HP 82000 IC Evaluation System

HP-IB Command Reference

Models D50, D100, D200 and D400

SERIAL NUMBERS

This manual affects all systems with software revisions up to 2.0.0

HEWLETT PACKARD

Legal Information

Notice

The information in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD
TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or
direct, indirect, special, incidental or consequential damages in connection with
the furnishing, performance, or use of this manual.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard
product and replacement parts can be obtained from your local Sales and
Service Office.

Printing History

New editions of this manual will incorporate all material updates since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.
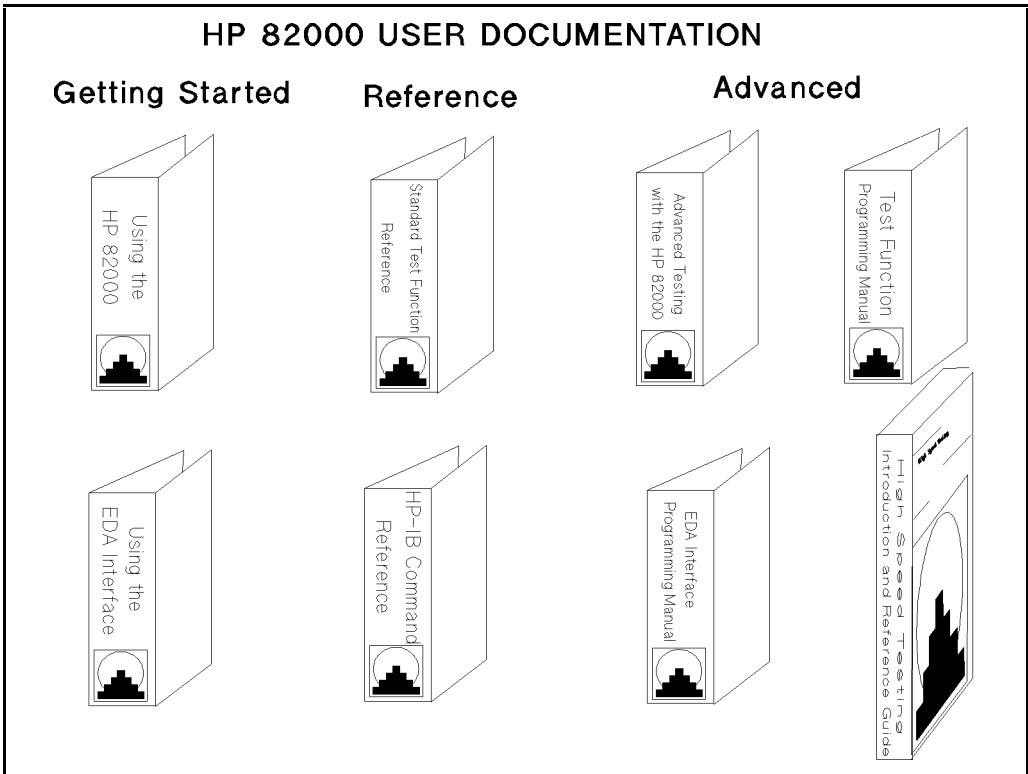
May 1989

Revision 1.0

October 1990

Revision 2.0 (Software versions up to 2.0.0)

Documentation Map

The following figure shows the User manuals available for the HP 82000 IC Evaluation System.

HP 82000 USER DOCUMENTATION

Getting Started   **Reference**   **Advanced**

Using the HP 82000

Standard Test Function Reference

Advanced Testing with the HP 82000

Test Function Programming Manual

Using the EDA Interface

HP-IB Command Reference

EDA Interface Programming Manual

High Speed Testing Introduction and Reference Guide

Installation and Service Documentation:

Site Preparation and Planning Guide

Installing the HP 82000

Maintaining the HP 82000

Servicing the HP 82000

Preface

Purpose

The purpose of this manual is to describe how to use HP-IB commands to program the HP 82000 IC Evaluation System. The manual does not aim to provide a description of HP-IB programming or the theory of the HP-IB interface.

Audience

This manual is intended for experienced users of the HP 82000, who want to develop test programs in BASIC or C, or write test functions in C.

The subject of using the HP-IB commands from a BASIC or C program is described in the manual *Advanced Testing with the HP 82000*.

Contents

A.   ERRS? Error Codes

B.   Test Function Error Codes

Index

Figures

Tables

1

Introduction

The HP-IB commands are the firmware instructions used by the Programming Work Station (PWS) to control the HP 82000 hardware.

You can include these commands as instructions within programs written in C or BASIC. In this way, you can combine sequences of HP-IB commands to form your own tests.

The HP-IB commands provide firmware level instructions for:

- configuring pins;
- setting the timing and signal formats;
- setting the levels;
- setting up vector data;
- controlling and programming the vector sequencer;
- programming the utility lines;
- setting up and executing test measurements;
- determining test results;
- performing diagnostics;
- calibrating the system;
- operating the Device Power Supply (DPS);
- handling errors;
- reporting the hardware status;
- and synchronizing commands.

The HP-IB commands for performing all of these tasks are described in this manual.

Hardware - Software Interaction

While you are using the HP 82000 from its interactive software, the interface between the Programming Workstation (PWS) and the hardware operates transparently. The test data and setup data that you enter in the software windows is downloaded to the hardware by interactive software, and you see the test results either as result values, or as an error message in the Report Window (if you have tried to do something that the software will not allow).

What actually happens when you use the interactive software, is that a series of HP-IB commands is sent from the PWS to the hardware. This series of commands may either cause a specific action in the hardware, or they may be sent in the contents of a setup file (such as level or timing setup files, or calibration files). Further HP-IB commands are used to interrogate the hardware, and this information is used to update the software windows.

Interpreting Firmware Instructions

An HP-UX process, running within the PWS, converts the commands that you enter in the software windows, into firmware level instructions which are transmitted via the HP-IB link to the hardware. Similarly, electrical data signals received from the HP-IB link are interpreted for the software windows by this same process.

This software process is called the *Message Concentrator and Distributor* (MCD).

As well as communicating with the tester hardware, the MCD process is also responsible for controlling the Device Power Supply (DPS) and High Speed Width Generator (HSWG) units, and for offline programming.

Figure 1-1 illustrates the interaction between the HP 82000 hardware and the PWS.
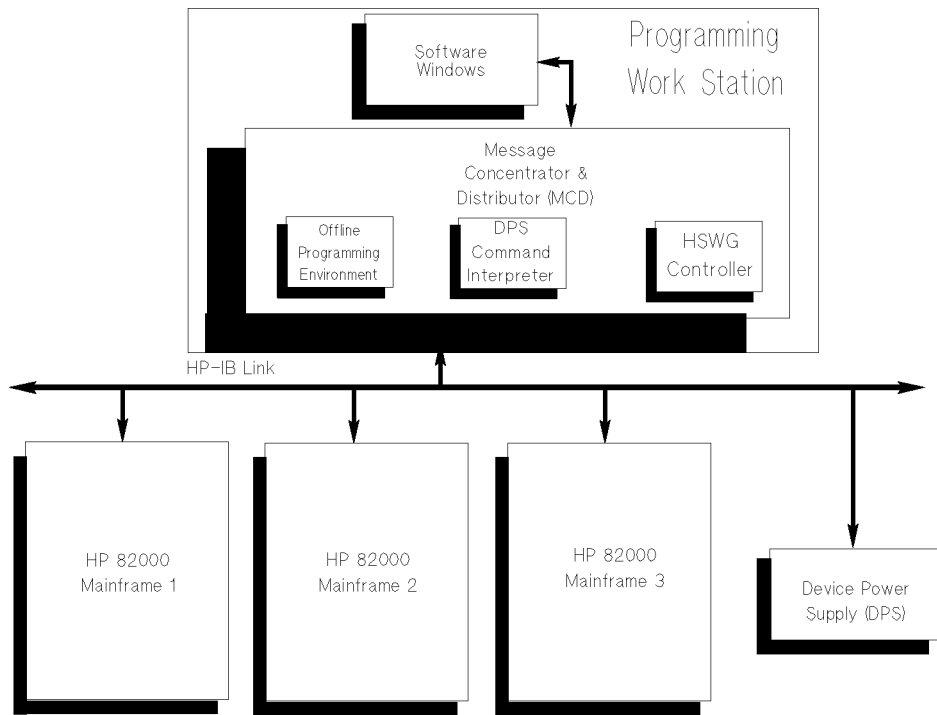
Figure 1-1. HP 82000 System Model

Notation of HP-IB Commands

Each HP-IB command is represented by a mnemonic, comprising up to four upper-case letters. When you enter an HP-IB command (either in a C or BASIC program, or by using the `hpt` utility to execute commands directly), the PWS expects:

■ the mnemonic name of the HP-IB command;

  followed by

■ a string of parameters, each separated by a comma (,).

Queries

HP-IB Commands which retrieve information from the hardware, are called *Queries*. You can identify an HP-IB Query by the question mark (?) at the end of the mnemonic name.

Some of the HP-IB commands form complementary pairs, comprising an instruction for setting the hardware state, and a corresponding query for finding the present hardware state.

For example, the command VEFA sets the first free vector address; the query VEFA? reads the present first free vector address.

Syntax

The second half of this manual lists all the HP-IB commands used by the HP 82000, in alphabetical order, and gives a complete syntax description, with a full list of parameters, for each command.

Using the `hpt` Utility

This is a utility program, provided as part of the HP 82000 software, which allows you to execute HP-IB commands *directly* from an HP-UX shell window, while the sytem software is running.

To start the hpt utility, at the HP-UX command line on the PWS, type in:

`$ /hp82000/pws/bin/hpt`

The PWS responds with the prompt:

`@`

To execute an HP-IB command, type in the command mnemonic and its parameters, followed by ⌈Return⌋. The PWS displays any values returned by the HP-IB command, and then gives the @ prompt again.

To exit the hpt utility, press ⌈Break⌋ on the keyboard. This returns you to the HP-UX command line.

2

Configuration Setup Commands

In the HP 82000 software, these commands are used in the Pin Configuration
Setup window. Configuration setup files are stored in the `configuration`
directory of the device in use. There are six commands for setting up tester
channels and four commands for setting up device power supply pins. The
commands are:

*OPT?	lists installed tester hardware.
DFPN	sets a pin name to a system channel.
DFPN?	lists all the pins defined with DFPN commands.
UDEF	removes a defined channel from the current configuration. This is the opposite of the DFPN command.
CONF	sets up the operation mode for a pin.
CONF?	lists the operation mode for a pin.

DFPS	sets a pin name to a power supply.
DFPS?	lists all the pins defined as power supply pins.
UDPS	removes a defined power supply from the current configuration. This is the opposite of the DFPS command.
PSCF?	reports the availability of power supplies

Listing Installed Boards

The *OPT? command lists all the boards installed in the specified mainframe. The mainframe can be either 0, 1, or 2 for the master, slave1 and slave2 respectively.

The command returns a string consisting of 18 fields separated by commas. These fields, which correspond to slot numbers 1 to 18 in a mainframe, contain either the ID code of an installed board or zero (0) for an empty slot.

The ID code is a decimal representation of an eight bit code (11 bit for slot 2).

Note   The first two bits (D0 and D1) indicate the hardware revision letter of the board. You can confirm the revision letter by checking the legend printed on the board.

Table 2-1. ID Code Definitions Bits 0 and 1

D1	D0	Representation
0	0	Not Used
0	1	Revision C
1	1	Revision D or E
1	0	Not Used

Table 2-2. ID Code Definitions Bits 2 and 3

D3	D2	D100, D200 or D400	D50
0	0	32K I/O Board	16K I/O board
0	1	128K I/O Board	64K I/O board
1	0	512K I/O Board	256K I/O Board
1	1	1M I/O Board	not used
0	0	PMU Board	PMU Board
0	1	512K Clock Board	512K Clock Board
1	0	2M Clock Board	2M Clock Board
0	0	64K Sequencer Board	32K Sequencer Board
0	1	256K Sequencer Board	128K Sequencer Board

Table 2-3. ID Code Definitions Bits 4 to 7

D7	D6	D5	D4	Representation
0	0	0	0	Clock/Microprocessor Board
0	0	0	1	Sequencer Board
0	0	1	0	100 MHz I/O Board
0	0	1	1	200 MHz I/O Board
0	1	0	0	50 MHz I/O Board
0	1	0	1	PMU Board
0	1	1	0	400 MHz I/O Board
0	1	1	1	Master Only Sequencer Board

The code for slot 2 is extended by two bits (D9, D8) which are used to describe the function of the sequencer in multi-mainframe mode. The meanings of these bits are:

Table 2-4. ID Code Definitions Bits 8 and 9

D10	D9	D8	Representation
0	0	0	Master
0	0	1	Slave 1
0	1	0	Slave 2
0	1	1	Slave 3
1	0	0	Slave 4

An example output for this command is:

`*OPT 5,273,49,49,49,49,49,49,49,49,49,49,49,49,49,49,81`

which indicates:

5	Revision C Clock Board
273	Revision C Sequencer Board in slave 1 mode
49	15 Revision C 200 MHz IO Boards
81	Revision C PMU Board

Defining Pin Names

The DFPN command is used to assign a pin name to a system channel. Normally that system software refers to system channels using these pin names and not actual channel numbers. The DFPN? command returns a list of system channels that have been assigned pin names.

Syntax

```
DFPN channel, [pin-number], pin-list
```

```
DFPN? pin-list
```

channel   is a five digit number, which describes the channel to be used for a pin. The five digits of the number fall into three groups in the format mbbcc.

- ■ m - the first digit specifies the mainframe.
- ■ bb - the second and third digits specify the channel board. A channel board is identified by the logical slot in which it is

installed. Slots for channel boards are numbered from 01 to 16.

- ■ `cc` - the fourth and fifth digits specify the channel on the board. A board may have 8 or 16 channels. They are numbered from 01 to 16.

These numbers are also printed on the DUT Board.

pin-number   is the physical pin number of the DUT and is included for comment purpose only. It is required by the system software.

pin-list   contains the name of one signal that is to be mapped to the system channel. This command can only take one pin name as its argument.

The DFPN? commands returns a list of defined pin names.

Power-On Defaults

At Power-On there are no pins defined.

Removing Pin Name Definitions

The UDEF command removes a definition made with the DFPN command. All the setups for the pin remain in memory and will come into effect again if the channel is redefined by another DFPN command. However, if the environment has changed between the UDEF and the DFPN commands in such a way that current settings are invalid, they will be silently adjusted to make them correct.

The pin configuration will always be forgotten after a UDEF command. If the pin channel is configured again, the operation mode for the pin will be set to OFF (see CONF below).

Setting Pin Capabilities

The CONF command is used to set the direction and termination, operating mode and scan path mode of pins that have been previously defined with a DFPN command, and also allows the definition of a global word mask. The CONF? query command returns mode information for the required pins.

Syntax

CONF [io-mode], [operation-mode], [scan-path-mode], pin-list

CONF? pin-list

`io-mode`

This parameter determines the io capabilities of the pin. Table 2-5 shows the availability of the io modes with each operating mode.

I The pin is considered to be a DUT input pin. Only the resources related to the channel driver are enabled (drive data, drive level, drive formats, and so on). All resources related to the channel receiver are not accessible and are disabled. The receiver will not contribute to any test results.

IX (D400 only). The pin is a DUT input pin, but is driven by a High Speed Width Generator (HSWG) channel. The driver side of the I/O board is routed through the HSWG, and the receiver resources are disconnected. This io mode corresponds to the `ihs` pin type, in the interactive software windows.

O The pin is considered to be a DUT output pin. Only the resources related to the channel receiver are enabled (expected data, receive data, compare edges, receive threshold, and so on). All resources related to the channel driver are not accessible and are disabled. The driver will be in high impedance state all the time.

IO The pin is considered to be a DUT bidirectional pin. Both driver and receiver resources are enabled. This mode is not available with 400 MHz I/O boards.

IOL This is the same as IO but threshold adjustments are made assuming that the DUT output is terminated by the low level of the driver. This mode is not available with 400 MHz I/O boards.

IOH This is the same as IO but threshold adjustments are made assuming that
     the DUT output is terminated by the high level of the driver. This mode
     is not available with 400 MHz I/O boards.
TERM The pin is considered to be a DUT output pin, but in contrast to the
     output mode (O), it is terminated by an active load. The behaviour of
     the receiver resources are the same as in output mode but the driver
     will force a constant voltage level. The driver setup is also accessible.
     This io mode corresponds to the ot pin type, in the interactive software
     windows.
NC   The pin is temporarily considered to be disconnected from the DUT. The
     resources of both driver and receiver are accessible, but the DUT pin is
     always physically disconnected from the resources (the final relay will
     not be closed). The receiver will not contribute to test results. This mode
     is not available with 400 MHz I/O boards.
DC   The pin is considered to be available for DC-measurements only. No
     AC-resources (driver/receiver) are accessible.
OFF The pin is not configured at all. It is not available for any measurements.

operation-mode

This determines the internal operation mode of the channel logic. Table 2-5
shows the operating modes that are available for each I/O board type, and the io
modes that can be used.

Table 2-5. Availability of Operating Modes and IO Modes

Operating Mode	I/O Board	IO Modes available
STD	50 MHz, 100 MHz and 200 MHz	I, O, IO, TERM, IOH, IOL, DC, NC, OFF
FD	50 MHz and 200 MHz	I, O, IO, TERM, IOH, IOL, DC, NC, OFF
MUX	50 MHz and 200 MHz	I, O, IO, TERM, IOH, IOL, DC, NC, OFF
FQ2	400 MHz	I, IX, O, TERM, DC, OFF
FQ	400 MHz	I, IX, O, TERM, DC, OFF
FQM	400 MHz	O, TERM, DC, OFF

STD This is the standard mode of operation for D50, D100 and D200 systems
     and provides a full range of signal formats. In the driver part, even bits

determine data and odd bits determine high impedance (tristate). This mode corresponds to:

25 mode	in D50 systems
100 mode	in D100 and D200 systems

FD   This mode doubles the frequency but reduces the available formats. In this case, even and odd bits in the driver determine DNRZ. This mode corresponds to:

50 mode	in D50 systems
200 mode	in D200 systems

MUX   This mode doubles the frequency by combining the resources of two consecutive channels. If a pin is configured in this mode, its co-channel cannot be accessed directly. The co-channel is automatically set up by setting up the master channel. This mode corresponds to:

50MUX mode	in D50 systems
200MUX mode	in D200 systems

FQ2   This mode provides the maximum no of signal format types for 400 MHz I/O boards. This mode corresponds to 200COM mode in the Setup window.

FQ   This mode provides 400 MHz drive and receive signals from a 400 MHz I/O board. This channel can be routed through an HSWG channel to provide an enhanced range of signal formats. This mode corresponds to 400 mode in the Setup window.

FQM   This mode combines the receiver resources of two 400 MHz channels to provide a dual threshold edge compare facility at 400 MHz operating frequency. This mode allows the channel to operate as an output pin only, and corresponds to 400MUX mode in the Setup window.

scan-path-mode

This determines the behavior of the pin during a serial scan vector and allows the definition of a global word mask. This parameter can be set independently of all other parameters.

- P A channel in this mode keeps its data during a serial scan. Its vector memory may be used for a channel in serial mode.
- S A channel in this mode serializes data from the vector memory of other channels during a serial scan. (Serial scan mode is not available with 400 MHz I/O boards.)
- W A channel in this mode is used as a global word mask. Serializing is broken at this channel.

`pin-list`

This is the name of the pin or pins to be configured or listed. The wildcard symbol @ can be used to configure or list all pins defined with a DFPN command

The CONF? query lists the current `io-mode`, `operation-mode` and `scan-path-mode` for all the pins in `pin-list`.

Setting up Power Supply Pins

The DFPS and DFPS? commands are used to setup and list the names used for the available power supply outputs. UDPS is used to undefine a power supply. PSCF? is a special command to read the type and channel count of an installed power supply.

```
DFPS channel, sign, pin

DFPS? pinlist

UDPS pinlist

PSCF?
```

where

`channel`	is a two digit figure which indicates the Device Power Supply (DPS) number and the DPS output channel used. For example, DPS1 output channel 3 is indicated by the number 13 in this parameter.
`sign`	is the polarity of the power supply output. It can be either POS or NEG.
`pin`	is the name to be used for the power supply channel.
	and for the DFPS? and UDPS? queries:
`pinlist`	is the name to be used for the power supply being defined or undefined. For UDPS, the wildcard symbol '@' is used to mean all power supply pins.

The PSCF? query command returns the string

```
PSCF dps-name, nr-of-channels
```

where

`dps_name`	is the model number of the power supply (for example, HP6624A). Power supplies supported by the system software, are

- HP 6621A to HP 6626A
- HP 6628A and HP 6629A
- and HP 6632A to HP 6634A

`nr_of_channels`	is the number of outputs that can be used

Power-On Defaults

At power-on the following default settings are made:

`io-mode` is OFF.
`operation-mode` is STD (for 50 MHz, 100 MHz and 200 MHz boards)
or FQ (for 400 MHz boards).
`scan-path-mode` is P.
no power supply pins are defined

Errors

An error will be generated if:

- `pin-list` in the DFPN command contains more than one pin.
- there are parameter range errors
- the tester is not idle
- a `Channel` is not available.
- the channel to be defined is in use.
- the pin to be defined is already in use.
- the resources that are being programmed are not installed.
- the resources being programmed are already in use.
- there is no device power supply installed in the system

Examples

Example 1

```
UDEF ( @ )
DFPN 10101, "001", ( reset )
CONF IO, MUX, P, ( reset )
```

In this example, the first line ensures that no pins are already defined. The second line defines channel 1 on board 1 in mainframe 1 to be the pin RESET. The third line configures RESET as an I/O multiplex pin with parallel scan behaviour.

```
UDPS (@)
DFPS 11, POS, (VCC)
```

The first line undefines all power supply pins. Channel 1 of the DPS1 is defined to output a positive voltage that can be referenced by the name VCC.

Example 2

This is the configuration file configuration/c_d200 in the device directory of the MC10H136 demo device.

```
hp82000,configuration,0.1
DFPN 10104,"13",(Clock)
DFPN 10201,"12",(D0)
DFPN 10202,"11",(D1)
DFPN 10206,"6",(D2)
DFPN 10205,"5",(D3)
DFPN 10203,"10",(NCin)
DFPN 10108,"4",(NCout)
DFPN 10101,"14",(Q0)
DFPN 10102,"15",(Q1)
DFPN 10106,"2",(Q2)
DFPN 10107,"3",(Q3)
DFPN 10204,"9",(S1)
DFPN 10207,"7",(S2)
DFPN 10105,"1",(Vcc1)
CONF I,STD,P,(Clock)
CONF I,STD,P,(D0)
CONF I,STD,P,(D1)
```

```
CONF I,STD,P,(D2)
CONF I,STD,P,(D3)
CONF I,STD,P,(NCin)
CONF TERM,STD,P,(NCout)
CONF TERM,STD,P,(Q0)
CONF TERM,STD,P,(Q1)
CONF TERM,STD,P,(Q2)
CONF TERM,STD,P,(Q3)
CONF I,STD,P,(S1)
CONF I,STD,P,(S2)
CONF DC,STD,P,(Vcc1)
DFPS 11,neg,(Vee)
```

3

Level Setup Commands

This chapter describes the HP-IB commands that are used to set driver and receiver levels, and power supply output levels.

Level Commands

A tester channel consists of a driver and receiver part. The driver part needs to know the drive signal low and high levels that are to be output for a 0 and 1 vector and the receivers require the compare high and low thresholds that are to be used when making measurements on DUT outputs.

To specify whether the Level Setup Screen sets the timing reference automatically,the DRLM and DRLM? queries are provided.

There are two commands to specify the driver levels and receiver thresholds, DRLX and RCLV respectively. Two query commands DRLX? and RCLV? can be used to read the current settings from the hardware.

If a device power supply is installed in the system, it is possible to program and read the output levels using the PSLV and PSLV? commands.

Default values are set after power on and after an HP-IB "reset" (*RST) command and remain valid until they are overwritten by downloading a Level Setup or Pin Configuration file. In the PWS software, these levels are set in the "Level Setup" window.

Specifying the Timing Reference

Syntax

DRLM $\begin{bmatrix} \texttt{L50} \\ \texttt{FRZN} \end{bmatrix}$

DRLM?

This command specifies that the Level Setup screen sets the timing reference to the 50% swing point (L50) or to a fixed level (FRZN).

Using L50 sets the reference to the difference of the programmed high and low levels, FRZN makes this setting dependent on the nominal-swing setting of the DRLX command.

Note	Setting the nominal-swing parameter in the DRLX command to anything other than -1 will result in FRZN mode being selected.

Driver Level Settings

Syntax

DRLX [logic-0-level], [logic-1-level], [nominal-swing], pinlist

DRLX? pinlist

The following table shows the value ranges and default (power-on) values for the driver level settings.

Table 3-1. Drive Level Settings

Parameter	Implicit Unit	Range (D100, D200 & D400)	Range (D50)	Range (ihs pins)	Default
logic-0-level	mV	−4000..7800	−2000..6500	−4000..4800	−1700
logic-1-level	mV	−3800..8000	−2500..7000	−3800..5000	−800
nominal swing	mV	0..8000	0..7000	0..5000	

There are swing limitations which mean that the minimum swing between logic-0-level and logic-1-level is 200 mV (500 mV for 50 MHz boards) and the maximum permitted swing is :

- 7000 mV for the 50 MHz boards;
- 5000 mV for IX channels (400 MHz channels connected to the HSWG)
- 8000 mV for 100 MHz boards, 200 MHz boards and 400 MHz boards (without HSWG)

Violations of these minimum or maximum swing values will cause a semantic error.

The optional parameters, logic-0-level or logic-1-level can be omitted. If this occurs, the firmware can auto-correct this parameter if necessary. This may be needed to ensure that programmed values are valid.

When a channel is configured to active termination (TERM), the drive part of the channel is forced to a static low level. The single level is programmed via the first parameter (logic-0-level). The second parameter is ignored and will be forced to an auto-corrected value depending on the required termination level. Querying the settings of such a channel returns the single level as the first parameter, the second parameter will be omitted.

The parameter nominal-swing specifies the level swing to be used for determining the 50% reference point for driver timing. The special value -1 automatically sets the swing to the difference between the logic-1 and logic-0 levels.

Errors

Errors will be generated for:

- low level out of range
- high level out of range
- swing violation
- attempting to program a non-applicable pin (configured as O, DC, or OFF)

Warnings

Warnings will be generated if:

- Auto correction occurs
- Level programming exceeds the hardware specs

Note	Setting the `nominal-swing` parameter in the DRLX command to anything other than `-1` will result in `FRZN` mode being selected in the DRLM command..

Driver Level Settings

Note	This command is only documented to support older software releases. The use of this command to set up driver levels is discouraged.

Syntax

DRLV [logic-0-level], [logic-1-level], pinlist

DRLV? pinlist

The following table shows the value ranges and default (power-on) values for the driver level settings.

Table 3-2. Drive Level Settings

Parameter	Implicit Unit	Range (200 MHz)	Range (50 MHz)	Default Value
logic-0-level	mV	-4000..7800	-2000..6500	-1700
logic-1-level	mV	-3800..8000	-2500..7000	- 800

There are swing limitations which mean that the minimum swing between logic-0-level and logic-1-level is 200 mV (500 mV for 50 MHz boards) and the maximum permitted swing is 8000 mV (7000 mV for the 50 MHz boards). Violations of these minimum or maximum swing values will cause a semantic error.

The optional parameters, logic-0-level or logic-1-level can be omitted. If this occurs, the firmware can auto-correct this parameter if it needs to. This may be necessary to ensure that programmed values are valid.

When a channel is configured to active termination (TERM), the drive part of the channel is forced to a static low level. The single level is programmed via the first parameter (logic-0-level). The second parameter is ignored and will be forced to an auto-corrected value depending on the required termination level. Querying the settings of such a channel returns the single level as the first parameter, the second parameter will be omitted.

Errors

Errors will be generated for:

- low level out of range
- high level out of range
- swing violation
- attempting to program a non-applicable pin (configured as O, DC, or OFF)

Warnings

Warnings will be generated if:

- Auto correction occurs
- Level programming exceeds the hardware specs

Receiver Threshold Commands

Syntax

RCLV [logic-0-threshold], [logic-1-threshold], pinlist

RCLV? pinlist

Table 3-3. Receiver Threshold Settings

Parameter	Implicit Unit	Range (D100,D200,D400)	Range (D50)	Default Value
logic-0-threshold	mV	−4000..7994	−2000..6950	−1.470
logic-1-threshold	mV	−3994..8000	−2010..7000	−1110

As in the driver level commands, there is a swing limitation such that there must be a minimum swing between logic-0-threshold and logic-1-threshold. The minimum swing limitation depends on the type of pin defined. The minimum threshold swing is

- 6000 mV for 100 MHz, 200 MHz and 400 MHz boards, and
- 10000 mV for 50 MHz boards.

In the case of a channel in FD or FQ operating mode, the required single threshold is set by the logic-1-threshold parameter. The first parameter, logic-0-threshold is ignored and silently auto-corrected to a value dependent on the given threshold. Querying such a pin will return the single threshold in the logic-1-threshold parameter and the first parameter logic-0-threshold will be omitted. Table 3-4 shows the logic levels that can be set for each Operating Mode.

Table 3-4. Receiver Logic Levels

Operating Mode	Low Level	High Level
FD, FQ	not programmable	threshold for single level compare
STD, MUX, FQ2, FQM	lower threshold for dual level compare	upper threshold for dual level compare

Errors

Errors will be generated for:

- low level out of range
- high level out of range
- swing violations
- attempting to program non-applicable pins (configured as I, DC, OFF)

Warnings

A warning will be generated if:

- auto-correction has occurred
- level programming exceeds the hardware specifications

Setting Power Supply Levels

The PSLV and PSLV? commands are used to set and query the current device power supply (DPS) output voltage and current limit settings. Additionally, it is also possible to specify a setup time, which causes the system to wait until the next command is executed, and an off-state impedance. The off-state impedance is used to set the DPS outputs when the system is in the disconnected state.

```
PSLV [volt], [off-imp], [setup-time], pinlist

PSLV? pinlist
```

where

volt   is a real number in the range 0 to 40 to specify the output
                voltage. The sign (positive or negative) of the `volt` parameter
                must be the same as that in the DFPS command used to define
                the power supply channel.

Note   If the installed power supply is capable of supplying more than
                40 Volt, the system will limit its output to 40 Volt.

---

off-imp   either `HIZ` or `LOZ` to specify high or low impedance in
                disconnect state. If `HIZ`is selected, tha channel will be
                programmed to the minimum possible current limit for the
                power supply. If `LOZ` is used, the power supply current limit
                will remain at the same lavel as in the active state.
setup-time   is an integer with the implied units milliseconds. It specifies the
                delay between setting the power supply and starting a test.

Defaults

At power-on and after a reset, all power supply outputs will be set to zero (0)
Volt, the minimum current limit programmable for the power supply (0.004 ..
0.15 A), HIZ, and a zero (0) millisecond setup time.

Errors

An error will be generated if

■ there are any syntax errors
■ the polarity of the voltage level is different from that in the DFPS command
  for the pin
■ the programmed voltage is higher than the maximum output voltage of the
  power supply.
■ the programmed current limit is greater than the power supply maximum
  output current
■ the range combination `volt` and `curr` is not allowed by the installed DPS

Examples

```
DRLX 0, 5000,, (A0, A1)
```

This example forces the address channels A0 and A1 (I type pins) to output 0 V if a digital 0 (low) and 5.0 V if a digital 1 (high) must be provided to the DUT.

```
DRLX 0,,, (TC)
```

If the DUT pin TC is configured as TERM (active termination), logic-1-level would be ignored anyway so it's a good idea to omit this parameter, only the logic-0-level is of interest.

```
RCLV 400, 4800, (D0, D1)
```

All levels received at D0 and D1 (O pins) lower than 0.4 Volt will be recognized as a digital 0 (low) and those greater than 4.8 Volt as a logical 1 (high). A stable level between these two values will be treated as intermediate.

```
PSLV 5, 1, 100, (VCC)
```

The power supply defined with the name VCC is set to output 5 Volt at a maximum current of 1 Amp. The test will be started 100 milliseconds after this command is executed.

The following listing is the level setup file used for the 10H136 demo device.

```
hp82000,level,0.1
PSLV -5.200,0.500000,hiz,15,(Vee)
PSLV -2.000,1.000000,hiz,15,(LPS)
DRLX -1800,-900,900,(Clock,D0,D1,D2,D3)
DRLX -1800,-900,2000,(NCin)
DRLX -2000,,,(NCout,Q0,Q1,Q2,Q3)
DRLX -1800,-900,900,(S1,S2,WORDMASK)
DRLM FRZN
RCLV -1260,-1250,(NCout,Q0,Q1,Q2,Q3)
RCLV 1990,2000,(WORDMASK)
```

4

Format and Timing Setup Commands

The commands covered in this section allow the system clock and driver and receiver timing and format to be set up. These can also be set up using the Timing Setup window of the PWS.

There are seven commands for setting up the system clock:

SCLK	set up system clock reference and period
SCLK?	retrieve system clock reference and period
SNCC	setup synchronization input behavior
SNCC?	retrieve synchronization input behavior
SYNC	enable or disable synchronization
SYNC?	retrieve synchronization mode
VCLK?	check setting of clock period for possible auto-corrections

Note   In multiple mainframe configurations, only the master mainframe responds to the queries SNCC?, SYNC?, and SCLK?.

There are four commands used for setting channel format and timing:

DRTM	sets the timing and format of driver outputs.
DRTM?	reads back the current timing and format settings of driver outputs.
RCTM	sets the timing and format of receiver inputs.
RCTM?	reads back the current timing and format settings of receiver inputs.

Note   All timing setup commands temporarily change the state of the sequencer to OFF, if it is not already in this state.

Setting Up the System Clock

The SCLK command is used to set the system vector rate and optionally synchronize this to an external input.

The SNCC command defines the synchronization circuitry behavior. It allows setting of the synchronization comparator slope sensitivity, threshold, and input impedance. Additionally, a delay can be specified which defines the time, the internal clock will be delayed against the external clock input signal.

The SYNC command allows the internal clock to be synchronized to an external signal. Synchronization takes place each time the internal clock is turned on.

The VCLK? query is used to check whether a period setting will cause edge delay auto-corrections to occur.

Setting the System Clock Period

```
SCLK [clock-source], [period], [fraction] SCLK?
```

clock-   can be INT or EXT to select between the internal reference
source   clock or an external clock connected to the Ext Clock input.
period   is a real number indicating the clock period in the ranges shown
              in the table below. This is in the implied units nanoseconds, so
              setting SCLK INT, 80 , will result in a vector rate of 12.5 MHz.
fraction   If you have specified an external clock, this parameter is used
              to multiply or divide the external clock signal by a fixed value.
              Allowed values for fraction are F1T8, F1T4, F1T2, F1T1,
              F2T1, F4T1, F8T1 which result in ratios from 1/8 to 8×.

Period can be programmed as shown in the table below:

Table 4-1. System Clock Period Settings

Range	Parameter	Resolution
10ns .. 99.9ns	10 .. 99.9	100ps
100ns .. 999ns	100 .. 999	1ns
1us .. 9.99us	1000 .. 9990	10ns
10us .. 99.9us	10000 .. 99900	100ns

For an external clock source, `period` is restricted to the following ranges :

Table 4-2. System Clock Period Settings for External Source

Fraction	Range
F1T8	10ns .. 720ns
F1T4	10ns .. 1.44μs
F1T2	10ns .. 2.88μs
F1T1	10ns .. 5.76μs
F2T1	10ns .. 11.5μs
F4T1	10ns .. 23μs
F8T1	13.3ns .. 46μs

The system clock is available at the BNC connector J11 located on the clock board. It is a 50% duty cycle square wave with ECL levels (-1.7 V low and +0.8 V high), if properly terminated (50 Ω).

Each time you modify the system clock, you should verify, and, if necessary, modify all programmed channel edge delays. If an edge becomes incompatible with a newly programmed period this edge will be silently autocorrected.

The VCLK? query can be used to verify the clock period setting.

The `fraction` parameter will be ignored if `clock-source` is set to internal.

To synchronize the clock, either the internal reference oscillator (`clock-source` = INT) or an external reference signal (`clock-source` = EXT) may be used. If an external reference has been chosen, the frequency of the external clock source has to be specified by passing the external to pattern clock frequency ratio as one of the command parameters (`fraction`). This parameter will be ignored for internal clock programming. It comes from the formula

$$< fraction > = \frac{< period >}{T_{ext}} = \frac{f_{ext}}{f_{per}}$$

Synchronizing the System Clock

SNCC [delay], [slope], [level], [impedance]

SNCC?

SYNC mode

SYNC?

delay	is a positive real number used to set a delay between the internal clock and the external phase synchronization signal. This has the implied units nanoseconds and can be set to a maximum of 1 millisecond.
slope	sets the active edge of the external phase synchronization signal and can be either POS or NEG
level	sets the threshold level of the external phase sync. input in the range ± 10 V. It has the implied units millivolts.
impedance	sets the impedance of the external phase synch. input to 10 kΩ (R10K) or 50 Ω (R50). Note that selecting a 50 Ω input impedance implies a restricted level range of ± 5 V.
mode	can be ON or OFF depending on whether the internal clock should be synchronized

Using an external clock allows phase synchronization of the system clock to a second external signal (normally delivered by the DUT).

The SNCC command affects only the synchronization parameters. It does not affect the mode selected by the SYNC command. If, however, synchronization is active ("SYNC ON"), the passed parameters are in effect immediately after sending the command.

Synchronization is only supported, if used in conjunction with an external clock source. Any other value than OFF for the mode parameter will therefore cause an error if used in internal clock mode. Selecting the internal clock source while synchronization is in effect, causes synchronization to be turned off. While system clock synchronization stays in effect, the system clock will be synchronized each time it has to be turned on. An error will be reported, if synchronization fails. This condition is also reported in the Hardware Status Register.

The external clock delay in the SNCC command can be set to a maximum of 1 millisecond with a three-digit ±50 picosecond resolution.

The query commands SCLK?, SNCC?, and SYNC? respond with the current settings of the appropriate commands.

Checking the System Clock Setting

```
VCLK? period
```

System responds with VCLK flag, where,

period   see Setting the System Clock Period
flag   Set to 0 or 1 to indicate whether auto-correction will occur for the desired period setting.

Each time you modify the system clock, you should verify, and, if necessary, modify all programmed channel edge delays. If an edge becomes incompatible with a newly programmed period this edge will be silently autocorrected.

If flag is set to 1, at least one timing edge needs to be corrected if the clock period setting is used. This command is used in the User Interface to avoid erroneous period changes which will cause an entire timing setup to become invalid.

Each mainframe will answer this query with the results of the pins in that mainframe.

Power-On Defaults

At power-on, the following settings are made:

```
SCLK INT,10,
fraction = F1T1
SNCC 0,POS,0,R10K
SYNC OFF
```

Errors

Errors will be generated if:

- programmed values contain value range errors
- there is a mismatch of period and fraction parameter (external clock)
- Synchronization is set and the clock source is internal
- Synchronization failed

Setting Driver Format and Timing

The DRTM command is used to set the format of a signal and the timing of the leading and trailing edges.

Syntax

DRTM [drive-format], [leading-edge], [trailing-edge], pinlist

DRTM? pinlist

Drive-format	can be one of:
	■ DNRZ for delayed non-return to zero format
	■ RZ for return-to-zero format
	■ R1 for return-to-one format
	■ RC for return-to-complement format
	■ RI for return-to-inhibit format (50MHz only)
leading-edge	this is the time delay for the leading edge of the signal. It is a real number with the implied units nanoseconds.
trailing-edge	this is the time delay for the trailing edge of the signal. It is a real number with the implied units nanoseconds.

Allowed settings for different system operating modes are shown in the tables below. All edges are programmed with the implicit unit nanoseconds, for example, DRTM RZ, 0, 5.05, (Pin) will set the machine channel signal Pin so that the leading edge is at 0 ns delay and the trailing edge at 5050 ps.

Leading Edge Delays

The values given in the following tables give the *minimum* guaranteed range of values accepted by the system before an error message is given. In certain circumstances, the system will accept a leading edge delay greater than these values.

Table 4-3. Driver Format and Timing Settings—D50 Systems

Operating Mode	Signal Formats	Period	Maximum Leading Edge Delay
STD	DNRZ, RZ, R1, RC, RI	40 ns to 41.7 ns 41.7 ns to 181.3 ns 181.3 ns to 600 ns over 600 ns	1 cycle period (4 cycle periods)–125 ns 600 ns 1 cycle period
FD	DNRZ	40 ns to 41.7 ns 41.7 ns to 181.3 ns 181.3 ns to 600 ns over 600 ns	1 cycle period (4 cycle periods)–125 ns 600 ns 1 cycle period
MUX	DNRZ, RZ R1	40 ns to 50 ns 50 ns to 207.1 ns 207.1 ns to 600 ns over 600 ns	1 cycle period (3.5 cycle periods)–125 ns 600 ns 1 cycle period

Table 4-4.
Driver Format and Timing Settings—D100 and D200 Systems

Operating Mode	Signal Formats	Period	Maximum Leading Edge Delay
STD	DNRZ, RZ, R1, RC,	10 ns to 24.7 ns 24.7 ns to 168.5 ns 168.5 ns to 600 ns over 600 ns	1 cycle period (4 cycle periods)–74 ns 600 ns 1 cycle period
FD	DNRZ	10 ns to 24.7 ns 24.7 ns to 168.5 ns 168.5 ns to 600 ns over 600 ns	1 cycle period (4 cycle periods)–74 ns 600 ns 1 cycle period
MUX	DNRZ, RZ R1	10 ns to 29.6 ns 29.6 ns to 192.6 ns 192.6 ns to 600 ns over 600 ns	1 cycle period (3.5 cycle periods)–74 ns 600 ns 1 cycle period

Table 4-5.
Driver Format and Timing Settings—D400 Systems

Operating Mode	Signal Formats	Period	Maximum Leading Edge Delay
FQ2	DNRZ, RZ R1	10 ns to 27.6 ns 27.6 ns to 190.6 ns 190.6 ns to 600 ns over 600 ns	0.5 cycle period (3.5 cycle periods)–76 ns 600 ns 1 cycle period
FQ	DNRZ, (and RZ, if HSWG fitted)	10 ns to 27.6 ns 27.6 ns to 190.6 ns 190.6 ns to 600 ns over 600 ns	0.5 cycle period (3.5 cycle periods)–76 ns 600 ns 1 cycle period

Trailing Edge Delays

Table 4-6. Trailing Edge Delays—Driver Channels

Trailing Edge Limits	D50	D100, D200 and D400 (except IX pins)	IX type pins (see note)
Minimum Delay	LE + 3 ns	LE + 1 ns	LE + 1 ns
Maximum Delay	LE + (cycle - 3 ns)	LE + (cycle - 1 ns)	LE + (cycle - 9.95 ns)

(where: LE = Leading Edge delay)

Note   You can further reduce the pulse width produced by the HSWG channels (IX pins), by entering a trailing edge value of 0.5 ns. However, the tester is then outside its calibrated operating range and the true width of this pulse may be nearer 700 ps.

