum absolute amplitude range for n1.
HP Logo	LOGO	LOGO0 = logo off for plots LOGO1 = logo on for plots
HP-IB Trigger	HPT	HPT
Identify	ID?	ID? outputs 7-character string "HP3562A"
Instrument Status	IS?	IS? Returns instrument status register contents.
Instrument Status Mask	ISM	ISMn where n is decimal equivalent of sum of bits to be unmasked.
Integrate Block	INGB	INGBn1[,n2] Integrates n1 and puts result in n2. If n2 is not specified, result is put in n1.
Key Press Simulation	KEY	KEYnn Where nn is key code from 0 to 70.

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Key Press Query	KEY?	KEY? Returns key code of last key pressed.
Key Press SRQ Disable	KEYD	KEYD
Key Press SRQ Enable	KEYE	KEYE
L (long exponent imag specifier)	L	Used in scientific notation as an exponent indicator, as is "E" and "D".
Label User SRQs One through Eight	LBS1 LBS2 LBS3 LBS8	LBSn'aaaaaa[,bbbbbb]' Where n is softkey number, aaaaaa is top line, bbbbbb is bottom line LBS8
Line Type	LT	LTn Where values for n are: 0 = solid lines (default) 1 = solid lines with endpoints 2 = long dashes 3 = short dashes.
Load Block in ANSI Binary	LBAN	LBAN Loads primitive block PBLKn in ANSI format.
Load Block in ASCII	LBAS	LBAS Loads primitive block PBLKn in ASCII format.
Load Block in Internal Binary	LBBN	LBBN Loads primitive block PBLKn in internal binary format.
Load Data in ANSI Binary	LDAN	LDAN Loads active trace in ANSI format.
Load Data in ASCII	LDAS	LDAS Loads active trace in ASCII format.
Load Data in Internal Binary	LDBN	LDBN Loads active trace in internal binary format.
Load State in ANSI Binary	LSAN	LSAN Loads state in ANSI format.
Load State in ASCII	LSAS	LSAS Loads state in ASCII format.
Load State in Internal Binary	LSBN	LSBN Loads state in internal binary format.
Load Table in ANSI Binary	LTAN	LTAN Loads synth/curve fit table in ANSI format.
Load Table in ASCII	LTAS	LTAS Loads synth/curve fit table in ASCII format.

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Load Table in Internal Binary	LTBN	LTBN Loads synth/curve fit table in Internal binary format.
Load User Display in ANSI Binary	LUAN	LUAN Loads active user display buffer in ANSI format.
Load User Display in ASCII	LUAS	LUAS Loads active user display buffer in ASCII format.
Load User Display in Internal Binary	LUBN	LUBN Loads active user display buffer in internal binary format.
Move Block	MOVB	MOVBn1,n2[,count] Moves n1 to n2. Optional count is used to move partial blocks.
Move Complex	MOVX	MOVXn1,n2,n3 [, count] Moves complex number n1,n2 (real,imag) into complex block n3.
Move Constant	MOVC	MOVCn1,n2[,count] Moves real constant n1 into n2. Optional count moves partial blocks.
Multiply Blocks	MPYB	MPYBn1,n2[,n3] Multiplies n1 by n2 and puts results in n3. If n3 is not specified, results are put in n2.
Multiply Block by Complex Constant	MPYX	MPYXn1,n2,n3[,n4] Multiplies complex block n3 by complex constant n1,n2 (real,imag). Result is put in n4 if specified, n3 otherwise.
Multiply Block by Real Constant	MPYC	MPYCn1,n2[,n3] Multiplies n2 by constant n1 and puts result in n3 is specified, n2 otherwise.
Multiply Block by jω	MPJW	MPJWω_{start},$\Delta\omega$,n1[,n2] Multiplies n1 by jω and puts result in n2 if specified, n1 if not. ω_{start} is the floating point starting value and $\Delta\omega$ is the incremental value of ω.
Multiply Block by Magnitude Squared	MPMG	MPMGn1,n2[,n3] multiplies real floating point block n1 by the magnitude squared of the complex floating point block n2 and puts the result in n3 if specified, n1 if not.
Multiply Block by Self Conjugate	MPSC	MPSC n1 [,n2]; Multiplies n1 by complex conjugate of n1 and puts the result in n2 if specified, n1 if not.
Negate Block	NEGB	NEGB n1 [,n2]; Negates n1 and puts result in n2 if specified, n1 if not.
Partial Block Clear	PCLR	PCLRn1,n Clears the first n points in n1.

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Pause	PAUS	PAUS
Peak Hold	PKHD	PKHDn1,n2 Compares values in blocks n1 and n2 and puts larger values in n2.
Pen Down	PD	PD
Pen Up	PU	PU
Plot Absolute	PA	PAx,y Where x is X-axis location; y is Y-axis location.
Plot Complex Block	PCBL	PCBLn1 Converts complex floating point block n1 to display format and puts it in the active display buffer.
Plot Real Block	PRBL	PRBLn1, n2 Converts real floating point block n1 vs. n2 to display format and puts it in the active buffer.
Plot Relative	PR	PRx,y Where x is relative X-axis location, y is relative Y-axis location.
Point Count	PTCT	PTCTn1,n2 Where n1 is block number, n2 is number of points.
Power Spectrum Exponential Average	PXAV	PXAVn1,n2,awf Computes power spectrum of the complex floating point block n1 and exponentially averages it with real floating point block n2. awf is the average weighting factor (a power of two).
Power Spectrum Peak Hold	PPEK	PPEKn1,n2 Computes power spectrum of n1 and compares the magnitudes of the result with real block n2, putting the larger values in n2.
Power Spectrum Summation	PSPS	PSPS n1, n2; Computes power spectrum of complex floating point block n1 and adds it to the real floating point block n2. The result is put into n2.
Primitive Block Number	PBLK	PBLKn1 Where n1 is the primitive block number, 0 to 31.
Read Marker	RDMK	RDMK Outputs 2 ASCII values, X-axis value then Y-axis value.
Read Special Marker Once	RSMO	RSMO See Chapter 6
Read Special Marker Group	RSMG	RSMG See Chapter 6

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Ready Query	RDY?	RDY? Always returns "1"
Ready Bit Disable	RDYD	RDYD
Ready Bit Enable	RDYE	RDYE
Real Fast Fourier Transform	RFFT	RFFTn1,n2 Performs real FFT on n1 and puts result in n2.
Real Inverse Fast Fourier Transform	RFT1	RFT1n1,n2 Performs real inverse FFT on n1 and puts result in n2.
Reject	REJT	REJT
Remote Entry Disable	REND	REND
Remote Entry Enable	RENE	RENE
Remote Entry Speed	RENS	RENSn, max where n is 0 for constant accelaration, >1 for variable accelaration. Max is maximum entry velocity.
Remote Entry Value	RENV	RENVn where n is value
Remote Marker Disable	RMKD	RMKD
Remote Marker Enable	RMKE	RMKE
Remote Marker Value	RMKV	RMKVn where n is value
Reset	RST	RST
Revision	REV?	REV? Outputs software revision date code and the revision date of the applicable codes and format document to which the software was designed.
Send Auto Carrier	SACR	SACR Returns 4 values: Auto carrier 1 Auto carrier 2 Phase offset 1 Phase offset 2
Send Measurement Done	SMSD	SMSD Returns a "1" if measurement is done, "0" if not
Send Missed Sample	SMSP	SMSP Returns a "1" if sample was missed; "0" if not
Send Overflow Status Channel 1	SOV1	SOV1 Returns 1 if over range, 0 if not
Send Overflow Status Channel 2	SOV2	SOV2 Returns 1 if over range, 0 if not

[ ] indicates optional parameter

Bus Only Commands (cont)

COMMAND	MNEM	SYNTAX/DATA FORMAT
Send Reference Locked	RLOK	RLOK Returns a "1" if locked; "0" if trying to lock externally.
Send Source Fault	SFLT	SFLT Returns 1 if source fault.
Send Sweep Point	SSWP	SSWP Returns five values: Input power Output power Cross spectrum real Cross spectrum imaginary Frequency First 4 are real; frequency is long real
Serial Number Query	SER?	SER? Outputs a 10-character string: serial number prefix (4 integers), country of origin (1 letter) and 5 zeros.
Setup State	SET	SET Loads instrument state; interchangeable with LSAN.
Setup State Query	SET?	SET? Dumps instrument state; interchangeable with DSAN.
SRQ Disable	SRQE	SRQD
SRQ Enable	SRQD	SRQE
State	STAT	STAT
Status/Event Query	STA?	STA? See Chapter 6
Subtract Block	SUBB	SUBBn1,n2[,n3] Subtracts n2 from n1 and puts result in n3 if specified, n2 otherwise.
Subtract Complex Constant	SUBX	SUBXn1,n2,n3[,n4] Subtracts complex constant n1,n2 (real,imag) from n3 and stores it in n4 if specified, n3 otherwise.
Subtract Real Constant	SUBC	SUBCn1,n2[,n3] Subtracts n1 from n2 and stores result in n3 if specified, n2 otherwise.
Time-out enable	TMOE	TMOE
Time-out disable	TMOD	TMOD
Unfloat Block	UFLB	UFLBn1,n2[,count] Converts floating point block n1 to integers and puts result in n2. Optional count partially unfloats n1.
Vector Display Buffer	VBLK	Points to vector display buffer to be dumped with DVAN, DVAS, DVBN.
Write Text	WRIT	WRIT'aaaaaa' where aaaaaa are alphanumeric characters.
Exponential averaging	XAVG	XAVG n1, n2, awf See Chapter 6.

STATUS BYTE

Table 2 shows the eight bits in the HP 3562A's status byte. The status byte is read by serial polling the analyzer (which also clears the status byte). Five of these bits are encoded; refer to table 3 for the condition codes. Chapter 6 provides complete explanations of the status byte conditions.

Table 2 The HP 3562A's Status Byte

Bit	Value	Description
7	128	see table 3
6	64	RQS (HP 3562A requested service)
5	32	ERR (HP-IB error)
4	16	RDY (ready to accept HP-IB commands)
3	8	see table 3
2	4	see table 3
1	2	see table 3
0	1	see table 3

Bit 6 (RQS) is set when the HP 3562A sends an SRQ. Bit 5 (ERR) is set when an HP-IB error has been made. Bit 4 (RDY) is set when the analyzer is ready to receive HP-IB commands.

Table 3 shows the condition codes represented by bits 7, 3, 2, 1 and 0 in the status byte.

Table 3 Status Byte Condition Codes

Status bit Numbers 7 3 2 1 0	Status Byte Value	Description
0 0 0 0 0	0	No service requested
0 0 0 0 1	1	User SRQ #1
0 0 0 1 0	2	User SRQ #2
0 0 0 1 1	3	User SRQ #3
0 0 1 0 0	4	User SRQ #4
0 0 1 0 1	5	User SRQ #5
0 0 1 1 0	6	User SRQ #6
0 0 1 1 1	7	User SRQ #7
0 1 0 0 0	8	User SRQ #8
0 1 0 0 1	9	End of disc action
0 1 0 1 0	10	End of plot action
0 1 0 1 1	11	Instrument status change
0 1 1 0 0	12	Power up
0 1 1 0 1	13	Key pressed
0 1 1 1 0	14	Device Clear Plotter, Listen HP 3562A
0 1 1 1 1	15	Unaddress Bus, Listen HP 3562A
1 0 0 0 0	128	Talk plotter, Listen HP 3562A
1 0 0 0 1	129	Talk disc execution. Listen HP 3562A
1 0 0 1 0	130	Talk disc report, Listen HP 3562A
1 0 0 1 1	131	Talk Amigo disc command, Listen HP 3562A
1 0 1 0 0	132	Talk Amigo disc data, Listen HP 3562A
1 0 1 0 1	133	Talk Amigo short status, Listen HP 3562A
1 0 1 1 0	134	Talk disc identify, Listen HP 3562A
1 0 1 1 1	135	Talk Amigo parallel poll, Listen HP 3562A
1 1 0 0 0	136	Listen Plotter, Talk HP 3562A
1 1 0 0 1	137	Listen disc command, Talk HP 3562A
1 1 0 1 0	138	Listen disc execution, Talk HP 3562A
1 1 0 1 1	139	Listen Amigo disc command, Talk HP 3562A
1 1 1 0 0	140	Listen Amigo disc data, Talk HP 3562A
1 1 1 0 1	141	Listen Amigo disc read, Talk HP 3562A
1 1 1 1 0	142	Listen Amigo disc write, Talk HP 3562A
1 1 1 1 1	143	Listen Amigo disc format, Talk HP 3562A

Masking SRQ Conditions in the Status Byte

When a condition is "masked," it is prevented from generating an SRQ when it becomes true. Table 4 shows how to mask the status byte conditions that can be masked. Conditions that cannot be masked are noted as well.

Table 4 Masking Status Byte Conditions

Condition	How to Mask
0	not maskable (never generates an SRQ)
1 – 8	not maskable
9 – 10	unmasked with SRQE; masked with SRQD
11	unmasked with ISMn, where n is decimal equivalent of the bits in the IS register to be unmasked. This bit is *completely* masked by sending ISM0.
12	masked with PSRQ0; unmasked with PSRQ1
13	masked with KEYD; unmasked with KEYE
14 – 143	not maskable.

SRQs are generated only by the status byte; the instrument status (IS) and activity status (AS) registers must generate SRQs indirectly through the status byte. The IS register can generate an SRQ if condition 11 in the status byte is enabled. The AS register is twice removed: bit 13 of the IS register and condition 11 of the status byte must be enabled for the AS to generate an SRQ. Chapter 6 has all the details.

THE INSTRUMENT STATUS REGISTER

Table 5 shows the instrument status (IS) register. The contents of the IS are read by sending the IS? command (which also clears the register). Unlike the status byte, the IS is not encoded: each bit represents a single condition. Complete information on the IS register is provided in Chapter 6. Remember that bit 11 in the status byte must be enabled (unmasked) before the IS can indirectly generate an SRQ.

Table 5 Instrument Status Register

Bit	Value	Condition
0	1	Measurement pause
1	2	Autosequence pause
2	4	End of measurement
3	8	End of autosequence
4	16	Sweep point ready
5	32	Channel 1 over range
6	64	Channel 2 over range
7	128	Channel 1 half range
8	256	Channel 2 half range
9	512	Source fault
10	1024	Reference unlocked
11	2048	Remote marker knob turn
12	4096	Remote entry knob turn
13	8192	Activity status register change
14	16384	Power-on test failed

Bits in the IS are masked with the ISMn command, where n is the decimal equivalent of the sum of the values of the bits to be unmasked. For example, ISM20 enables (unmasks) bit 2 (value = 4) and bit 4 (value = 16). All other bits are masked.

The Status Query (STA?)

The status query command (STA?) provides some information from both the status byte and the instrument status register. Sending STA? causes the HP 3562A to return the 16-bit word shown in table 6.

Table 6 The STA? Word

Bit	Value	Condition/Event
0	1	Not used
1	2	Not used
2	4	Key pressed
3	8	Not used
4	16	RDY
5	32	ERR
6	64	RQS
7	128	Message on screen
8	256	Measurement pause
9	512	Auto sequence pause
10	1024	End of measurement
11	2048	End of auto sequence
12	4096	Sweep point ready
13	8192	Channel 1 over range
14	16384	Channel 2 over range
15	32768	Not used

THE ACTIVITY STATUS REGISTER

Table 7 shows the activity status (AS) register. The contents of the AS are read by sending the AS? command (which does not clear the register). Unlike the status byte and like the IS, the AS is not encoded: each bit represents a single condition. Complete information on the AS register is provided in Chapter 6. Remember that *both* bit 13 of the IS and condition 11 of the status byte must be enabled before the AS can indirectly generate an SRQ.

Table 7 Activity Status Register

Bit	Value	Condition
0	1	Check fault log
1	2	Filling time record
2	4	Filters settling
3	8	Curve fit in progress
4	16	Missed sample
5	32	Time preview
6	64	Accept data
7	128	Waiting for trigger
8	256	Waiting for arm
9	512	Not used
10	1024	Ramping Source
11	2048	Diagnostic in Progress
12	4096	Marker Calc in Progress

Bits in the AS are masked with the ASMLn and ASMHn commands, where n is the decimal equivalent of the sum of the values of the bits to be unmasked. ASML unmasks for the negative-going transition; ASMH unmasks for the positive-going transition.

ERROR CODES

The Error query (ERR?) causes the analyzer to return the error code of the last HP-IB error. Each error code has a corresponding description in table 8. Note that these are the same errors as those encountered in front panel operation. For complete descriptions, with suggested corrective actions, refer to Appendix B of the *HP 3562A Operating Manual.*

Table 8 Error Codes

Code	Error
100	No Peak Avg in HIST Meas
101	No Peak Avg in CORR Meas
102	Freq Resp, No 1 Ch Demod
103	Cross Corr, No 1 Ch Demod
104	No Fundamental
105	X Marker Must Be Active
106	Buffer Overflow
107	No Coord Change Allowed
108	Not In Frequency Domain
109	No Data
110	Measurement In Progress
111	Trace Not Compatible
112	Data Type Incompatible
113	Data Blocks Incompatible
114	Source Block Empty
115	User Display Not Enabled
116	No Active Display Buffer
117	Recursive Call
118	Not A Valid Auto Math
119	Bad Setup State
120	Bad Auto Sequence Table
121	Bad Synth Table
122	Bad Non-Volatile State
123	Bad Data Block
124	Bad Data Header
125	Marker Not On
126	No Valid Marker Units
127	No Capture Data
128	No Thruput Data
129	Thruput Data Too Long
130	Bad Curve Fit Table
131	Bad Capture
132	Bad Thruput
133	Not A Valid User Window
134	Bad Primitive Block
135	View Input Disabled
136	Cannot Use Zoom Data
137	Already Running
138	May Be Inaccurate
139	Cannot Be Complex
140	Bad Delete Freq Table
141	Loops Nested Too Deep
142	Demod In Zoom Only
143	Numeric Overflow
144	Invalid: Nyquist/Nichols
145	Invalid: Log Data
146	No Carrier
147	No Peak Hold in Time Avg
148	Calibration in Progress
149	No Avg For Demod Hist
200	Not Active Softkey
201	Unknown Mnemonic
202	Line Too Long
203	Command Too Long
204	Alpha Delimiter Expected
205	Not A Valid Terminator
206	Extra Chars In Command
207	Function Inactive
300	Missing Input
301	Not Valid Units
302	Not A Valid Number
303	Alpha Too Long
304	Number Too Long
305	Out Of Range
306	Unable To Curve Fit
307	Bad # Of Parameters
308	Auto Carrier Selected
309	ENTRY Not Enabled
400	Not A Valid Block Length
401	Not A Valid Block Mode
402	Not HP-IB Controller
403	HP-IB Time Out
500	Bad Plotter Data Read
600	Cannot Recall Throughput
601	Not A Valid Catalog
602	Unformatted Disc
603	Catalog Full
604	Not A Valid Name
605	Not A Valid Display
606	File Not Found
607	Disc Full
608	Disc Reject
609	Recall Active Auto Seq
610	Unknown Disc Command Set
611	No Disc In Drive
612	Disc Write Protected
613	Disc Fault
614	Disc Transfer Error
615	No Spares Or Fault Areas
616	No Thruput File
617	Catalog Not In Memory
618	File Size Not Specified
619	Select Capture To Recall
620	Source = Destination
621	Sector Size < > 256 Bytes
622	Not Valid Format Option
623	Not Valid For This Disc
624	Destination Too Small

KEY CODES

Table 8 lists the HP 3562A's key codes. Note that the eight softkey buttons have unique codes, but individual softkey labels do not. The code of the last key pressed (since power-up or reset) is returned by the KEY? command. Key presses are simulated by sending the analyzer the KEYn command, where n is the code of the key to be simulated.

Table 8 Key Codes

Key Name	Code	Key Name	Code
No Key Pressed	0		
ENGR UNITS	1	Softkey 4	36
INPUT COUPLE	2	Softkey 5	37
TRIG DELAY	3	Softkey 2	38
HP-IB FCTN	4	Softkey 1 (top)	39
DISC	5	Softkey 3	40
SELECT TRIG	6	5	41
CAL	7	6	42
RANGE	8	4	43
AVG	9	Softkey 7	44
SELECT MEAS	10	Softkey 6	45
WINDOW	11	1	46
LOCAL	12	3	47
PLOT	13	2	48
SOURCE	14	MARKER VALUE	49
FREQ	15	- (negative sign)	50
MEAS MODE	16	BACKSPACE	51
START	17	Softkey 8 (bottom)	52
SPCL FCTN	18	VIEW INPUT	53
PRESET	19	0	54
MATH	20	, (comma)	55
SYNTH	21	. (decimal point)	56
AUTO SEQ	22	A	57
PAUSE CONT	23	B	58
SAVE RECALL	24	A&B	59
Y <marker>	25	COORD	60
SPCL MARKER	26	MEAS DISP	61
HELP	27	ARM	62
AUTO MATH	28	SINGLE	63
CURVE FIT	29	UPPER LOWER	64
X OFF	30	STATE TRACE	65
X	31	UNITS	66
Y OFF	32	FRONT BACK	67
8	33	SCALE	68
9	34	UP arrow	69
7	35	DOWN arrow	70

EXAMPLES

PURPOSE OF THIS APPENDIX

This appendix contains example HP BASIC 3.0 programs written for for the HP 3562A. These programs were written to provide you with with ideas for controlling the HP 3562A via HP-IB. They are not intended to be final solutions to any particular programming problems, but rather to demonstrate the analyzer's power and flexibility.

