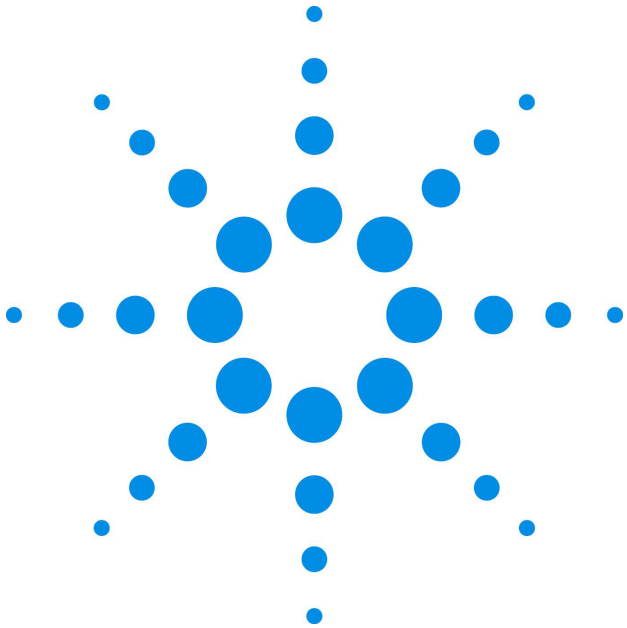


Agilent 53131A/132A
225 MHz Universal Counters

Programming Guide



Agilent Technologies

Manual Part Number 53131-90044
Printed in Malaysia

Agilent 53131A/132A Programming Guide

This guide describes how to program the Agilent 53131A/132A 225 MHz Universal Counter.

Agilent 53131A/132A 225 MHz Universal Counter

Copyright 1996-2006
Agilent Technologies, Inc.

All Rights Reserved.
Reproduction, adaptation, or translations without prior written permission is prohibited, except as allowed under the copyright laws.

Printed: October 2006

Printed in Malaysia

Manual part number
53131-90044

Certification and Warranty

Certification

Agilent Technologies certifies that this product met its published specification at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (formerly National Bureau of Standards), to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

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

Safety Considerations

General

This product and related documentation must be reviewed for familiarization with this safety markings and instructions before operation.

This product is a safety Class I instrument (provided with a protective earth terminal).

Before Applying Power

Verify that the product is set to match the available line voltage and the correct fuse is installed. Refer to instructions in Chapter 1 of the Manual.

Safety Earth Ground

An uninterruptible safety earth ground must be provided from the mains power source to the product input wiring terminals or supplied power cable.

Warning Symbols That May Be Used In This Book



Instruction manual symbol; the product will be marked with this symbol when it is necessary for the user to refer to the instruction manual.



Indicates hazardous voltages.

Safety Considerations (contd)



Indicates earth (ground) terminal.



or



Indicated terminal is connected to chassis when such connection is not apparent.



Indicates Alternating current.



Indicates Direct current.

WARNING

BODILY INJURY OR DEATH MAY RESULT FROM FAILURE TO HEED A WARNING. DO NOT PROCEED BEYOND A WARNING SIGN UNTIL THE INDICATED CONDITIONS ARE FULLY UNDERSTOOD AND MET.

CAUTION

Damage to equipment, or incorrect measurement data, may result from failure to heed a caution. Do not proceed beyond a *CAUTION* sign until the indicated conditions are fully understood and met.

Agilent Technologies, Inc.
900 South Taft Avenue
Loveland, Colorado 80537 U.S.A.

Warranty (contd)

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

Agilent warrants that its software and firmware designed by Agilent for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

Limitation of Warranty

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

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. AGILENT SPECIFICALLY DISCLAIMS THAT IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Exclusive Remedies

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

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales and Service Office.

Safety Considerations (contd)

WARNING

ANY INTERRUPTION OF THE PROTECTIVE GROUNDING CONDUCTOR (INSIDE OR OUTSIDE THE INSTRUMENT) OR DISCONNECTING THE PROTECTIVE EARTH TERMINAL WILL CAUSE A POTENTIAL SHOCK HAZARD THAT COULD RESULT IN PERSONAL INJURY. (GROUNDING ONE CONDUCTOR OF A TWO CONDUCTOR OUTLET IS NOT SUFFICIENT PROTECTION.)

Whenever it is likely that the protection has been impaired, the instrument must be made inoperative and be secured against any unintended operation.

If this instrument is to be energized via an autotransformer (for voltage reduction) make sure the common terminal is connected to the earthed pole terminal (neutral) of the power source.

Instructions for adjustments while covers are removed and for servicing are for use by service-trained personnel only. To avoid dangerous electric shock, do not perform such adjustments or servicing unless qualified to do so.

For continued protection against fire, replace the line fuse(s) only with 250V fuse(s) of the same current rating and type (for example, normal blow, time delay). Do not use repaired fuses or short circuited fuseholders.

Acoustic Noise Emissions

LpA 27 dB at operator position, at normal operation, tested per EN 27779. All data are the results from type test.

Geräuschemission

LpA 27 dB am Arbeits platz, normaler Betrieb, geprüft nach EN 27779. Die Angagen beruhen auf Ergebnissen von Typenprüfungen.

Contents

1 Before You Start ...

Introduction 1-2

Differences Between Prior and Current Revisions of the Agilent 53131A/132A 1-3

Agilent 53131A Containing Firmware Revisions (3317, 3335, or 3402) 1-3

Agilent 53132A Time Interval Delay Arming 1-5

Getting Started 1-6

How to Use This Guide 1-6

New Users 1-6

What You Should Understand 1-6

Learning to Program the Counter 1-7

Experienced Programmers 1-7

Applications 1-8

Programming Guide Contents 1-9

Assumptions 1-9

Related Documentation 1-10

2 Command Summary

Introduction 2-2

Chapter Summary 2-2

Front Panel to SCPI Command Maps 2-3

Some SCPI Syntax Conventions 2-3

Input Channels Conditioning Keys to SCPI Command Map 2-4

Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map 2-6

MEASURE Keys to SCPI Command Map 2-8

Gate & ExtArm Key to SCPI Command Map 2-10

Gate & ExtArm Key to SCPI Command Map — For Agilent 53131A (and Agilent 53132A With S/N Prefix Below 3646) 2-13

LIMITS and MATH Keys to SCPI Command Map 2-16

Calibration Menu to SCPI Command Map 2-18

Agilent 53131A/132A Command Summary 2-20

SCPI Conformance Information 2-20

IEEE 488.2 Common Commands 2-21
Agilent 53131A/132A SCPI Subsystem Commands 2-24
Std/New Column 2-24
Parameter Form Column 2-24
***RST Response 2-40**

3 Programming Your Universal Counter for Remote Operation

Introduction 3-2
Chapter Summary 3-2
Where to Find Some Specific Information 3-2
Where to Find BASIC Programming Examples 3-3
Where to Find QuickBASIC Programming Examples 3-3
Where to Find Turbo C Programming Examples 3-3
Configuring the GPIB 3-4
To Set the GPIB Mode and Address 3-4
To Connect the Counter to a Computer 3-6
Remote/Local Operation 3-6
Overview of Command Types and Formats 3-7
Common Command Format 3-7
SCPI Command and Query Format 3-7
Elements of SCPI Commands 3-8
Subsystem Command Syntax 3-8
Common Command Syntax 3-8
Abbreviated Commands 3-9
Keyword Separator 3-9
Optional Keyword 3-9
Implied Channel (Optional Numeric Keyword Suffix) 3-10
Parameter Types 3-11
Parameter Separator 3-12
Query Parameters 3-12
Suffixes 3-12
Suffix Elements 3-12
Suffix Multipliers 3-13
Command Terminator 3-13

Using Multiple Commands 3-14

Program Messages 3-14

Program Message Syntax 3-14

Overview of Response Message Formats 3-16

Response Messages 3-16

Response Message Syntax 3-16

Response Message Data Types 3-17

Status Reporting 3-19

Status Byte Register and Service Request Enable Register 3-21

Status Byte Register 3-21

Service Request Enable Register 3-23

Standard Event Status Register Group 3-24

Standard Event Status Register 3-24

Standard Event Status Enable Register 3-26

Operation Status Register Group and Questionable Data/Signal Status Register Group 3-27

Condition Register 3-28

Transition Filter 3-28

Event Register 3-29

Event Enable Register 3-29

Operation Status Register Group 3-30

Questionable Data/Signal Status Register Group 3-32

Command Settings for Optimizing Throughput 3-35

Commands to Set Counter for Optimal Throughput 3-35

Typical Optimizing Throughput Results for Different Computers 3-37

How to Program the Counter for Status Reporting 3-38

Determining the Condition of the Counter 3-38

Resetting the Counter and Clearing the GPIB Interface—Example 1 3-38

Using the Standard Event Status Register to Trap an Incorrect GPIB command—Example 2 3-39

Event Status Register 3-39

Using the Questionable Data/Signal Status Register to Alert the Computer When Automatic Interpolator Calibration is Disabled—Example 3 3-39

Questionable Data Status Register 3-40

Using the Operation Status Register to Alert the Computer When Measuring has Completed— Example 4 3-40
Operation Status Register 3-40
How to Program the Counter to Display Results 3-43
Configuring the Counter's Display 3-43
Commands for Displaying Non-Scaled/Offset Results 3-43
Commands for Displaying Scaled/Offset Results 3-44
Commands for Displaying the Limit Graph 3-44
Commands for Displaying Statistics Results 3-44
Commands for Enabling and Disabling the Display 3-45
How to Program the Counter to Synchronize Measurements 3-46
Synchronizing Measurement Completion 3-46
Resetting the Counter and Clearing the GPIB Interface 3-46
Using the *WAI Command 3-46
Using the *OPC? Command 3-47
Using the *OPC Command to Assert SRQ 3-48
How to Program the Counter for Math/Limit Operations 3-49
Updating Math and Limit Results Over GPIB 3-49
Using the Scale and Offset Over GPIB 3-50
How to Program the Counter to Define Macros 3-52
Writing SCPI Programs 3-55
Programming Examples 3-58
Using BASIC 3-58
To Send a Double-Quoted String 3-58
To Send a Single-Quoted String 3-58
Using QuickBASIC 3-59
Using Turbo C 3-59
List of the Programming Examples 3-59
Easiest Way to Make a Measurement (BASIC) 3-60
To Make a Frequency Measurement (BASIC) 3-62
To Perform Limit Testing (BASIC) 3-63
To Measure the Statistics of 50 Measurements (BASIC) 3-64

To Use Limits to Filter Data Before Measuring Stats (BASIC) 3-66

To Read and Store Calibration Information (BASIC) 3-68

To Perform a Time Interval Calibration (BASIC) 3-69

To Optimize Throughput (BASIC) 3-73

To Use Macros (BASIC) 3-75

To Make a Frequency Measurement (QuickBASIC) 3-77

To Perform Limit Testing (QuickBASIC) 3-78

To Measure the Statistics of 50 Measurements (QuickBASIC) 3-80

To Use Limits to Filter Data Before Measuring Stats (QuickBASIC) 3-82

To Read and Store Calibration Data (QuickBASIC) 3-85

To Optimize Throughput (QuickBASIC) 3-86

To Use Macros (QuickBASIC) 3-88

To Make a Frequency Measurement (Turbo C) 3-91

To Use Limits to Filter Data Before Measuring Statistics (Turbo C) 3-93

To Optimize Throughput (Turbo C) 3-96

4 Command Reference

Introduction 4-2

:ABORt Command 4-4

:CALCulate Subsystems 4-5

:CALCulate[1] Subsystem 4-7

:CALCulate[1]:MATH Subtree 4-9

:CALCulate2 Subsystem 4-11

:CALCulate2:LIMit Subtree 4-12

:CALCulate3 Subsystem 4-19

:CALCulate3:AVERage Subtree 4-19

:CALCulate3:LFILter Subtree 4-23

:CALibration Subsystem 4-26

:CALibration:SECurity Subtree 4-28

:CONFigure Subsystem 4-30

Device Clear 4-31

:DIAGnostic Subsystem 4-32

:DISPlay Subsystem 4-37

:FETCh Subsystem 4-40

:FORMat Subsystem 4-41

Group Execute Trigger (GET) 4-42

:HCOPy Subsystem 4-43

:INITiate Subsystem 4-44

:INPut[1|2] Subsystem 4-48

:INPut3 Subsystem 4-50

:MEASure Subsystem 4-51

Measurement Instructions (:CONFigure, :FETCh, :MEASure, :READ) 4-52

Using :MEAsure 4-75

Using :CONFigure with :READ? 4-76

Using :CONFigure with :INITiate and :FETCh? 4-76

Firmware Revision Work-Around Commands 4-77

:MEMory Subsystem 4-79

[:SENSe] Subsystem 4-80

[:SENSe]:EVENt[1|2] Subtree 4-80

[:SENSe]:EVENt3 Subtree 4-84
[:SENSe]:FREQuency Subtree 4-85
[:SENSe]:FREQuency:ARM Subtree 85
[:SENSe]:PHASe Subtree 4-91
[:SENSe]:PHASe:ARM Subtree 4-91
[:SENSe]:ROSCillator Subtree 4-92
[:SENSe]:TINTerval Subtree (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646) 4-95
[:SENSe]:TINTerval:ARM Subtree (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646) 4-95
[:SENSe]:TINTerval Subtree (Agilent 53132A With S/N Prefix 3646 and Above) 4-98
[:SENSe]:TINTerval:ARM:ESTART and :ESTOP Subtrees (Agilent 53132A With S/N Prefix 3646 and Above) 4-98
[:SENSe]:TOTalize Subtree 4-104
[:SENSe]:TOTalize:ARM Subtree 4-104

:STATus Subsystem 4-107

:STATus:OPERation Subtree 4-107
:STATus:QUEStionable Subtree 4-110

:SYSTem Subsystem 4-114

:SYSTem:COMMunicate Subtree 4-114

:TRACe Subsystem 4-119

:TRIGger Subsystem 4-121

***CAL? (Calibration Query) 4-122**

***CLS (Clear Status Command) 4-123**

***DDT <arbitrary block> (Define Device Trigger Command) 4-124**

***DMC <string>, <arbitrary block> (Define Macro Command) 4-125**

***EMC <NRf> (Enable Macro Command) 4-126**

***EMC? (Enable Macro Query) 4-126**

***ESE <NRf> (Standard Event Status Enable Command) 4-127**

***ESE? (Standard Event Status Enable Query) 4-127**

***ESR? (Event Status Register Query) 4-128**

***GMC? <string> (Get Macro Contents Query) 4-129**

***IDN? (Identification Query) 4-130**
***LMC? (Learn Macro Query) 4-131**
***OPC (Operation Complete Command) 4-132**
***OPC? (Operation Complete Query) 4-133**
***OPT? (Option Identification Query) 4-134**
***PMC (Purge Macro Command) 4-135**
***RCL <NRf> (Recall Command) 4-136**
***RST (Reset Command) 4-137**
***SAV <NRf> (Save Command) 4-138**
***SRE <NRf> (Service Request Enable Command) 4-139**
***SRE? (Service Request Enable Query) 4-139**
***STB? (Status Byte Query) 4-140**
***TRG (Trigger Command) 4-141**
***TST? (Self-Test Query) 4-142**
***WAI (Wait-to-Continue Command) 4-143**

5 Errors

Introduction 5-2
Displaying Errors 5-2
Reading an Error 5-2
Error Queue 5-3
Error Types 5-4
No Error 5-4
Command Error 5-4
Execution Error 5-5
Device- or Counter-Specific Error 5-5
Query Error 5-6

1

Before You Start ...

Introduction

This programming guide contains programming information for the Agilent 53131A/132A Universal Counter.

This guide assumes you are familiar with the front-panel operation of the Counter. See the *Agilent 53131A/132A Operating Guide* for detailed information about front-panel operation. You should use this programming guide together with the operating guide. Knowing how to control the Counter from the front panel and understanding the measurements you wish to perform makes the programming task much easier. The operating guide provides explanations and task procedures for all of the Counter's measurement functions, and contains the specifications for the Counter.

By sending Standard Commands for Programmable Instruments (SCPI) commands, all of the Counter's front-panel functions can be remotely operated via the General Purpose Interface Bus (GPIB),
as well as the additional throughput optimizing function not available from the front panel.

This Counter programming commands conform to the *Standard Commands for Programmable Instruments (SCPI) Standard Version 1992.0*. The SCPI standard does not completely redefine how to program instruments over the General Purpose Interface Bus (GPIB). However, it does standardize the structure and content of an instrument's command set to reflect the best programming practices developed by people using GPIB. It also establishes standard command mnemonics for similar functions in all of the instruments that conform to the SCPI standard.

If you have programmed any Agilent instruments that have been released over the last few years, you will have seen a general trend toward the techniques specified in the SCPI standard. For example, several instruments are already using a hierarchy of commands that is similar to the command structure defined by the SCPI standard.

Differences Between Prior and Current Revisions of the Agilent 53131A/132A

If you have an Agilent 53131A containing one of the prior firmware revisions (3317, 3335, or 3402), read the subsection below titled "Agilent 53131A Containing Firmware Revisions (3317, 3335, or 3402)" to get an overview of the differences between the earlier firmware revisions and current firmware revision.

If you have an Agilent 53132A with a serial number prefix below 3646, read the subsection titled "Agilent 53132A Time Interval Delay Arming" on page 1-5.

NOTE

Note that throughout the guide, differences between the earlier and current firmware revisions are noted where applicable.

Agilent 53131A Containing Firmware Revisions (3317, 3335, or 3402)

There are four main areas that differ:

- Calibrations
- Measurements
- Statistics
- GPIB Commands

Calibrations

If your Counter contains other than the current firmware revision, the following calibration features are different:

- The calibration functions are in the Utility menu instead of the Calibration menu, which is accessed by pressing and holding the front-panel **Utility** key and then cycling **POWER** key.
- Calibrations are not protected by a security code.
- A calibration count does not exist to aid in monitoring the number of calibrations performed.
- A more accurate Time Interval calibration (FINE TI) is not available.

See the section titled "Using the Calibration Menu" in Chapter 2 of the *Agilent 53131A/132A Operating Guide* for details.

Differences Between Prior and Current Revisions of the Agilent 53131A/132A

Measurements

If your Counter contains other than the current firmware revision, the following measurement capabilities are different:

- Ratio channel selections Ratio 2 to 1 and Ratio 3 to 1 (for those counters equipped with Channel 3) are not available.
- Ratio "AUTO-armed" does not automatically extends gate to capture sufficient edges.

  If Channel 1 input frequency is less than approximately 10 Hz, the Ratio gate time is not extended to capture sufficient Channel 1 edges to produce a valid measurement. Default gate time is 100 msec, which is not long enough to capture two edges on a low-frequency signal. The user is required to extend the gate by switching to TIME arming, and selecting a gate time appropriately long.

- Sensitivity for firmware revision below does not have adjusted controls to LO and MED sensitivity.

  In some Counters that contained firmware revision 3317, LO sensitivity fails to correctly count very high frequency signals.

Statistics

If your Counter contains other than the current firmware revisions, single-shot statistics are not available using the **ON SINGLE:** menu item found in the Statistics menu (use **Stats** key).

GPIB Commands

[:SENSe]:EVENt[1|2}:HYSTeresis:RELative

If your Counter contains firmware revisions 3402 and below, the input hysteresis command and query does not operate in the conventional way. That is, [:SENSe]:EVENt[1|2]:HYSTeresis:RELative sets high sensitivity when the parameter is MINimum or 0 percent, and sets low sensitivity when the parameter is MAXimum or 100 percent.

In the prior firmware revisions (3317, 3335, or 3402), MINimum or 0 percent corresponded to low sensitivity, and MAXimum or 100 percent corresponded to high sensitivity.

:CONFigure:TOTalize:TIMed
:CONFigure:TOTalize:CONTinuous
:MEASure:TOTalize:TIMed?

If your Counter contains firmware revisions 3402 and below, the Totalize Measurement Instruction commands (shown above) are not available to disable auto-trigger.

In the firmware revisions 3402 and below, these commands enabled auto-trigger at the 50% level.

Agilent 53132A Time Interval Delay Arming

Agilent 53131A and Agilent 53132A Counters with a serial number prefix below 3646 are identical in their TI arming modes. Both only offer Time Interval Delay, where the STOP trigger of a time interval measurement can be delayed by a user-specified time.

Getting Started

Before attempting to program the Counter, take some time to familiarize yourself with the content of this guide. The remainder of this chapter contains the following information:

- An explanation of how you should use the programming guide based on your experience programming instruments and your testing requirements.
- A description of the guide contents.
- A statement of assumptions that are made in the guide.
- A list of related documentation.