The following figures show the possible formats and their resulting states.
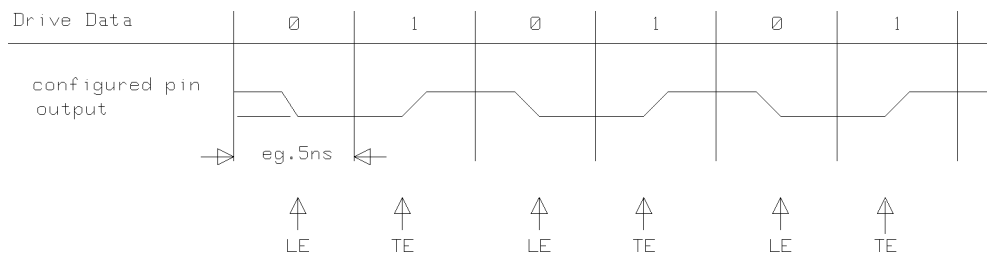
Figure 4-1. Example of DNRZ Format in 200 MHz Mode
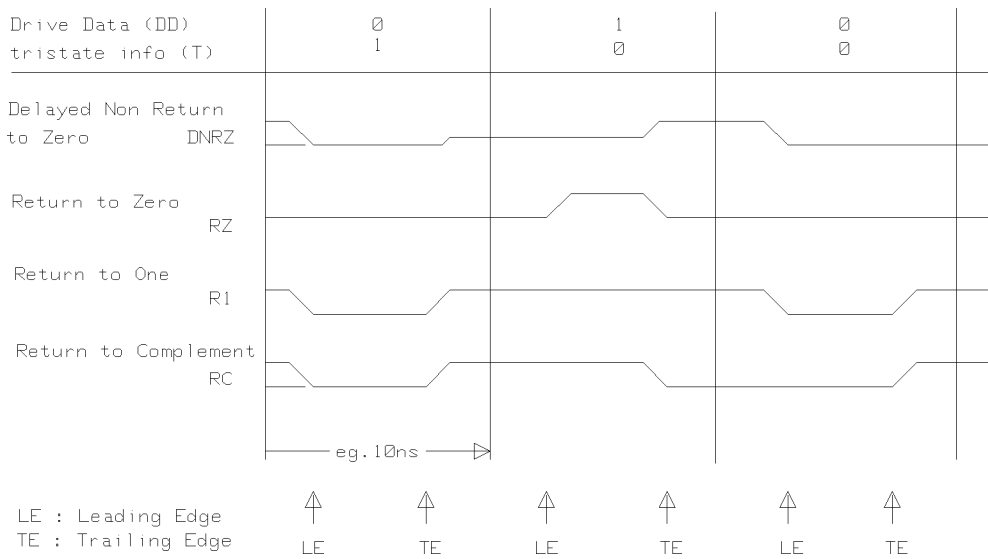
Figure 4-2. Example of Different Formats with Tristate
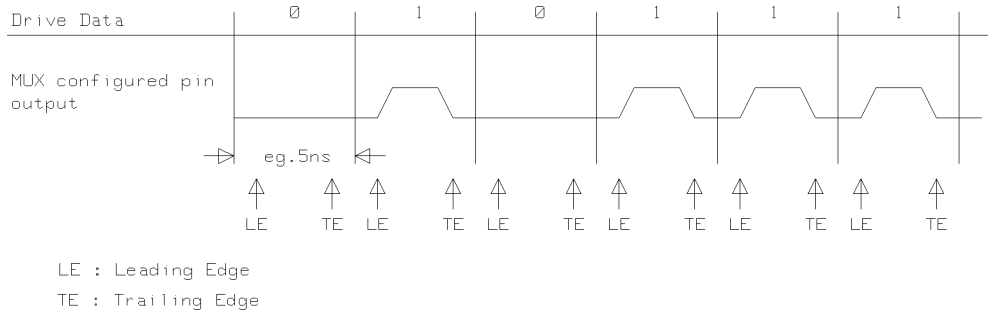
Figure 4-3. Example of RZ Format in 200MUX Mode

Programmed edges that do not have the necessary HW resources, for example, no tristate edge (TE) in the case of "FD" or "MUX" configured pins and DNRZ format are ignored and no warning message is generated.

All edge delays are rounded to the hardware resolution (50 ps for 200 MHz boards, 200 ps for 50 MHz boards).

Pins that are configured for serial scan are handled in the same way but you should remember that LSSD execution is possible only up to a frequency of 25MHz.

Contextual edge mismatching generated by using optional edge parameters is autocorrected in the following manner: If the value of a programmed edge exceeds the allowed range, the optional parameter is set to the maximum possible value. On the other hand, if a programmed value goes below the minimum, the optional parameter is autocorrected to the minimum possible value.

The DRTM? query command returns the format and timing setups of all the signal names in `pinlist`.

Power-On Defaults

At power-on, the following settings are made:

`DRTM DNRZ, 0, 0, all-pins`

Errors

An error will be generated if:

- there are any value range errors
- attempting to program non-applicable pins (configured as O, DC, OFF, or TERM)

Setting Receiver Format and Timing

The RCTM command is used to select the compare mode of receiver channels and program the delays for measurements.

Syntax

`RCTM [receive-format], [leading-edge], [trailing-edge], pinlist`

`RCTM? pinlist`

`receive-` `format`	can be either WIDW for window compare mode or EDGE for edge compare mode.
`leading-` `edge`	this is the time delay to the start of a window in window compare mode or the delay for the measurement in edge compare mode. It is a real number with the implied units nanoseconds.
`trailing-` `edge`	this is the time delay to the end of a window in window compare mode. It is a real number with the implied units nanoseconds.

Allowed settings for different system operating modes are shown in the table below. All edges are programmed with the implicit unit ns, for example, `RCTM WIDW, 1, 5.05, (Pin)` will set the receiver of the machine channel with signal name `Pin` so that the window compare will start after 1 ns and stop after 5050 picoseconds.

The limits on leading edge delays for receiver channels are the same as those available for driver channels. These limits are shown in Table 4-3 to Table 4-5.

The limits on trailing edge delays for receiver channels are shown in Table 4-7 below.

Table 4-7. Trailing Edge Delays—Receiver Channels

Trailing Edge Limits	D50	D100 and D200	D400
Minimum Delay	LE + 3 ns	LE + 2.5 ns	N/A
Maximum Delay	LE + (cycle - 3 ns)	LE + (cycle - 2.5 ns)	N/A

(where: LE = Leading Edge delay)

Programmed edges that do not have the necessary HW resources are ignored and no warning message is generated.

All edge delays are rounded to the hardware resolution (50 ps for 200 MHz boards, 200 ps for 50 MHz boards).

Contextual edge mismatching generated by using optional edge parameters is autocorrected in the following manner: If the value of a programmed edge

exceeds the allowed range, the optional parameter is set to the maximum possible value. On the other hand, if a programmed value goes below the minimum, the optional parameter is autocorrected to the minimum possible value.

The RCTM? query command returns the receiver format and timing setups of all the signal names in `pinlist`.

Power-On Defaults

At power-on, the following settings are made:

`RCTM WIDW, 0, 2, std-pins, mux-pins`

`RCTM EDGE, 0, , fd-pins`

Errors

An error will be generated if:

- there are any value range errors
- attempting to program non-applicable pins (configured as I, DC, or OFF)

Example

The following listing shows the format and timing setup for the MC10H136 demo device.

```
hp82000,timing,0.1
SCLK INT,20,
SNCC 0,POS,0,R10K
SYNC OFF
DRTM RZ,9,19,(Clock)
DRTM RZ,2,15,(D0)
DRTM RZ,2,15,(D1)
DRTM RZ,2,15,(D2)
DRTM RZ,2,15,(D3)
DRTM DNRZ,2,2,(NCin)
DRTM DNRZ,2,2,(S1)
DRTM DNRZ,2,2,(S2)
RCTM EDGE,18,,(NCout)
```

```
RCTM EDGE,18,,(Q0)
RCTM EDGE,18,,(Q1)
RCTM EDGE,18,,(Q2)
RCTM EDGE,18,,(Q3)
```

5

Vector Setup

The vector setup commands fall into two categories:

- static vector setup commands, and

- vector transfer commands.

The vector setup commands temporarily change the sequencer to the Break state if it is already running.

The static vector setup commands include commands to define driver channel static vectors such as masks and break vectors, and static receiver data vectors such as mask and expected data pattern vectors when the tester is in data acquisition (DA) mode.

The vector transfer commands are used to transfer large amounts of vector data via HP-IB, utilizing the DMA capabilities implemented on the tester's local microprocessor board. These commands differ from the general HP 82000 commands. Refer to the command descriptions for a details of these differences.

Note	It is not possible to create vector setup files using the vi editor in HP-UX. The representation of data in these files uses characters that cannot be edited or displayed by the vi editor.

Static Vector Setup Commands

There are four commands to set and read the channel static vectors. These are:

SDRV   defines the driver channel's pin mask and break vector.
SDRV?   reads the current settings of driver channel mask and break
               vectors
SREC   defines the receiver channel's pin mask and the expected data
               vector used in data acquisition (DA) mode.
SREC?   reads the current settings of receiver pin mask and expected
               data vectors to be used in data acquisition (DA) mode.

Syntax

    SDRV [dr-mask], [break-vec], pin-list

    SDRV? pin-list

    SREC [rec-mask], [aqi-exp-data], pin-list

    SREC? pin-list

Both commands affect the channel's behavior without modifying the contents of
the vector memory.

Setting Driver Channel Static Vectors

It is possible to set dr-mask to ACT for active or TT for tristate. (Tristate mode
for IX type pins is simulated by opening the HSWG output relays)

Masking a driver pin forces the output driver to tristate mode (dr-mask = TT).

break-vec can be set to one of LO, HI, LZ, HZ, or HOLD (LZ and HZ are not
available with channels on 400 MHz I/O boards).

This parameter modifies the pin's behavior during break cycles. The pin may be
programmed to output the signal passed as parameter (LO, HI, LZ or HZ) or to
repeat the last active vector (HOLD). This vector is valid for the entire machine
cycle (i.e. two vectors in FD and MUX modes, or four vectors in FQ mode).

Setting Receiver Channel Static Vectors

`rec-mask` can be either `ACT` to activate it or `X` to change the pin to a "Don't Care" pin in real-time compare mode. The receiver pin mask has no effect in DA mode. Word mask pins are always masked. Attempting to unmask a word mask pin results in a warning.

The `aqi-exp-data` parameter of the SREC command is valid for DA mode only. This parameter allows you to define a vector which will be compared against the incoming data in acquisition mode, to trigger a sequencer jump when a *match condition* is found.

It can be set to:

`IM`	for an intermediate level
`LO`	for a low level
`HI`	for a high level, or
`X`	for a "Don't Care" condition

As with the break vector, this vector will be valid for an entire machine cycle. For 200 MHz pins, an error is generated if the test fails in either of the two vectors; for 400 MHz, an error is generated for a fail in any of the four vectors which are output during that machine cycle.

Power-On Defaults

After power-on, the static vector setup is defined as follows:

- All driver channels are set to tristate, the break vector is set to LOW.
- All receiver channels are masked (set to Don't Care mode), the expected data vector (valid in DA mode only) is set to Don't Care.

Autocorrection Rules

If the configuration of an already defined pin is changed, the following autocorrection rules apply. Note that no warnings will be generated for the corrections noted below.

Invalid break vector and expected data setups are changed from:

- LZ to LO, and HZ to HI
- IM to LO and X to HI (FD mode only)

Errors

- Badly positioned mnemonic
- Execution conflicts with current configuration
- LZ | HZ for 200 MHz pins
- IM | X for 200 MHz double frequency pins

Warnings

A warning will be generated if you attempt to unmask a word mask pin.

Example

```
CONF  I, STD, P, (A0)
CONF  I, FD, P, (A1)
CONF  O, STD, P, (D0)
CONF  O, MUX, P, (D1)
SDRV  ACT, HOLD, (A0, A1)
SREC  ACT, LOW, (D0, D1)
```

In the above example, the DUT input pins A0 and A1 are active. Both pins will repeat the last machine cycle vector in break mode. The DUT output pins D0 and D1 will be monitored. In data acquisition mode, the failure condition becomes false, if the output signal of both pins is LOW for an entire machine cycle.

Vector Transfer Commands

There are eight commands used for transferring vector data to and from the hardware. These are:

DRVD	transfers driver vector data to the hardware using a packed binary data format.
DRVD?	transfers driver vector data from the hardware using a packed binary data format.
EXPD	transfers receiver vector data to the hardware using a packed binary data format.
EXPD?	transfers receiver vector data from the hardware using a packed binary data format.

VESA	defines the vector start address. This address is used as the base address for all subsequent vector data transfers.
VESA?	reports the current value of the vector start address
VEFA	modifies the first free vector address. This address is updated after each vector data transfer to show the first unused location in the vector data memory.
VEFA?	reports the current setting of the first free vector address.

Syntax

```
DRVD rel-start-addr, vector-length, pin-list, #spec(binary-data)

DRVD? rel-start-addr, vector-length, pin-list

EXPD rel-start-addr, vector-length, pin-list, #spec(binary-data)

EXPD? rel-start-addr, vector-length, pin-list

VESA start-addr

VESA?

VEFA ffvad

VEFA?
```

Transferring Driver and Receiver Vector Data

rel-start-addr = V-LAST = 0 .. highest vector memory address

vector-length = V-MEM = 1 .. size of vector memory

spec = digit{digit}

start-addr = 0 .. V-LAST

ffvad = 0 .. V-MEM

The vector data transfer commands DRVD and DRVD? are used to transfer large amounts of driver vectors via HP-IB. The data itself is transferred in a non ASCII format and will be transferred as an entire block for the specified pin.

The pin-list parameter must specify exactly one pin for the vector setup commands DRVD and EXPD, while pin-list for the DRVD? and EXPD? commands may hold any number of pins. Transferring data from the hardware

results in a string of data being sent to the host in the same format as the DRVD and EXPD commands.

The **#** marks the start of the binary data block to transfer. It is immediately followed by the `spec` parameter, which defines the number of bytes to transfer. This parameter follows the following rules:

1. The first digit specifies the number of digits the length specifier consists of.

2. The specifier itself has to be transferred immediately after the first `spec` digit.

3. The vector bytes are sent immediately after the length specifier and terminated with a linefeed character (\n).

Note	The number of bytes has to match the length specifier value. No white space is allowed between the elements of the length specifier described above.

The vector data block will be loaded to the vector RAM unchanged. Therefore, the contents of the data stream depends on the hardware vector RAM organization.

If the number of vectors to be downloaded exceeds the vector memory size, the remaining bytes are ignored and a warning message will be issued.

The vector transfer starts at the byte location of the vector specified by the previously defined vector start address and the `rel-start-addr` parameter (which is regarded as an offset value, added to the vector start address). The vector base address can be altered by the VESA command.

Note	All vectors affected by the bytes sent will be overwritten. To avoid erroneous vector RAM contents, you should set the vector start address, the `rel-start-addr`, and the vector length to multiples of 4 (because one byte specifies four vectors).

If the requested download operation was successful, the first free vector address will be updated to show the first unused vector RAM location. This address can be altered by the VEFA command.

Power-On Defaults

After power-on, the entire vector memory contents will be initialized to

- LO (no tristate) for driver channels
- 'X' for receiver channels

In addition, the vector start address and the first free vector address are both set to 0.

Errors

- Parameter out of range
- Address exceeds vector memory limits (VESA, VEFA)
- Vector length exceeds data block size (DRVD, EXPD)
- Invalid vector length (DRVD?, EXPD?)
- Vector start address exceeds vector memory limits (DRVD, EXPD)
- More than one pin in pin-list (DRVD, EXPD)
- Tester is not in real-time compare mode (EXPD, EXPD?)
- Execution conflicts with current configuration for pin-list (DRVD, EXPD)
- Vector data transfer failed or aborted (DRVD, EXPD, DRVD?, EXPD?) Note, that the vector memory contents may be partially overwritten if this error occurs.

Warnings

- Vector data truncated (DRVD, EXPD)

Example

```
CONF I, MUX, P, (A0)

VESA 2

DRVD 0, 12, (A0), #42048(2 KByte binary data)
```

VEFA? now responds with VEFA 14

In the above example, the vector RAM of the 200 MHz (MUX) input pin A0 is loaded from vector address 0 through 15 with the first 4 bytes of each binary 1K data block. Bytes 1 through 4 are loaded to the master channel's vector RAM, bytes 1025 through 1028 are loaded to the slave channel's vector RAM. All

other bytes are discarded and will have no effect. The vectors 0, 1, 14 and 15 will be overwritten. The first free vector address `ffvad` will be set to 14.

Vector Data Format

Calculating the Data Transfer Block Size

You can use the following equation to calculate the minimum number of data bytes to be transferred for a given `vector-length`, `start-addr` and `rel-start-addr`.

$$ValidBytes = \frac{(StartAddr + RelStartAddr)MOD4 + (VecLength - 1)}{4} + 1$$

Where:

ValidBytes   = Minimum size for data block
StartAddr   = Base address for vector transfer (`start-addr` parameter of VESA command).
RelStartAddr   = Offset relative to base address (`rel-start-addr` parameter of DRVD or EXPD command).
VecLength   = Vector Length (specified by DRVD or EXPD command).
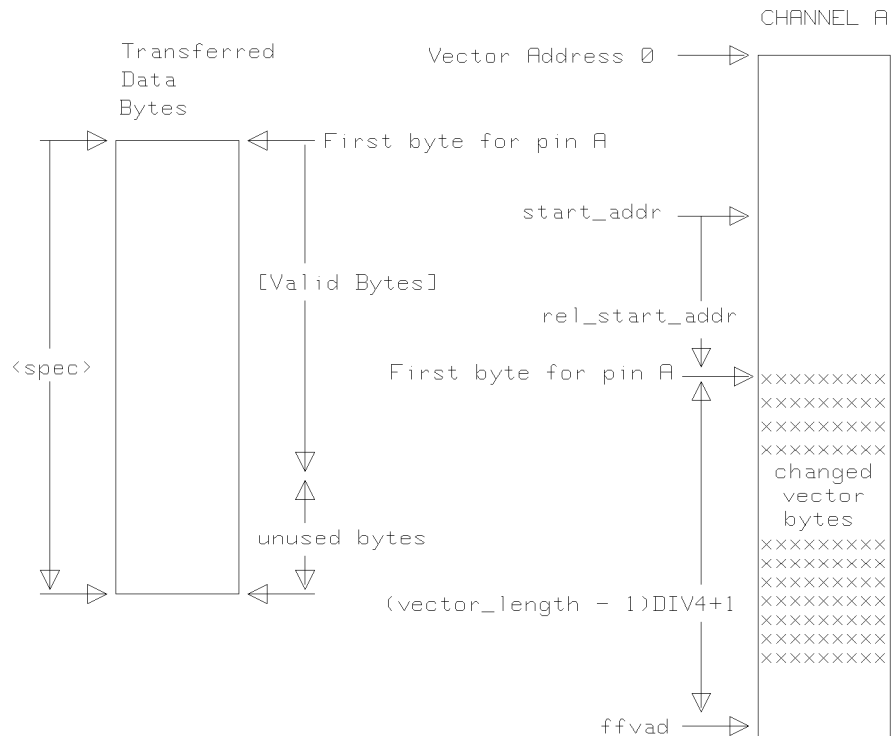
Figure 5-1. Data Alignment (STD and FD mode)
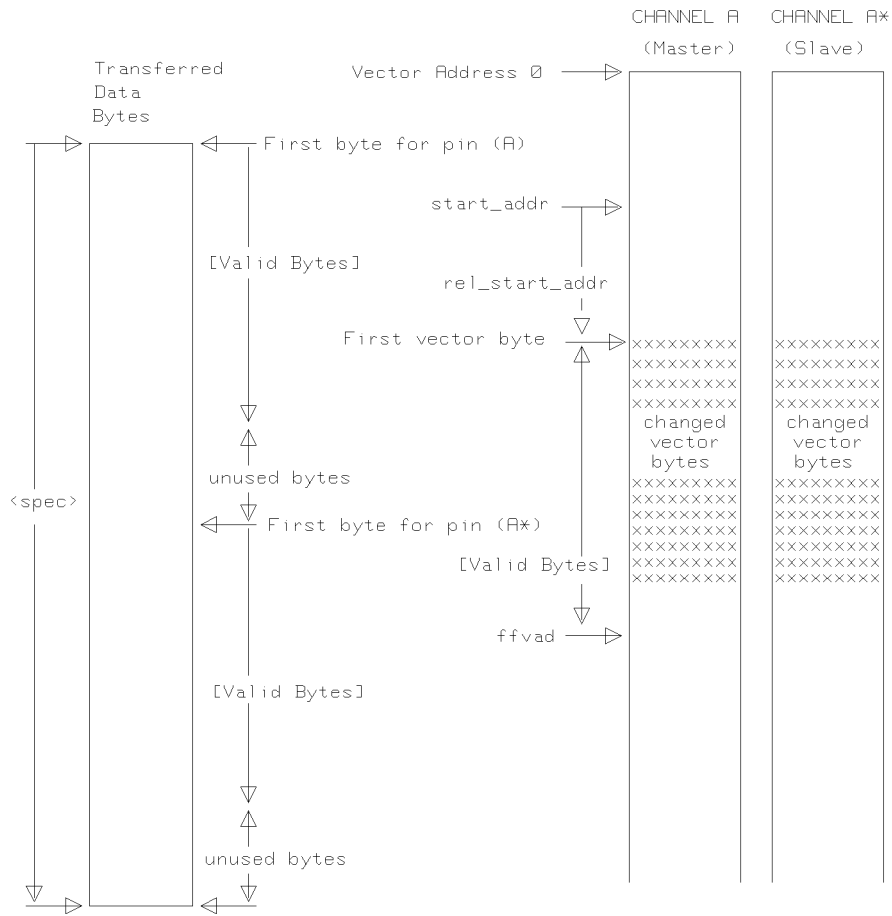
Figure 5-2. Transfer Data / Vector RAM alignment (MUX mode)

Note that an odd length specifier `spec` causes the size of the gap between the valid data blocks to be one less than the number of bytes ignored at the end of the entire data block.

Data Formats

The following tables show the data bit alignment for the different operating modes.

STD Mode

In this mode, data contains data and tristate bits.

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	D0	T0	D1	T1	D2	T2	D3	T3
	E0/0	E0/1	E1/0	E1/1	E2/0	E2/1	E3/0	E3/1

where

Dn   drive data
Tn   tristate on (1) or off (0)
En/0, En/1   compare mode (as shown below)

Compare Modes for STD Mode

En/0	En/1	compare mode
0	0	intermediate
0	1	low
1	0	high
1	1	don't care

Word Mask Pins

These are only valid for expected data.

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	M0	X	M1	X	M2	X	M3	X

where

Mn   mask bit for vector n
                          0 = word mask not active
                          1 = vector masked

FD Mode

In this mode, there are no tristate bits.

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	D0.0	D0.1	D1.0	D1.1	D2.0	D2.1	D3.0	D3.1
	E0.0	E0.1	E1.0	E1.1	E2.0	E2.1	E3.0	E3.1

where

Dn.0	drive data (first half-cycle)
En.0	expected data (first half-cycle)
Dn.1	drive data (second half-cycle)
En.1	expected data (second half-cycle)

MUX Mode and FQ2 mode

Drive Data

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	D0.0	"0"	D1.0	"0"	D2.0	"0"	D3.0	"0"
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Dw.0	"0"	Dx.0	"0"	Dy.0	"0"	Dz.0	"0"
	D0.1	"0"	D1.1	"0"	D2.1	"0"	D3.1	"0"
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Dw.1	"0"	Dx.1	"0"	Dy.1	"0"	Dz.1	"0"

Expected Data

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	E0.0/0	E0.0/1	E1.0/0	E1.0/1	E2.0/0	E2.0/1	E3.0/0	E3.0/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.0/0	Ew.0/1	Ex.0/0	Ex.0/1	Ey.0/0	Ey.0/1	Ez.0/0	Ez.0/1
	E0.1/0	E0.1/1	E1.1/0	E1.1/1	E2.1/0	E2.1/1	E3.1/0	E3.1/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.1/0	Ew.1/1	Ex.1/0	Ex.1/1	Ey.1/0	Ey.1/1	Ez.1/0	Ez.1/1

where

Dn.0	drive data (first half-cycle)
Dn.1	drive data (second half-cycle)
En.0/0, En.0/1	expected data (first half-cycle)
En.1/0, En.1/1	expected data (second half-cycle)

FQ Mode

In this mode, there are no tristate bits.

Drive Data

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	D0.0	D0.1	D1.0	D1.1	D2.0	D2.1	D3.0	D3.1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Dw.0	Dw.1	Dx.0	Dx.1	Dy.0	Dy.1	Dz.0	Dz.1
	D0.2	D0.3	D1.2	D1.3	D2.2	D2.3	D3.2	D3.3
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Dw.2	Dw.3	Dx.2	Dx.3	Dy.2	Dy.3	Dz.2	Dz.3

Expected Data

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	E0.0	E0.1	E1.0	E1.1	E2.0	E2.1	E3.0	E3.1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.0	Ew.1	Ex.0	Ex.1	Ey.0	Ey.1	Ez.0	Ez.1
	E0.2	E0.3	E1.2	E1.3	E2.2	E2.3	E3.2	E3.3
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.2	Ew.3	Ex.2	Ex.3	Ey.2	Ey.3	Ez.2	Ez.3

where

Dn.0	drive data (first quarter-cycle)
Dn.1	drive data (second quarter-cycle)
Dn.2	drive data (third quarter-cycle)
Dn.3	drive data (fourth quarter-cycle)
En.0	expected data (first quarter-cycle)
En.1	expected data (second quarter-cycle)
En.2	expected data (third quarter-cycle)
En.3	expected data (fourth quarter-cycle)

FQM Mode

Drive Data cannot be generated in this mode.

Expected Data

Vector	Vector 0		Vector 1		Vector 2		Vector 3	
Bit Pos.	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	E0.0/0	E0.0/1	E1.0/0	E1.0/1	E2.0/0	E2.0/1	E3.0/0	E3.0/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.0/0	Ew.0/1	Ex.0/0	Ex.0/1	Ey.0/0	Ey.0/1	Ez.0/0	Ez.0/1
	E0.1/0	E0.1/1	E1.1/0	E1.1/1	E2.1/0	E2.1/1	E3.1/0	E3.1/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.1/0	Ew.1/1	Ex.1/0	Ex.1/1	Ey.1/0	Ey.1/1	Ez.1/0	Ez.1/1
	E0.2/0	E0.2/1	E1.2/0	E1.2/1	E2.2/0	E2.2/1	E3.2/0	E3.2/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.2/0	Ew.2/1	Ex.2/0	Ex.2/1	Ey.2/0	Ey.2/1	Ez.2/0	Ez.2/1
	E0.3/0	E0.3/1	E1.3/0	E1.3/1	E2.3/0	E2.3/1	E3.3/0	E3.3/1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	Ew.3/0	Ew.3/1	Ex.3/0	Ex.3/1	Ey.3/0	Ey.3/1	Ez.3/0	Ez.3/1

where

En.0/0, En.0/1	expected data (first quarter-cycle)
En.1/0, En.1/1	expected data (second quarter-cycle)
En.2/0, En.2/1	expected data (third quarter-cycle)
En.3/0, En.3/1	expected data (fourth quarter-cycle)

6

Vector Sequencer Commands

The vector sequencer allows you to modify the order in which vectors are sent to the DUT. Conditional jumping and looping statements allow the generation of complex vector sequences dependent on the results of tests.

Programming the Vector Sequencer

Sequencer programming can be very involved, and for this reason, there are two sequencer programming windows provided in the PWS software; the Sequencer Control Screen and the Sequencer Programming Screen. The instructions that are programmed in these windows are converted to the commands described in this section.

The commands in this section are:

SQST	set sequencer state
SQST	read sequencer state
SQPG	write sequencer instructions
SQPG?	read sequencer instructions
SQCL	clear sequencer label
SQFA	set first free sequencer instruction
SQFA?	retrieve first free sequencer instruction
SQSA	set start address of next sequencer instruction line
SQSA?	retrieve start address of next sequencer instruction line
SQGB	define global sequencer parameter
SQGB?	get global sequencer parameter
SQXI	define sequencer external input
SQXI?	get sequencer external input definition

Syntax

```
SQST sequencer_state

SQST?

SQPG inst-no,[label], [state], [vector-type], [vector-start]
, [vector-length], [instruction-count],[jump-destination]
, [jump-condition], [analyzer-stop], [failure-reset-flag],
[apg-on-flag], [flag-2], [flag-1], [flag-0]

SQPG? inst-no

SQFA inst-no

SQFA?

SQSA inst-no

SQSA?

SQCL

SQGB [start-mode], [trigger-vector], [analyze-stop-delay]
,[serial-word-length], [break-on-fail/full]

SQGB?

SQXI input-id, [purpose], [active], [threshold], [impedance]

SQXI? input-id
```

Setting and Reading Sequencer States

The SQST command puts the sequencer into the specified `sequencer-state`.
The possible states are:

OFF   (sequencer and clock off) The clock (and therefore the entire
timing system) will be stopped. Only static signals are available
at the connected DUT pins.

BRK   (sequencer in break) The clock and the timing system are fully
                functional. The input signals at the connected DUT pins match
                the defined break vector. On entering the "BRK" state, a test
                currently in process will be aborted.
RUN   (sequencer running) This command parameter causes a
                sequencer program to be executed starting at the currently
                defined start label. Depending on the global sequencer settings
                (SQGB), the sequencer will either be armed (waiting for an
                external condition) or active (generating test vectors). The
                sequence of test vectors generated depends on the previously
                downloaded sequencer program.

Selecting the sequencer RUN state results in a temporary change to the BRK state,
if the sequencer was in either the RUN or OFF states.

Selecting the sequencer OFF state results in a temporary change to the BRK state,
if the sequencer was in the RUN state.

Changing the sequencer state from RUN to BRK may also be forced by the
sequencer program. In this case, the SQST? query reports the sequencer state
entered under hardware control.

Writing a Sequencer Instruction

The SQPG command defines one instruction for the sequencer. While inst-no
is mandatory, all other fields of the sequencer are optional. If they are omited,
they will retain their previous values.

inst-no   is an integer in the range 0 .. 999 that defines the
                instruction number to write. It is given relative to the
                sequencer-start-instruction given in the SQSA
                command. The resulting absolute instruction number must
                not exceed 0999. The resulting absolute instruction number
                will update the first free sequencer instruction automatically,
                if it is greater.
label   is a string of max 8 characters. Labels can be used later for
                specifying the start of a test. Giving a label with is currently
                defined in another instruction will remove the old definition
                and will generate a warning. Specifying a null string will
                remove a label. The number of labels concurrently defined is
                limited. The firmware can hold up to 15 labels concurrently.
state   controls the state of the sequencer. It can be either
                ■ CONT - continue in processing instructions

- BRK - go to break state (finish sequence)
- ARM - go to arm state and wait for trigger condition before processing instruction

`vector-type`   controls the way in which vectors will be output during this instruction. It can be either

- LIN - output a linear sequence of `vector-length` vectors starting at address `vector-start`
- REP - output vector at address `vector-start` the number of times defined by `vector-length`.
- SCAN - output `vector-length` vectors in serial scan mode starting at address `vector-start` and using `serializing-word-length`.

`vector-start`   defines the start address of the vectors to use in this instruction. The address is given relative to the vector-start-address set by the VESA command. The resulting absolute address must not exceed the highest address of vector memory installed. The value (-1) will specify the vector address after the last vector address used in the last instruction executed.

`vector- length`   defines the number of vectors to output. The minimum vector length is 32 except for jump source EVER or NEV when the minimum is 8. The maximum vector length is 1048575 (2^{20} - 1).

`instruction-count`   specifies how often this instruction is executed. It must be an integer in the range 1 .. 2048. The value (-1) specifies an infinite repetition of this instruction. If the jump condition condition occurs, this repetition will be stopped independently of the `instruction-count`.

`jump-destination`   specifies the instruction to process when the jump condition has been detected. The `inst-no` is given relative to the `sequencer-start-instruction` given in the SQSA command. The resulting absolute instruction number must not exceed 999.

`jump-condition`   specifies the condition which selects whether the next instruction processed is the following instruction or the instruction specified in `jump-destination`. The jump conditions can be derived from :

NEV   *NEVER* jump (default) - this causes the next consecutive sequencer instruction to be processed.

SS	Software Signal - a condition arbitrarily defined by the HP-IB command SQSS
NSS	(equivalent to $\overline{\text{SS}}$)
APG	APG ready
NAPG	(equivalent to $\overline{\text{APG}}$)
FAIL	Compare Failed
NFL	(equivalent to $\overline{\text{FAIL}}$)
XA	External Input A Active
NXA	(equivalent to $\overline{\text{XA}}$)
XB	External Input B Active
NXB	(equivalent to $\overline{\text{XB}}$)
EVER	Unconditional jump to the jump-instr
analyzer-stop	the first occurrence of this flag will trigger the end of data analysis after analyze-delay vectors.
failure-reset-flag	resets the test failed condition at the start of this instruction, otherwise a jump on FAIL may be taken after a failure in a previous instruction
apg-on-flag, flag-2, flag-1, flag-0	are boolean (1|0) flags for later use. They are not currently used.

The query command SQPG? returns the sequencer instruction specified by inst-no. The special value (-1) will define all sequencer instruction in the range from sequencer-start to first-free-sequencer-instruction-1. The query is not defined if vector-start and sequencer-start are set to anything other than zero.

The DEFAULT Sequencer Program. The firmware automatically installs and maintains a default sequencer program. This program outputs vectors from vector 0 to the last transferred vector (FFVAD-1) in a linear sequence. It can be referred to under the label "DEFAULT". Neither the Default sequencer program nor its label can be altered be the user.

Setting the First Free Sequencer Address

The SQFA command sets the first free sequencer instruction to inst-no. inst-no is given absolute and must be in the range from 0 to 999. The first free sequencer instruction will also be updated by the SQPG command, if it is above the inst-no specified by an SQFA command.

The SQFA? query returns the current first free sequencer instruction.

Setting the Sequencer Start Address

The SQSA command sets the start sequencer instruction to `inst-no`. `inst-no` is given absolute and must be an integer in the range 0 .. 999. All instruction numbers given in the SQPG command are treated relative to the current start sequencer instruction.

The SQSA? query retrieves the current start sequencer instruction.

Clearing Sequencer Labels

The SQCL command removes the definition of all labels defined by SQPG commands.

Setting Sequencer Global Conditions

The SQGB command specifies global parameters for the sequencer. They are:

`start-mode`	specifies how the sequencer will be started at a sequencer start command. It can be either ■ ARM the sequencer will start only after a external run signal has received. ■ RUN the sequencer will start immediately after the sequencer start command.
`trigger-vector`	specifies the vector-address, which should activate the trigger output of the tester when applied. It must be a valid vector address (0..V-MEM).
`analyze-stop-delay`	specifies how many cycles the analyze memory full event will occur after a instruction with the `analyze-stop` flag set was executed. This event may put the sequencer into break mode. The last 64k vector are available at stop. The value of the `analyze-stop-delay` must be in the range 1 to 16777000.
`serial-word-length`	specifies the word-length which is to use during instructions with `vector-type` "SCAN". It must be in the range from 4 to the total number of pins installed.
`break-on-fail / end`	if set, the tester will go into break if a compare error occurs or the analyze memory is full. If reset, the tester will continue testing even if a compare error occurs or the analyze memory is full.

The SQGB? query retrieves the sequencer global setting

Setting the Sequencer External Input

The sequencer can be made to react to an external input which could come from the DUT or any other source.

Two external inputs with identical capabilities are available. They are named XA and XB and are located on the Sequencer Board. The parameter `input-id` selects the external input to set. It is mandatory, while all other parameter are optional. Omitting a parameter leaves it value unchanged. The parameters are:

`input-id`	can be either XA or XB. This selects the external input to to be modified. This parameter is mandatory.
`purpose`	selects the task the input is used for. It can be
	■ BRK - to put the sequencer from active into break state (the external input must not be used for instruction jumps)
	■ CONT - to put the sequencer from arm into active state (the external input is available for instruction jumps also)
	■ JUMP - to cause jumps in appropriate sequencer instructions (the external input is available for instruction jumps only)
`slope`	defines what is treated as active for this input. This can be
	■ H - high level
	■ L - low level
	■ R - rising edge
	■ F - falling edge
`threshold`	defines the threshold of the input. The integer given is treated as millivolts and must be in the range -10000 (mV) to +10000 (mV). A warning will be generated, if the value is outside the range ± 5000 and low impedance is selected, because applying voltages out of this range may damage the tester. However, the input will be programmed accordingly.
`impedance`	specifies how the external input is terminated. It must be either
	■ H high = 10 kΩ
	■ L low = 50 Ω

Power-On Defaults

At power-on, only the default label is defined and the following settings are made:

```
SQPG all-instructions,"",BRK,LIN,0,8,1,0,NEV,0,0,0,0,0,0

SQFA 0

SQSA 0

SQGB RUN,0,16777000,8,0


SQXI XA,JUMP,H,0,H

SQXI XB,JUMP,H,0,H
```

Errors

- Value out of Range

- Bad label

- Too many labels

Warnings

- External Input Threshold out of spec for low impedance

- Label redefined

Example

```
SQPG 999,"Harry",CONT,LIN,64000,64000,2000,1000,NEV,1,1,1,1,1,1
SQCL
SQGB RUN,64000,16000000,126,1
SQXI XA,BRK,H,-10000,H
```

7

Test Function Commands

This chapter describes the commands used to:

- set the tester state
- perform functional tests
- perform ac measurements
- perform dc measurements using the DC Measurement Unit
- perform dc measurements using the DPS

Additional low level commands in this chapter are used to:

- switch relays
- control the PMU
- set the sequencer software signal
- set and read the system utility lines

Setting the Tester States

In the PWS software, there are tester states such as CONNECT, DISCONNECT, and TESTING. These states are not explicitly reflected by any HP-IB commands. However, the firmware supports these states by providing some primitives to handle tester states.

These primitives consist of:

- relay control commands to allow DUT pins to be connected and disconnected
- clock/sequencer commands that control clock and sequencer states

Examples

Switching from DISCONNECT to CONNECT

```
<report-message> "CONNECT"
PSST ON
RLYC AC, X, (<all-ac-pins>)
SQST BRK
```

The RLYC command closes all the ac relays, the PSST command the power supply relays, and the SQST command sets the sequencer to the BRK state. The RLYC, PSST, and SQST commands are described in the following sections.

Switching from CONNECT to DISCONNECT

```
SQST OFF
RLYC IDLE,OFF,(@)
PSST OFF
<report-message> "DISCONNECT"
```

In this example, the sequencer is stopped, all relays are opened, the DPS is switched off and the "DISCONNECT" message is sent to the display.

Functional Test Commands

A functional test consists of sending vectors to the DUT and checking that the resultant outputs are as expected. No ac or dc measurements are made during a functional test, the result is simply PASSED or FAILED.

To perform a functional test, the following requirements need to be fulfilled:

- Vector, timing, level and other setups must be loaded in the hardware - these commands are described in other parts of this manual.
- A sequencer start label must be defined to determine the order in which vectors must be sent to the DUT
- The comparators must be set to either acquire incoming data or compare it to expected data.
- The sequencer must be started
- When the sequencer stops, the test result needs to be read.

There are eight commands that are used for functional testing:

SQSL	defines the sequencer start label
SQSL?	retrieves the sequencer start label
RCMD	defines the receive mode
RCMD?	retrieves the receive mode
SQST	changes the sequencer state
SQST?	retrieves the sequencer state
FTST?	runs a functional test and returns the result of the test (PASSED/FAILED).
FTCK?	runs a functional test and keeps the system clock running.

Syntax

```
SQSL label

SQSL?

RCMD receive-mode

RCMD?

SQST sequencer-state

SQST?

FTST?

FTCK?
```

label	is a string of ≤ 8 characters indicating a sequencer start label. This label must have been previously defined either by using the Vector Sequencer window or the SQPG command (refer to the Vector Sequencer chapter).
receive- mode	is either DA for data acquisition mode or RTC for real-time compare mode.
sequencer-state	can be either OFF, BRK, or RUN which sets the sequencer to the OFF, BREAK, and RUN states respectively.
test- result	returns either P or F depending on whether the test passed or failed.

Sequencer Start Label

The start label defined by the SQSL command is used to identify the first instruction (and therefore the first test vector) executed by subsequent tests.

If the defined start label is no longer valid (this could be caused by downloading a different sequencer program or by clearing the previously declared labels), the start label used by subsequent tests (and reported by the SQSL? query) will be changed to the "DEFAULT" label.

Starting a test at the "DEFAULT" start label causes the default sequencer program to be executed. This is described in the Vector Sequencer chapter.

Specifying a start label while the sequencer is running will interrupt the test being executed. When the sequencer is subsequently restarted, the first vector to be output will be the vector at `label`.

Receive Mode

The `receive-mode` defined by the RCMD command specifies the way in which the tester will process the incoming data of subsequent tests. If the tester is set up for real-time compare mode (RTC), the DUT response will be compared against the expected data downloaded previously using the EXPD command.

If the system is set up in data acquisition mode (DA), the DUT response will be stored in the received data memory. The RECD? query may be used to examine the sampled data.

The query commands described in the Test Results chapter are provided to allow further interpretation of the test results obtained. These queries depend on the currently selected received data processing mode.

Sending the RCMD command while the sequencer is running will interrupt the test being executed. The test will be restarted at the specified start label, with the current `receive-mode`.

Sequencer States

The SQST command puts the sequencer into the specified `sequencer-state`. The possible states are:

OFF   (sequencer and clock off) The clock (and therefore the entire timing system) will be stopped. Only static signals are available at the connected DUT pins.

BRK   (sequencer in break) The clock and the timing system are fully functional. The input signals at the connected DUT pins match the defined break vector. On entering the "BRK" state, a test currently in process will be aborted.

RUN   (sequencer running) This command parameter causes a sequencer program to be executed starting at the currently defined start label. Depending on the global sequencer settings (SQGB), the sequencer will either be armed (waiting for an external condition) or active (generating test vectors). The sequence of test vectors generated depends on the previously downloaded sequencer program.

Selecting the sequencer RUN state results in a temporary change to the BRK state, if the sequencer was in either the RUN or OFF states.

Selecting the sequencer OFF state results in a temporary change to the BRK state, if the sequencer was in the RUN state.

Changing the sequencer state from RUN to BRK may also be forced by the sequencer program. In this case, the SQST? query reports the sequencer state entered under hardware control.

Performing a Functional Test

The FTST? query starts a functional test and returns the test result after the sequencer has stopped. The pattern used for the functional test is selected by the current sequencer start label, defined with the SQSL command. The received data processing mode has to be selected prior to a functional test by sending the RCMD command.

If the sequencer is already running, the functional test query aborts the test in process and restarts the sequencer at the defined start label.

A functional test running in data acquisition mode always returns P (passed), reflecting that data comparison has been disabled.

The test-result returned by a functional test in real-time compare mode will be F (failed), if any of the unmasked DUT output signals did not match the expected data for at least one test vector.

The Device Clear command (DCL) may be used to abort a functional test currently in process.

The following example performs a functional test and returns the number of failed cycles.

```
RCMD RTC        Set to real-time compare mode
SQSL "TEST"     Select the pattern to be used
FTST?           Perform the functional test
ERCT?           Read the number of errors
```

The following sequence of low level commands may be used to simulate the FTST? query.

```
SQST RUN     Start sequencer
...          wait until SQST? returns BRK
PASS?        read test results
```

Reloading the Pattern Memory during a Functional Test

The FTCK? query allows you to execute functional tests in a similar way to FTST?, but unlike the FTST? query, FTCK? does not stop the system clock before starting the sequencer.

This facility is useful if your test vector pattern exceeds the length of the pattern memory, because in this case, you have to reload the pattern memory during your test. If the clock is stopped during the reload, a reset signal is issued in the hardware and, instead of staying in the Break state, the sequencer is turned off.

You can reload the pattern memory during a functional test by:

■ downloading your first set of test vectors;
■ performing your test with the FTST?;
■ reload the pattern memory with the second set of vectors;
■ perform the test again, but using FTCK?.

Because the clock is not stopped between vectors, the tester supplies your DUT with the break vector while the pattern memory is being reloaded. This means that your device is in a known logic condition when the test restarts.

In addition, the error flags (accessible with the CHER? command) are not reset, and will hold the total number of errors when the test finishes.

Although the error flags remain valid until the end of the test, the value in the error counter (read using ERCT?), is reset each time the sequencer is started, so you must read the value of this counter between tests, if you need this information.

Note	Use the FTCK? command in real-time compare mode. If it is used in data acquisition mode, the acquired data may be invalid.

Example

The following example illustrates the use of the FTCK? command.

```
DRVD 0,65536,(Din),#516384....
EXPD 0,65536,(Dout),#516384....    Load first part of vectors
SQSL "DEFAULT"
FTST?                               Perform functional test
ERCT?                               Count errors in first part of test
DRVD 0,65536,(Din),#516384....     Load second part of vectors
EXPD 0,65536,(Dout),#516384....

                                    Perform functional test,
FTCK?                               without stopping the clock
ERCT?                               Count errors in second part of test
CHER?                               Read channel error flags
                                    for complete test
```

Power-On Defaults

At power-on, the following settings are valid:

```
SQSL "DEFAULT"
```

```
RCMD RTC
```

```
SQST OFF
```

Errors

Errors will be generated by:

■ Trying to define an undeclared label.