NOTE

These programs are not warranted, guaranteed, or supported by Hewlett-Packard or any of its representatives in any manner whatsoever.

```
20    !    APPENDIX C - EXAMPLE PROGRAM 1
30    !///////////////////////////////////////////////////////////
40    !
50    !  DEMO PROGRAM PASS CONTROL
60    !
70    ! (c)   COPYRIGHT 1985, Hewlett-Packard Co.
80    !       last update 4-23-85
90    !       BASIC    3.0
100   !
110   !  PURPOSE:
120   !
130   !    This program responds to a request for sevice by
140   !    the HP3562A so that it can make direct digital
150   !    plots, etc., while attached to a Series 200
160   !    controller by passing control to the analyzer.
170   !
180   !  DATA DICTIONARY:
190   !
200   !
210   !   Spoll_byte    Masked serial poll byte
220   !
230   !   @Io           HP-IB code assignment of the 3562
240   !
250   !   Hpib_intr     HPIB interupt service routine
260   !
270   !//////////////////////////////////////////////////////////////
280   !
290   ASSIGN @Io TO 720
300   LOCAL @Io
310   ON INTR 7 GOSUB Hpib_intr
320   ENABLE INTR 7;2
330   !
340 W_loop:GOTO W_loop     ! Wait for interupt
350   !
360 Hpib_intr:  ! Pass control interrupt service routine
370   Spoll_byte=BINAND(SPOLL(@Io),143)! MASK OUT BITS 4,5,6
380   IF Spoll_byte>=14 AND Spoll_byte<=143 THEN
390     SEND 7;UNL UNT TALK 20 CMD 9
400     GOTO End_intr
410   END IF
420 End_intr:ENABLE INTR 7
430   RETURN
440   END
```

```
20    !   APPENDIX C - EXAMPLE PROGRAM 2
30    !//////////////////////////////////////////////////////////
40    !
50    !  DEMO PROGRAM DUMP DATA TRACE
60    !
70    !  (c)  COPYRIGHT 1985, Hewlett-Packard Co.
80    !       last updated 4-23-85
90    !       BASIC    3.0
100   !
110   !
120   !  PURPOSE:
130   !
140   !    This program will read data directly from a
150   !    HP3562A analyzer over the HP-IB bus
160   !    using the capability of the series 200.
170   !    The data is assumed to be linear
180   !    resolution data and is plotted; if complex,
190   !    in real and imag formats.
200   !
210   !  DATA DICTIONARY:
220   !
230   !   Max_val(*)    The data array max and/or min value used
240   !   Min_val(*)    in determining the plotting limit.
250   !
260   !   Header_len    Data header length (constant)
270   !
280   !   Data_len      Data buffer length (bytes)
290   !
300   !   N_points      Number of data points
310   !
320   !   Start_f       Start frequency
330   !
340   !   Delta_f       Frequency or time spacing
350   !
360   !   Hbuf(*)       Real buffer containing data header
370   !
380   !   Fbuf(*)       Real buffer containing data trace
390   !
400   !//////////////////////////////////////////////////////////
410   !
420   INTEGER I,Real,Imag,Mag,Phase
430   DIM A$[2],Max_val(1:2),Min_val(1:2)
440   Real=1
450   Imag=2
460   GINIT
470   !
480   Header_len=66             ! Data header length
490   ASSIGN @Io TO 720
500   REMOTE @Io
510   !
520   ! GET DATA
530   !
```

```
540    DISP "DUMP DATA"
550    OUTPUT @Io;"DDAN"        ! Dump data ANSI format
560    ENTER @Io USING "#,2A,W";A$,Data_len
570    ASSIGN @Io;FORMAT OFF    ! Turn ASCII formatter off
580    ALLOCATE REAL Hbuf(1:Header_len)
590    ENTER @Io;Hbuf(*)        ! Read data header
600    !
610    !  EXTRACT HEADER INFORMATION
620    !
630    N_points=Hbuf(2)         !Number of data points
640    Cmplx_flg=Hbuf(37)       !Complex data flag
650    Start_f=Hbuf(66)         !Data start frequency
660    Delta_f=Hbuf(56)         !Delta frequency or time
670    IF Cmplx_flg=1 THEN
680      ALLOCATE Fbuf(1:N_points,1:2)
690    ELSE
700      ALLOCATE Fbuf(1:N_points,1:1)
710    END IF
720    ENTER @Io;Fbuf(*)        !Read data trace
730    ASSIGN @Io;FORMAT ON
740    DISP "DATA TRANSFER COMPLETE"
750    !
760    !FIND MAX VALUE
770    !
780    DISP "FINDING MAX MIN FOR PLOT"
790 Max_min:     !Calculates the MAX and MIN for plotting
800          !Initialize Variables
810    Max_val(Real)=0          !Real trace Max
820    Max_val(Imag)=0          !Imag trace Max
830    Min_val(Real)=0          !Real trace Min
840    Min_val(Imag)=0          !Imag trace Min
850    FOR I=1 TO N_points         ! Find Max's and Min's
860      FOR J=1 TO Cmplx_flg+1
870        IF Fbuf(I,J)>Max_val(J) THEN Max_val(J)=Fbuf(I,J)
880        IF Fbuf(I,J)<Min_val(J) THEN Min_val(J)=Fbuf(I,J)
890      NEXT J
900    NEXT I
910 Plot_out:    !   Plots data
920    GCLEAR
930    GRAPHICS ON
940    X_min=Start_f
950    X_max=(N_points-1)*Delta_f+Start_f
960    ALPHA OFF
970    FOR K_funct=1 TO Cmplx_flg+1
980      IF Cmplx_flg=0 THEN
990        VIEWPORT 10,110,15,85
1000     ELSE
1010       VIEWPORT 10,110,15,48
1020     END IF
1030     IF K_funct=2 THEN VIEWPORT 10,110,53,85
1040     WINDOW X_min,X_max,Min_val(K_funct),Max_val(K_funct)!
```

```
1050     MOVE Start_f,Fbuf(1,K_funct)
1060     FOR I=2 TO N_points
1070       PLOT Start_f+(I*Delta_f),Fbuf(I,K_funct)
1080     NEXT I
1090   NEXT K_funct
1100   !
1110 Border:  !  Plots border around data
1120   VIEWPORT 10,110,10,90
1130   WINDOW 0,1000,0,1000
1140   MOVE 0,500
1150   DRAW 0,1000
1160   PLOT 1000,1000
1170   PLOT 1000,500
1180   PLOT 0,500
1190   PLOT 0,0
1200   PLOT 1000,0
1210   PLOT 1000,500
1220   !
1230   DISP ""
1240   LOCAL @Io
1250   END
```

```
20    !    APPENDIX C - EXAMPLE PROGRAM 3
30    !//////////////////////////////////////////////////////////////
40    !
50    !  DEMO PROGRAM 1/3 RD OCTAVE
60    !
70    ! (c)   COPYWRIGHT 1985, Hewlett-Packard Co.
80    !       last update 4-23-85
90    !       BASIC    3.0
100   !
110   !  PURPOSE:
120   !
130   !    This program will read data directly from a
140   !    HP3562A analyzer over the HP-IB of the HP9000
150   !    Series 200 controller.
160   !    The data is assumed to be in Log res mod and
170   !    amplitude units of vlt^2 it is converted to
180   !    a psuedo 1/3 octave format and dumped back
190   !    to the HP3562A analyzer.
200   !
210   !
220   !
230   !  SUB PROGRAMS REQUIRED:
240   !
250   !    F_shape      Computes the ANSI class III filter shape
260   !
270   !  DATA DICTIONARY:
280   !
290   !
300   !   Header_len    Data header length (constant)
310   !
320   !   Data_len      Data buffer length (bytes)
330   !
340   !   N_points      Number of data points
350   !
360   !   Start_f       Start frequency
370   !
380   !   Delta_f       Frequency spacing in dec/pt
390   !
400   !   Pt_dec        Points per decade
410   !
420   !   Hbuf(*)       Real buffer containing data header
430   !
440   !   Fbuf(*)       Real buffer containing log res data
450   !
460   !   Oct_buf(*)    Buffer with synthesized 1/3 oct data
470   !
480   !//////////////////////////////////////////////////////////////
490   !
500   INTEGER I,N_points,Header_len,Pt_dec,N_fact,Flag
510   N_fact=32                        !+- NUMBER OF LINES IN 1/3
520   ALLOCATE Trans(-N_fact:N_fact)   !   OCT FILTER
530   !
```

```
540   Header_len=66
550   GOSUB Get_data         !Gets data from the HP3562A
560   PRINT "GOT DATA"
570   GOSUB Get_pwr          !Reads total power using markers
580   GOSUB Oct_1_3          !Calculates 1/3d Octave Spec
590   GOSUB Restore_dat      !Restores data to Analyzer
600   LOCAL @Io
610   GOTO W_loop
620 Get_pwr:  !              !Reads power using power marker
630   OUTPUT @Io;"XOFF;PWR;RSMO"
640   ENTER @Io;Pwr_a,Pwr_b
650   RETURN
660 Get_data:                !Reads data block
670   ASSIGN @Io TO 720
680   ASSIGN @Io;FORMAT ON
690   REMOTE 720
700   OUTPUT @Io;"COME"
710   OUTPUT @Io;"DDAN"
720   PRINT "DUMP DATA"
730   ENTER @Io USING "#,2A,W";A$,Data_len
740   ASSIGN @Io;FORMAT OFF
750   ALLOCATE REAL Hbuf(1:Header_len)
760   ENTER @Io;Hbuf(*)
770   CALL Fshape(Trans(*),N_fact)   !Calculates 1/3d Oct filter
780   !
790   !  EXTRACT HEADER INFORMATION
800   N_points=Hbuf(2)
810   Cmplx_flg=Hbuf(37)
820   Log_data=Hbuf(41)
830   IF Log_data=0 THEN GOTO Fmt_error   !Data not log res
840   Amp_units=Hbuf(10)
850   IF Amp_units<>1 THEN GOTO Fmt_error !Units not Vlt^2
860   Hbuf(10)=0.
870   Start_f=Hbuf(66)
880   Pt_dec=1/Hbuf(56)      ! pts per decade
890   Delta_f=1/Pt_dec       ! in decades
900   ALLOCATE Fbuf(1:N_points)
910   ALLOCATE Oct_buf(1:N_points)
920   ENTER @Io;Fbuf(*)
930   ASSIGN @Io;FORMAT ON
940   PRINT "DATA TRANSFER COMPLETE"
950   RETURN
960   !
970 Oct_1_3:                 ! Refomats data in 1/3 Octaves
980   FOR I=1 TO N_points STEP 8
990     Oct_buf(I)=0
1000    FOR J=-(N_fact-1) TO (N_fact-1)
1010      IF (I-J)<1 OR (I-J)>N_points THEN
1020        IF (I-J)<1 THEN Oct_dum=Fbuf(1)
1030        IF (I-J)>N_points THEN Oct_dum=Fbuf(N_points)
1040      ELSE    !
```

```
1050        Oct_dum=Fbuf(I-J)
1060      END IF
1070      Oct_buf(I)=Oct_dum*Trans(J)+Oct_buf(I)
1080    NEXT J
1090    Oct_dum=Oct_buf(I)
1100    FOR J=-3 TO 4
1110      IF (I+J)>=1 AND (I+J)<=N_points THEN
1120        Oct_buf(I+J)=SQR(Oct_dum)
1130      END IF
1140    NEXT J
1150  NEXT I
1160  !
1170  PRINT " Total Power is  = ";Pwr_a;"  dB"
1180  RETURN
1190  !
1200 Restore_dat: !
1210  PRINT "RE-STORING DATA"
1220  OUTPUT @Io;"LDAN"
1230  OUTPUT @Io USING "#,2A,W";"#A",Data_len
1240  ASSIGN @Io;FORMAT OFF
1250  OUTPUT @Io;Hbuf(*);Oct_buf(*);END
1260  RETURN
1270  !
1280 W_loop:!
1290  LOCAL @Io
1300  STOP
1310 Fmt_error: !
1320  BEEP
1330  PRINT "DATA NOT IN PROPER MEAS MODE FOR"
1340  PRINT "1/3rd OCTAVE.  MEASUREMENT MUST "
1350  PRINT "BE MADE IN LOG RESOLUTION MODE "
1360  PRINT "AND IN AMP UNITS OF VLT^2      "
1370  CLEAR @Io
1380  LOCAL @Io
1390  END
1400  SUB Fshape(Trans(*),INTEGER N_fact)
1410  !
1420  !  SUB PROGRAM TO CALCULATE THE
1430  !  FILTER SHAPE OF A 1/3 RD OCT
1440  !  CLASS III FILTER
1450  !
1460    INTEGER N
1470    FOR N=-N_fact TO N_fact
1480      IF N<=4 AND N>=-4 THEN
1490        Atten=1
1500      ELSE
1510        Atten=(8/13+2500*(10^(N/80)-10^(-N/80))^6)
1520      END IF
1530      Trans(N)=1/Atten
1540    NEXT N
1550  SUBEND
```

```
20     !   APPENDIX C - EXAMPLE PROGRAM 4
30     !/////////////////////////////////////////////////////////////
40     !
50     !  DEMO PROGRAM DUMP COORDINATE TRANSFORM BLOCK
60     !
70     !  (c) COPYRIGHT 1985, Hewlett-Packard Co.
80     !       last update  3-14-85
90     !       BASIC    3.0
100    !
110    !
120    !  PURPOSE:
130    !
140    !    This program will read coord transform block from
150    !    HP3562A analyzer over the HP-IB bus using
160    !    the capability of the Series 200.
170    !    The data is assumed to be dB magnitude data and
180    !    Hz frequency domain power spectrum data.
190    !    The data is repeatedly read and displayed in a
200    !    spectral map format. Only the data actual displayed
210    !    is read and plotted.
220    !
230    !  DATA DICTIONARY:
240    !
250    !
260    !   Header_len    Data header length (constant)
270    !
280    !   Chead_len     Coordinate transform header length
290    !
300    !   Data_len      Data buffer length (bytes)
310    !
320    !   N_points      Number of data points
330    !
340    !   Cbuf(*)       Real buffer for coord transform header
350    !
360    !   Hbuf(*)       Real buffer containing data header
370    !
380    !   Buff(*)       Real buffer containing coord trans data
390    !
400    !   Mask(*)       Data buffer containing max values; used
410    !                 for hidden line calculations
420    !
430    !   Penc(*)       Pen control buffer for hidden lines
440    !
450    !/////////////////////////////////////////////////////////////
460    !
470    INTEGER I,Real,Imag,Mag,Phase,Done_flg
480    DIM A$[3]
490    Real=1
500    Imag=2
510    !
520    Header_len=66             ! Data header length
530    Chead_len=50              ! Coord transform header length
540    Done_flg=0                ! Measurement done flag
```

```
550   ASSIGN @Io TO 720
560   REMOTE @Io
570   ALLOCATE REAL Hbuf(1:Header_len),Cbuf(1:Chead_len)
580 Control:  !
590   N_spect=25
600   GOSUB Dsa_setup
610   GOSUB Get_head
620   GOSUB Plot_init
630   GOSUB Hpib_init
640   FOR K=0 TO N_spect-1
650 !
660 ! Wait for End of Measurement
670 W_data:IF Done_flg=0 THEN GOTO W_data
680 !
690     GOSUB Get_data
700     GOSUB Meas_start
710     GOSUB Plot_out
720   NEXT K
730   LOCAL @Io
740 W_loop:GOTO W_loop ! Wait (suppress softkey menu)
750   !
760 Get_data:  ! Gets data and calculates hidden lines
770   GOSUB Mask_update
780   OUTPUT @Io;"DCAN"  !Dump Coord trans Ansi
790   ENTER @Io USING "#,2A,W";A$,Data_len
800   ASSIGN @Io;FORMAT OFF
810   ENTER @Io;Cbuf(*);Hbuf(*)
820   ENTER @Io;Buff(*)
830   ASSIGN @Io;FORMAT ON
840   FOR I=0 TO N_points-1   ! Set clipping boundary
850     IF Buff(I)<Y_min1 THEN Buff(I)=Y_min1
860     IF Buff(I)>Y_max1 THEN Buff(I)=Y_max1
870   NEXT I
880    !
890    !  Set pen control for plotting
900   MAT Penc= Buff-Mask
910   FOR I=0 TO N_points-1
920     Penc(I)=SGN(Penc(I))
930   NEXT I
940   Done_flg=1
950   RETURN
960   !
970 Mask_update:  ! Does X & Y axis shifting and mask update
980   FOR I=N_points-N_delta_x TO N_points-1
990     Mask(I)=(Y_min-Delta_y)
1000  NEXT I
1010 Xshift:FOR I=N_delta_x TO N_points-1
1020    Buff(I)=Buff(I)-Delta_y
1030    Mask(I-N_delta_x)=MAX(Mask(I)-Delta_y,Buff(I))
1040  NEXT I
1050  RETURN !
```

```
1060  !
1070 Get_head: !
1080  OUTPUT @Io;"DCAN"
1090  ENTER @Io USING "#,2A,W";A$,Data_len
1100  ASSIGN @Io;FORMAT OFF
1110  ENTER @Io;Cbuf(*);Hbuf(*)     !Read Coord Trans and Data header
1120  N_points=Cbuf(2)              !Number of points
1130  ALLOCATE Buff(0:N_points-1)
1140  ENTER @Io;Buff(*)             !Read Coordinate Transform block
1150  ASSIGN @Io;FORMAT ON
1160  CLEAR @Io
1170  RETURN
1180  !
1190 Plot_out:  !
1200  !Set viewport boundaries to match spec'd min/max's
1210  X1=X_min_view+X_inc*K
1220  X2=X_min_view+X_delta_view+X_inc*K
1230  Y1=Y_min_view+Y_inc*K
1240  Y2=Y_min_view+Y_delta_view+Y_inc*K
1250  VIEWPORT X1,X2,Y1,Y2
1260  WINDOW 0,N_points-1,Y_min,Y_max    ! Set window
1270  MOVE 0,Buff(0)
1280  FOR I=1 TO N_points-1
1290    !Put Pen control in proper format
1300    Pnt_cnt=Penc(I)-1
1310    IF Pnt_cnt=0 THEN Pnt_cnt=1
1320    PLOT I,Buff(I),Pnt_cnt
1330  NEXT I
1340  RETURN
1350   !
1360   !
1370 Plot_init:! Initialize plot
1380  PLOTTER IS CRT,"INTERNAL"
1390  GINIT
1400  GCLEAR
1410  GRAPHICS ON
1420   !
1430   !  FOLLOWING PARAMETERS IN ENGINEERING UNITS
1440   !
1450  Y_min1=Cbuf(34)          ! Read Y min from header
1460  Y_max1=Cbuf(35)          ! Read Y max from header
1470  Y_scale_f=Cbuf(41)       ! Amplitude scale factor
1480  X_min=Cbuf(49)           ! Read X min from header
1490  X_max=Cbuf(50)           ! Read X max from header
1500  Y_off=ABS(.05*(Y_min1-Y_max1))  ! Cal offset= 5% full scale
1510  Y_min=Y_min1-Y_off       ! Adjust Y min
1520  Y_max=Y_max1+Y_off       !    and Y max
1530  Y_delta=Y_max-Y_min      ! Calculate Y span
1540  !
1550  ! VIEWPORT VALUES FOR INDIVIDUAL SPECTRA
1560  !       IN % OF FULL SCALE
1570  !
1580  INTEGER N_delta_x        !
```

```
1590  Y_min_view=10              ! Y min for single spectrum (in %)
1600  X_min_view=10              ! X min for single spectrum (in %)
1610  Y_delta_view=45            ! Single spectrum height (in %)
1620  X_delta_view=80            ! Single spectrum width (in %)
1630  Y_delta_bound=85           ! Entire map height (in %)
1640  X_delta_bound=100          ! Entire map width (in %)
1650  Y_inc=(Y_delta_bound-Y_delta_view)/(N_spect-1)
1660        !Y_inc is incremental vertical movement (in %)
1670  X_inc=(X_delta_bound-X_delta_view)/(N_spect-1)
1680        !X_inc is incremental horizontal movement (in %)
1690  Delta_y=Y_inc*(Y_max-Y_min)/Y_delta_view
1700        !Delta_y is incremental vert movement in plot units
1710  N_delta_x=X_inc*(N_points-1)/X_delta_view
1720        !N_delta_x is incremtal horizonal movement in number
1730        !of data points (rounded integer)
1740  !
1750  ! RECALULATE X_INC FOR INTEGER N_DELTA
1760  !
1770  X_inc=N_delta_x/(N_points-1)*X_delta_view
1780  X_delta_bound=X_inc*(N_spect-1)+X_delta_view
1790  !
1800 Init_hidden: ! Initial for hidden lines
1810  ALLOCATE Penc(0:N_points-1),Mask(0:N_points-1)
1820  MAT Buff= (Y_min1)        ! Set to Min Y value
1830  MAT Mask= (Y_min1)
1840  ALPHA OFF
1850  GOSUB Plot_axis
1860  RETURN
1870  !
1880 Plot_axis:!  DRAW THE AXIS AND BOUNDARIES OF THE PLOT
1890  X1=X_min_view
1900  X2=X_min_view+X_delta_bound
1910  Y1=Y_min_view
1920  Y2=Y_min_view+Y_delta_bound
1930  VIEWPORT X1,X2,Y1,Y2
1940  WINDOW X1,X2,Y1,Y2
1950  Offset_y=.05*Y_delta_view
1960  MOVE X1,Y1+Offset_y
1970  DRAW X1,Y1
1980  DRAW X1+X_delta_view,Y1
1990  DRAW X1+X_delta_view,Y1+Offset_y
2000  DRAW X2,Y2-Y_delta_view+Offset_y
2010  DRAW X2,Y2
2020  DRAW X2-X_delta_view,Y2
2030  DRAW X2-X_delta_view,Y2-Y_delta_view+Offset_y
2040  DRAW X1,Y1+Offset_y
2050  !
2060 Right_tics: !  DOES VERTICAL TICK MARKS
2070  ! Reset viewport and window
2080  X1=X_min_view+X_inc*(N_spect-1)
2090  X2=X_min_view+X_delta_view*1.2+X_inc*(N_spect-1)
2100  Y1=Y_min_view+Y_inc*(N_spect-1)
2110  Y2=Y_min_view+Y_delta_view+Y_inc*(N_spect-1)
2120  VIEWPORT X1,X2,Y1,Y2
2130  WINDOW 0,(N_points-1)*1.20,Y_min,Y_max !
```

```
2140  MOVE N_points-1,Y_min1
2150  DRAW (N_points-1)*1.03,Y_min1  !Draw lower tick mark
2160  CSIZE (3)
2170  LORG (2)
2180  Y_label$="dB"
2190  Y_fmt$="X,SDDD.D"
2200  LABEL USING Y_fmt$;(Y_min1*Y_scale_f)
2210  MOVE N_points-1,Y_max1
2220  DRAW (N_points-1)*1.03,Y_max1  !Draw upper tick mark
2230  LABEL USING Y_fmt$;(Y_max1*Y_scale_f)
2240  MOVE (N_points-1)*1.05,Y_min1+(Y_max1-Y_min1)*.5
2250  LABEL Y_label$
2260  !
2270 Lower_tics: ! DOES FREQUENCY AXIS
2280  ! Reset viewport and window
2290  X1=X_min_view-X_delta_view*.10
2300  X2=X_min_view+X_delta_view*1.15
2310  Y1=Y_min_view-Y_delta_view*.15
2320  Y2=Y_min_view+Y_delta_view
2330  VIEWPORT X1,X2,Y1,Y2
2340  !
2350  X1=0-(N_points-1)*.10
2360  X2=(N_points-1)*1.15
2370  Y1=Y_min-Y_delta*.15*1.1
2380  Y2=Y_max
2390  WINDOW X1,X2,Y1,Y2
2400  MOVE 0,Y_min
2410  DRAW 0,Y_min-Y_off
2420  LORG (6)
2430  X_fmt$="SDD.D"
2440  X_label$="HZ"
2450  LABEL (X_min)
2460  MOVE N_points-1,Y_min
2470  DRAW N_points-1,Y_min-Y_off
2480  LABEL (X_max)
2490  MOVE (N_points-1)*.5,Y_min-Y_off
2500  LABEL X_label$
2510  RETURN
2520 !
2530 Dsa_setup:!  SETS UP ANALYZER TO INTERUPT ON EOM
2540  OUTPUT @Io;"COMD"        ! Disable command echo
2550  OUTPUT @Io;"UNIT; HZS"   ! Sets X axis units to Hertz
2560  OUTPUT @Io;"MGDB"        ! Sets Y axix to Mag dB
2570 !
2580 Hpib_init:  !
2590  OUTPUT @Io;"SRQE "       ! Enable SRQ's
2600  OUTPUT @Io;"ISM 4"       ! End of Measurement status mask
2610  ON INTR 7 GOSUB Hpib_intr
2620  ENABLE INTR 7;2
2630 !
2640 Meas_start:  !
2650  Done_flg=0
2660  OUTPUT @Io;"STRT"        ! Start measurement
2670  RETURN !
```

```
2680 !
2690 Hpib_intr:! Processes End of Measurement and Request for
2700           ! Control interupts
2710  Spoll_byte=SPOLL(@Io)
2720  Stest_byte=BINAND(Spoll_byte,143) !  MASK OUT BITS 4,5,6
2730  IF Stest_byte>=14 AND Stest_byte<=143 THEN
2740    PASS CONTROL @Io
2750  END IF
2760  IF Stest_byte=11 THEN
2770    ASSIGN @Io;FORMAT ON
2780    OUTPUT @Io;"IS?"
2790    ENTER @Io;Stat
2800    IF BINAND(Stat,4)=4 THEN Done_flg=1
2810  END IF
2820 End_intr:ENABLE INTR 7
2830  RETURN
2840  END
```

```
20    !     APPENDIX C - EXAMPLE PROGRAM 5
30    !///////////////////////////////////////////////////////////////
40    !
50    !  DEMO PROGRAM THRUPUT SPECTRAL MAP
60    !
70    !  (c) COPYRIGHT 1985, Hewlett-Packard Co.
80    !       last updated 4-23-85
90    !       BASIC    3.0
100   !
110   !  SUBPROGRAMS REQUIRED
120   !
130   !   Int_real(INTEGER I1,I2,REAL Real)
140   !
150   !       converts data in HP3562A internal
160   !       real format to BASIC real format
170   !
180   !   L_int_real(INTEGER I1,I2,I3,I4,REAL Real)
190   !
200   !       converts data in HP3562A internal
210   !       long real format to BASIC real format
220   !
230   !  PURPOSE:
240   !
250   !      This program will read coord transform block from
260   !      the HP3562A analyzer over the HP-IB buss using
270   !      capabilities of the HP9000 Series 200 computer.
280   !      The data is assumed to be dB magnitude and
290   !      Hz frequency domain power spectrum data.
300   !      The data is assumed to reside on an auxillery
310   !      disc having been created by a thru put session.
320   !      The data is displayed in a spectral map format
330   !      and only the data ranges displayed are read
340   !      and plotted
350   !
360   !  DATA DICTIONARY:
370   !
380   !   Chead_len     Coordinate transform header length
390   !
400   !   Header_len    Data header length (constant)
410   !
420   !   Data_len      Data buffer length (bytes)
430   !
440   !   N_points      Number of data points
450   !
460   !   N_spec        Number of spectra in map(constant)
470   !
480   !   Start_f       Start frequency
490   !
500   !   Delta_f       Linear res frequency spacing
510   !
520   !   Th_head(*)    Integer buffer for thruput header
530   !
540   !   Cbuf(*)       Real buffer for coord transform header
```

```
550   !
560   !   Hbuf(*)        Real buffer containing data header
570   !
580   !   Buff(*)        Real buffer containig coord trans data
590   !
600   !   Mask(*)        Data buffer contains max values used for
610   !                  hidden line calculations.
620   !
630   !   Penc(*)        Pen control buffer for hidden lines
640   !
650   !
660   !/////////////////////////////////////////////////////////////
670   !
680   OPTION BASE 1
690   INTEGER I,Real,Imag,Mag,Phase,N_avgs,Done_flg
700   INTEGER Th_head(1:512)
710   REAL Per_ovrlp,Delay
720   DIM A$[3],Max_val(1:2),Min_val(1:2)
730   Real=1
740   Imag=2
750   !
760   Header_len=66              ! Data header length
770   Chead_len=50               ! Coord trans form header length
780   ASSIGN @Io TO 720
790   ASSIGN @Io;FORMAT ON
800   CLEAR 7
810   REMOTE @Io
820   ALLOCATE REAL Hbuf(1:Header_len),Cbuf(1:Chead_len)
830   !
840   Thru_name$="THRU"          ! Thruput file name
850                              !   MUST ALREADY EXIST
860   GOSUB Read_header          ! Reads thruput header and
870                              ! extract parameters
880   GOSUB Dsa_set              ! Sets up the analyzer
890 Intr_on:    ! ENABLE INTERUPTS FOR PASS CONTROL ETC
900   GOSUB Hpib_init
910   ON INTR 7 GOSUB Hpib_intr
920   ENABLE INTR 7;2
930   !
940 Control:  !   Loops to preform the desire number of spectrum
950   N_spect=20                 ! Constant for number of spectrum
960   FOR K=0 TO N_spect-1
970     Done_flg=0
980     OUTPUT @Io;"TRGD";Delay+K*Inc_delay;"REC"
990     OUTPUT @Io;"STRT"          ! Start measurement
1000    ENABLE INTR 7;2
1010 Meas_wait:    !  Wait for measurement to be done
1020    IF Done_flg=0 THEN GOTO Meas_wait
1030    GOSUB Plot_out           ! Plot out data
1040  NEXT K
1050   !
```