How to Use This Guide

How you use this guide depends upon how much you already know about programming instruments and how complex your measurement requirements are. Let's start by establishing your programming background, and then discuss the type of measurements you want to perform.

New Users

What You Should Understand

As a new user, you should understand that you must have some understanding of a high-level language such as Pascal, BASIC, C, or FORTRAN before you can use the command set defined in this guide to control the Counter. (In Chapter 3, "Programming Your Universal Counter for Remote Operation," there are programming examples provided in BASIC, Microsoft® QuickBASIC, and Borland® Turbo C.) However, whatever language you use, command strings that control the Counter remain the same.

Learning to Program the Counter

To learn how to program the Counter, perform the following:

● Scan the summary tables in Chapter 2, “Command Summary,” to get a feeling for the number and structure of commands available to you.

● Read and study map drawings in the section titled “Front Panel to SCPI Command Maps” in Chapter 2.

● Read Chapter 3, “Programming Your Universal Counter for Remote Operation,” for an overview of the SCPI concepts as they relate to the Agilent 53131A/132A Universal Counter. Look at the flowcharts, which illustrate some of the decisions you must make when programming the Counter.

● Read the section at the end of Chapter 3 titled “Programming Examples for Making Common Measurements,” which provides programming examples.

● Modify some of the programming examples to select specific measurement functions. If the programs work, consider yourself an experienced programmer and use Chapter 4, “Command Reference,” as a reference for detailed information of all the Counter’s SCPI commands.

Experienced Programmers

If you have programmed other GPIB instruments, you will probably be familiar with many of the concepts and techniques discussed in this guide. Also, you will find that using the SCPI commands is very similar to using the older GPIB commands. The main difference is the hierarchy of the subsystem commands. (However, this type of structure has been previously used on other instruments.)

Because the SCPI command set and some of the status reporting techniques are new, you may want to use the following sequence to learn the Counter programming requirements:

● Look over the steps for a new user and perform any that you think are applicable to your current level of knowledge. In particular, look at the measurement techniques and examples provide in Chapter 3, “Programming Your Universal Counter for Remote Operation.”

● Review the summary tables in Chapter 2, “Command Summary.” If this chapter contains sufficient information to get you started, write some programs to explore the Counter’s capabilities. If you need additional information on any command, refer to the applicable command description in Chapter 4, “Command Reference.”

● Review the remaining information in this guide to determine what is applicable to your programming requirements.

If you need more information than is contained in this guide, see the section in this chapter titled “Related Documentation.”

Applications

After you have read the appropriate information and written some measurement programs, you may want to expand the scope of your applications. The following two techniques are explained in detail:

● If you are going to write interrupt-driven programs (or if you just want to determine the status of the Counter), read the section titled “Status Reporting” in Chapter 3.

● If you are going to write programs to transfer data between the Counter and an external computer, read the sections titled “Overview of Response Message Formats,” and “Command Settings for Optimizing Throughput” in Chapter 3.

Programming Guide Contents

The following information is contained in this guide:

● **Table of Contents**

● Chapter 1 (this chapter) ,"**Before You Start**," is a preface that introduces you to the programming guide.

● Chapter 2, "**Command Summary**," is a quick reference that summarizes the Counter's programming commands. It provides you with front-panel to SCPI command maps, SCPI conformance information, and command summary tables.

● Chapter 3, "**Programming Your Universal Counter for Remote Operation,**" describes how to set up the Counter for remote operation, briefly explains the SCPI elements and formats, describes status reporting, describes how to write programs,
and provides programming examples for each of the main tasks that you will want your Counter to perform.

● Chapter 4, "**Command Reference**," is a dictionary that describes the SCPI subsystems and IEEE 488.2 Common commands.

● Chapter 5, "**Errors**," lists all the error messages the Counter can generate and what caused the error.

● **Index**

Assumptions

This guide assumes the Counter is correctly installed and interfaced to an external computer. If it is not, see IEEE GPIB Interconnection information in *Agilent Technologies, Tutorial Description of the General Purpose Interface Bus, 1987*. (See the following section in this chapter titled "Related Documentation" for ordering information.)

As previously mentioned, this guide also assumes you are familiar with the front-panel operation of the Counter. See the Agilent *53131A/132A Operating Guide* for detailed information about front-panel operation. Knowing how to control the Counter from the front panel and understanding the measurements you wish to perform makes the programming task much easier.

Related Documentation

This section contains a list of documentation related to the use of the Counter. Additional information that you may find useful can be found in the following publications:

1. Agilent 53131A/132A Operating Guide (Agilent Part Number 53131-90043)
2. Beginner's Guide to SCPI (Agilent Part Number H2325-90002, July 1990 Edition).
3. Beginner's Guide to SCPI, Barry Eppler (Hewlett-Packard Press, Addison-Wesley Publishing Co. 1991).
4. Standard Commands for Programmable Instruments (SCPI), Version 1992.0.

   This standard is a guide for the selection of messages to be included in programmable instrumentation. It is primarily intended for instrument firmware engineers. However, you may find it useful if you are programming more than one instrument that claims conformance to the SCPI standard.
   You can verify the use of standard SCPI commands in different instruments.

   To obtain a copy of this standard, contact:

   SCPI Consortium
   8380 Hercules, Suite P3
   La Mesa, CA 91942
   Phone: (619) 697-8790
   FAX: (619) 697-5955

5. The International Institute of Electrical Engineers and Electronic Engineers, IEEE Standard 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.

   This standard defines the technical details required to design and build an GPIB (IEEE 488.1) interface. This standard contains electrical specification and information on protocol that is beyond the need of most programmers. However, it can be useful to clarify formal definitions of certain terms used in related documents.

To obtain a copy of this standard, write to:

The Institute of Electrical and Electronic Engineers Inc.
345 East 47th Street
New York, NY 10017 USA

6. The International Institute of Electrical Engineers and Electronic Engineers, IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols, and Common Commands for Use with ANSI/IEEE Std 488.1-1987 Programmable Instrumentation.

   This standard defines the underlying message formats and data types used in SCPI. It is intended more for firmware engineers than for instrument users/programmers. However, it can be useful if you need to know the precise definition of specific message formats, data type, or common commands.

   To obtain a copy of this standard, write to:

   The Institute of Electrical and Electronic Engineers Inc.
   345 East 47th Street
   New York, NY 10017 USA

7. Agilent Technologies, Inc,
   BASIC 5.0/5.1 Interfacing Techniques Vol 2.,
   Specific Interfaces, 1987.

   This BASIC manual contains a good non-technical description of the GPIB (IEEE 488.1) interface in Chapter 12, “The GPIB Interface.” Subsequent revisions of BASIC may use a slightly different title for this manual or chapter.
   This manual is the best reference on I/O for BASIC programmers.

   To obtain a copy of this manual, contact your nearest Agilent Technologies Sales office.

8. Agilent Technologies, Inc., Tutorial Description of the General Purpose Interface Bus, 1987.

   To obtain a copy of this manual, contact your nearest Agilent Technologies Sales office.

2

Command Summary

A Quick Reference

Introduction

This chapter is a quick reference that summarizes the Counter's programming commands.

Chapter Summary

- Front Panel to SCPI Command Maps[1] pg. 2-3
  - Some SCPI Syntax Conventions pg. 2-3
  - Input Channels Conditioning Keys to SCPI Command Map pg. 2-4
  - Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map pg. 2-6
  - MEASURE Keys to SCPI Command Map pg. 2-8
  - Gate & ExtArm Key to SCPI Command Map pg. 2-10
  - Gate & ExtArm Key to SCPI Command Map For Agilent 53131A (and Agilent 53132A With S/N Prefix Below 3646) Time Interval Arming Commands pg. 2-13
  - Gate & ExtArm Key to SCPI Command Map For Agilent 53132A (With S/N Prefix 3646 and Above) Time Interval Arming Commands pg. 2-14
  - LIMITS and MATH Keys to SCPI Command Map pg. 2-16
  - Calibration Menu to SCPI Command Map pg. 2-18
- Agilent 53131A/132A Command Summary[2] pg. 2-20
  - SCPI Conformance Information pg. 2-20
  - IEEE 488.2 Common Commands pg. 2-21
  - Agilent 53131A/132A SCPI Subsystem Commands pg. 2-24
- *RST Response[3] pg. 2-40

[1]The section titled "Front Panel to SCPI Command Maps," provides maps that show the front-panel keys and their corresponding (or related) SCPI commands.

[2]The section titled "Agilent 53131A/132A Command Summary," lists the IEEE 488.2 Common and the SCPI Subsystem commands in tables 2-1 and 2-2, respectively.

[3]The section titled *RST Response, lists the states of all of the commands that are affected by the *RST command in Table 2-3. This section also lists commands that are unaffected by *RST in Table 2-4.

Front Panel to SCPI Command Maps

Figures 2-1 through 2-6 provide maps that show the one-to-one relationship of the front-panel keys and the SCPI commands. These maps should help with identifying commands if you are already familiar with the front panel.

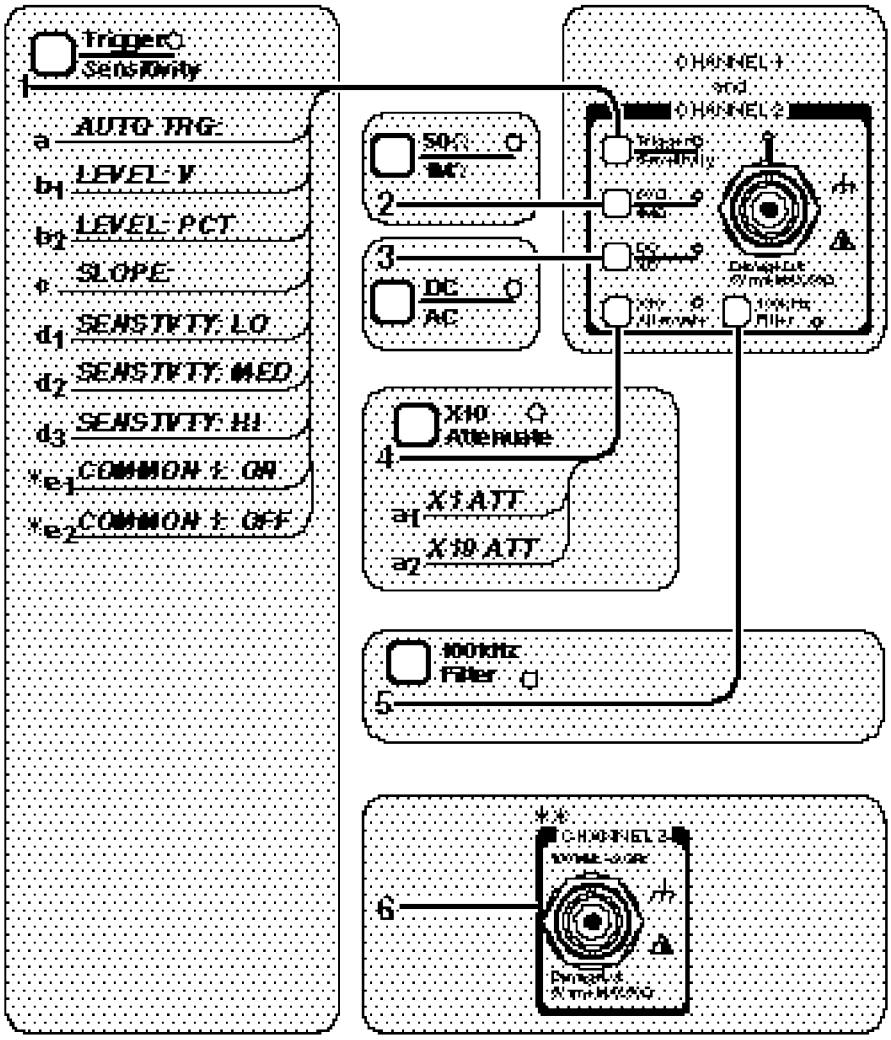
Some SCPI Syntax Conventions

[ ]	An element inside brackets is optional. Note, the brackets are NOT part of the command and should NOT be sent to the Counter.
1 | 2	Means use either 1 or 2.
<numeric_value>	Means enter a number.
SENSe	Means you MUST use either all the upper case letters or the entire word. The lower case letters are optional. For example, SENS and SENSE are both valid. However, SEN is not valid. (Note SENSe is used here as an example, but this convention is true for all SCPI commands.)

NOTE

When you see quotation marks in the command's parameter (shown in the "Parameter Form" column in Table 2-2), you must send the quotation marks with the command. Refer to the section titled "Using BASIC" in Chapter 3 (page 3-61) of this guide for details on how to use double quotes or single quotes to enclose the string parameter of a command.

Input Channels Conditioning Keys to SCPI Command Map



*For ***TI 1 TO 2*** (Time Interval measurements) only.

**Channel 3 is optional.

Figure 2-1. Input Channels Conditioning Keys to SCPI Command Map (Part 1 of 2)

Input Channels Conditioning Keys to SCPI Command Map (Cont.)

1 **a.** [:SENSe]:EVENt[1|2]:LEVel[:ABSolute]:AUTO ON|OFF

b_1. [:SENSe]:EVENt[1|2]:LEVel[:ABSolute] <numeric_value> [V]
b_2. [:SENSe]:EVENt[1|2]:LEVel:RELative <numeric_value> [PCT]

c. [:SENSe]:EVENt[1|2]:SLOPe POSitive | NEGative

d_1. [:SENSe]:EVENt[1|2]:HYSTeresis:RELative 100 *
d_2. [:SENSe]:EVENt[1|2]:HYSTeresis:RELative 50
d_3. [:SENSe]:EVENt[1|2]:HYSTeresis:RELative 0 **

e_1. [:SENSe]:EVENt2:FEED "[:]INPut[1]"
e_2. [:SENSe]:EVENt2:FEED "[:]INPut2"

2 :INPut[1|2]:IMPedance <numeric_value> [OHM]

3 :INPut[1|2]:COUPling AC|DC

4 **a_1.** :INPut[1|2]:ATTenuation 1
a_2. :INPut[1|2]:ATTenuation 10

5 :INPut[1|2]:FILTer ON | OFF

6 :INPut3:COUPling?
:INPut3:IMPedance?

---

*Note, in firmware revisions 3317, 3335 and 3402, use 0.

**Note, in firmware revisions 3317, 3335 and 3402, use 100.

Figure 2-1. Input Channels Conditioning Keys to SCPI Command Map (Part 2 of 2)

Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map

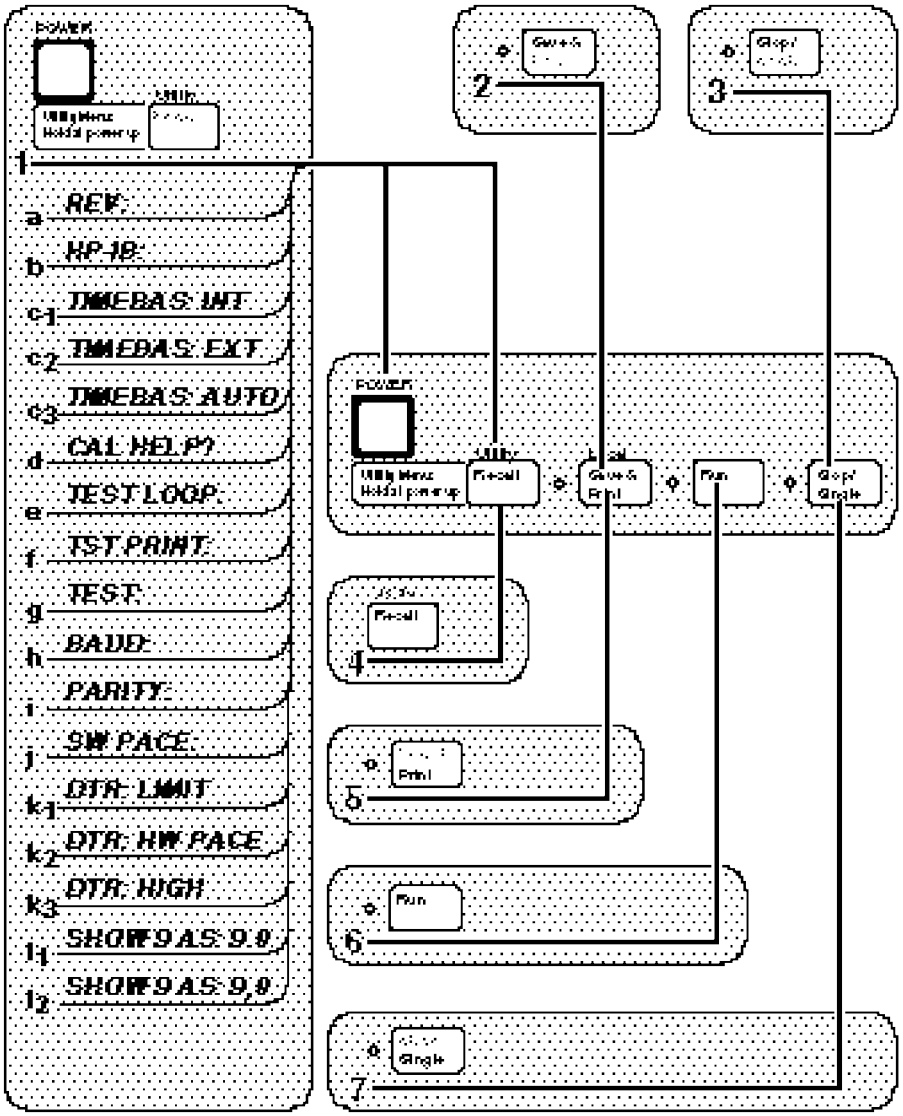


Figure 2-2. Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map (Part 1 of 2)

Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map (Cont.)

1 **a.** *IDN?
b. No command

c_1. [:SENSe]:ROSCillator:SOURce INTernal
c_2. [:SENSe]:ROSCillator:SOURce EXTernal
c_3. [:SENSe]:ROSCillator:SOURce:AUTO ON

d. No command (see Calibration menu, Figure 2-6)

e. No command
f. No command
g. *TST?

h. :SYSTem:COMMunicate:SERial:TRANsmit:BAUD <numeric_value>
I. :SYSTem:COMMunicate:SERial:TRANsmit:PARity[:TYPE] EVEN | ODD | NONE
j. :SYSTem:COMMunicate:SERial:TRANsmit:PACE XON | NONE

k_1. :SYSTem:COMMunicate:SERial:CONTrol:DTR LIMit
k_2. :SYSTem:COMMunicate:SERial:CONTrol:DTR IBFull
k_3. :SYSTem:COMMunicate:SERial:CONTrol:DTR ON

l_1. :DISPlay[:WINDow]:TEXT:RADix DPOint
l_2. :DISPlay[:WINDow]:TEXT:RADix COMMa

2 *SAV <Nrf>

3 :INITiate:CONTinuous OFF (*if running*)
OR
:ABORt (*if single measurement in progress*)

4 *RCL <Nrf>

5 :HCOPy:CONTinuous ON | OFF

6 :INITiate:CONTinuous ON (*if in single*)
OR
:ABORt (*if running*)

7 :INITiate[:IMMediate]

Figure 2-2. Instrument Control, Utility, Recall, and Save & Print Keys to SCPI Command Map (Part 2 of 2)

MEASURE Keys to SCPI Command Map

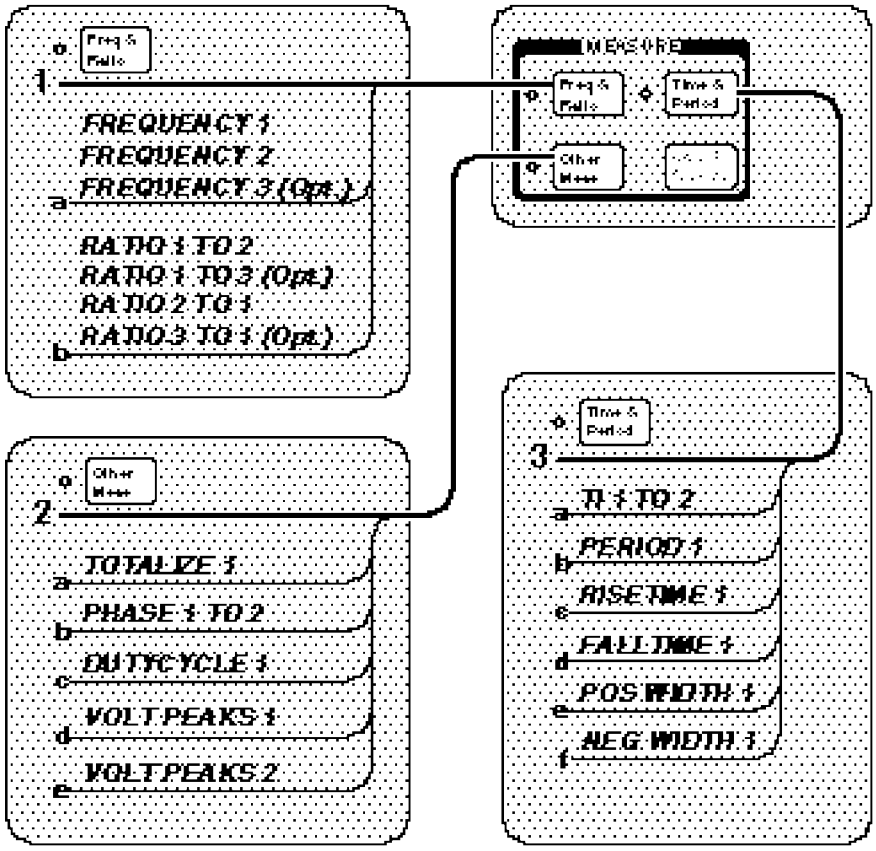


Figure 2-3. MEASURE Keys to SCPI Command Map (Part 1 of 2)

MEASURE Keys to SCPI Command Map (Cont.)