AC Measurement Commands

The commands described in this section are provided to support the various AC test functions. Most of them need more than one test to be run. Therefore, the sequencer may be restarted several times (at the defined start label). It is up to the user to ensure that all external conditions affecting the sequencer program flow behave the same during all tests.

The following commands are described in this section:

PDFT?   performs a combined propagation delay and data hold time measurement and returns a PASS/FAIL indication

TPDM?   performs a propagation delay measurement and returns a value.

TDHM?   performs a data hold time measurement and returns a value.

ESGB   sets up the step width settings for searches - used by RCES?, TSUP?, and THLD? and the minimum pulse width settings for input timing measurements - TSUP? and THLD?

ESGB?   reports the current settings of ESGB.

RCES?   sets the limits and search mode for output pin measurements.

TSUP?   performs a setup time measurement and returns a PASS/FAIL indication

THLD?   performs a hold time measurement and returns a PASS/FAIL indication

TPWM?   measures the minimum pulse and pause width for any number of input pins

DRLS?   performs a $V_{IL\ max}$ and $V_{IH\ min}$ test and returns a PASS/FAIL indication

RCLS?   performs a $V_{OL\ max}$ and $V_{OH\ min}$ test and returns a PASS/FAIL indication

All commands will modify the tester setup (format, timing or levels) during execution. However, the original settings will be restored upon termination. Aborting the command by sending a Device Clear message (SDC) aborts the test currently executing and restores all modified setups. For all commands modifying the timing setup, an unstable (or entirely missing) external phase sync signal will cause the system clock to run out of phase if external phase synchronization has been selected (see SCLK, SYNC and SNCC). No warnings or error messages will be sent in this case.

For all test function commands it is recommended that the operation modes of all pins specified in the passed `pinlist` are the same (either MUX, STD, FD, FQ2, FQ or FQM).

The resolution of timing measurement results depends on the programmed tester period as shown in the table below.

Note	This is the smallest discrete increment that the tester can use for presenting results; this does not affect the *accuracy* of the tester.

Table 7-1.
AC Test Function Timing Resolution (D100, D200 and D400)

Clock Period	Resolution
≤ 900 ns	50 ps
< 10 μs	1 ns
< 100 μs	10 ns

Table 7-2. AC Test Function Timing Resolution (D50)

Clock Period	Resolution
≤ 900 ns	200 ps
< 10 μs	1 ns
< 100 μs	10 ns

Syntax

```
PDFT? tpd, tdh, pinlist
```

returns

```
PDFT test-result
```

```
TPDM? pinlist
```

returns

```
TPDM rel-op, meas-val
```

```
TDHM? pinlist
```

returns

```
TDHM rel-op, meas-val
```

```
ESGB [pure_lin_step], [lin_bin_step], [min_pulse_width]

ESGB?

RCES? start, stop, mode, pinlist
```
returns
```
RCES rel-op, meas-val

TSUP? start, [limit], pinlist
```
returns
```
TSUP res-spec, meas-val

THLD? start, [limit], pinlist
```
returns
```
THLD res-spec, meas-val

TPWM? le_delay, te_delay, pinlist
```
returns
```
TPWM res-spec, min-pulse-width, res-spec, min-pause-width

DRLS? level_select, pass_value, [limit], pinlist
```
returns
```
DRLS res-spec, meas-value

RCLS? level_select, pass_value, [limit], pinlist
```
returns
```
RCLS res-spec, meas-value
```
where

tpd   is a real number with the implied units nanoseconds used for
           the propagation delay time
tdh   is a real number with the implied units nanoseconds used for
           the data hold time

`test_result`	can be either P or F to indicate passed or failed
`rel_op`	is a relational operator that can be one of:
	LT - less than
	LE - less than or equal to
	EQ - equal to
	GE - greater that or equal to
	GT - greater than
`meas_value`	is a real number with the implied units nanoseconds or millivolts used for the measured value
`pure_lin_step`	is a real number with implied units nanoseconds used to define a linear step
`lin_bin_step`	is a real number with implied units nanoseconds used to define a linear/binary step
`start`	is a real number with implied units nanoseconds used to define the start of a window
`stop`	is a real number with implied units nanoseconds used to define the end of a window
`mode`	either L or LB to specify a linear or linear/binary search sequence
`limit`	is a real number with implied units nanoseconds or millivolts used to define the limit for a measurement
`res_spec`	can be a relational operator of type `rel_op`, or P or F to indicate a pass or fail
`level_select`	can be H or L to indicate a high or low level
`min_pulse_width`	'is a real number with implied units nanoseconds giving the minimum measured pulse width.
`min_pause_width`	is a real number with implied units nanoseconds giving the maximum measure pause width.
`le_delay`	is a real number with implied units nanoseconds used to define a leading edge delay
`te_delay`	is a real number with implied units nanoseconds used to define a trailing edge delay

Defining Edge Search Parameters

The ESGB command is used to setup the step width required for a linear edge search (RCES?, TSUP?, THLD?) and the minimum pulse width required for all input pin timing measurements (TSUP?, THLD?). Refer to the description of the commands in parentheses to understand how the values setup by the ESGB command are actually used.

A negative value for `pure_lin_step` or `lin_bin_step` is used to specify an infinite step width (see RCES?).

`Min_pulse_width` must not be negative.

At power-on the following default values are active

    ESGB -1,2,2.5   *For D100, D200 and D400*
    ESGB -1,4,5   *For D50*

Measuring Propagation Delay and Data Hold Time

Three commands are available to measure the propagation delay and/or data hold time of a DUT output pin.

Pass/Fail Test for 100 MHz and 200 MHz MUX-mode Pins. The PDFT? command returns a passed/failed result of a combined propagation delay/data hold time test. The test is done in window compare mode, setting the compare window of all selected pins to

`tpd .. tdh` + period (STD pins)
`tpd .. tdh` + period/2 (MUX pins)

If the required window exceeds the maximum tester window width (period - 3 ns), two tests with overlapping windows are performed.

`Test_result` will be either P (test passed) or F (test failed).

The allowed value ranges depend on the configuration of the pins passed in `pinlist`. If no MUX pins are specified, pin_period = tester_period. If at least one MUX pin is specified, pin_period = tester_period / 2

`tpd` and `tdh` must be in the following ranges:

For a propagation delay measurement,

$tpd \geq 0$
$tpd \leq pin\_period$

For a data hold time measurement,

$tdh \geq 0$
$tdh \geq tpd - pin\_period + 2$ ns
$tdh \leq tpd$
$tdh \leq tester\_period - 3$ ns

In a multiple mainframe configuration, only the master mainframe responds to the PDFT? query. Pins in `pinlist` must be configured as O, IO, IOL, IOH, TERM, or NC pins. FD, FQ2, FQ and FQM pins are not allowed to be part of `pinlist`.

Exceeding the value ranges for the `start` and `stop` parameters causes an error message to be sent to the host and no test to be run.

Value Test Data Hold Time for 100 MHz and 200 MHz MUX-mode Pins. The TPDM? command returns the measured (maximum) propagation delay of any number of DUT output pins.

A binary search algorithm is used to measure the propagation delay. Measurement is done in window compare mode, varying the leading edge to find the minimum delay at which a functional test passes. The trailing edge will be initially set to

period - 2 ns (STD pins only)
period/2 - 2 ns (at least one MUX pin)

but may be auto-corrected during the search in the range

period - 3 ns .. period + 2 ns (STD pins only)
period/2 .. period/2 + 2 ns (at least one MUX pin)

The returned `rel_op` will be

EQ if a passed/failed transition could be detected
LE if all tests passed
GT if all tests failed

In a multiple mainframe configuration, only the master mainframe responds to the TPDM? query. Pins in `pinlist` must be configured as O, IO, IOL, IOH, TERM, or NC pins. FD, FQ2, FQ and FQM pins are not allowed to be part of `pinlist`.

Value Test Propagation Delay for 100 MHz and 200 MUX-mode Pins. The TDHM? command returns the measured (minimum) data hold time of any number of DUT output pins.

A binary search algorithm is used to measure the data hold time. Measurement is done in window compare mode, varying the trailing edge to find the maximum delay at which a functional test passes. The leading edge will be initially set to

period (STD pins only)
period/2 (at least one MUX pin)

but may be auto-corrected during the search in the range

period -2 ns .. period (STD pins only)
period/2 - 2 ns .. period/2 (at least one MUX pin)

The returned `rel_op` will be

    EQ if the maximum data hold time could be measured
    GE if all tests passed
    LT if all tests failed

Pins in `pinlist` must be configured as O, IO, IOL, IOH, TERM, or NC pins. FD, FQ2, FQ and FQM pins are not allowed to be part of `pinlist`.

Testing 200 MHz FD, FQ2, FQ and FQM mode pins. The RCES? command is used for both value and pass-fail tests of any number of DUT output pins. The command performs a linear search, varying the leading edge, starting at `start` and terminating after the first test fails or after the search limit specified by `stop` has been reached. If a linear-binary search has been selected (`mode` = LB), an additional binary search in the interval between the last passed and the first failed test will be done, to improve the measurement resolution. The trailing edge may be auto- corrected, if the command is used for window compare mode pins.

Both, `start` and `stop` must be in the range

0 .. period (no MUX pin)
0 .. period/2 (at least one MUX pin)

An increasing search will be performed, if `start` is less than or equal to `stop`, otherwise a decreasing search will be used.

The step width used for the linear search depends on the selected mode and will be

`pure_lin_step` (if `mode` = L)
`lin_bin_step` (if `mode` = LB)

In either case, an infinite step width results in performing no more than two tests, one at the specified `start` and the other at the specified `stop` value.

The returned `rel_op` depends on the search direction and will be

    EQ if the maximum data hold time could be measured
    GT if the first test failed (decreasing search)
    GE if all tests passed (increasing search)
    LE if all tests passed (decreasing search)
    LT if the first test failed (increasing search)

In a multiple mainframe configuration, only the master mainframe responds to the RCES? query. Pins in `pinlist` must be configured as O, IO, IOL, IOH, TERM,

or NC pins. Exceeding the value ranges for the `start` and `stop` parameters causes an error message to be sent to the host and no test to be run.

Example

The following example shows how to use these queries to measure the propagation delay and data hold time of a DUT output pin. Assuming a tester period of 100 ns and expected pass values of 2 ns (tpd) and 0 ns (tdh) and a reference signal (tref) at 10 ns.

>   PDFT? 12,10,(pin)

Propagation Delay and Data Hold Time Test. The command returns "PDFT P" if the test passed, otherwise "PDFT F" will be returned.

>   TPDM? (pin)

Propagation Delay Measurement. If the returned `rel_op` is EQ, the propagation delay can be derived from the returned value : tpd = `return_value` - tref

>   TDHM? (pin)

Data Hold Time Measurement. If the returned `rel_op` is EQ, the data hold time can be derived from the returned value : tdh = `return_value` - tref

If the pin to be measured is configured as a 200 MHz FD-mode pin, the same example would read:

>   RCES? 50,12,L,(pin)

Propagation Delay Pass/Fail Test. We expect the output signal to be stable in the interval tref + tpd .. period/2. Therefore, a passed test will return the LE `rel_op`. All other return values indicate a failed test.

>   RCES? 50,60,L,(pin)

Data Hold Time Pass/Fail Test. We expect the output signal to be stable in the interval period/2 .. tref + tdh + period/2. Therefore, a passed test will return the GE `rel_op`. All other return values indicate a failed test.

>   RCES? 50,0,LB,(pin)

Propagation Delay Measurement. The pass/fail transition is expected to be in the interval 0 .. period/2. If the returned `rel_op` is EQ, the propagation delay can be derived from the returned value : tpd = `return_value` - tref

>   RCES? 50,100,LB,(pin)

Data Hold Time Measurement. The pass/fail transition is expected to be in the interval period/2 .. period. If the returned `rel_op` is EQ, the data hold time can be derived from the returned value : tdh = `return_value` - tref - period/2

Setup / Hold Time Measurement

Two commands (TSUP?, THLD?) are available, to measure the setup and/or hold hold time of a DUT input pin. Both commands are used for pass/fail and value tests.

Setup Time Pass/Fail and Value Test. The TSUP? command returns a passed/failed result of a setup time test, if `limit` has been omitted. The test is done by a single functional test, setting the driver leading edge of all pins in `pinlist` to the expected pass value, passed as `start`. The trailing edge may be auto-corrected (for Rx format pins only) to ensure the `min_pulse_width` specified (see ESGB). The passed/failed test returns a `res_spec` of either P (pass) or F (fail). The returned value will be the passed `start` value.

A value test is done by varying the driver leading edge of all pins passed in `pinlist`. The first measurement, done at the expected pass value `start`, is used to select either a binary search (test failed) or linear search (test passed) algorithm to measure the DUT pins setup time. The linear search will be done, increasing the edge delay by `lin_bin_step` until the first test fails (or the hardware programming limit, period or period/2, is reached). An additional binary search improves the resolution of the returned result value. The binary search will be done in the interval `limit` through `start` to measure the setup time. For Rx format pins, the trailing edge may be auto-corrected during the search, to ensure the `min_pulse_width` specified (see ESGB). The returned `res_spec` will be

EQ if the setup time could be measured
GE if all tests passed
LT if all tests failed

The allowed value range for the `start` parameter depends on the configuration of the pins passed in `pinlist`

0 .. period (no MUX pins)
0 .. period/2 (at least one MUX pin)

The `limit` parameter, if specified, must be in the range

0 .. `start`

In addition, the `min_pulse_width` specified prior to this command must not exceed the hardware limits and has to be in the range

0 .. period - 0.5 ns (no MUX pins)
0 .. period/2 (at least one MUX pin)

In a multiple mainframe configuration, only the master mainframe responds to the TSUP? query. Pins in `pinlist` must be configured as I, IO, IOL, IOH, or NC pins; IX pins must not be included in the pinlist. Exceeding the value ranges for the `start` and `limit` parameters causes an error message to be sent to the host and no test to be run.

Hold Time Pass/Fail and Value Test. The THLD? command returns a passed/failed result of a hold time test, if `limit` has been omitted. The test is done by one or two functional tests, depending on the operation mode of the pins passed in `pinlist`. The driver trailing edge will be set to the expected pass value `start`, while the format will be set up according to the table given below. The leading edge may be auto-corrected to ensure the `min_pulse_width` specified (see ESGB). The passed/failed test returns a `res_spec` of either P (pass) or F (fail). The returned value will be the passed `start` value.

A value test is done by varying the driver trailing edge of all pins passed in `pinlist`. The first measurement, done at the expected pass value `start` is used to select either a binary search (test failed) or linear search (test passed) algorithm to measure the DUT pins hold time. Again, each test done may consist of two functional tests (see table below). The linear search will be done, decreasing the edge delay by `lin_bin_step` until the first test fails or the hardware programming limit (0) is reached. An additional binary search improves the resolution of the returned result value. The binary search will be done in the interval `limit` through `start` to measure the hold time. The leading edge may be auto-corrected during the search, to ensure the `min_pulse_width` specified (see ESGB). The returned `res_spec` will be one of:

EQ if the hold time could be measured
LE if all tests passed
GT if all tests failed

The allowed value range for the `start` parameter depends on the configuration of the pins passed in `pinlist`

min_pulse_width .. 2 * period - 1 ns (no MUX pins)
min_pulse_width .. period (at least one MUX pin)

The `limit` parameter, if specified, must be in the range

start .. 2 * period - 1 ns (no MUX pins)
start .. period (at least one MUX pin)

Table 7-3. Driver Formats for Hold Time Tests

Pin Mode	Pin Type	Format	1st *pass*	2nd pass
STD	i,i/o	DNRZ,R1,RZ	RC	
	i,i/o	RI	unchanged	
	ioh,iol	R1,RZ,RI	unchanged	
	ioh	DNRZ	R1	
	iol	DNRZ	RZ	
MUX	i,i/o	DNRZ,R1,RZ	RZ	R1
	ioh,iol	R1,RZ,RI	unchanged	
	ioh	DNRZ	R1	
	iol	DNRZ	RZ	
FQ2	i	DNRZ,RZ,R1	RZ	R1

In a multiple mainframe configuration, only the master mainframe responds to the THLD? query. Pins in `pinlist` must be configured as I, IO, IOL, IOH, or NC pins. IX pins must not be included in the pinlist. Exceeding the value ranges for the `start` and `limit` parameters causes an error message to be sent to the host and no test to be run. FD, and FQ pins are not allowed to be part of `pinlist`.

Example. The following example shows how to use these queries to measure the setup and hold time of a 100 MHz DUT input pin (DNRZ), assuming a tester period of 100 ns and expected pass values of 2 ns (tsu) and 3 ns (thld). We further assume a reference signal (tref) at 10 ns.

        TSUP? 8,,(pin)

Setup Time Pass/Fail Test. The expected pass value is (tref - tsu). The command returns "TSUP P,8" if the test passed, otherwise "TSUP F,8" will be returned.

        THLD? 13,,(pin)

Hold Time Pass/Fail Test. The expected pass value is (thld - tref). The command returns "THLD P,13" if the test passed, otherwise "THLD F,13" will be returned.

        TSUP? 8,0,(pin)

Setup Time Value Test. We assume that (pin) is a DNRZ pin, so `limit` can be set to 0. For Rx format pins, `limit` should be set to the programmed trailing edge value minus the actual period if the programmed delay is greater than the

period. If the returned `rel_op` is EQ, the setup time can be derived from the returned value : tsu = tref - `return_value`

> THLD? 13,108,(pin)

Hold Time Value Test. Limit is set to period plus the programmed leading edge delay. If the returned `rel_op` is EQ, the hold time can be derived from the returned value : thld = `return_value` - tref

Pulse / Pause Width Measurement

TPWM? `le_delay`, `te_delay`, `pinlist`

Return string:

TPWM `rel_op`, `min_pulse_width`, `rel_op`, `min_pause_width`

The TPWM? command is used to measure the minimum pulse- and pause-width of any number of DUT input pins.

The value test is done by varying the driver trailing edge of all pins passed in `pinlist`, using a binary search algorithm. The binary search is performed in the interval `te_delay` through (`le_delay` + 0.5 ns) to measure the minimum pulse width and in the interval `te_delay` through `le_delay` + pin_period to measure the minimum pause width.

Only R1, RZ, RI or RC format pins are allowed to be part of `pinlist`. The passed `le_delay` and `te_delay` setups must be in the range described in the timing setup section (see the description of the DRTM command). Note that the value ranges for MUX pins are used if one or more pins `pinlist` are configured as MUX mode pins.

The returned `rel_op` will be one of

> EQ if the min. pulse (or pause) width could be measured
> LE if all tests passed
> GT if all tests failed

In a multiple mainframe configuration, only the master mainframe responds to the TPWM? query. Pins in `pinlist` must be configured as I, IO, IOL, IOH, or NC pins. Exceeding the value ranges for the `le_delay` and `te_delay` parameters causes an error message to be sent to the host and no test to be run.

Input / Output Level Measurement

Two commands (DRLS?, RCLS?) are available to measure the input level sensitivity of a DUT input pin and the min./max. output levels of a DUT output pin. Both commands are used for pass/fail and value tests.

V_{IH}, V_{IL} **Pass/Fail and Value Test.** The DRLS? command returns a passed/failed result of a V_{IHmin} or V_{ILmax} test, if `limit` has been omitted. The test is done by a single functional test, setting the driver high (`level_select` = H) or low (`level_select` = L) level of all pins in `pinlist` to the expected `pass_value`. The corresponding high or low level may be auto-corrected to avoid violating the driver swing limitations (see DRLV). The passed/failed test returns a `res_spec` of either P (pass) or F (fail). The returned value will be the passed `pass_value`.

A value test is done by varying the driver high (`level_select` = H) or low (`level_select` = L) level of all pins passed in `pinlist`. For value tests, `pass_value` gives the high level (or low level) at which a functional test is expected to pass. In general, the value passed will be the programmed value.

`Limit` specifies the binary search limit which will in general be the programmed low level (`level_select` = H) or the programmed high level (`level_select` = L). The binary search will be done in the interval `limit` through `pass_value`. During the search, the corresponding high or low level may be auto-corrected to avoid violating the driver swing limitations (see DRLV). The returned `res_spec` will be one of:

    EQ if VIHmin or VILmax could be measured
    GT if all tests failed (`level_select` = H)
    GE if all tests passed (`level_select` = L)
    LE if all tests passed (`level_select` = H)
    LT if all tests failed (`level_select` = L)

The allowed value ranges are

`pass_value` : -4000 mV .. +8000 mV
`limit` : -4000 mV .. `pass_value` (H) and `pass_value` .. +8000 mV (L)

In a multiple mainframe configuration, only the master mainframe responds to the DRLS? query. Pins in `pinlist` must be configured as I, IO, IOL, IOH, or NC pins. IX pins must not be included in the pinlist. Exceeding the value ranges for the `pass_value` and `limit` parameters causes an error message to be sent to the host and no test to be run.

V$_{OH}$, V$_{OL}$ Pass/Fail and Value Test. The RCLS? command returns a passed/failed result of a V$_{OHmin}$ or V$_{OLmax}$ test, if `limit` has been omitted. The test is done by a single functional test, setting the receiver high (`level_select` = H) or low (`level_select` = L) level of all pins in `pinlist` to the expected `pass_value`. The corresponding high or low level may be auto-corrected to avoid violating the receiver swing limitations (see RCLV). The passed/failed test returns a `res_spec` of either P (pass) or F (fail). The returned value will be the passed `pass_value`.

A value test is done by varying the receiver high (`level_select` = H) or low (`level_select` = L) level of all pins passed in `pinlist`. For value tests, `pass_value` gives the high level (or low level) at which a functional test is expected to pass. In general, the value passed will be the programmed value. `Limit` specifies the binary search limit which will in general be the programmed low level (`level_select` = H) or the programmed high level (`level_select` = L). The binary search will be done in the interval `hardware_limit` through `limit`. During the search, the corresponding high or low level may be auto-corrected to avoid violating the receiver swing limitations (see RCLV). The returned `res_spec` will be one of:

EQ if VOHmin or VOLmax could be measured
GT if all tests failed (`level_select` = L)
GE if all tests passed (`level_select` = H)
LE if all tests passed (`level_select` = L)
LT if all tests failed (`level_select` = H)

The allowed value ranges are

`pass_value` : -4000 mV .. +8000 mV
`limit` : -4000 mV .. `pass_value` (H) and `pass_value` .. +8000 mV (L)

In a multiple mainframe configuration, only the master mainframe responds to the RCLS? query. Pins in `pinlist` must be configured as O, IO, IOL, IOH, TERM, or NC pins. Exceeding the value ranges for the `pass_value` and `limit` parameters causes an error message to be sent to the host and no test to be run.

Example

Assuming a DUT with one input (Din) and one output (Dout) pin, we want to measure the input level sensitivity and the maximum and minimum output levels. We further assume that the tester configuration and vector setup has been already done, such that a functional test will pass.

The (hypothetical) specs of the device are as follows :

Parameter	Value
Vil (Din)	$\leq$1.13 V
Vih (Din)	$\geq$1.48 V
Vol (Dout)	$\leq$0.98 V
Voh (Dout)	$\geq$1.63 V

The following firmware command sequence is used to measure the DUT parameters :

```
DRLV -1700,-800,(Din)
RCLV -1300,-1250,(Dout)
```

Setup pin levels such that a functional test will pass.

```
DRLS? H,-1130,,(Din)        Vih
DRLS? L,-1480,,(Din)        Vil
RCLS? H,-980,,(Dout)        Voh
RCLS? L,-1630,,(Dout)       Vol
```

All queries shown above will return P,value if the functional test passed, otherwise F,value will be returned.

```
DRLS? H,-800,-1700,(Din)        Vih
DRLS? L,-1700,-800,(Din)        Vil
RCLS? H,-1250,-1300,(Dout)      Voh
RCLS? L,-1300,-1250,(Dout)      Vol
```

If the returned rel_spec is EQ, the returned value of each test gives the requested level (V_{ih}, V_{il}, V_{oh}, or V_{ol}) without any further conversions.

High Resolution Timing Diagram

The commands described in this section are provided to support the High Resolution Timing Diagram. There are four commands in this section:

HRTP   sets up the parameters for the high resolution timing diagram
HRTP?   reads the current parameters set for the high resolution timing diagram
HRTM   starts a high resolution timing diagram measurement
HRTR?   gets the results of the last HRTM measurement
HRTL?   reports the limitations imposed on the high resolution timing diagram by the hardware.

A high resolution timing diagram test is done by collecting the DUT data sampled at various sampling points distributed evenly over the tester period. The number of sampling points depends on the required resolution. The samples are taken in edge compare mode, varying the comparator leading edge.

Because the test requires that more than one test be run, the sequencer will be restarted several times at the defined start label. You should ensure that all external conditions affecting sequencer program flow behave the same way for all tests.

Note	An unstable or missing external phase sync signal will cause the system clock to run out of phase, if external phase synchronization has been selected (see SCLK, SYNC, and SNCC). No error messages or warnings will be issued if this occurs.

During the test, the tester timing setup will be modified. When the test terminates, the original values will be restored. The last activity of the HRTM command is to perform a functional test in DA mode, with the original settings.

Aborting the command using a Device Clear message (SDC) aborts the test currently running and restores the original setup values. In this case, no functional test will be performed.

Syntax

```
HRTP cycles, cycle-offset, samples-per-cycle, samples,
sample-offset

HRTP?

HRTM pin-list

HRTR?
```

returns

```
HRTR pin, #spec result
```

where

spec = digit {digit}

result = resultchar {resultchar}

resultchar = 0 | 1 | I | ?

```
HRTL?
```

returns

```
HRTL no_of_pins, no_of_samples
```

Finding the Limitations Imposed by the Hardware

The HRTL? query reports the maximum number of pins, and the maximum number of samples which may be measured in parallel by a single HRTM command, and also the maximum number that can be specified in the HRTP command.

While the number of samples is fixed at 1024, the number of pins depends on the clock board revision and the tester model. With the 512 Kbyte firmware memory clock board, the maximum number of pins is 32 pins for the D200; and 4 pins for the D50. With the 2 Mbyte clock board, 32 pins are available for all models (D50, D100, D200 and D400)

Defining the Test Function Parameters

The test function parameters are defined with the HRTP command. The current settings can be read using HRTP?. This query will only be answered by the master mainframe.

```
HRTP cycles, cycle-offset, samples-per-cycle, samples,
sample-offset
```

cycles	specifies the number of cycles to be examined in high resolution mode. It may be any integer value greater than zero taking the sampling limits, given below, into consideration. Regardless of the current operating modes for the pins in `pin-list`, the term `cycles` always refers to sequencer cycles (100 MHz).
cycle-offset	specifies the first cycle to be examined. It is given relative to the start of the analyzer delay counter and may be negative.
samples-per- cycle	specifies the timing resolution by defining the number of samples (> 0) to be taken in each cycle.
samples	specifies the number of samples to take for the next HRTM command
sample- offset	specifies the first sample. It is given relative to the first sample in the cycle and can be in the range 0 .. (`samples_per_cycle` - `samples`).

Value range errors will occur if:

```
cycles < 1
(cycles × samples) > 1024
samples > (samples_per_cycle - sample_offset)
```

Running the Test Function

The test function is started for the pins in `pin-list` using the HRTM command. The number of tests that is actually performed is determined by the `samples` variable specified in the last HRTP command. If an error occurs, no results will be available and the number of tests actually performed is not specified.

Errors will occur if:

- The tester is not in acquisition mode (see RCMD)
- The requested number of cycles after `cycle-offset` are not available
- The number of pins in `pin-list` > 32

■ `pin-list` contains non-applicable pins (`I`, `DC`, or `OFF`)

In this case, the results will still be available for all applicable pins.

Getting the Test Results

The HRTR? query returns the results of the last HRTM executed. A blank line (line-feed only) will be sent if no test has yet been performed since power-on or the last HRTP command. No response will be sent for a pin whose configuration has been changed since the last measurement.

The values of `resultchar` have the following meanings:

0	a level below the lower threshold was detected
1	a level above the upper threshold was detected
I	an intermediate level was detected (not returned for 200 MHz pins)
?	the level detected was below the pin's lower threshold and above the upper threshold. This can only occur if the pin is not calibrated correctly (not returned for 200 MHz pins).

Examples

If the period has been set to 100 ns, the following set of commands can be used to obtain a high resolution timing diagram for 10 cycles starting at cycle 0, with a resolution of 1 ns.

```
HRTP 10,0,100,100,0
HRTM (Q0,Q1,Q2,Q3)
HRTR?
```

To perform a test without initialization cycles, the following set of commands can be used (assuming a test length of 1000 cycles).

```
HRTP 10,-1000,100,100,0
HRTM (Q0,Q1,Q2,Q3)
HRTR?
```

To run a test with a resolution exceeding the maximum number of samples (say 100 picoseconds), it has to be split as in the following example.

```
SCLK ,300,              first part
HRTP 1,10,3000,100,0
HRTM (Q0,Q1,Q2,Q3)
HRTR?
```

```
HRTP 1,10,3000,100,100   second part
HRTM (Q0,Q1,Q2,Q3)
HRTR?
HRTP 1,10,3000,100,200   third part
HRTM (Q0,Q1,Q2,Q3)
HRTR?
```

DC Measurement Commands using the PMU

There are six commands for performing dc measurements using the PMU. These are:

DCGB   sets the global dc measurement modes
DCGB?   retrieves the global dc measurement modes
DPAR   defines a set of dc measurement parameters
DPAR?   retrieves a set of dc measurement parameters
DTST   performs a dc measurement
DCPR?   retrieves the result of pin-related dc measurements
DCSR?   retrieves the result of non-pin-related dc measurements

Syntax

```
DCGB [meas-mode], [connect-mode], [pre-action]

DCGB?

DPAR set-id, [connect-u], [force-u], [force-i], [time]

DPAR? set-id

DTST set-id, pinlist

DCPR? id

DCSR? id
```

The DCGB command defines the global parameters for the next dc measurement. The parameters:

meas-mode   specifies whether the measurement will be performed in
             parallel or serial. It can be
             ■ P - parallel. Only one measurement per mainframe is
               performed. All pins of one mainframe are connected
               simultaneously to a PMU and the combined current and
               voltage is measured. The result is available using DCSR?
             ■ S - serial. Every pin is measured alone. The result is available
               using DCPR?. It is calculated taking the value of the Rs
               parameter of the PATR command into consideration.
             ■ D - direct. Every pin is measure separately. The result is
               available via the DCPR? query. It is calculated at the tester
               pin.

             This parameter greatly influences the speed of the measurement
             and the quality of the results available after the measurement.
             Parallel measurements, of course, will not generate individual
             per-pin results.

connect-mode specifies how the PMU is connected to the pin. It can be one of
             ■ A - additional. The PMU is also connected to the IO channel
               during the measurement. In this case, any AC electronics on
               the IO board can affect the measurement and can be used to
               measure the DUT pins under loaded conditions.
             ■ E - exclusive. The PMU is connected to the DUT only. The IO
               board is disconnected from the DUT for the duration of the
               measurement.

pre-action   specifies the actions, to put the DUT into a state desired for the
             measurement. It can be
             ■ NONE - nothing is done.
             ■ FTST - a functional test, as described by the FTST command,
               is performed before the dc measurement.

At Power On it is defined as: DCGB S,E,NONE

Setting the Parameters for a DC Measurement

The DPAR commands specifies the parameter sets for the dc measurements.
Multiple parameter sets can be held simultaneously in the firmware.

The parameters of this command are:

set-id   this integer selects the set to specify. This parameter is
             mandatory. It must be in the range 0 .. 1.

connect_u	is a fixed point number specifying the force voltage of the PMU in millivolts. It must be in the range -20000 (mV) .. +20000 (mV). If the value given exceeds this range, it will be set to the limit. This voltage will be applied before the PMU is connected to the pin. The parameter allows reduction of the current required to charge the DC rail to the level of the DUT pin. Only devices that are sensitive to current peaks caused by this charging effect need this parameter to be set. Other devices will normally set this value to force_u.
force-u	is a fixed point number specifying the force voltage of the PMU in millivolts. It must be in the range -20000 (mV) .. +20000 (mV). If the value given exceeds this range, it will be set to the limit. The PMU will force this voltage as long as current specified by the force-i parameter is not exceeded. The voltage range of the PMU is automatically selected for best fit. If the dc measurement aims to a voltage measurement, this parameter should be specified to the highest absolute voltage expected to prevent PMU overflow conditions.
force-i	is a fixed point number specifying the force current of the PMU in microamperes. It must be in the range 0 (μA) .. 500000 (μA). If the value given exceeds this range, it will be set to the limit. The current range of the PMU is automatically selected for best fit. The PMU will never generate a absolute current above this value. If the dc measurement aims to a current measurement, this parameter should be treated as clamp current, the current is limited by the PMU.
time	is a fixed point number specifying the wait time used before a dc measurement in milliseconds. This time is measured from the point the DUT is connected to the PMU till the PMU read out. It is to allow charging of capacitive loads on the DUT wiring without influencing the measurement results. The firmware takes care for any internal capacitive loads, so the user need not compensate for them.

All values in this command can be given in an arbitrary precision. However, the firmware will map them to the accuracy of the available hardware.

At Power On all sets are defined as: DPAR x,0,0,0

Retrieving Measurement Results from a DTST Command

Two commands are available for retrieving DC test results. The DCPR query retrieves the measured values of the last DTST command for each pin (if the

test was performed per pin), and DCSR? gives the result per pinlist. The parameter

set-id   selects, which pins are reported. All pins measured for this
              set since the most recent DPAR command for this set will be
              reported.

The DCPR? query responds with the syntax:

    DCPR id, valid-code, measured-u, measured-i, pin-list

and the DCSR? query responds with the syntax:

    DCSR id, valid-code, measured-u, measured-i

where

valid-code   This integer describes the condition of the PMU during the
              measurement. It is in the range 0 .. 63. If valid-code is coded
              binary it has the format OFGppppp where
              ■ ppppp is the pmu-state as described in the PMUM? query.
              ■ G is the force-i overflow flag. It is set when the PMU could
                not be set to the necessary force current.
              ■ F is the functional fail flag. It is set if preceding functional
                test result in a fail.
measured-u   is the measured voltage given in millivolts. It is only valid, if
              the valid-code indicates neither unstable nor voltage overflow.
measured-i   is the measured current given in microampers. It is only valid,
              if the valid-code indicates neither unstable nor current
              overflow.
pinlist   specifies the pins the former parameters are related to.

After Power On no DTST command has be executed and the query will not respond for any pin.

Performing a DC Measurement

The DTST command will perform a dc measurement according to the current setup. The parameters of the command are

set-id   specifies the parameter set to use for result generation. The
              parameter set is specified by the DPAR command.

`pinlist`   specifies the pins to measure.

The following pseudo "C" should clarify how the different DCGB parameters will influence the DTST command.

```
switch (pre-action){
case NONE:
  break;
case FTST:
  perform-functional-test;
  break;
}
  PMU.voltage=force-u[set-id];
  PMU.current=force-i[set-id];
switch mode{
case S /* serial */
  for (pin in pins-to-measure){
    connect pin   to PMU;
    wait time;
    pin.valid-code=PMU.state | FTST-result ;
    pin.measured-i=PMU.i;
    pin.measured-u=PMU.u;
    restore-connectivity-of-pin;
  }
  break;
case P /* parallel */
  connect all pins-to-measure to PMU;
  wait time;
  valid-code=PMU.state | FTST-result | PMU-current-overflow;
  measured-i=PMU.i;
  measured-u=PMU.u;
}
```

Note	Before executing DTST?, the relays of the IX pins included in the pin list must be switched to DC DCB or DCC using the RLYC command.

DC Measurement Details

For a detailed understanding of things going on during a dc measurement the following notes will give some hints. Every detail not defined here should be treated as undefined and even if a specific behavior seems to be stable it should not be relied on. A non documented behavior may be observed not in all case and/or may change without notice.

Relay Setting

When a DTST command is accepted the `pin-mode` (see RLYC command) of all pins of the system must be either IDLE or AC. These `pin-modes` will have been restored when the command has finished. Any other `pin-mode` of any pin in the system may result in an inaccurate measurement or even in damage of the DUT.

Measurement Sequence

The sequence, the pins of DTST command are measured, is arbitrary. The firmware will neither define any particular sequence nor that two identical DTST commands will measure in the same sequence. Any measurement, sensitive to a specific measurement sequence, should be performed by separate DTST commands.

PMU Usage

The firmware will neither specify which of the available PMUs is used for a specific pin nor that two identical DTST command will use the same PMU for a specific pin. Any measurement, sensitive to the use of a specific PMU, should be performed by low level PMU commands (PMUS, PMUM?).

PMU Ranges

The DTST command will select a PMU range for voltage and current which fits best for specified force values. During a measurement the ranges of the PMU will be adjusted to achieve the best resolution available or to prevent an overflow condition. This method assumes that the signal of the measured pin is stable over the whole measurement sequence, and may result in marginal current drops. Any measurement sensitive to the use of a specific range or not satisfying the stated assumptions, should be performed by low level PMU commands (PMUS, PMUM?).

DC Measurement Commands using the DPS

The supported power supplies have the capability of reporting the actual
voltage and current being supplied. The accuracy of this measurement is not
very high, but it may be useful in some applications.

    PSME? pin-list

This command returns the following string

    PSME status, voltage, current, pin

where

status   reports the output state of the power supply. It can be either
                CV or CC for constant voltage and current respectively.
voltage   is a real number indicating the voltage at the DPS output.
                The polarity of the voltage is taken from the relevant DFPS
                command.
current   is a real number indicating the current flowing at the DPS
                output.
pin   specifies the DPS output to be read

Errors

An error will be generated if there are no power supplies defined.

Low Level Test Commands

The low-level HP-IB commands allow you to design test functions with
increased flexibility.

The other, high-level HP-IB test commands combine the operations of several
low-level commands, to improve programming convenience. However, under
some circumstances, your test application may require additional manipulation
of the tester hardware, which can only be carried out using these low-level
operations.

Low-level commands are provided for:

- setting the pin connection relays;
- setting the DPS relays;

- setting the PMU relays;
- controlling the vector sequencer;
- setting the utility lines.

General Considerations

If you do use low-level commands, you must be aware of the following considerations:

- Low-level commands interfere with the operation of high-level commands in many ways. You must have a detailed understanding of the operations being carried out in the tester hardware, *before* you combine high-level and low-level commands.
- Low-level commands perform only a rudimentary check of the parameters that you enter, and they do not warn you if the system is put into a state which could cause damage to the tester or DUT.
- Some low-level commands do not have corresponding queries associated with them. (so you might not be able to check the present hardware state).
- Values for low-level commands are not defined at power-up. The power-up states are controlled by the high level commands implicitly.

Relay Commands

RLYC controls relays connected to pins and resources on the DC rail. Relays are in the tester for mainly two purposes:

- connect the pin electronics to the DUT (AC-relay)
- connect the DC measurement equipment (internal or external) to the DUT

PSST controls the relays that connect the DPS to the DUT.

While the AC connections can be made for each pin independently from other connections, DC connections share common resources. There is only one common DC line on each channel board and there are only two DC rail on a mainframe.
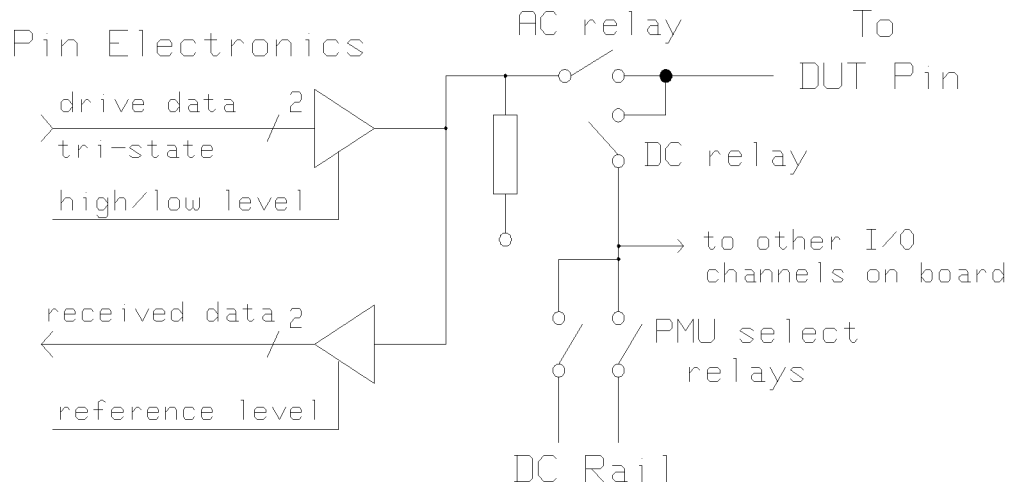
Figure 7-1. DC Wiring Schematics

These impose that

■ the pins of any one I/O board may only be connected to the same DC
  equipment
■ external DC equipment can only be used as an alternative to PMU1
■ pins can only be connected to DC equipment on the same mainframe

With these restrictions in mind we come to a command that is based on the idea
of connecting a pin to the DC equipment by specifying the setting of

■ the pin relays and
■ the Common DC relays

independently.

Syntax:

    RLYC [pin-mode], [dc-target], pinlist

The parameters:

pin-mode   how a pin is connected to the tester The token
                ■ IDLE will disconnect the pin from any tester electronics.
                ■ AC will connect the pin to the pin electronics only.

- DC will connect the pin to the specified `dc-target` only. With IX pins, the non-inverted output channel of the HSWG is connected.
- BOTH will connect the pin to the pin electronics and the specified `dc-resource`.
- DCC connects the inverted (complementary) output of an HSWG channel to `dc-target`, when measuring an IX pin.
- DCB connects both the inverted and non-inverted outputs of an HSWG channel (in parallel) to `dc-target`, when measuring an IX pin.

`dc-target`   specifies to what dc resource the pin should be connected. The token
- PMU1 will connect the boards affected by the given pins to PMU1.
- PMU2 will connect the boards affected by the given pins to PMU2.
- EXT will connect the boards affected by the given pins to the external input at the clock board.
- OFF will disconnect the board specified by the given pins from the both dc rails.
- X will not change the relays associate with dc resources at all.

`pinlist`   The setting of `pin-relay` will only occur on the pins specified by here. The setting of dc resource related relays will be done an all mainframes, if the command is send to all mainframes (as usual). However the computation of "affected" and "unaffected" boards is done on the basis of the `pinlist`.

This command will support either the setting of only pin relays or Performing your own connection routine, so that you can either route a signal path from a dc-target to one or more pins with or without simultaneous setting of pin relays for normal serial or parallel DC measurement, or for precharging the rails. A definition of all relay settings is given below.
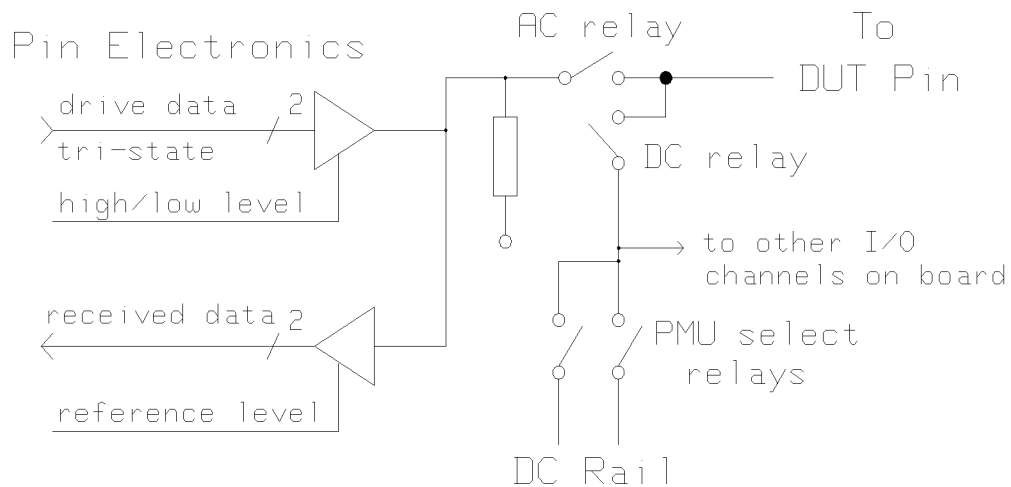
Figure 7-2. Relay Switching Matrix

Note	You should use this command with care as some relay settings may lead to unwanted side effects.

Power Supply Relay Commands

The PSST command is used to connect and disconnect the DPS to/from the DUT, the PSST? query reports the current state of the DPS outputs.