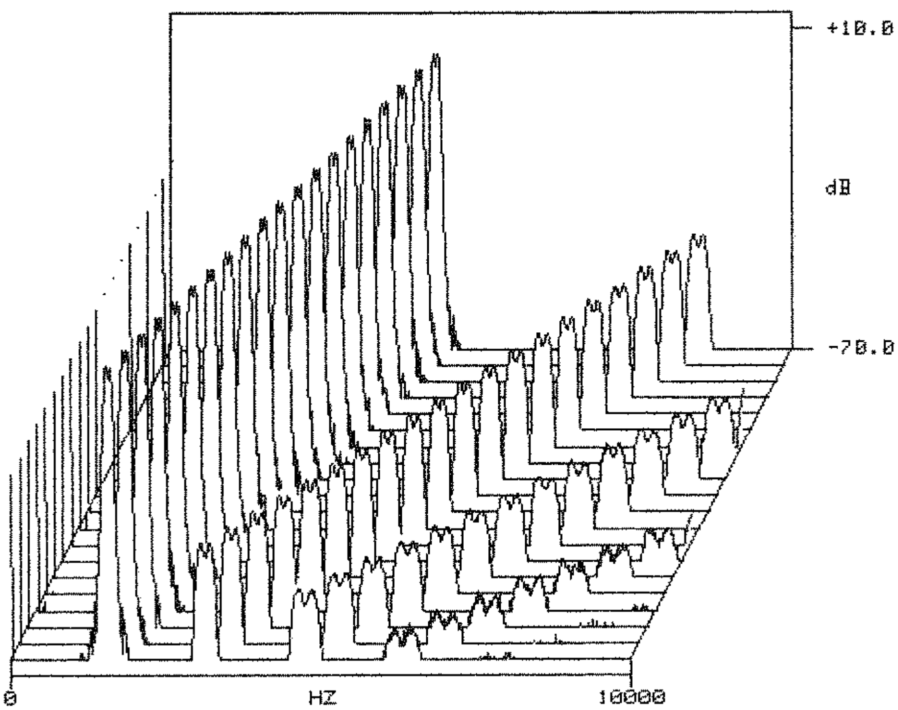
```
1060 W_loop:GOTO W_loop         ! Wait (suppresses softkey menu)
1070            !
1080 Get_data: !  Gets data and calculates hidden lines
1090  OUTPUT @Io;"DCAN"         ! Dump Coordinate Transform Block
1100                            ! in ANSI floating point format
1110  ENTER @Io USING "#,2A,W";A$,Data_len
1120  ASSIGN @Io;FORMAT OFF
1130  ENTER @Io;Cbuf(*);Hbuf(*) ! Read Coord Trans & Data headers
1140  IF K=0 THEN               ! For firs spectrum initialize
1150    N_points=Cbuf(2)        ! Read number of points from header
1160    ALLOCATE Buff(0:N_points-1)       ! Allocate buffer
1170    GOSUB Plot_init         ! Initialize Plotting
1180  END IF
1190!
1200 Mask_update:  ! Up date hidden line mask
1210 Yshift:FOR I=N_points-N_delta_x TO N_points-1
1220    Mask(I)=(Y_min-Delta_y)
1230  NEXT I
1240 Xshift:FOR I=N_delta_x TO N_points-1! Horizonal shift of mask
1250    Buff(I)=Buff(I)-Delta_y
1260    Mask(I-N_delta_x)=MAX(Mask(I)-Delta_y,Buff(I))
1270  NEXT I
1280!
1290  ASSIGN @Io;FORMAT OFF
1300  ENTER @Io;Buff(*)         !Read Coord Trans Block
1310  ASSIGN @Io;FORMAT ON
1320  FOR I=0 TO N_points-1     ! Set lower clipping value
1330    IF Buff(I)<Y_min1 THEN Buff(I)=Y_min1
1340  NEXT I
1350  MAT Penc= Buff-Mask       ! Calc pen control for plotting
1360  FOR I=0 TO N_points-1
1370    Penc(I)=SGN(Penc(I))
1380  NEXT I
1390  Done_flg=1
1400  RETURN
1410  !
1420 Plot_out:  !
1430  ! Set viewport boundaries to match specified min's & max's
1440  X1=X_min_view+X_inc*K
1450  X2=X_min_view+X_delta_view+X_inc*K
1460  Y1=Y_min_view+Y_inc*K
1470  Y2=Y_min_view+Y_delta_view+Y_inc*K
1480  VIEWPORT X1,X2,Y1,Y2
1490  WINDOW 0,N_points-1,Y_min,Y_max
1500  MOVE 0,Buff(0)
1510  FOR I=1 TO N_points-1
1520    Pnt_cnt=Penc(I)-1
1530    IF Pnt_cnt=0 THEN Pnt_cnt=1
1540    PLOT I,Buff(I),Pnt_cnt
1550  NEXT I
1560   !
1570  RETURN
1580   !
```

```
1590 Plot_init:! Initialize plot
1600  PLOTTER IS CRT,"INTERNAL"
1610  GINIT
1620  GCLEAR
1630  GRAPHICS ON
1640   !
1650   !  FOLLOWING PARAMETER IN PHYSICAL UNITS
1660   !
1670  Y_min1=Cbuf(34)           ! Read Y min from header
1680  Y_max1=Cbuf(35)           ! Read Y max from header
1690  Y_scale_f=Cbuf(41)        ! Amp scale factor
1700  X_min=Cbuf(49)            ! Read X min from header
1710  X_max=Cbuf(50)            ! Read X max from header
1720  Y_off=ABS(.05*(Y_min1-Y_max1))    ! Calc offset=5% f.s.
1730  Y_min=Y_min1-Y_off        ! Adjust Y min and
1740  Y_max=Y_max1+Y_off        !     Y max by calculated offset
1750  Y_delta=Y_max-Y_min       ! Calcutate Y span
1760   !
1770   ! VIEWPORT VALUES FOR INDIVIDUAL SPECTRA
1780   !       IN % OF FULL SCALE
1790   !
1800  INTEGER N_delta_x         ! Horizontal offset of spectrum
1810                            !   in units of # of Delta_x
1820  Y_min_view=10             ! Y min for single spectrum (in %)
1830  X_min_view=10             ! X min for single spectrum (in %)
1840  Y_delta_view=45           ! Single spectrum height (in %)
1850  X_delta_view=80           ! Single spectrum width (in %)
1860  Y_delta_bound=85          ! Entire map height (in %)
1870  X_delta_bound=100         ! Entire map width  (in %)
1880  Y_inc=(Y_delta_bound-Y_delta_view)/(N_spect-1)
1890         ! Y_inc is incremental vertical movement between
1900         !   spectrum (in %)
1910  X_inc=(X_delta_bound-X_delta_view)/(N_spect-1)
1920         ! X_inc is incremental horizonal movement between
1930         !   spectrum (in %)
1940  Delta_y=Y_inc*(Y_max-Y_min)/Y_delta_view
1950         ! Delta_y is incremental vertical movement in plot units
1960  N_delta_x=X_inc*(N_points-1)/X_delta_view
1970         ! N_delta_x is inremental vertical movement in
1980         ! number of data points (rounded integer)
1990         ! only even number of data point shift allowed
2000  ! RECALULATE X_INC FOR INTEGER N_DELTA
2010  X_inc=N_delta_x/(N_points-1)*X_delta_view
2020  X_delta_bound=X_inc*(N_spect-1)+X_delta_view
2030  !
2040 Init_hidden: ! Initialize hidden lines
2050  ALLOCATE Penc(0:N_points-1),Mask(0:N_points-1)
2060  MAT Buff= (Y_min1)
2070  MAT Mask= (Y_min1)
2080  ALPHA OFF
2090  GOSUB Plot_axis           ! Draw plot axis
2100  RETURN
2110  !
```

```
2120 Plot_axis:!  DRAW THE AXIS AND BOUNDARIES OF THE PLOT
2130  X1=X_min_view
2140  X2=X_min_view+X_delta_bound
2150  Y1=Y_min_view
2160  Y2=Y_min_view+Y_delta_bound
2170  VIEWPORT X1,X2,Y1,Y2
2180  WINDOW X1,X2,Y1,Y2
2190  Offset_y=.05*Y_delta_view
2200  MOVE X1,Y1+Offset_y
2210  DRAW X1,Y1
2220  DRAW X1+X_delta_view,Y1
2230  DRAW X1+X_delta_view,Y1+Offset_y
2240  DRAW X2,Y2-Y_delta_view+Offset_y
2250  DRAW X2,Y2
2260  DRAW X2-X_delta_view,Y2
2270  DRAW X2-X_delta_view,Y2-Y_delta_view+Offset_y
2280  DRAW X1,Y1+Offset_y
2290  !
2300 Right_tics: ! DOES VERTICAL TICK MARKS
2310  ! Reset viewport and window
2320  Kk=N_spect-1
2330  X1=X_min_view+X_inc*(N_spect-1)
2340  X2=X_min_view+1.2*X_delta_view+X_inc*(N_spect-1)
2350  Y1=Y_min_view+Y_inc*(N_spect-1)
2360  Y2=Y_min_view+Y_delta_view+Y_inc*(N_spect-1)
2370  VIEWPORT X1,X2,Y1,Y2
2380  WINDOW 0,(N_points-1)*1.20,Y_min,Y_max
2390  MOVE N_points-1,Y_min1    ! Lower tic
2400  DRAW (N_points-1)*1.03,Y_min1
2410  CSIZE (3)
2420  LORG (2)
2430  Y_label$="dB"
2440  Y_fmt$="X,SDD.D"
2450  LABEL USING Y_fmt$;(Y_min1*Y_scale_f)
2460  MOVE N_points-1,Y_max1    ! Upper tic
2470  DRAW (N_points-1)*1.03,Y_max1
2480  LABEL USING Y_fmt$;(Y_max1*Y_scale_f)
2490  MOVE (N_points-1)*1.05,Y_min1+(Y_max1-Y_min1)*.5
2500  LABEL Y_label$
2510  !
2520 Lower_tics: ! DOES FREQUENCY AXIS
2530  X1=X_min_view-X_delta_view*.10
2540  X2=X_min_view+X_delta_view*1.15
2550  Y1=Y_min_view-Y_delta_view*.15
2560  Y2=Y_min_view+Y_delta_view
2570  VIEWPORT X1,X2,Y1,Y2
2580  X1=0-(N_points-1)*.10
2590  X2=(N_points-1)*1.15
2600  Y1=Y_min-Y_delta*.15*1.1
2610  Y2=Y_max
2620  WINDOW X1,X2,Y1,Y2
2630  MOVE 0,Y_min                ! Left tic
2640  DRAW 0,Y_min-Y_off
2650  LORG (6)!
```

```
2660  X_fmt$="SDD.D"
2670  X_label$="HZ"
2680  LABEL (X_min)
2690  MOVE N_points-1,Y_min     ! Right tic
2700  DRAW N_points-1,Y_min-Y_off
2710  LABEL (X_max)
2720  MOVE (N_points-1)*.5,Y_min-Y_off
2730  LABEL X_label$
2740  RETURN
2750  !
2760 Dsa_set: ! SUBROUTINE TO SETUP DATA AQUISITION
2770  N_avg=2                       ! Number of averages per spectrum
2780  Per_ovlp=50                   ! % overlap
2790  Delay=.5                      ! Delay in records for first spectrum
2800  Inc_delay=(N_avg)*Per_ovlp/100   ! Calculates the delay between
2810                                ! spectrum in records
2820  CALL Int_real(Th_head(43),Th_head(44),Fmax) ! Converts thruput
2830                                ! header values into max frequency
2840  CALL L_int_real(Th_head(39),Th_head(40),Th_head(41),Th_head(42),Fcent)
2850                                ! Converts thruput header values into center
2860                                ! frequency
2870  !
2880  ! SET ANALYSER STATE
2890  OUTPUT @Io;"TMTH"
2900  OUTPUT @Io;"PSPC;CH1;STBL;NAVG ";N_avg;"ENT;OVLP";Per_ovlp
2910  OUTPUT @Io;"C1RG;AUTO 0"
2920  OUTPUT @Io;"C1IN;TRGD";Delay;"REC"
2930  OUTPUT @Io;"ACFL '";Thru_name$;"'"
2940  OUTPUT @Io;"FRS ";Fmax;" HZ"
2950  OUTPUT @Io;"SF 0 HZ"
2960  RETURN
2970 Hpib_init: !
2980  ! SETUP End of Measurement Mask
2990  OUTPUT @Io;"SRQE"
3000  OUTPUT @Io;"ISM 4"
3010  RETURN
3020  !
3030 Hpib_intr:     !  Handles end of measurement and request
3040                !    for control interrupts
3050  Spoll_byte=SPOLL(@Io)
3060  PRINT "SPOLL =";Spoll_byte
3070  Spoll_test=BINAND(Spoll_byte,143)!  MASK OUT BITS 4,5,6
3080  IF Spoll_test>=14 AND Spoll_test<=143 THEN
3090          ! Request for control of bus
3100    PASS CONTROL @Io
3110  END IF
3120  IF Spoll_test=11 THEN              ! End of Measurement
3130    ASSIGN @Io;FORMAT ON
3140    OUTPUT @Io;"IS?"
3150    ENTER @Io;Stat
3160    IF BINAND(Stat,4)=4 THEN GOSUB Get_data
3170  END IF
3180 End_intr:ENABLE INTR 7
3190  RETURN !
```

```
3200  !
3210 Read_header:     ! READS THRUPUT HEADER
3220  ASSIGN @Disc TO "TS"&Thru_name$&":,700,0"
3230  ENTER @Disc;Th_head(*)
3240  ASSIGN @Disc TO *
3250  RETURN
3260  !
3270  END
3279 !SUB Int_real(INTEGER I1,I2,REAL Real)
3280 !SUB L_int_real(INTEGER I1,I2,I3,I4,REAL Real)
```