1 **a.** [:SENSe]:FUNCtion[:ON] "[:][XNONe:]FREQuency [1 | 2 | 3]"

b. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]FREQuency:RATio [1,2 | 1,3 | 2,1 | 3,1]"

2 **a.** [:SENSe]:FUNCtion[:ON] "[:][XNONe:]TOTalize [1]"

b. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]PHASe [1,2]"

c. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]DCYCle [1]"

d. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]VOLTage:MINimum [1]"
OR
[:SENSe]:FUNCtion[:ON] "[:][XNONe:]VOLTage:MAXimum [1]"

e. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]VOLTage:MINimum 2"
OR
[:SENSe]:FUNCtion[:ON] "[:][XNONe:]VOLTage:MAXimum 2"

3 **a.** [:SENSe]:FUNCtion[:ON] "[:][XNONe:]TINTerval [1,2]"

b. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]PERiod [1]"

c. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]RISE:TIME [1]"

d. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]FALL:TIME [1]"

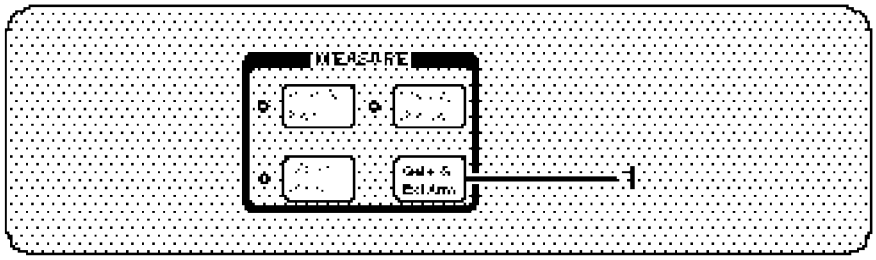
e. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]PWIDth [1]"

f. [:SENSe]:FUNCtion[:ON] "[:][XNONe:]NWIDth [1]"

Since the primary purpose of these front-panel keys is to change the function, the corresponding [:SENSe]:FUNCtion[:ON] command is listed in the menu map above. The front-panel keys, however, invoke couplings which affect other settings, whereas the [:SENSe]:FUNCtion[:ON] command does not.

Figure 2-3. MEASURE Keys to SCPI Command Map (Part 2 of 2)

Gate & ExtArm Key to SCPI Command Map



Freq, Period, Ratio (Agilent 53131A/132A)	**Phase** (Agilent 53131A/132A)	**Totalize** (Agilent 53131A/132A)	**Rise Time, Fall Time,+/-Width, Dutycycle** (Agilent 53131A/132A)	**Time Interval** (Agilent 53131A and Agilent 53132As with S/N prefix below 3646) *See page **2-14** for Agilent 53132A (with S/N prefix 3646 and above).*
Auto Arming: **a.** GATE: AUTO	*Auto Arming*: **a.** ARM: AUTO	*Auto Arming*: **a.** GATE: AUTO	*Auto Arming*: **a.** ARM: AUTO	*Auto Arming*: **a.** ARM: AUTO **b_1.** DELAY: NONE **b_2.** DELAY: TIME **c.** TIME: <time>
Digits Arming: **b.** GATE: DIGITS **c.** DIGITS: <digits>	_____	_____	_____	_____
Time Arming: **d.** GATE: TIME **e.** TIME: <time>	_____	*Time Arming*: **b.** GATE: TIME **c.** TIME: <time>	_____	_____
External Arming: **f.** GATE: EXTERNL **g.** START: POS NEG **h_1.** STOP: AUTO **h_2.** STOP: NEG POS **h_3.** STOP: TIME **I.** TIME: <time>	*External Arming*: **b.** ARM: EXTERNL **c.** SLOPE: POS NEG	*External Arming*: **d.** GATE: EXTERNL **e.** START: POS NEG **f_1.** STOP: TIME **f_2.** STOP: NEG POS **g.** TIME: <time>	*External Arming*: **b.** ARM: EXTERNL **c.** SLOPE: POS NEG	*External Arming*: **d.** ARM: EXTERNL **e.** SLOPE: POS NEG **f_1.** DELAY: NONE **f_2.** DELAY: TIME **g.** TIME: <time>

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 1 of 6)

Gate & ExtArm Key to SCPI Command Map (Cont.)

1

Freq, Period, Ratio

Auto Arming:

a. [:SENSe]:FREQuency:ARM[:STARt]:SOURce IMMediate
[:SENSe]:FREQuency:ARM:STOP:SOURce IMMediate

Digits Arming:

b. [:SENSe]:FREQuency:ARM[:STARt]:SOURce IMMediate
[:SENSe]:FREQuency:ARM:STOP:SOURce DIGits

c. [:SENSe]:FREQuency:ARM:STOP:DIGits <numeric_value>

Time Arming:

d. [:SENSe]:FREQuency:ARM[:STARt]:SOURce IMMediate
[:SENSe]:FREQuency:ARM:STOP:SOURce TIMer

e. [:SENSe]:FREQuency:ARM:STOP:TIMer <numeric_value>

External Arming:

f. [:SENSe]:FREQuency:ARM[:STARt]:SOURce EXTernal

g. [:SENSe]:FREQuency:ARM[:STARt]:SLOPe POSitive | NEGative

h_1. [:SENSe]:FREQuency:ARM:STOP:SOURce IMMediate
h_2. [:SENSe]:FREQuency:ARM:STOP:SOURce EXTernal
[:SENSe]:FREQuency:ARM:STOP:SLOPe POSitive | NEGative
h_3. [:SENSe]:FREQuency:ARM:STOP:SOURce TIMer

I. [:SENSe]:FREQuency:ARM:STOP:TIMer <numeric_value> [S]

Phase

Auto Arming:

a. [:SENSe]:PHASe:ARM[:STARt]:SOURce IMMediate

External Arming:

b. [:SENSe]:PHASe:ARM[:STARt]:SOURce EXTernal
c. [:SENSe]:PHASe:ARM[:STARt]:SLOPe POSitive | NEGative

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 2 of 6)

Gate & ExtArm Key to SCPI Command Map (Cont.)

Totalize

Auto Arming:

a. [:SENSe]:TOTalize:ARM[:STARt]:SOURce IMMediate
[:SENSe]:TOTalize:ARM:STOP:SOURce IMMediate

Time Arming:

b_1. [:SENSe]:TOTalize:ARM[:STARt]:SOURce IMMediate
b_2. [:SENSe]:TOTalize:ARM:STOP:SOURce TIMer

c. [:SENSe]:TOTalize:ARM:STOP:TIMer <numeric_value> [S]

External Arming:

d. [:SENSe]:TOTalize:ARM[:STARt]:SOURce EXTernal

e. [:SENSe]:TOTalize:ARM[:STARt]:SLOPe POSitive | NEGative

f_1. [:SENSe]:TOTalize:ARM:STOP:SOURce TIMer
f_2. [:SENSe]:TOTalize:ARM:STOP:SOURce EXTernal
[:SENSe]:TOTalize:ARM:STOP:SLOPe POSitive | NEGative

g. [:SENSe]:TOTalize:ARM:STOP:TIMer <numeric_value> [S]

Rise Time, Fall Time, +/- Pulse Width, Dutycycle

Auto Arming:

a. [:SENSe]:TINTerval:ARM[:STARt]:SOURce IMMediate

External Arming:

b. [:SENSe]:TINTerval:ARM[:STARt]:SOURce EXTernal

c. [:SENSe]:TINTerval:ARM[:STARt]:SLOPe POSitive | NEGative

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 3 of 6)

Gate & ExtArm Key to SCPI Command Map — For Agilent 53131A (and Agilent 53132A With S/N Prefix Below 3646)

Time Interval (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646)

Auto Arming:

a. [:SENSe]:TINTerval:ARM[:STARt]:SOURce IMMediate

b_1. [:SENSe]:TINTerval:ARM:STOP:SOURce IMMediate
b_2. [:SENSe]:TINTerval:ARM:STOP:SOURce TIMer

c. [:SENSe]:TINTerval:ARM:STOP:TIMer <numeric_value> [S]

External Arming:

d. [:SENSe]:TINTerval:ARM[:STARt]:SOURce EXTernal

e. [:SENSe]:TINTerval:ARM[:START]:SLOPe POSitive | NEGative

f_1. [:SENSe]:TINTerval:ARM:STOP:SOURce IMMediate
f_2. [:SENSe]:TINTerval:ARM:STOP:SOURce TIMer

g. [:SENSe]:TINTerval:ARM:STOP:TIMer <numeric_value> [S]

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 4 of 6)

Gate & ExtArm Key to SCPI Command Map (Cont.)— For Agilent 53132A (With S/N Prefix 3646 and Above)

Time Interval (Agilent 53132A With S/N Prefix 3646 and Above)
Auto Arming: **a.** :START: AUTO **b1.** DELAY? NONE **b2.** DELAY? TIME **b3.** DELAY? EVENT **c.** T? <time> **d.** E? <events>
External Arming: **e.** :START: EXT **f.** :SLOPE: POS NEG **g1.** :DELAY: NONE **g2.** :DELAY: TIME **g3** :DELAY: EVENT **h.** :T: <time> **i.** :E: <events> **j1.** STOP? AUTO **j2.** STOP? EXT **k.** SLOPE? POS NEG **m1.** DELAY? NONE **m2.** DELAY? TIME **m3.** DELAY? EVENT **n.** T? <time> **o.** E? <events>

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 5 of 6)

Gate & ExtArm Key to SCPI Command Map (Cont.) — For Agilent 53132A (With S/N Prefix 3646 and Above)

Time Interval (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646)

Auto Arming:

a. [:SENSe]:TINTerval:ARM:ESTART:LAYer2:SOURce IMMediate
[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce IMMediate

b_1. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce IMMediate
b_2. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce TIMer
b_3. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce INTernal2

c. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:TIMer <numeric_value> [S]

d. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:ECOunt <numeric_value>

External Arming:

e. [:SENSe]:TINTerval:ARM:ESTART:LAYer2:SOURce EXTernal

f. [:SENSe]:TINTerval:ARM:ESTART:LAYer2:SLOPe POSitive | NEGative

g_1. [:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce IMMediate
g_2. [:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce TIMer
g_3. [:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce INTernal1

h. [:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:TIMer <numeric_value> [S]
i. [:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:ECOunt <numeric_value>

j_1. [:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce IMMediate
j_2. [:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce EXTernal
k. [:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SLOPe POSitive I NEGative

m_1. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce IMMediate
m_2. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce TIMer
m_3. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce INTernal2

n. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:TIMer <numeric_value> [S]

o. [:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:ECOunt <numeric_value>

Figure 2-4. Gate & ExtArm Key to SCPI Command Map (Part 6 of 6)

LIMITS and MATH Keys to SCPI Command Map

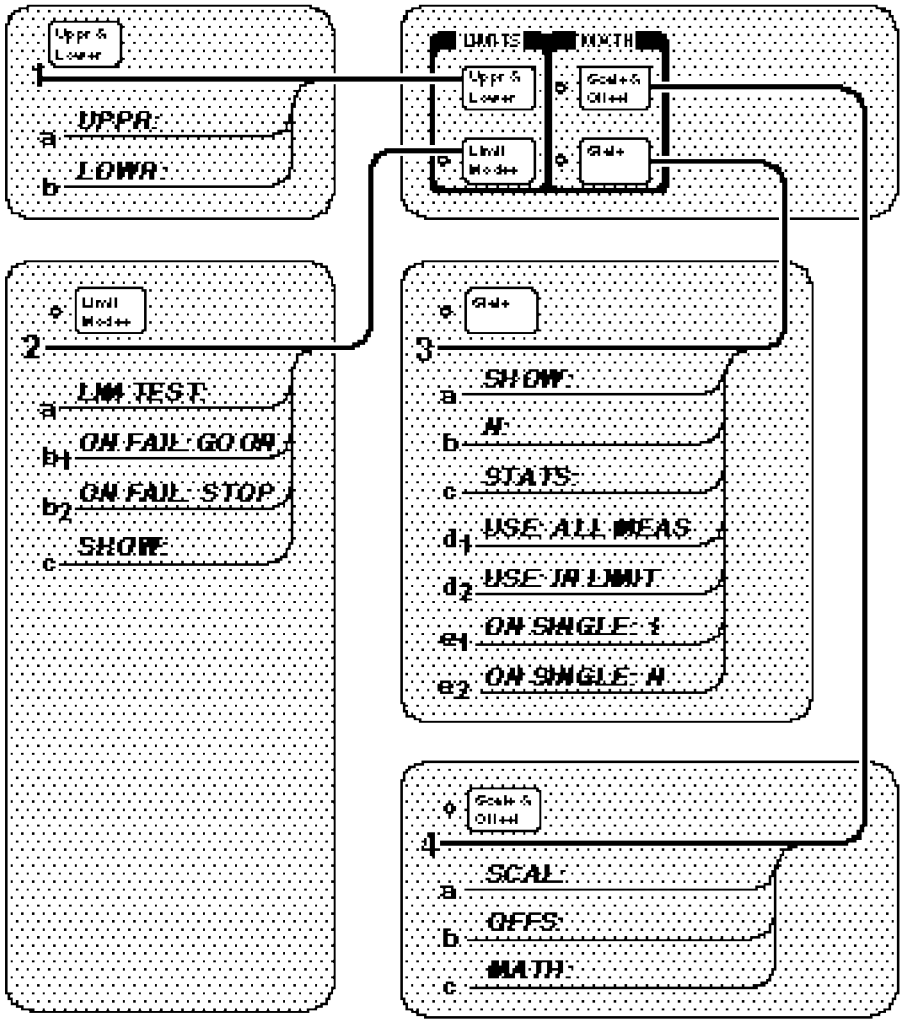


Figure 2-5. LIMITS and MATH Keys to SCPI Command Map (Part 1 of 2)

LIMITS and MATH Keys to SCPI Command Map (Cont.)

1 **a.** :CALCulate2:LIMit:UPPer[:DATA] <numeric_value> [HZ | S | DEG]
b. :CALCulate2:LIMit:LOWer[:DATA] <numeric_value> [HZ | S | DEG]

2 **a.** :CALCulate2:LIMit:STATe OFF | ON

b_1. :INITiate:AUTO OFF
b_2. :INITiate:AUTO ON

c. :CALCulate2:LIMit:DISPlay GRAPh | NUMBer

3 **a.** :DISPlay[:WINDow]:TEXT:FEED “CALC3” *
:CALCulate3:AVERage:TYPE MAXimum | MINimum | SDEViation | MEAN *
OR
:DISPlay[:WINDow]:TEXT:FEED “CALC2” *

b. :CALCulate3:AVERage:COUNt <numeric_value>
c. :CALCulate3:AVERage[:STATe] OFF | ON
d_1. :CALCulate3:LFILter:STATe OFF
d_2. :CALCulate3:LFILter:STATe ON

e_1. :TRIGger:COUNt:AUTO OFF
e_2. :TRIGger:COUNt:AUTO ON

4 **a.** :TRACe[:DATA] SCALE, <numeric_value>
b. :TRACe[:DATA] OFFSET, <numeric_value> [HZ | S | DEG]
c. :CALCulate:MATH:STATe OFF | ON

*Use CALC3:AVER:TYPE and :DISP[:WIND]:TEXT:FEED “CALC3” to specify SHOW: STD DEV, MEAN, MAX, or MIN. Use DISP[:WIND]:TEXT:FEED “CALC2” to specify SHOW: MEAS.

Figure 2-5. LIMITS and MATH Keys to SCPI Command Map (Part 2 of 2)

Calibration Menu to SCPI Command Map

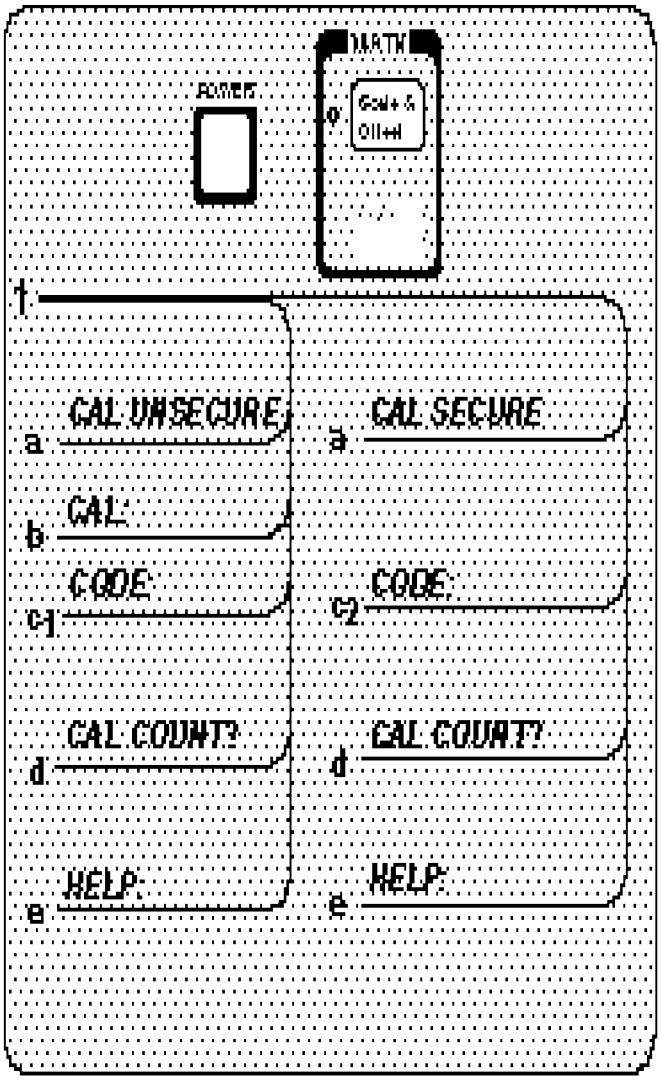


Figure 2-6. Calibration Menu to SCPI Command Map (Part 1 of 2)

Calibration Menu to SCPI Command Map (Cont.)

1 **a.** :CALibration:SECurity:STATe?

b. :DIAGnostic:CALibration:INPut1:OFFSet:AUTO ONCE
:DIAGnostic:CALibration:INPut2:OFFSet:AUTO ONCE
:DIAGnostic:CALibration:INPut1:GAIN:AUTO ONCE
:DIAGnostic:CALibration:INPut2:GAIN:AUTO ONCE
:DIAGnostic:CALibration:TINTerval:QUICk
:DIAGnostic:CALibration:TINTerval:FINE[1 | 2 | 3 | 4]
:DIAGnostic:CALibration:ROSCillator:AUTO ONCE

c_1. :CALibration:SECurity:CODE <new_code>
OR
:CALibration:SECurity:STATe ON, <present_code>

c_2. :CALibration:SECurity:STATe OFF, <present_code>

d. :CALibration:COUNt?

e. No command

The Calibration Menu is accessed by holding the **Scale & Offset** key and cycling **POWER** key.

Figure 2-6. Calibration Menu to SCPI Command Map (Part 2 of 2)

Agilent 53131A/132A Command Summary

This section summarizes both the IEEE 488.2 Common and Agilent 53131A/132A Standard Commands for Programmable Instruments (SCPI) commands in tabular format. IEEE 488.2 Common commands are listed first, followed by SCPI commands.