        PSST status PSST?

where

status   can be either ON or OFF to indicate whether the DPS is
                connected or not.

The PSST command changes the state of the defined DPS outputs to the states defined in the relevant PSLV commands. If the state is changed to OFF, the setup time is used as a delay. The power supply outputs are switched in the reverse order to that used when they were switched on.

Errors

■ Parameter error

■ No power supplies connected

Warnings

Warnings will be issued if

■ PSST OFF is executed and a power supply output is in the CC mode.
■ PSST OFF is executed and the power supply is in the power on state.

Examples

PMU measurement in parallel mode:

```
PMUS P11, ...
RLYC PMU1,DC,(Pin1,Pin2, ...)
/* wait for charging parasitic Capacities, if necessary */
PMUM? 1
RLYC IDLE,DC,(Pin1,Pin2, ...)
```

PMU measurement in serial mode:

```
for Pin-i in (Pin1,Pin2, ...)
do
  PMUS P11, ...
  RLYC PMU1,DC,(Pin-i)
  /* wait for charging parasitic capacities, if necessary */
  PMUM?      1
  RLYC AC,X,(Pin-i)
done
```

Sequencer Commands

The SQSS command sets a software signal to the sequencer according to its parameter. A sequencer program can test for this condition and branch on it.

The SQSS?-query retrieves the state of the software signal.

Syntax

```
SQSS boolean

SQSS?
```

Errors

■ Value out of range

PMU Commands

There are two commands used to control the PMU. These are:

PMUS   Puts PMU into specified setup
PMUM?   Measures state of PMU

Syntax

```
PMUS pmu-id, u-force, [u-range], i-force, [i-range]

PMUM? pmu-id
```

Setting up the PMU

The PMUS command setups a parametric measurement unit installed in the test system. The PMU is put into the specified setup during the execution of the command. It stays in this setup as long as it is not needed by other commands such as dc measurement or calibration commands. The parameter:

pmu-id   specifies the PMU to setup. It has the form Pmp where m
                specifies the mainframe the PMU is installed in and p the PMU
                inside the mainframe. m must be in the range 1 .. 3 and p must
                be in the range 1 .. 2.
u-force   specifies the voltage to force by the PMU. It is given in
                millivolts. It must be in the range -20000.000 (mV) ..
                +20000.000(mV). The value can be given in any precision, but
                it will be rounded according to the resolution of the PMU in the
                specified range. PMU resolution is 1/2000 of the specified range.
u-range   specifies the the voltage range to use. The meaning of the
                ranges are:

Table 7-4. PMU Voltage Range Coding

Range	Meaning
U2E3	-2000 mV .. +2000 mV
U1E4	-10000 mV .. +10000 mV
U2E4	-20000 mV .. +20000 mV
AUTO	autoranging: the smallest range to force u-force

i-force   specifies the current to be forced by the PMU. It is given in
                microamperes. It must be in the range 0.000 (μA) .. 200000.000
                (μA). The value can be given in any precision, but it will be
                rounded according to the resolution of the PMU in the specified
                range. PMU resolution is 1/4000 of the specified range.

Table 7-5. PMU Current Range Coding

Range	Meaning
I2E0	0.001 μA .. 2 μA
I2E1	0.01 μA .. 20 μA
I2E2	0.1 μA .. 200 μA
I2E3	1 μA .. 2000 μA
I2E4	10 μA .. 20000 μA
I2E5	100 μA .. 200000 μA
I5E5	200 μA .. 500000 μA
AUTO	autoranging: the smallest range to force `i-force`

This command has no query associated, because it interferes with internal dc measurement commands and the a query response cannot be guaranteed to be valid.

Making a PMU Measurement

The PMUM? query measures a PMU and give the result of the measurement. The parameter

pmu-id   specifies the PMU to setup. It has the form `Pmp` where `m` specifies the mainframe the PMU is installed in and `p` the PMU inside the mainframe. `m` must be in the range 1 .. 3 and `p` must be in the range 1 .. 2.

The query has the following syntax:

    PMUM pmu-id, pmu-state, u-read, i-read

where

pmu-id   to specify the PMU measured. This parameter is identical to the query given.
pmu-state   is an integer which specifies the state of the PMU. It is in the range 1 .. 31. If `pmu-mode` is coded binary is has the form `Ovismm` where

- **v** is voltage overflow flag. It is set when the PMU is in an overflow condition during the voltage measurement.
- **i** is current overflow flag. It is set when the PMU run in a overflow condition during the current measurement. (NOTE: Will not occur with current PMU)
- **s** is the stability flag. It is cleared when the PMU is stable, set otherwise.
- **mm** is limit detection code.

Table 7-6. PMU Limit Detection Coding

mm	Meaning
00	not used
01	negative current limit
10	positive current limit
11	voltage limit

It gives the section of the I/U characteristics in which the PMU is operating.
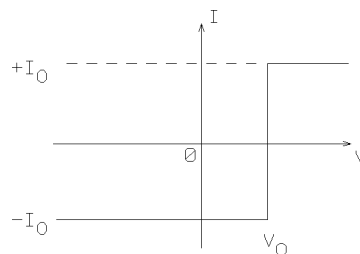
Figure 7-3. PMU I/U Characteristics

`u-read`   gives the measured voltage in millivolts.

`i-read`   gives the measured current in microamperes.

Errors

- Value out of range
- Resource not available

Utility Lines

There are three commands to set and read the state of the system utility lines.
These commands are:

UTOT   define state of utility output
UTOT?   get state of utility output
UTIN?   get state of utility input

Syntax

```
UTOT utility-data

UTOT?

UTIN?
```

Setting the Utility Output Lines

The UTOT command will set the seven utility output to the value specified by
`utility-data`. `utility-data` must be in the range 0..127.

Reading the Utility Output Lines

The UTOT? query will retrieve the current state of the utility output port.

Reading the Utility Input Lines

The UTIN? query will read the current state of the utility input port. The state
of the port is given/read as binary coded integer.

Power-On Defaults

At Power On the utility output port will set to:

`UTOT 0`

Example

```
    UTOT 127
```

This command sets all the output utility lines to logic high.

High Throughput Commands

These commands allow you to write test routines for applications where you need to perform the same sequence of tests on a number of devices of the same type. By using the high throughput commands, you can minimize the amount of time taken for each device to be tested. Incoming inspection, and quality control testing are typical of these applications.

By preloading several sets of test parameters into RAM (local to the hardware), you can rapidly execute a sequence of different tests on your DUT, simply by switching between sets of parameters. In this way, the High Throughput commands reduce the amount of data transferred between the PWS and the tester hardware during your test routine.

A full description of how to use these commands to optimize the tester throughput, is given in the manual *Advanced Testing with the HP 82000*.

The High Throughput commands are:

PSAV	to save the hardware state as a set of ac test parameters (a *parameter set*) in RAM;
LDHW	to reload a pre-saved parameter set to hardware;
LDHW?	to read parameter sets already loaded in hardware;
STST?	to perform a functional test using a parameter set;
DFVM	to define a set of dc parameters for a PMU voltage measurement;
DFCM	to define a set of dc parameters for a PMU current measurement;
DFPM?	to read pre-saved dc parameters;
PTST?	to execute a PMU test from pre-saved dc parameters.
RSWM	to set the switching methods of the AC and DC relays.
RSWM?	to examine the relay switching method that has been set.

Performing AC Functional Tests

Saving a Parameter Set

The PSAV command saves the *foreground* settings of the tester hardware, into RAM, and assigns an id number (between 1 and 64) to enable these settings to be quickly reloaded during a test.

The foreground settings are the values that you enter using the interactive windows, or via the other HP-IB commands.

When you execute a PSAV command, with the instruction:

`PSAV` *id*

the status of the following hardware settings are saved:

- Pin Configuration

  □ pin operating mode (STD, FD, MUX, etc . . .)
  □ pin type (I, IO, IX, etc . . .)
  □ scan mode (parallel or serial)
  □ channel relay setting
  □ DPS status (on or off)

- Level Setup

  □ driver low level
  □ driver high level
  □ receiver low level
  □ receiver high level
  □ DPS voltage
  □ DPS current

- Channel Timing

  □ driver signal format (DNRZ, RZ, etc . . .)
  □ receiver compare format (Edge, Window, etc . . .)
  □ leading edge delay
  □ trailing edge delay

- Clock Timing

  □ period
  □ clock source (internal/external)
  □ sync mode

- Vector Information

  □ break vector
  □ sequencer start label

Note   After a system reset, all 64 parameter sets are loaded with the foreground values.

Example. To save the present foreground settings as a parameter set with the id number 24, you can enter the instruction:

`PSAV 24`

Loading a Parameter Set

The command LDHW loads the hardware with a parameter set which you previously saved with PSAV. The id number which you specify (1 to 64) determines which parameter set is loaded. The id number "0" used in conjunction with LDHW, loads the hardware with the *foreground* settings.

The values loaded by the LDHW command are used by the hardware, until either:

■ you perform another LDHW command (a new parameter set is loaded);

■ you perform an STST command (a new parameter set is loaded);

■ you perform an HP-IB measurement command which changes any configuration settings (the foreground settings, id = 0, are reloaded);

■ you change the hardware settings explicitly, using HP-IB setup commands (the foreground settings are reloaded).

■ or you use any of the windows in the interactive software. (the foreground settings are reloaded).

When the foreground settings are reloaded into the hardware, the interactive windows of the PWS are immediately updated.

The query LDHW? returns the id number of the parameter set which is presently loaded in the hardware.

Example. To load the parameter set with the id number 24 into the hardware, you enter:

`LDHW 24`

To find out the id of the parameter set which is presently loaded in the hardware, you can enter the query:

`LDHW?`

If parameter set 24 is still loaded, the returned string would be:

`LDHW 24`

Executing a Functional Test

The command:

STST? *id*

performs a functional test using a presaved parameter set (saved with PSAV). The STST? command is functionally the same as:

LDHW *id*
FTST?

The return string from the STST? command contains the id number of the parameter set that has been used, and a pass (P) or fail (F) result.

Example. To perform a functional test with the parameter set id 24, you can enter the command string:

STST? 24

If the test result is a pass, the returned string is:

STST 24,P

Performing DC tests

The high throughput commands for performing dc tests allow you to store up to 64 different sets of dc parameters in RAM. These parameters can be used to perform either voltage measurements or current measurements with the PMU. The dc parameters are identified with an id number *id_dc* (which can be in the range 1 to 64).

Note	There is no foreground set for dc parameters.

Setting Up a PMU Voltage Measurement

The command DFVM allows you to define a set of dc parameters for a voltage measurement test with the PMU, and assign an id number *id_dc* to it.

The DFVM command allows you to define:

the identification number *id_dc* for the dc parameters (in the range 1 to 64);

the force-current to specify the PMU current to be used to simulate a load on the pin (in μA);

the clamp-voltage to specify the output voltage limit of the PMU during the measurement (in mV);

the max. and min. pass values to specify the pass voltage limits (in mV);

the connect voltage to specify the voltage forced by the PMU while the DC relay is connected. If this parameter is omitted, then the mean value between the max. and min. pass values is used.

the settling time to specify how long to wait between the connection of the PMU, and performing the measurement (in milliseconds);

and **the measurement mode**, which can be:

■ TRM (terminated)—the pin is connected to the PMU and the pin electronics;

■ NTRM (not terminated)—the pin is connected to the PMU only;

■ SPOL (specified polarity)—the measurement is taken with the polarity specified by the clamp voltage;

■ BPOL (both polarities)—measurements are taken with both polarities (a pass result is given if the pin passes the test in *either* direction).

Note   SPOL and NTRM are synonymous.

Example. To define the dc parameters for a PMU voltage measurement test, on pins Q1, Q2 and Q3 of your device, with

■ the id number (*id_dc*) "3";
■ a force current of 5 milliamps;
■ a clamp voltage of 5.5 Volts;
■ a minimum pass value of 4.5 Volts;
■ a maximum pass value of 5.0 Volts;
■ the default connect voltage (in this case, 4.75 V)
■ a settling time of 1.5 milliseconds;
■ with the PMU measurement carried out while the pin electronics are still connected;

you enter the following string:

```
DFVM 3,5000,5500,4500,5000,,1.5,TRM,(Q1,Q2,Q3)
```

Setting Up a PMU Current Measurement

The command DFCM allows you to define a set of dc parameters for a current measurement test with the PMU, and assign an id number id_dc to it.

With the DFCM command, you can set:

the identification number id_dc for the dc parameters (in the range 1 to 64);

the force-voltage to specify the PMU voltage to be used during the measurement (in mV);

the maximum current to specify the maximum current allowed during the measurement (in μA);

the max. and min. pass values to specify the pass current limits (in μA);

the connect voltage to specify the voltage forced by the PMU while the DC relay is connected. If this parameter is omitted, then the force-voltage value is used.

the settling time to specify how long to wait between the connection of the PMU, and performing the measurement (in milliseconds);

and **the measurement mode** which can be:

- GNG (ganged)—measurements are taken while all the pins are connected to the PMU;
- NGNG (not ganged)—each pin is measured individually.

Example. To define the dc parameters for a PMU current measurement test, on pins Q1, Q2 and Q3 of your device, with

- the id number (id_dc) "4";
- a force of voltage -1.7 Volts;
- a maximum current of 0.1 milliamps ;
- a minimum pass value of 20 microamps;
- a maximum pass value of 80 microamps;
- the default connect voltage (in this case, -1.7 V)
- a settling time of 1 millisecond;
- with the PMU measurement carried out on all three pins together;

you enter the string:

```
DFCM 4,-1700,100,20,80,,1,GNG,(Q1,Q2,Q3)
```

Reading Pre-saved DC Parameters

You can check the dc measurement settings already stored in RAM, with the DFPM? query:

DFPM? *id_dc*

If a voltage measurement has been defined for the *id_dc* number that you enter, the return string begins with DFVM, followed by a string of parameters which follow the syntax of the DFVM command.

For example, to check the dc parameters with the *id_dc* number "3", you enter the command string:

DFPM? 3

If *id_dc* "3" was defined as in the DFVM example above, you get the returned string:

DFVM 3,5000,5500,4500,5000,,1.5,TRM,(Q1,Q2,Q3)

Similarly, if a current measurement was previously defined for the *id_dc* number, the returned string begins with DFCM, and follows the syntax of the DFCM command. So, if the DFCM example above, was entered, the the command:

DFPM? 4

would give the return string:

DFCM 4,-1700,100,20,80,,1,GNG,(Q1,Q2,Q3)

Executing a DC Test

The command PTST? allows you to execute a test, using the dc parameters that you have previously entered with the DFVM and DFCM commands. The syntax for the command is:

PTST? *id_dc,level*

As well as specifying the *id_dc* of the dc parameter set to be executed, the syntax for PTST? allows a choice of test method. You can enter the test mode as:

- gpf (global pass/fail)—for a single pass/fail result;
- ppf (per pin pass/fail)—for a pass/fail result per pin;
- pval (per pin value)—for a value result per pin.

After you have executed the dc test, this is the string that is returned:

PTST *id_dc*, *result*, (*pin list*)

where:

- *id_dc* is the id number of the dc parameter set that has been used in the test;
- *result* is either P for a passed test, F for a failed test, or the measured value;
- *pin list* is a list of the pins to which *result* applies. If the test mode gpf was chosen for the test, then *result* applies to all the pins, and *pin list* is returned empty.

Setting the Relay Switching Method

The command:

RSWM method

allows you to specify the order in which the AC and DC relays are switched, when you perform a measurement using PTST? or DTST?. The method parameter can be either

- bbm - Break Before Make, i.e.
   1. all relays are opened,
   2. the relays needed for the measurement are closed.
- mbb - Make Before Break, i.e.
   1. the relays needed for the measurement are closed.
   2. the relays not needed for the measurement are opened.

8

Test Result Commands

The test result commands are used to load the results of one test into the host.

Commands and their Modes

The commands that are available depend on the measurement mode. The commands and their valid modes are listed below:

Commands valid for compare mode

ERMP?	returns the contents of the error map in a binary format.
CHER?	returns passed / failed information for every given pin.
ERCT?	returns the number of errors in the error map.
PASS?	returns a quick passed / failed information for the last test.

Commands valid for acquisition mode

RECD?	returns the contents of the data acquisition memory in a binary format.

Commands valid for both modes

VALD?	returns the number of valid data in the error map / received data ram.
AQST?	returns the number of cycle between the event address and the found acquisition stop, and returns a -1 if no event was found before stop.
GETV?	returns the vector and sequencer instruction number for the given cycles. Although the number of cycles to the logical 0 address is reported.

Except for the PASS? and CHER? commands, the commands return the results of a test of zero cycles length after changing the analyze mode or before at least one test has been started.

Note	RECD? and ERMP? work on byte boundaries.
	In the following descriptions, the term "cycle" is related to tester periods.

Syntax

```
ERMP? sample-start-cycle-#, #-of-cycles

CHER? pin-list

ERCT?

PASS?

RECD? sample-start-cycle-#, #-of-cycles, pin-list

VALD?

AQST?

GETV? cycle-#, #-of-cycles
```

Reading the Error Map

Sending the ERMP? command will result in the specified contents of the Error Map being transferred to the host. The data returned is in the format:

    ERMP sample-start-cycle-#, #-of-cycles, #spec(binary-data)

where

sample-start-cycle-#	is an integer in the range 1 .. size-of-error-map - 8 indicating the first cycle
#-of-cycles	is an integer in the range 1 .. size-of-error-map - 8 indicating the number of cycles to be read.

The actual test date is transmitted in an unreadable binary format.

Note that the error map will report only one error per machine cycle. Due to this cycle oriented error map, two errors occurring in the same machine cycle (but in distinct user cycles) are sampled as a single error and the error count (accessible using the ERCT command) is incremented by one (instead of two).

Getting Pass/Fail Information for a Pin

The CHER? query returns passed / failed information for pin-list. The value returned will be either P (passed) or F (failed) for every pin name in pin-list.

The CHER? query causes a temporary change to the sequencer OFF state to read the hardware channel error flags.

Getting the Number of Errors Detected in the Last Test

ERCT? returns the number of errors currently in the error map. The returned value will be an integer in the range 0 .. size-of-error-map - 4.

The reported error number returns only the number of errors sampled in the error map.

Getting pass/fail information for the last test

PASS? returns either P or F to indicate whether the last test passed or failed.

Reading the Data Acquisition Memory

RECD? returns the contents of the data acquisition memory in binary format.

Note	This command is only valid in data acquisition mode.

sample-start-   is an integer in the range (1 .. `vector-memory-size` - 8)
cycle-#   indicating the first cycle
#-of-cycles   is an integer in the range (1 .. `vector-memory-size` - 8)
                    indicating the number of cycles to be read.
pin-list   contains a list of pin names for which data should be read

The system replies in the format:

    RECD sample-start-cycle-#, #-of-cycles, #spec(binary data)

for each pin in `pin-list`.

Data transfer is performed in a by pin oriented manner, each byte containing information of 4 cycles for one pin. The contents of the data stream reflects the hardware RAM organization and will not be altered by the transfer commands.

`sample- start-cycle-#` and (`sample-start-cycle-#` + `#-of-cycles`) will be rounded to the next byte boundaries for binary transfers.

Reading the amount of data in the Error Map/ Received Data RAM

VALD? returns the number of valid logged data cycles in the error map / received data ram.

Reading how many cycles there were between the event address and acquisition stop

AQST? returns the number of cycles between the event address and the found acquisition stop, and returns a -1 if no event was found before stop. The returned value is an integer between -1 and 2^{24}.

From the start of the test, error map (in compare mode) and received data (in acquisition mode) will be monitored. After the event address is reached, the stop delay counter will be counted down and stop monitoring after the counter reaches the value 0. The firmware performs the conversion between the logical acquisition start address 0 and the sampling offset in the memory map and the received data memory. All reported values refer to the logical start address 0.

Mapping Machine Cycles to Sequencer Instructions

The GETV? query returns the cycle - vector relationship in a compressed form. It reports the cycle number after the logical 0 address, the vector number, the sequencer instruction number and gives the information how the next cycle was generated and how often (see examples).

Sending the command to the hardware will result in the hardware responding with:

GETV cycle-#, [vector-#], [instr-#], [instr-type], rep-factor

where

cycle-#	the actual cycle number in the range (0 .. last-cycle-number - 4)
vector-#	the actual vector number in the range (-1 .. size-of-vector-memory)
instr-#	an integer in the range 0 .. 2^{11} indicating the actual the actual instruction number.
instr-type	the type of the current instruction, one of INC, HOLD, SCAN or DEAD.
rep-factor	the repeat factor of the current instruction in the range 1 .. V-MEM - 4

Note that under some circumstances, the vector and instruction for a given cycle may be not available. In this case, the related parameters are omitted in the answer string. The break vector, which will be output during dead cycles, is shown by returning a vector number of -1.

Errors

- Parameter range errors
- Command not allowed in compare mode
- Command not allowed in acquisition mode
- Insufficient data available
- Asking for compare results in acquisition mode will cause an error, asking for received data in compare mode will also generate an error.

Examples

In this example of part of a vector sequence, the cycle-vector relationship will be reported as shown below.

```
cycle #   vector #  sequ instr # instr type
 800      ??           ??            ??
 more lines
 900      ??            0            inc
 more lines
1000       2            1            hold
1001       2            1            hold
1002       2            1            hold
1003       2            1            hold
1004      BREAK                 2         dead cycle
1005       4            2            inc
1006       5            2            inc
1007       6            2            inc
1008       7            2            inc
1009       8            2            inc
1010       9            2            inc
1011      10            2            inc
```

Sending GETV? 800, 210

will result in

```
GETV      800,,,,100
GETV      900,,0,INC,100
GETV      1000,2,1,HOLD,4
GETV      1004,-1,2,DEAD,1
GETV      1005,4,2,INC,6
```

Result Data Storage

The following section describes the data storage format used in the Error Map and Received Data memories.

Figure 13. Error Map / Received Data RAM alignment

Error Map Data Bit Alignment

D7	D6	D5	D4	D3	D2	D1	D0
(not used)				Cycle 1	Cycle 2	Cycle 3	Cycle 4
X	X	X	X	E0	E1	E2	E3

Received Data Bit Alignment (STD mode)

Cycle 0		Cycle 1		Cycle 2		Cycle 3	
D7	D6	D5	D4	D3	D2	D1	D0
E0/0	E0/1	E1/0	E1/1	E2/0	E2/1	E3/0	E3/1

where

En/0, En/1   is received data (as shown below)

Received Data States

En/0	En/1	received data
0	0	intermediate
0	1	low
1	0	high
1	1	don't care

Received Data Bits alignment (FD mode)

Cycle 0		Cycle 1		Cycle 2		Cycle 3	
D7	D6	D5	D4	D3	D2	D1	D0
E0.0	E0.1	E1.0	E1.1	E2.0	E2.1	E3.0	E3.1

where

En.0   received data (first user-cycle)
En.1   received data (second user-cycle)

Received Data Bit Alignment (MUX and FQ2 modes)

Cycle 0		Cycle 1		Cycle 2		Cycle 3	
D7	D6	D5	D4	D3	D2	D1	D0
E0.0/0	E0.0/1	E1.0/0	E1.0/1	E2.0/0	E2.0/1	E3.0/0	E3.0/1
. . .							
Ew.0/0	Ew.0/1	Ex.0/0	Ex.0/1	Ey.0/0	Ey.0/1	Ez.0/0	Ez.0/1
. . .							
E0.1/0	E0.1/1	E1.1/0	E1.1/1	E2.1/0	E2.1/1	E3.1/0	E3.1/1

where

En.0/0, En.0/1   received data (first user-cycle)
En.0/0, En.0/1   received data (second user-cycle)

Received Data Bit Alignment (FQ mode)

Cycle 0		Cycle 1		Cycle 2		Cycle 3	
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
E0.0	E0.1	E1.0	E1.1	E2.0	E2.1	E3.0	E3.1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.0	Ew.1	Ex.0	Ex.1	Ey.0	Ey.1	Ez.0	Ez.1
E0.2	E0.3	E1.2	E1.3	E2.2	E2.3	E3.2	E3.3
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.2	Ew.3	Ex.2	Ex.3	Ey.2	Ey.3	Ez.2	Ez.3

Where:

En.0   = received data (first user-cycle)
En.1   = received data (second user-cycle)
En.2   = received data (third user-cycle)
En.3   = received data (fourth user-cycle)

Received Data Bit Alignment (FQM mode)

Cycle 0		Cycle 1		Cycle 2		Cycle 3	
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
E0.0/0	E0.0/1	E1.0/0	E1.0/1	E2.0/0	E2.0/1	E3.0/0	E3.0/1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.0/0	Ew.0/1	Ex.0/0	Ex.0/1	Ey.0/0	Ey.0/1	Ez.0/0	Ez.0/1
E0.1/0	E0.1/1	E1.1/0	E1.1/1	E2.1/0	E2.1/1	E3.1/0	E3.1/1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.1/0	Ew.1/1	Ex.1/0	Ex.1/1	Ey.1/0	Ey.1/1	Ez.1/0	Ez.1/1
E0.2/0	E0.2/1	E1.2/0	E1.2/1	E2.2/0	E2.2/1	E3.2/0	E3.2/1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.2/0	Ew.2/1	Ex.2/0	Ex.2/1	Ey.2/0	Ey.2/1	Ez.2/0	Ez.2/1
E0.3/0	E0.3/1	E1.3/0	E1.3/1	E2.3/0	E2.3/1	E3.3/0	E3.3/1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ew.3/0	Ew.3/1	Ex.3/0	Ex.3/1	Ey.3/0	Ey.3/1	Ez.3/0	Ez.3/1

Where:

En.0/0, En.0/1	= received data (first user-cycle)
En.1/0, En.1/1	= received data (second user-cycle)
En.2/0, En.2/1	= received data (third user-cycle)
En.3/0, En.3/1	= received data (fourth user-cycle)

9

Status and Error Commands

There are a number of HP-IB commands that are used to check the status of the system hardware and to detect when any errors have occurred. This chapter describes the use of these commands.

The chapter is divided into the following parts:

- Status Byte - contains a summary of the current system status.
- Standard Event Status Register - contains a summary of any command-related errors and status information.
- Hardware Status Register - contains a summary of the current state of the hardware.
- Test Function Status Register - contains a summary of test function-related status information.
- Error Handling - contains a description of the commands that are used to handle any errors that may occur.

Other registers are available that control the output states of the status registers. These are described in the relevant sections below.

Note	Status reports generated by the hardware conform to the definitions in IEEE Standard 488.2 Section 11, Status Reporting Model. Refer to this standard for a more formal discussion of this topic.

Status Byte


```
        SRQ              Status
       enable             Byte
         ┌───┐            ┌───┐
  ◄──────┤ 7 ├◄───────────┤ 0 ├──◄──── Self Test
         │ 6 │            │RQS/MSS├◄──── Master Status Summery Bit
  ◄──────┤ 5 ├◄───────────┤ESB├──◄──── standard Event Status Summary Bit
 Master ◄┤ 4 ├◄───────────┤MAU├──◄──── Meassage AVailable Bit
 Status  │ 3 │            │ 0 │      not used
 Summary │ 2 │            │ 0 │      not used
  ◄──────┤ 1 ├◄───────────┤TFS├──◄──── Test Function Status Summary Bit
  ◄──────┤ 0 ├◄───────────┤HWS├──◄──── HardWare Status Summary Bit
         └───┘            └───┘

    *SRE < int_val>       <SPOLL>
        *SRE?              *STB?
```
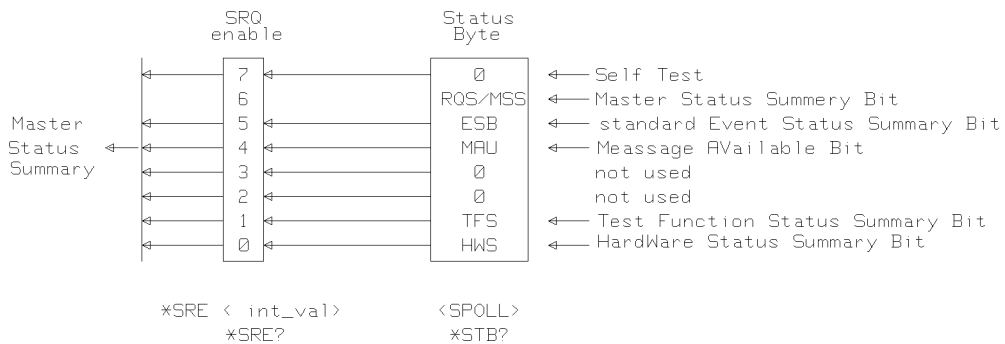
Figure 9-1. Status Model

The Status Byte is the summary of the tester status. It is eight-bits wide. Due to
the fact that the number of tester states cannot fit into one byte, each bit in the
Status Byte is the summary of a more detailed status in another status byte. The
Status Byte can be read with the *STB? command or an HP-IB serial poll bus
cycle. This can be performed by sending SPOLL (@dev) from BASIC.

The Status Byte bits and their meaning are listed below:

Table 9-1. Status Byte Organization

Bit	Meaning
7	SELF TEST report available.
6	Request Service / Master Status Summary (RQS/MSS) - indicates that the tester has a pending service request or is the summary of the Status Byte itself.
5	Standard Event Summary Bit (ESB) - the summary of the Standard Event Status Register.
4	Message Available (MAV) - indicates that the tester has data to send.
3	Unused, always set to 0.
2	Unused, always set to 0.
1	Test function Status Summary (TFS) - the summary of the Test Function Status Register.
0	Hardware Status Summary (HWS) - the summary of the Hardware Status Register.

The Master Status Summary Bit (MSS) is controlled by the Service Request Enable Register (SRE). Each bit in the Status Byte can be enabled individually by setting the corresponding bit in the SRE. The enabled bits together will form the MSS so that MSS is true if one of the enabled bits of the Status Byte is true. The MSS, which is also part of the Status Byte, cannot be set. The SRE can be set and read with the *SRE and *SRE? commands respectively. The content of the SRE also defines the generation of interrupts (Service Request (SRQ)). If MSS becomes true, a Service Request will be generated. The Service Request will go to false again after it has been acknowledged by the controller via a SPOLL or the *STB? command. The MSS however will stay true as long as long as the cause has not disappeared.

Event Status Register

The Standard Event Status Register (ESR) reflects a status which can be found on many HP-IB devices and is therefore standardized. Not all of the standardized meanings of the bit are applicable to firmware; those that are not used will always be set to zero.

```
              Standard Event      Standard Event
              Status Enable           Status

                 ┌───┐              ┌───┐
              ◄──│ 7 │◄─         ◄──│ 0 │       not used
              ◄──│ 6 │◄─         ◄──│ 0 │       not used
    Event     ◄──│ 5 │◄─         ◄──│CME│  ◄──  CoMmand Error
    Status  ◄ ◄──│ 4 │◄─         ◄──│EXE│  ◄──  EXecution Error
    Summary    ◄──│ 3 │◄─         ◄──│ATC│  ◄──  AuTo Correction
              ◄──│ 2 │◄─         ◄──│QYE│  ◄──  QuerY Error
              ◄──│ 1 │◄─         ◄──│ 0 │       not used
              ◄──│ 0 │◄─         ◄──│OPC│  ◄──  Operation Complete
                 └───┘              └───┘

              *ESE < int_val>
                   *ESE?                    *ESR?
```
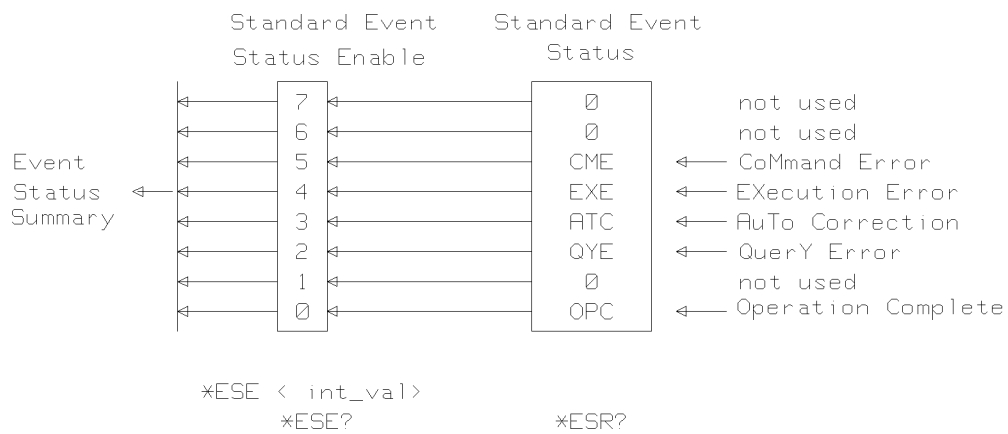
Figure 9-2. Standard Event Status Model

Table 9-2. Event Status Register Organization

Bit	Meaning
7	Not used, always set to 0.
6	Not used, always set to 0.
5	Command Error (CME) - indicates that a command error has occurred.
4	Execution Error (EXE) - indicates that an execution error has occurred.
3	Autocorrection (ATC) - indicates that an autocorrection has occurred.
2	Query Error (QYE) - indicates a problem in reading a response from the tester. The will occur either: 1. When the controller attempts to read from the system, without having sent a query. 2. Data relating to pending queries have been lost (possibly due to memory restrictions).
0	Operation Complete (OPC) - this bit is generated in response to a *OPC command. It is used for synchronization purposes. See the Command Synchronization chapter for more details.

Event Summary Register

The Standard Event Summary Bit (ESB) of the Status Byte is controlled by the Standard Event Status Enable register (ESE). Each bit in the ESR can be enabled to set the ESB by setting the corresponding bit in the ESE. The ESB will become true if one of the enabled bits in the ESR is true. This allows the propagation of selected standard events to the Status Byte, and together with the SRE, the generation of a Service Request.

Related commands are:

*ESR?	to read the current state of the ESR
*ESE	to set the ESE
*ESE?	to read the current setting of the ESE

Tester Status Registers

```
        Event        Event     Transition   Status
        Enable      Register      Filter    Register
        Register

      ┌─────┐      ┌─────┐      ┌─────┐      ┌─────┐
  ◄───┤  n  ├◄─────┤  n  ├◄─────┤  *  ├◄─────┤  n  ├◄─────
  ◄───┤ n-1 ├◄─────┤ n-1 ├◄─────┤  *  ├◄─────┤ n-1 ├◄─────
  ◄───┤  :  ├◄─────┤  :  ├◄─────┤  *  ├◄─────┤  :  ├◄─────         Tester
 Event◄───┤  :  ├◄─────┤  :  ├◄─────┤  *  ├◄─────┤  :  ├◄─────      Status
Summary◄───┤  :  ├◄─────┤  :  ├◄─────┤  *  ├◄─────┤  :  ├◄─────
  ◄───┤  2  ├◄─────┤  2  ├◄─────┤  *  ├◄─────┤  2  ├◄─────
  ◄───┤  1  ├◄─────┤  1  ├◄─────┤  *  ├◄─────┤  1  ├◄─────
  ◄───┤  0  ├◄─────┤  0  ├◄─────┤  *  ├◄─────┤  0  ├◄─────
      └─────┘      └─────┘      └─────┘      └─────┘

                              xxFL <int-val>
         xxEN <int-val>       xxRS <int-val>
                                xxFL?
         xxEN?       xxET?      xxRS?         xxST?
```
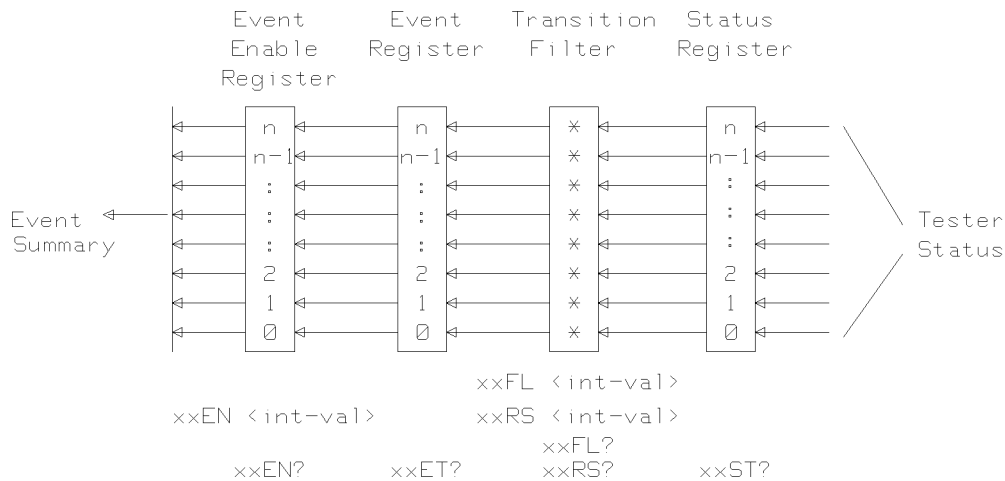
Figure 9-3. Tester Status Model

Tester status is reported by two registers. These are the Hardware and Test
Function Status Registers. They both operate in the same manner as described
below.

In the following descriptions the abbreviation xx relates to both the HW—and
TF—commands.

Status Register

Every bit in a Status Register reflects the state of part of the tester. Every bit
has a meaning of its own and its interpretation does not depend on the state of
other bits.

The Status Register can be read by the xxST?-query. The response will be the
decimal coded value of the register. The content of the Status Register can
not be changed directly. It always reflects the underlying tester state and will
change only if the tester state changes.

Transition Filter

This controls the changes of the Status Register that are considered to be an event. For every bit of the Status Register you can specify whether a transition from 0 to 1 or from 1 to 0, both transitions, or no transition are to be considered as an event. If the specified transition occurs for a bit that has been selected, the corresponding bit in the Event Register will be set. The contents of the Transition Filter can be specified by the xxFL and the xxRS commands.

Every bit set in the `int_val` of the xxFL command, will result the corresponding bit in the Event Register being set, if the corresponding bit in the Status Register changes from 1 to 0. Similarly, the `int_val` of the xxRS command, will result in setting the corresponding bit in the Event Register, if the corresponding bit in the Status Register changes from 0 to 1. These two commands in combination these command allow you to specify all the possible conditions. The current settings of the transition filter can be read with the xxRS? and xxFL? queries.

Event Register

This holds the events specified by the Transition Filter. The bits in the Event Register are sticky in the sense that a bit once set (to 1) will not be changed by any transitions in the Status Register. This may lead to the situation that the Event Register does not reflect the current status (which is the purpose of the status register), but it ensures that no event can be missed between successive reads of the event register. The Event Register can be read with the xxET? query command.

The Event Register is cleared when read with the xxET? query. All Event Registers are cleared with the *CLS-command, the DCL HP-IB message, and at power up.

Event Enable Register

This controls how the Summary bit for the Event Register is generated. Setting a bit in the Event Enable Register enables the corresponding bit in the Event Register. The Summary Bit will become true, if one of the enabled bits of the Event Register becomes true. The Summary Bit has a place in the Status Byte.

The Event Register can be written with the xxEN-command and read with the xxEN?-query.

The whole model is to ensure that every tester status change can be reflected in the Status Byte (STB) and generate an IRQ in a controller defined manner. This should become clear in the following example.

Situation:

Suppose the controller wants to receive an SRQ when bit 0 of the tester Status xx changes from 0 to 1 or bit 2 of the tester Status xx changes in either direction. Nothing else should cause a SRQ. The summary bit of Status xx is bit 0 in the STB.

Actions:

To setup the desired SRQ generation the controller has to send:

xxRS 5   *This enables event generation for transitions from 0 to 1 for bits 0 and 2 of the Status Register xx.*

xxFL 4   *This enables event genertation for a transition from 1 to 0 for bit 2 but not for bit 0 of the Status Register xx.*

xxEN 5   *Enables the summary bit generation for bits 0 and 2 of the Event Register xx.*

*SRE 1   *Enables the generation of SRQ for the summary bit xx, which is bit 0 of the STB.*

If the controller has received a SRQ, it should perform:

SPOLL   *- goes to all connected instrumments. The mainframe which has caused the SRQ will be the only one with RQS-bit set. The SPOLL clears the RQS-bit after reading the STB. This ensure that each SRQ will only be serviced once. If the SRQ was caused by the mainframe in our example, bit 0 will also be set indicating that an event has occurred in Status Register xx.*

xxET?   *This transfers the content of the Event Register xx to the controller. The controller can now decide whether bit 0 or bit 2 of Status Register xx has caused the event. This also clears the Event Register xx, allowing new events to generate a new SRQ.*

xxST?   *Reads the Status Register xx to the controller to allow status specific actions. It does not change the register.*

Hardware Status

The available hardware status commands are:

HWEN   defines the Status Enable Register of the Hardware Status
HWFL   defines the falling transition event generation of Hardware
            Status
HWRS   defines the rising transition event generation of Hardware
            Status
HWEN?   reads the Status Enable Register of the Hardware Status
HWET?   read the Event Register of the Hardware Status
HWFL?   read the falling transition event generation of Hardware Status
HWRS?   read the rising transition event generation of Hardware Status
HWST?   reads the Status Register of the Hardware Status

The bits in the Hardware Status Register have the following meaning: (This is
the meaning when the bit is set)

Table 9-3. Hardware Status

Bit	Meaning
1	Clock out of sync
2	Channel drivers active they drive the programmed levels.
3	clock active (running)
4	Sequencer busy (not BREAK) armed or active, test has not finished
5	Sequencer active (not ARMED and not BREAK))

Test Function Status

The available test function status commands are:

TFEN   defines the Status Enable Register of the Hardware Status
TFFL   defines the falling transition event generation of Test Function
            Status
TFRS   defines the rising transition event generation of Test Function
            Status
TFEN?   reads the Status Enable Register of the Hardware Status
TFET?   read the Event Register of the Test Function Status
TFFL?   read the falling transition event generation of Test Function
            Status

TFRS?   read the rising transition event generation of Test Function
               Status
TFST?   read the Status Register of the Test Function Status

The bits in the Test Function Status Register have following meaning: (This is
the meaning when the bit is set).

Table 9-4. Test Function Status

Bit	Meaning
1	Last test FAILED

Error Handling

Introduction

This section describes the error handling strategy of the system firmware. The
ERRS? command is used to check whether any errors have occurred.

Error Categories

When the firmware reads commands from the HP-IB, several errors may occur.
Because commands are interpreted in several stages, the firmware reacts
differently to certain errors. These errors are classified as follows:

CME   Command Error - is caused by a violation of the command
             syntax. The command will be completely ignored and all data
             will be ignored until the next semicolon (;), new line or EOI.

EXE   Execution Error - A command has been received with correct
             syntax, but it can not be executed. Reasons for this may be:

             1. A command parameter is outside the defined range
             2. Command parameters does not satisfy command specific
                restrictions, for example; high_level - low_level <
                min_swing.
             3. The command can not be applied with the hardware in its
                current state,for example; an attempt to define pins while
                the system is not idle.

The command descriptions at the back of this manual contain a list of the restrictions for each command.

Note	If a restriction applies only to some of the pins specified in a command, the command will be executed on all or some of the good pins.

ATC   Auto correction - The state of the tester was in conflict with a parameter. In order to execute this command, this state will also be changed. This is called a Side Effect. It is up to the user to retrieve the changed state from the firmware.

The errors classified here will be reported in the Standard Event Register (see below).

Note	The firmware will accept settings as long as the hardware can be programmed reasonably even if this is 'out of spec'! Programming 'out of spec' will not cause any errors, but hardware may fail to achieve the desired settings.

Error Queue

The occurence of any error will be logged in the error queue. The queue can contain 10 errors. If the error queue overflows, the last error in the queue will be replaced by the 'Error-Queue-Overflow' error.

The error queue is cleared by reading its contents, using the *CLS-command and the DCL HP-IB message.

The contents of the error queue can be read by the 'ERRS?'-query. The mainframe responds with all the errors queued so far and then clears the error queue.

The response of the ERRS? query is

```
ERRS? error-code, token, (Pins)
```

where

error-code   is an integer indicating the problem.
token   is the mnemonic of the command that caused the error, or zero
             (0) if the error is not related to a command.

`pins`   are the pins that are affected by the command causing the
                error. This list may be empty.

SELF TEST

The firmware performs checks to detect anomalies in the hardware or firmware
at power-on and download and during command execution. Any anomalies that
are found are recorded and can be queried using the SELF? query. The presence
of any self test reports is indicated by bit 7 of the Status Byte.