SAMPLE PLOT FROM EXAMPLE PROGRAMS 4 & 5

```
20    !  APPENDIX C - EXAMPLE PROGRAM 6
30    !///////////////////////////////////////////////////////////////
40    !
50    !  DEMO PROGRAM TSUNAMI PLOT
60    !
70    ! (c) COPYRIGHT 1985, Hewlett-Packard Co.
80    !       last update 4-04-85
90    !       BASIC    3.0
100   !              w/GRAPH
110   !
120   !
130   !  PURPOSE:
140   !
150   !  This program will dump the display buffers from
160   !  display of the HP3562 using the DUMP VECTOR BINARY cmd.
170   !  The program decodes the HP1345 display commands and
180   !  translates them to HP-GL then plots it on the CRT
190   !
200   !  SUB PROGRAMS USED:
210   !
220   !    Ggplot
230   !
240   !    Read_binary
250   !
260   !  DATA DICTIONARY:
270   !
280   !   Ibuf(*)        Integer buffer for storing the dumped
290   !                  data prior to plotting
300   !
310   !   Ix,Iy,Idx      Integer in range 0 to 2048 corresponding
320   !                  to the current value of the increment for
330   !                  use with the graph command
340   !
350   !   Chr_buf        Common arrays containing the definition of
360   !                  charecter set not standard to HP-GL
370   !
380   !   First_dspbuf  First display buffer
390   !
400   !   Num_dspbuf     Number of display buffers used
410   !
420   !
430   !   Data_len       Length of buffer in bytes
440   !
450   !   @Io            HP-IB code assignment of the 3562
460   !
470   !///////////////////////////////////////////////////////////////
480   !
490   !PROGRAM TO PLOT TSUNAMI GRAPHICS BUFFER
500   OPTION BASE 1
510   DIM A$[1]
520   INTEGER Ibuf(4096),Ix,Iy,Idx
530   INTEGER Data_len
540   !
```

```
550   !
560   ! THE FOLLOWING ARE USED TO DESCRIBE NONE HPGL
570   !     STANDARD CHARACTERS USED IN HP1745 DISPLAY
580   !
590   COM /Char_buf/ Triangle(6,3),Tri_2(5,3),Sqr_rt(5,3)
600   COM /Char_buf/ Rgt_arrow(6,3),Hp_log(20,3)
610   READ Triangle(*),Tri_2(*),Sqr_rt(*),Rgt_arrow(*),Hp_log(*)
620   !
630 Triangle:DATA 0,0,6,     0,0,-2,      9,4,-1
640   DATA 0,8,-1,    0,0,-1,      0,0,7
650 Tri_2:DATA 1,0,-2,    8,0,-1,      5,8,-1
660   DATA 1,0,-1,    0,0,-2
670 Sqr_rt:DATA 0,3,-2,    1,4,-1,      3,0,-1
680   DATA 6,8,-1,    0,0,-2
690 Rgt_arrow:DATA 0,4,-2,    8,4,-1,      6,6,-1
700   DATA 6,2,-2,    8,4,-1,      0,0,-2
710 Hp_log:DATA 6,4,-2,    6,12,-1,     6,8,-2
720   DATA 9,8,-1,    9,4,-1,      11,0,-2
730   DATA 11,8,-1,   14,8,-1,     14,4,-1
740   DATA 11,4,-1
750   DATA 0,2,-2,    0,10,-1,    2,12,-1
760   DATA 18,12,-1,  20,10,-1,   20,2,-1
770   DATA 18,0,-1,   2,0,-1,     0,2,-1
780   DATA 20,0,-2
790   !
800   ASSIGN @Io TO 720
810   CREATE BDAT "PLOTFILE:,700",200    ! Creates plotfile
820   PLOTTER IS "PLOTFILE:,700","HPGL" ! Assigns plotfile
830   LORG 1
840   CSIZE 3.4,.67             ! Set character size
850   WINDOW 0,2100,0,2100      ! Set window in HP1345 units
860   DEG                       ! Default to degrees
870   OUTPUT @Io;"COMD"         ! Disable command echo
880   First_dspbuf=4            ! Start with buffer 4 (ie. ignore
890                             !     menu and command fields)
900   FOR J=First_dspbuf TO 17 ! 17 is last buffer
910     OUTPUT @Io;"VBLK ";J    ! Output buffer "J"
920     CALL Read_binary(@Io,"DVBN",Data_len,Ibuf(*))
930     IF Data_len=0 THEN End_j   ! Ignore if buffer empty
940     FOR I=1 TO Data_len DIV 2
950       CALL Ggplot(Ibuf(I),Ix,Iy,Idx)
960     NEXT I
970 End_j:NEXT J
980   PLOTTER IS 3,"INTERNAL"    ! This closes the plotfile out
990   END
1000  !
1010  SUB Ggplot(INTEGER Coordt,Xcoord,Ycoord,Delta_x)
1020  !
1030  !  This subprogram translate the HP1745 commands
1040  !     to HPGL commands.
1050  !
```

```
1060    OPTION BASE 1
1070    COM /Char_buf/ Triangle(6,3),Tri_2(5,3),Sqr_rt(5,3)
1080    COM /Char_buf/ Rgt_arrow(6,3),Hp_log(20,3)
1090    INTEGER Op_code
1100    DIM A$[1]
1110 Decode_instr: !
1120    Op_code=(BINAND(Coordt,24576))    ! Mask out opcode
1130    IF Op_code=24576 THEN GOSUB Set_cond    ! Set condition
1140    IF Op_code=0 THEN GOSUB Plot            ! Plot vector
1150    IF Op_code=8192 THEN GOSUB Graph        ! Graph vector
1160    IF Op_code=16384 THEN GOSUB Text        ! Write text
1170    SUBEXIT
1180     !
1190 Set_cond:    ! Sets Linetype
1200    IF BIT(Coordt,8)=1 AND BIT(Coordt,7)=0 THEN LINE TYPE 4
1210    IF BIT(Coordt,8)=1 AND BIT(Coordt,7)=1 THEN LINE TYPE 4
1220    IF BIT(Coordt,8)=0 AND BIT(Coordt,7)=1 THEN LINE TYPE 1
1230    IF BIT(Coordt,8)=0 AND BIT(Coordt,7)=0 THEN LINE TYPE 1
1240    RETURN
1250 Plot:          !  Plots/move pen
1260    IF BIT(Coordt,12)=0 THEN Xplot
1270    Ycoord=BINAND(Coordt,2047)
1280    IF BIT(Coordt,11)=0 THEN
1290      Pflag=-2
1300    ELSE
1310      Pflag=-1
1320    END IF
1330    PLOT Xcoord,Ycoord,Pflag
1340    RETURN
1350 Xplot:Xcoord=BINAND(Coordt,2047)
1360    RETURN
1370 Graph:    ! Graph data
1380    IF BIT(Coordt,12)=0 THEN Set_deltax
1390    Ycoord=BINAND(Coordt,2047)
1400    Xcoord=Xcoord+Delta_x
1410    IF BIT(Coordt,11)=0 THEN
1420      Pflag=-2
1430    ELSE
1440      Pflag=-1
1450    END IF
1460    PLOT Xcoord,Ycoord,Pflag
1470    RETURN
1480 Set_deltax:Delta_x=BINAND(Coordt,2047)
1490    RETURN
1500 Text:          !   Text control and output
1510    LINE TYPE 1
1520    A$=CHR$(BINAND(Coordt,255))
1530    IF BIT(Coordt,8)=0 THEN GOTO Type
1540    IF BIT(Coordt,10)=0 AND BIT(Coordt,9)=0 THEN LDIR 0
1550    IF BIT(Coordt,10)=0 AND BIT(Coordt,9)=1 THEN LDIR 90
1560    IF BIT(Coordt,10)=1 AND BIT(Coordt,9)=0 THEN LDIR 180
1570    IF BIT(Coordt,10)=1 AND BIT(Coordt,9)=1 THEN LDIR 270
1580    IF BIT(Coordt,12)=0 AND BIT(Coordt,11)=0 THEN CSIZE 3.4,.67!
```

```
1590    IF BIT(Coordt,12)=0 AND BIT(Coordt,11)=1 THEN CSIZE 5.1,.67
1600    IF BIT(Coordt,12)=1 AND BIT(Coordt,11)=0 THEN CSIZE 6.8,.67
1610    IF BIT(Coordt,12)=1 AND BIT(Coordt,11)=1 THEN CSIZE 8.5,.67
1620 Type:     ! Special characters
1630    IF A$=CHR$(95) THEN LABEL USING "#,A";CHR$(8)
1640    IF A$=CHR$(17) THEN GOTO Marker
1650    IF A$=CHR$(127) THEN GOTO Chr_tri
1660    IF A$=CHR$(24) THEN GOTO Chr_tri2
1670    IF A$=CHR$(22) THEN GOTO Chr_sqrt
1680    IF A$=CHR$(21) THEN GOTO Chr_rarw
1690    IF A$=CHR$(9) THEN GOTO Half_dn
1700    IF A$=CHR$(12) THEN GOTO Half_up
1710    IF A$=CHR$(1) THEN GOTO Chr_hp
1720    IMOVE 0,-20
1730    LABEL USING "#,A";A$
1740    IMOVE 0,20
1750    RETURN
1760 Marker:      !
1770    IMOVE -13,-28
1780    LABEL USING "#,A";CHR$(111)
1790    RETURN
1800 Half_up:     !
1810    IMOVE 0,18
1820    RETURN
1830 Half_dn:     !
1840    IMOVE 0,-18
1850    RETURN
1860 Chr_tri:!
1870    SYMBOL Triangle(*)
1880    LABEL USING "#,A";CHR$(32)
1890    RETURN
1900 Chr_tri2:     !
1910    SYMBOL Tri_2(*)
1920    LABEL USING "#,A";CHR$(32)
1930    RETURN
1940 Chr_sqrt:      !
1950    SYMBOL Sqr_rt(*)
1960    LABEL USING "#,A";CHR$(32)
1970    RETURN
1980 Chr_rarw:     !
1990    SYMBOL Rgt_arrow(*)
2000    LABEL USING "#,A";CHR$(32)
2010    RETURN
2020 Chr_hp:     !
2030    SYMBOL Hp_log(*)
2040    RETURN
2050  SUBEND
2060  !
2070  SUB Read_binary(@Io,Com$,INTEGER Data_len,INTEGER Buf(*))
2080  !
2090  !  This routine preforms a "generic" read binary
2100  !    from the HP3562A
2110  !
```

```
2120    OPTION BASE 1
2130    ASSIGN @Io;FORMAT ON
2140    OUTPUT @Io;Com$         ! Output command
2150    ENTER @Io USING "%,2A,W";A$,Data_len
2160    IF A$<>"#A" THEN        ! Check for correct response
2170      DISP "NOT CORRECT BLOCK MODE"
2180      CLEAR @Io
2190    ELSE
2200      IF Data_len=0 THEN SUBEXIT
2210      REDIM Buf(Data_len DIV 2) ! Set buffer to proper length
2220      ASSIGN @Io;FORMAT OFF
2230      ENTER @Io;Buf(*)      ! Read data into buffer
2240      ASSIGN @Io;FORMAT ON
2250    END IF
2260    LOCAL @Io
2270  SUBEND
```

```
20     !   APPENDIX C - DEMO PROGRAM 7
30     !///////////////////////////////////////////////////////////////
40     !
50     !  DEMO PROGRAM CONVERSION SUBPROGRAMS
60     !
70     ! (c)   COPYRIGHT 1985, Hewlett-Packard Co.
80     !       last updated  3-1-85
90     !       Basic 3.0
100    !
110    !  PURPOSE:
120    !       These subprograms are used to convert data
130    !       from the internal format of the HP3562A to
140    !       standard ANSI floating point representation
150    !       used by the HP Series 200 computer. The
160    !       internal real and long real data is stored
170    !       as integers (2 & 4 respectively) and passed
180    !       to the programs and converted.
190    !
200    !  SUBPROGRAMS:
210    !
220    !    Int_real              Converts two integers to the
230    !                          corresponding ANSI real value
240    !
250    !    L_int_real            Converts four integers to the
260    !                          corresponding ANSI real value
270    !
280    !///////////////////////////////////////////////////////////////
290    !
300    SUB Int_real(INTEGER I1,I2,REAL Real)
310     !
320     ! CONVERTS DATA IN HP3562A INTERNAL REAL FORMAT
330     ! TO BASIC REAL FORMAT
340     !
350      INTEGER I2_low
360      I2_low=BINAND(I2,255)     !EXTRACT EXPONENT
370      I2_hi=SHIFT(I2,8)*2^(-23)
380      Real=(I1/32768.+I2_hi)*2^I2_low
390    SUBEND
391     !
400    SUB L_int_real(INTEGER I1,I2,I3,I4,REAL Real)
410     !
420     ! CONVERTS DATA IN HP3562A INTERNAL LONG REAL FORMAT
430     ! TO BASIC REAL FORMAT
440     !
450      INTEGER I4_low    ! LEAST SIG BYTE
460      R1=I1
470      IF I2<=0 THEN
480        R2=(I2+32768.)*2.^(-15)
490      ELSE
500        R2=(I2)*2.^(-15)
510      END IF
520      IF I3<=0 THEN
530        R3=(I3+32768.)*2.^(-30)
540      ELSE                        !
```

```
550        R3=(I3)*2.^(-30)
560      END IF
570      I4_low=BINAND(I4,255)     !EXTRACT EXPONENT
580      R4=SHIFT(I4,8)*2^(-30-8)
590      Real=(R1+R2+R3+R4)*2^(I4_low-15)
600    SUBEND
610    !
```