SCPI Conformance Information

The SCPI commands used in the Agilent 53131A/132A are in conformance with the SCPI Standard Version 1992.0. The SCPI command set consists of the following:

- Common commands as defined in IEEE 488.2-1987—listed and summarized in Table 2-1.

- SCPI Subsystem commands as confirmed (and listed) in the SCPI Standard—the commands defined in Table 2-2 as "Std."

- SCPI Subsystem commands designed for the instrument in conformance with SCPI standards but not yet listed in the SCPI Standard—the commands defined in Table 2-2 as "New."

Details of all Agilent 53131A/132A commands can be found in Chapter 4, "Command Reference" of this programming guide.

Information on the SCPI commands format, syntax, parameter, and response types is provided in Chapter 3, "Programming Your Universal Counter for Remote Operation," of this programming guide.

IEEE 488.2 Common Commands

The Common Commands are general purpose commands that are common to all instruments (as defined in IEEE 488.2). Common Commands are easy to recognize because they all begin with an "*" (for example, *RST, *IDN?, *OPC). These commands are generally not related to measurement configuration. They are used for functions like resetting the instrument, identification, or synchronization.

Table 2-1 lists the Common Commands in alphabetical order by mnemonic, name and function. More information concerning the operation of IEEE 488.2 status reporting commands and structure can be found in the "Status Reporting" section of Chapter 3. Standard explanations of the IEEE 488.2 Common commands can be found in the *ANSI/IEEE Std. 488.2-1987, IEEE Standard Codes, Formats, Protocols, and Common Commands* document.

Table 2-1. IEEE 488.2 Common Commands

Mnemonic	Command Name	Function
*CAL?	Calibration	Causes the Counter to perform an internal interpolator self-calibration and returns a response that indicates whether or not the instrument completed the self-calibration without error.
*CLS	Clear Status	Clears Status data structures (Event Registers and Error Queue).
*DDT <arbitrary block>	Define Device Trigger Command	Defines either INIT, FETC?, READ?, or nothing to be executed when the Counter receives a GET or *TRG command.
*DMC <string>, <arbitrary block>	Define Macro Command	Assigns a sequence of zero or more commands/queries to a macro label. No query form.
*EMC <NRf>	Enable Macro Command	Enables and disables expansion of macros. Non-zero value enables; zero value disables.
*EMC?	Enable Macro Query	Queries whether macros are enabled.
*ESE <NRf>	Standard Event Status Enable	Sets the Standard Event Status Enable Register.
*ESE?	Standard Event Status Enable Query	Queries the Standard Event Status Enable Register.
*ESR?	Event Status Register Query	Queries the Standard Event Status Register.
*GMC? <string>	Get Macro Contents Query	Queries the current definition of a currently defined macro label.
*IDN?	Identification Query	Queries the Counter identification.
*LMC?	Learn Macro Query	Queries the currently defined macro labels.
*OPC	Operation Complete	Causes Counter to set the operation complete bit in the Standard Event Status Register when all pending operations (see Note) are finished.
*OPC?	Operation Complete Query	Places an ASCII “1” in the Output Queue when all pending operations (see Note) are completed.
Note: Pending operations include measurements in progress.		

Table 2-1. IEEE 488.2 Common Commands (Continued)

Mnemonic	Command Name	Function
*OPT?	Option Identification Query	Identifies the options installed in the Counter.
*PMC	Purge Macro Command	Deletes all macros previously defined using the *DMC command.
*RCL <NRf>	Recall	Restores the state of the Counter from a copy stored in local non-volatile memory (0 through 20 are valid memory registers).
*RST	Reset	Resets the Counter to a known state.
*SAV <NRf>	Save	Stores the current state of the Counter in local non-volatile memory (1 through 20 are valid memory registers).
*SRE <NRf>	Service Request Enable	Set the Service Request Enable register.
*SRE?	Service Request Enable Query	Queries the Service Request Enable register.
*STB?	Status Byte Query	Queries the Status Byte and Master Summary Status bit.
*TRG	Trigger	This trigger command is the device-specific analog of the IEEE 488.1 defined GET. It initiates measurement, unless *DDT was used to redefine device trigger.
*TST?	Self-Test Query	Executes an internal self-test and reports the results.
*WAI	Wait-to-Continue	Makes Counter wait until all pending operations (see Note) are completed before executing commands following *WAI command.
Note: Pending operations include measurements in progress.		

Agilent 53131A/132A SCPI Subsystem Commands

SCPI Subsystem commands include all measurement functions and some general purpose functions. SCPI Subsystem Commands use a hierarchy relationship between keywords that is indicated by a ":" (colon). For example, in the SYST:ERR? query, the ":" between SYST and ERR? indicates ERR? is subordinate to SYST.

Table 2-2 lists the SCPI Subsystem Commands in alphabetical order by the command keyword. The table shows the Subsystem commands hierarchical relationship, related parameters (if any), and any associated information and comments.

Not all commands have a query form. Unless a command is specified as "No Query" or "Query Only" in the "Comments" column of Table 2-2, it has both a command and a query form. Any command in the table that is shown with a "?" at the end,
is a "Query Only" command.

Std/New Column

The **Std/New** column in Table 2-2 gives the status of the command with respect to the SCPI standard. The "Std" commands operate as defined in the SCPI standard and as defined in this guide.

The category of "New" consists of commands that could be:

- SCPI approved but are not yet in the SCPI manual
- Agilent approved and submitted for SCPI approval.
- Not approved at all.

The "New" commands operate as defined in this guide.

Parameter Form Column

Refer to the section titled "Parameter Types" on page 3-11 in Chapter 3, "Programming Your Universal Counter for Remote Operation," for descriptions of the different parameter types (such as <Boolean>, <NRf>, <arbitrary block>, etc.).

Table 2-2. Agilent 53131A/132A SCPI Command Summary

Keyword/Syntax	Parameter Form	Std/ New	Comments
:ABORt		Std	Event; no query. Aborts measurement in progress.
:CALCulate[1]		Std	Subsystem. Performs post-acquisition math processing (scale and offset) and data transfer on the data acquired by a SENSe function.
:DATA?		Std	Query only. Returns scaled/offset measurement result.
:FEED	"[:]SENSe[1]"	Std	Sets the data flow to be fed into the CALCulate block.
:IMMediate		Std	Event or query; causes the Counter to recalculate existing data without re-acquiring.
:AUTO	<Boolean>	Std	Enables/disables automatic post-processing.
:MATH		Std	Subtree.
[:EXPRession]		Std	Subtree.
:CATalog?		New	Returns the name of the defined equation, SCALE_OFFSET.
[:DEFine]?		New	Returns the expression (equation) used for math (scale/offset) processing.
:NAME | :SELect	SCALE_OFFSET	New	Sets the name of selected math expression (equation).
:STATe	<Boolean>	Std	Enables/disables math (scale/offset) processing. Note that this setting must be enabled for any of the other :CALC[1] settings to be used.
:CALCulate2		Std	Subsystem. Performs post-acquisition LIMit testing and data transfer.
:FEED	"[:]CALCulate[1]"	Std	Sets the data flow to be fed into the CALCulate2 block.
:IMMediate		Std	Event; no query. Causes the Counter to recalculate existing data without re-acquiring.
:AUTO	<Boolean>	Std	Enables/disables automatic post-processing.
:LIMit		Std	Subtree. Collects together the commands associated with controlling and getting reports from a single LIMit test.
:CLEar		Std	Subtree.
:AUTO	<Boolean>	Std	Enables the automatic clearing of limit test results.
[:IMMediate]		Std	Event; no query. Clears the limit test results.
:DISPlay	GRAPh | NUMBer	New	Sets whether the measurement display is numeric or symbolic (on a graph).
:FAIL?		Std	Query only. Returns a 0 or 1 to indicate if the last tested measurement passed or failed the limit test. 0 = pass; 1 = fail.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:CALCulate2 (Cont.)			
:LIMit (Cont.)			
:FCOunt		Std	Subtree. An abbreviation for Fail COunt.
:LOWer?		New	Query only. Returns the number of limit test failures at the lower limit.
:UPPer?		New	Query only. Returns the number of limit test failures at the upper limit.
[:TOTal]?		New	Query only. Returns the total number of measurements that failed the limit test.
:LOWer		Std	Subtree.
[:DATA]	<numeric_value> [HZ | S | DEG]	Std	Sets lower limit used in limit testing.
:STATe	<Boolean>	Std	Sets the limit test enable. Note that this setting must be enabled for any of the other :CALC2 settings can be used.
:UPPer		Std	Subtree.
[:DATA]	<numeric_value> [HZ | S | DEG]	Std	Sets upper limit used in limit testing.
:PCOunt		New	Subtree. An abbreviation for Pass COunt.
[:TOTal]?		New	Query only. Returns the total number of measurements that passed the limit test.
:CALCulate3		Std	Subsystem. Performs post-acquisition statistics computation and data transfer.
:AVERage		Std	Subtree. Collects together the commands associated with the Statistics capabilities.
:ALL?		New	Returns all four Statistics results (i.e., mean, standard deviation, maximum, and minimum).
:CLEar		Std	Event; no query. Clears the statistics results and statistics count.
:COUNt	<numeric_value>	Std	Selects number of measurements to combine for statistics.
:CURRent?		New	Query only. Returns the current number of data values collected, thus far.
[:STATe]	<Boolean>	Std	Enables/disables statistics post-processing. Note that this setting must be enabled for any of the other :CALC3 settings to be used.
:TYPE	MAXimum | MINimum | SDEViation | SCALar or MEAN	Std	Selects which statistic will be in :CALC3:DATA?,and on the front-panel display.
:DATA?		Std	Query only. Returns statistic result specified by :CALC3:AVER:TYPE.
:FEED	"[:]CALCulate[1]"	Std	Sets the data flow to be fed into the CALCulate3 block.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:CALCulate3 (Cont.)			
:LFILter		New	Subtree. Limit FILter for statistics.
:LOWer		New	Subtree.
[:DATA]	<numeric_value> [HZ | S | DEG]	New	Sets the statistics filter lower limit.
:STATe	<Boolean>	New	Sets the statistics filter enable.
:UPPer		New	Subtree.
[:DATA]	<numeric_value> [HZ | S | DEG]	New	Sets the statistics filter upper limit.
:PATH?		Std	Query only. Returns LFIL, AVER.
:CALibration		Std	Subsystem.
[:ALL]?		Std	Query only. Causes an internal interpolator self-calibration.
:COUNt?		New	Query only. Returns value indicating number of times the Counter has been calibrated.
:DATA	<arbitrary block>	Std	Transfers the calibration data (input gain, input offset, reference oscillator, and time interval).
:SECurity		New	No query. Sets the calibration security code.
:CODE	<NRf>	New	Enables or prevents calibration of the Counter.
:STATe	<Boolean>, <NRf>	New	Query returns security status. 0 = unsecure; calibration allowed. 1 = secure; calibration disallowed.
:CONFigure		Std	See Measurement Instructions in this table.
:DIAGnostic		Std	Subsystem.
:CALibration		New	Subtree.
:INPut[1|2]		New	Subtree. 1 | 2 specifies channel.
:GAIN		New	Subtree.
:AUTO	ONCE | OFF	New	ONCE calibrates input gain.
:OFFSet		New	Subtree.
:AUTO	ONCE | OFF	New	ONCE calibrates input offset.
:INTerpolator		New	Subtree.
:AUTO	ONCE | OFF | ON	New	ONCE calibrates the interpolators.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:DIAGnostic (Cont.)			
:CALibration (Cont.)			
:ROSCillator		New	Subtree. ROSCillator is an abbreviation for Reference OSCillator.
:AUTO	ONCE | OFF	New	ONCE calibrates the timebase. This command is usable only if the instrument contains the medium or high stability oscillator option.
:STATus?		New	Query only. Returns status of last calibration. 0 = pass; 1 = fail.
:TINTerval		New	Subtree.
:FINE	[1 | 2 | 3 | 4]	New	Event; no query. Four steps which calibrate out Ch1, 2 electrical path length differences.
:QUICk		New	Event; no query. Calibrates out Ch1, 2 electrical path length differences.
:DISPlay		Std	Subsystem. Controls the selection and presentation of textual information on the display.
:ENABle	<Boolean>	Std	Controls whether the whole display is visible.
:MENU		Std	Subtree.
[:STATe]	OFF	Std	Sets the Counter to switch from the menu display to the result display.
[:WINDow]		Std	Subtree.
:TEXT		Std	Subtree. Allows for the display of textual information.
:FEED	"[:]CALCulate2" | "[:]CALCulate3"	Std	Sets which data flow is fed into the display. "CALC2" specifies the raw measurement, scaled/offset measurement, or Limit Graph display. "CALC3" specifies the statistics result display.
:RADix	COMMa | DPOint	New	Sets the character used to separate integral and fractional portions of a number. (USA numerical convention is Decimal POint.)
:FETCh		Std	See Measurement Instructions in this table.
:FORMat		Std	Subsystem. Sets a data format for transferring numeric information.
[:DATA]	ASCii | REAL	Std	Sets the data format.
:HCOPy		New	
:CONTinuous	<Boolean>	New	Enables or disables printing results.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:INITiate		Std	Subsystem. Controls the initiation of measurements.
:AUTO	<Boolean>	New	AUTO ON enables the Counter to automatically stop measuring on a limit test failure. AUTO OFF disables the automatic stop.
:CONTinuous	<Boolean>	Std	Sets the enable for continuously initiated measurements.
[:IMMediate]		Std	Event; no query. Causes the instrument to initiate the number of measurements specified by :TRIGger:COUNt:AUTO.
:INPut[1|2]		Std	Subsystem. Controls the characteristics of the instrument's input ports. :INPut1= channel 1 and :INPut2= channel 2
:ATTenuation	1 | 10	Std	Sets input attenuation.
:COUPling	AC | DC	Std	Sets input coupling.
:FILTer		Std	Subtree. Allows a low pass filter to be inserted in the path of the measurement signal.
[:LPASs]		Std	Subtree. Controls the Low PASs filter.
[:STATe]	<Boolean>	Std	Sets the Low PASs filter enable.
:FREQuency?		Std	Query only. Returns the cutoff frequency of the low pass filter. Units are Hertz.
:IMPedance	<numeric_value> [OHM]	Std	Sets input impedance (50 W or 1 MW).
:INPut3		Std	Subsystem. Queries the characteristics of the Counter's input channel 3.
:COUPling?		Std	Query only. Returns channel 3 input coupling.
:IMPedance?		Std	Query only. Returns channel 3 input impedance.
:MEASure		Std	See Measurement Instructions in this table.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
Measurement Instructions*			
:CONFigure[:SCALar]:<function>	See <parameters> and <source_list> in table on the next page.	Std	Configures instrument to perform specified measurement.
:CONFigure?		Std	Returns function configured by the last :CONF or :MEAS command.
:MEASure[:SCALar]:<function>?	See <parameters> and <source_list> in table on the next page.	Std	Configures instrument, initiates measurement, and queries for the result (i.e., provides complete measurement sequence).
:READ[[:SCALar]:<function>]?		Std	Initiates measurement, and queries for the result. (Performs a :FETCh? on “fresh” data.)
:FETCh[[:SCALar]:<function>]?		Std	Queries the result.
*The <function> and corresponding <parameters> and <source list> are defined by the following listing in this table (see next page).			

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

<function> *	<parameters>	[,<source_list>] **	Std/ New
[:VOLTage]:DCYCle or [:VOLTage]:PDUTycycle	[<reference>]	[(@1)]	Std
[:VOLTage]:FALL:TIME or [:VOLTage]:FTIMe	[<lower_reference>[,upper_reference>]]	[(@1)]	Std
[:VOLTage]:FREQuency	[<expected_value>[,<resolution>]]	[(@1) | (@2) | (@3)]	Std
[:VOLTage]:FREQuency:RATio***	[<expected_value>[,<resolution>]]	[(@1), (@2 | @3) | (@2 | @3), (@1)]	New
[:VOLTage]:MAXimum		[(@1) | (@2)]	Std
[:VOLTage]:MINimum		[(@1) | (@2)]	Std
[:VOLTage]:NWIDth	[<reference>]	[(@1)]	Std
[:VOLTage]:PERiod	[<expected_value>[,<resolution>]]	[(@1) | (@2) | (@3)]	Std
[:VOLTage]:PHASe		[(@1), (@2)]	Std
[:VOLTage]:PTPeak		[(@1) | (@2)]	Std
[:VOLTage]:PWIDth	[<reference>]	[(@1)]	Std
[:VOLTage]:RISE:TIME or [:VOLTage]:RTIMe	[<lower_reference>[,upper_reference>]]	[(@1)]	Std
[:VOLTage]:TINTerval		[(@1), (@2)]	New
[:VOLTage]:TOTalize:CONTinuous****		[(@1)]	New
[:VOLTage]:TOTalize:TIMed	[<gate_time>]	[(@1)]	New

* The only functions which can be derived (using FETC? or READ?) from the stored data are period to/from frequency, maximum to/from minimum, maximum to/from peak-to-peak, and minimum to/from peak-to-peak. All other functions require an acquisition of a new type.

** <source_list> has the same syntax as SCPI <channel _list> syntax. For example, a single-channel function (e.g., frequency, period, etc.) would use (@1) to specify channel 1, whereas a two-channel function (e.g., time interval, phase, and frequency ratio) would use (@1), (@2) to specify a measurement between channel 1 and channel 2.

*** For Agilent 53131A firmware revisions below 3335, only Ratio 1 to 2 and Ratio 1 to 3 were offered.

**** This <function> is only allowed with :CONFigure.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:MEMory		Std	Subsystem. Manages instrument memory.
:DELete		Std	Subtree.
:MACRo	<string>	New	Event; no query. Deletes the macro with the name specified by the string parameter.
:FREE		Std	Subtree.
:MACRo?		Std	Query only. Returns memory usage and availability corresponding to macro data.
:NSTates?		Std	Query only. Returns the number of available *SAV/*RCL states in the instrument.
:READ		Std	See Measurement Instructions in this table.
[:SENSe]		Std	Subsystem setup commands.
:DATA?	["[:]SENSe[1]"]	Std	Query only. Returns the current measurement result data of the SENSe subsystem (no scale or offset applied).
:EVENt[1|2]		New	Subtree. Defines the "trigger event."
:HYSTeresis		New	Subtree.
:RELative	<numeric_value> [PCT]	New	[1]Sets the size of the hysteresis window as a percentage of allowable hysteresis. [2]Sets the sensitivity of the input channel as a percentage of allowable sensitivity.
:LEVel		New	Subtree.
[:ABSolute]	<numeric_value> [V]	New	Sets the level at the center of the hysteresis window.
:AUTO	<Boolean>	New	Sets the "auto-trigger" enable.
:RELative	<numeric_value> [PCT]	New	Sets the percentage of the peak-to-peak range of the signal at which the instrument will auto trigger. 0-100%.
:SLOPe	POSitive | NEGative	New	Sets which edge of the input signal will be considered an event.
:EVENt2		New	Subtree.
:FEED	"[:]INPut[1] | [:]INPut2"	New	Sets the common/separate enable. INPut2 is separate; INPut1 is common. (Only applies for Time Interval function.)

[1]Current firmware revision.

[2]Prior firmware revisions 3317, 3335, and 3402.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:EVENt3		New	Subtree. Queries the characteristics of the “trigger event” for channel 3 input.
:LEVel		New	Subtree.
[:ABSolute]?		New	Query only. Returns the channel 3 input trigger level.
:SLOPe?		New	Query only. Returns the edge of the channel 3 input that will be considered an event.
:FREQuency		Std	Subtree. Controls the frequency, frequency ratio, and period measuring capabilities of the instrument.
:ARM		New	Subtree. Synchronizes the frequency start and stop arm with events.
[:STARt]		New	Subtree.
:SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming frequency, frequency ratio, and period measurements. Only applies when [:SENS]:FREQ:ARM[:STAR]:SOUR EXT is selected.
:SOURce	IMMediate | EXTernal	New	Sets the start arm for frequency, frequency ratio, and period measurements.
:STOP		New	Subtree.
:DIGits	<numeric_value>	New	Sets the resolution in terms of digits used in arming frequency, frequency ratio, and period measurements. Only applies when [:SENS]:FREQ:ARM:STOP:SOUR DIG is selected.
:SLOPe	POSitive | NEGative	New	Sets the slope of the external stop arm signal used in external arming frequency, frequency ratio, and period measurements. Only applies when [:SENS]:FREQ:ARM:STOP:SOUR EXT is selected.
:SOURce	IMMediate | EXTernal | TIMer | DIGits	New	Sets the stop arm for frequency, frequency ratio, and period measurements.
:TIMer	<numeric_value> [S]	New	Sets the gate time used in arming frequency, frequency ratio, and period measurements. Only applies when [:SENS]:FREQ:ARM:STOP:SOUR TIM is selected.
:EXPected[1|2|3]	<numeric_value> [HZ]	New	Specifies the approximate frequency of a signal you expect to measure at channel 1, 2, or 3.
:AUTO	ON	New	Configures Counter to perform a pre-measurement step to automatically determine the approximate frequency of the measurement signal(s).