The response of the SELF? query is

    `SELF? major, minor`

where `major` and `minor` are integers indicating the problem. They are intended
for the HP service personnel.

Command Synchronization

Introduction

The problem of command synchronization arises because the firmware can
accept data faster than it can be processed. This is done to release the HP-IB
as quickly as possible and to allow parallel operation of the controller and
firmware. However, at some times, that the controller needs to know when the
firmware has reached a certain point in the processing of a data stream on the
HP-IB. There are two ways of doing this - by using handshake holdoff, or the
OPC and WAI commands.

At other times, the communications between the PWS and the hardware may be
disturbed, or may need to be stopped. If this occurs, the hardware has to be be
brought to a known state before any other commands can be processed. This is
also described in the following pages.

Handshake Holdoff

The firmware will change from parallel to serial processing if a data byte is sent with the EOI line true. The physical acceptor handshake AH1 or source handshake SH1 for the next byte of data is held off for the time required to process any pending data. This results in having this and all previous data processed, before the next data byte is accepted or transfered by the firmware.

Example

The controller sends a file of commands to the firmware and wants to check whether an error has occurred during the processing of these commands. It does this by reading the Status Register with the SPOLL command. It then needs to know when the command has been processed.

If the controller performs the SPOLL just after the last byte of the commands, there is a chance that not all of the commands have been processed and the value returned by SPOLL reflects a state in the middle of processing rather than at the end.

If the last byte of the commands is sent with EOI true, the SPOLL will not be executed completely before all the commands have been executed. So it ensures that the returned value reflects the state of the firmware after completely processing all the commands.

Caution	Using this way of synchronizing may inhibit all traffic on the HP-IB until the firmware has processed all pending commands. The firmware cannot guarantee that this is done in a certain time!

Operation Complete Message

These commands provide a synchronization method which does not rely on HP-IB holdoff.

The *OPC command is used to set an OPC event in the Standard Event Register. Refer to the chapter on status and error reporting for more information on the status registers.

The OPC event is generated after processing all previously received commands. By configuring status reporting appropriately, this can generate a SRQ which can be detected by SPOLL.

The OPC query will respond with a "1" after it and all previously sent commands are processed.

The *WAI command will wait until all pending overlapping commands are complete. Because the firmware has no overlapping commands, *WAI degenerates to a NOP.

There are three HP-IB commands that allow this type of synchronization:

*OPC   sets an OPC message in the Standard Event register.

*OPC?   responds with "1", when the command has been processed.

*WAI   waits for completion of overlapping operations.

Protocol Re-synchronization

Sometimes the communication between controller and firmware is disturbed or the controller needs to abort a currently active message. Therefore, it is neccessary to have a way of forcing the firmware to a known state. Because there are three levels of protocol (or handshake), there are also three levels of re-synchronization:

Note	Re-synchronization of one level is always done at the next lower lever. The three levels are handshake, message, and command level. So resynchronizing the command level must be performed at the message level.

IFC	Interface Clear - There is an extra line on the HP-IB called Interface Clear or IFC. If it is activated, the firmware will abort any pending physical handshake for the current HP-IB bus message (data or command byte). This ensures that the firmware will be able to physically accept the next bus message sent (for example, DCL).
DCL	Device Clear - This is an HP-IB bus command. It is sent with the ATN-line held true. It will reset all internal parsers, so that the next data byte sent will treated as a program_message_header. It will also clear all the status registers, except the output queue. This ensures that the firmware will be able to accept the next program_message (for example *RST).

*RST   RESET command - This is a global command, sent as data, which will place the tester in its Power Up state. (See the System Reset section, below.)

System Reset

The *RST command brings the tester hardware into a state caused by a power up, however *the original firmware remains loaded.* All registers in the hardware contain system defaults.

Caution	The RESET command should only be sent when the tester is in the disconnected state, since it may result in 'hot switching' of the relays (which may reduce the lifetime of the relays.)

If this command is used, the tester will be in an uncalibrated state. You must download base, dc, ac, and any user calibration files before performing any more tests. Failure to do this will result in unpredictable test results.

The files that need to be downloaded are:

    /hp82000/fw/data/bc_cal_dXX   *base calibration data*
    /hp82000/fw/data/dc_cal_dXX   *dc calibration data*
    /hp82000/dev_tech/YYY/ac_cal_dXX   *ac calibration data*

where XX is 50, 100, 200 or 400 depending on the system, and YYY could be either the supplied technologies ecl or cmos, or a user generated device technology.

Additionally, if you have any user calibration setting files, these must also be downloaded.

10

Calibration Commands

This section describes the commands used to calibrate the system, load calibration settings into the hardware and perform system diagnostics. The chapter is divided into the following parts:

- level calibration
- ac calibration
- 400 MHz calibration commands
- transfer of calibration data
- PMU calibration
- system diagnostics

Level Calibration

This section describes the firmware commands that support the level calibration of both driver and receiver levels. There are four commands used to perform a level calibration:

ADCM?	return ADC conversion result
MFLC?	mainframe level calibration
DRLC?	driver channel level calibration
RCLC?	receiver channel level calibration

Note	Performing a level or timing calibration may destroy the receiver vector memory contents. You should reload expected data before running a functional test after calibration.

Syntax

```
ADCM? multi-mainframe-id

ADCM multi-mainframe-id, adc-value

MFLC multi-mainframe-id, lol-offset, lol-gain, hil-offset,
hil-gain, drv-lol-adjust, drv-hil-adjust, thr-offset, thr-gain

MFLC? multi-mainframe-id

DRLC channel, lol-offset, lol-gain, hil-offset, hil-gain,
lol-adjust, hil-adjust

DRLC? query-channel

RCLC channel,lol-offset, lol-gain, hil-offset, hil-gain

RCLC? query-channel
```

where

`multi-main frame-id`	is 1, 2, 3, or ALL
`adc-value`	is an integer between -1 and 4095
`channel`	see DFPN syntax
`query-channel`	channel or ALL

All other parameters are floating point numbers.

Calibrating the ADC

The ADC, located on the clock board is used for all measurements during the level calibration process. It has therefore to be calibrated prior to all subsequent calibration steps.

With a multiple mainframe system, an ADC calibration must be performed for each installed mainframe.

The ADC calibration is part of the basic calibration procedure. To calibrate the ADC, an external reference supply and voltage measurement unit is required.

ADCM? measures the level seen at the external PMU rail connection and returns the conversion result in ADC scaling units. In practice, the calibration process requires at least two measurements. Note, that any number of measurements is

allowed. This may be used to compute the average values, which will in general lead to higher accuracy.

Assuming the measurements have been done applying the voltages U2 and U1 (causing the ADC values X2 and X1), the ADC gain to be downloaded (see MLCD) can be obtained from

$$Gain = \frac{(U2[uV] - U1[uV])}{(X2 - X1)}$$

Note	ADCM? returns -1 for `adc_value` if the measurement failed.

Calibrating the Mainframe

The mainframe level calibration command MFLC? returns the clock driver and cal-receiver calibration values. These values must be downloaded (see MLCD) prior to a subsequent timing calibration.

Calibrating Driver Outputs and Receiver Thresholds

Two commands are provided to calibrate the driver output levels (DRLC?) and the receiver thresholds (RCLC?).

The values returned must be downloaded using the DLCD and RLCD commands.

AC Calibration Commands

The AC calibration commands are used with a program running on the PWS to generate calibration values to be downloaded into the HW. The AC calibration commands do not alter any calibration values in the HW. The main functions of the calibration commands are to

- initiate the calibration process (ITMC?),
- to find the pin actually connected to the calibration probe, which is to be connected by the user to the pin to be calibrated (SCCH?),
- to perform driver and receiver AC calibration (DTMC?, RTMC?),
- to perform a user calibration (DTUC?, RTUC?),
- and to terminate the calibration process (TTMC).

The controller software calculates the actual driver and receiver offsets to be downloaded into the hardware by the appropriate command (DTCD, RTCD). The commands DTUD and RTUD perform the same function for the user calibration settings.

All calibration commands are channel oriented, so no configuration is required for AC calibration. However, user calibration uses the current pin configuration and timing settings.

For best timing accuracy a DC calibration must have been performed before starting an ac calibration.

Syntax

Initialize Calibration Process

    ITMC? [period | u]

The u option is used to start user calibration.

    ITMC error,#-of-probe-channels, period, round-trip
    { [,le-bpoints, te-bpoints] }

    CALP cal-period, cal-mclk

    CALP?

Generate Standard Calibration Values

    DTMC? channel,[probe-channel-nr]

    DTMC error, channel, probe-channel-nr, period, [le-rx]
    ,[le-rxslave]{ ,[le-bpoints] }, [te-rx],[te-hz],[te-rxslave]
    { ,[te-bpoints] }

    RTMC? channel, [probe-channel-nr]

    RTMC error, channel, probe-channel-nr, period, le-wc, le-wcslave,
    le-ec, le-ecslave, { [le-bpoints] }, te-wc, te-wcslave, te-ec,
    { [te-bpoints] }

Generate User Calibration Values

    DTUC? channel, [probe-channel-nr], driver-base-delay

```
DTUC error, channel, probe-channel-nr, period, base-delay, [le],
[le_slave], [te], [te-slave]

RTUC? channel, [probe-channel-nr], receiver-base-delay

RTUC error, channel, probe-channel-nr, period, base-delay, [le],
[le_slave], [te], [te-slave]
```

Search Channel

```
SCCH? [probe-channel-nr]

SCCH channel, probe-channel-nr, [pin-name]
```

Terminate Calibration Process

```
TTMC
```

Read Temperature

```
TEMP?

TEMP mainframe, temperature
```

where

`#-of-probe-channels`	1 or 16 for the number of calibration channels in the calprobe.
`probe-channel-nr`	an integer between 1 and 16 indicating the cal probe channel in use
`channel`	the channel to be calibrated.
`cal-period`	the calibration period in ns.
`error-code`	an integer in the range 0 .. 999999 indicating the calibration error code, 0 = no error.
`-offset`	found channel delays in ns for leading and trailing edges.
`-bp`	0 .. 4095 calibration probe fine delay dac values (up to 10) for leading and trailing edges.
`-basepoints`	0 .. 255 driver / receiver fine delay dac values (up to 10) for leading and trailing edges.
`le-`	leading edge
`te-`	trailing edge
`wc-`	window compare
`ec-`	edge compare

Calibrating the Cal Probe

ITMC? performs timing calibration of calibration probe and sets the system into a mode called calibration mode. In this mode, only calibration commands (except ITMC?) are allowed. The given calibration period is used for all subsequent calibration commands.

This command also returns the number of usable calibration probe channels. This is done when the probe is connected to any valid channel.

Calibration Period

The CALP command and CALP? query are used to set and query the `period` and `mclk` values corresponding to the downloaded AC calibration values.

In calibration mode, CALP? returns the actual calibration period and mclk value being used.

Determining which Channel is being Calibrated

SCCH? returns the channel, which is currently connected to the specified calibration probe channel (default is probe channel 1)

Performing a Driver Timing Calibration

DTMC? performs timing driver calibration of the specified pin using the given calibration probe channel (default is channel 1).

Performing a Receiver Timing Calibration

RTMC? performs timing receiver calibration of the specified pin using the given calibration probe channel (default is channel 1).

User Calibration

The commands for this function are used by the User Calibration part of the system Calibration software.

User calibration consists of calculating additional offsets to the standard calibration values for a fixed set of pin configuration, calibration, timing, and level setups.

These files are first downloaded into the hardware and the User Calibration routine then produces a set of additional offsets that are later stored in a user_cal file. The calibration probe is used to measure the actual timing at the

DUT pins. Any differences that are found between the actual timing and the expected timing are offset by the user calibration offsets.

For more details on user calibration, refer to Maintaining the HP 82000 and Servicing the HP 82000.

Stopping the Calibration Routine

TTMC terminates the calibration process

Note	TACC restores the user settings, destroys received and expected data, and sets the system into the disconnected state.
	Calibration does not change any loaded cal values.

Calibration Data Transfer

The calibration values obtained by running the system calibration, are used throughout all timing and level setup commands to ensure that the tester signals meet their specified accuracy. The commands described in this section are provided to read or modify the calibration values. Downloading a new set of calibration data causes the timing and levels to be updated, utilizing the new calibration values.

These are the commands that are used to transfer calibration data to the hardware:

MLCD	modifies mainframe level calibration data
MLCD?	obtains mainframe level calibration data
MTCD	modifies mainframe timing calibration data
MTCD?	obtains mainframe timing calibration data
DLCD	modifies driver level calibration data
DLCD?	obtains driver level calibration data
DTCD	modifies driver timing calibration data
DTCD?	obtains driver timing calibration data
RLCD	modifies receiver level calibration data
RLCD?	obtains receiver level calibration data
RTCD	modifies receiver timing calibration data
RTCD?	obtains receiver timing calibration data
DTUD	modifies driver calibration data at user setting

DTUD?	obtains driver calibration data at user setting
RTUD	modifies receiver calibration data at user setting
RTUD?	obtains receiver calibration data at user setting
CALP	sets the period and MCLK value corresponding to the downloaded AC calibration values
CALP?	in calibration mode, returns the current period and MCLK value
CALI	sets calibration data information
CALI?	reads calibration data information
CALS?	reads tester calibration state

Syntax

```
MLCD multi-mainframe-id, [adc-gain], [drv-lol-offset],
[drv-lol-gain], [drv-hil-offset], [drv-hil-gain], [drv-lol-adj],
[drv-hil-adj], [cmp-offset], [cmp-gain]
```

```
MLCD? multi-mainframe-id
```

```
MTCD multi-mainframe-id, [rec-basic-offset]{ , [le-basepoints] }
{ , [te-basepoints] }
```

```
MTCD? multi-mainframe-id
```

```
DLCD channel, [lol-offset], [lol-gain], [hil-offset], [hil-gain]
, [lol-adj], [hil-adj]
```

```
DLCD? query-channel
```

```
DTCD channel, [le-offset], [le-mux-offset]{ , [le-basepoints] }
, [te-offset], [te-hiz-offset], [te-mux-offset]
{ , [te-basepoints] }
```

```
DTCD? query-channel
```

```
RLCD channel, [lol-offset], [lol-gain], [hil-offset], [hil-gain]
```

```
RLCD? query-channel
```

RTCD channel, [le-wc-offset], [le-wc-mux-offset], [le-ec-offset]
, [le-ec-mux-offset]{, [le-basepoints]}, [te-wc-offset],
[te-wc-mux-offset], [te-ec-offset]{, [te-basepoints]}

RTCD? query-channel

Load User Calibration Values

DTUD? channel

DTUD channel, [le], [le-slave], [te], [te-slave]

RTUD? channel

RTUD channel, [le], [le-slave], [te], [te-slave]

CALP cal-period, cal-mclk

CALP?

CALS?

CALS state

CALI mainframe-id, cal-data-set, [cal-date], [cal-temperature]

CALI? mainframe-id, cald-data-set

CALI mainframe-id, cal-data-set, cal-date, cal-temperature

multi-mainframe-id	can be 1, 2, 3, or ALL
channel	(see DFPN syntax)
query-channel	channel | ALL

All other parameters are floating point numbers.

The parameters passed for the level calibration download commands (xLCD) must not exceed the following ranges:

Level Value Ranges (D100, D200 and D400)

		min	max	unit
Driver	Offset	- 300	+ 300	[mV]
	Gain	3.09236	3.28346	[mV/LSB]
	Adjust	0	7	[mV/V]
Receiver	Offset	- 500	+ 500	[mV]
	Gain	3.44040	3.87960	[mV/LSB]
Cal. Comp.	Offset	- 900	+ 900	[mV]
	Gain	5494.50	6715.50	[μV/LSB]
ADC	Gain	2500	4000	[μV/LSB]

Level Value Ranges (D50)

		min	max	unit
Driver	Offset	- 300	+ 300	[mV]
	Gain	2.0	2.5	[mV/LSB]
	Adjust	0	0	[mV/V]
Receiver	Offset	- 500	+ 500	[mV]
	Gain	4.3	5.4	[mV/LSB]
Cal. Comp.	Offset	- 900	+ 900	[mV]
	Gain	5494.50	6715.50	[μV/LSB]
ADC	Gain	2500	4000	[μV/LSB]

The parameters passed for the timing calibration download commands (xTCD) must not exceed the following ranges:

Timing Value Ranges (D100, D200 and D400)

		min	max	unit
Base- points	Channel Clock Driver	0 0	255 4095	
Offset	All Edges	- 10.0	+31.0	[ns]
User cal offsets		±32		[ns]
Round Trip Delay		0.0	+50.0	[ns]

Timing Value Ranges (D50)

			min	max	unit
Base- points	Channel Clock Driver		-32.768 0	+32.767 4095	
Offset	Leading Edge	Standard	0.0	+31.0	[ns]
		MUX mode	- 5.0	+31.0	[ns]
	Trailing Edge	Standard	- 1.0	+31.0	[ns]
		MUX mode	- 6.0	+31.0	[ns]
	Tristate Edge		0.0	+31.0	[ns]
Round Trip Delay			0.0	+50.0	[ns]

The mainframe level calibration values are accessible with the MLCD command.
One set of calibration data is held for each mainframe. The calibration
values include the clock driver data (`drv-lol-offset`, `drv-lol-gain`,
`drv-hil-offset`, `drv-hil-gain`, `drv-lol-adj`, `drv-hil-adj`), the calibration
comparator data (`cmp-offset`, `cmp-gain`), and the clock board ADC gain
(`adc-gain`).

The mainframe timing calibration values are accessible with the MTCD command. One set of calibration data is held for each mainframe. The calibration values include the clock driver/calibration comparator data (`le-offset`, `le-gain`, `te-offset`, `te-gain`) and the system roundtrip delay (`rec-basic-offset`).

The channel level calibration values are accessible with the DLCD (driver) or RLCD (receiver) command. One set of calibration data is held for each channel. The calibration values include the driver (`drv-lol-offset`, `drv-lol-gain`, `drv-hil-offset`, `drv-hil-gain`, `drv-lol-adj`, `drv-hil-adj`) and receiver data (`rec-lol-offset`, `rec-lol-gain`, `rec-hil-offset`, `rec-hil-gain`).

The channel timing calibration values are accessible with the DTCD (driver) or RTCD (receiver) command. One set of calibration data is held for each channel. The calibration values include the driver (`le-offset`, `te-offset`, `le-mux-offset`, `te-mux-offset`, `te-hiz-offset`) and receiver (`le-wc-offset`, `te-wc-offset`, `le-ec-offset`, `te-ec-offset`, `le-wc-mux-offset`, `te-wc-mux-offset`, `le-ec-mux-offset`) offsets, and the driver and receiver fine timing basepoints (`le-basepoints`, `te-basepoints`).

Additional Calibration Data Information

The CALI command is used to store additional information about the currently active set of calibration values in the HP 82000 firmware.

One data set is available on each connected mainframe. A single mainframe can be selected by specifying the `mainframe-id`, which may be in the range 1 through 5. Additionally, ALL may be used to alter or retrieve the calibration information of all mainframes at once.

The firmware distinguishes between four sets of calibration values, which may be accessed independently of each other specifying the proper `cal-data-set` value: The basic calibration data (BC), the level calibration data (DC), the standard timing calibration data (AC), and the timing calibration data for a given user setting (UC). Again, ALL may be used to alter or retrieve all data sets at once.

The `cal-date` is passed as a `string` (enclosed in double quotes ") and is given as `MM/DD/YY`.

The `cal-temperature` passed for each mainframe is given in Kelvin. Downloading a `cal-temperature` value more than 5 K away from the actual temperature results in a warning message sent by the affected mainframe(s). The power-on default for the cal-temperature is 0. Therefor no warning message will be sent if this value is downloaded.

After power-on (or after processing the *RST command), the cal-temperature is set to 0, and the cal-date is set to an empty string ("")

Reading the Calibration State

The CALS query returns 1 if the tester is in its calibration state. This state will be entered after a successful ITMC query and left after the corresponding TTMC command has been received by the tester. If the tester is not in its calibration state, 0 will be returned.

Power-On Defaults

D100, D200 and D400

```
MLCD mfid,3207.75,0,3188.48,0,3188.48,5.2,1.2,0,7024.74

MTCD mfid,13,327,654,981,1308,1635,1962,2289,2616,2943,
3270,327,654,981,1308,1635,1962,2289,2616,2943,3270

DLCD chan,0,3188.48,0,3188.48,5.2,1.2

DTCD chan,0,0,20.4,40.8,61.2,81.6,102,122.4,142.8,163.2,
183.6,204,0,0,0, 20.4,40.8,61.2,81.6,102,122.4,142.8,163.2,
183.6,204

RLCD chan,0,3662.11,0,3662.11

RTCD chan,0,0,0,0,20.4,40.8,61.2,81.6,102,122.4,142.8,163.2,
183.6,204,0,0,0,20.4, 40.8,61.2,81.6,102,122.4,142.8,163.2,
183.6,204,0,0,0,
```

50 MHz

```
MLCD mfid,4884,0,3188.48,0,3188.48
```

User Calibration

```
DTUD 0, 0, 0, 0
RTUD 0, 0, 0, 0
```

Notes

Commands specifying invalid channels or invalid mainframe IDs are silently ignored.

The sequencer must be in the OFF state.

Errors

Sequencer not in OFF state.

Value range errors

Warnings

Level/Timing setup out of specification (for the download commands).

D400 Calibration Commands

ILCD, ITCD	width generator calibration data
IXCM	width generator calibration setup
IXTC	width generator calibration
XTCD	400 MHz channel calibration data
XTMC	400 MHz channel calibration
XTUC	perform 400 MHz user calibration
XTUD	get or set 400 MHz user calibration data

HSWG Calibration:

Syntax

```
ILCD channel, [low-ft], [high-ft], [low-wd], [high-wd], [a, b, c]

ILCD? channel
```
Query Response:
```
ILCD channel, low-ft, high-ft, low-wd, high-wd, a, b, c

ITCD channel, edge, [offset]
```

```
ITCD? channel, edge
```

Query Response:

```
ITCD channel, edge, offset

IXCM pulse-width, low-level, high-level, connect-code,
channel-id

IXCM?
```

Query Response:

```
IXCM pulse-width, low-level, high-level, connect-code,
channel-id

IXTC? channel, op-code, [probe-channel]
```

Query Response:

```
IXTC error, probe-channel, period, channel, 0 , low-level,
high-level

IXTC error, probe-channel, period, channel, 1 , low-level,
high-level

IXTC error, probe-channel, period, channel, 2 , edge, basepoints

IXTC error, probe-channel, period, channel, 3 , edge, offset

IXTC error, probe-channel, period, channel, 4 .. 7 , edge, offset
```

Width Generator Calibration Data

The ILCD and ITCD commands are used to define a new set of calibration values
for high speed width generator (HSWG) channels. The currently active values
may be read by using the corresponding queries.

Both queries may be used to read either the calibration values of a dedicated
channel or of all installed 400 MHz channels (with channel set to ALL).

The edge parameter of the ITCD command is used to specify one of the four
available edges and must therefor be in the range 1 through 4 . For the ITCD
query ALL may be specified to read the calibration offsets of all edges.

The ILCD command defines the driver levels, setup to stimulate the high speed width generator. Low-ft and high-ft are the driver levels used in 'feed-through' mode, while low-wd and high-wd are used in 'width' mode. Additionally the three parameters a , b , and c are setup to minimize the delay variation, depending on the width generator's pulse width.

The ITCD command defines offset values for all edges. These offset values are used, if the high speed width generator is used in 'width' mode (400 MHz RZ). The offsets used in 'feed-through' mode are specified using the XTCD command.

Width Generator Calibration Setup

The IXCM command is used to define the width generator setup required to calibrate the channel selected by channel-id. The setup defined will be used by the next calibration command (see IXTC).

The IXCM query returns the required width generator settings together with the channel-id to perform the next step in the width generator calibration process. Nothing will be returned, if the tester is not in its calibration state (see CALS) or if no IXCM command has been sent since the last ITMC query.

Pulse-width is the required pulse width, given in ns units. A value of -1 shows that the width generator will be used in its 'feed-through' mode.

Low-level and high-level are the required output levels of the high speed width generator. The levels are returned in mV and are specified into open.

The connect-code specifies the required relay setting of the E1215 width generator. The meaning of the values 0 through 3 is the same as documented for the IXMD query response. In addition to these states 4 or 5 may be specified to connect either the width generators normal or complement path to the DUT board.

Executing the Width Generator Calibration

The IXTC query performs a timing calibration step, specified in op-code for the selected channel . The response to the IXTC query depends on the measurement selected by op-code.

Calibration measurements are done using the width generator settings specified by the IXCM command, sent most recently. If the passed channel number doesn't match the channel-id specified (see IXCM), an error message will be generated and no measurements are performed.

For multiplex calibration probes, the specified probe-channel will be used. If no probe-channel is specified, channel 1 is used, which is also appropriate for a standard calibration probe.

An op-code of 0 or 1 is used to obtain the driver levels required to stimulate the high speed width generator in either 'feed-through' or 'width' mode. The query therefor returns the required low-level and high-level values to be setup. These values must be downloaded using the ILCD command.

An op-code of 2 selects the fine timing linearity measurement. The measurement depends on the level calibration values, obtained in step 0 and 1 , which must be downloaded prior to this calibration step. The values returned are the delay DAC values used to obtain delay steps of 750 ps. These values must be downloaded using the XTCD command and are required for the following calibration steps.

An op-code of 3 or 4 selects an offset measurement in either feeed-through or 'width' mode. In 'width' mode, the pulse width used must be setup by the IXCM command. The offset values determined in 'feed-through' mode are downloaded using the XTCD command, while the offset adjust values for 'width' mode are downloaded using the ITCD command. The op-codes 5 through 7 have the same functionality as op-code 4 . These codes are required by the calibration program on the PWS to perform additional offset measurements in 'width' mode, used to calculate the coefficients for the delay-width dependency compensation (see ILCD).

400 MHz Channel AC Calibration

Syntax

```
XTCD channel, edge { , [basepoint] }, [driver-offset],
[single-threshold-offset], [dual-threshold-offset]

XTCD? channel, edge
```

Query Response:

```
XTCD channel, edge { , basepoint }, driver-offset,
single-threshold-offset, dual-threshold-offset

XTMC? channel, [probe-channel]
```

Query Response:

```
XTMC error-code, probe-channel, period, channel, edge
{ , basepoint }, driver-offset, single-threshold-offset,
dual-threshold-offset
```

400 MHz Channel Calibration Data

The XTCD command defines a new set of basepoints and driver offsets for standard 400 MHz and high speed width generator channels. Additionally the receiver offsets for standard 400 MHz channels are defined with this command. The currently active values may be read by using the corresponding query.

The query may be used to read either the calibration values of a dedicated channel or of all installed 400 MHz channels (with channel set to ALL).

The edge parameter of the XTCD command is used to specify one of the four available edges and must therefor be in the range 1 through 4 . For the XTCD query ALL may be specified to read the calibration values of all edges.

As for the 200 MHz channels, ten basepoints are available to describe the edge generator linearity. Depending on the period, some of the basepoints may be unused. The basepoints are given as offset values (in ns) for fixed fine delay DAC programming values (0, 16, 34, 54, 76, 100, 126, 154, 184, 216, and 250).

The driver-offset parameter specifies the edge offset (in ns) used for 400 MHz I pins or the offset in 'feed-through' mode for IX pins.

The single-threshold-offset (in ns) is used for FQ pins configured as O pins. The dual-threshold-offset (in ns) for edge 1 and 3 is used for FQ2 output pins and for the master channel of an FQM output pin. For FQM pins, the corresponding co-channel edge offsets are specified as the dual-threshold-offset parameters of edge 2 and 4 .

400 MHz Channel Calibration

The XTMC command returns the calibration values for a 400 MHz channel using the calibration probe-channel specified. If no probe-channel is given, a single probe calibration probe will be assumed.

If the calibration measurement fails for any reason, and error-code will be returned, describing the failure. Note, that in this case all other values returned are invalid.

The period returned is the calibration period, selected by the ITMC command.

400 MHz User Calibration

Syntax

```
XTUC? channel, [probe-channel-nr], driver-base-delay,
receiver-base-delay
```

Query Response:

```
XTUC error, channel, probe-channel-nr, driver-base-delay,
receiver-base-delay, period, drive-edge1, drive-edge2,
drive-edge3, drive-edge4, receive-edge1, receive-edge2,
receive-edge3, receive-edge4
```

```
XTUD channel, [drive-edge1], [drive-edge2], [drive-edge3]
, [drive-edge4], [receive-edge1], [receive-edge2],
[receive-edge3], [receive-edge4]
```

```
XTUD? channel
```

Query Response:

```
XTUD channel, drive-edge1, drive-edge2, drive-edge3, drive-edge4,
receive-edge1, receive-edge2, receive-edge3, receive-edge4
```

Performing 400 MHz User Calibration

The user calibration offsets are obtained for all edges at once. Not all edges are required for a particular pin configuration. Therefor the unused edge offsets are returned as 0 . The edge offsets are given in ns .

The probe-channel-nr selects a MUX-probe channel to be used for calibration. Omitting this parameter selects channel 1 , which is also used for a standard (single channel) calibration probe.

The driver-base-delay and receiver-base-delay are calculated by the standard calibration and are found in the calibration raw-data file as DRMA and RCMI values.

The user calibration offsets are downloaded for all edges at once. Because not all edges are used for a particular pin configuration, the unused edge offsets should be set to 0 . The edge offsets are given in ns .

PMU Calibration

There are six commands used for calibrating any installed PMUs. These are:

PMBC?   perform pmu basic calibration results
PMAC?   perform pmu auto calibration results
PBCD   downloads a new set of calibration values for the PMU board
               measurement unit.
PBCD?   reads the current setting of the calibration values
PACD   downloads a new set of calibration values for the PMU board
               voltage and current force units.
PACD?   reads the current setting of the voltage and current force units

```
    PMBC? pmu-id
```

returns

```
    PMBC pmu-id, adcval, offset-2V, offset-10V, offset-20V,
    offset-200nA, offset-2uA, offset-20uA, offset-200uA, offset-2mA,
    offset-20mA, offset-200mA, offset-500mA
```

```
    PMAC? pmu-id
```

returns

```
    PMAC pmu-id, gain-u, offset-2V, offset-20V, offset-20V,
    gain-i-pos, offset-ip-200nA, offset-ip-2uA, offset-ip-20uA,
    offset-ip-200uA, offset-ip-2mA, offset-ip-20mA, offset-ip-200mA,
    offset-ip-500mA, gain-i-neg, offset-in-200nA, offset-in-2uA,
    offset-in-20uA, offset-in-200uA, offset-in-2mA, offset-in-20mA,
    offset-in-200mA, offset-in-500mA, leakage, leakage-per-V
```

```
    PBCD? pmu-id
```

```
    PBCD pmu-id, [gain], [offset-2V], [offset-20V], [offset-20V],
    [offset-200nA], [offset-2uA], [offset-20uA], [offset-200uA],
    [offset-2mA], [offset-20mA], [offset-200mA], [offset-500mA]
```

```
    PACD? pmu-id
```

```
PACD pmu-id, [gain-u], [offset-2V], [offset-20V], [offset-20V]
, [gain-i-pos], [offset-ip-200nA], [offset-ip-2uA],
[offset-ip-20uA], [offset-ip-200uA], [offset-ip-2mA],
[offset-ip-20mA], [offset-ip-200mA], [offset-ip-500mA]
, [gain-i-neg], [offset-in-200nA], [offset-in-2uA],
[offset-in-20uA], [offset-in-200uA], [offset-in-2mA],
[offset-in-20mA], [offset-in-200mA], [offset-in-500mA],
[leakage], [leakage-per-V]$
```

where

pmu-id	is one of P11 | P12 | P21 | P22 | P31 | P32 (see test functions section)
adcval	an int-val (-1 .. 4095)
all others	fix-point-val

Adjustment of the system PMU boards requires two steps.

At first a "Basic Calibration" has to be done to adjust the on-board ADC, which will be used for all further measurements. The Basic Calibration is supported by the PMBC firmware command.

After downloading the ADC calibration values, PMAC may be used to return the calibration values for the voltage and current force DACs on the PMU board.

Note, that both calibration commands require at least one installed channel board.

Up- and downloading the PMU calibration values is supported by the PBCD and PACD commands.

The PMBC? returns the ADC offset values together with the binary conversion result achieved by measuring an external voltage source. While the offset values can be downloaded without any modification, the gain has to be calculated (using the returned ADC readouts). To calculate the ADC gain two measurements must be done, applying two different voltages of approximately +-18V. The gain to download will be

$$gain = \frac{U_2(+18V) - U1(-18V)}{adcval[1] - adcval[2]}$$

Adcval is the ADC conversion result after measuring the external voltage (0 .. 4095). The measurement will be done in the ± 20 V range.

Note	A value of -1 will be returned if the external voltage cannot be measured.

The offset values are returned as ADC binary readouts and are obtained by performing ten measurement cycles.

The PMAC? command is used to adjust an installed PMU and to transmit the calibration values to the host.

The gain calibration values (gain-u, gain-i-pos, gain-i-neg) are returned as fix point values using an implicit unit of 1 μV (or 1nA). Gain calibration is done using the ±20V (or ±20mA) range.

The units for the offset calibration values are range dependent as shown below:

Table 10-1.

Range	Unit
2 V .. 20 V	1 μV
200 nA .. 200 μA	1 fA
2 mA .. 200 mA	1 nA
500 mA	1 μA

Leakage and Leakage-per-V are the measured leakage current at (approx.) 0 V (passed in nA) and the leakage current voltage dependency (passed in nA / V).

The PBCD command is used to download a new set of calibration values for the PMU board measurement unit.

Gain is the ADC gain for the 20 V measurement range, passed as a fix point value using an implicit unit of 1 μV. Gain may be in the range 7 mV .. 13 mV.

The offset values are passed as binary ADC values. The valid range is 1500 .. 2500 for all voltage and current ranges.

The PACD command is used to download a new set of calibration values for the PMU board voltage and current force units.

The gain calibration values (gain-u, gain-i-pos, gain-i-neg) are passed as fix point values using an implicit unit of 1 μV (or 1nA). The valid ranges for the gain values are 7 mV .. 13 mV and 3.5 μA .. 6.5 μA.

The units of the offset calibration values are range dependent as shown below :

<div align="center">

Table 10-2.

</div>

Range	Unit	Range Limits
2 V .. 20 V	1 uV	± 2 V
200 nA .. 200 μA	1 fA	± 2 μA
2 mA .. 200 mA	1 nA	± 2 mA
500 mA	1 μA	± 2 A

`Leakage` and `Leakage-per-V` are the expected leakage current at (approx.) 0 V (passed in nA) and the leakage current voltage dependency The leakage current may be in the range ± 2 mA while the voltage dependency must not exceed ±100 μA / V.

Errors

Tester not in DISCONNECT state

Selected PMU not installed

Value range error

Diagnostic Commands

Diagnostics are performed with a standard query command. The DIAG? command is used by the Diagnostics User Interface.

Performing System Diagnostics

There is one command DIAG used for performing diagnostics. It is used from the Diagnostics User Interface. Refer to Servicing the HP 82000 manual for more information on system diagnostics.

Syntax

`DIAG? code`

`code`   tests to be performed.

When this command is sent, the system responds with:

`DIAG code, arbitrary-data`

where `arbitrary-data` indicates the results of the test specified in `code`.

11

Pin and System Attribute Commands

These commands are used to set or read pin and system attributes. A pin is a system channel which is referenced by a pin name (see DFPN command in the Configuration Commands chapter). The attributes are:

- Pin attributes
  - □ the series resistor wired on the DUT board
  - □ the output resistor of the DUT
  - □ an additional timing offset for driver/receiver edges.
- System attributes
  - □ a round trip adjust value

These attributes are used to correct the comparator thresholds and timing setup automatically.

The PATR command is used to set individual pin attributes. The query command PATR? can be used to read the current settings. The SATR and SATR? commands are used to set round trip adjust values for all pins.

Pin Attribute Commands

Syntax

```
PATR [Rs], [Rout-source], [Rout-sink], [t-drive], [t-rec],
pinlist

PATR? pinlist
```

where

Rs   is a real in the range 0 .. 6553.5 with the implied units Ohm
Rout-source   is a real in the range 0 .. 6553.5 with the implied units Ohm

```
Rout-sink    is a real in the range 0 .. 6553.5 with the implied units Ohm
t-drive      is a real in the range -32.767 .. +32.767 with the implied units
             nanoseconds
t-rec        is a real in the range -32.767 .. +32.767 with the implied units
             nanoseconds
```

Description

R_s is the series resistor wired on the DUT board in order to increase the signal fidelity. R_{out_source} and R_{out_sink} are the DUT output source and sink resistors respectively.

Defining these resistor values by executing the command, results in an automatic comparator threshold adjustment according to the following formulae:

Threshold Correction without active termination

$$U_{th} = U_x \times \frac{R_c}{R_c + R_{out-source} + R_s} \tag{1}$$

Threshold Correction for active termination (TERM, IOH, IOL) pins

$$U_{th} = \frac{(U_x \times R_t) + U_t \times (R_{out} + R_s)}{R_t + R_{out} + R_s} \tag{2}$$

where

U_{th} is the actual level programmed in the hardware
R_c is the comparator input impedance (10k $\pm$ 1% for a 50 Ohm
              system and 32k $\pm$ 20% for a 100 Ohm system)
R_t is the termination resistor (50 Ohm $\pm$ 3 Ohm for a 50 Ohm
              system and 100 Ohm for a 100 Ohm system)
U_x is the level programmed as a result of downloading a Level
              Setup or by using the Level Setup commands.
U_t is the termination level (low level for TERM and IOL and high
              level for IOH pins)
R_{out} DUT output resistor. Either R_{out_source} or R_{out_sink}, depending on
              U_x and U_t.

    If $U_x > U_t$ then
        $R_{out} = R_{out\_source}$
    else
        $R_{out} = R_{out\_sink}$

Using this adjustment formula ensures that comparator thresholds are programmed so that the user programmed thresholds are seen by the DUT output in case the output is not loaded.

This adjustment always takes place each time the comparator thresholds or the resistor values are changed by the appropriate commands.

T_{drive} and t_{rec} are timing offsets that are used to adjust the timing settings and are always added when programming timing edges. T_{drive} will be added to all the driver edges of the specified pin(s), while t_{rec} is added to all the receiver edges of the specified pin(s).

The timing offsets can be used to adjust the calibration values, for example, in the case of a longer DUT interface cable being used for connecting a prober, and the calibration was performed at the DUT Board with the Calibration Matrix.

The ranges of t_{drive} and t_{rec} specified above are theoretical, you should ensure that the sum of $t_{programmed}$, $t_{calibration}$, and t_{drive}/t_{rec} does not exceed the allowed range.

Note	Changing the pin attributes results in a temporary change to the sequencer OFF state, if it is not already in this state.

Power-On Defaults

At power-on, the pin attribute settings are:

`PATR 0,0,0,0,0,all-installed_pins`

Errors

An error will be generated for:

■ any value range errors.
■ attempting to program a non-applicable pin (configured as DC or OFF)

Warnings

Warnings will be generated by this command if:

- Level programming exceeds hardware specifications
- Timing programming exceeds hardware specifications
- Comparator threshold(s) auto-corrected

Setting System Attributes

Syntax

```
SATR [round-trip-adjust]

SATR?
```

where

round-trip-adjust   is a fixed point value with the implied units
                           nanoseconds.

Description

The `round-trip-adjust` defined in the SATR command is used to alter the
common receiver delay calculated by the calibration routines. The value will be
used for all the receivers in all mainframes.

The total value of `round-trip-adjust` and the round trip time calculated by
calibration (see MTCD command) must not exceed 50 nanoseconds, otherwise an
error will be generated and the setting will be ignored.

The SATR? query will be answered by the master mainframe only.

Note   Changing the system attributes results in a temporary change to
                  the sequencer OFF state, if it is not already in this state.

Power-On Defaults

At power-on, the system attribute settings are:

`SATR 0`

Errors

An error will be generated for:

- any value range errors (combined round trip delay value greater than 50 nanoseconds).

12

HP-IB Command Syntax

Introduction

This chapter contains general information on the HP-IB commands followed by an alphabetical list of all the commands used in the hardware.

Each command description consists of a short description, the required syntax, possible error and warning messages and a short example. For more information on specific topics refer to the relevant section in the following chapters.

General Information

The system HP-IB command language conforms to IEEE Standard 488.2 Draft 18 with the exception of the optional parameters.

All commands sent via HP-IB consist of ASCII strings except the vectors, which are transferred in a binary format. The DMA used in the system allows a vector transfer rate of approximately 250 kbyte/sec.

Hardware channels can only be accessed by their user defined name, as specified in the pin configuration setup menu.

Exceptions to this rule are the calibration and diagnostics routines, which act directly on machine channels. During calibration, the channel is selected manually by probing the DUT board pin (returning the pin name if set up). Diagnostics may check channels which are not connected to the DUT board or used for he current application. The Input and Output part of the same channel must have the same name. The wildcard symbol (@) can be used to access all applicable channels.

All settings can be read by the controller via queries, the responses to queries have the same format as the corresponding setup commands. Exceptions are noted in the command description.

Example:

```
DRLV?  (RESET)          leads to the response
DRLV 0, 5, (RESET)
```

The above example returns the high and low level of the pin called RESET.

Hardware programming is contextually independent which means that, for example, MUX pin vectors will be downloaded to the hardware as if there were two independent pins.

Note   The only exceptions to this rule are that pin naming must
            precede all other pin commands and clock programming should
            precede edge programming.

Command headers consist of four letters with no upper/lower case significance. The corresponding query command uses the same name followed by a question mark (?).

For optional parameters, the current value will remain set in the hardware if it is not affected by subsequent use of the command. In the case of a parameter mismatch, optional parameters may be autocorrected and generate a warning. Mismatched parameters in a command generate an error.

System HP-IB Commands

Global definitions

Spaces are not explicitly shown in the syntax definitions. They are always allowed between "none terminal symbols", which means that you can insert any number of spaces (ASCII 255) between any "none terminal symbol".

Except for pin names, there is no significance between upper and lower case letters.

The general form of strings downloaded to the system is shown in the figure below.

```
{<d2_command>}

d2_command ::= [program_message]
                          <program_message_terminator>

                                    ⎡  EOI     ⎤
program_message_terminator ::=      ⎢ NL + EOI ⎥
                                    ⎣   NL     ⎦

program_message ::= <program_message_unit>

program_message_unit ::= <program_header>
                          <program_data>, <program_data>

program_header ::= <char><char><char><char>

                          ⎡   <int_val>       ⎤
                          ⎢ <fix_point_val>   ⎥
program_data ::=          ⎢  <enumerated>     ⎥
                          ⎣   <pin_list>      ⎦

int_val ::= {<digit>}

fix_point_val ::= [<sign>] [<digit>.<digit>]

           ⎡ + ⎤
sign ::=   ⎣ - ⎦

enumerated ::= <char><char><char><char>

pin_list ::= <pin_name>, [<pin_name>]

pin_name ::= <pin_char>{<pin_char>} <= 15

pin_char ::= ASCII 21H thru 7EH except " # ´ ( ) , ;
```

Figure 12-1. Downloaded Strings

Note

- CR is allowed and will be taken as a white space.
- Value parameters (eg. edge delays, level voltages, etc.) are represented in fixed point format and take advantage of implicit units (eg. implicit unit of edge delays is ns). Which implicit unit is taken for what value parameter is explained in the particular chapters.
- the at character (@) is the 'all pins' wildcard and will be expanded by the firmware to a list of all applicable pins.

The general form of the query responses is shown in the figure below.

```
{<d2_response>}
d2_response ::= [response_message]
                    <response_message_terminator>
response_message_terminator ::=  NL + EOI

response_message ::= <response_message_unit>

response_message_unit ::= <response_header>
                    <response_data>, <response_data>

response_header ::= <char><char><char><char>
                    ⎡    <int_val>        ⎤
                    ⎢ <fix_point_val>     ⎥
response_data ::= ⎢                       ⎥
                    ⎢   <enumerated>      ⎥
                    ⎣    <pin_list>       ⎦
int_val ::= {<digit>}
fix_point_val ::= [<sign>] [<digit>.<digit>]
sign ::= -
enumerated ::= <char><char><char><char>
pin_list ::= <pin_name>, [<pin_name>]
pin_name ::= <pin_char>{<pin_char>} <= 15
pin_char ::= ASCII 21H thru 7EH except  "  #  ´  (  )  ,  ;
```

Figure 12-2. Uploaded Strings

Note	CR NL+ EOI may be an alternate response message terminator, configured by a special HP-IB command.