APPENDIX C - EXAMPLE PROGRAM 8

```
10!RE-STORE"COMMAND"
20    REM THIS PROGRAM SENDS COMMANDS TO HP-IB
30    OPTION BASE 1
40    DIM A$[80],B$[100],Buf(2200)
50    INTEGER Ibuf(4400)
60    INTEGER Data_len
70    True=1
80    Report=16
90    Commnd=5
100   Xecute=14
110   A_commnd=8
120   A_data=0
130   A_dsj=16
140   A_format=12
150   A_read=10
160   A_write=9
170   Italk=128
180   Ilisten=136
190   Isdc=14
200   Iskey=13
210   Iischange=11
220   Daddr=1
230   Paddr=5
240   Haddr=20
250   ASSIGN @Io TO 700+Haddr
260   Rep_flg=True
270   End_flg=False
280   Iplot=0
290   Ignore=False
300   Idisk=1
310   Tct_flg=False
320   DISP ""
330   REMOTE 7
340   GOSUB Label1
350   DISP "HP-IB address = ";Haddr
360   GOTO Com_loop
370 Label1:  ON INTR 7,15 GOSUB Hpib_intr
380   ON KEY 0 LABEL "",14 GOSUB Nothing
390   ON KEY 1 LABEL "Command",14 GOSUB Com_intr
400   ON KEY 2 LABEL "Address",14 GOSUB Labeladdr
410   ON KEY 3 LABEL "Ser Poll",14 GOSUB Ser_poll
420   ON KEY 4 LABEL "Xfer",14 GOSUB Label2
430   ON KEY 5 LABEL "Clear",14 GOSUB Device_clear
440   ON KEY 6 LABEL "Read",14 GOSUB Read_intr
450   ON KEY 7 LABEL "Demos",14 GOSUB Labeldemo
460   ON KEY 8 LABEL "EXIT",15 GOSUB Exit_intr
470   ON KEY 9 LABEL "EXIT",15 GOSUB Exit_intr
480   ENABLE INTR 7;2
490   RETURN
500 Com_loop: IF NOT End_flg THEN Com_loop
510   WAIT .5 ! MAKE SURE COMMAND FINISHED
520   LOCAL 7
530   DISP "End of Commands"
540   STOP
```

```
550 Labeladdr: !
560   ON KEY 0 LABEL "",14 GOSUB Nothing
570   ON KEY 1 LABEL "HPIB Adr",14 GOSUB Hpib_addr
580   ON KEY 2 LABEL "Disk Adr",14 GOSUB Disk_addr
590   ON KEY 3 LABEL "TCT Flag",14 GOSUB Set_tct
600   ON KEY 4 LABEL "",14 GOSUB Nothing
610   ON KEY 5 LABEL "",14 GOSUB Nothing
620   ON KEY 6 LABEL "",14 GOSUB Nothing
630   ON KEY 7 LABEL "",14 GOSUB Nothing
640   ON KEY 8 LABEL "EXIT",14 GOSUB Label1
650   ON KEY 9 LABEL "EXIT",14 GOSUB Label1
660   RETURN
670 Labeldemo: !
680   ON KEY 0 LABEL "",14 GOSUB Nothing
690   ON KEY 1 LABEL "Help",14 GOSUB Help_list
700   ON KEY 2 LABEL "Help Plot",14 GOSUB Help_plot
710   ON KEY 3 LABEL "Ignore",14 GOSUB Ignore_intr
720   ON KEY 4 LABEL "Rpg Demo",14 GOSUB Rpg_demo
730   ON KEY 5 LABEL "Demo 1",14 GOSUB Demo
740   ON KEY 6 LABEL "Demo 2",14 GOSUB Demo2
750   ON KEY 7 LABEL "Control",14 GOSUB Canned
760   ON KEY 8 LABEL "EXIT Demo",15 GOSUB End_demo
770   ON KEY 9 LABEL "EXIT Demo",15 GOSUB End_demo
780   RETURN
790 End_demo: Demo_flg=False
800   GOSUB Label1
810   RETURN
820 Set_tct: Tct_flg=NOT Tct_flg
830   DISP "TCT flag = ";Tct_flg
840   RETURN
850 Help_list: OUTPUT @Io;"BEEP -85;ERRE;ERR?"
860   ENTER @Io;X
870   Err_flag=False
880   Demo_flg=True
890 Loop_send: OUTPUT @Io;"BEEP -86"
900   IF (Err_flag=False) AND (Demo_flg=True) THEN Loop_send
910   RETURN
920 Help_plot: OUTPUT @Io;"BEEP -85;ERRE;SRQE;ERR?"
930   ENTER @Io;X
940   OUTPUT @Io;"ROT 1"
950   Demo_flg=True
960   Err_flag=False
970 Loop_plot: OUTPUT @Io;"BEEP -86"
980   IF Err_flag OR (Demo_flg=False) THEN RETURN
990   End_plot=False
1000  OUTPUT @Io;"STPL"
1010 Wait_plot:IF End_plot=False THEN Wait_plot
1020  OUTPUT 700+Paddr;"EC;AH"
1030  GOTO Loop_plot
1040  RETURN
1050 Canned:B$="HP-IB control program"
1060  GOSUB Out_hpib
```

```
1070  B$="Change Setup State"
1080  GOSUB Out_hpib
1090  OUTPUT @Io;"SRQE;STAT"
1100  WAIT 2
1110  OUTPUT @Io;"SMES;TRGD;FREQ"
1120  OUTPUT @Io;"RNG"
1130  WAIT 2
1140  B$="Synthesize Data"
1150  GOSUB Out_hpib
1160  Disk_flg=False
1170  OUTPUT @Io;"CTRC"
1180  WAIT 5
1190  B$="Display LOG MAG data"
1200  GOSUB Out_hpib
1210  OUTPUT @Io;"CORD;MGLG"
1220  WAIT 5
1230  B$="Display REAL data"
1240  GOSUB Out_hpib
1250  OUTPUT @Io;"CORD;REAL"
1260  WAIT 5
1270  B$="Display IMAGINARY data"
1280  GOSUB Out_hpib
1290  OUTPUT @Io;"CORD;IMAG"
1300  WAIT 5
1310  B$="End of HP-IB test"
1320  GOSUB Out_hpib
1330  OUTPUT @Io;"STAT;COME"
1340  WAIT 1
1350  LOCAL @Io
1360  RETURN
1370 Out_hpib: OUTPUT @Io;"COMD;DBSZ 100,0"
1380  OUTPUT @Io;"DBAC 0;PU;PA 20,1000"
1390  OUTPUT @Io;"CHSZ 2"
1400  OUTPUT @Io;"WRIT '";B$;"'"
1410  OUTPUT @Io;"CHSZ 0;DBUP 0"
1420  WAIT 3
1430  RETURN
1440 Ser_poll:  Poll_byte=SPOLL(@Io)
1450   DISP "Ser Poll = ";Poll_byte
1460   RETURN
1470 Hpib_intr:   Poll_byte=SPOLL(@Io)
1480  PRINT "SRQ =",Poll_byte
1490  ! MASK OUT RQS, ERR, RDY
1500  IF BIT(Poll_byte,5) THEN Err_flag=True
1510  Poll_byte=BINAND(Poll_byte,143)
1520  IF Ignore THEN End_intr
1530  !
1540  ! PASS CONTROL
1550  !
1560  IF Tct_flg AND Poll_byte>=14 AND Poll_byte<=143 THEN
1570  DISP "PASS CONTROL"
1580  SEND 7;UNL UNT TALK Haddr CMD 9
1590  GOTO End_intr
```

```
1600  END IF
1610  !
1620  ! END OF DISK ACTION
1630  !
1640  IF Poll_byte=9 THEN
1650  Disk_flg=True
1660  GOTO End_intr
1670  END IF
1680  !
1690  ! END OF PLOT ACTION
1700  !
1710  IF Poll_byte=10 THEN
1720  End_plot=True
1730  GOTO End_intr
1740  END IF
1750  !
1760  ! TALK PLOTTER
1770  !
1780  IF Poll_byte=Italk+Iplot THEN
1790  SEND 7;UNL UNT LISTEN Haddr CMD 1 TALK Paddr DATA
1800  GOTO End_intr
1810  END IF
1820  !
1830  ! LISTEN PLOTTER
1840  !
1850  IF Poll_byte=Ilisten+Iplot THEN
1860  SEND 7;UNL UNT LISTEN Paddr TALK Haddr DATA
1870  GOTO End_intr
1880  END IF
1890  !
1900  ! LISTEN DISK COMMAND
1910  !
1920  IF Poll_byte=Ilisten+Idisk THEN
1930  SEND 7;UNL UNT LISTEN Daddr SEC Commnd TALK Haddr DATA
1940  Rep_flg=False
1950  GOTO End_intr
1960  END IF
1970  !
1980  ! LISTEN DISK EXECUTION
1990  !
2000  IF Poll_byte=Ilisten+Idisk+1 THEN
2010  GOSUB Parallel_poll
2020  SEND 7;UNL UNT LISTEN Daddr SEC Xecute TALK Haddr DATA
2030  Rep_flg=False
2040  GOTO End_intr
2050  END IF
2060  !
2070  ! LISTEN AMIGO COMMAND
2080  !
2090  IF Poll_byte=Ilisten+Idisk+2 THEN
2100  SEND 7;UNT UNL TALK Haddr LISTEN Daddr
2110  WAIT .001
```

```
2120  SEND 7;SEC A_commnd
2130  WAIT .001
2140  SEND 7;DATA
2150  GOTO End_intr
2160  END IF
2170  !
2180  ! LISTEN AMIGO DATA
2190  !
2200  IF Poll_byte=Ilisten+Idisk+3 THEN
2210  SEND 7;UNT UNL TALK Haddr LISTEN Daddr
2220  WAIT .001
2230  SEND 7;SEC A_data
2240  WAIT .001
2250  SEND 7;DATA
2260  GOTO End_intr
2270  END IF
2280  !
2290  ! LISTEN AMIGO READ COMMAND
2300  !
2310  IF Poll_byte=Ilisten+Idisk+4 THEN
2320  SEND 7;UNT UNL TALK Haddr LISTEN Daddr
2330  WAIT .001
2340  SEND 7;SEC A_read
2350  WAIT .001
2360  SEND 7;DATA
2370  GOTO End_intr
2380  END IF
2390  !
2400  ! LISTEN AMIGO WRITE COMMAND
2410  !
2420  IF Poll_byte=Ilisten+Idisk+5 THEN
2430  SEND 7;UNT UNL TALK Haddr LISTEN Daddr
2440  WAIT .001
2450  SEND 7;SEC A_write
2460  WAIT .001
2470  SEND 7;DATA
2480  GOTO End_intr
2490  END IF
2500  !
2510  ! LISTEN AMIGO FORMAT
2520  !
2530  IF Poll_byte=Ilisten+Idisk+6 THEN
2540  SEND 7;UNT UNL TALK Haddr LISTEN Daddr
2550  WAIT .001
2560  SEND 7;SEC A_format
2570  WAIT .001
2580  SEND 7;DATA
2590  GOTO End_intr
2600  END IF
2610  !
2620  ! TALK DISK EXECUTION
2630  !
2640  IF Poll_byte=Italk+Idisk THEN
```

```
2650 GOSUB Parallel_poll
2660 SEND 7;UNL UNT LISTEN Haddr CMD 1 TALK Daddr SEC Xecute DATA
2670 Rep_flg=False
2680 GOTO End_intr
2690 END IF
2700 !
2710 ! TALK DISK REPORT
2720 !
2730 IF Poll_byte=Italk+Idisk+1 THEN
2740 IF NOT Rep_flg THEN GOSUB Parallel_poll
2750 SEND 7;UNL UNT LISTEN Haddr CMD 1 TALK Daddr
2760 WAIT .001
2770 SEND 7;SEC Report DATA
2780 Rep_flg=True
2790 GOTO End_intr
2800 END IF
2810 !
2820 ! TALK AMIGO STATUS / COMMAND
2830 !
2840 IF Poll_byte=Italk+Idisk+2 THEN
2850 SEND 7;UNT UNL LISTEN Haddr CMD 1 TALK Daddr
2860 WAIT .001
2870 SEND 7;SEC A_commnd DATA
2880 GOTO End_intr
2890 END IF
2900 !
2910 ! TALK AMIGO DATA
2920 !
2930 IF Poll_byte=Italk+Idisk+3 THEN
2940 SEND 7;UNT UNL LISTEN Haddr CMD 1 TALK Daddr
2950 WAIT .001
2960 SEND 7;SEC A_data DATA
2970 GOTO End_intr
2980 END IF
2990 !
3000 ! TALK AMIGO DSJ
3010 !
3020 IF Poll_byte=Italk+Idisk+4 THEN
3030 SEND 7;UNT UNL LISTEN Haddr CMD 1 TALK Daddr
3040 WAIT .001
3050 SEND 7;SEC A_dsj DATA
3060 GOTO End_intr
3070 END IF
3080 !
3090 ! TALK DISK IDENT
3100 !
3110 IF Poll_byte=Italk+Idisk+5 THEN
3120 SEND 7;UNT UNL LISTEN Haddr CMD 1 UNT
3130 WAIT .001
3140 SEND 7;SEC Daddr DATA
3150 GOTO End_intr
3160 END IF
3170 !
```

```
3180  ! TALK AMIGO PARALLEL POLL
3190  !
3200  IF Poll_byte=Italk+Idisk+6 THEN
3210  SEND 7;UNT UNL DATA
3220  GOSUB Parallel_poll
3230  GOTO End_intr
3240  END IF
3250  !
3260  ! CLEAR PLOTTER
3270  !
3280  IF Poll_byte=Isdc THEN
3290  CLEAR 700+Paddr
3300  SEND 7;UNT UNL LISTEN Haddr CMD 1 DATA
3310  GOTO End_intr
3320  END IF
3330  !
3340  ! UNADDRESS BUS
3350  !
3360  IF Poll_byte=Isdc+1 THEN
3370  SEND 7;UNT UNL LISTEN Haddr CMD 1 UNL DATA
3380  GOTO End_intr
3390  END IF
3400  !
3410  ! REDIRECTED KEY HIT
3420  !
3430  IF Poll_byte=Iskey THEN
3440  OUTPUT @Io;"COM?"
3450  ENTER @Io;Keycode,Keystr$
3460  DISP "KEY CODE = ",Keycode,"STR->";Keystr$;"<-"
3470  IF Keycode>0 THEN OUTPUT @Io;"KEY ";Keycode
3480  GOTO End_intr
3490  END IF
3500  !
3510  ! INSTRUMENT STATUS CHANGE
3520  !
3530  IF Poll_byte=11 THEN
3540  OUTPUT @Io;"IS?"
3550  ENTER @Io;Stat_word
3560  PRINT "IS = ",Stat_word
3570  IF BINAND(Stat_word,4096) THEN Entry_changed=True
3580  IF BINAND(Stat_word,8192) THEN
3590    OUTPUT @Io;"AS?"
3600    ENTER @Io;As_word
3610    PRINT "AS = ",As_word
3620  END IF
3630  LOCAL @Io
3640  END IF
3650  !
3660  ! UNKNOWN SRQ
3670  !
3680 End_intr:   ENABLE INTR 7
3690  RETURN
```

```
3700  !
3710  !
3720 Parallel_poll:   Ppoll_byte=PPOLL(7)
3730  IF BIT(Ppoll_byte,7-Daddr)=0 THEN Parallel_poll
3740  RETURN
3750  !
3760 Exit_intr:End_flg=True
3770  RETURN
3780  !
3790  !
3800 Com_intr:DISP "Enter Command"
3810  ENTER 2;A$
3820  IF A$<>"" THEN
3830  OUTPUT @Io;A$
3840  LOCAL @Io
3850  END IF
3860  DISP
3870  RETURN
3880 Demo: I=0
3890  Demo_flg=True
3900  P1x=0
3910  P1y=250
3920  P1dir=1
3930  P2x=1750
3940  P2y=2000
3950  P2dir=3
3960  OUTPUT @Io;"COMD;DBSZ 200,0,2"
3970 Demoloop: OUTPUT @Io;"DBAC ";I
3980  OUTPUT @Io;"PU;PA ";P1x,P1y
3990  OUTPUT @Io;"PD;PA ";P2x,P2y
4000  OUTPUT @Io;"LT 0"
4010  OUTPUT @Io;"DBSW ";I,1-I
4020  I=1-I
4030  CALL Nextpoint(P1x,P1y,P1dir)
4040  CALL Nextpoint(P2x,P2y,P2dir)
4050  IF Demo_flg THEN Demoloop
4060  RETURN
4070  !
4080 Demo2: P1x=0
4090   P1y=250
4100   P2x=1750
4110   P2y=2000
4120   Demo_flg=True
4130   Incr=50
4140   OUTPUT @Io;"COMD;DBSZ 300,0;DBAC 0;PU"
4150   OUTPUT @Io;"PA ";P1x,P1y
4160   OUTPUT @Io;"PD;DBUP 0"
4170 Demo2_loop: IF End_flg THEN Demo2_end
4180   OUTPUT @Io;"DBAA 0;PA ";P2x,P1y
4190   OUTPUT @Io;"DBUP 0"
4200   P2y=P2y-Incr
4210   IF P2y<P1y THEN Demo2_end
```

```
4220   OUTPUT @Io;"DBAA 0;PA ";P2x,P2y
4230   OUTPUT @Io;"DBUP 0"
4240   P1x=P1x+Incr
4250   IF P1x>P2x THEN Demo2_end
4260   OUTPUT @Io;"DBAA 0;PA ";P1x,P2y
4270   OUTPUT @Io;"DBUP 0"
4280   P1y=P1y+Incr
4290   IF P1y>P2y THEN Demo2_end
4300   OUTPUT @Io;"DBAA 0;PA ";P1x,P1y
4310   OUTPUT @Io;"DBUP 0"
4320   P2x=P2x-Incr
4330   IF P2x<P1x THEN Demo2_end
4340   GOTO Demo2_loop
4350 Demo2_end: IF Demo_flg THEN Demo2
4360   RETURN
4370   !
4380 Rpg_demo: OUTPUT @Io;"COMD;ISM 4096;RENE"
4390   Demo_flg=True
4400   Entry_changed=True
4410 Rpg_loop: IF Entry_changed THEN
4420   Entry_changed=False
4430   OUTPUT @Io;"RENV?"
4440   ENTER @Io;Entry_value
4450   DISP "Entry = ";Entry_value
4460   END IF
4470   IF Demo_flg THEN Rpg_loop
4480   OUTPUT @Io;"REND;ISM 0;COME"
4490   RETURN
4500 Ignore_intr:Ignore=NOT Ignore
4510  DISP "Ignore = ",Ignore
4520  GOTO End_intr
4530 Device_clear:CLEAR 700+Haddr
4540  DISP "Device clear sent"
4550  GOTO End_intr
4560 Read_intr:ENTER 700+Haddr;A$
4570  DISP "String ->";A$;"<-"
4580  RETURN
4590 Hpib_addr:DISP USING "3(K)";"Enter HP-IB address (",Haddr,")"
4600  ENTER 2;Haddr
4610  ASSIGN @Io TO 700+Haddr
4620  DISP
4630  RETURN
4640 Disk_addr:DISP USING "3(K)";"Enter Disk address (",Daddr,")"
4650  ENTER 2;Daddr
4660  DISP
4670  RETURN
4680  !
4690 Label2: !
4700  ON KEY 0 LABEL "",14 GOSUB Nothing
4710  ON KEY 1 LABEL "Display",14 GOSUB Labeldsp
4720  ON KEY 2 LABEL "Synth",14 GOSUB Labelt
4730  ON KEY 3 LABEL "Memory",14 GOSUB Labelm
4740  ON KEY 4 LABEL "Setup",14 GOSUB Labels
```

```
4750  ON KEY 5 LABEL "Data",14 GOSUB Labeld
4760  ON KEY 6 LABEL "Coord",14 GOSUB Labelc
4770  ON KEY 7 LABEL "Vector",14 GOSUB Labelv
4780  ON KEY 8 LABEL "EXIT",14 GOSUB Label1
4790  ON KEY 9 LABEL "EXIT",14 GOSUB Label1
4800  !
4810 Nothing: RETURN
4820  !
4830 Labels: !
4840  ON KEY 0 LABEL "",14 GOSUB Nothing
4850  ON KEY 1 LABEL "LSAS",14 GOSUB Lsas
4860  ON KEY 2 LABEL "LSBN",14 GOSUB Lsbn
4870  ON KEY 3 LABEL "LSAN",14 GOSUB Lsan
4880  ON KEY 4 LABEL "EXIT",14 GOSUB Label2
4890  ON KEY 5 LABEL "",14 GOSUB Nothing
4900  ON KEY 6 LABEL "DSAS",14 GOSUB Dsas
4910  ON KEY 7 LABEL "DSBN",14 GOSUB Dsbn
4920  ON KEY 8 LABEL "DSAN",14 GOSUB Dsan
4930  ON KEY 9 LABEL "EXIT",14 GOSUB Label2
4940  RETURN
4950  !
4960 Labeldsp: !
4970  ON KEY 0 LABEL "",14 GOSUB Nothing
4980  ON KEY 1 LABEL "",14 GOSUB Nothing
4990  ON KEY 2 LABEL "",14 GOSUB Nothing
5000  ON KEY 3 LABEL "",14 GOSUB Nothing
5010  ON KEY 4 LABEL "",14 GOSUB Nothing
5020  ON KEY 5 LABEL "Ascii",14 GOSUB Disp_asc
5030  ON KEY 6 LABEL "Binary",14 GOSUB Disp_bin
5040  ON KEY 7 LABEL "Ansi",14 GOSUB Disp_ans
5050  ON KEY 8 LABEL "EXIT",14 GOSUB Label2
5060  ON KEY 9 LABEL "EXIT" GOSUB Label2
5070  RETURN
5080  !
5090 Labeld: !
5100  ON KEY 0 LABEL "",14 GOSUB Nothing
5110  ON KEY 1 LABEL "LDAS",14 GOSUB Ldas
5120  ON KEY 2 LABEL "LDBN",14 GOSUB Ldbn
5130  ON KEY 3 LABEL "LDAN",14 GOSUB Ldan
5140  ON KEY 4 LABEL "EXIT",14 GOSUB Label2
5150  ON KEY 5 LABEL "",14 GOSUB Nothing
5160  ON KEY 6 LABEL "DDAS",14 GOSUB Ddas
5170  ON KEY 7 LABEL "DDBN",14 GOSUB Ddbn
5180  ON KEY 8 LABEL "DDAN",14 GOSUB Ddan
5190  ON KEY 9 LABEL "EXIT" GOSUB Label2
5200  RETURN
5210  !
5220 Labelt: !
5230  ON KEY 0 LABEL "",14 GOSUB Nothing
5240  ON KEY 1 LABEL "LTAS",14 GOSUB Ltas
5250  ON KEY 2 LABEL "LTBN",14 GOSUB Ltbn
5260  ON KEY 3 LABEL "LTAN",14 GOSUB Ltan
5270  ON KEY 4 LABEL "EXIT",14 GOSUB Label2
5280  ON KEY 5 LABEL "",14 GOSUB Nothing
```

```
5290  ON KEY 6 LABEL "DTAS",14 GOSUB Dtas
5300  ON KEY 7 LABEL "DTBN",14 GOSUB Dtbn
5310  ON KEY 8 LABEL "DTAN",14 GOSUB Dtan
5320  ON KEY 9 LABEL "EXIT" GOSUB Label2
5330  RETURN
5340  !
5350 Labelc: !
5360  ON KEY 0 LABEL "",14 GOSUB Nothing
5370  ON KEY 1 LABEL "",14 GOSUB Nothing
5380  ON KEY 2 LABEL "",14 GOSUB Nothing
5390  ON KEY 3 LABEL "",14 GOSUB Nothing
5400  ON KEY 4 LABEL "",14 GOSUB Nothing
5410  ON KEY 5 LABEL "DCAS",14 GOSUB Dcas
5420  ON KEY 6 LABEL "DCBN",14 GOSUB Dcbn
5430  ON KEY 7 LABEL "DCAN",14 GOSUB Dcan
5440  ON KEY 8 LABEL "EXIT",14 GOSUB Label2
5450  ON KEY 9 LABEL "EXIT",14 GOSUB Label2
5460  RETURN
5470  !
5480 Labelv: !
5490  ON KEY 0 LABEL "",14 GOSUB Nothing
5500  ON KEY 1 LABEL "",14 GOSUB Nothing
5510  ON KEY 2 LABEL "",14 GOSUB Nothing
5520  ON KEY 3 LABEL "",14 GOSUB Nothing
5530  ON KEY 4 LABEL "",14 GOSUB Nothing
5540  ON KEY 5 LABEL "DVAS",14 GOSUB Dvas
5550  ON KEY 6 LABEL "DVBN",14 GOSUB Dvbn
5560  ON KEY 7 LABEL "DVAN",14 GOSUB Dvan
5570  ON KEY 8 LABEL "EXIT",14 GOSUB Label2
5580  ON KEY 9 LABEL "EXIT",14 GOSUB Label2
5590  RETURN
5600  !
5610 Labelm: !
5620  ON KEY 0 LABEL "",14 GOSUB Nothing
5630  ON KEY 1 LABEL "DISP Memory",14 GOSUB Display_mem
5640  ON KEY 2 LABEL "Load FltAnsi",14 GOSUB Load_flt
5650  ON KEY 3 LABEL "Dump FltAnsi",14 GOSUB Dump_flt
5660  ON KEY 4 LABEL "Mem Addr",14 GOSUB Mem_addr
5670  ON KEY 5 LABEL "Dump Ascii",14 GOSUB Dump_asc
5680  ON KEY 6 LABEL "Dump Binary",14 GOSUB Dump_bin
5690  ON KEY 7 LABEL "Dump ANSI",14 GOSUB Dump_ans
5700  ON KEY 8 LABEL "EXIT",14 GOSUB Label2
5710  ON KEY 9 LABEL "EXIT",14 GOSUB Label2
5720  RETURN
5730  !
5740 Display_mem: CALL Disp_mem(@Io,Ibuf(*))
5750  RETURN
5760 Lsas: CALL Write_ascii(@Io,"LSAS",Data_len,Buf(*))
5770  RETURN
5780 Dsas: CALL Read_ascii(@Io,"DSAS",Data_len,Buf(*))
5790  RETURN
5800 Lsbn: CALL Write_binary(@Io,"LSBN",Data_len,Ibuf(*))
5810  RETURN
```

```
5820 Dsbn: CALL Read_binary(@Io,"DSBN",Data_len,Ibuf(*))
5830  RETURN
5840 Lsan: CALL Write_float(@Io,"LSAN",Data_len,Buf(*))
5850  RETURN
5860 Dsan: CALL Read_float(@Io,"DSAN",Data_len,Buf(*))
5870  RETURN
5880  !
5890 Ldas: CALL Write_ascii(@Io,"LDAS",Data_len,Buf(*))
5900  RETURN
5910 Ddas: CALL Read_ascii(@Io,"DDAS",Data_len,Buf(*))
5920  RETURN
5930 Ldbn: CALL Write_binary(@Io,"LDBN",Data_len,Ibuf(*))
5940  RETURN
5950 Ddbn: CALL Read_binary(@Io,"DDBN",Data_len,Ibuf(*))
5960  RETURN
5970 Ldan: CALL Write_float(@Io,"LDAN",Data_len,Buf(*))
5980  RETURN
5990 Ddan: CALL Read_float(@Io,"DDAN",Data_len,Buf(*))
6000  RETURN
6010  !
6020 Ltas: CALL Write_ascii(@Io,"LTAS",Data_len,Buf(*))
6030  RETURN
6040 Dtas: CALL Read_ascii(@Io,"DTAS",Data_len,Buf(*))
6050  RETURN
6060 Ltbn: CALL Write_binary(@Io,"LTBN",Data_len,Ibuf(*))
6070  RETURN
6080 Dtbn: CALL Read_binary(@Io,"DTBN",Data_len,Ibuf(*))
6090  RETURN
6100 Ltan: CALL Write_float(@Io,"LTAN",Data_len,Buf(*))
6110  RETURN
6120 Dtan: CALL Read_float(@Io,"DTAN",Data_len,Buf(*))
6130  RETURN
6140  !
6150 Dcas: CALL Read_ascii(@Io,"DCAS",Data_len,Buf(*))
6160  RETURN
6170 Dcbn: CALL Read_binary(@Io,"DCBN",Data_len,Ibuf(*))
6180  RETURN
6190 Dcan: CALL Read_float(@Io,"DCAN",Data_len,Buf(*))
6200  RETURN
6210  !
6220 Dvas: CALL Read_ascii(@Io,"DVAS",Data_len,Buf(*))
6230  RETURN
6240 Dvbn: CALL Read_binary(@Io,"DVBN",Data_len,Ibuf(*))
6250  RETURN
6260 Dvan: CALL Read_float(@Io,"DVAN",Data_len,Buf(*))
6270  RETURN
6280  !
6290 Dump_asc: CALL Read_ascii(@Io,"DMAS",Data_len,Buf(*))
6300  RETURN
6310 Dump_bin: CALL Read_binary(@Io,"DMBN",Data_len,Ibuf(*))
6320  RETURN
6330 Load_bin: CALL Write_binary(@Io,"LMBN",Data_len,Ibuf(*))
6340  RETURN
```

```
6350 Dump_ans: CALL Read_float(@Io,"DMAN",Data_len,Buf(*))
6360  RETURN
6370 Dump_flt: CALL Read_float(@Io,"DFAN",Data_len,Buf(*))
6380  RETURN
6390 Load_flt: CALL Write_float(@Io,"LFAN",Data_len,Buf(*))
6400  RETURN
6410 Mem_addr: DISP "Enter High word, Low word, Mem size"
6420  ENTER 2;Ihigh,Ilow,Isize
6430  OUTPUT @Io;"MEMA ";Ihigh;",";Ilow;";MEMS ";Isize
6440  LOCAL @Io
6450  RETURN
6460  !
6470 Disp_asc: !
6480 Disp_ans: DISP "Enter Start, Count"
6490  ENTER 2;Istart,Icount
6500  FOR I=Istart TO Istart+Icount-1
6510    PRINT I;" = ";Buf(I)
6520  NEXT I
6530  RETURN
6540  !
6550 Disp_bin: DISP "Enter Start, Count"
6560  ENTER 2;Istart,Icount
6570  FOR I=Istart TO Istart+Icount-1
6580    PRINT I;" = ";Ibuf(I)
6590  NEXT I
6600  RETURN
6610  !
6620  END
6630  !
6640  !
6650  SUB Nextpoint(X,Y,Dir)
6660  Incr=100
6670  ON Dir GOSUB Dright,Dup,Dleft,Ddown
6680  SUBEXIT
6690 Dright:X=X+Incr
6700  Y=250
6710  IF X+Incr>1750 THEN Dir=2
6720  RETURN
6730 Dup: Y=Y+Incr
6740  X=1750
6750  IF Y+Incr>2000 THEN Dir=3
6760  RETURN
6770 Dleft: X=X-Incr
6780  Y=2000
6790  IF X-Incr<0 THEN Dir=4
6800  RETURN
6810 Ddown:Y=Y-Incr
6820  X=0
6830  IF Y-Incr<250 THEN Dir=1
6840  RETURN
6850  SUBEND
6860  !
6870  !
```

```
6880  SUB Read_ascii(@Io,Com$,INTEGER Data_len,REAL Buf(*))
6890  OPTION BASE 1
6900  ASSIGN @Io;FORMAT ON
6910  DISP "Start Read Ascii"
6920  OUTPUT @Io;Com$
6930  ENTER @Io USING "2A,K";A$,Float_len
6940  Data_len=INT(Float_len+.5)
6950  REDIM Buf(Data_len)
6960  IF A$<>"#I" THEN
6970    DISP "NOT CORRECT BLOCK MODE"
6980    CLEAR @Io
6990  ELSE
7000    FOR I=1 TO Data_len
7010      ENTER @Io;Buf(I)
7020    NEXT I
7030    DISP "End Read, Data length = ";Data_len
7040  END IF
7050  LOCAL @Io
7060  SUBEND
7070  !
7080  !
7090  SUB Write_ascii(@Io,Com$,INTEGER Data_len,REAL Buf(*))
7100  OPTION BASE 1
7110  ASSIGN @Io;FORMAT ON
7120  DISP "Start Write Ascii, Data_len = ";Data_len
7130  OUTPUT @Io;Com$
7140  OUTPUT @Io;"#I";Data_len
7150  FOR I=1 TO Data_len-1
7160    OUTPUT @Io;Buf(I)
7170  NEXT I
7180  OUTPUT @Io;Buf(I),END
7190  DISP "End Write"
7200  LOCAL @Io
7210  SUBEND
7220  !
7230  !
7240  SUB Read_binary(@Io,Com$,INTEGER Data_len,INTEGER Buf(*))
7250  OPTION BASE 1
7260  ASSIGN @Io;FORMAT ON
7270  DISP "Read Binary"
7280  OUTPUT @Io;Com$
7290  ENTER @Io USING "%,2A,W";A$,Data_len
7300  IF A$<>"#A" THEN
7310    DISP "NOT CORRECT BLOCK MODE"
7320    CLEAR @Io
7330  ELSE
7340    REDIM Buf(Data_len DIV 2)
7350    ASSIGN @Io;FORMAT OFF
7360    ENTER @Io;Buf(*)
7370    ASSIGN @Io;FORMAT ON
7380    DISP "End read, Data_len = ";Data_len
```

```
7390  END IF
7400  LOCAL @Io
7410  SUBEND
7420  !
7430  !
7440  SUB Write_binary(@Io,Com$,INTEGER Data_len,INTEGER Buf(*))
7450  OPTION BASE 1
7460  ASSIGN @Io;FORMAT ON
7470  DISP "Start Write Binary, Data_len = ";Data_len
7480  OUTPUT @Io;Com$
7490  REDIM Buf(Data_len DIV 2)
7500  OUTPUT @Io USING "#,2A,W";"#A";Data_len
7510  ASSIGN @Io;FORMAT OFF
7520  OUTPUT @Io;Buf(*)
7530  ASSIGN @Io;FORMAT ON
7540  DISP "End Write"
7550  LOCAL @Io
7560  SUBEND
7570  !
7580  !
7590  SUB Read_float(@Io,Com$,INTEGER Data_len,REAL Buf(*))
7600  OPTION BASE 1
7610  ASSIGN @Io;FORMAT ON
7620  DISP "Start Read float"
7630  OUTPUT @Io;Com$
7640  ENTER @Io USING "%,2A,W";A$,Data_len
7650  IF (A$<>"#A") OR (Data_len MOD 8<>0) THEN
7660    DISP "NOT CORRECT BLOCK MODE"
7670    CLEAR @Io
7680  ELSE
7690    ASSIGN @Io;FORMAT OFF
7700    REDIM Buf(Data_len DIV 8)
7710    ENTER @Io;Buf(*)
7720    DISP "End Read, Data_len = ";Data_len
7730    ASSIGN @Io;FORMAT ON
7740  END IF
7750  LOCAL @Io
7760  SUBEND
7770  !
7780  !
7790  SUB Write_float(@Io,Com$,INTEGER Data_len,REAL Buf(*))
7800  OPTION BASE 1
7810  ASSIGN @Io;FORMAT ON
7820  DISP "Start Write Float, Data_len = ",Data_len
7830  OUTPUT @Io;Com$
7840  OUTPUT @Io USING "#,2A,W";"#A",Data_len
7850  ASSIGN @Io;FORMAT OFF
7860  REDIM Buf(Data_len DIV 8)
7870  OUTPUT @Io;Buf(*)
7880  ASSIGN @Io;FORMAT ON
7890  DISP "End Write"
7900  LOCAL @Io
7910  SUBEND
7920  !
7930  !
7940 Disp_mem:  SUB Disp_mem(@Io,INTEGER Buf(*))
7950  OPTION BASE 1
7960  INTEGER Byte_len
7970  DIM R$[10]
```

```
7980  DISP "Memory Address"
7990  ENTER KBD;R$
8000  Mem_addr=DVAL(R$,16)
8010  Hi_mem=Mem_addr/65536
8020  Lo_mem=Mem_addr MOD 65536
8030  DISP "Byte Length"
8040  ENTER KBD;Byte_len
8050  IF BIT(Byte_len,0) THEN Byte_len=Byte_len+1
8060  OUTPUT @Io;"MEMA";Hi_mem,Lo_mem
8070  OUTPUT @Io;"MEMS";Byte_len DIV 2
8080  CALL Read_binary(@Io,"DMBN",Byte_len,Buf(*))
8090  CALL Disp_buf(Buf(*),Mem_addr,Byte_len)
8100  SUBEND
8110  !
8120  !
8130 Disp_buf: SUB Disp_buf(INTEGER Buf(*),REAL Mem_addr,INTEGER Byte_len)
8140  OPTION BASE 1
8150  DIM A$[16]
8160  INTEGER Word_len
8170  Word_len=Byte_len DIV 2
8180  New_addr=Mem_addr
8190  Byte_cnt=0
8200  I=1
8210  WHILE I<=Word_len
8220    IF Byte_cnt=0 THEN
8230      PRINT DVAL$(New_addr,16);": ";
8240      New_addr=New_addr+16
8250      A$=""
8260    END IF
8270    Byte_val=Buf(I) DIV 256
8280    IF Byte_val<0 THEN Byte_val=Byte_val+256
8290    B$=IVAL$(Byte_val,16)
8300    PRINT B$[3,4];" ";
8310    IF (CHR$(Byte_val)<" ") OR (CHR$(Byte_val)>"~") THEN
8320      A$=A$&"."
8330    ELSE
8340      A$=A$&CHR$(Byte_val)
8350    END IF
8360    Byte_val=Buf(I) MOD 256
8370    B$=IVAL$(Byte_val,16)
8380    PRINT B$[3,4];" ";
8390    IF (CHR$(Byte_val)<" ") OR (CHR$(Byte_val)>"~") THEN
8400      A$=A$&"."
8410    ELSE
8420      A$=A$&CHR$(Byte_val)
8430    END IF
8440    I=I+1
8450    Byte_cnt=Byte_cnt+2
8460    IF Byte_cnt>=16 THEN
8470      PRINT "|";A$;"|"
8480      Byte_cnt=0
8490    END IF
8500  END WHILE
8510  IF Byte_cnt>0 THEN PRINT
8520  SUBEND
8530  !
8540  !
```