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:FUNCtion		Std	Subtree. Selects the <sensor function> to be sensed by the instrument.
[:ON]	<sensor_function> (See below)	Std	Sets the <sensor function> to be sensed by the instrument.
	"[:][XNONe:]DCYCle [1]"	New	(Positive) duty cycle on channel 1.
	"[:][XNONe:]PDUTycycle [1]"	New	(Positive) duty cycle on channel 1.
	"[:][XNONe:]FALL:TIME [1]"	New	Fall Time on channel 1.
	"[:][XNONe:]FTIMe [1]"	New	Fall Time on channel 1.
	"[:][XNONe:]FREQuency [1 | 2 | 3]"	Std	Frequency on channel 1, 2, or 3.
	"[:][XNONe:]FREQuency:RATio [1,2 | 1,3 | 2,1 | 3,1]"	Std	[1]Frequency Ratio 1 to 2, 1 to 3, 2 to 1, or 3 to 1.
	"[:][XNONe:]NWIDth [1]"	New	Negative Pulse Width on channel 1.
	"[:][XNONe:]PERiod [1 | 2 | 3]"	Std	Period on channel 1, 2, or 3.
	"[:][XNONe:]PHASe [1,2]"	Std	Phase of channel 1 relative to channel 2.
	"[:][XNONe:]PWIDth [1]"	New	Positive Pulse Width on channel 1.
	"[:][XNONe:]RISE:TIME [1]"	New	Rise Time on channel 1.
	"[:][XNONe:]RTIMe [1]"	New	Rise Time on channel 1.
	"[:][XNONe:]TINTerval [1,2]"	New	Time Interval channel 1 to channel 2.
	"[:][XNONe:]TOTalize [1]"	Std	Totalize on channel 1.
	"[:][XNONe:]VOLTage:MAXimum [1 | 2]"	Std	Voltage Maximum on channel 1 or 2.
	"[:][XNONe:]VOLTage:MINimum [1 | 2]"	New	Voltage Minimum on channel 1 or 2.
	"[:]{XNONe:]VOLTage:PTPeak [1 | 2]	New	Voltage Peak to Peak on Channel 1 or 2.
:PHASe		New	Subtree. Controls the phase measuring capabilities of the instrument.
:ARM		New	Subtree.
[:STARt]		New	Subtree. Synchronizes Phase start arm with events.
:SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming phase measurements. Only applies when [:SENS]:PHAS:ARM[:STAR]:SOUR EXT is selected.
:SOURce	IMMediate | EXTernal	New	Sets the start arm for phase measurements.
:ROSCillator		Std	Subtree. Controls the Reference OSCillator.
:EXTernal		Std	Subtree.
:CHECk	ON | OFF | ONCE	New	Set the enable for checking the validity and presence of the external reference.
:FREQuency?		Std	Query only. Returns the frequency value of the external reference oscillator.

[1]For Agilent 53131A firmware revisions below 3335, only Ratio 1 to 2 and Ratio 1 to 3 were offered.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:ROSCillator (Cont.)			
:SOURce	INTernal | EXTernal	Std	Sets the selection of a reference timebase.
:AUTO	<Boolean>	Std	Sets the enable for automatically selecting a reference timebase.
:TINTerval		New	Subtree. Controls the time interval (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width functions) measuring capabilities of the instrument.
:ARM **(Agilent 53131A and Agilent 53132A with S/N prefix below 3646)**	**See Table 2-2A on page 2-38 for Agilent 53132A (with S/N 3646 and above) TINTerval ARM commands)**	New	Subtree. Synchronizes the time interval start and stop arm with events.
		New	Subtree.
[:STARt] :SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming time interval measurements (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width functions). Only applies when [:SENS]:TINT:ARM[:STAR]:SOUR EXT is selected.
:SOURce	IMMediate | EXTernal	New	Sets the start arm for time interval (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width) measurements.
		New	Subtree.
:STOP :SOURce	IMMediate | TIMer	New	Sets the stop arm (i.e., the delay) for Time Interval measurements.
:TIMer	<numeric_value> [S]	New	Sets the time used to delay the stop arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:STOP:SOUR TIM is selected.
:TOTalize		New	Subtree. Controls the totalize measuring capabilities of the instrument.
:ARM		New	Subtree. Synchronizes totalize start and stop arm with events.
		New	Subtree.
[:STARt] :SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming totalize measurements. Only applies when [:SENS]:TOT:ARM[:STAR]:SOUR EXT is selected.
:SOURce	EXTernal | IMMediate	New	Sets the start arm for totalize measurements.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:TOTalize (Cont.)			
:ARM (Cont.)			
:STOP		New	Subtree.
:SLOPe	POSitive | NEGative	New	Sets the slope of the external stop arm signal used in external arming totalize measurements. Only applies when [:SENS]:TOT:ARM:STOP:SOUR EXT is selected.
:SOURce	EXTernal | TIMer | IMMediate	New	Sets the stop arm for totalize measurements.
:TIMer	<numeric_value> [S]	New	Sets the gate time used in totalize measurements. Only applies when [:SENS]:TOT:ARM:STOP:SOUR TIM is selected.
:STATus		Std	Subsystem. Controls the SCPI-defined (Operation and Questionable) status-reporting structures.
:OPERation		Std	Subtree.
:CONDition?		Std	Query only. Queries the Operation Condition Status Register.
:ENABle	<non-decimal numeric> |	Std	Sets the Operation Event Status Enable Register.
[:EVENt]?	<NRf>	Std	Query only. Queries the Operation Event Status Register.
:NTRansition	<non-decimal numeric> | <Nrf>	Std	Sets the negative transition filter for the Operation status reporting structure.
:PTRansition	<non-decimal numeric> | <NRf>	Std	Sets the positive transition filter for the Operation status reporting structure.
:PRESet		Std	Event; No query. Presets the enable registers and transition filters associated with the Operation and Questionable status reporting structures.
:QUEStionable		Std	Subtree.
:CONDition?		Std	Query only. Queries the Questionable Data Condition Status Register.
:ENABle	<non-decimal numeric> | <NRf>	Std	Sets the Questionable Data Event Status Enable Register.
[:EVENt]?		Std	Query only. Queries the Questionable Data Event Status Register.
:NTRansition	<non-decimal numeric> | <NRf>	Std	Sets the positive transition filter for the Questionable Data status reporting structure.
:PTRansition	<non-decimal numeric> | <NRf>	Std	Sets the negative transition filter for the Questionable Data status reporting structure.

Table 2-2. Agilent 53131A/132A SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
:SYSTem		Std	Subsystem. Collects the functions that are not related to instrument performance.
:COMMunicate		Std	Subtree. Collects together configuration of control/communication interfaces.
:SERial		Std	Subtree. Controls the physical configuration of the RS- 232C port.
:CONTrol		Std	Subtree.
:DTR	IBFull | ON | LIMit	Std	Sets the usage of the DTR line of the RS-232 port.
:TRANsmit		Std	Subtree. Affects parameters associated with transmission.
:BAUD	<numeric_value>	Std	Sets the baud rate.
:PACE	XON | NONE	Std	Sets the software pacing scheme.
:PARity		Std	Subtree. Controls the parity of the channel.
[:TYPE]	EVEN | ODD | NONE	Std	Sets the parity scheme.
:ERRor?		Std	Query only. Queries the oldest error in the Error Queue and removes the error from the queue (first in, first out).
:KEY	<numeric_value>	Std	Simulates the pressing of a front-panel key.
:LOG?		New	Query only. Returns a comma-separated list of integers representing all of the entries in the Key Queue.
:VERSion?		Std	Query only. Returns the SCPI version number with which the Counter complies.
:TRACe		Std	Subsystem.
:CATalog?		Std	Query only. Returns list of intrinsic constants.
[:DATA]	OFFSET, <numeric_value> [HZ | S | DEG]	Std	Sets the offset value.
[:DATA]	SCALE, <numeric_value>	Std	Sets the scale value.
[:DATA]?	OFFSET	Std	Queries the offset value.
[:DATA]?	SCALE	Std	Queries the scale value.
:TRIGger		Std	Subsystem.
:COUNt		Std	Subtree.
:AUTO	<Boolean>	New	Controls the number of measurements to be made when :INIT[:IMM] is performed.

Table 2-2A. Agilent 53132A (S/N Prefix 3646 and Above) Time Interval Arming SCPI Command Summary

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:TINTerval		New	Subtree. Controls the time interval (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width functions) measuring capabilities of the instrument.
:ARM		New	Subtree. Synchronizes the time interval start and stop arm with events.
:ESTART		New	
:LAYer2		New	
:SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming Time Interval measurements (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width).
:SOURce	IMMediate | EXTernal	New	Sets the start arm for time interval (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width) measurements.
[:LAYer[1]]		New	
:ECOunt	<numeric_value>	New	Sets the event count used to delay the start arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:ESTART:SOUR INT1 is selected.
:SOURce	IMMediate | TIMer | INTernal1	New	Sets the start arm delay for Time Interval measurements.
:TIMer	<numeric_value>	New	Sets the time used to delay the start arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:ESTART:SOUR TIM is selected.
:ESTOP		New	
:LAYer2		New	
:SLOPe	POSitive | NEGative	New	Sets the slope of the external stop arm signal used in external arming Time Interval measurements.
:SOURce	IMMediate | EXTernal	New	Sets the stop arm for Time Interval measurements.
[:LAYer[1]]		New	
:ECOunt	<numeric_value>	New	Sets the event count used to delay the stop arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:ESTOP:SOUR INT2 is selected.

Table 2-2A. Agilent 53132A (S/N Prefix 3646 and Above) Time Interval Arming SCPI Command Summary (Continued)

Keyword/Syntax	Parameter Form	Std/ New	Comments
[:SENSe] (Cont.)			
:TINTerval (Cont.)			
:ARM (Cont.)			
:ESTOP (Cont.)			
[:LAYer[1]]			
:SOURce	IMMediate | TIMer | INTernal2	New	Sets the stop arm delay for Time Interval measurements.
:TIMer	<numeric_value>	New	Sets the time used to delay the stop arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:ESTOP:SOUR TIM is selected.
[:STARt]		New	
:SLOPe	POSitive | NEGative	New	Sets the slope of the external start arm signal used in external arming Tme Interval measurements (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width). Only applies when [:SENS]:TINT:ARM[:STAR]:SOUR EXT is selected.
:SOURce	IMMediate | EXTernal	New	Sets the start arm for time interval (including Time Interval, Rise Time, Fall Time, Dutycycle, and Pulse Width) measurements.
:STOP		New	Subtree.
:SOURce	IMMediate | TIMer	New	Sets the stop arm (i.e., the delay) for Time Interval measurements.
:TIMer	<numeric_value>	New	Sets the time used to delay the stop arm for Time Interval measurements. Only applies when [:SENS]:TINT:ARM:STOP:SOUR TIM is selected.

*RST Response

The IEEE 488.2 *RST command returns the instrument to a specified state optimized for remote operation. (Use *CLS to clear the status event registers and the SCPI error queue.)

The states of commands affected by the *RST command are described in Table 2-3. Since the Agilent 53131A and Agilent 53132A have different arming capabilities for Time Interval measurements, sub-tables (Table 2-3A and Table 2-3B) are provided. Table 2-3A describes the state of the Agilent 53131A's [:SENSe]:Interval commands affected by the *RST command, and Table 2-3B describes the Agilent 531312A's [:SENSe]:Interval commands affected by the *RST command. Table 2-4 lists commands that are unaffected by *RST.

Table 2-3. Agilent 53131A/132A *RST State

Command Header	Parameter	State
:CALCulate[1]:FEED :CALCulate[1]:IMMediate:AUTO :CALCulate[1]:MATH[:EXPRession]:NAME :CALCulate[1]:MATH:STATe	"[:]SENSe[1]" <Boolean> SCALE_OFFSET <Boolean>	"SENSe[1]" OFF SCALE_OFFSET OFF
:CALCulate2:FEED :CALCulate2:IMMediate:AUTO :CALCulate2:LIMit:CLEar:AUTO :CALCulate2:LIMit:DISPlay :CALCulate2:LIMit:LOWer[:DATA] :CALCulate2:LIMit:STATe :CALCulate2:LIMit:UPPer[:DATA]	"[:]CALCulate[1]" <Boolean> <Boolean> GRAPh | NUMBer <numeric_value> <Boolean> <numeric_value>	"CALCulate[1]" OFF ON NUMBer 0.0000000000 OFF 0.0000000000
:CALCulate3:AVERage:COUNt :CALCulate3:AVERage[:STATe] :CALCulate3:AVERage:TYPE :CALCulate3:FEED :CALCulate3:LFILter:LOWer[:DATA] :CALCulate3:LFILter:STATe :CALCulate3:LFILter:UPPer[:DATA]	<numeric_value> <Boolean> MAXimum | MINimum | SDEViation | MEAN "[:]CALCulate[1]" <numeric_value> <Boolean> <numeric_value>	100 OFF MEAN "CALCulate[1]" 0.0000000000 OFF 0.0000000000
*DDT	<arbitrary block>	#14INIT
:DIAGnostic:CALibration:INTerpolator:AUTO	ON | OFF | ONCE	ON
:DISPlay:ENABle :DISPlay:MENU[:STATe] :DISPlay[:WINDow]:TEXT:FEED	<Boolean> OFF "[:]CALCulate2" | "[:]CALCulate3"	ON OFF "CALCulate2"

Table 2-3. Agilent 53131A/132A *RST State (Continued)

Command Header	Parameter	State
*EMC	<NRf>	0 (i.e., disabled)
:FORMat[:DATA]	ASCii | REAL	ASCii
:HCOPy:CONTinuous	<Boolean>	OFF
:INITiate:AUTO :INITiate:CONTinuous	<Boolean> <Boolean>	OFF OFF
:INPut[1|2]:ATTenuation :INPut[1|2]:COUPling :INPut[1|2]:FILTer[:LPASs]:STATe :INPut[1|2]:IMPedance	1 | 10 AC | DC <Boolean> <numeric_value> [OHM]	1 AC OFF 1E6 OHM
[:SENSe]:EVENt[1|2]:HYSTeresis:RELative	<numeric_value> [PCT]	100 PCT *
[:SENSe]:EVENt[1|2]:LEVel[:ABSolute]:AUTO [:SENSe]:EVENt[1|2]:LEVel:RELative [:SENSe]:EVENt[1|2]:LEVel:SLOPe	<Boolean> <numeric_value> [PCT] POSitive | NEGative	ON 50 PCT POSitive
[:SENSe]:EVENt2:FEED	"INPut1 | INPut2"	"INPut2" (i.e., separate mode)
[:SENSe]:FREQuency:ARM[:STARt]:SLOPe [:SENSe]:FREQuency:ARM[:STARt]:SOURce	POSitive | NEGative IMMediate | EXTernal	POSitive IMMediate
[:SENSe]:FREQuency:ARM:STOP:DIGits [:SENSe]:FREQuency:ARM:STOP:SLOPe [:SENSe]:FREQuency:ARM:STOP:SOURce [:SENSe]:FREQuency:ARM:STOP:TIMer [:SENSe]:FREQuency:EXPected[1|2|3]:AUTO	<numeric_value> POSitive | NEGative IMMediate | EXTernal | TIMer | DIGits <numeric _value> [S] ON	4 NEGative TIMer 100E-3 S ON
[:SENSe]:FUNCtion[:ON]	<sensor_function>	"FREQuency 1"
[:SENSe]:PHASe:ARM[:STARt]:SLOPe [:SENSe]:PHASe:ARM[:STARt]:SOURce	POSitive | NEGative IMMediate | EXTernal	POSitive IMMediate
[:SENSe]:ROSCillator:EXTernal:CHECk [:SENSe]:ROSCillator:SOURce:AUTO	ON | OFF | ONCE <Boolean>	ON ON

*For prior firmware revisions (3317, 3335, and 3402), *RST sets up "100%" meaning maximum sensitivity.

For current firmware revision, *RST sets up "0%" meaning minimum hysteresis.

Table 2-3A. Agilent 53131A (and Agilent 53132A With S/N Prefix Below 3646)Time Interval *RST State

Command Header	Parameter	State
[:SENSe]:TINTerval:ARM[:STARt]:SLOPe	POSitive | NEGative	POSitive
[:SENSe]:TINTerval:ARM[:STARt]:SOURce	IMMediate | EXTernal	IMMediate
[:SENSe]:TINTerval:ARM:STOP:SOURce	IMMediate | TIMer	IMMediate
[:SENSe]:TINTerval:ARM:STOP:TIMer	<numeric_value> [S]	10E-3 S

Table 2-3B. Agilent 53132A (S/N 3646 and above) Time Interval *RST State

Command Header	Parameter	State
[:SENSe]:TINTerval:ARM:ESTART:LAYer2:SLOPe	POSitive | NEGative	POSitive
[:SENSe]:TINTerval:ARM:ESTART:LAYer2:SOURce	IMMediate | EXTernal	IMMediate
[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:ECOunt	<numeric_value>	1
[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce	IMMediate | TIMer | INTernal1	IMMediate
[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:TIMer	<numeric_value>	0.1s
[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SLOPe	POSitive | NEGative	POSitive
[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce	IMMediate | TIMer	IMMediate
[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:ECOunt	<numeric_value>	1
[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce	IMMediate | TIMer | INTernal2	IMMediate
[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:TIMer	<numeric_value>	0.1s
[:SENSe]:TINTerval:ARM[:STARt]:SLOPe	POSitive | NEGative	POSitive
[:SENSe]:TINTerval:ARM[:STARt]:SOURce	IMMediate | EXTernal	IMMediate
[:SENSe]:TINTerval:ARM:STOP:SOURce	IMMediate | TIMer	IMMediate
[:SENSe]:TINTerval:ARM:STOP:TIMer	<numeric_value>	0.1s

Table 2-4. Unaffected by *RST

Item
*ESE
*OPC?
*SRE
*WAI
:CALibration:COUNt?
:CALibration:DATA
:CALibration:SECurity:CODE
:CALibration:SECurity:STATe
:DISPlay[:WINDow]:TEXT:RADix
:STATus:OPERation:ENABle
:STATus:OPERation:NTRansition
:STATus:OPERation:PTRansition
:STATus:QUEStionable:ENABle
:STATus:QUEStionable:NTRansition
:STATus:QUEStionable:PTRansition
:SYSTem:COMMunicate:SERial:CONTrol:DTR
:SYSTem:COMMunicate:SERial:TRANsmit:BAUD
:SYSTem:COMMunicate:SERial:TRANsmit:PACE
:SYSTem:COMMunicate:SERial:TRANsmit:PARity[:TYPE]
:SYSTem:ERRor? (Error Queue)
GPIB Address

***RST Response**

3

Programming Your Universal Counter for Remote Operation

Introduction

This chapter provides remote operation setup, and programming information that helps you operate the Counter as a remote device.