ADCM?

Calibration ADC Calibration

ADCM? returns the ADC level measured at the PMU rail external connection in ADC scaling units.

where `mainframe_id` is either 1, 2, 3, 4, 5 or ALL.

This returns the following string:

`ADCM adc-value`

where `adc-value` is the ADC conversion result

Errors

An error will be generated for the following reasons:

■ error
■ error

Example

Sending `ADCM? 1` will return a string similar to

`ADCM 1,1519`

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

AQST?

Test Results Number of cycles

AQST? returns the number of cycles between the `event address` and any detected stop condition, and returns a -1 if no event was found before the vector sequence was completed.

Sending this command causes the system to return `cycles_after_event_addr` which is an integer between -1 and 2^{24} indicating the number of cycles of data acquired by the last test.

The command is valid for acquisition and compare modes.

Errors

An error will be generated for the following reasons:

■ Parameter range errors

Example

Sending aqst? without previously performing a test returns -1.

Sending it after performing a test will return the number of cycles acquired in the last test

`ftst?`	*Perform functional test*
`FTST F`	*returned by system if test fails*
`aqst?`	*Query number of cycles*
`AQST 1001`	*1001 cycles were acquired since the delay counter was started*

CALI, CALI?

Calibration Set and read calibration data information

The CALI command stores additional information about the currently active set of calibration values in the HP 82000 hardware.
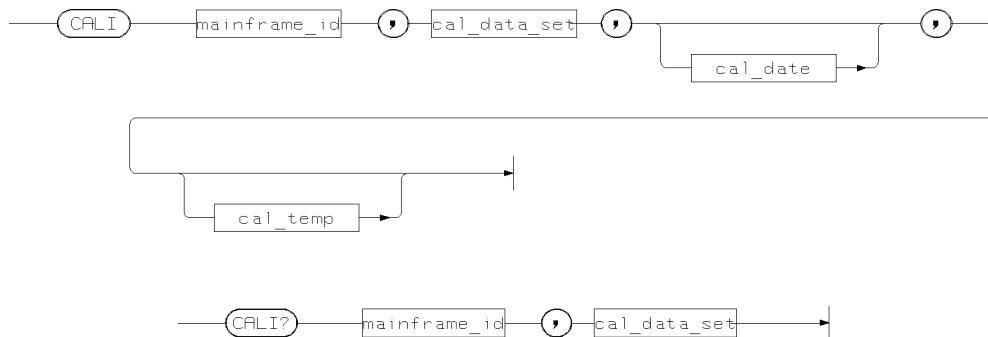
where:

`mainframe_id`	can be 1, 2, 3, 4, 5 or ALL
`cal_data_set`	can be:
	BC (base calibration)
	DC (level calibration)
	AC (standard calibration)
	UC (calibration at user settings)
`cal_date`	is the calibration date—a character sting enclosed in double-quotes (") in the form "MM/DD/YY" (i.e. Month/Day/Year).
`cal_temp`	is the temperature at calibration time, in units of Kelvin (K). (default=0)

Errors

■ `cal_temp` more than ±5K from actual temperature.

CALI, CALI?

Example

```
CALI 1,AC,"10/03/90",293
```

means that the standard calibration for mainframe 1 was performed on 3rd
October 1990, at an ambient temperature of 293k (20°C) ±5K.

CALP, CALP?

Calibration Set/read period and MCLK value

This command sets/returns the period and MLCK value corresponding to the downloaded AC calibration values.

In calibration mode, CALP? returns the current value of calibration period and MCLK.

Example

Refer to Chapter 10 for more information on system calibration.

CALS?

Calibration Read tester calibration state

$$\longrightarrow \left(\text{CALS?} \right) \longrightarrow |$$

Returns the string:

$$\longrightarrow \left(\text{CALS} \right) \longrightarrow \boxed{\text{state}} \longrightarrow |$$
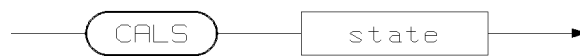
where:

`state` can be either:

■ 1 indicating that the system is calibrated; or
■ 0 indicating that the system is not calibrated.

CHER?

Test Results Get pass/fail information per pin

CHER? returns passed / failed information for every given pin.

CHER? ─── pin_list ───▶

Sending the command returns `passed/failed` which can be either P or F depending on whether an error was detected in the last test.

The command is only valid in RTC mode.

Note	Each time the CHER? query is executed, the PASS/FAIL flag is reset to PASS. So if you execute CHER? twice in succession, without performing a test between executions, the second execution of CHER? will always return `pass`.

Errors

An error will be generated for the following reasons:

■ Parameter range errors
■ Command used in acquisition mode
■ No data available - no test has been performed

Example

After performing a test in RTC mode, sending CHER? will return the pass./fail for each pin in pinlist.

```
ftst?           Perfrom functional test
FTST F          Test failed
cher? (@)       Q0 caused fail
CHER F,(Q0)
CHER P,(Q1)
CHER P,(Q2)
CHER P,(Q3)
```

CMNT, CMNT?

Vector Setup Add or read comments in vector table

```
──(CMNT)──[ start_line ]──(,)──[ stop_line ]──(,)──[ comment ]──►

      ──(CMNT)──[ start_line ]──(,)──[ stop_line ]──►
```
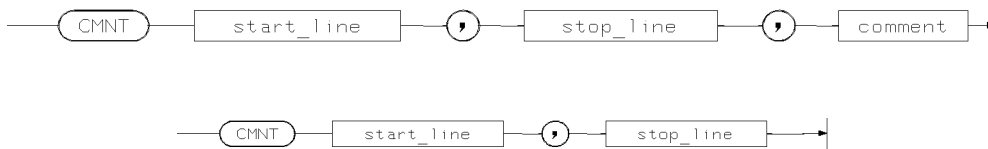
where:

`start_line`	is the first line to be commented;
`stop_line`	is the last line to be commented;
`comment`	is the comment text.

The comment text must be enclosed in double quotes ("). A vertical bar (|) in the comment text acts as a separator between lines.

Alternatively, you can designate lines without comments by entering an empty comment string.

Example

`CMNT 0,1,"Comment for line 0|Comment for line 1"`

`CMNT 10,100,""` *(lines 10 to 100 have no comments)*

CONF, CONF?

Configuration Pin operating modes

The CONF command sets up the operating modes for a pin and checks for sufficient calibration data for those modes. The related CONF? query command returns the mode settings of the named pins.

```
─(CONF)─┬──────────┬─(,)─┬─────────────────┬─(,)─┬─────────────────┬─
        └─[io_mode]─┘     └─[operation_mode]─┘     └─[scan_path_mode]─┘

        ┌─(,)─[pin_list]─┤

─(CONF?)─[pin_list]─┤
```
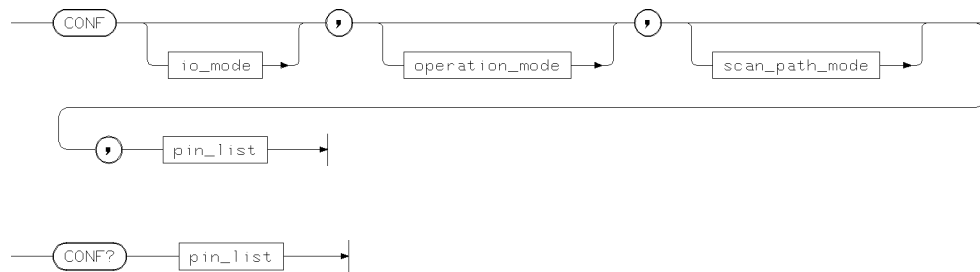
The parameter `io_mode` determines the io capabilities of the pin. `io_mode` can consist of one of:

I	input
IX	high speed (400 MHz) input (HSWG channel)
O	output
IO	bidirectional
TERM	DUT output pin, but in contrast to output (O) mode, it is terminated by an active load.
NC	The pin is considered to be disconnected from the DUT.
DC	The pin is considered available for DC (PMU) measurements only. No AC-resources are accessible.

The parameter `operation_mode` determines the internal operation mode of the channel logic and consists of one of:

STD	This is the standard mode of operation for D50, D100 and D200 boards.
FD	This mode doubles the STD mode frequency of D50 and D200 I/O boards by reducing the available formats.
MUX	This mode doubles the STD mode frequency of D50 and D200 I/O boards, by combining the resources of two consecutive channels.

CONF, CONF?

FQ2   This mode provides the maximum number of signal formats for
                       D400 I/O boards, at a frequency of 200 MHz.
FQ   This mode provides DNRZ signals at 400 MHz. Additional signal
                       formats are available if this channel is routed through a High
                       Speed Width Generator (HSWG).
FQM   This mode combines the receiver resources of two channels, to
                       provide dual threshold edge compare at 400 MHz. No driver
                       resources are available.

The parameter `scan_path_mode` determines the behavior of the pin during a
serial scan vector.

P   The channel keeps its data during a serial scan. Its vector
                       memory may be used for a channel in serial mode.
S   The channel serializes data from the vector memory of other
                       channels during a serial scan.

The query command CONF? responds with the IO, operation, and scan path
modes for the required pin(s).

Errors

An error will be generated if `pin_list` contains more than one pin in the CONF
command.

Example

Entering

```
DFPN 0101,"",(NOT_CS)
CONF I,STD,,(NOT_CS)
```

defines NOT_CS, which was previously defined by the DFPN command, being
configured as an input pin, operating in 100 MHz mode and parallel (default).

DCGB, DCGB?

Test Functions DC measurement globals

This command is used to set and read the global conditions for dc measurements using the PMU.
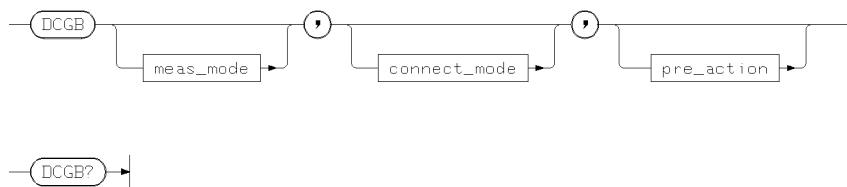
where

meas-mode   is the measurement mode to be used. Can be:
                P parallel - all pins are measured and the combined current
                and voltage is measured. Read the value using DCSR?
                S serial - each pin is measured separately taking the values in
                PATR into consideration for the calculation. Read the values
                using DCPR?
                D direct - each pin is measured separately. The value at the
                DUT pin can be read using DCPR?
connect-mode   is the connection mode used for the PMU. Can be:
                A additional - PMU is connected to the IO channel during the
                measurement. Any AC circuitry on the IO board can affect
                the measurement. Can be used to make measurements under
                loaded conditions.
                E exclusive - PMU is connected only to DUT pin. The IO
                Board is disconnected during the measurement.
pre-action   sets up the DUT before the measurement takes place.
                NONE - no setup is performed.
                FTST - a functional test is performed before the measurement
                takes place.

DCGB, DCGB?

Power On Defaults

At power-on the following setups are made:

`DCGB S,E,NONE`

Example

Sending `DCGB,P,E,NONE` will cause the next PMU measurement to be performed in parallel with the IO Board disconnected from the DUT pin.

DCL

HP-IB Device Clear

```
—(DCL)—|
```

```
DCL
```

Refer to BASIC/UX Interfacing Techniques manual for more information.

DCPR?

Test Functions Retrieve results for per pinlist (parallel) dc tests.

```
────(DCPR?)────[ set_id ]────▶
```

Returns the string:

```
(DCPR)─(,)─[ set_id ]─(,)─[ valid_code ]─(,)─[ measured_u ]─(,)─┐
   ┌──────────────────────────────────────────────────────────┘
   └─[ measured_i ]─(,)─[ pin_list ]─▶
```

where

valid-code	describes the condition of the PMU during the measurement. It is an integer in the range 0 .. 63. If `valid-code` is coded binary it has the format `OFGppppp` where
	■ `ppppp` is the `pmu-state` as described in the PMUM? query.
	■ `G` is the force-i overflow flag. It is set when the PMU could not be set to the necessary force current.
	■ `F` is the functional fail flag. It is set if preceding functional test result in a fail.
measured-u	is the measured voltage given in millivolts. It is only valid, if the `valid-code` indicates neither unstable nor voltage overflow.
measured-i	is the measured current given in microampers. It is only valid, if the `valid-code` indicates neither unstable nor current overflow.
pinlist	specifies the pins the former parameters are related to.

DCSR?

Test Functions Retrieve results for per pin (series) dc tests.
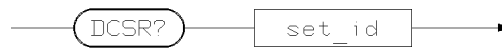
Returns the string:

where

valid-code   describes the condition of the PMU during the measurement. It
              is an integer in the range 0 .. 63. If `valid-code` is coded binary
              it has the format `OFGppppp` where
              ■ `ppppp` is the `pmu-state` as described in the PMUM? query.
              ■ `G` is the force-i overflow flag. It is set when the PMU could
                not be set to the necessary force current.
              ■ `F` is the functional fail flag. It is set if preceding functional
                test result in a fail.

measured-u   is the measured voltage given in millivolts. It is only valid, if
              the `valid-code` indicates neither unstable nor voltage overflow.

measured-i   is the measured current given in microampers. It is only valid,
              if the `valid-code` indicates neither unstable nor current
              overflow.

DFCM

High Throughput Tests Define current measurement parameters.

where:

`dc_set_id`	is the identification number (id_dc) for the dc parameters (in the range 1 to 64);
`force_voltage`	is the PMU voltage to be used during the measurement (in mV);
`max_current`	is the maximum current allowed during the measurement (in μA);
`pass_min`	is the minimum pass current value (in μA);
`pass_max`	is the maximum pass current value (in μA);
`connect_voltage`	specifies the voltage forced by the PMU while the DC relay is connected. If this parameter is omitted, then the force-voltage value is used.
`settling_time`	specifies how long to wait between the connection of the PMU, and performing the measurement (in milliseconds);
`mode`	can be:
	■ `GNG` (ganged)—measurements are taken while all the pins are connected to the PMU;
	■ `NGNG` (not ganged)—each pin is measured individually.
`pin_list`	are the pins included in the test.

The dc parameter sets can be read using the DFPM? query.

Example

To define the dc parameters for a PMU current measurement test, on pins Q1, Q2 and Q3 of your device, with

- the id number (*id_dc*) "4";
- a force of voltage -1.7 Volts;
- a maximum current of 0.1 milliamps ;
- a minimum pass value of 20 microamps;
- a maximum pass value of 80 microamps;
- the default connect voltage (in this case, -1.7 V)
- a settling time of 1 millisecond;
- with the PMU measurement carried out on all three pins together;

you enter the string:

```
DFCM 4,-1700,100,20,80,,1,GNG,(Q1,Q2,Q3)
```

You can check the dc measurement settings already stored in RAM, with the DFPM? query:

```
DFPM? id_dc
```

DFPM?

High Throughput Tests Read PMU measurement parameters, perviously defined by DFCM and DFVM.

```
─( DFPM? )─[ param_set_id ]─►
```

returns either:

```
─( DFCM )─[ dc_set_id ]─(,)─[ force_voltage ]─(,)─[ max_current ]─(,)─[ pass_min ]─(,)─
─[ pass_max ]─(,)──────────────────────────(,)─[ settling_time ]─(,)─[ mode ]─(,)─[ pin_list ]─►
                    └─[ connect_voltage ]─┘
```

or:

```
─( DFVM )─[ dc_set_id ]─(,)─[ force_current ]─(,)─[ clamp_voltage ]─(,)─[ pass_min ]─(,)─
─[ pass_max ]─(,)──────────────────────────(,)─[ settling_time ]─(,)─[ mode ]─(,)─[ pin_list ]─►
                    └─[ connect_voltage ]─┘
```
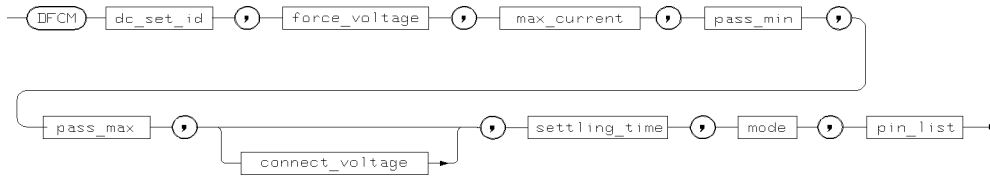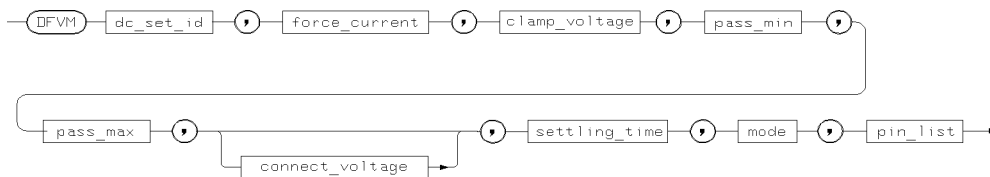
See DFVM and DFPM for the parameter descriptions.

DFPN, DFPN?

Configuration Setup Define pin name

The DFPN command sets up a new name for one unused channel. Later you can refer to this channel using the given name.

The corresponding DFPN? command responds with the channel number of the specified pin.
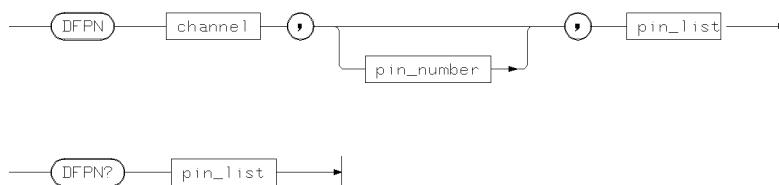
`channel` describes the channel to use. It is a five digit number that describes the channel to use for a pin. The five digits fall into three groups and the number has the format `mbbcc`

m	specifies the mainframe and can be 1, 2, or 3.
bb	specifies the slot number of the channel board in the mainframe. This can be a number between 01 and 16.
cc	specifies the number of the channel on the board. Boards can have either eight or sixteen channels and are numbered 01 to 16.

`pin_number` is for comment purpose only. It is required by the system software.

`pin_list` contains one previously undefined `pin_name` that is to be defined.

At power-up there are no pins defined.

The query command DFPN? returns the channel and pin number for the named channel.

DFPN, DFPN?

Errors

An error will result if `pin_list` contains more than one pin.

Example

The commands

```
DFPN 10405,"",(pin1)
```

results in channel 5 on the fourth IO Board in mainframe 1 being given the name `pin1`. You must define a pin with the DFPN command before you can access it from any other commands.

Sending the command

```
DFPN? (@)
```

will result in a list of the currently defined pins being output.

DFPS, DFPS?

DPS Define DPS pins

These commands are used to modify and read DPS settings.

where

channel   is a two-digit figure which indicates the DPS number and the
                 DPS channel used. For example, DPS1, channel 3 is indicated
                 by the number 13 in this field. on the type of DPS installed.
                 See PSCF? to read the number of DPS channels.

sign   sets the polarity of the DPS output. Can be either POS or NEG.

pin   defines the pin name for the DPS channel.

pinlist   pin names previously defined by a DFPS command.

Example

To define a DPS2, channel 2 as a positive supply to a pin called `dps1`:

```
DFPS 22,POS,dps1
```

DFVM

High Throughput Tests Define voltage measurement parameters.

where:

`dc_set_id`	is the identification number (id_dc) for the dc parameters (in the range 1 to 64);
`force _current`	is the PMU current to be used during the measurement (in μA);
`clamp_voltage`	is the maximum voltage allowed during the measurement (in mV);
`pass_min`	is the minimum pass voltage value (in mV)
`pass_max`	is the maximum pass voltage value (in mV)
`connect_voltage`	specifies the voltage forced by the PMU while the DC relay is connected. If this parameter is omitted, then the mean value between `pass_min` and `pass_max` is used.
`settling_time`	specifies how long to wait between the connection of the PMU, and performing the measurement (in milliseconds);
`mode`	can be:

- `TRM` (terminated)—the pin is connected to the PMU and the pin electronics;
- `NTRM` (not terminated)—the pin is connected to the PMU only;
- `SPOL` (specified polarity)—the measurement is taken with the polarity specified by the clamp voltage;
- `BPOL` (both polarities)—measurements are taken with both polarities (a pass result is given if the pin passes the test in *either* direction).

(SPOL and NTRM are synonymous.)

`pin_list`	are the pins included in the test.

The dc parameter sets can be read using the DFPM? query.

Example

To define the dc parameters for a PMU voltage measurement test, on pins Q1, Q2 and Q3 of your device, with

- the id number (*id_dc*) "3";
- a force current of 5 milliamps;
- a clamp voltage of 5.5 Volts;
- a minimum pass value of 4.5 Volts;
- a maximum pass value of 5.0 Volts;
- the default connect voltage (in this case, 4.75 V)
- a settling time of 1.5 milliseconds;
- with the PMU measurement carried out while the pin electronics are still connected;

you enter the following string:

```
DFVM 3,5000,5500,4500,5000,,1.5,TRM,(Q1,Q2,Q3)
```

You can check the dc measurement settings already stored in RAM, with the DFPM? query:

DFPM? *id_dc*

DIAG

Diagnostics Perform system diagnostics

This command is used by the Diagnostics User Interface.

DLCD, DLCD?

Calibration Driver level data

These commands are used to modify and read driver level calibration values.
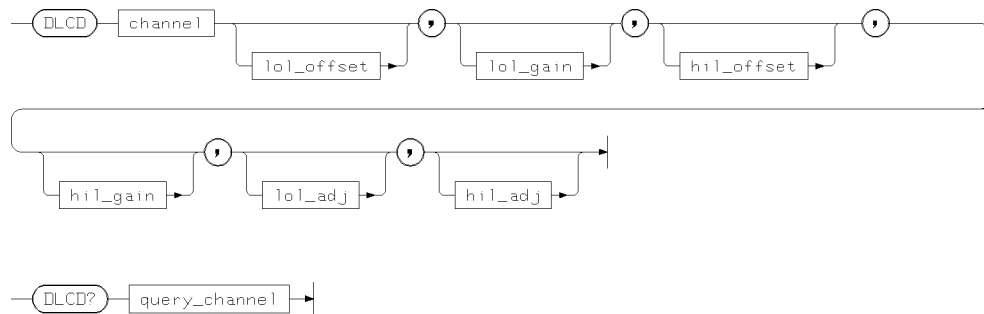
where `channel` is the sequential channel number defined with the DFPN command.

The values for the offsets and gains are shown below.

Level Parameter Value Ranges

		min	max	implied unit
Driver	Offset	-300	+300	mV
	Gain	3092.36	3283.46	μV/LSB

Errors

Errors are generated for the following reasons:

- Sequencer not idle
- Gain value out of range
- Trying to set or read driver calibration data for an O channel

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

DPAR, DPAR?

Test Functions Define dc measurement parameters
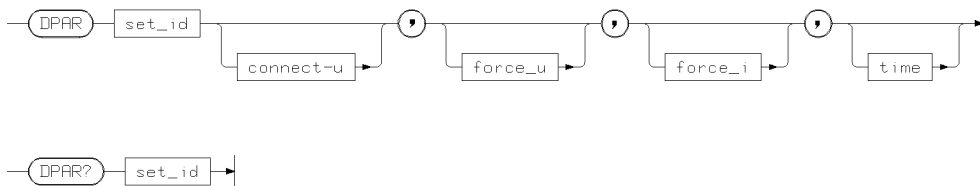
where

`set-id`	0 or 1 for the parameter set to be defined
`connect_u`	sets a charging voltage on the PMU pins to reduce current flow. Must be in the range −20000 .. +20000 millivolts.
`force_u`	force voltage for PMU measurement. Must be in the range −20000 .. +20000 millivolts.
`force_i`	force current for PMU measurement. Must be in the range 0 .. 500000 (μA).
`time`	additional settling time between connection of PMU and measurement taking place in milliseconds

DPAR? returns the settings for the specified parameter set.

Power On Defaults

At power-on the following settings are made:

```
DPAR 0,0,0,0,0
DPAR 1,0,0,0,0
```

Example

Sending `DPAR 0,1000,,500,5` will precharge to 1 Volt, and force a current of 500 μA after 5 milliseconds.

DRLC?

Calibration Performs a driver level calibration

```
─( DRLC? )─┤ query_channel ├─▶│
```

where `query-channel` is the driver channel to be calibrated.

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

DRLM,DRLM?

Level Setup Specify timing reference level

```
──(DRLM)──┬──(L50)───┤
          └──(FRZN)──►
```

```
──(DRLM?)──►│
```
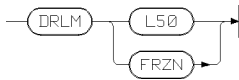
These commands specify and read the current settings for the timing reference levels. Using the setting L50 sets the level to the 50% point of the level setting. FRZN sets the reference level to the setting of nominal-swing in the DRLX command.

Warnings

This command may be overridden by any settings made in DRLX commands. Any value of nominal-swing except -1 sets the mode to FRZN for that level.

DRLS?

AC tests Perform an input voltage test
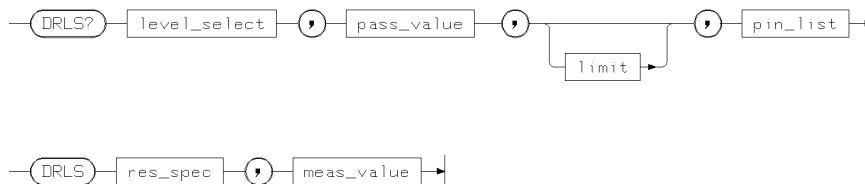
where

level-select is either H or L to specify the level to be measured
pass-value   is the pass limit, specified in millivolts
limit   either H or L to specify the limit to be used for a binary search.

The return value consists of `res-spec` which can be one of:

EQ   The value could be measured
GT   if all tests failed, level select = H
GE   if all tests passed, level select = L
LE   if all tests passed, level select = H
LT   if all tests failed, level select = L

The `meas-value` parameter returns the actual measured value in millivolts.

Errors

In a multi-mainframe system, only the master mainframe responds to this command.

`Pin-list` must contain pins with `i`, `io`, `iol`, `ioh`, or `nc` mode only.

If the value ranges are exceeded for `pass-value` or `limit`, no test will be run.

Example

See Chapter 9 for an example of command usage.

DRLV, DRLV?

Level Setup Driver levels

Note This command is only included for compatibility with earlier
software releases. Use the DRLX and DRLX? commands in place
of these commands.

These commands are used to set up and query the voltage settings of the system
driver channels (DUT inputs). The default values and setting ranges are shown
below.

```
──( DRLV )──┬──────────────────┬──( , )──┬──────────────────┬──( , )──┤ pin_list ├──►
            └─┤ logic_0_level ├─┘         └─┤ logic_1_level ├─┘
```

```
──( DRLV? )──┤ pin_list ├──►
```
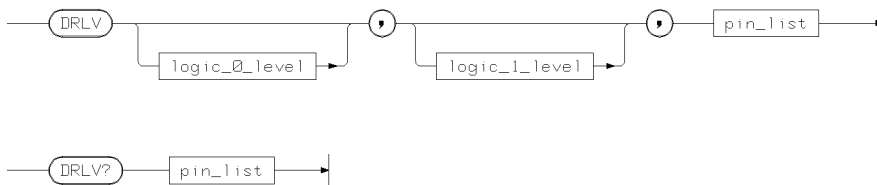
The following table shows the value ranges and default (power-on) values for
the driver level settings.

Table 12-1. Drive Level Settings

Parameter	Unit	Range (200 MHz)	Range (50 MHz)	Default Value
logic-0-level	mV	-4000..7800	-2000..6500	-1700
logic-1-level	mV	-3800..8000	-2500..7000	- 800

There are swing limitations which mean that the minimum swing between
`logic-0-level` and `logic-1-level` is 200 mV (500 mV for 50 MHz boards) and
the maximum permitted swing is 8 V (7 V for the 50 MHz boards). Violations of
these minimum or maximum swing values will cause a semantic error.

The optional parameters, `logic-0-level` or `logic-1-level` can be omitted. If
this occurs, the firmware can auto-correct this parameter if it needs to. This
may be necessary to ensure that programmed values are valid.

When a channel is configured to active termination (TERM), the drive part of the channel is forced to a static low level. The single level is programmed via the first parameter (logic-0-level). The second parameter is ignored and will be forced to an auto-corrected value depending on the required termination level. Querying the settings of such a channel returns the single level as the first parameter, the second parameter will be omitted.

The DRLV? command returns the high and low levels that have been set for the pins in pin_list.

Errors

Errors will be generated for:

■ low level out of range
■ high level out of range
■ swing violation
■ attempting to program a non-applicable pin (configured as O, DC, or OFF)

Warnings

Warnings will be generated if:

■ Auto correction occurs
■ Level programming exceeds the hardware specs

Example

```
DRLV 0,5000,(A0,A1)
```

This example forces the 2 address channels (A0 and A1) to output 0 Volt if a digital 0 (low) and 5 Volt if a digital 1 (high) must be provided to the DUT.

```
DRLV 0,, (TC)
```

If TC is configured as TERM (active termination), the logic_1_level would be ignored anyway so it's a good idea to omit this parameter, only the logic_0_level is of interest.

DRLX, DRLX?

Level Setup Driver levels

These commands are used to set up and query the voltage settings of the system driver channels (DUT inputs). The default values and setting ranges are shown below.
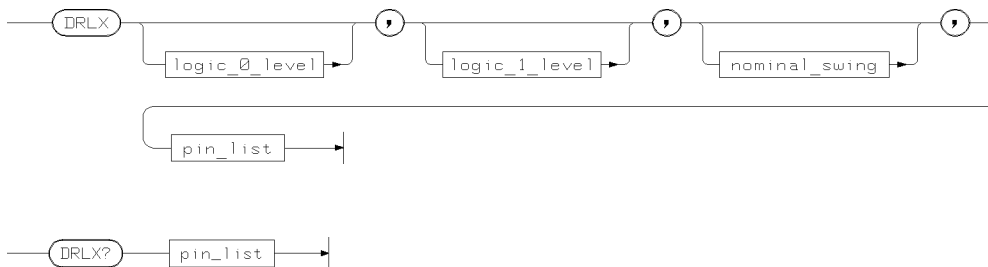
The following table shows the value ranges and default (power-on) values for the driver level settings.

Table 12-2. Drive Level Settings

Parameter	Implicit Unit	Range (D100, D200 & D400)	Range (D50)	Range (ihs pins)	Default
logic-0-level	mV	−4000..7800	−2000..6500	−4000..4800	−1700
logic-1-level	mV	−3800..8000	−2500..7000	−3800..5000	−800
nominal swing	mV	0..8000	0..7000	0..5000	

There are swing limitations which mean that the minimum swing between `logic-0-level` and `logic-1-level` is 200 mV (500 mV for 50 MHz boards) and the maximum permitted swing is 8 V (7 V for the 50 MHz boards). Violations of these minimum or maximum swing values will cause a semantic error.

The optional parameters, `logic-0-level` or `logic-1-level` can be omitted. If this occurs, the firmware can auto-correct this parameter if it needs to. This may be necessary to ensure that programmed values are valid.

When a channel is configured to active termination (`TERM`), the drive part of the channel is forced to a static low level. The single level is programmed via the

first parameter (`logic-0-level`). The second parameter is ignored and will be forced to an auto-corrected value depending on the required termination level. Querying the settings of such a channel returns the single level as the first parameter, the second parameter will be omitted.

The DRLX? command returns the high and low levels that have been set for the pins in `pin_list`.

Errors

Errors will be generated for:

■ low level out of range
■ high level out of range
■ swing violation
■ attempting to program a non-applicable pin (configured as O, DC, or OFF)

Warnings

Warnings will be generated if:

■ Auto correction occurs
■ Level programming exceeds the hardware specs

Example

```
DRLX 0,5000,,(A0,A1)
```

This example forces the 2 address channels (A0 and A1) to output 0 Volt if a digital 0 (low) and 5 Volt if a digital 1 (high) must be provided to the DUT. The timing reference will be set to the 50% level.

```
DRLX 0,,,(TC)
```

If TC is configured as TERM (active termination), the logic_1_level would be ignored anyway so it's a good idea to omit this parameter, only the logic_0_level is of interest.

DRTM, DRTM?

Timing Setup Driver timing

These commands set up and query the timing and signal formats of system driver outputs.

Each system channel can have its own individual format and edge timing settings.
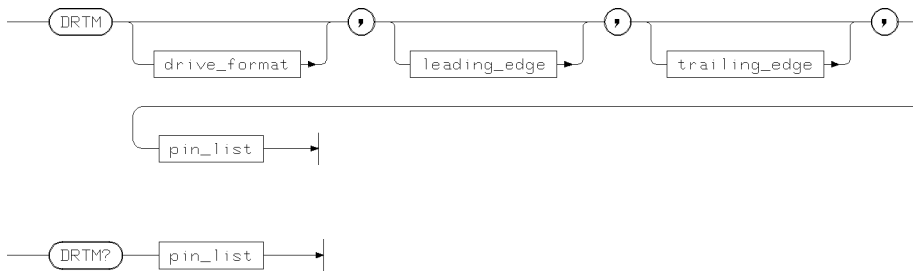
`drive_format` can be one of:

- DNRZ - for delayed non-return to zero format
- RZ - for return-to-zero format
- R1 - for return-to-one format
- RC - for return-to-complement format
- RI - for return-to-inhibit format (50 MHz only)

Allowed settings for different system operating modes are shown in the table below.

All edges (driver and receiver) are programmed with the implicit unit ns, for example, DRTM RZ, 0, 5.05, (Pin) will set the machine channel signal Pin so that the leading edge is at 0 ns delay and the trailing edge at 5050 ps.

Programmed edges that do not have the necessary HW resources, for example, no tristate edge (TE) in the case of FD or MUX configured pins and DNRZ format or no TE in case of EDGE compare, are ignored and no warning message is generated.

Pins that are configured for serial scan are handled in the same way but you should remember that LSSD execution is possible only up to a frequency of 25MHz.

The DRTM? query returns the settings for the drive format and the leading and trailing edges for the named pins.

Errors

Errors will be generated if:

■ Edges not compatible with format and operation mode
■ All range violations see table above
■ Command not applicable for this channel
    □ applied to O, TERM, DC, or OFF configured pin

Autocorrection

Contextual edge mismatching generated by using optional edge parameters is autocorrected in the following manner:

If the value of a programmed edge exceeds the allowed range, the optional parameter is set to the maximum possible value. On the other hand, if a programmed value goes below the minimum, the optional parameter is autocorrected to the minimum possible value.

Examples

Example 1

```
DFPN 1,,(NOT_CS)
CONF I,STD,,(NOT_CS)
SCLK INT,10,
DRTM RZ,6,15,(NOT_CS)
DRTM ,0,,(NOT_CS)
```

This would result in TE - LE being greater than
(LE + period - 0.5 ns), so autocorrection sets
TE = LE + period - 0.5 ns.

Querying the setting at this point with DRTM? (NOT_CS)

would give DRTM RZ,0,9.5,(NOT_CS).

Example 2

```
DRTM DNRZ, 10, 15.50, (A0, A1)
```

The above example forces the 2 address channels (tester driver channels A0 and A1) to output pattern in DNRZ format with a delay of 10ns.

DRTM, DRTM?

Note that the trailing edge is programmed to give timing information for the tristate data bit which could set the output to tristate at 15.50 if it is set.

DRVD, DRVD?

Vector Setup Driver data

These commands are used to transfer driver vector data for one pin from or to the host using a packed binary data format.
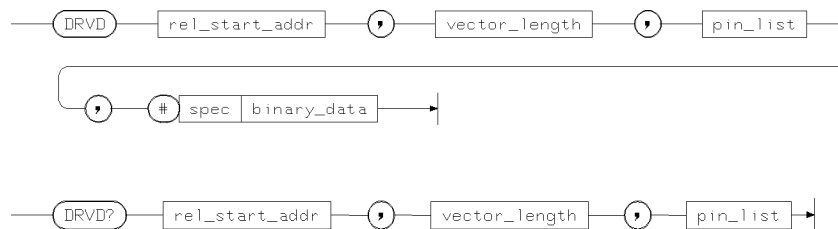
`rel_start_addr` is an integer in the range 0 .. highest vector memory address (65535 for 64k IO Boards) (`V_LAST`)

`vector_length` is an integer in the range 1 .. size of vector memory (65536 for 64k IO Boards) (`V_MEM`)

These commands are used to transfer vector data via the HP-IB. The data itself is transferred in an unreadable (non ASCII) format and will be transferred as an entire block for the single pin specified in `pin_list`.

The `pin_list` parameter must specify exactly one pin for the vector setup command DRVD and may hold any number of pins for DRVD?.

The `#` marks the start of the binary data block to transfer. This is immediately followed by the `spec` parameter, which defines the number of bytes to transfer. This parameter follows the following rules: The first digit specifies the number of digits the length specifier consists of. The specifier itself has to be transferred immediately after the first `spec` digit. The vector bytes are transferred immediately after the `spec` parameter.

Note that the number of bytes has to match the length specifier value. No white space is allowed between the elements of the length specifier.

If the number of vectors to be downloaded exceeds the vector memory size, the remaining bytes are ignored and a warning message will be issued.

DRVD, DRVD?

The vector transfer starts at the byte location of the vector specified by the previously defined vector start address and the `rel_start_addr` parameter (which is regarded as an offset value, added to the vector start address).

The vector base address can be altered by the VESA command.

Note that all vectors affected by the bytes sent will be overwritten. To avoid erroneous vector ram contents, it is recommended to set both the vector start address and the `rel_start_addr` to multiples of 4 (because one byte specifies four vectors).

If the requested download operation was successful, the first free vector address will be updated to show the first unused vector ram location.

This address can also be altered by the VEFA command.

At power on the entire driver vector memory contents will be initialized to LO (no tristate) all channels.

In addition, the vector start address and the first free vector address are both set to 0.

Errors

Errors will be generated if a parameter is out of range. This can be caused by:

- Vector length exceeds data block size
- Invalid vector length (DRVD?)
- Vector start address exceeds vector memory limits (DRVD)
- More than one pin in `pin_list` (DRVD)
- Execution conflicts with current configuration for `pin_list` (DRVD)
- DMA transfer failed or aborted (DRVD, DRVD?). If this occurs, the vector memory contents may be partially overwritten if this error occurs.

Warnings

A warning will be generated if vector data has been truncated (DRVD).

Example

In this example, the vector ram of the 200 MHz (MUX) input pin A0 is loaded from vector address 0 to 15 with the first 4 bytes of each binary 1K data block. Bytes 1 through 4 are loaded to the master channel's vector ram, bytes 1025 through 1028 are loaded to the slave channel's vector ram. All other bytes are discarded and will have no effect. The vectors 0, 1, 14 and 15 will be overwritten. The first free vector address `ffvad` will be set to 14.

```
CONF I, MUX, P, (AO)
VESA 2
DRVD 0, 12, (AO), #42048[1] - followed by 2 KByte of binary data
VEFA? -> VEFA 14
```

[1] **#** specifies the start of a binary block, 4 specifies the number of digits for the length specifier, 2048 is the length of data in bytes.

Note   Vector data cannot be edited or generated using a text editor. Use the EDA Interface or Vector Setup editor to generate the binary data required.

DTCD, DTCD?

Calibration Driver timing

These commands are used to modify and read driver timing calibration values.

```
──(DTCD)──┤channel├──(,)──┬──────────────────┬──(,)──┬──────────────────────┬──(,)──┐
                          └─┤le_offset├───────┘       └─┤le_mux_offset├──────┘       │
  ┌───────────────────────────────────────────────────────────────────────────────┘
  └──┬─────────────────┬──(,)──┬──────────────┬──(,)──┬──────────────────┬──(,)──┐
     └─┤le_basepoints├──┘       └─┤te_offset├──┘       └─┤te_hiz_offset├──┘       │
  ┌───────────────────────────────────────────────────────────────────────────┘
  └──┬──────────────────┬──(,)──┬──────────────────┬──┤
     └─┤te_mux_offset├───┘       └─┤te_basepoints├──┘


──(DTCD?)──┤query_channel├──┤
```
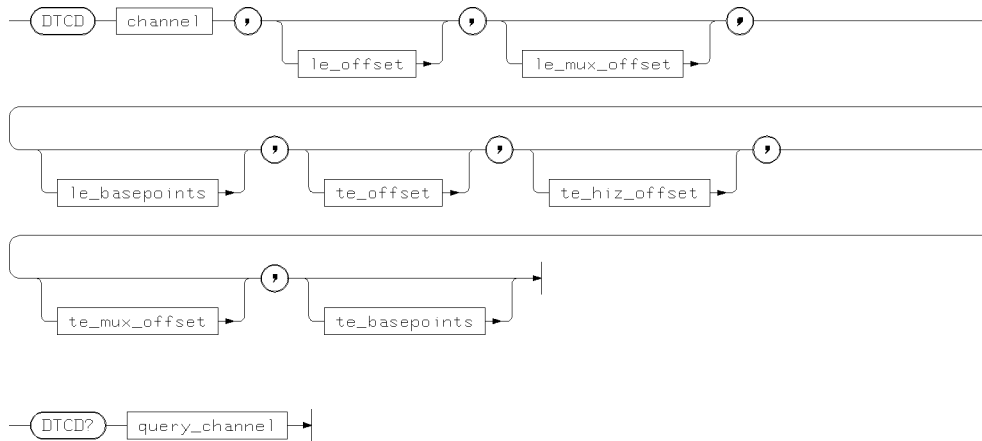
where `channel` is the sequential channel number defined with the DFPN command.

Note	There are ten values returned for `le_basepoints` and `te_basepoints`.

The values for the offsets and gains and their implied units are shown below.

Timing Parameter Value Ranges

			min	max	unit
Gain	Channel		29.41	39.22	ps/LSB
Offset	Leading Edge	Standard	0.0	+16.0	ns
		MUX Mode	-5.0	+16.0	ns
	Trailing Edge	Standard	-1.0	+16.0	ns
		MUX Mode	-6.0	+16.0	ns
	Tristate Edge		0.0	+16.0	ns

Errors

An error will be generated for the following reasons:

■ Sequencer not idle
■ Gain value out of range
■ Trying to set or read receiver calibration data for an I channel

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

DTMC?

Calibration Driver timing calibration

Performs driver timing calibration of the specified pin using the given calibration probe channel (default is channel 1).

where

`channel`	is the channel number to be calibrated
`prb_chan`	is the probe channel to be used. Default is 1.

The returned values for this command are:

`error`	an integer in the range 0 .. 99999 indicating a calibration error code. Value 0 indicates no error.
`channel`	the channel being calibrated
`probe-channel- nr`	the cal probe channel used for the calibration
`period`	the period used in nanoseconds
`le-rx`	Leading edge return to x format - master channel
`le-rxslave`	Leading edge return to x format - slave channel
`le-bpoints`	Leading edge fine delay basepoints (10 values)
`te-rx`	Trailing edge return to x format - master channel
`te-hz`	Trailing edge return to high impedance - master channel
`te-rxslave`	Trailing edge return to x format - slave channel
`te-bpoints`	Trailing edge fine delay basepoints (10 values)

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

DTST

Utility Perform dc measurement

```
──(DTST)──┤ set_id ├──┤ pin_list ├──▶│
```

This command performs a dc measurement on the pins in `pin_list` using the dc parameter set (set by DPAR) specified in `set_id`.

Example

See Chapter 7 for more details of the use of this command.

DTUC?

Calibration Generate user calibration values for a driver
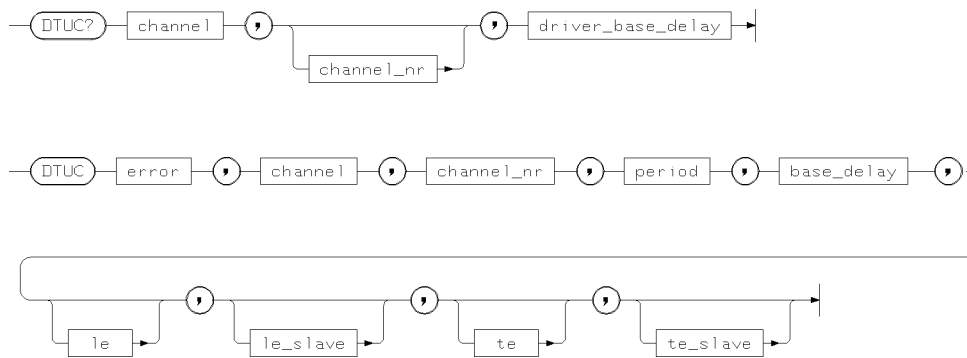
where

channel   is the channel number to be calibrated
channel_nr   is the probe channel to be used. Default is 1.
driver-base- delay
The returned values for this command are:

error   an integer in the range 0 .. 99999 indicating a
                        calibration error code.Value 0 indicates no error.
channel   the channel being calibrated
probe-channel- nr   the cal probe channel used for the calibration
period   the period used in nanoseconds
base-delay
le   Leading edge correction - master
le-slave   Leading edge correction - slave
te   Trailing edge correction - master
te-slave   Trailing edge correction - slave

Example

For more information on user calibration, refer to Chapter 10.