PROGRAMMING INDEX

NOTE: For quick information on the command for any key or softkey, please refer to the Quick Reference Guide in Appendix B

PROGRAMMING INDEX

PROGRAMMING INDEX

NOTE: For quick information on the command for any key or softkey, please refer to the Quick Reference Guide in Appendix B

PROGRAMMING INDEX

NOTE: For quick information on the command for any key or softkey, please refer to the Quick Reference Guide in Appendix B

Product Line Sales/Support Key

Key Product Line
A Analytical
CM Components
C Computer Systems Sales only
CH Computer Systems Hardware Sales and Services
CS Computer Systems Software Sales and Services
E Electronic Instruments & Measurement Systems
M Medical Products
MP Medical Products Primary SRO
MS Medical Products Secondary SRO
P Personal Computation Products
*** Sales only for specific product line**
**** Support only for specific product line**

IMPORTANT: These symbols designate general product line capability. They do not insure sales or support availability for all products within a line, at all locations. Contact your local sales office for information regarding locations where HP support is available for specific products.

HP distributors are printed in italics.

HEADQUARTERS OFFICES

If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA
Hewlett-Packard S.A.
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hpse
Cable: HEWPACKSA Geneve

ASIA
Hewlett-Packard Asia Ltd.
6th Floor, Sun Hung Kai Centre
30 Harbour Rd.
G.P.O. Box 795
HONG KONG
Tel: 5-832 3211
After Jan. 1, 1984
47th Floor, China Resources Bldg.
26 Harbour Rd., Wanchai
HONG KONG
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG

CANADA
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE
Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA**, Austria
Tel: (222) 2365110
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE
Hewlett-Packard S.A.
Uilenstede 475
P.O.Box 999
NL-1180 AZ **AMSTELVEEN**
The Netherlands
Tel: 20 437771

SOUTH EAST EUROPE
Hewlett-Packard S.A.
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hpse
Cable: HEWPACKSA Geneve

OTHER EUROPE
Hewlett-Packard S.A.
P.O. Box
150, Rte du Nant-D'Avril
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 8111
Telex: 22486 hpsa
Cable: HEWPACKSA Geneve

MEDITERRANEAN AND MIDDLE EAST
Hewlett-Packard S.A.
Mediterranean and Middle East Operations
Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, **ATHENS**
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

EASTERN USA
Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA
Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA
Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500

WESTERN USA
Hewlett-Packard Co.
3939 Lankershim Blvd.
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

OTHER INTERNATIONAL AREAS
Hewlett-Packard Co.
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ANGOLA
Telectra
Empresa Técnica de Equipamentos
R. Barbosa Rodrigues, 41-I DT.
Caixa Postal 6487
LUANDA
Tel: 35515,35516
E,P

ARGENTINA
Hewlett-Packard Argentina S.A.
Avenida Santa Fe 2035
Martinez 1640 **BUENOS AIRES**
Tel: 798-5735, 792-1293
Telex: 17595 BIONAR
Cable: HEWPACKARG
A,E,CH,CS,P

Biotron S.A.C.I.M. e I.
Av Paseo Colon 221, Piso 9
1399 ***BUENOS AIRES***
Tel: 30-4846, 30-1851
Telex: 17595 BIONAR
M

AUSTRALIA

Adelaide, South Australia Office
Hewlett-Packard Australia Ltd.
153 Greenhill Road
PARKSIDE, S.A. 5063
Tel: 272-5911
Telex: 82536
Cable: HEWPARD Adelaide
A*,CH,CM,,E,MS,P

Brisbane, Queensland Office
Hewlett-Packard Australia Ltd.
10 Payne Road
THE GAP, Queensland 4061
Tel: 30-4133
Telex: 42133
Cable: HEWPARD Brisbane
A,CH,CM,E,M,P

Canberra, Australia Capital Territory Office
Hewlett-Packard Australia Ltd.
121 Wollongong Street
FYSHWICK, A.C.T. 2609
Tel: 80 4244
Telex: 62650
Cable: HEWPARD Canberra
CH,CM,E,P

Melbourne, Victoria Office
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne
A,CH,CM,CS,E,MS,P

Perth, Western Australia Office
Hewlett-Packard Australia Ltd.
261 Stirling Highway
CLAREMONT, W.A. 6010
Tel: 383-2188
Telex: 93859
Cable: HEWPARD Perth
A,CH,CM,E,MS,P

Sydney, New South Wales Office
Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 887-1611
Telex: 21561
Cable: HEWPARD Sydney
A,CH,CM,CS,E,MS,P

AUSTRIA
Hewlett-Packard Ges.m.b.h.
Grottenhofstrasse 94
A-8052 **GRAZ**
Tel: (0316) 291 5 66
Telex: 32375
CH,E

Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 23 65 11-0
Telex: 134425 HEPA A
A,CH,CM,CS,E,MS,P

BAHRAIN
Green Salon
P.O. Box 557
Manama
BAHRAIN
Tel: 255503-255950
Telex: 84419
P

Wael Pharmacy
P.O. Box 648
BAHRAIN
Tel: 256123
Telex: 8550 WAEL BN
E,C,M

BELGIUM
Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,MP,P

BRAZIL
Hewlett-Packard do Brasil I.e.C. Ltda.
Alameda Rio Negro, 750
Alphaville
06400 **BARUERI SP**
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo
A,CH,CM,CS,E,M,P

Hewlett-Packard do Brasil I.e.C. Ltda.
Avenida Epitacio Pessoa, 4664
22471 **RIO DE JANEIRO-RJ**
Tel: (02l) 286.0237
Telex: 021-21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro
A,CH,CM,E,MS,P*

ANAMED I.C.E.I. Ltda.
Rua Bage, 103
04012 ***SAO PAULO***
Tel: (011) 570-5726
Telex: 021-21905 HPBR-BR
M

SALES & SUPPORT OFFICES

Arranged alphabetically by country

CANADA

Alberta

Hewlett-Packard (Canada) Ltd.
3030 3rd Avenue N.E.
CALGARY, Alberta T2A 6T7
Tel: (403) 235-3100
A,CH,CM,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
11120A-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666
A,CH,CM,CS,E,MS,P

British Columbia

Hewlett-Packard (Canada) Ltd.
10691 Shellbridge Way
RICHMOND,
British Columbia V6X 2W7
Tel: (604) 270-2277
Telex: 610-922-5059
A,CH,CM,CS,E*,MS,P*

Manitoba

Hewlett-Packard (Canada) Ltd.
380-550 Century Street
WINNIPEG, Manitoba R3H 0Y1
Tel: (204) 786-6701
A,CH,CM,E,MS,P*

Nova Scotia

Hewlett-Packard (Canada) Ltd.
P.O. Box 931
900 Windmill Road
DARTMOUTH, Nova Scotia B2Y 3Z6
Tel: (902) 469-7820
CH,CM,CS,E*,MS,P*

Ontario

Hewlett-Packard (Canada) Ltd.
3325 N. Service Rd., Unit 6
BURLINGTON, Ontario P3A 2A3
Tel: (416) 335-8644
CS,M*

Hewlett-Packard (Canada) Ltd.
552 Newbold Street
LONDON, Ontario N6E 2S5
Tel: (519) 686-9181
A,CH,CM,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
A,CH,CM,CS,E,MP,P

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483
A,CH,CM,CS,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
220 Yorkland Blvd., Unit #11
WILLOWDALE, Ontario M2J 1R5
Tel: (416) 499-9333
CH

Quebec

Hewlett-Packard (Canada) Ltd.
17500 South Service Road
Trans-Canada Highway
KIRKLAND, Quebec H9J 2M5
Tel: (514) 697-4232
A,CH,CM,CS,E,MP,P*

Hewlett-Packard (Canada) Ltd.
Les Galeries du Vallon
2323 Du Versont Nord
STE. FOY, Quebec G1N 4C2
Tel: (418) 687-4570
CH

CHILE

Jorge Calcagni y Cia. Ltda.
Av. Italia 634 Santiago
Casilla 16475
***SANTIAGO** 9*
Tel: 222-0222
Telex: Public Booth 440001
A,CM,E,M

Olympia (Chile) Ltda.
Av. Rodrigo de Araya 1045
Casilla 256-V
***SANTIAGO** 21*
Tel: (02) 22 55 044
Telex: 240-565 OLYMP CL
Cable: Olympiachile Santiagochile
CH,CS,P

CHINA, People's Republic of

China Hewlett-Packard Rep. Office
P.O. Box 418
1A Lane 2, Luchang St.
Beiwei Rd., Xuanwu District
BEIJING
Tel: 33-1947, 33-7426
Telex: 22601 CTSHP CN
Cable: 1920
A,CH,CM,CS,E,P

COLOMBIA

Instrumentación
H. A. Langebaek & Kier S.A.
Carrera 4A No. 52A-26
Apartado Aereo 6287
***BOGOTA** 1, D.E.*
Tel: 212-1466
Telex: 44400 INST CO
Cable: AARIS Bogota
CM,E,M

Casa Humboldt Ltda.
Carrera 14, No. 98-60
Apartado Aereo 51283
***BOGOTA** 1, D.E.*
Tel: 256-1686
Telex: 45403 CCAL CO.
A

COSTA RICA

Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
SAN JOSE
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
CM,E,M

CYPRUS

Telerexa Ltd.
P.O. Box 4809
14C Stassinos Avenue
NICOSIA
Tel: 62698
Telex: 2894 LEVIDO CY
E,M,P

DENMARK

Hewlett-Packard A/S
Datavej 52
DK-3460 **BIRKEROD**
Tel: (02) 81-66-40
Telex: 37409 hpas dk
A,CH,CM,CS,E,MS,P

Hewlett-Packard A/S
Rolighedsvej 32
DK-8240 **RISSKOV**, Aarhus
Tel: (06) 17-60-00
Telex: 37409 hpas dk
CH,E

DOMINICAN REPUBLIC

Microprog S.A.
Juan Tomás Mejía y Cotes No. 60
Arroyo Hondo
SANTO DOMINGO
Tel: 565-6268
Telex: 4510 ARENTA DR (RCA) P

ECUADOR

CYEDE Cia. Ltda.
Avenida Eloy Alfaro 1749
Casilla 6423 CCI
QUITO
Tel: 450-975, 243-052
Telex: 2548 CYEDE ED
CM,E,P

Hospitalar S.A.
Robles 625
Casilla 3590
QUITO
Tel: 545-250, 545-122
Telex: 2485 HOSPTL ED
Cable: HOSPITALAR-Quito
M

EGYPT

International Engineering Associates
24 Hussein Hegazi Street
Kasr-el-Aini
CAIRO
Tel: 23829, 21641
Telex: IEA UN 93830
CH,CS,E,M

EGYPOR
P.O.Box 2558
42 El Zahraa Street
***CAIRO**, Egypt*
Tel: 65 00 21
Telex: 93 337
P

EL SALVADOR

IPESA de El Salvador S.A.
29 Avenida Norte 1216
SAN SALVADOR
Tel: 26-6858, 26-6868
Telex: 20539 IPESASAL
A,CH,CM,CS,E,P

FINLAND

Hewlett-Packard Oy
Revontulentie 7
PL 24
SF-02101 **ESPOO** 10
Tel: (90) 4550211
Telex: 121563 hewpa sf
CH,CM,CS,P

Hewlett-Packard Oy
(Olarinluoma 7)
PL 24
02101 **ESPOO** 10
Tel: (90) 4521022
A,E,MS

Hewlett-Packard Oy
Aatoksenkatv 10-C
SF-40720-72 **JYVASKYLA**
Tel: (941) 216318
CH

Hewlett-Packard Oy
Kainvuntie 1-C
SF-90140-14 **OULU**
Tel: (981) 338785
CH

FRANCE

Hewlett-Packard France
Z.I. Mercure B
Rue Berthelot
F-13763 Les Milles Cedex
AIX-EN-PROVENCE
Tel: 16 (42) 59-41-02
Telex: 410770F
A,CH,E,MS,P*

Hewlett-Packard France
64, rue Marchand Saillant
F-61000 **ALENCON**
Tel: 16 (33) 29 04 42

Hewlett-Packard France
Boite Postale 503
F-25026 **BESANCON**
28 rue de la Republique
F-25000 **BESANCON**
Tel: 16 (81) 83-16-22
CH,M

Hewlett-Packard France
13, Place Napoleon III
F-29000 **BREST**
Tel: 16 (98) 03-38-35

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
F-69130 **ECULLY** Cedex (Lyon)
Tel: 16 (78) 833-81-25
Telex: 310617F
A,CH,CS,E,MP

Hewlett-Packard France
Tour Lorraine
Boulevard de France
F-91035 **EVRY** Cedex
Tel: 16 6 077-96-60
Telex: 692315F
E

Hewlett-Packard France
Parc d'Activite du Bois Briard
Ave. du Lac
F-91040 **EVRY** Cedex
Tel: 16 6 077-8383
Telex: 692315F
E

Hewlett-Packard France
5, avenue Raymond Chanas
F-38320 **EYBENS** (Grenoble)
Tel: 16 (76) 25-81-41
Telex: 980124 HP GRENOB EYBE
CH

Hewlett-Packard France
Centre d'Affaire Paris-Nord
Bâtiment Ampère 5 étage
Rue de la Commune de Paris
Boite Postale 300
F-93153 **LE BLANC MESNIL**
Tel: 16 (1) 865-44-52
Telex: 211032F
CH,CS,E,MS

Hewlett-Packard France
Parc d'Activités Cadera
Quartier Jean Mermoz
Avenue du Président JF Kennedy
F-33700 **MERIGNAC** (Bordeaux)
Tel: 16 (56) 34-00-84
Telex: 550105F
CH,E,MS

Hewlett-Packard France
Immeuble "Les 3 B"
Nouveau Chemin de la Garde
ZAC de Bois Briand
F-44085 **NANTES** Cedex
Tel: 16 (40) 50-32-22
CH**

FRANCE (Cont'd)

Hewlett-Packard France
125, rue du Faubourg Bannier
F-45000 **ORLEANS**
Tel: 16 (38) 68 01 63

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
F-91947 Les Ulis Cedex **ORSAY**
Tel: (6) 907-78-25
Telex: 600048F
A,CH,CM,CS,E,MP,P