Chapter Summary

- Configuring the GPIB pg. 3-4
- Overview of Command Types and Formats pg. 3-7
- Elements of SCPI Commands pg. 3-8
- Using Multiple Commands pg. 3-13
- Overview of Response Message Formats pg. 3-15
- Status Reporting pg. 3-18
- Command Settings for Optimizing Throughput pg. 3-20
- How to Program the Counter for Status Reporting pg. 3-38
- How to Program the Counter to Display Results pg. 3-43
- How to Program the Counter to Synchronize
- Measurements pg. 3-46
- How to Program the Counter for Math/Limit Operation pg. 3-49
- How to Program the Counter to Define Macros pg. 3-52
- Writing SCPI Programs pg. 3-55
- **Programming Examples** pg. 3-58

Where to Find Some Specific Information

- To Set the GPIB Mode and Address pg. 3-4
- To Connect the Counter to a Computer pg. 3-6
- Remote/Local Operation pg. 3-6
- Common Command Format pg. 3-7
- SCPI Command and Query Format pg. 3-7
- Abbreviated Commands, Keyword Separator pg. 3-9
- Optional Keyword pg. 3-9
- Implied Channel (Optional Numeric Keyword Suffix) pg. 3-10
- Parameter Types pg. 3-10
- Parameter Separator, Query Parameters, Suffixes pg. 3-11
- Command Terminator pg. 3-13

- Program Messages pg. 3-14
- Response Messages, Response Message Syntax pg. 3-16

Where to Find BASIC Programming Examples

- Easiest Way to Make a Measurement pg. 3-59
- To Make a Frequency Measurement pg. 3-62
- To Perform Limit Testing pg. 3-63
- To Measure the Statistics of 50 Measurements pg. 3-64
- To Use Limits to Filter Data Before Measuring Stats pg. 3-66
- To Read and Store Calibration Data pg. 3-68
- To To Perform a Time Interval Calibration pg. 3-69
- To Optimize Throughput pg. 3-73
- To Use Macros pg. 3-75

Where to Find QuickBASIC Programming Examples

- To Make a Frequency Measurement pg. 3-77
- To Perform Limit Testing pg. 3-78
- To Measure the Statistics of 50 Measurements pg. 3-80
- To Use Limits to Filter Data Before Measuring Stats pg. 3-82
- To Read and Store Calibration Data pg. 3-85
- To Optimize Throughput pg. 3-86
- To Use Macros pg. 3-88

Where to Find Turbo C Programming Examples

- To Make a Frequency Measurement pg. 3-91
- To Use Limits to Filter Data Before Measuring Stats pg. 3-93
- To Optimize Throughput pg. 3-96

Configuring the GPIB

This section gives information on connecting and configuring the GPIB to enable remote operation of the Counter.

The Counter has two GPIB operating modes:

- Addressed (talk/listen)—This mode is for bi-directional communication. The Counter can receive commands and setups from the computer, and can send data and measurement results.

  To select the talk/listen operating mode, set the Counter's GPIB address from 0 to 30. Refer to the following section titled "To Set the GPIB Mode and Address" for instructions on how to set an GPIB address from the front-panel.

- Talk-only—In this mode, the Counter can send data to a printer. It cannot receive commands or setups from the computer.

  To select the talk-only operating mode, set the Counter's GPIB mode to "TALK". Refer to the following section titled "To Set the GPIB Mode and Address" for instructions on how to set the talk-only mode from the front-panel.

When the Counter is shipped from the factory, it is configured as addressed (talk/listen) with the address set to "3."

To Set the GPIB Mode and Address

1 Press and hold **Recall (Utility)** key, then cycle **POWER**.

2 Press **Recall (Utility)** key until ***HP-IB****:* is displayed.

To best demonstrate how to set the address, let's assume that ***HP-IB: 3*** is currently being displayed.

3a To set the address to “15”, perform the following:

a. Press s key.

HP-IB: ***0****3* is displayed. Note that “0” digit appears and is highlighted, indicating that this digit will change when the d or f arrow key is pressed.

b. Press d key.

HP-IB: ***1****3* is displayed.

c. Press g key

HP-IB: ***13*** is displayed, but now the “3” digit is highlighted, indicating that this digit will change when the d or f arrow key is pressed.

d. Press d key twice or until 15 is displayed.

HP-IB: ***15*** is displayed.

e. Go on to step 4.

3b To set the GPIB mode to “TALK,” perform the following:

a. Press s key.

HP-IB: ***0****3* is displayed. Note that “0” digit appears and is highlighted, indicating that this digit will change when the d or f arrow key is pressed.

b. Press d key until “TALK” is displayed.

HP-IB: ***TALK*** is displayed.

c. Go on to step 4.

4 Press **Enter** key.

NOTE

BE SURE to press the Enter key to complete the entry.

The address/mode is now stored in non-volatile memory, and *does not* change when power is cycled or after a remote interface reset.

To Connect the Counter to a Computer

Connect the Counter to a computer by simply installing an GPIB cable (such as an Agilent 10833A cable) between the two units as shown in Figure 3-1.

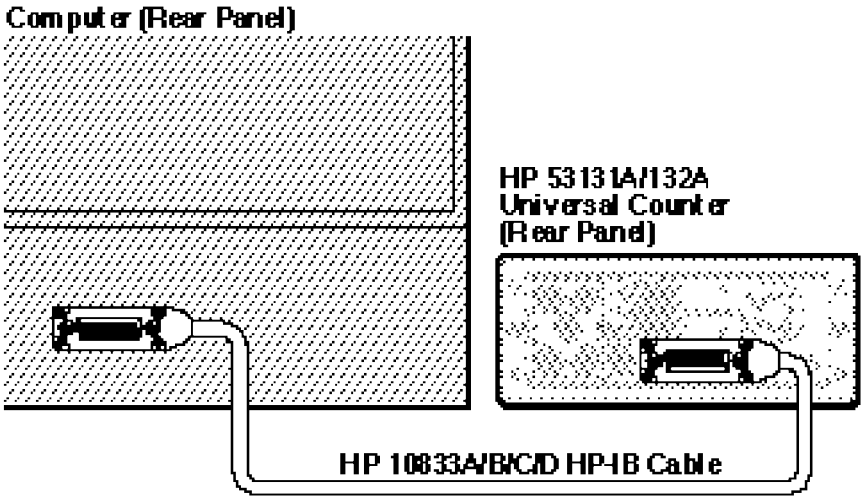


Figure 3-1. GPIB Interconnection

Remote/Local Operation

At power-up, the Counter is under front-panel (local) control. Once in remote, the Counter settings cannot be affected by the front-panel controls. The **Save & Print** key may be used to manually return to local control (only if local-lockout is off).

Overview of Command Types and Formats

There are two types of Agilent 53131A/132A programming commands: IEEE 488.2 Common Commands and Standard Commands for Programmable Instruments (SCPI). The IEEE 488.2 Common Commands control and manage communications between the Agilent 53131A/132A and the controller or personal computer. The SCPI commands control instrument functions. The format of each type of command is described in the following paragraphs. (Refer to
Chapter 2, “Command Summary,” for SCPI conformance information.)

Common Command Format

The IEEE 488.2 Standard defines the Common commands as commands that perform functions like reset, self-test, status byte query, and identification. Common commands always begin with the asterisk (*) character, and may include parameters. The command keyword is separated from the first parameter by a space character. Some examples of Common commands are as follows:

```
*RST          *IDN?          *RCL 1
```

SCPI Command and Query Format

SCPI commands perform functions like instrument setup. A subsystem command has a hierarchical structure that usually consists of a top level (or root) keyword, one or more lower-level keywords, and parameters. The following example shows a command and its associated query:

```
:INPut:COUPling AC
:INPut:COUPling?
```

INPut is root-level keyword with COUPling the second level keyword, and AC is the command parameter.

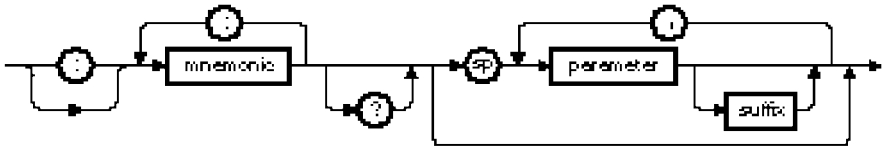
Elements of SCPI Commands

A program command or query is composed of functional elements that include a header (or keywords with colon separators), program data, and terminators. These elements are sent to the Counter over the GPIB as a sequence of ASCII data messages. Examples of a typical Common Command and Subsystem Command are:

```
OUTPUT 712;"*CLS"
OUTPUT 712;":INP1:COUP AC;IMP 1.0 MOHM"
```

Subsystem Command Syntax

Figure 3-2 shows the simplified syntax of a Subsystem Command. You must use a space (SP) between the last command mnemonic and the first parameter in a Subsystem Command. Note that if you send more than one parameter with a single command, you must separate adjacent parameters with a comma.

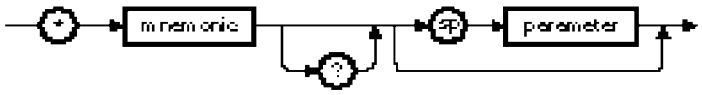


NOTE: sp = space. ASCII character decimal 32

Figure 3-2. Simplified Program Command Syntax Diagram

Common Command Syntax

Figure 3-3 shows the simplified syntax of a Common Command. You must use a space (SP) between the command mnemonic and the parameter in a Common Command.



NOTE: sp = space. ASCII character decimal 32

Figure 3-3. Simplified Common Command Syntax Diagram

Abbreviated Commands

The command syntax shows most keywords as a mixture of upper and lower case letters. Upper case letters indicate the abbreviated spelling for the command. For better program readability, you may send the entire keyword. The Agilent 53131A/132A accepts either command form and is not case sensitive.

For example, if the command syntax shows CALCulate, then CALC and CALCULATE are both acceptable forms. Other forms of CALCulate, such as CALCU or CALCULA will generate an error. You may use upper and/or lower case letters. Therefore, CALCULATE, calculate, and CaLcUlAtE are all acceptable.

Keyword Separator

A colon (:) always separates one keyword from the next lower-level keyword as shown below:

```
:INPut:COUPling?
```

Optional Keyword

Optional keywords are those which appear in square brackets ([ ]) in the command syntax. (Note that the brackets are not part of the command and are not sent to the Counter.)

Suppose you send a second level keyword without the preceding optional keyword. In this case, the Counter assumes you intend to use the optional keyword and responds as if you had sent it.

Examine the portion of the [:SENSe] subsystem shown below:

```
[:SENSe]
  :FREQuency
    :ARM
      :STOP
        :SOURce EXTernal
```

The root-level keyword [:SENSe] is an optional keyword. To set the Counter's frequency stop arm to external, you can use either of the following:

```
:SENS:FREQ:ARM:STOP:SOUR EXT
                or
:FREQ:ARM:STOP:SOUR EXT
```

Implied Channel (Optional Numeric Keyword Suffix)

Some commands allow specifying a channel with an optional numeric keyword suffix. These commands will show the channel numbers within square brackets. The brackets are not part of the command and are not sent to the Counter.

For example, :INPut[1|2]:COUPling AC | DC represents coupling commands for channels 1 and 2:

```
:INPut[1]:COUPling  AC | DC
:INPut2:COUPling  AC | DC
```

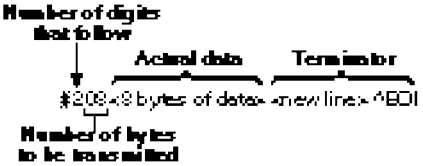
If you do not specify the channel number, the implied channel is 1. For example, you can send either of the following to configure
channel 1's coupling to AC:

```
:INPut1:COUPling AC
        or
:INPut:COUPling AC
```

Parameter Types

Table 3-1 contains explanations and examples of parameter types. Parameter types may be numeric value, Boolean, literal, NRf, string, non-decimal numeric, or arbitrary block.

Table 3-1. Command and Query Parameter Types

TYPE	EXPLANATIONS AND EXAMPLES
<numeric value>	Accepts all commonly used decimal representation of numbers including optional signs, decimal points, and scientific notation: 123, 123e2, -123, -1.23e2, .123, 1.23e-2, 1.23000E-01. Special cases include MINimum and MAXimum as follows: MINimum selects minimum value available. MAXimum selects maximum value available. Queries using MINimum or MAXimum return the associated numeric value.
<Boolean>	Represents a single binary condition that is either true or false: 1 or ON, 0 or OFF (Query response returns only 1 or 0.) An <NRf> is rounded to an integer. A non-zero value is interpreted as 1.
<literal>	Selects from a finite number of choices. These parameters use mnemonics to represent each valid setting. An example is the INPut:COUPling AC | DC command parameters (AC | DC).
<NRf>	Flexible numeric representation. Only positive integers are used for NRf parameters in the Counter.
<string>	A string parameter is delimited by either single quotes or double quotes. Within the quotes, any characters in the ASCII 7-bit code may be specified. The following BASIC program statement sends a command containing a <string> parameter: `OUTPUT 703;"FUNC 'FREQ'"`
<non-decimal numeric>	Format for specifying hexadecimal (#H1F), octal (#Q1077), and binary (#B10101011) numbers using ASCII characters. May be used in :STATus subsystem commands.
<arbitrary block>	The syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the subsequent decimal integer. The decimal integer specifies the number of 8-bit data bytes being sent. This is followed by the actual data. The terminator is a line feed asserted with EOI. For example, for transmitting 8 bytes of data, the format could be:  The “2” indicates the number of digits that follow and the two digits “08” indicate the number of *data* bytes to be transmitted. A zero-length block has the format: #0<new line>^EOI <new line> is defined as a single ASCII-encoded byte corresponding to 10 decimal.

Parameter Separator

If you send more than one parameter with a single command, you must separate adjacent parameters with a comma.

Query Parameters

All selectable <numeric value> parameters can be queried to return the minimum or maximum values they are capable of being set to by sending a MINimum or MAXimum parameter after the "?." For example, consider the INPut:IMPedance? query.

If you send the query without specifying a parameter (INP:IMP?), the present impedance value is returned. If you send the MIN parameter (using INP:IMP? MIN), the command returns the minimum level currently available. If you send the MAX parameter, the command returns the maximum level currently available. Be sure to place a space between the question mark and the parameter.

Suffixes

A suffix is the combination of suffix elements and multipliers that can be used to interpret the <numeric value> sent. If a suffix is not specified, the Counter assumes that <numeric value> is unscaled (that is, Volts, seconds, etc.)

For example, the following two commands are equivalent:

```
OUTPUT 703;"INP:IMP 1 MOHM"
OUTPUT 703;"INP:IMP 1E+6"
```

Suffix Elements

Suffix elements, such as HZ (Hertz), S (seconds), V (volts), OHM (Ohms), PCT (percent), and DEG (degrees) are allowed within this format.

Suffix Multipliers

Table 3-2 lists the suffix multipliers that can be used with suffix elements (except PCT and DEG).

Table 3-2. Suffix Multipliers

DEFINITION	MNEMONIC	NAME
1E18	EX	ETA
1E15	PE	PETA
1E12	T	TERA
1E9	G	GIGA
1E6	MA (or M for OHM and HZ)*	MEGA
1E3	K	KILO
1E-3	M (except for OHM and HZ)*	MILLI
1E-6	U	MICRO
1E-9	N	NANO
1E-12	P	PICO
1E-15	F	FEMTO
1E-18	A	ATTO

*The suffix units, MHZ and MOHM, are special cases that should not be confused with <suffix multiplier>HZ and <suffix multiplier>OHM.

Command Terminator

A command may be terminated with a <new line> (ASCII character decimal 10), an EOI (End-of-Identify) asserted concurrent with last byte, or an EOI asserted concurrent with a <new line> as the last byte.

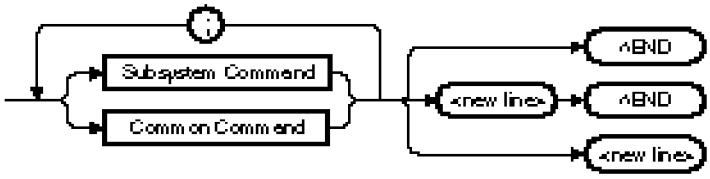
Using Multiple Commands

Program Messages

Program Messages are a combination of one or more properly formatted SCPI Commands. Program messages always go from a computer to the Counter. They are sent to the Counter over the Counter's GPIB as a sequence of ASCII data messages.

Program Message Syntax

Figure 3-4 shows the simplified syntax of a program message. You can see Common Commands and Subsystem Commands in the same program message. If you send more than one command in one message, you must separate adjacent commands with a semicolon.



NOTE:
<new line> = ASCII character decimal 10
^END = EOI asserted concurrent with last byte

Figure 3-4. Simplified Program Message Syntax Diagram

When using IEEE 488.2 Common commands with SCPI Subsystem commands on the same line, use a semicolon between adjacent commands. For example:

```
*RST;:INP:COUP AC
```

When multiple subsystem commands are sent in one program message, the first command is always referenced to the root node. Subsequent commands, separated by ";", are referenced to the same level as the preceding command if no ":" is present immediately after the command separator (the semicolon).

For example, sending :INP:COUP AC;IMP 50 is equivalent to sending:

```
:INP:COUP AC
:INP:IMP 50
      or
:INP:COUP AC;:INP:IMP 50
```

The ":" must be present to distinguish another root level command. For example:

```
:INP:COUP AC;:INIT:CONT OFF
```

is equivalent to sending:

```
:INP:COUP AC
:INIT:CONT OFF
```

If the ":"(which is following the ";" and is in front of INIT) is omitted, the Counter assumes that the second command is ":INP:INIT:CONT OFF" and generates a syntax error.

Overview of Response Message Formats

Response Messages

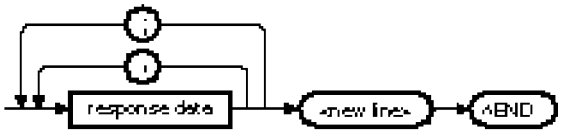
Response messages are data sent from the Counter to a computer in response to a query. (A query is a command followed by a question mark. Queries are used to find out how the Counter is currently configured and to transfer data from the Counter to the computer.)

After receiving a query, the Counter interrogates the requested configuration and places the response in its GPIB output queue.
The output message remains in the queue until it is read or another command is issued. When read, the message is transmitted across the GPIB to the computer. You read the message by using some type of enter statement that includes the device address and an appropriate variable. Use a print statement to display the message. The following BASIC example illustrates how to query the Counter and display the message:

```
10  OUTPUT 703;":INP:COUP?"
20  ENTER 703; A$
30  PRINT A$
40  END
```

Response Message Syntax

Figure 3-5 shows the simplified syntax of a Response Message. Response messages may contain both commas and semicolon separators. When a single query command returns multiple values, a comma is used to separate each item. When multiple queries are sent in the same program message, the groups of data corresponding to each query are separated by a semicolon. Note that a <new line> ^END is always sent as a response message terminator.



NOTE:
<new line> = ASCII character decimal 10
^END = EOI asserted concurrent with last byte
; = multiple response separator (ASCII character decimal 59)
, = data separator within a response (ASCII character decimal 44)

Figure 3-5. Simplified Response Message Syntax Diagram

Response Message Data Types

Table 3-3 contains explanations of response data types.

Table 3-3. Response Message Data Types

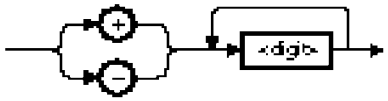
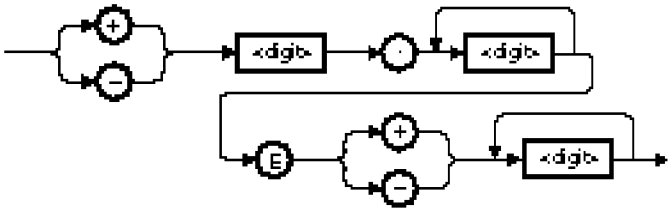
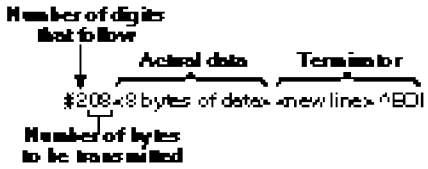
Type	Description
<NR1>	This numeric representation has an implicit radix point.  The maximum number of characters in <NR1> response data is 17 (maximum 16 digits, 1 sign).
<NR2>	This numeric representation has an explicit radix point.  The maximum number of characters in <NR2> response data is 17 (maximum 15 mantissa digits, 1 sign, 1 decimal point).
<NR3>	This numeric representation has an explicit radix point and an exponent.  The maximum number of characters in <NR3> response data is 22 (maximum 15 mantissa digits, 2 signs, 1 decimal point, 1 'E' character, 3 exponent digits).
Not a Number	Not a Number is represented by the value 9.91E37. (Not a Number is defined in IEEE 754). The Counter responds with this numeric value when queried for a floating point number it cannot provide. This value will be formatted as an <NR3>.

Table 3-3. Response Message Data Types (Continued)

Type	Description
<Boolean>	A single ASCII-encoded byte, 0 or 1, is returned for the query of settings that use <Boolean> parameters.
<literal>	ASCII-encoded bytes corresponding to the short form of the literal used as the command parameter. For example, if the :CALC3:AVER:TYPE MAXimum command is sent to the Counter, the :CALC3:AVER:TYPE? response would be MAX.
<string>	A string response consists of ASCII characters enclosed by double quotes. For example, string data is used for the "<error description>" portion of :SYST:ERR? response and for [:SENS]:FUNC? response.
<definite length block>	The syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the subsequent decimal integer. The decimal integer specifies the number of 8-bit data bytes being sent. This is followed by the actual data. The terminator is a line feed asserted with EOI. For example, for transmitting 8 bytes of data, the format might be:  The "2" indicates the number of digits that follow and the two digits "08" indicate the number of *data* bytes to be transmitted. A zero-length block has the format: #0<new line>^EOI <new line> is defined as a single ASCII-encoded byte corresponding to 10 decimal.