DTUD, DTUD?

Calibration Transfer user calibration values for a driver
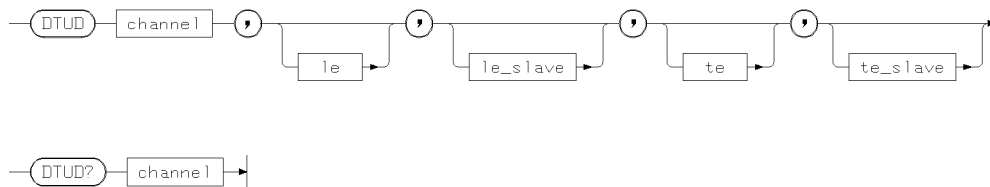
where

channel	channel whose user calibration data is to be downloaded
le	Leading edge correction in nanoseconds - master
le-slave	Leading edge correction in nanoseconds - slave
te	Trailing edge correction in nanoseconds - master
te-slave	Trailing edge correction in nanoseconds - slave

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about user calibration.

ERCT?

Test Results Read number of errors in Error Map

ERCT? returns the number of errors in the error map.

This command return the the number of errors detected in the last test. This number is an integer in the range 0 .. `size-of-error-map`

This command is only valid for compare mode.

Errors

An error will be generated for the following reasons:

- Parameter range errors
- Command not allowed in acquisition mode
- No data available

Example

`ftst?`	*Performs a functional test and returns*
`FTST F`	*if the test fails*
`erct?`	*Returns the number of errors detected*
`ERCT xx`	*xx is base 10*

ERMP?

Test Results Upload Error Map

ERMP? returns the contents of the error map in a binary format.

This command returns

`sample_start_cycle_#` is in the range 1 .. `size-of-error-map` -8.

`#_of_cycles` is in the range 1 .. `size-of-error-map`.

`spec` is in the range `digit{ digit}`

`#_of_cycles` is in the range 1 .. `size-of-error-map`.

The command is only valid for compare mode and only applicable to the master mainframe.

The transfer command ERMP sets up the mainframe to transfer a large amount of data via the HP-IB bus to be displayed in the Error Map results display. The data consists of full 8-bit bytes in an unreadable (non ASCII) format and will be transferred in one block for all the received data specified.

The '#' marks the start of the binary data.

`spec` defines the number of transferred bytes in the following manner: the first <digit> specifies the number of valid number digits, immediately followed by the digits defining the number of transferred bytes, followed by the binary data. No white space is allowed after the '#' sign!

The `sample_start_address` and the (`sample_start_address` + `#_of_cycles`) will be rounded to the next byte boundaries for binary transfers.

Note that the error map will report only one error per machine cycle. Due to this cycle oriented error map, two errors occurring in the same machine cycle (but in distinct user cycles) are sampled as a single error and the error count (accessible using the ERCT command) is incremented by one (instead of two).

This command works on byte boundaries

Errors

An error will be generated for the following reasons:

- Parameter range errors
- Command not allowed in acquisition mode
- No data available

ERRS?

Error Handling Interrogate hardware

This command is used to interrogate the hardware to see if an error has been detected.

```
──(ERRS?)──▶│
```

Sending this command will result in the following message being output

```
    ERRS error_code , command , pin_list
```

where

`error_code` is an integer value. Refer to Chapter 9 for more information.

`command` is the command that caused the error or 0 if no error has been detected.

`pin_list` will contain all 'bad pins', that is, the pins which caused an execution error, or those changed by an Autocorrection. If the error is not related to a pin_list the field will contain an empty pin_list.

The ERRS? query reads the contents of the Error Queue. The mainframe responds with all the errors queued so far and then clears the error queue.

Example

See Chapter 9 for more information on this command.

ESGB, ESGB?

AC tests Define edge search parameters
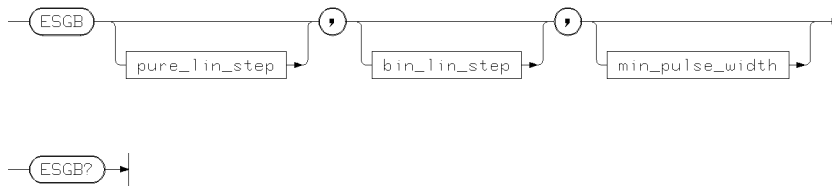
where

pure_lin_step   the step width to be used for a linear edge search in nanoseconds. See RCES?, TSUP?, and THLD? If this is negative, an infinite step width is used (see RCES?).

bin_lin_step   the step width used for a linear/binary search in nanoseconds.

min_pulse_width   the minimum pulse width to be used for an input pin timing measurement. (See TSUP? and THLD?)

The query ESGB? returns the current settings.

Power On Defaults

At power-on, the current settings are:

```
ESGB -1,2,2.5     D100, D200 and D400
ESGB -1,4,5       D50
```

EXPD, EXPD?

Vector Setup Transfer receiver data

These commands are used to transfer receiver vector data for one pin from or to the host using a packed binary data format.
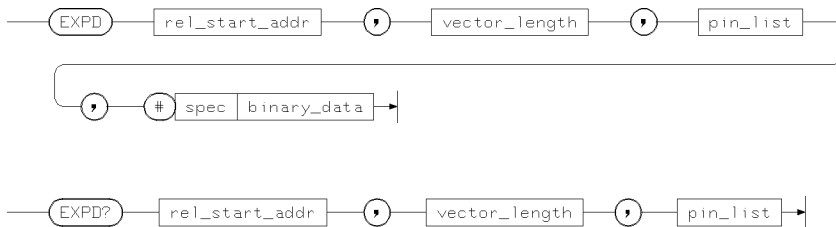
The vector data transfer command EXPD and the corresponding query EXPD? are used to transfer large size vectors via the HP-IB. The data itself is transferred in an unreadable (non ASCII) format and will be transferred as an entire block for the specified pin.

The `pin_list` parameter must specify exactly one pin for the setup command EXPD while the `pin_list` for the query EXPD? may hold any number of pins.

The `#` marks the start of the binary data block to transfer. It will be immediately followed by the `spec` parameter, which defines the number of bytes to transfer. This parameter follows the following rules: The first digit specifies the number of digits the length specifier consists of. The specifier itself has to be transferred immediately after the first `spec` digit. The vector bytes are transferred immediately after the `spec` parameter.

Note that the number of bytes has to match the length specifier value. No white space is allowed between the elements of the length specifier.

The vector data block itself will be loaded to the vector ram unchanged. Therefore the contents of the data stream depends on the hardware vector ram organization. Refer to the Vector Data Format section at the beginning of this chapter for a detailed explanation of the data block format used for the various pin configurations.

If the number of vectors to be downloaded exceeds the vector memory size, the remaining bytes are ignored and a warning message will be issued.

The vector transfer starts at the byte location of the vector specified by the previously defined vector start address and the `rel_start_addr` parameter (which is regarded as an offset value, added to the vector start address).

The vector base address can be altered by the VESA command.

Note that all vectors affected by the bytes sent will be overwritten. To avoid erroneous vector ram contents, you should set both the vector start address and the `rel_start_addr` to multiples of 4 (because one byte specifies four vectors).

If the requested download operation was successful, the first free vector address will be updated to show the first unused vector ram location.

This address can also be altered by the VEFA command.

At power on the entire receiver vector memory contents will be initialized to X and the vector start address and the first free vector address are both set to 0.

Errors

Errors will be generated if:

- A parameter is out of range
  - Vector length exceeds data block size (EXPD)
  - Invalid vector length (EXPD?)
  - Vector start address exceeds vector memory limits (EXPD)
- More than one pin in `pin_list` (EXPD)
- Tester is not in real-time compare mode (EXPD, EXPD?)
- Execution conflicts with current configuration for `pin_list` (EXPD)
- DMA transfer failed or aborted (EXPD, EXPD?). In this case the vector memory contents may be partially overwritten if this error occurs.

Warnings

A warning will be generated if vector data has been truncated (EXPD)

Example

In the following example, the vector ram of the 200 MHz (MUX) input pin A0 is loaded from vector address 0 through 15 with the first 4 bytes of each binary 1K data block. Bytes 1 through 4 are loaded to the master channel's vector ram, bytes 1025 through 1028 are loaded to the slave channel's vector ram. All other bytes are discarded and will have no effect. The vectors 0, 1, 14 and 15 will be overwritten. The first free vector address `ffvad` will be set to 14.

EXPD, EXPD?

```
CONF I, MUX, P, (A0)
VESA 2
EXPD 0, 12, (A0), #42048¹ followed by 2 KByte of binary data
VEFA?  now returns VEFA 14
```

[1] **#** specifies the start of a binary block, 4 specifies the number of digits for the length specifier, 2048 is the length of data in bytes.

Note   Vector data cannot be edited or generated using a text editor.
                Use the EDA Interface or Vector Setup editor to generate this
                binary data.

FTCK?

AC Tests Run a functional test and not stop clock

FTCK?

This command can be used when you have a vector pattern that exceeds the length of the pattern memory. Unlike the FTST? command, the FTCK? command does not stop the clock before starting the sequencer. In addition, the error flags (accessible with the CHER? command) do not get reset and will hold the total number of errors when the test finishes.

Note	The FTCK? command is meant to be used in real-time compare mode. If it is used in data acquisition mode, the acquired data may be invalid.

Example

The following example illustrates the use of the FTCK? command.

`DRVD 0,65536,(Din),#516384....`	*load first part of vectors*
`EXPD 0,65536,(Dout),#516384....`	
`SQSL "DEFAULT"`	*perform functional test*
`FTST?`	
`ERCT?`	*read error counter for first part of test*
`DRVD 0,65536,(Din),#516384....`	*load second part of vectors*
`EXPD 0,65536,(Dout),#516384....`	*perform functional test without stopping the clock*
`FTCK?`	
`ERCT?`	*read error counter for second part of test*
`CHER?`	*read channel error flags for complete test*

FTST?

AC tests Perform functional test

FTST? **AC tests** Perform functional test

```
─(FTST?)─►|
```

This command will perform a functional test with the current settings.

Example

```
ftst?     Performs a functional test and returns
FTST F    if the test fails
```

The number of errors can be read by using ERCT?

GETV?

Test Results Vector and sequencer line number

GETV? returns the vector and sequencer instruction number for the given
cycles. Although the number of cycles to the logical 0 address is reported.

This results in the following message being sent by the system.

`GETV cycle_#, vector_#, instr_#, instr_type, rep_factor` where

`#_of_cycles`	is in the range 1 .. highest vector memory address
`vector_#`	is in the range 0 .. size of vector memory
`instr_#`	is in the range 0 .. 2^{11}
`instr_type`	is one of INC, HOLD, SCAN, or DEAD
`repeat_factor`	is in the range 1 .. last cycle number
`cycle_#`	is in the range 0 .. last cycle number
`#_of_cycles`	is in the range 1 .. highest vector memory address

The GETV? query returns the cycle - vector relationship in a compressed form.
It reports the cycle number after the logical 0 address, the vector number, the
sequencer instruction number and gives the information how the next cycles
were generated by giving the mode how the next cycle was generated and how
often (see examples).

Errors

An error will be generated for the following reasons:

- Parameter range errors
- No data available

GETV?

Example

In this example of part of a vector sequence, the cycle-vector relationship will be reported as shown below.

```
cycle #    vector #  sequ instr # instr type
 800     ??          ??             ??
 more lines
 900     ??          0              inc
 more lines
1000      2          1              hold
1001      2          1              hold
1002      2          1              hold
1003      2          1              hold
1004      BREAK             2            dead cycle
1005      4          2              inc
1006      5          2              inc
1007      6          2              inc
1008      7          2              inc
1009      8          2              inc
1010      9          2              inc
1011     10          2              inc
```

Sending GETV? 800, 210

will result in

```
GETV    800,,,,100
GETV    900,,0,INC,100
GETV    1000,2,1,HOLD,4
GETV    1004,-1,2,DEAD,1
GETV    1005,4,2,INC,6
```

HRTL?

Test Functions Read limitations imposed on the High Res. Timing Diagram by hardware.

Returns the string:

Where:

max_no_pins	is the maximum number of pins which can be measured in parallel, by a single HRTM command.
max_samples	is the maximum number of samples that can be measured in parallel, by a single HRTM command.

HRTM

Test Functions Run high res timing diagram

```
─( HRTM )─┤ pinlist ├─►
```
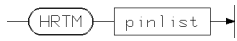
Errors

■ Pin not applicable
■ Measurement failed
■ Too many pins
■ Invalid tester mode

Example

Refer to Chapter 7 for an example of the use of this command.

HRTP, HRTP?

Test Functions Setup high res timing diagram

```
(HRTP)──(,)──[cycles]──(,)──[cycle_offset]──(,)──[samples_per_cycle]──(,)─┐
  ┌──────────────────────────────────────────────────────────────────────┘
  └──[samples]──(,)──[sample_offset]──►
```

```
──(HRTP?)──►
```
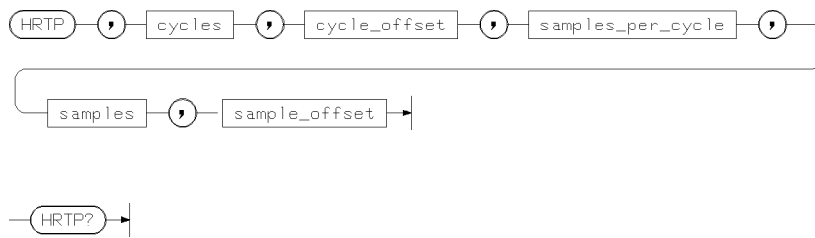
Errors

Value range errors.

Example

Refer to Chapter 7 for an example of the use of this command.

HRTR?

Test Functions Read high res timing diagram

```
──( HRTR? )──►│
```

This command returns the data to be displayed in the Timing Diagram results display.

The data is in an unreadable format, similar to the vector transfer commands.

Example

Refer to Chapter 7 for an example of the use of this command.

HWEN, HWEN?

Status Hardware Status Enable Register

These two commands set and read the Status Enable Register of the Hardware Status.
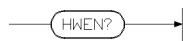
Example

Refer to Chapter 9 for more details.

HWET?

Status Hardware Event Register

This command reads the Event Register of the Hardware Status Register.

```
──( HWET? )──▶│
```

Example

Refer to Chapter 9 for more details.

HWFL, HWFL?

Status Hardware Status Register

These commands set and define the falling transition event generation of Hardware Status Register.

Example

Refer to Chapter 9 for more details.

HWRS, HWRS?

Status Hardware Status Register

These commands define and read the rising transition event generation of Hardware Status Register.

```
──(HWRS)────| hardware_status |────▶|
```

```
──(HWRS?)────▶|
```

Example

Refer to Chapter 9 for more details.

HWST?

Status Hardware Status

This command reads the Hardware Status of the Hardware Status Register.

—(HWST?)———▸|

The bits in the Hardware Status Register have the following meaning: (This is the meaning, when the bit is set; the meaning when the bit is cleared is the opposite)

- [1] Relays active, the relays of connected channels are closed.
- [2] Channel drivers active and driving programmed levels.
- [3] clock active (running)
- [4] Sequencer busy (not BREAK), armed or active, test has not finished
- [5] Sequencer active (not ARMED and not BREAK))

Example

Sending HWST? while the sequencer is in BREAK will return 4 (bit3). After sending `sqst off`, HWST? will return 0.

IFC

HP-IB Commands Interface Clear

```
─( IFC )─►|
```

```
    IFC
```

Example

Refer to the BASIC-UX Interfacing Techniques manual for more information on this command.

ITMC?

Calibration Initialize calibration process

```
─(ITMC?)─┤ cal_period ├─▶│
```

This command performs timing calibration of the calibration probe and sets
the system into calibration mode. The given calibration period is used for all
subsequent calibration commands. If this command is sent with the u option, a
user calibration will be started.

This command returns

■ a calibration error number in the range 0 .. 999999, 0 indicates no error
■ the number of usable calibration probe channels
■ the period in use
■ the round-trip delay measured
■ leading and trailing edge basepoints

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about
system calibration.

IXMD?

Configuration List configuration of IX mode pins.

```
 ─(IXMD?)─ pinlist ─►
```

Returns:

```
─(IXMD)─(,)─ pulse_width ─(,)─ low_level ─(,)─ high_level ─(,)─┐
         ┌─────────────────────────────────────────────────────┘
         └─ relay_state ─(,)─ channel_no. ─(,)─ pin_list ─►
```
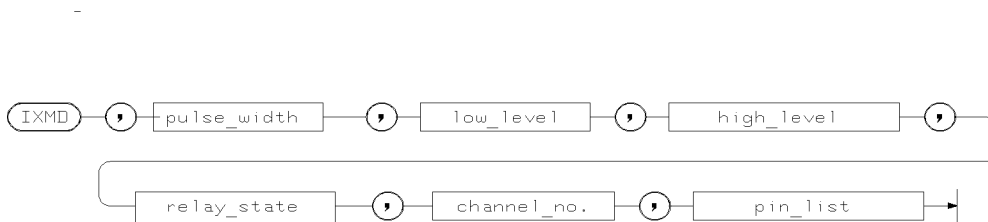
where:

pulse_width   is the programmed pulse width of the 400 MHz RZ channel (in
               ns). A value of -1 shows that the channel is in AC feed-through
               mode.
low_level   is the programmed low level of the High Speed Width
               Generator, HSWG (in mV).
high_level   is the programmed high level of the High Speed Width
               Generator, HSWG (in mV).
relay_state   is the state of the HSWG relays:
               ■ 0 = disconnected (pin is inactive)
               ■ 1 = connected (pin is active)
               ■ 2 = non-inverted output is connected to PMU
               ■ 3 = inverted output is connected to PMU
               ■ 6 = both HSWG outputs are connected to PMU
channel_no   is the tester channel to which the IX pin is connected.

LDHW, LDHW?

High Throughput Tests Load the hardware with a parameter set which was previously saved with PSAV.

```
──(LDHW)──┤ param_set_id ├─►
```

```
──(LDHW?)──┤ param_set_id ├─►
```
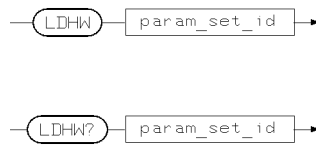
where:

param_set_id   is the id number (1 to 64) of the parameter set to be loaded. The id number "0" loads the hardware with the *foreground* settings.

The query LDHW? returns the id number of the parameter set which is presently loaded in the hardware.

Example

To load the parameter set with the id number 24 into the hardware, you enter:

LDHW 24

To find out the id of the parameter set which is presently loaded in the hardware, you can enter the query:

LDHW?

If parameter set 24 is still loaded, the returned string would be:

LDHW 24

MFLC?

Calibration Perform mainframe level calibration

```
──( MFLC? )──┤ mainframe ├─►
```

The mainframe level calibration command MFLC? returns the clock driver and
cal-receiver calibration values. These values must be downloaded (see MLCD)
prior to a subsequent timing calibration.

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about the
use of this command.

MLCD, MLCD?

Calibration Read and modify mainframe level calibration
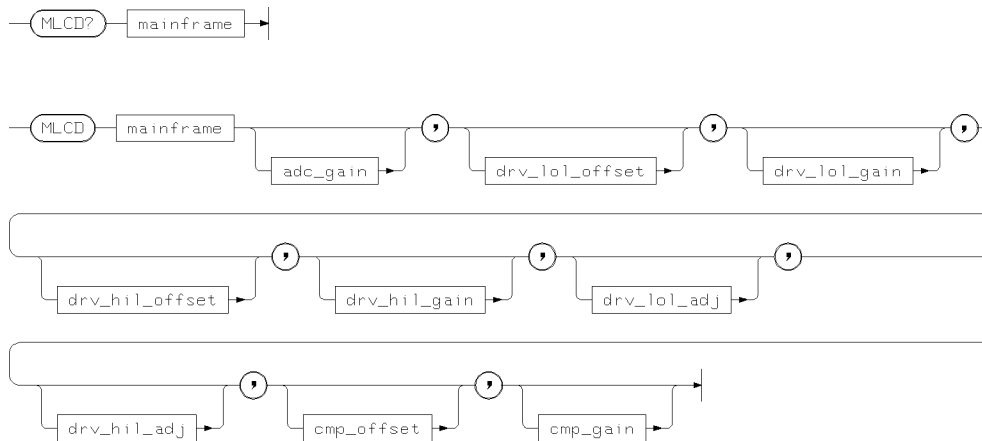
The mainframe level calibration values are accessible with the MLCD command. One set of calibration data is held for each mainframe. The calibration values include the clock driver data (`drv-lol-offset`, `drv-lol-gain`, `drv-hil-offset`, `drv-hil-gain`, `drv-lol-adj`, `drv-hil-adj`), the calibration comparator data (`cmp-offset`, `cmp-gain`), and the clock board ADC gain (`adc-gain`).

Errors

An error will be generated for the following reasons:

■ error

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

MTCD, MTCD?

Calibration Read or modify mainframe timing data
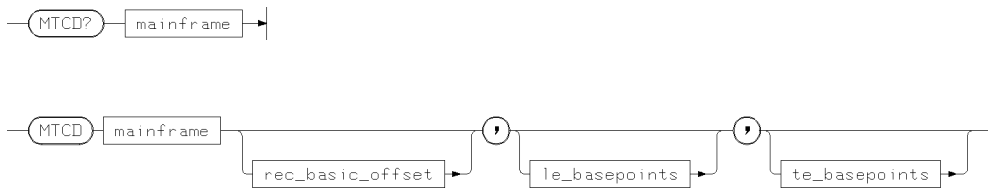
This command is used to read and modify timing calibration data for a mainframe.

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

PASS?

Test Results Last test result

PASS? returns passed / failed information for the last test.

returns either P or F depending on whether the last test passed or failed.

The command is only valid for compare mode.

Errors

An error will be generated for the following reasons:

- Command not allowed in acquisition mode
- No data available

Example

Sending `PASS?` after a test has been performed returns the string `PASS F` or `PASS P` depending on whether the test failed or passed.

PATR, PATR?

Attributes Set or read pin attributes
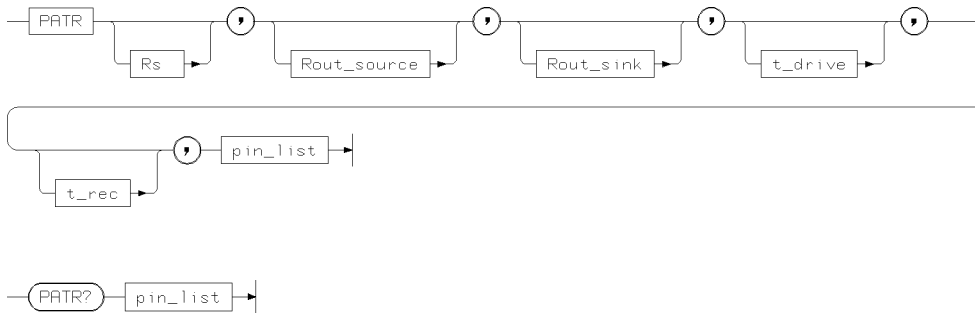
Rs   is a real in the range 0 .. 6553.5 with the implied units Ohm
Rout-source   No longer used. Leave this parameter blank.
Rout-sink   No longer used. Leave this parameter blank.
t-drive   is a real in the range -32.767 .. +32.767 with the implied units nanoseconds
t-rec   is a real in the range -32.767 .. +32.767 with the implied units nanoseconds

Errors

An error will be generated for the following reasons:

■ Parameter out of range
■ Attempting to program a non-applicable pin (DC or OFF)

Example

PATR? (@) returns the current settings of pin attributes for all configured
pins. Files containing this data and system attribute data (see SATR) should be
written using the Pin Attributes setup window, or using a text editor and stored
in the pin_attributes directory for the DUT.

PDFT?

AC tests Perform a propagation delay/hold time test

where

`tpd`	propagation delay pass value in nanoseconds
`tdh`	data hold time pass value in nanoseconds
`pin_list`	pins to be tested
`test_result`	P = passed; F = failed.

Errors

An error will be generated for the following reasons:

■ Non-applicable pins

Example

`PDFT? 10,2,(Q0,Q1,Q2,Q3)` will check that the propagation delay is less than 10 nanoseconds and the data hold time less than 2 nanoseconds for the pins Q0 to Q3.

PMUM?

DC tests Read PMU state

```
─( PMUM? )──│ pmu_id │─▶│
```
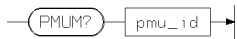
where `pmu_id` has the form `Pmp` where `m` is the mainframe whose PMU is being read, and `p` is the PMU channel.

Sending this command results in the following string being sent by the hardware.

`PMUM pmu_id, pmu_state, u_read, i_read`

where

`pmu_id`	is the PMU being read as described above
`pmu_state`	is the current PMU state in an integer in the range 0 .. 31 that must be read in binary format:
	Bit 5 - Voltage overflow if set
	Bit 4 - Current overflow if set
	Bit 3 - Measurement unstable if set
	Bits 2 and 1 - limit detection
	00 - not used
	01 - Negative current limit
	10 - Positive current limit
	11 - Voltage limit
`u_read`	measured voltage in millivolts
`i_read`	measured current in microamperes

Example

Sending `PMUM? P12` results in the output string `PMUM P12,1,-5.012,0` which indicates the state of the second PMU in mainframe 1.

PMUS

DC tests Set up PMU

where

pmu_id	is the PMU being read. It has the form Pmp where m is the mainframe whose PMU is being read, and m is the PMU channel.
u_force	the force voltage in millivolts. Must be in the range -20000 to +20000.
u_range	the voltage range to be used. This has an effect on the rounding that will occur to the force value being programmed.

    U2E3 $\pm$ 2000 mV
    U1E4 $\pm$ 10000 mV
    U2E4 $\pm$ 20000 mV
    AUTO autoranging - will use the smallest range possible to force u_force.

i_force	the force current in microamperes. Must be in the range 0 to 200000.
i_range	the current range to be used. The resolution of a measurement is fixed at 1/4000 of the specified range.

    I2E0 0.001 μA .. 2 μ A
    I2E1 0.01 μA .. 20 μ A
    I2E2 0.1 μA .. 200 μ A
    I2E3 1 μA .. 2000 μ A
    I2E4 10 μA .. 20000 μ A
    I2E5 100 μA .. 200000 μ A
    I5E5 200 μA .. 500000 μ A
    AUTO autoranging - uses the lowest range possible for the required current

PSAV

High Throughput Save the hardware state as a set of ac test parameters (a *parameter set*) in RAM;
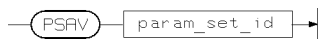
The PSAV command saves the *foreground* settings of the tester hardware, into RAM, and assigns an id number (between 1 and 64) to enable these settings to be quickly reloaded during a test.

The foreground settings are the values that you enter using the interactive windows, or via the other HP-IB commands.

Note After a system reset, all 64 parameter sets are loaded with the foreground values.

Example

To save the present foreground settings as a parameter set with the id number 24, you can enter the instruction:

PSAV 24

PSCF?

Pin Configuration Read DPS channels

This command reads the number of channels available in the DPS. For each DPS unit fitted to the system, this query returns the string:

`PSCF dps_name, nr_of_channels`

where

`dps_name`   is the model number of the installed DPS
`nr_of_channels`   is the number of outputs available

PSLV, PSLV?

Level Setup Read and modify DPS settings

```
PSLV ──┬─────────┬──,──┬────────┬──,──┬───────────┬──,──┬──────────────┬──,──
       └─ volt ──┘     └─ curr ─┘     └─ off_imp ─┘     └─ setup_time ─┘

   ──┬─────────────┬──
     └─ pinlist ───┘
```

```
── PSLV? ── pinlist ──┤
```
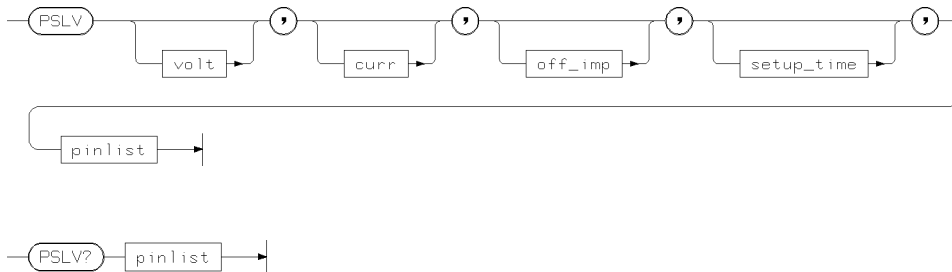
These commands are used to read and modify DPS channels that have
previously been defined with a DFPS command.

volt   real number in the range 0 to 40000 to specify the output
                voltage in millivolts.

Note   Maximum voltage output is limited to 40 Volt for power supplies
☞   with higher levels available.

---

curr   current limit in Amperes - maximum depends on power supply
                in use.
off_imp   either HIZ or LOZ to specify high or low impedance in off state.
setup_time   specifies an additional setup time between applying power and
                starting a test. Implied units are milliseconds.

Example

Entering

```
PSLV -5.200,0.500,hiz,15,(Vee)
```

sets the previously defined power supply pin Vee to a level of -5.2 Volt and
a current limit of 0.5 A. When the test system is in the disconnected state,

the power supply will be set to its minimum output current. When the power supply is connected or disconnected, there will be an additional delay of 15 milliseconds before the next command is executed.

PSME?

DC measurements Perform measurement with DPS

Sending this command will result in the following string being output for each pin in `pin_list`.

`PSME status, voltage, current, pin`

where

`status`	is either CV for constant voltage or CC for constant current.
`voltage`	real number indicating DPS output voltage with polarity as given in DFPS command.
`current`	real number indicating current flow at DPS output.
`pin`	DPS pin being read

PSST, PSST?

Utility Set and read state of DPS relays
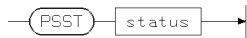
The PSST command switches the DPS relays to connect the DPS (state = ON) or disconnect the DPS (state = OFF). The DPS settings will be as set up with the relevant PSLV commands.

PSST returns the current state of the DPS relays.

Errors

Parameter error

No power supplies defined

Warnings

PSST OFF and a DPS output is in CC mode.

PSST OFF and the power supply is in the power on state.

PTST?

High Throughput Tests Execute a PMU test from pre-saved dc parameters that you have previously entered with the DFVM and DFCM commands.

```
—(PTST)—| dc_set_id |—| result |—| pin_list |—►|
```

Returns the parameters:

```
—(PTST?)—| dc_set_id |—| test_mode |—►|
```

where:

- `dc_test_set` is the id number of the dc parameter set that has been used in the test;
- `mode` is either:
  - □ `gpf` (global pass/fail)—for a single pass/fail result;
  - □ `ppf` (per pin pass/fail)—for a pass/fail result per pin;
  - □ `pval` (per pin value)—for a value result per pin.
- `result` is either `P` for a passed test, `F` for a failed test, or the measured value;
- `pin list` is a list of the pins to which *result* applies. If the test mode `gpf` was chosen for the test, then *result* applies to all the pins, and *pin list* is returned empty.

PULL, PULL?

Level Setup Set/read driver pull-up or pull-down resistor (D50 only).

```
─(PULL)─┤ state ├─┤ pin_list ├─▶
```

```
─(PULL?)─┤ pinlist ├─▶
```
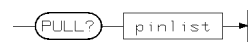
where:

state can be :

- UP - to connect a 1 KΩ pull-up resistor (from the driver output to +5V.)
- DOWN - to connect a 1 KΩ pull-down resistor (from the driver output to 0V.)
- INT - to connect 1 KΩ resistors from the driver output to +5V and 0V.
- OFF - no pull-up or pull-down resistors connected.

RCES?

AC tests Perform linear search

```
─(RCES?)──┤start├──(,)──┤stop├──(,)──┤mode├──(,)──┤pin_list├─►
```

```
─(RCES)──┤rel_op├──(,)──┤meas_val├─►
```

This command performs a propagation time/data hold time measurement using a linear search pattern.

start   is a real number with implied units nanoseconds used to define the start of a window

stop   is a real number with implied units nanoseconds used to define the end of a window

mode   either L or LB to specify a linear or linear/binary search sequence

pin_list   the pins on which the measurement is to take place

Errors

An error will be generated for the following reasons:

■ Non-applicable pin

RCLC?

Calibration Perform receiver level calibration

```
─(RCLC?)─┤query_channel├─▶
```

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

RCLS?

AC tests Perform an output voltage test
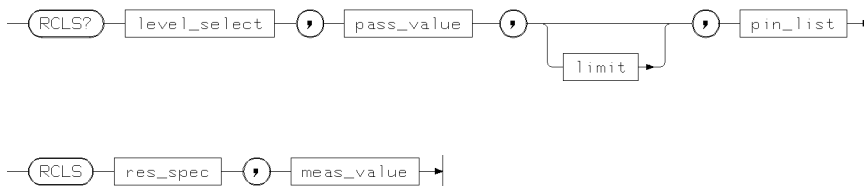
```
─(RCLS?)─┤level_select├─(,)─┤pass_value├─(,)─┬───────────┬─(,)─┤pin_list├─┤
                                             └─┤limit├───┘
```

```
─(RCLS)─┤res_spec├─(,)─┤meas_value├─┤
```

where

`level-select`	is either H or L to specify the level to be measured
`pass-value`	is the pass limit, specified in millivolts
`limit`	either H or L to specify the limit to be used for a binary search.

The return value consists of `rel-spec` which can be one of:

EQ	The value could be measured
GT	if all tests failed, level select = H
GE	if all tests passed, level select = L
LE	if all tests passed, level select = H
LT	if all tests failed, level select = L

The `meas-value` parameter returns the actual measured value in millivolts.

Errors

In a multi-mainframe system, only the master mainframe responds to this command.

`Pin-list` must contain pins with `o`, `io`, `iol`, `ioh`, or `nc` mode only.

If the value ranges are exceeded for `pass-value` or `limit`, no test will be run.

Example

See Chapter 9 for an example of command usage.

RCLV, RCLV?

Level Setup Receiver Levels

These commands are used to set up and query the threshold settings of the system receiver channels (DUT outputs). The default values and setting ranges are shown below.

```
RCLV ──┬──────────────────┬── , ──┬──────────────────┬── , ── pin_list ──►
       └── logic_0_threshold ──┘      └── logic_1_threshold ──┘
```

```
RCLV? ── pin_list ──►
```
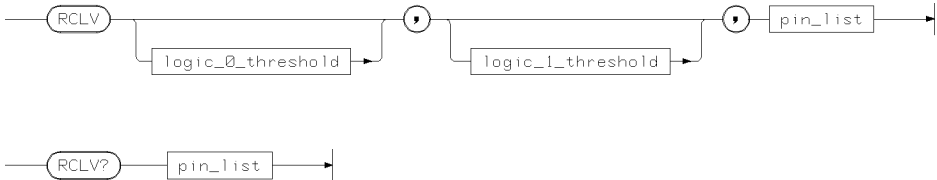
Table 12-3. Receiver Threshold Settings

Parameter	Implicit Unit	Range (D100,D200,D400)	Range (D50)	Default Value
logic-0-threshold	mV	−4000..7994	−2000..6950	−1.470
logic-1-threshold	mV	−3994..8000	−2010..7000	−1110

As in the driver level commands, there is a swing limitation such that there must be a minimum swing between logic-0-threshold and logic-1-threshold. The minimum swing limitation depends on the defined pin's attributes. The minimum threshold swing is 6 mV (10 mV for 50 MHz boards). See the pin attribute chapter for the relationship between the pin and comparator threshold values.

In the case of 200 MHz operation_mode the necessary single threshold is set by the logic_1_threshold parameter. The first parameter logic_0_threshold is ignored and silently autocorrected to the necessary value depending on the given single threshold. Querying such a configured pin returns the single threshold with the logic_1_threshold parameter and the first parameter logic_0_threshold will be omitted.

RCLV, RCLV?

Errors

Errors will be generated for:

- low level out of range
- high level out of range
- swing violations
- attempting to program non-applicable pins (configured as I, DC, OFF)

Warnings

A warning will be generated if:

- auto-correction has occurred
- level programming exceeds the hardware specifications

Example

```
RCLV 400, 4800, (D0, D1)
```

All levels received at D0 and D1 (O pins) lower than 0.4 Volt will be recognized as a digital 0 (low) and greater than 4.8 Volt as a logical 1 (high). A stable level in between these two levels will be interpreted as intermediate.

RCMD, RCMD?

Test results Set receive mode

`receive_mode` either RTC to set real time compare mode, or DA for data
acquisition mode.

In RTC mode, the DUT response will be compared against the expected data
downloaded previously using the EXPD command.

In data acquisition mode (DA), the DUT response will be stored in the received
data memory. The RECD? query may be used to examine the sampled data.

Example

Sending the command RCMD? responds with either RCMD RTC for real time
compare mode, or RCMD DA for data acquisition mode.

RCTM, RCTM?

Timing Setup Receiver timing

These commands are used to set and query receiver timing and compare mode settings.
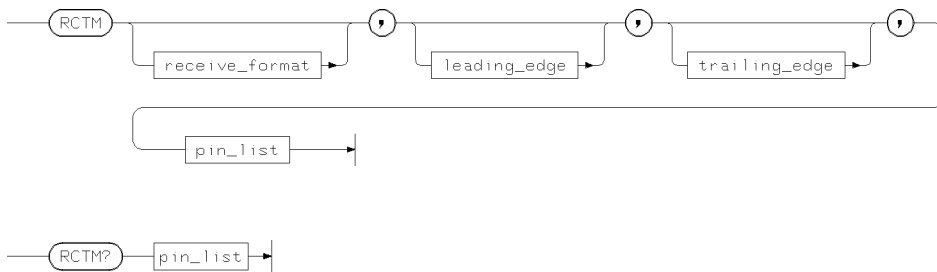
`receive_format` can be either WIDW for window compare mode or EDGE for edge compare mode.

Programmed edges that do not have the necessary HW resources are ignored and no warning message is generated.

The default receiver timing is EDGE format with LE= 0.

Errors

An error will be generated if:

■ there are any value range errors
■ attempting to program non-applicable pins (configured as I, DC, or OFF)

Warnings and Autocorrections

Contextual edge mismatching generated by using optional edge parameters is autocorrected in the following manner:

If the value of a programmed edge exceeds the allowed range, the optional parameter is set to the maximum possible value. On the other hand, if a programmed value goes below the minimum, the optional parameter is autocorrected to the minimum possible value.

Examples

```
DFPN 1,,(TC)
CONF 0,STD,,(TC)
SCLK INT,10,
RCTM EDGE,5,,(TC)

RCTM WIDW,,6,(TC)
```

This would result in TE - LE < 2ns. So autocorrection sets LE = TE - 2ns.

```
RCTM? (TC)
```

would return RCTM WIDW,4,6,(TC) at this point.

RECD?

Test Results Read received data

RECD? returns the contents of the data acquisition memory in a binary format.

```
──(RECD?)──┤sample_start_cycle_no├──(,)──┤no_of_cycles├──(,)──┤pin_list├──►
```

`sample_start_cycle_#` is in the range 0 .. highest vector memory address
`V_LAST`
`no_of_cycles` is in the range 1 .. size of vector memory `V_MEM`

The transfer commands RECD sets up the system to transfer a large amount of
data via the HP-IB bus for the state list or timing diagram results displays. The
data consists of full 8-bit bytes in an unreadable (non ASCII) format and will be
transferred in one block for all the specified pins (received_data).

The '#' marks the start of the binary data.

`spec` defines the number of transferred bytes in the following manner: the first
<digit> specifies the number of valid number digits, immediate followed by the
digits defining the number of transferred bytes, followed by the binary data. No
white space is allowed after the '#' sign!

Up-loading will be performed in a by pin oriented manner (RECD), each byte
containing information of 4 cycles for one pin. The contents of the data stream
reflects the HW RAM organization and will not be altered by the transfer
commands.

The `sample_start_address` and the (`sample_start_address` + `#_of_cycles`)
will be rounded to the next byte boundaries for binary transfers.

This command works on byte boundaries

Errors

An error will be generated for the following reasons:

■ Parameter range errors
■ Command not allowed in real-time compare mode
■ No data available

RLCD, RLCD?

Calibration Receiver Levels

These commands are used to modify and read receiver level calibration values.
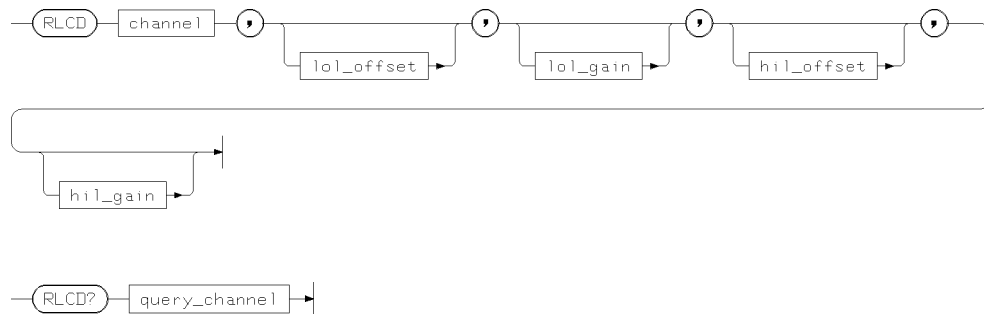
where `channel` is the sequential channel number defined with the DFPN command.

The values and implied units for the offsets and gains are shown below.

Level Parameter Value Ranges

		min	max	unit
Receiver	Offset	-500	+500	mV
	Gain	3440.40	3879.60	μV/LSB

Errors

An error will be generated for the following reasons:

■ Sequencer not idle
■ Gain value out of range
■ Trying to set or read receiver calibration data for an I channel

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

RLYC

Test Functions Relay Control

RLYC is a low level command used to connect DUT pins to tester resources.

`pin_mode`   sets up how a pin is connected to the tester. Can be:
      IDLE - pin is disconnected
      AC - pin is connected only to pin electronics
      DC - pin is connected to `dc_target` only
      BOTH - pin is connected to pin electronics and `dc_target`
      DCC - complementary output of HSWG is connected to
      `dc_target`
      DCB - both outputs of HSWG are connected to `dc_target`

`dc_target`   defines where a pin is connected for dc measurements. Can be:
      PMU1 connected to PMU1
      PMU2 connected to PMU2
      EXT connected to external input
      OFF disconnected from `dc_target`
      X change relay states except those for `dc_target`

`pin_list`   list of pins to be connected to tester resources.

Example

See Chapter 7 for an example of the use of this command.

RSWM

High Throughput Tests Set the relay switching method

RSWM allows you to specify the order in which the AC and DC relays are switched, when you perform a measurement using PTST? or DTST?.

$$\text{—(RSWM)—}\boxed{\text{mode}}\text{—|}$$
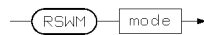
The `method` parameter can be either

- `bbm` - Break Before Make, i.e.
  1. all relays are opened,
  2. the relays needed for the measurement are closed.
- `mbb` - Make Before Break, i.e.
  1. the relays needed for the measurement are closed.
  2. the relays not needed for the measurement are opened.

RTCD, RTCD?