Hewlett-Packard France
Paris Porte-Maillot
15, Avenue de L'Amiral Bruix
F-75782 **PARIS** CEDEX 16
Tel: 16 (1) 502-12-20
Telex: 613663F
CH,MS,P

Hewlett-Packard France
124, Boulevard Tourasse
F-64000 **PAU**
Tel: 16 (59) 80 38 02

Hewlett-Packard France
2 Allée de la Bourgonnette
F-35100 **RENNES**
Tel: 16 (99) 51-42-44
Telex: 740912F
CH,CM,E,MS,P*

Hewlett-Packard France
98 Avenue de Bretagne
F-76100 **ROUEN**
Tel: 16 (35) 63-57-66
CH**,CS

Hewlett-Packard France
4 Rue Thomas Mann
Boite Postale 56
F-67033 **STRASBOURG** Cedex
Tel: 16 (88) 28-56-46
Telex: 890141F
CH,E,MS,P*

Hewlett-Packard France
Le Péripole
20, Chemin du Pigeonnier de la Cépière
F-31083 **TOULOUSE** Cedex
Tel: 16 (61) 40-11-12
Telex: 531639F
A,CH,CS,E,P*

Hewlett-Packard France
9, rue Baudin
F-26000 **VALENCE**
Tel: 16 (75) 42 76 16

Hewlett-Packard France
Carolor
ZAC de Bois Briand
F-57640 **VIGY** (Metz)
Tel: 16 (8) 771 20 22
CH

Hewlett-Packard France
Immeuble Péricentre
F-59658 **VILLENEUVE D'ASCQ** Cedex
Tel: 16 (20) 91-41-25
Telex: 160124F
CH,E,MS,P*

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Geschäftsstelle
Keithstrasse 2-4
D-1000 **BERLIN** 30
Tel: (030) 24-90-86
Telex: 018 3405 hpbln d
A,CH,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Herrenberger Strasse 130
D-7030 **BÖBLINGEN**
Tel: (7031) 14-0
Telex:
A,CH,CM,CS,E,MP,P

Hewlett-Packard GmbH
Geschäftsstelle
Emanuel-Leutze-Strasse 1
D-4000 **DUSSELDORF**
Tel: (0211) 5971-1
Telex: 085/86 533 hpdd d
A,CH,CS,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Schleefstr. 28a
D-4600 **DORTMUND**-Aplerbeck
Tel: (0231) 45001

Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Berner Strasse 117
Postfach 560 140
D-6000 **FRANKFURT** 56
Tel: (0611) 50-04-1
Telex: 04 13249 hpffm d
A,CH,CM,CS,E,MP,P

Hewlett-Packard GmbH
Geschäftsstelle
Aussenstelle Bad Homburg
Louisenstrasse 115
D-6380 **BAD HOMBURG**
Tel: (06172) 109-0

Hewlett-Packard GmbH
Geschäftsstelle
Kapstadtring 5
D-2000 **HAMBURG** 60
Tel: (040) 63804-1
Telex: 021 63 032 hphh d
A,CH,CS,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Heidering 37-39
D-3000 **HANNOVER** 61
Tel: (0511) 5706-0
Telex: 092 3259
A,CH,CM,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Rosslauer Weg 2-4
D-6800 **MANNHEIM**
Tel: (0621) 70050
Telex: 0462105
A,C,E

Hewlett-Packard GmbH
Geschäftsstelle
Messerschmittstrasse 7
D-7910 **NEU ULM**
Tel: 0731-70241
Telex: 0712816 HP ULM-D
A,C,E*

Hewlett-Packard GmbH
Geschäftsstelle
Ehhericherstr. 13
D-8500 **NÜRNBERG** 10
Tel: (0911) 5205-0
Telex: 0623 860
CH,CM,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 6117-1
Telex: 0524985
A,CH,CM,E,MS,P

GREAT BRITAIN

See United Kingdom

GREECE

Kostas Karaynnis S.A.
8 Omirou Street
***ATHENS** 133*
Tel: 32 30 303, 32 37 371
Telex: 215962 RKAR GR
A,CH,CM,CS,E,M,P

PLAISIO S.A.
G. Gerardos
24 Stournara Street
ATHENS
Tel: 36-11-160
Telex: 221871
P

GUATEMALA

IPESA
Avenida Reforma 3-48, Zona 9
GUATEMALA CITY
Tel: 316627, 314786
Telex: 4192 TELTRO GU
A,CH,CM,CS,E,M,P

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG
E,CH,CS,P

CET Ltd.
1402 Tung Wah Mansion
199-203 Hennessy Rd.
*Wanchia, **HONG KONG***
Tel: 5-729376
Telex: 85148 CET HX
CM

Schmidt & Co. (Hong Kong) Ltd.
Wing On Centre, 28th Floor
Connaught Road, C.
HONG KONG
Tel: 5-455644
Telex: 74766 SCHMX HX
A,M

ICELAND

Elding Trading Company Inc.
Hafnarnvoli-Tryggvagotu
P.O. Box 895
IS-REYKJAVIK
Tel: 1-58-20, 1-63-03
M

INDIA

Computer products are sold through Blue Star Ltd. All computer repairs and maintenance service is done through Computer Maintenance Corp.

Blue Star Ltd.
Sabri Complex II Floor
24 Residency Rd.
***BANGALORE** 560 025*
Tel: 55660
Telex: 0845-430
Cable: BLUESTAR
A,CH,CM,CS*,E*

Blue Star Ltd.
Band Box House
Prabhadevi
***BOMBAY** 400 025*
Tel: 422-3101
Telex: 011-3751
Cable: BLUESTAR
A,M

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
***BOMBAY** 400 025*
Tel: 422-6155
Telex: 011-4093
Cable: FROSTBLUE
A,CH,CM,CS*,E,M*

Blue Star Ltd.
Kalyan, 19 Vishwas Colony
*Alkapuri, **BORODA**, 390 005*
Tel: 65235
Cable: BLUE STAR
A

Blue Star Ltd.
7 Hare Street
***CALCUTTA** 700 001*
Tel: 12-01-31
Telex: 021-7655
Cable: BLUESTAR
A,M

Blue Star Ltd.
133 Kodambakkam High Road
***MADRAS** 600 034*
Tel: 82057
Telex: 041-379
Cable: BLUESTAR
A,M

Blue Star Ltd.
Bhandari House, 7th/8th Floors
91 Nehru Place
***NEW DELHI** 110 024*
Tel: 682547
Telex: 031-2463
Cable: BLUESTAR
A,CH,CM,CS*,E,M*

Blue Star Ltd.
15/16:C Wellesley Rd.
***PUNE** 411 011*
Tel: 22775
Cable: BLUE STAR
A

Blue Star Ltd.
2-2-47/1108 Bolarum Rd.
***SECUNDERABAD** 500 003*
Tel: 72057
Telex: 0155-459
Cable: BLUEFROST
A,E

Blue Star Ltd.
T.C. 7/603 Poornima
Maruthankuzhi
***TRIVANDRUM** 695 013*
Tel: 65799
Telex: 0884-259
Cable: BLUESTAR
E

Computer Maintenance Corporation Ltd.
115, Sarojini Devi Road
***SECUNDERABAD** 500 003*
Tel: 310-184, 345-774
Telex: 031-2960
*CH***

SALES & SUPPORT OFFICES

Arranged alphabetically by country

INDONESIA

BERCA Indonesia P.T.
P.O.Box 496/Jkt.
Jl. Abdul Muis 62
JAKARTA
Tel: 21-373009
Telex: 46748 BERSAL IA
Cable: BERSAL JAKARTA
P

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 17th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 21-344-181
Telex: BERSAL IA
A,CS,E,M

BERCA Indonesia P.T.
P.O. Box 174/SBY.
Jl. Kutei No. 11
SURABAYA
Tel: 68172
Telex: 31146 BERSAL SB
Cable: BERSAL-SURABAYA
A,E,M,P*

IRAQ

Hewlett-Packard Trading S.A.
Service Operation
Al Mansoor City 9B/3/7
BAGHDAD
Tel: 551-49-73
Telex: 212-455 HEPAIRAQ IK
CH,CS

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439
A,CH,CM,CS,E,M,P

Cardiac Services Ltd.
Kilmore Road
Artane
***DUBLIN** 5*
Tel: (01) 351820
Telex: 30439
M

ISRAEL

Eldan Electronic Instrument Ltd.
P.O.Box 1270
***JERUSALEM** 91000*
16, Ohaliav St.
***JERUSALEM** 94467*
Tel: 533 221, 553 242
Telex: 25231 AB/PAKRD IL
A

Electronics Engineering Division
Motorola Israel Ltd.
16 Kremenetski Street
P.O. Box 25016
TEL-AVIV 67899
Tel: 3 88 388
Telex: 33569 Motil IL
Cable: BASTEL Tel-Aviv
CH,CM,CS,E,M,P

ITALY

Hewlett-Packard Italiana S.p.A
Traversa 99C
Via Giulio Petroni, 19
I-70124 **BARI**
Tel: (080) 41-07-44
M

Hewlett-Packard Italiana S.p.A.
Via Martin Luther King, 38/III
I-40132 **BOLOGNA**
Tel: (051) 402394
Telex: 511630
CH,E,MS

Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43G/C
I-95126 **CATANIA**
Tel: (095) 37-10-87
Telex: 970291
C,P

Hewlett-Packard Italiana S.p.A.
Via G. Di Vittorio 9
I-20063 **CERNUSCO SUL NAVIGLIO**
(Milano)
Tel: (02) 923691
Telex: 334632
A,CH,CM,CS,E,MP,P

Hewlett-Packard Italiana S.p.A.
Via C. Colombo 49
I-20090 **TREZZANO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 322116
C,M

Hewlett-Packard Italiana S.p.A.
Via Nuova San Rocco a
Capodimonte, 62/A
I-80131 **NAPOLI**
Tel: (081) 7413544
Telex: 710698
A,CH,E

Hewlett-Packard Italiana S.p.A.
Viale G. Modugno 33
I-16156 **GENOVA PEGLI**
Tel: (010) 68-37-07
Telex: 215238
E,C

Hewlett-Packard Italiana S.p.A.
Via Pelizzo 15
I-35128 **PADOVA**
Tel: (049) 664888
Telex: 430315
A,CH,E,MS

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 **ROMA EUR**
Tel: (06) 54831
Telex: 610514
A,CH,CM,CS,E,MS,P*

Hewlett-Packard Italiana S.p.A.
Via di Casellina 57/C
I-50018 **SCANDICCI-FIRENZE**
Tel: (055) 753863

Hewlett-Packard Italiana S.p.A.
Corso Svizzera, 185
I-10144 **TORINO**
Tel: (011) 74 4044
Telex: 221079
CH,E

JAPAN

Yokogawa-Hewlett-Packard Ltd.
152-1, Onna
ATSUGI, Kanagawa, 243
Tel: (0462) 28-0451
CM,C*,E

Yokogawa-Helwett-Packard Ltd.
Meiji-Seimei Bldg. 6F
3-1 Hon Chiba-Cho
CHIBA, 280
Tel: 472 25 7701
E,CH,CS

Yokogawa-Hewlett-Packard Ltd.
Yasuda-Seimei Hiroshima Bldg.
6-11, Hon-dori, Naka-ku
HIROSHIMA, 730
Tel: 82-241-0611

Yokogawa-Hewlett-Packard Ltd.
Towa Building
2-3, Kaigan-dori, 2 Chome Chuo-ku
KOBE, 650
Tel: (078) 392-4791
C,E

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi 82 Bldg
3-4 Tsukuba
KUMAGAYA, Saitama 360
Tel: (0485) 24-6563
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Asahi Shinbun Daiichi Seimei Bldg.
4-7, Hanabata-cho
KUMAMOTO,860
Tel: (0963) 54-7311
CH,E

Yokogawa-Hewlett-Packard Ltd.
Shin-Kyoto Center Bldg.
614, Higashi-Shiokoji-cho
Karasuma-Nishiiru
Shiokoji-dori, Shimogyo-ku
KYOTO, 600
Tel: 075-343-0921
CH,E

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Bldg
4-73, Sanno-maru, 1 Chome
MITO, Ibaraki 310
Tel: (0292) 25-7470
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Sumitomo Seimei 14-9 Bldg.
Meieki-Minami, 2 Chome
Nakamura-ku
NAGOYA, 450
Tel: (052) 571-5171
CH,CM,CS,E,MS

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
A,CH,CM,CS,E,MP,P*

Yokogawa-Hewlett-Packard Ltd.
27-15, Yabe, 1 Chome
SAGAMIHARA Kanagawa, 229
Tel: 0427 59-1311

Yokogawa-Hewlett-Packard Ltd.
Daiichi Seimei Bldg.
7-1, Nishi Shinjuku, 2 Chome
Shinjuku-ku,**TOKYO** 160
Tel: 03-348-4611
CH,E

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-611
Telex: 232-2024 YHPTOK
A,CH,CM,CS,E,MP,P*

Yokogawa-Hewlett-Packard Ltd.
Daiichi Asano Building
2-8, Odori, 5 Chome
UTSUNOMIYA, Tochigi 320
Tel: (0286) 25-7155
CH,CS,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
YOKOHAMA 221
Tel: (045) 312-1252
CH,CM,E

JORDAN

Mouasher Cousins Company
P.O. Box 1387
AMMAN
Tel: 24907, 39907
Telex: 21456 SABCO JO
CH,E,M,P

KENYA

ADCOM Ltd., Inc., Kenya
P.O.Box 30070
NAIROBI
Tel: 331955
Telex: 22639
E,M

KOREA

Samsung Electronics HP Division
12 Fl. Kinam Bldg.
San 75-31, Yeoksam-Dong
Kangnam-Ku
Yeongdong P.O. Box 72
SEOUL
Tel: 555-7555, 555-5447
Telex: K27364 SAMSAN
A,CH,CM,CS,E,M,P

KUWAIT

Al-Khaldiya Trading & Contracting
P.O. Box 830 Safat
KUWAIT
Tel: 42-4910, 41-1726
Telex: 22481 Areeg kt
CH,E,M

Photo & Cine Equipment
P.O. Box 270 Safat
KUWAIT
Tel: 42-2846, 42-3801
Telex: 22247 Matin kt
P

LEBANON

G.M. Dolmadjian
Achrafieh
P.O. Box 165.167
BEIRUT
Tel: 290293
*MP***

Computer Information Systems
P.O. Box 11-6274
BEIRUT
Tel: 89 40 73
Telex: 22259
C

LUXEMBOURG

Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,MP,P

MALAYSIA

Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
1st Floor, Bangunan British
American
Jalan Semantan, Damansara Heights
KUALA LUMPUR 23-03
Tel: 943022
Telex: MA31011
A,CH,E,M,P*

MAYLAYSIA (Cont'd)

Protel Engineering
P.O.Box 1917
Lot 6624, Section 64
23/4 Pending Road
*Kuching, **SARAWAK***
Tel: 36299
Telex: MA 70904 PROMAL
Cable: PROTELENG
A,E,M

MALTA

Philip Toledo Ltd.
Notabile Rd.
MRIEHEL
Tel: 447 47, 455 66
Telex: Media MW 649
E,P

MEXICO

Hewlett-Packard Mexicana, S.A. de C.V.
Av. Periferico Sur No. 6501
Tepepan, Xochimilco
16020 **MEXICO D.F.**
Tel: 6-76-46-00
Telex: 17-74-507 HEWPACK MEX
A,CH,CS,E,MS,P

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Colonia del Valle 409
Col. del Valle
Municipio de Garza Garcia
MONTERREY, Nuevo Leon
Tel: 78 42 41
Telex: 038 410
CH

ECISA
José Vasconcelos No. 218
Col. Condesa Deleg. Cuauhtémoc
***MEXICO** D.F. 06140*
Tel: 553-1206
Telex: 17-72755 ECE ME
M

MOROCCO

Dolbeau
81 rue Karatchi
CASABLANCA
Tel: 3041-82, 3068-38
Telex: 23051, 22822
E

Gerep
2 rue d'Agadir
Boite Postale 156
CASABLANCA
Tel: 272093, 272095
Telex: 23 739
P

NETHERLANDS

Hewlett-Packard Nederland B.V.
Van Heuven Goedhartlaan 121
NL 1181KK **AMSTELVEEN**
P.O. Box 667
NL1180 AR **AMSTELVEEN**
Tel: (020) 47-20-21
Telex: 13 216 HEPA NL
A,CH,CM,CS,E,MP,P

Hewlett-Packard Nederland B.V.
Bongerd 2
NL 2906VK **CAPELLE A/D IJSSEL**
P.O. Box 41
NL 2900AA **CAPELLE A/D IJSSEL**
Tel: (10) 51-64-44
Telex: 21261 HEPAC NL
A,CH,CS,E

Hewlett-Packard Nederland B.V.
Pastoor Petersstraat 134-136
NL 5612 LV **EINDHOVEN**
P.O. Box 2342
NL 5600 CH **EINDHOVEN**
Tel: (040) 326911
Telex: 51484 hepae nl
A,CH**,E,M

NEW ZEALAND

Hewlett-Packard (N.Z.) Ltd.
5 Owens Road
P.O. Box 26-189
Epsom, **AUCKLAND**
Tel: 687-159
Cable: HEWPACK Auckland
CH,CM,E,P*

Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, **WELLINGTON 3**
P.O. Box 9443
Courtenay Place, **WELLINGTON 3**
Tel: 877-199
Cable: HEWPACK Wellington
CH,CM,E,P

Northrop Instruments & Systems Ltd.
369 Khyber Pass Road
P.O. Box 8602
AUCKLAND
Tel: 794-091
Telex: 60605
A,M

Northrop Instruments & Systems Ltd.
110 Mandeville St.
P.O. Box 8388
CHRISTCHURCH
Tel: 486-928
Telex: 4203
A,M

Northrop Instruments & Systems Ltd.
Sturdee House
85-87 Ghuznee Street
P.O. Box 2406
WELLINGTON
Tel: 850-091
Telex: NZ 3380
A,M

NORTHERN IRELAND

See United Kingdom

NORWAY

Hewlett-Packard Norge A/S
Folke Bernadottes vei 50
P.O. Box 3558
N-5033 **FYLLINGSDALEN** (Bergen)
Tel: 0047/5/16 55 40
Telex: 16621 hpnas n
CH,CS,E,MS

Hewlett-Packard Norge A/S
Österndalen 16-18
P.O. Box 34
N-1345 **ÖSTERÅS**
Tel: 0047/2/17 11 80
Telex: 16621 hpnas n
A,CH,CM,CS,E,M,P

OMAN

Khimjil Ramdas
P.O. Box 19
MUSCAT
Tel: 722225, 745601
Telex: 3289 BROKER MB MUSCAT
P

Suhail & Saud Bahwan
P.O.Box 169
MUSCAT
Tel: 734 201-3
Telex: 3274 BAHWAN MB

PAKISTAN

Mushko & Company Ltd.
1-B, Street 43
Sector F-8/1
ISLAMABAD
Tel: 51071
Cable: FEMUS Rawalpindi
A,E,M

Mushko & Company Ltd.
Oosman Chambers
Abdullah Haroon Road
***KARACHI** 0302*
Tel: 524131, 524132
Telex: 2894 MUSKO PK
Cable: COOPERATOR Karachi
*A,E,M,P**

PANAMA

Electrónico Balboa, S.A.
Calle Samuel Lewis, Ed. Alfa
Apartado 4929
PANAMA 5
Tel: 63-6613, 63-6748
Telex: 3483 ELECTRON PG
A,CM,E,M,P

PERU

Cía Electro Médica S.A.
Los Flamencos 145, San Isidro
Casilla 1030
***LIMA** 1*
Tel: 41-4325, 41-3703
Telex: Pub. Booth 25306
CM,E,M,P

PHILIPPINES

The Online Advanced Systems Corporation
Rico House, Amorsolo Cor. Herrera Street
Legaspi Village, Makati
P.O. Box 1510
*Metro **MANILA***
Tel: 85-35-81, 85-34-91, 85-32-21
Telex: 3274 ONLINE
A,CH,CS,E,M

Electronic Specialists and Proponents Inc.
690-B Epifanio de los Santos Avenue
*Cubao, **QUEZON CITY***
P.O. Box 2649 Manila
Tel: 98-96-81, 98-96-82, 98-96-83
Telex: 40018, 42000 ITT GLOBE MACKAY BOOTH
P

PORTUGAL

Mundinter
Intercambio Mundial de Comércio S.A.R.L.
P.O. Box 2761
Av. Antonio Augusto de Aguiar 138
P-LISBON
Tel: (19) 53-21-31, 53-21-37
Telex: 16691 munter p
M

Soquimica
Av. da Liberdade, 220-2
*1298 **LISBOA** Codex*
Tel: 56 21 81/2/3
Telex: 13316 SABASA
P

Telectra-Empresa Técnica de Equipmentos Eléctricos S.A.R.L.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
***P-LISBON** 1*
Tel: (19) 68-60-72
Telex: 12598
CH,CS,E,P

PUERTO RICO

Hewlett-Packard Puerto Rico
Ave. Muñoz Rivera #101
Esq. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800

Hewlett-Packard Puerto Rico
Calle 272 Edificio 203
Urb. Country Club
RIO PIEDRAS, Puerto Rico
P.O. Box 4407
CAROLINA, Puerto Rico 00628
Tel: (809) 762-7255
A,CH,CS

QATAR

Computearbia
P.O. Box 2750
DOHA
Tel: 883555
Telex: 4806 CHPARB
P

Eastern Technical Services
P.O.Box 4747
DOHA
Tel: 329 993
Telex: 4156 EASTEC DH

Nasser Trading & Contracting
P.O.Box 1563
DOHA
Tel: 22170, 23539
Telex: 4439 NASSER DH
M

SAUDI ARABIA

Modern Electronic Establishment
Hewlett-Packard Division
P.O. Box 22015
Thuobah
AL-KHOBAR
Tel: 895-1760, 895-1764
Telex: 671 106 HPMEEK SJ
Cable: ELECTA AL-KHOBAR
CH,CS,E,M