Status Reporting

The Agilent 53131A/132A status registers conform to the SCPI and IEEE 488.2 standards.

Figure 3-6 shows all the status system register groups and queues in the Counter. This is a high level drawing that does not show all the registers that are contained in each group. It is intended as a guide to the bits used in each of these register groups to monitor the Counter's status. Note that besides the Operation Status and the Questionable Data/Signal Register groups, a summary of the Standard Status Structure Registers (defined by IEEE 488.2-1987) is shown.

Refer to the section in this chapter titled "How to Program the Counter for Status Reporting" and the flowchart in Figure 3-10 for detailed information on programming the status reporting system.

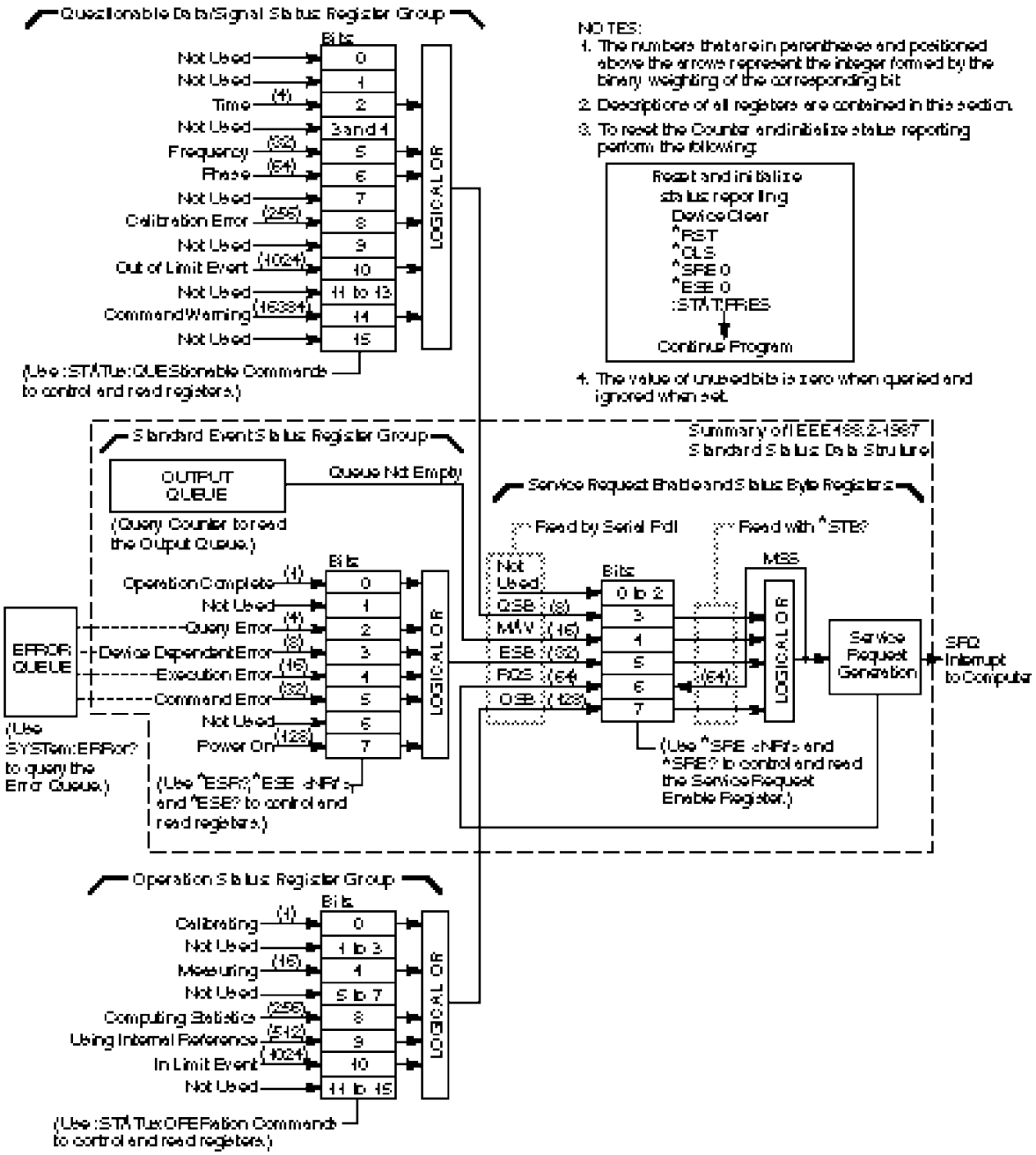


Figure 3-6. Agilent 53131A/132A SCPI Status Reporting Summary Functional Diagram

Status Byte Register and Service Request Enable Register

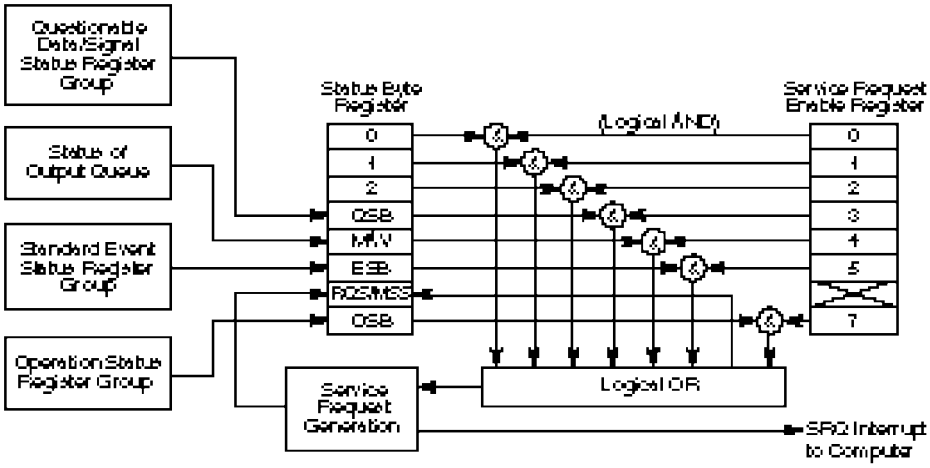


Figure 3-7. Status Byte and Service Request Enable

Status Byte Register

The Status Byte Register is the summary-level register in the status reporting structure. It contains summary bits that monitor activity in the other status registers and queues as shown in Figure 3-7.
The Status Byte Register is a live register—its summary bits are set TRUE or FALSE (one or zero) by the presence or absence of the condition which is being summarized.

The Status Byte Register can be read with either a serial poll or the *STB? query.

The Status Byte Register is altered only when the state of the overlying status data structures is altered.

The entire Status Byte Register can be cleared by sending the *CLS command, by itself in a program message, to the Counter.

Table 3-4 lists the Status Byte Register bits and briefly describes each bit.

Table 3-4. Status Byte Register

BIT	WEIGHT	SYMBOL	DESCRIPTION
0			Not used
1			Not used
2			Not used
3	8	QSB	Questionable Data/Signal Status Register Summary Bit
4	16	MAV	Message Available Summary Bit
5	32	ESB	Standard Event Status Register Summary Bit
6	64	RQS/MSS	Request Service/Master Status Summary Bit
7	128	OSB	Operation Status Register Summary Bit

A detailed description of each bit in the Status Byte Register follows:

- **Bits 0–2** are not used.

- **Bit 3 (QSB)** summarizes the Questionable Data/Signal Status Event Register.

  This bit indicates whether or not one or more of the enabled Questionable Data/Signal events have occurred since the last reading or clearing of the Questionable Data/Signal Status Event Register.

  This bit is set TRUE (one) when an enabled event in the Questionable Data/Signal Status Event Register is set TRUE. Conversely, this bit is set FALSE (zero) when no enabled events are set TRUE.

- **Bit 4 (MAV)** summarizes the Output Queue.

  This bit indicates whether or not the Output Queue is empty.

  This bit is set TRUE (one) when the Counter is ready to accept a request by the external computer to output data bytes; that is, the Output Queue is not empty. This bit is set FALSE (zero) when the Output Queue is empty.

- **Bit 5 (ESB)** summarizes the Standard Event Status Register.

  This bit indicates whether or not one of the enabled Standard Event Status Register events have occurred since the last reading or clearing of the Standard Event Status Register.

  This bit is set TRUE (one) when an enabled event in the Standard Event Status Register is set TRUE. Conversely, this bit is set FALSE (zero) when no enabled events are set TRUE.

- **Bit 6 (RQS/MSS)** summarizes IEEE 488.1 RQS and Master Summary Status.

  When a serial poll is used to read the Status Byte Register, the RQS bit indicates if the device was sending SRQ TRUE. The RQS bit is set FALSE by a serial poll.

  When *STB? is used to read the Status Byte Register, the MSS bit indicates the Master Summary Status. The MSS bit indicates whether or not the Counter has at least one reason for requesting service.

- **Bit 7 (OSB)** summarizes the Operation Status Event Register.

  This bit indicates whether or not one or more of the enabled Operation events have occurred since the last reading or clearing of the Operation Status Event Register.

  This bit is set TRUE (one) when an enabled event in the Operation Status Event Register is set TRUE. Conversely, this bit is set FALSE (zero) when no enabled events are set TRUE.

Service Request Enable Register

The Service Request Enable Register selects which summary bits in the Status Byte Register may cause service requests as shown in Figure 3-7.

Use *SRE to write to this register and *SRE? to read this register.

Use *SRE 0 to clear the register. A cleared register does not allow status information to generate the service requests. (Power-on also clears this register.)

Standard Event Status Register Group

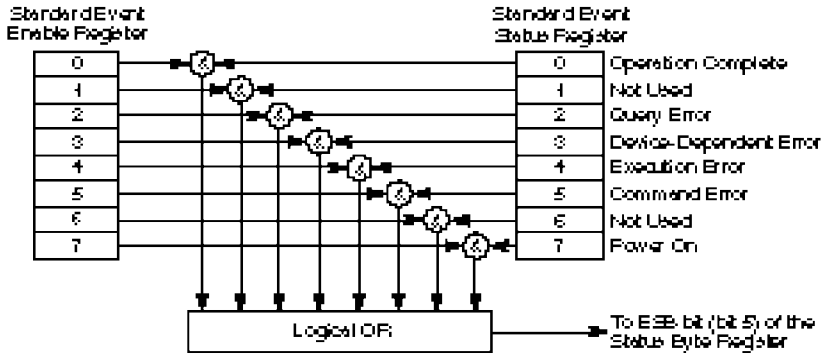


Figure 3-8. Standard Event Status Reporting

Standard Event Status Register

The Standard Event Status Register contains bits that monitor specific IEEE 488.2-defined events as shown in Figure 3-8.

Use *ESR? to read this register.

Use *ESR? or *CLS to clear this register.

Table 3-5 lists the Standard Event Status Register bits and briefly describes each bit.

Table 3-5. Standard Event Status Register

BIT	WEIGHT	SYMBOL	DESCRIPTION
0	1	OPC	Operation Complete
1		(RQC)	Not used because this instrument cannot request permission to become active IEEE 488.1 controller-in-charge.
2	4	QYE	Query Error
3	8	DDE	Device-Specific Error
4	16	EXE	Execution Error
5	32	CME	Command Error
6		(URQ)	Not used because this instrument does not define any local controls as “User Request” controls.
7	128	PON	Power On

A detailed description of each bit in the Standard Event Status Register follows:

- **Bit 0 (Operation Complete)** is an event bit which is generated in response to the *OPC command. This bit indicates that the Counter has completed all pending operations.

  If there are no pending operations at the time *OPC executes, this bit sets immediately.

  If there is a pending operation at the time *OPC executes, this event bit sets when the pending operation condition transistions from TRUE to FALSE.

  If

  the :TRIGger:COUNt:AUTO is OFF, or

  statistics are disabled, or

  the function is set to Totalize or Voltage Peaks,

  then the pending operation condition is set TRUE when either: 1) a single measurement is initiated, or 2) a continuous measurement cycle is initiated. The pending operation condition is set FALSE when the measurement cycle terminates.

  If

  the TRIGger:COUNt:AUTO is ON, and

  statistics are enabled, and

  the function is set to Totalize or Voltage Peaks,

  then the pending operation condition is set TRUE when a block of measurements is initiated. The pending operation condition is set FALSE when the block of measurements completes; that is, when the last measurement in the block completes.

  Bit 1 is not used.

? - **Bit 2 (Query Error)** is an event bit which indicates that either 1) an attempt was made to read the Output Queue when it was empty or 2) data in the Output Queue has been lost.

  Errors -400 through -499 are query errors.

- **Bit 3 (Device-Specific Error)** is an event bit which indicates an operation did not properly complete due to some condition of the Counter.

  Errors -300 through -399 and all those with positive error numbers (+2000 through ...) are device-specific errors.

- **Bit 4 (Execution Error)** is an event bit which indicates that a command could not be executed 1) because the parameter was out of range or inconsistent with the Counter's capabilities, or 2) because of some condition of the Counter.

  Errors -200 through -299 are execution errors.

- **Bit 5 (Command Error)** is an event bit which indicates one of the following has occurred: 1) an IEEE 488.2 syntax error, 2) a semantic error indicating an unrecognized command, or 3) a Group Execute Trigger was entered into the input buffer inside of a program message.

- **Bit 6** is not used.

- **Bit 7 (Power On)** is an event bit which indicates that an off-to-on transition has occurred in the Counter's power supply.

Standard Event Status Enable Register

The Standard Event Status Enable Register selects which events in the Standard Event Status Register are reflected in the ESB summary bit (bit 5) of the Status Byte Register as shown in
Figure 3-8.

Use *ESE to write to this register and *ESE? to read this register.

Use *ESE 0 to clear the register. (Power-on also clears this register.)

Operation Status Register Group and Questionable Data/Signal Status Register Group

The Operation Status Register Group and the Questionable Data/Signal Status Register Group each have a complete set of registers that consists of the following:

- a condition register
- a positive transition filter register
- a negative transition filter register
- an event register
- an event enable register

Figure 3-9 shows the model that these register groups follow.

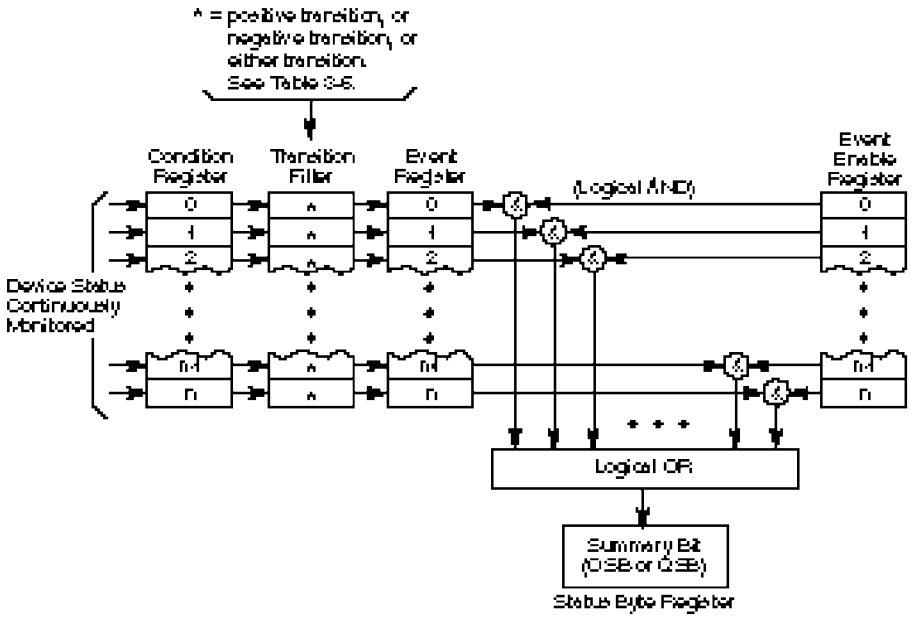


Figure 3-9. Operation and Questionable Status Reporting Model

Condition Register

A condition register continuously monitors the hardware and firmware status of the Counter. There is no latching or buffering for this register; it is updated in real time. Reading a condition register does not change its contents.

To read the condition registers use:

```
:STATus:OPERation:CONDition?
:STATus:QUEStionable:CONDition?
```

Transition Filter

A transition filter specifies the transition criteria for setting event bits TRUE.

When the transition filter specifies a positive transition, the event becomes TRUE when its associated condition makes a FALSE to TRUE transition only.

When the transition filter specifies a negative transition, the event becomes TRUE when its associated condition makes a TRUE to FALSE transition only.

When the transition filter specifies either a positive or a negative transition, the event becomes TRUE when its associated condition makes either a FALSE to TRUE or a TRUE to FALSE transition.

A transition filter is defined by a positive and negative transition filter register. Table 3-6 describes how the transition filter registers define the transition criteria for setting an event bit TRUE.

Table 3-6. Transition Filter Definition

Positive Transition Filter Bit	Negative Transition Filter Bit	Transition Which Causes the Event-Bit to be set TRUE
TRUE	FALSE	positive transition
FALSE	TRUE	negative transition
TRUE	TRUE	either a positive or negative transition
FALSE	FALSE	neither transition (event reporting is disabled)

Transition filters are unaffected by *CLS or queries. Transition filters are set to default values by :STATus:PRESet and power-on.

To write to the transitions filter registers use:

```
:STATus:OPERation:PTRansition
:STATus:OPERation:NTRansition
:STATus:QUEStionable:PTRansition
:STATus:QUEStionable:NTRansition
```

To read the transition filter registers use:

```
:STATus:OPERation:PTRansition?
:STATus:OPERation:NTRansition?
:STATus:QUEStionable:PTRansition?
:STATus:QUEStionable:NTRansition?
```

Event Register

An event register captures changes in conditions.

An event register bit (event bit) shall be set TRUE when an associated event occurs. These bits, once set, are “sticky.” That is, they cannot be cleared even if they do not reflect the current status of a related condition, until they are read.

To read the event registers use:

```
:STATus:OPERation[:EVENt]?
:STATus:QUEStionable[:EVENt]?
```

Use event register queries or *CLS to clear event registers.

Event Enable Register

An event enable register selects which event bits in the corresponding event register can generate a summary bit.

To write the event enable registers use:

```
:STATus:OPERation:ENABle
:STATus:QUEStionable:ENABle
```

To read the event enable registers use:

```
:STATus:OPERation:ENABle?
:STATus:QUEStionable:ENABle?
```

The event enable registers are cleared by :STATus:PRESet and power-on.

Operation Status Register Group

The Operation Status Register Group monitors conditions which are part of the Counter's normal operation.

Table 3-7 lists the Operation Status Register bits and briefly describes each bit.

Table 3-7. Operation Status Register

BIT	WEIGHT	DESCRIPTION
0	1	Calibrating
1		Not used
2		Not used
3		Not used
4	16	Measuring
5		Not used
6		Not used
7		Not used
8	256	Computing Statistics
9	512	Using Internal Reference
10	1024	In Limit Event
11–14		Not used
15		Not used since some controllers may have difficulty reading a 16-bit unsigned integer. The value of this bit shall always be 0.

A detailed description of each bit in the Operation Status Register follows:

- **Bit 0 (Calibrating)** is a condition bit which indicates the Counter is currently performing a (front-panel invoked or GPIB invoked) calibration.

  The condition bit is TRUE (one) during a calibration and FALSE (zero) otherwise.

- **Bits 1–3** are not used.

- **Bit 4 (Measuring)** is a condition bit which indicates the Counter is actively measuring.

  The condition bit is TRUE (one) during a measurement and FALSE (zero) otherwise.

  If the external reference has been explicitly selected and an absent or invalid signal at the external reference input is detected, then the Counter will not report Measuring (even though it may perform an auto trigger) in response to the user initiating a measurement.

- **Bits 5–7** are not used.

- **Bit 8 (Computing Statistics)** is a condition bit which indicates the Counter has begun collecting measurements for the next statistical computation.

  The condition bit is TRUE (one) once the first of N measurements has begun, and remains TRUE until the last of N measurements has completed.

- **Bit 9 (Using Internal Reference)** is a condition bit which indicates the Counter is using the internal reference.

  The condition bit is TRUE (one) while the Counter is using the internal reference. The condition bit is FALSE (zero) while the Counter is using the external reference.

  This bit monitors both explicit and automatic reference changes. Explicit reference changes occur when you select internal or external using the front-panel Utility menu or the GPIB command, [:SENS]:ROSC:SOUR. Automatic reference changes occur when the Counter is configured to select the reference (automatically) by detecting whether or not an external reference is being supplied.

- **Bit 10 (In Limit Event)** is an *event* bit indicating the last measurement limit tested was “in limit.”