Calibration Receiver timing

These commands are used to modify and read receiver timing calibration values.
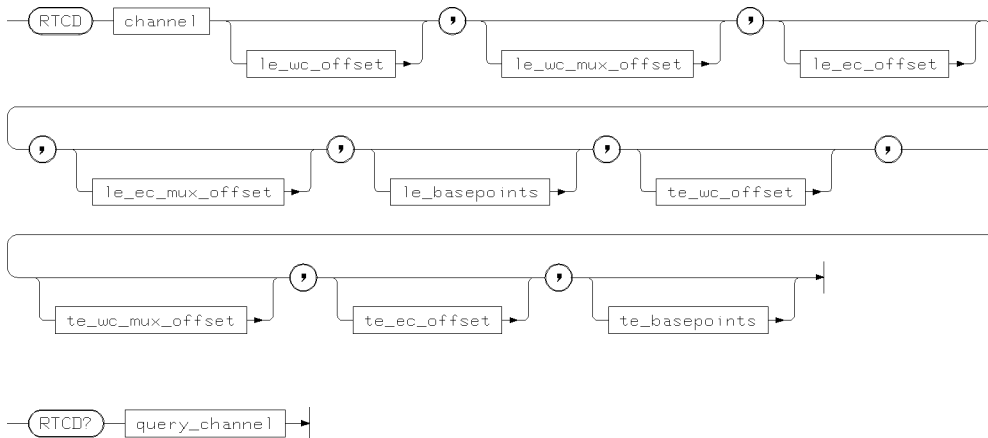
Note `le-` and `te-basepoints` may contain up to ten values.

where `channel` is the sequential channel number defined with the DFPN command.

The values for the offsets and gains are shown below.

Timing Parameter Value Ranges

			min	max	unit
Gain	Channel		29.41	39.22	ps/LSB
Offset	Leading Edge	Standard	0.0	+16.0	ns
		MUX Mode	-5.0	+16.0	ns
	Trailing Edge	Standard	-1.0	+16.0	ns
		MUX Mode	-6.0	+16.0	ns
	Tristate Edge		0.0	+16.0	ns

Errors

An error will be generated for the following reasons:

- Sequencer not idle
- Gain value out of range
- Trying to set or read receiver calibration data for an I channel

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

RTMC?

Calibration Receiver timing calibration

Performs receiver timing calibration of the specified pin using the given calibration probe channel (default is channel 1).

where

`channel`   is the channel number to be calibrated

`prb_chan`   is the probe channel to be used. Default is 1.

The returned values for this command are:

`error`	an integer in the range 0 .. 99999 indicating a calibration error code.Value 0 indicates no error.
`channel`	the channel being calibrated
`probe-channel- nr`	the cal probe channel used for the calibration
`period`	the period used in nanoseconds
`le-wc`	Leading edge to window compare - master channel
`le-wcslave`	Leading edge to window - slave channel
`le-ec`	Leading edge return to edge compare - master channel
`le-ecslave`	Leading edge return to edge compare - slave channel
`le-bpoints`	Leading edge fine delay basepoints (10 values)
`te-wc`	Trailing edge return to window compare - master channel

`te-wcslave`	Trailing edge return to window compare - slave channel
`te-ec`	Trailing edge return to edge compare - master channel
`te-bpoints`	Trailing edge fine delay basepoints (10 values)

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

RTUC?

Calibration Generate receiver timing user calibration
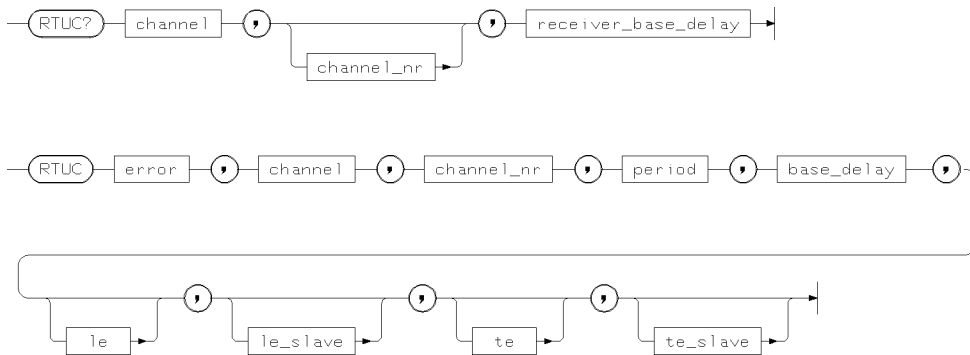
where

channel	is the channel number to be calibrated
channel-nr	is the probe channel to be used. Default is 1.
receiver-base-delay	

The returned values for this command are:

error	an integer in the range 0 .. 99999 indicating a calibration error code.Value 0 indicates no error.
channel	the channel being calibrated
probe-channel- nr	the cal probe channel used for the calibration
period	the period used in nanoseconds
base-delay	
le	Leading edge - master channel
le-slave	Leading edge - slave channel
te	Trailing edge - master channel
te-slave	Trailing edge - slave channel

Example

Refer to Chapter 10 and Servicing the HP 82000 for more information about
user calibration.

RTUD, RTUD?

Calibration Transfer receiver timing user calibration data
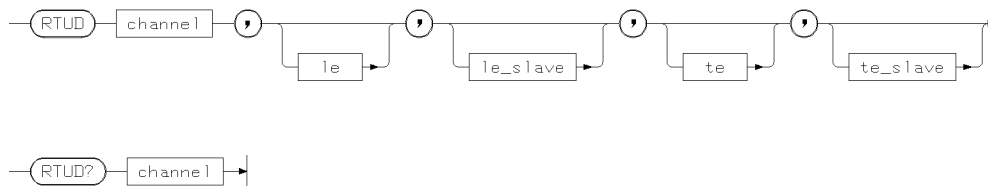
The parameters are:

channel	The name of the channel to be calibrated
le	The leading edge correction for the master channel
le_slave	The leading edge correction for the slave channel
te	The trailing edge correction for the master channel
te_slave	The trailing edge correction for the slave channel

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

SATR, SATR?

Attributes Set system attributes

```
―| SATR |―| round_trip_adjust |→|
```

```
―( SATR? )→|
```

`round-trip-adjust` is a fixed point value with the implied units nanoseconds.

See Chapter 11 for more details of this command.

Errors

An error will be generated if the parameter is out of range.

Example

`SATR?` returns the current settings of the system round-trip adjust. Files containing this data and pin attribute data (see PATR) should be written using a text editor and stored in the `pin_attributes` directory for the DUT, or entered in the Pin Attributed setup window.

SCCH?

Calibration Set calibration probe channel
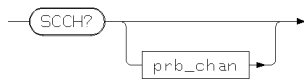
The command returns the pin name of the channel where the selected probe channel is connected.

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

SCLK, SCLK?

Timing Setup System Clock

This command sets the source, period and fraction (for external clock source only) of the system clock.
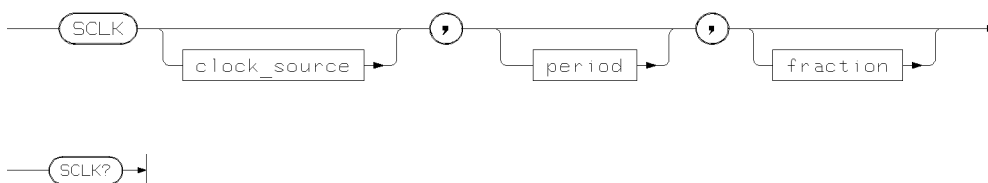
`clock_source` is either INT - for the internal clock, or EXT - for a signal connected to the external clock input.

`period` has the implicit unit ns and can be programmed as shown in the following table.

Table 12-4. System Clock Ranges and Resolutions

Range	Parameter	Resolution
$10\mu s$ .. 99.9ns	10 .. 99.9	100ps
$100\mu s$ .. 999ns	100 .. 999	1ns
$1\mu s$ .. $9.99\mu s$	1000 .. 9990	10ns
$10\mu s$ .. $99.9\mu s$	10000 .. 99900	100ns

Table 12-5.
System Clock Period Settings for External Source

Fraction	Range
F1T8	10ns .. 720ns
F1T4	10ns .. 1.44μs
F1T2	10ns .. 2.88μs
F1T1	10ns .. 5.76μs
F2T1	10ns .. 11.5μs
F4T1	10ns .. 23μs
F8T1	13.3ns .. 46μs

`fraction` - needs to be programmed when the external clock is used. It comes out of the formula:

$$\text{fraction} = \frac{\text{f(ext.clock)}}{\text{f(int.clock)}}$$

and only the quotients F1t8, F1t4, F1t2, F1t1, F2t1, F4t1 and F8t1 are allowed. These quotients correspond to ratios of 1:8, 1:4, 1:2, 1:1, 2:1, 4:1 and 8:1 respectively. This parameter is ignored if `clock_source` is internal.

The default (power on) settings for the system clock are `INT, 10,`.

Errors

Errors are generated if:

■ Period is out of range

SDRV, SDRV?

Vector Setup Mask and break vectors

Set and read driver channel pin mask and break vectors.
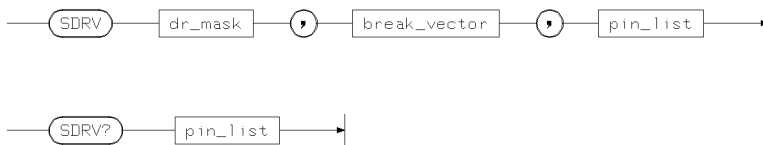
`dr_mask` is either ACT for active or TT for tristate mode.

`break_vec` modifies the pin's behavior during break cycles. and can be one of:

LO	channel goes to low
HI	channel goes to high
LZ	low combined with tri-state[1]
HZ	high combined with tri-state[1]
HOLD	channel repeats last vector

[1] The driver signal will be forced to tristate at the first trailing edge (for 100 mode pins with DNRZ format only).

This command affects the channel's behavior without modifying the contents of the vector memory.

Masking a driver pin forces the output driver to tristate mode (`dr_mask` = TT).

At power on, all driver channels are set to tristate, the break vector is set to LOW.

Autocorrection

If the configuration of an already defined pin is changed, the following autocorrection rules apply. Note that no warnings will be generated for the corrections noted below.

100 MHz to 200 MHz modes

Invalid break vector and expected data setups are changed from:

■ LZ to LO, HZ to HI
■ IM to LO, X to HI

Errors

Errors will be generated for:

- Bad positioned mnemonic
- Execution conflicts with current configuration
- LZ | HZ for 200 MHz pins
- IM | X for 200 MHz double frequency pins

Warnings

A warning will be generated by attempting to unmask a word mask pin.

Example

```
CONF I, STD, P, (A0)
CONF I, FD, P, (A1)
CONF O, STD, P, (D0)
CONF O, MUX, P, (D1)
SDRV ACT, HOLD, (A0, A1)
SREC ACT, LOW, (D0, D1)
```

In this example, the DUT input pins A0 and A1 are active. Both pins will repeat the last machine cycle vector in break mode. The DUT output pins D0 and D1 will be monitored. In data acquisition mode, the failure condition becomes false, if the output signal of both pins is LOW for an entire machine cycle.

SELF?

Diagnostics Self test

The firmware performs checks at power-on and during download and command execution. The SELF? query is used to read self test reports.

```
─( SELF? )─►|
```

This command is for HP service personnel.

SNCC, SNCC?

Timing Setup External clock synchronization
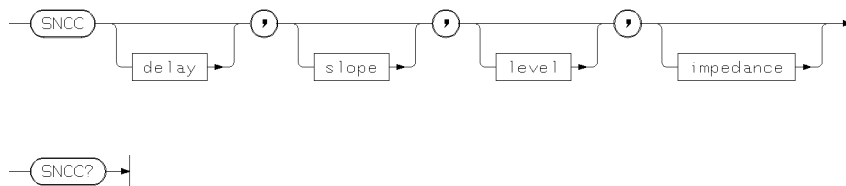
This command is used to setup external clock synchronization. The parameters are:

delay
: The delay between the external clock and the internal clock. It is a positive real number in the range 0 to 10^6 (0 ns to 1 ms) with the implied unit nanoseconds.

slope
: Sets the active edge to be used for synchronization. Can be either POS or NEG.

level
: Sets the threshold level for the external clock signal. This must be in the range ± 10000 millivolt.

impedance
: Input impedance to be used for external clock input. Can be either R10K for 10 kΩ or R50 for 50 Ω. Note that if 50 Ω is selected, the threshold levels are limited to ± 5000 mV.

Errors

An error will be generated for the following reasons:

■ Parameter out of range

SPOLL

HP-IB Serial Poll

Refer to the BASIC-UX Interfacing Techniques manual for more information on this command.

SQCL

Sequencer Control Clear label

```
 ─( SQCL )─►|
```

This command clears all the label definitions of previous SQSL command.

SQFA?

Sequencer Control First free address

This command returns the first free vector address (ffvad).

SQGB, SQGB?

Sequencer Control Global setup
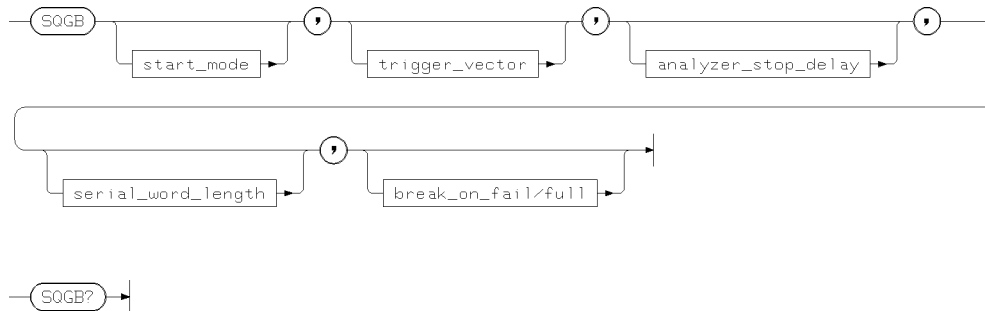
This command defines and reads sequencer global parameters. These parameters
are:

`start-mode`	How the sequencer reacts to a sequencer start command SQST. Can be either: ■ `ARM` - the sequencer goes into the armed state ■ `RUN` - the sequencer starts immediately
`trigger- vector`	specifies the address of the vector that activates the trigger output. Must be a valid vector address.
`analyzer-stop-` `delay`	specifies the number of cycles that will be output and analyzed after the analyze-stop flag was set in an SQPG command. This can be in the range 1 to 16777000.
`serial-word- length`	The number of channels that are to be serialized for serial vectors. Must be in the range 4 to total installed channels.
`break-on-` `fail/error`	If set, the tester goes into break if a compare error occurs or the analyze memory is full. If it is not set, no break occurs.

Example

Look at the sample vectors sequence files supplied with the system for examples
of the use of this command. Refer to the Advanced Testing manual for more
information about sequencer programming.

SQPG, SQPG?

Sequencer Control Program line
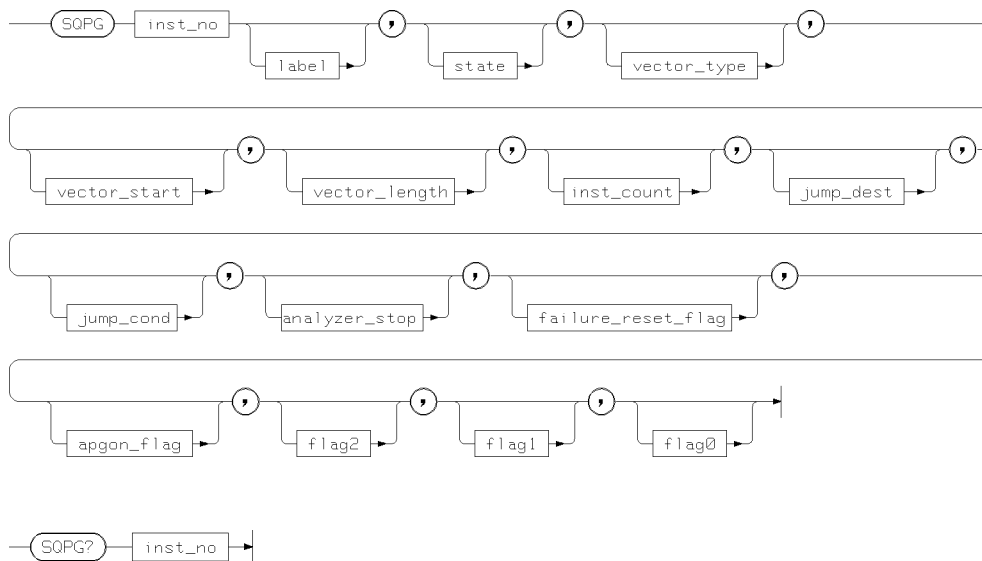
This command is used to set up sequences of vectors. The parameters are:

`inst-no`	Instruction number, this must be an integer in the range 0 .. 999.
`label`	The label assigned to the program line.
`state`	Sequencer state. Can be either:
	■ `CONT` - to continue
	■ `BRK` - go to break state
	■ `ARM` - got to arm state
`vector-type`	Specifies how vectors are output. Can be either:
	■ `LIN` - outputs vectors set by `vector-start` and `vector-length`.
	■ `REP` - outputs vector at `vector-start` number of times set by `vector-length`.
	■ `SCAN` - outputs `vector-length` vectors starting at `vector-start` in serial mode using number of

	channels set in `serial-word-length` of SQGB command.
`vector-start`	Start address for vectors. Entered relative to VESA.
`vector-length`	Number of vectors to be output. Minimum is 32 (except for jump source `EVER` or `NEV` then 8), maximum 1048575 (2^{20} -1).
`instruction-count`	Repetition factor for this instruction in range 1 .. 2048. Value -1 indicates infinite repetition or until jump condition occurs.
`jump-destination`	Specifies the instruction to process after the jump condition has occurred.
`jump-condition`	Specifies the jump condition. Can be either:

- `SS` - software signal. This is set by the SQSS command.
- `APG` - not used
- `FAIL` - Compare failed
- `XA` - External Input A active. This condition is set with SQXI.
- `XB` - External Input B active. This condition is set with SQXI.
- `EVER` - Used as a constant for unconditional jumps. If positive, jumps to value of `jump-destination`, if negative, to the next instruction.

	Additionally, all of these conditions can be negated by preceding them with `N`, for example, `NSS`, `NXA` and so on.
`analyzer- stop`	The first occurrence of this flag triggers the end of data analysis after `analyzer-stop` vectors.
`failure- reset-flag`	Resets the test failed condition before the instruction is executed. Prevents a jump on fail coming from a previous instruction affecting this line.
`apg-on-flag`	Not used
`flag-2`	Not used
`flag-1`	Not used
`flag-0`	Not used

SQPG? returns the program line for the given `inst_no`. The special value -1 lists all the instructions from `sequencer-start` to `first-free-sequencer-instruction - 1`.

SQPG, SQPG?

Errors

An error will be generated for the following reasons:

- Range errors
- Instruction number -1 and `vector-start` or 'sequencer-start'' not set to zero.

Example

See vector files supplied with system software and Advanced Testing manual.

SQSA, SQSA?

Sequencer Control Start address

This command sets and reads the absolute sequencer start instruction number. `inst_no` must be in the range 0 .. 999.

SQSL, SQSL?

AC tests Set/read sequencer start label

```
─(SQSL)──┤label├─►│
```

```
─(SQSL?)─►│
```
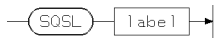
This command sets and reads the current sequencer start label.

Errors

An error will be generated for the following reasons:

■ Label not defined by a previous SQPG command.

Example

See vector files supplied with system software and Advanced Testing manual.

SQSS, SQSS?

Test Functions Sequencer software signal

```
─(SQSS)─[boolean]─►┤
```

```
─(SQSS?)─►┤
```

This command is used to set the sequencer software signal to allow conditional jumps in a vector sequence.

Errors

An error will be generated for the following reasons:

■ Value error

SQST, SQST?

AC tests Set/read current sequencer state

```
─(SQST)─┤sequencer_state├─►│
```

```
─(SQST?)─►│
```

This command sets and reads the current state of the sequencer. Allowed sequencer states are:

- OFF - sequencer and system clock are stopped. Only static signals are at the DUT pins.
- BRK - sequencer in break state. Output vectors are break vectors.
- RUN - sequencer is set to running. Whether it starts or goes into the armed state is determined by the SQGB command. If it starts, it starts at the defined start label.

Example

See Advanced Testing and Test Function manuals.

SQXI, SQXI?

Sequencer Control External Inputs
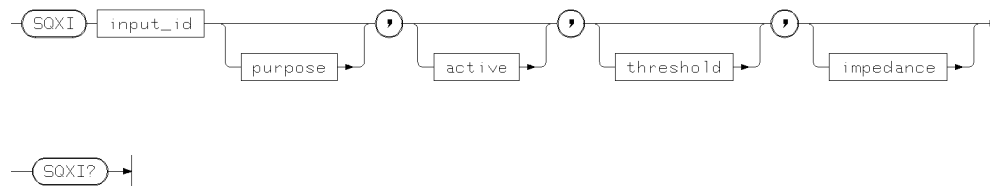
Sets and reads operating conditions for sequencer external inputs. The parameters are:

input-id	Can be XA or XB to set up A or B input respectively.
purpose	Sets purpose of input. Can be either:
	■ BRK - input is used to switch sequencer to break state. Cannot be used for conditional jumps.
	■ CONT - Changes sequencer from arm state to active (RUN) state.
	■ JUMP - input is used for jump instructions
active	Defines the active edge or level for the input. Can be either H or L for high or low levels, or R or F for rising and falling edges.
threshold	Threshold level in millivolts in range -10000 to 10000. If the value is greater than ±5000, a warning will be generated if impedance is set to L as higher voltages could result in damage to the tester. They will however be programmed.
impedance	Specifies the input impedance of the external input. Can be either L for 50 Ω or H for 10 kΩ.

SREC, SREC?

Vector Setup Read mask and expected data

These commands set and read the state of the mask and expected data vector in data acquisition mode.
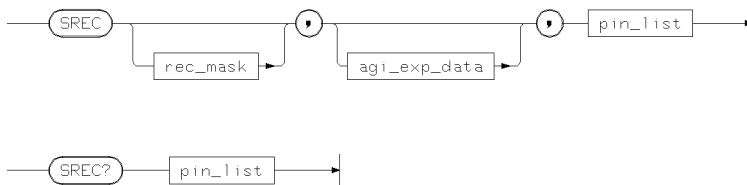
`rec_mask` can be ACT or X for active or masked respectively.

`aqi_exp_data` can be:

IM	for an expected intermediate level (tristate)
LO	for an expected low level
HI	for for an expected high level
X	for the don't care (masked) condition

This command affects the channel's behavior without modifying the contents of the vector memory.

Masking a receiver pin (`rec_mask` = X) changes the pin to a "Don't Care" pin in real time compare mode. The receiver pin mask has no effect in data acquisition mode.

The `aqi_exp_data` parameter of the SREC command is valid for data acquisition mode only. This parameter allows you to define a vector which will be compared against the incoming data in acquisition mode. Similar to the break vector, this vector will be valid for an entire machine cycle. For 200 MHz pins an error will be generated if the test fails in either of the user cycles.

At power on, all receiver channels are masked (set to Don't Care mode), the expected data vector (valid in AQI mode only) is set to Don't Care.

Autocorrection

If the configuration of a previously defined pin is changed, the following autocorrection rules apply. Note, that no warnings will be generated for the corrections noted below.

- Any mode to 200 MHz frequency doubling mode
  - □ Invalid expected data vector setups are changed from IM to LO and X to HI

Errors

Errors will be generated for:

- Badly positioned mnemonic
- Execution that conflicts with current configuration, for example, IM or X for 200 MHz pins.

Example

```
CONF I, STD, P, (AO)
CONF I, FD, P, (A1)
CONF O, STD, P, (DO)
CONF O, MUX, P, (D1)
SDRV ACT, HOLD, (AO, A1)
SREC ACT, LOW, (DO, D1)
```

In this example, the DUT input pins A0 and A1 are active. Both pins will repeat the last machine cycle vector in break mode. The DUT output pins D0 and D1 will be monitored. In data acquisition mode, the failure condition becomes false, if the output signal of both pins is LOW for an entire machine cycle.

STST?

High Throughput Tests Perform a functional test using a presaved parameter set (saved with PSAV).

```
—(STST)—| param_set_id |—| status |→
```

Returns the parameters:

```
—(STST?)—| param_set_id |→
```

where:

- `param_set_id` is the id number of the parameter set that has been used in the test;
- `status` is either `P` for a passed test, `F` for a failed test, or the measured value;

The STST? command is functionally the same as:

`LDHW` *id*
`FTST?`

Example

To perform a functional test with the parameter set id 24, you can enter the command string:

`STST? 24`

If the test result is a pass, the returned string is:

`STST 24,P`

SYNC, SYNC?

Timing Setup Clock synchronization

When selected, the internal clock will be synchronized to the external clock input defined by a previous SNCC command.

`mode` can be `ON` or `OFF` depending on whether the internal clock should be synchronized.

The SYNC? query returns the current state of `mode`.

Example

```
SYNC ON
```

This synchronizes the system clock with an external synchronizing signal.

TDHM?

AC tests Perform data hold time test
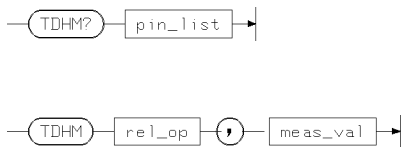
This command performs a data hold time test on the pins listed in `pin_list`. The parameters are:

rel-op Relational operator to describe the test result. Can be one of LT, LE, EQ, GE, or GT.

meas-val Returns the measured data hold time

Refer to Chapter 7 for more information on this command.

Errors

An error will be generated for the following reasons:

■ Pin not applicable

TEMP?

Calibration Read mainframe temperature

```
─( TEMP? )─▶│
```

This command returns the current temperature for each mainframe in Kelvin units. (Centigrade = Kelvin - 273) (Fahrenheit = 1.8 × (Kelvin - 273) + 32)

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

TFEN, TFEN?

Status Test function enable register

These commands define and read the Status Enable Register of the Hardware
Status Register.

TFEN ── tfstatuscontent ──▶

TFEN? ──▶

The response to the query commands is an integer value.

Refer to Chapter 9 for more details.

TFET?

Status Test function event register

This command reads the Event Register of the Test Function Status.

```
──( TFET? )──────┤
```

The response to the query commands is an integer value.

Refer to Chapter 9 for more details.

TFFL, TFFL?

Status Test function transition event

These commands set and read the falling transitions event for Test Function status.

```
    ──(TFFL)────┤tfstatuscontent├────→┤


    ──(TFFL?)────→┤
```

The response to the query commands is an integer value.

Refer to Chapter 9 for more details.

TFRS, TFRS?

Status Test function transition event

These commands set and read the rising transition for test function status.

The response to the query commands is an integer value.

Refer to Chapter 9 for more details.

TFST?

Status Test function status register

This command reads the Status Register of the Test Function Status.

The response to the query commands is an integer value in the range 0 .. -1000.

Refer to the Test Function Programming Manual for full details.

THLD?

AC tests Perform a hold time test
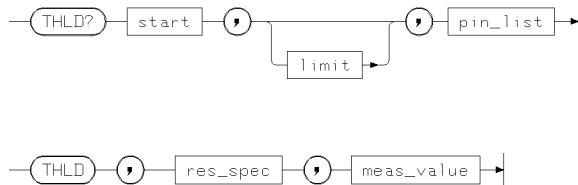
This command performs a hold time measurement and returns a pass/fail
indication. The parameters have the following meanings:

start Defines the start of a measurement window. The value has the implied units nanoseconds.

limit Sets the pass limit for the measurement. If this is omitted, the test returns only pass/fail information.

Errors

An error will be generated for the following reasons:

■ Non applicable pins

Example

Refer to Chapter 7 for examples of this command.

TPDM?

AC tests Perform a propagation delay measurement
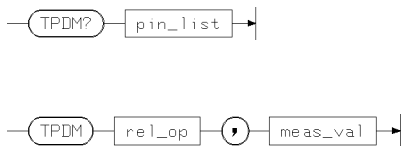
```
──(TPDM?)──┤ pin_list ├──▶
```

```
──(TPDM)──┤ rel_op ├──(,)──┤ meas_val ├──▶
```

This command performs a propagation delay measurement and returns a
pass/fail indication.

Errors

An error will be generated for the following reasons:

■ Non applicable pins

Example

See Chapter 7 for examples of this command.

TPWM?

AC tests Perform a pulse width/pause measurement.

```
──(TPWM?)──┤ le_delay ├──(,)──┤ te_delay ├──(,)──┤ pin_list ├──
```

```
──(TPWM)──┤ res_spec ├──(,)──┤ min_pulse_width ├──(,)──┤ res_spec ├──(,)──┤ min_pause_width ├──
```
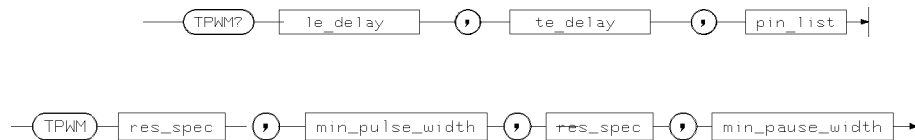
This command performs a minimum pulse width and maximum pause width measurement and returns a pass/fail indication.

The query parameters are:

le-delay	Real number with implied units nanoseconds to specify leading edge delay
te-delay	Real number with implied units nanoseconds to specify trailing edge delay
pinlist	Input pins to be tested

The command returns the following parameters:

res-spec	can be a relational operator of type rel-op (see DRLS) or P or F to indicate whether the test test or failed.
min_width	Minimum measured pulse width
min_pause	Maximum measured pause width

Errors

An error will be generated for the following reasons:

- Non applicable pins
- Exceeding le_delay or te_delay

Example

See Chapter 7 for examples of this command.

TSUP?

AC tests Perform a setup time test
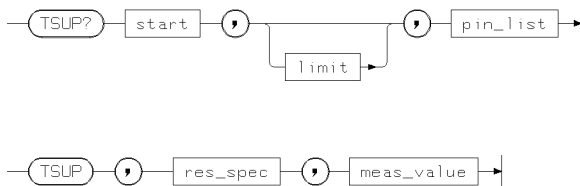
The query command measures the setup time for the pins in `pin-list`. The parameters are:

start A real number with implied units nanoseconds to define the start of the measurement window.

limit A real number with implied units nanoseconds to define the limit for the measurement. If this parameter is omitted, the test is made in pass/fail mode only.

The returned values are:

res-spec A relational operator to show the test results. Can be of type `rel-op` (see DRLS) or `P` or `F` to indicate the test result.

meas-val The measured value of setup time.

Errors

An error will be generated for the following reasons:

■ Non-applicable pins

Example

TTMC

Calibration Terminate calibration process

```
──(TTMC)──►|
```

Refer to Chapter 10 and Servicing the HP 82000 for more information about system calibration.

UDEF

Configuration Setup Undefine pins

This command undefines a pin that has been set up using the DFPN command.

All setups for the pin remain in memory and will come into effect again if the channel is defined by a subsequent DFPN command.

```
──(UDEF)──┤pin_list├──▶
```
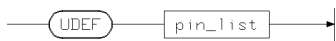
`pin_list` can consist of single pin names or the at symbol '@' which can be used to undefine all pins.

Example

    UDEF pin1

This command undefines the machine channel whose pin name was `pin1`.

UDPS

Configuration Setup Undefine DPS pins

UDPS — pin_list

This command undefines a power supply pin previously defined by a DFPS command. `pin-list` can consist of a single pin name or the at symbol "@" which can be used to undefine all DPS pins.

UTIN?

Utility Lines Read inputs

The UTIN? query reads the current state of the utility input port.

```
──( UTIN? )──▶│
```

results in the string UTIN? data_in being output where data_in is a binary
coded integer in the range 0000 to 1111.

UTOT, UTOT?

Utility Lines Read and set outputs

These commands set and read the state of the system utility output lines.
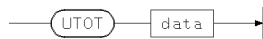
The UTOT command sets the utility output to the value specified by `utility_data`. This must be an integer in the range 0..127. The utility outputs will be set to a state equivalent to the binary equivalent of `utility_data`.

The UTOT? query retrieves the current state of the utility output port and respond with the message UTLI `utility_data` where `utility_data` is the decimal representation of the current setting of the utility output lines.

Example

```
UTOT 127
```

sets all seven utility lines to high

Errors

An error will be generated if the value is not in the range 0 to 255.

VALD?

Test Results Valid data in received data

VALD? returns the number of valid data in the error map / received data ram.

returns where `#_of_valid_data` is in the range 0 .. `V_MEM`.

This command is only applicable for the master mainframe.

Errors

An error will be generated for the following reasons:

■ Parameter range errors
■ No data available

VCLK?

Timing Setup Verify system clock

This command returns a flag to indicate whether auto-correction will take place
at the current system clock setting.

mode returns 0 or 1 to indicate whether autocorrection will occur

Errors

None

VEFA, VEFA?

Vector Setup First free vector address

VEFA reads or modifies the first free vector address. This address is updated after each vector data transfer to show the first unused location in the vector data memory.

`ffvad` must be in the range 0 .. the size of the vector memory `V_MEM`.

At power on the first free vector address is set to 0.

Errors

An error is generated if the specified address exceeds vector memory limits.

VESA, VESA?

Vector Setup Vector Start address

These commands are used to set and query the current setting of the Vector Start Address. This address is used as the base address for all subsequent vector data transfers.

```
────(VESA)───┤ start_address ├────▶│
```

```
────(VESA?)───▶│
```

`start_addr` must be in the range 0 .. highest vector memory address `V_LAST`.

A vector transfer starts at the byte location of the vector specified by the previously defined vector start address and the `rel_start_addr` parameter (which is regarded as an offset value, added to the vector start address).

The vector base address can be altered by the VESA command. It is recommended that you set the start address to a multiple of 4.

At power on the vector start address is set to 0.

Errors

Errors will be generated if the address specified exceeds vector memory limits.

*ESE, *ESE?

Error Handling Standard Event Enable Register

These commands set and read the Standard Event Enable Register.

The Standard Event Enable Register can be written with the *ESE command and can be read with the *ESE? query.

Example

See Chapter 9 for more information on this command.

*ESR?

Error Handling Event Status Register

The *ESR? query command reads the state of the Event Status Register.

Using this command causes the system to respond with an integer value which the current value of the Event Status Register.

The Standard Event Register can only be read with the *ESR? query. It can not be explicitly set. It will be cleared by the *ESR? query, the DCL HP-IB message, the *CLS command and at power on.

Refer to Table 9-2 for more information on this register.

*IDN?

Configuration Setup Identify firmware

The query *IDN? returns a firmware identification string.

This command returns the identification string for the hardware which includes the system type, and firmware type and revision.

Errors

There are no errors associated with this command.

Example

Sending `*IDN?` will return a string similar to:
`HEWLETT PACKARD,82000 D200,0,REV 1.0`
indicating a 200 MHz system with revision 1.0 firmware.

*OPC, *OPC?

Status Operation Complete message

*OPC generates an OPC message in the Standard Event register. *OPC?
responds with "1" when a command has been processed.

These commands provide a synchronization method without HP-IB holdoff. The
*OPC-command will generate an OPC event in the Standard Event Register.
The OPC-event is generated after the processing of all previous commands is
complete. By configuring status reporting appropriately, this can generate a SRQ
and can be detected by a serial poll (SPOLL).

The *OPC?-query will respond with a "1" after it and all previously sent
commands have been processed.

*OPT?

Configuration Setup Installed hardware

The *OPT? query reports the type of hardware installed in each slot of a
mainframe. The query response is a list of 18 fields separated by commas. Each
field contains an identification of the board installed in the corresponding slot or
a zero "0" if no board is installed.

```
────(*OPT?)────┤ mainframe ├──►
```

For each mainframe, the response to this command consists of 18 fields
containing either a zero (for an empty slot) or a `board_id` which consists of an
integer. Refer to Chapter 3 for more information on these codes.

Example

The command

`*OPT?` results in the following response being issued.

`4,17,52,52,52,80,0,0,0,0,0,0,0,0,0,0,0,0`

In this particular system, there is one mainframe, with three 200 MHz IO Boards,
1 PMU Board and twelve unused slots.

HP-IB Reset

```
──(*RST)──►
```

   `*RST`

Performs complete system reset, and loads all registers with default values.

*SRE, *SRE?

Status Service request enable register

These commands set and read the Service Request Enable Register. This allows you to make the system produce Service Requests only for specific error or status conditions.
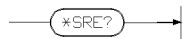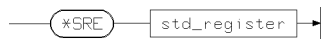
std_register is an integer value between 0 and 256.

Refer to Chapter 9 for details of the SRE register.

*STB?

HP-IB Read Status Byte

This command reads the Status Byte with MSS-bit via an ordinary query.

There are two ways to read the Status Byte. They differ in the format of the response and some side effects.

1. The controller can perform a SPOLL. This is usually done when it reacts to a Service Request. The firmware will send the status byte (without any coding). The SPOLL will also cause the Service Request to go to false for a subsequent SPOLL. This ensures that the RQS/MSS bit in the Status is only true for the first SPOLL after the Service Request. This enables the controller to decide whether this mainframe has generated the current Service Request. While a SPOLL clears the RQS/MSS bit, it does not change any other bits. The RQS/MSS will go to true again if a new reason for a Service Request is encountered.

2. The controller can send the *STB? query. The firmware will respond with the decimal coded value of the Status Byte. However, in contrast to SPOLL, MSS is always returned, as it is not affected by resetting RQS.

*WAI

Status Perform wait

The *WAI command is used to make the system wait for the completion of overlapping operations.

The *WAI command will wait until all pending overlapping commands are complete.

Because the firmware has no overlapping commands, *WAI degenerates to a NOP.

A

ERRS? Error Codes

This appendix contains a list of the error codes asociated with the ERRS? command. These codes are taken from the file /hp82000/pws/data/mc_fw_err.

Note that the variables @4 and @3 will be substituted by ERROR and WARNING respectively.

```
000      $0
001      @4 HPIB interupted state
002      @4 HPIB unterminated state
003      @4 HPIB deadlock state
004      @4 FW programming bug
005      @4 nothing to say
009      reserved
010      @4 decimal number too big
020      @4 invalid mnemonic received
021      @4 undefined data type
022      @4 invalid field seperator
023      @4 empty data field with a multielement message unit
024      @4 character data not allowed
025      @4 character data overflow
026      @4 invalid character data character
027      @4 invalid character data token
028      @4 improper string data type termination
029      @4 string data is not allowed for entered mnemonic
030      @4 block data is not allowed for entered mnemonic
031      @4 invalid block type
032      @4 improper block data type termination
033      @4 non-decimal data type not allowed for entered mnemonic
034      @4 invalid non-decimal data type character
035      @4 non-decimal data type data overflow
036      @4 improper non-decimal data type termination
037      @4 decimal data not allowed for entered mnemonic
038      @4 improper decimal data type format
039      @4 improper character whithin decimal data type field
```

040	@4 data type exponent overflow
041	@4 data type suffix is not allowed for enterd mnemonic
042	@4 data type received a invalid suffix
043	reserved
044	reserved
045	@4 query is not allowed for current mnemonic
046	@4 group execute trigger received in the midle of a message record
047	@4 improper non-decimal data type format
048	@4 improper expression data type terminaltion
049	@4 expression data type is not allowed for entered mnemonic
050	@4 reserved
051	@4 invalid character received in the defined block length filed
052	reserved
053	reserved
054	@4 invalid format
055	@4 invalid paramter
070	@4 bad positioned parameter in $0
071	@4 bad pinlist in $0
072	@4 can not apply $0 to $1
099	@4 bad syntax in message
100	@4 bad syntax in $0
101	@4 bad positioned mnemonic in $0
200	@4 parameter out of range in $0
210	@4 low level out of range in $0
211	@4 high level out of range in $0
212	@4 level swing out of range in $0
213	@4 level swing out of range for $1 in $0
220	@4 invalid format for $1 in $0
221	@4 leading edge out of range for $1 in $0
222	@4 trailing edge out of range for $1 in $0
223	@4 invalid combination of fraction and period in $0
224	@4 $0 can not be executed for internal clock source
225	@4 clock synchronization failed
230	@4 insufficient result data for $0
300	@4 $0 can not be executed in current tester state
310	@4 tester must be in idle-state for $0
320	@4 execution of $0 conflicts with current analyzer mode
400	@4 execution of $0 conflicts with current configuration
401	@4 execution of $0 conflicts with current configuration for $1
410	@4 channel to be defined in $0 is currently in use by $1
411	@4 resource requested in $0 for $1 is already in use

412	@4 resource requested in $0 for $1 is not installed
413	@4 $0 can not be executed while $1 is in use
500	@4 invalid SW loaded into instrument
501	@4 SW blocksize greater 64 Kbyte
502	@4 to much SW loaded into instrument
599	@4 fatal error occured in SW
600	@4 driver/receiver calibration aborted
601	@4 calibration allready in process
602	@4 calibration not in process
603	@4 ground connected to probe
604	@4 no ground connected to probe
605	@4 calibration probe not found
606	@4 calibration probe channel not found
607	@4 specified pin not found in mainframe
608	@4 user calibration not in process
609	@4 specified pin has no AC user setting
801	@4 bad instruction number in $0
802	@4 bad label in $0
803	@4 $0 ignored; too many labels defined
804	@4 insufficient hardware installed for $0
805	@4 channel error flags invalid
806	@4 address not readable / writeable
807	@4 no calibration probe connected
808	@4 execution of $0 conflicts with pin setup
900	@3 missing calibration data for $1
901	@3 $1 may exist in multiple mainframes
902	@3 stoped logging errors
903	@3 vector data truncated
904	@3 re-definition of label in $0
905	@3 threshold exceeds specifications in $0
910	@3 propably wrong levels for $1 in $0
912	@3 propably wrong delays for $1 in $0
913	@3 missing link to master mainframe
914	@3 can't unmask $1
950	@3 execution of $0 caused autocorrection
951	@3 execution of $0 caused autocorrection of $1

B

Test Function Error Codes

Status Messages

The following test function status will be generated at the end of test function execution.

1   Test function result is failed

0   Test Function result is passed

-1   Warning were generated during test function execution.

If errors were generated, the following codes will be generated:

MCD Errors

-2	Cannot execute firmware task successfully
-3	Internal answer buffer too small
-4	Internal task buffer too long
-5	Cannot open file
-6	Cannot read file
-7	No data in file
-8	Test file empty
-9	File is not an hp82000 file
-10	Wrong file type
-11	Wrong file revision
-12	File revision is newer
-13	file revision is older
-14	Invalid data in file
-15	Cannot write to file
-16	Cannot overwrite file
-17	File does not exist

BASIC Errors

-50	Filename in Exec_tf_file too long
-51	Parameter string in Exec_tf_param too long
-52	Badly dimensioned result array
-53	Buffer problem

Test Function Process Errors

-100	TF environment failure
-101	Bad tf keyword
-102	TF parameter check failure
-103	TF execution error
-104	Configuration change during tf execution
-105	Fatal error during tf execution

Break Key

-1000	The test function was terminated with the Break key.

Index

A

ac calibration commands, 10-3
acquisition stop, 8-4
adjusting the PMU, 10-21
adjusting timing settings, 11-3
autocorrection
    driver timing, 4-11
    receiver timing, 4-12

B

boards
    listing id codes, 2-2
break vector, 5-2

C

calibrating driver outputs and receiver
        thresholds, 10-3
calibrating the ADC, 10-2
calibrating the cal probe, 10-6
calibration, 10-1–23
    pmu, 10-20
calibration channel, 10-6
calibration data
    timing offsets, 11-3
calibration data transfer, 10-7
calibration period, 10-6
cal probe
    calibrating, 10-6
channel being calibrated, 10-6
clock output, 4-3
command synchronization, 9-12
comparator thresholds

corrections, 11-1
configuration setup commands, 2-1–12

D

data acquisition memory
    reading, 8-3
data formats
    bit alignment, 5-11
data transfer
    calibration data, 10-7
dc measurements
    DPS, 7-33
    PMU, 7-27
defining pin names, 2-4
defining power supply pins, 2-9
defining the system clock, 4-2
device power supply
    defining, 2-9
diagnostics
    commands, 10-23
diagnostics commands, 10-23–24
DPS
    maximum voltage, 3-8
    supported models, 2-10
driver
    setting format and timing, 4-6
    setting levels, 3-2, 3-4
    timing reference, 3-2
DUT
    output resistor, 11-1
DUT Board
    series resistor, 11-1, 11-2

setting, 4-2

T

tester states
   setting, 7-1
tester status registers, 9-6
test function
   status, 9-9
test result per pin, 8-3
threshold correction, 11-2
timing offset, 11-1
timing offsets, 11-3
timing setup
   corrections, 11-1
Timing Setup
   leading edge, 4-7
   trailing edge, 4-9

trailing edge
   timing, 4-9
transition filter, 9-7

U

utility lines, **7-43**

V

vector data size, 5-8
vector sequencer commands, 6-1–8
vector setup commands, 5-1–8
vector transfer
   format, 5-6
vector transfer commands, 5-1, 5-4

W

wildcard, 12-3