Modern Electronic Establishment
Hewlett-Packard Division
P.O. Box 1228
Redec Plaza, 6th Floor
JEDDAH
Tel: 644 38 48
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH
CH,CS,E,M

Modern Electronic Establishment
Hewlett-Packard Division
P.O.Box 22015
RIYADH
Tel: 491-97 15, 491-63 87
Telex: 202049 MEERYD SJ
CH,CS,E,M

Abdul Ghani El Ajou
P.O. Box 78
RIYADH
Tel: 40 41 717
Telex: 200 932 EL AJOU
P

SCOTLAND

See United Kingdom

SINGAPORE

Hewlett-Packard Singapore (Sales) Pte. Ltd.
#08-00 Inchcape House
450-2 Alexandra Road
P.O. Box 58 Alexandra Rd. Post Office
SINGAPORE, 9115
Tel: 631788
Telex: HPSGSO RS 34209
Cable: HEWPACK, Singapore
A,CH,CS,E,MS,P

SALES & SUPPORT OFFICES
Arranged alphabetically by country

SINGAPORE (Cont'd)
Dynamar International Ltd.
Unit 05-11 Block 6
Kolam Ayer Industrial Estate
***SINGAPORE** 1334*
Tel: 747-6188
Telex: RS 26283
CM

SOUTH AFRICA
Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 120
Howard Place **CAPE PROVINCE** 7450
Pine Park Center, Forest Drive,
Pinelands
CAPE PROVINCE 7405
Tel: 53-7954
Telex: 57-20006
A,CH,CM,E,MS,P

Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 37099
92 Overport Drive
DURBAN 4067
Tel: 28-4178, 28-4179, 28-4110
Telex: 6-22954
CH,CM

Hewlett-Packard So Africa (Pty.) Ltd.
6 Linton Arcade
511 Cape Road
Linton Grange
PORT ELIZABETH 6000
Tel: 041-302148
CH

Hewlett-Packard So Africa (Pty.) Ltd.
P.O.Box 33345
Glenstantia 0010 **TRANSVAAL**
1st Floor East
Constantia Park Ridge Shopping
Centre
Constantia Park
PRETORIA
Tel: 982043
Telex: 32163
CH,E

Hewlett-Packard So Africa (Pty.) Ltd.
Private Bag Wendywood
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877
Cable: HEWPACK Johannesburg
A,CH,CM,CS,E,MS,P

SPAIN
Hewlett-Packard Española S.A.
Calle Entenza, 321
E-BARCELONA 29
Tel: 322.24.51, 321.73.54
Telex: 52603 hpbee
A,CH,CS,E,MS,P

Hewlett-Packard Española S.A.
Calle San Vicente S/No
Edificio Albia II
E-BILBAO 1
Tel: 423.83.06
A,CH,E,MS

Hewlett-Packard Española S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-MADRID
Tel: (1) 637.00.11
CH,CS,M

Hewlett-Packard Española S.A.
Avda. S. Francisco Javier, S/no
Planta 10. Edificio Sevilla 2,
E-SEVILLA 5
Tel: 64.44.54
Telex: 72933
A,CS,MS,P

Hewlett-Packard Española S.A.
Calle Ramon Gordillo, 1 (Entlo.3)
E-VALENCIA 10
Tel: 361-1354
CH,P

SWEDEN
Hewlett-Packard Sverige AB
Sunnanvagen 14K
S-22226 **LUND**
Tel: (046) 13-69-79
Telex: (854) 17886 (via Spånga office)
CH

Hewlett-Packard Sverige AB
Östra Tullgatan 3
S-21128 **MALMÖ**
Tel: (040) 70270
Telex: (854) 17886 (via Spånga office)

Hewlett-Packard Sverige AB
Västra Vintergatan 9
S-70344 **ÖREBRO**
Tel: (19) 10-48-80
Telex: (854) 17886 (via Spånga office)
CH

Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781
A,CH,CM,CS,E,MS,P

Hewlett-Packard Sverige AB
Frötallisgatan 30
S-42132 **VÄSTRA-FRÖLUNDA**
Tel: (031) 49-09-50
Telex: (854) 17886 (via Spånga office)
CH,E,P

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Clarastrasse 12
CH-4058 **BASEL**
Tel: (61) 33-59-20
A

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case Postale 365
CH-1217 **MEYRIN** 2
Tel: (0041) 22-83-11-11
Telex:27333 HPAG CH
CH,CM,CS

Hewlett-Packard (Schweiz) AG
Allmend 2
CH-8967 **WIDEN**
Tel: (0041) 57 31 21 11
Telex: 53933 hpag ch
Cable: HPAG CH
A,CH,CM,CS,E,MS,P

SYRIA
General Electronic Inc.
Nuri Basha Ahnaf Ebn Kays Street
P.O. Box 5781
DAMASCUS
Tel: 33-24-87
Telex: 411 215
Cable: ELECTROBOR DAMASCUS
E

Middle East Electronics
P.O.Box 2308
Abu Rumnaneh
DAMASCUS
Tel: 33 4 5 92
Telex: 411 304
M

TAIWAN
Hewlett-Packard Far East Ltd.
Kaohsiung Office
2/F 68-2, Chung Cheng 3rd Road
KAOHSIUNG
Tel: (07) 241-2318
CH,CS,E

Hewlett-Packard Far East Ltd.
Taiwan Branch
8th Floor
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable:HEWPACK Taipei
A,CH,CM,CS,E,M,P

Ing Lih Trading Co.
3rd Floor, 7 Jen-Ai Road, Sec. 2
TAIPEI 100
Tel: (02) 3948191
Cable: INGLIH TAIPEI
A

THAILAND
Unimesa
30 Patpong Ave., Suriwong
***BANGKOK** 5*
Tel: 235-5727
Telex: 84439 Simonco TH
Cable: UNIMESA Bangkok
A,CH,CS,E,M

Bangkok Business Equipment Ltd.
5/5-6 Dejo Road
BANGKOK
Tel: 234-8670, 234-8671
Telex: 87669-BEQUIPT TH
Cable: BUSIQUIPT Bangkok
P

TRINIDAD & TOBAGO
Caribbean Telecoms Ltd.
50/A Jerningham Avenue
P.O. Box 732
PORT-OF-SPAIN
Tel: 62-44213, 62-44214
Telex: 235,272 HUGCO WG
CM,E,M,P

TUNISIA
Tunisie Electronique
31 Avenue de la Liberte
TUNIS
Tel: 280-144
E,P

Corema
1 ter. Av. de Carthage
TUNIS
Tel: 253-821
Telex: 12319 CABAM TN
M

TURKEY
Teknim Company Ltd.
Iran Caddesi No. 7
*Kavaklidere, **ANKARA***
Tel: 275800
Telex: 42155 TKNM TR
E

E.M.A.
Medina Eldem Sokak No.41/6
Yuksel Caddesi
ANKARA
Tel: 175 622
Telex: 42 591
M

UNITED ARAB EMIRATES
Emitac Ltd.
P.O. Box 2711
ABU DHABI
Tel: 82 04 19-20
Cable: EMITAC ABUDHABI

Emitac Ltd.
P.O. Box 1641
SHARJAH
Tel: 591 181
Telex: 68136 Emitac Sh
CH,CS,E,M,P

UNITED KINGDOM
GREAT BRITAIN
Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
ALTRINCHAM
Cheshire WA14 1NU
Tel: 061 928 6422
Telex: 668068
A,CH,CS,E,M,MS,P

Hewlett-Packard Ltd.
Elstree House, Elstree Way
BOREHAMWOOD, Herts WD6 1SG
Tel: 01 207 5000
Telex: 8952716
E,CH,CS,P

Hewlett-Packard Ltd.
Oakfield House, Oakfield Grove
Clifton **BRISTOL,** Avon BS8 2BN
Tel: 0272 736806
Telex: 444302
CH,CS,E,P

Hewlett-Packard Ltd.
Bridewell House
Bridewell Place
LONDON EC4V 6BS
Tel: 01 583 6565
Telex: 298163
CH,CS,P

Hewlett-Packard Ltd.
Fourier House
257-263 High Street
LONDON COLNEY
Herts. AL2 1HA, St. Albans
Tel: 0727 24400
Telex: 1-8952716
CH,CS

Hewlett-Packard Ltd.
Pontefract Road
NORMANTON, West Yorkshire WF6 1RN
Tel: 0924 895566
Telex: 557355
CH,CS,P

Hewlett-Packard Ltd.
The Quadrangle
106-118 Station Road
REDHILL, Surrey RH1 1PS
Tel: 0737 68655
Telex: 947234
CH,CS,E,P

GREAT BRITAIN (Cont'd)
Hewlett-Packard Ltd.
Avon House
435 Stratford Road
Shirley, **SOLIHULL**, West Midlands
B90 4BL
Tel: 021 745 8800
Telex: 339105
CH,CS,E,P

Hewlett-Packard Ltd.
West End House
41 High Street, West End
SOUTHAMPTON
Hampshire S03 3DQ
Tel: 04218 6767
Telex: 477138
CH,CS,P

Hewlett-Packard Ltd.
Eskdale Rd.
Winnersh, **WOKINGHAM**
Berkshire RG11 5DZ
Tel: 0734 696622
Telex: 848884
E

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178
A,CH,CS,E,M,MP,P

Hewlett-Packard Ltd.
Nine Mile Ride
Easthampstead, **WOKINGHAM**
Berkshire, 3RG11 3LL
Tel: 0344 773100
Telex: 848805
CH,CS,E,P

IRELAND

NORTHERN IRELAND
Hewlett-Packard Ltd.
Cardiac Services Building
95A Finaghy Road South
BELFAST BT10 0BY
Tel: 0232 625-566
Telex: 747626
CH,CS

SCOTLAND
Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682
CH,CM,CS,E,M,P

UNITED STATES

Alabama
Hewlett-Packard Co.
700 Century Park South, Suite 128
BIRMINGHAM, AL 35226
Tel: (205) 822-6802
A,CH,M

Hewlett-Packard Co.
420 Wynn Drive
HUNTSVILLE, AL 35805
P.O. Box 7700
HUNTSVILLE, AL 35807
Tel: (205) 830-2000
CH,CM,CS,E,M*

Arizona
Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
2424 East Aragon Road
TUCSON, AZ 85706
Tel: (602) 889-4631
CH,E,MS**

California
Hewlett-Packard Co.
99 South Hill Dr.
BRISBANE, CA 94005
Tel: (415) 330-2500
CH,CS

Hewlett-Packard Co.
P.O. Box 7830 (93747)
5060 E. Clinton Avenue, Suite 102
FRESNO, CA 93727
Tel: (209) 252-9652
CH,CS,MS

Hewlett-Packard Co.
P.O. Box 4230
1430 East Orangethorpe
FULLERTON, CA 92631
Tel: (714) 870-1000
CH,CM,CS,E,MP

Hewlett-Packard Co.
320 S. Kellogg, Suite B
GOLETA, CA 93117
Tel: (805) 967-3405
CH

Hewlett-Packard Co.
5400 W. Rosecrans Boulevard
LAWNDALE, CA 90260
P.O. Box 92105
LOS ANGELES, CA 90009
Tel: (213) 970-7500
Telex: 910-325-6608
CH,CM,CS,MP

Hewlett-Packard Co.
3155 Porter Oaks Drive
PALO ALTO, CA 94304
Tel: (415) 857-8000
CH,CS,E

Hewlett-Packard Co.
4244 So. Market Court, Suite A
P.O. Box 15976
SACRAMENTO, CA 95852
Tel: (916) 929-7222
A*,CH,CS,E,MS

Hewlett-Packard Co.
9606 Aero Drive
P.O. Box 23333
SAN DIEGO, CA 92139
Tel: (619) 279-3200
CH,CM,CS,E,MP

Hewlett-Packard Co.
2305 Camino Ramon "C"
SAN RAMON, CA 94583
Tel: (415) 838-5900
CH,CS

Hewlett-Packard Co.
3005 Scott Boulevard
SANTA CLARA, CA 95050
Tel: (408) 988-7000
Telex: 910-338-0586
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
5703 Corsa Avenue
WESTLAKE VILLAGE, CA 91362
Tel: (213) 706-6800
E*,CH*,CS*

Colorado
Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000
A,CH,CM,CS,E,MS

Connecticut
Hewlett-Packard Co.
47 Barnes Industrial Road South
P.O. Box 5007
WALLINGFORD, CT 06492
Tel: (203) 265-7801
A,CH,CM,CS,E,MS

Florida
Hewlett-Packard Co.
2901 N.W. 62nd Street
P.O. Box 24210
FORT LAUDERDALE, FL 33307
Tel: (305) 973-2600
CH,CS,E,MP

Hewlett-Packard Co.
6177 Lake Ellenor Drive
P.O. Box 13910
ORLANDO, FL 32859
Tel: (305) 859-2900
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
5750B N. Hoover Blvd., Suite 123
P.O. Box 15200
TAMPA, FL 33614
Tel: (813) 884-3282
A*,CH,CM,CS,E*,M*

Georgia
Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500
Telex: 810-766-4890
A,CH,CM,CS,E,MP

Hawaii
Hewlett-Packard Co.
Kawaiahao Plaza, Suite 190
567 South King Street
HONOLULU, HI 96813
Tel: (808) 526-1555
A,CH,E,MS

Illinois
Hewlett-Packard Co.
304 Eldorado Road
P.O. Box 1607
BLOOMINGTON, IL 61701
Tel: (309) 662-9411
CH,MS**

Hewlett-Packard Co.
1100 31st Street, Suite 100
DOWNERS GROVE, IL 60515
Tel: (312) 960-5760
CH,CS

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066
A,CH,CM,CS,E,MP

Indiana
Hewlett-Packard Co.
7301 No. Shadeland Avenue
P.O. Box 50807
INDIANAPOLIS, IN 46250
Tel: (317) 842-1000
A,CH,CM,CS,E,MS

Iowa
Hewlett-Packard Co.
1776 22nd Street, Suite 1
WEST DES MOINES, IA 50265
Tel: (515) 224-1435
CH,MS**

Kansas
Hewlett-Packard Co.
7804 East Funston Road, #203
WICHITA, KS 67207
Tel: (316) 684-8491
CH

Kentucky
Hewlett-Packard Co.
10300 Linn Station Road, #100
LOUISVILLE, KY 40223
Tel: (502) 426-0100
A,CH,CS,MS

Louisiana
Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100
A,CH,CS,E,MS

Maryland
Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370
A,CH,CM,CS,E,MP

Massachusetts
Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500
A,C,CH,CS,CM,E,MP,P*

Hewlett-Packard Co.
32 Hartwell Avenue
LEXINGTON, MA 02173
Tel: (617) 861-8960
CH,CS,E

Michigan
Hewlett-Packard Co.
4326 Cascade Road S.E.
GRAND RAPIDS, MI 49506
Tel: (616) 957-1970
CH,CS,MS

Hewlett-Packard Co.
1771 W. Big Beaver Road
TROY, MI 48084
Tel: (313) 643-6474
CH,CS

Minnesota
Hewlett-Packard Co.
2025 W. Larpenteur Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100
A,CH,CM,CS,E,MP

Missouri
Hewlett-Packard Co.
11131 Colorado Avenue
KANSAS CITY, MO 64137
Tel: (816) 763-8000
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100
A,CH,CS,E,MP

SALES & SUPPORT OFFICES
Arranged alphabetically by country

UNITED STATES (Cont'd)

Nebraska
Hewlett-Packard
10824 Old Mill Rd., Suite 3
OMAHA, NE 68154
Tel: (402) 334-1813
CM,MS

New Jersey
Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07652
Tel: (201) 265-5000
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
60 New England Av. West
PISCATAWAY, NJ 08854
Tel: (201) 981-1199
A,CH,CM,CS,E

New Mexico
Hewlett-Packard Co.
11300 Lomas Blvd.,N.E.
P.O. Box 11634
ALBUQUERQUE, NM 87112
Tel: (505) 292-1330
CH,CS,E,MS

New York
Hewlett-Packard Co.
5 Computer Drive South
ALBANY, NY 12205
Tel: (518) 458-1550
A,CH,E,MS

Hewlett-Packard Co.
9600 Main Street
P.O. Box AC
CLARENCE, NY 14031
Tel: (716) 759-8621
CH

Hewlett-Packard Co.
200 Cross Keys Office Park
FAIRPORT, NY 14450
Tel: (716) 223-9950
CH,CM,CS,E,MS

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820
A,CH,CM,E,MS

Hewlett-Packard Co.
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
MANHATTAN NY 10119
Tel: (212) 971-0800
CH,CS,E*,M*

Hewlett-Packard Co.
250 Westchester Avenue
WHITE PLAINS, NY 10604
Tel: (914) 684-6100
CM,CH,CS,E

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 921-0300
A,CH,CM,CS,E,MS

North Carolina
Hewlett-Packard Co.
5605 Roanne Way
P.O. Box 26500
GREENSBORO, NC 27420
Tel: (919) 852-1800
A,CH,CM,CS,E,MS

Ohio
Hewlett-Packard Co.
9920 Carver Road
CINCINNATI, OH 45242
Tel: (513) 891-9870
CH,CS,MS

Hewlett-Packard Co.
16500 Sprague Road
CLEVELAND, OH 44130
Tel: (216) 243-7300
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
962 Crupper Ave.
COLUMBUS, OH 43229
Tel: (614) 436-1041
Eff: Nov. 25, 1983
675 Brooksedge Blvd.
WESTERVILLE, OH 43081
CH,CM,CS,E*

Hewlett-Packard Co.
330 Progress Rd.
DAYTON, OH 45449
Tel: (513) 859-8202
A,CH,CM,E*,MS

Oklahoma
Hewlett-Packard Co.
304 N. Meridian, Suite A
P.O. Box 75609
OKLAHOMA CITY, OK 73147
Tel: (405) 946-9499
A*,CH,E*,MS

Hewlett-Packard Co.
3840 S. 103rd E. Avenue, #100
P.O. Box 35747
TULSA, OK 74153
Tel: (918) 665-3300
A**,CH,CS,M*

Oregon
Hewlett-Packard Co.
9255 S. W. Pioneer Court
P.O. Box 328
WILSONVILLE, OR 97070
Tel: (503) 682-8000
A,CH,CS,E*,MS

Pennsylvania
Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400
A,CH,CS,E,MP

Hewlett-Packard Co.
2750 Monroe Boulevard
P.O. Box 713
VALLEY FORGE, PA 19482
Tel: (215) 666-9000
A,CH,CM,E,M

South Carolina
Hewlett-Packard Co.
Brookside Park, Suite 122
1 Harbison Way
P.O. Box 21708
COLUMBIA, SC 29221
Tel: (803) 732-0400
CH,E,MS

Hewlett-Packard Co.
Koger Executive Center
Chesterfield Bldg., Suite 124
GREENVILLE, SC 29615
Tel: (803) 297-4120

Tennessee
Hewlett-Packard Co.
224 Peters Road, Suite 102
P.O. Box 22490
KNOXVILLE, TN 37922
Tel: (615) 691-2371
A*,CH,MS

Hewlett-Packard Co.
3070 Directors Row
MEMPHIS, TN 38131
Tel: (901) 346-8370
A,CH,MS

Texas
Hewlett-Packard Co.
4171 North Mesa
Suite C-110
EL PASO, TX 79902
Tel: (915) 533-3555
CH,E*,MS**

Hewlett-Packard Co.
10535 Harwin Drive
P.O. Box 42816
HOUSTON, TX 77042
Tel: (713) 776-6400
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
930 E. Campbell Rd.
P.O. Box 1270
RICHARDSON, TX 75080
Tel: (214) 231-6101
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
1020 Central Parkway South
P.O. Box 32993
SAN ANTONIO, TX 78216
Tel: (512) 494-9336
CH,CS,E,MS

Utah
Hewlett-Packard Co.
3530 W. 2100 South
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700
A,CH,CS,E,MS

Virginia
Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
P.O. Box 9669
RICHMOND, VA 23228
Tel: (804) 747-7750
A,CH,CS,E,MS

Washington
Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
Suite A
708 North Argonne Road
SPOKANE, WA 99212
Tel: (509) 922-7000
CH,CS

West Virginia
Hewlett-Packard Co.
4604 MacCorkle Ave.
P.O. Box 4297
CHARLESTON, WV 25304
Tel: (304) 925-0492
A,MS

Wisconsin
Hewlett-Packard Co.
150 S. Sunny Slope Road
BROOKFIELD, WI 53005
Tel: (414) 784-8800
A,CH,CS,E*,MP

URUGUAY
Pablo Ferrando S.A.C. e I.
Avenida Italia 2877
Casilla de Correo 370
MONTEVIDEO
Tel: 80-2586
Telex: Public Booth 901
A,CM,E,M

VENEZUELA
Hewlett-Packard de Venezuela C.A.
3RA Transversal Los Ruices Norte
Edificio Segre 1, 2 & 3
Apartado 50933
CARACAS 1071
Tel: 239-4133
Telex: 251046 HEWPACK
A,CH,CS,E,MS,P

Hewlett-Packard de Venezuela C.A.
Calle-72-Entre 3H y 3Y, No. 3H-40
Edificio Ada-Evelyn, Local B
Apartado 2646
4001, **MARACAIBO**, Estado Zulia
Tel: (061) 80.304
C,E*

Hewlett-Packard de Venezuela C.A.
Calle Vargas Rondon
Edificio Seguros Carabobo, Piso 10
VALENCIA
Tel: (041) 51 385
CH,CS,P

Bioelectronica Medica C.A.
Calle Buen Pastor
Edif. Cota Mil-Piso 2 y Semi Sotano 1
Boleita Norte
Apartado 50710 ***CARACAS*** *1050A*
Tel: 239 84 41
Telex: 26518

ZIMBABWE
Field Technical Sales
45 Kelvin Road, North
P.B. 3458
SALISBURY
Tel: 705 231
Telex: 4-122 RH
C,E,M,P

July 1983 5952-6900

Indicates main office

HP distributors are printed in italics.