  Each and every time a measurement is limit tested and found to be in limit, this event will be reported.

Note that this is the only bit in the Operation Status Register which is not representing a *condition*. Therefore, the transition filters have no effect on this bit.

The Counter does not monitor the condition indicating whether the last measurement was in or out of limit. Hence, the In Limit Event bit does NOT represent the transition from an "out of limit measurement" to "in limit measurement."

- **Bits 11–15** are not used.

Questionable Data/Signal Status Register Group

The Questionable Data/Signal Status Register Group monitors SCPI-defined conditions.

Table 3-8 lists the Questionable Data/Signal Status Register bits and briefly describes each bit.

Table 3-8. Questionable Data/Signal Status Register

BIT	WEIGHT	DESCRIPTION
0		Not used
1		Not used
2	4	Time (Time Interval, Rise Time, Fall Time, Pulse Width, Duty Cycle, and Period)
3		Not used
4		Not used
5	32	Frequency (only Frequency; not Frequency Ratio)
6	64	Phase
7		Not used
8	256	Calibration Error
9		Not used
10	1024	Out of Limit Event
11–13		Not used
14	16384	Command Warning
15		Not used since some controllers may have difficulty reading a 16-bit unsigned integer. The value of this bit shall always be 0.

A detailed description of each bit in the Questionable Data/Signal Status Register Group follows:

- **Bits 0–1** are not used.

- **Bit 2 (Time)** is a condition bit which indicates that the Time measurements (Period, Time Interval, Rise/Fall Time, Pulse Width, and Duty Cycle) may be affected by the disabling of automatic interpolator calibration.

  The condition bit is TRUE when automatic interpolator calibration is disabled. The condition bit is FALSE when automatic interpolator calibration is enabled.

- **Bits 3–4** are not used.

- **Bit 5 (Frequency)** is a condition bit which indicates that

  Frequency measurements (this does not include the Frequency Ratio measurements) may be affected by the disabling of automatic interpolator calibration.

  The condition bit is TRUE when automatic interpolator calibration is disabled. The condition bit is FALSE when automatic interpolator calibration is enabled.

- **Bit 6 (Phase)** is a condition bit which indicates that Phase measurements may be affected by the disabling of automatic interpolator calibration.

  The condition bit is TRUE when automatic interpolator calibration is disabled. The condition bit is FALSE when automatic interpolator calibration is enabled.

- **Bit 7** is not used.

- **Bit 8 (Calibration Error)** is an *event* bit which indicates that one of the following has occurred: 1) an GPIB invoked calibration failed, 2) a front-panel invoked calibration failed, 3) an automatic interpolator calibration failed during the measurement cycle, or 4) an automatic measurement calibration failed during the measurement cycle.

  Since this is an event bit, the transition filters have no effect on this bit.

- **Bit 9** is not used.

- **Bit 10 (Out of Limit Event)** is an *event* bit indicating the last measurement limit tested was “out of limit.”

Each and every time a measurement is limit tested and found to be out of limit, this event will be reported.

Note that this bit is not representing a *condition*. Therefore, the transition filters have no effect on this bit.

The Counter does not monitor the condition indicating whether the last measurement was in or out of limit. Hence, the Out of Limit Event bit does NOT represent the transition from an “in limit measurement” to “out of limit measurement.”

- **Bits 11–13** are not used.

- **Bit 14 (Command Warning)** is an *event* bit indicating a command, such as CONFigure or MEASure, ignored a parameter during execution.

  Since this is an event bit, the transition filters have no effect on this bit.

- **Bit 15** is not used.

Command Settings for Optimizing Throughput

This section lists the commands which enable the Counter to transfer data at the fastest possible rate. See the "To Optimize Throughput" sample programs on pages 3-73 , 3-86, and 3-96

Commands to Set Counter for Optimal Throughput

Unless otherwise noted, these settings are stored on Save (*SAV).

All of these settings are reset by *RST or a power cycle.

Disable auto trigger on measurement channel(s):

Specify absolute trigger levels—

```
[:SENSe]:EVENt[1|2]:LEVel[:ABSolute]   <numeric_value> [V]
```

or simply disable auto-trigger—

```
[:SENSe]:EVENt[1|2]:LEVel[:ABSolute]:AUTO OFF
```

Set gate/arm to auto for appropriate measurement:

For Frequency, Period, and Ratio—

```
[:SENSe]:FREQuency:ARM[:STARt]:SOURce  IMMediate
[:SENSe]:FREQuency:ARM:STOP:SOURce  IMMediate
```

For Phase—

```
[:SENSe]:PHASe:ARM[:STARt]:SOURce  IMMediate
```

For Time Interval (Agilent 53131A Only)—

```
[:SENSe]:TINTerval:ARM[:STARt]:SOURce  IMMediate
[:SENSe]:TINTerval:ARM:STOP:SOURce  IMMediate
```

For Time Interval (Agilent 53132A Only)—

```
[:SENSe]:TINTerval:ARM:ESTART:LAYer2:SOURce  IMMediate
[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce  IMMediate
[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce  IMMediate
[:SENSe]:TINTerval:ARM::ESTOP[:LAYer[1]]:SOURce  IMMediate
```

Define device trigger to FETC?:

When the device trigger is defined as FETC?, the Group Execute Trigger should be used to query for a result.

```
*DDT   #15FETC?
```

Set reference oscillator to non-auto state (internal or external):

[:SENSe]:ROSCillator:SOURce INTernal | EXTernal (See Note below.)

Disable checking of external source if using external reference oscillator:

[:SENSe]:ROSCillator:EXTernal:CHECk OFF (See Note below.)

Disable automatic interpolator calibration:

:DIAGnostic:CALibration:INTerpolator:AUTO OFF (See Note below.)

Disable display:

:DISPlay:ENABle OFF (See Note below.)

Disable printing:

:HCOPy:CONTinuous OFF

Disable post-processing (math, limit testing, statistics):

:CALCulate:MATH:STATe OFF

:CALCulate2:LIMit:STATe OFF

:CALCulate3:AVERage[:STATe] OFF

Specify expected frequency for Frequency, Period, Ratio, Phase, and Duty Cycle measurements:

[:SENSe]:FREQuency:EXPected[1|2|3] <numeric_value> [HZ]
(See Note below.)

Specify ASCII format for result query responses:

:FORMat[:DATA] ASCii

Specify continuous measurements:

:INITiate:CONTinuous ON

Configure the read/fetch function memory:

Issue the following query and read the response.

:FETCh[:SCALar]:<function>?

Note: This value is not stored on Save and is not part of the non-volatile state.

Typical Optimizing Throughput Results for Different Computers

Table 3-9 lists the typical performance for three different computers. The “To Optimize Throughput” sample programs on pages 3-73 , 3-86, and 3-96 were used to generate the numbers in the table. The actual examples listed in this guide show the Frequency Auto Arming function, but the technique is the same for the other two functions (Frequency Time Arming .001 and Time Interval 1 to 2). You only have to change the function in the program to generate the numbers in the table.

Table 3-9. Typical Optimizing Throughput Results in Measurements per Second

Function	IBM PC Compatible 486/25 MHz Agilent 82335A Card	Agilent 82324A Basic Language Processor	HP 9000 Series 300 Model 360
Time Interval 1 to 2 Auto Arming	235	225	230
Frequency Auto Arming	200	190	195
Frequency Time Arming .001	155	155	160

How to Program the Counter for Status Reporting

Determining the Condition of the Counter

The Counter has status registers that are used to indicate its condition. There are four register groups that can be examined individually, or used to alert a computer. These registers, shown in Figure 3-6, are:

- Operation Status Register Group
- Questionable Data/Signal Register Group
- Standard Event Status Register Group
- Status Byte Register Group

The first three groups all have event registers that can be fed into the Status Byte Register. The Status Byte Register can be used to assert the SRQ line of the GPIB and thus alert the computer that the Counter needs attention. The following examples show how each of the register groups can be used. (Figure 3-10 is a flowchart diagram of how to program the Counter for Status Reporting.)

Resetting the Counter and Clearing the GPIB Interface—Example 1

Before attempting any programming, it is a good idea to set the Counter to a known state . The following command grouping shows how to reset the Counter. Before issuing these commands, execute a device clear to reset the interface and Counter. Consult your interface card's documentation for how to issue a device clear since the device clear command will be specific to the interface you are using. Perform the following:

1. Issue a Device Clear (See your computer or interface card documentation for how to issue this command).
2. Issue the following commands:

```
*RST
*CLS
*SRE 0
*ESE 0
:STAT:PRES
```

Using the Standard Event Status Register to Trap an Incorrect GPIB command—Example 2

The following command grouping shows how to use the Standard Event Status Register and the Status Byte Register to alert the computer when an incorrect command is sent to the Counter.
The command *ESE 32 tells the Counter to summarize the command error bit (bit 5 of the Event Status Register) in the Status Byte Register. The command error bit is set when an incorrect command is received by the Counter. The command *SRE 32 tells the Counter to assert the SRQ line when the Event Status Register summary bit is set to 1. If the Counter is serial polled after a command error, the serial poll result will be 96.

Event Status Register

*ESE 32 Enable for bad command.

*SRE 32 Assert SRQ from Standard Event Status Register summary.

Using the Questionable Data/Signal Status Register to Alert the Computer When Automatic Interpolator Calibration is Disabled—Example 3

The default operation of the Counter is for automatic interpolator calibration to occur before every measurement. To optimize throughput over the GPIB, the automatic calibration can be disabled. When it is disabled, the most recent calibration values are used. These values may not be the optimal values for a particular temperature or other environmental condition. For this reason, the Time, Frequency and Phase bits in the Questionable Data register are set whenever the automatic calibration is disabled.

In the following Questionable Data Status Register example, the first line tells the Counter to detect a transition from negative (non-questionable data) to positive (questionable data) of bits 2, 5, and 6 in the Questionable Data Register. The next line tells the Counter to summarize the detected events in the Status Byte Register. The command *SRE 8 tells the Counter to assert the SRQ line when the summary bit for the Questionable Data register is set to 1. A serial poll will return the value 72 when the automatic calibration transitions from on to off.

Questionable Data Status Register

:STAT:QUES:PTR 100; NTR 0	Detect transition from non-questionable to questionable data.
:STAT:QUES:ENABLE 100	Enable to detect for auto cal off.
*SRE 8	Assert SRQ on Questionable Summary bit.

Using the Operation Status Register to Alert the Computer When Measuring has Completed— Example 4

The following command grouping illustrates how to use the Operation Status register and the Status Byte register to alert the computer when measuring has completed. This is useful if the Counter is making a long measurement. For example, a frequency measurement with a gate time of 10 seconds. When the measurement is complete, the Counter can alert the computer.

The first line tells the Counter to watch for a negative transition from true (measuring) to false (non-measuring) of bit 4. This negative transition indicates that the Counter has completed a measurement. The next line tells the Counter to summarize the detected event (bit 4 of the Operation Status Register) in the Status Byte Register. The command *SRE 128 tells the Counter to assert SRQ when the summary bit for the Operation Status register is set to 1. A serial poll will return 192 when a measurement has completed.

Operation Status Register

:STAT:OPER:PTR 0; NTR 16	Detect transition from measuring to non-measuring.
:STAT:OPER:ENABLE 16	Enable to detect measuring.
*SRE 128	Assert SRQ on Operation Summary bit.

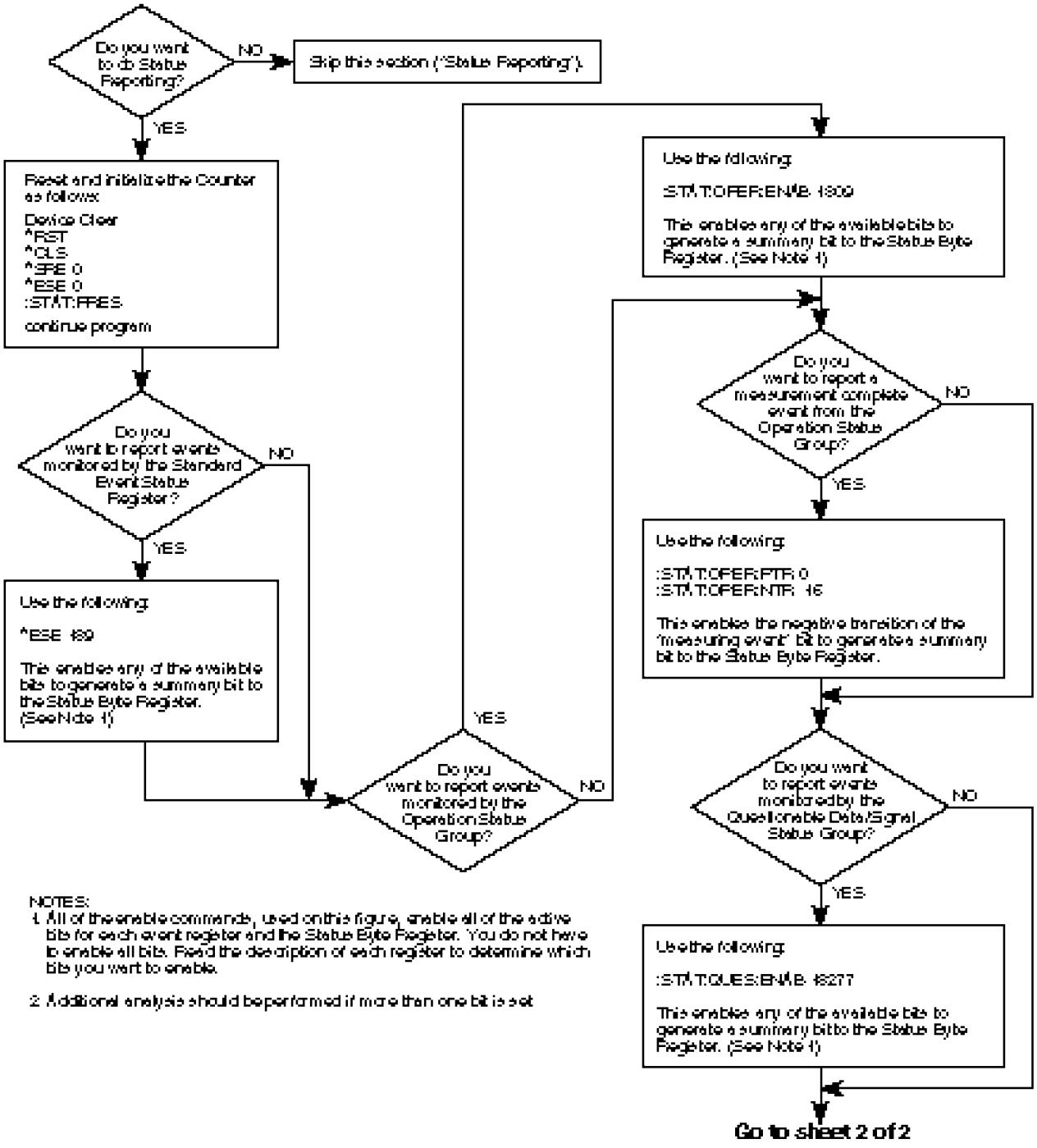


Figure 3-10. Status Reporting Flowchart (1 of 2)

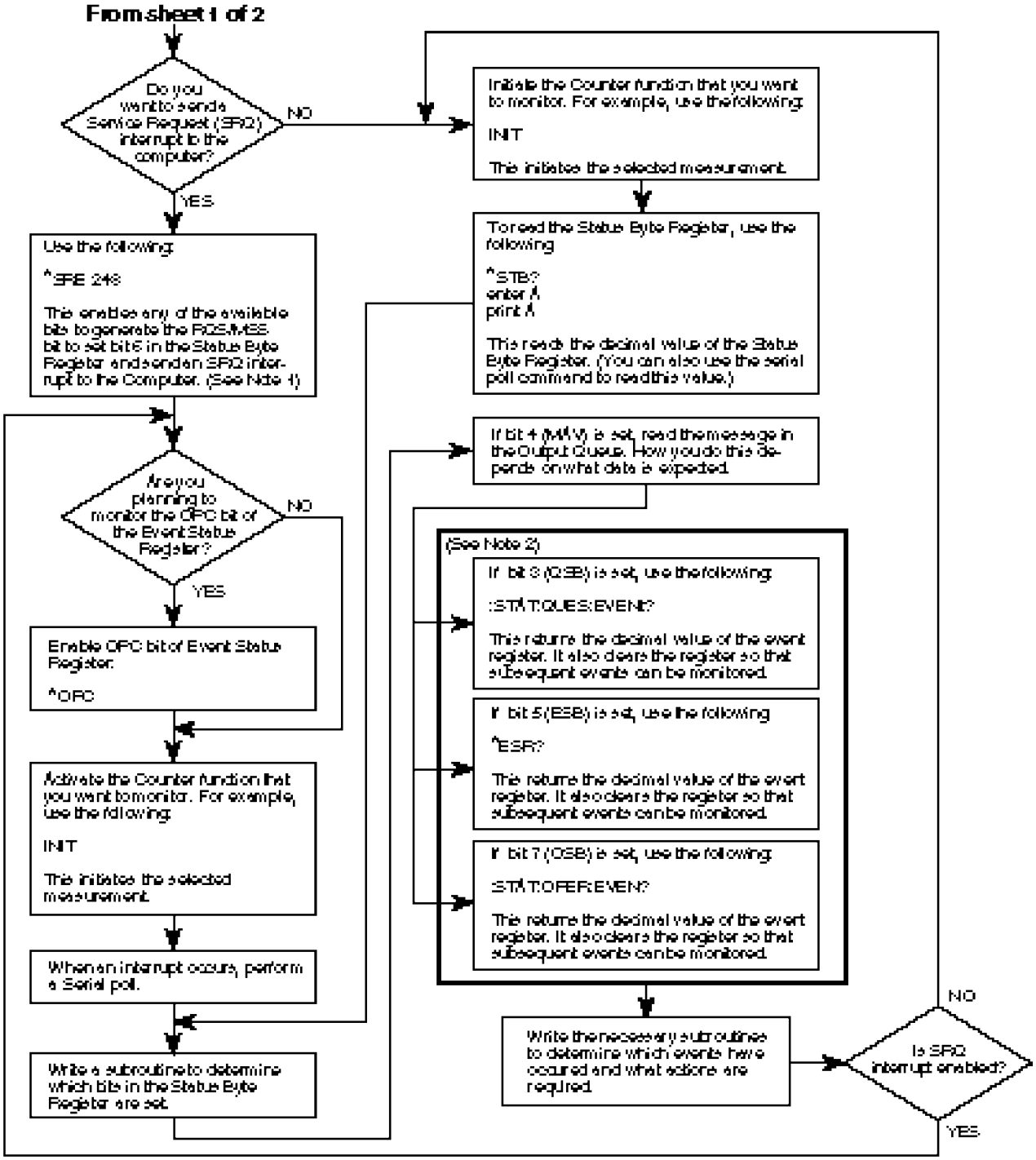


Figure 3-10. Status Reporting Flowchart (2 of 2)

How to Program the Counter to Display Results

Configuring the Counter's Display

The Counter has five different display modes:

1. Non-scaled/offset results—frequency, period, time interval, etc. This display mode is used on power-up.
2. Scaled/offset results—results modified by scale and offset values
3. Limit graph—a graphical look that shows if a measurement is within limits
4. Statistics—mean, min, max or standard deviation
5. Display Enable—All segments and LEDs (except Remote and SRQ) off or on

The following command groupings show how to program the Counter to any of the above display modes.

Commands for Displaying Non-Scaled/Offset Results

The following lines will always show the raw (non-scaled/offset) measurement result.

:DISP:MENU OFF	Clear any menu items that may be on display.
:DISP:TEXT:FEED 'CALC2'	Show the non-statistical result.
:CALC2:LIM:DISP NUMBER	Use the numeric display mode.
:CALC:MATH:STATE OFF	Disable math so scale and offset not used.
:CALC:IMM	Cause a calculation to be made to update display.

Note that :CALC2:LIM:DISP NUMBER will only show the raw result if the command :DISP:TEXT:FEED 'CALC2' is also issued. These commands must be issued in pairs.

Commands for Displaying Scaled/Offset Results

The following lines will enable Math (scale/offset). It is assumed that the values for scale and offset are already set. If not, the default value for scale is 1 and for offset is 0.

:DISP:MENU OFF	Clear any menu items that may be on display.
:DISP:TEXT:FEED 'CALC2'	Show the non-statistical result.
:CALC2:LIM:DISP NUMBER	Use the numeric display mode.
:CALC:MATH:STATE ON	Enable math.
:CALC:IMM	Cause a calculation to be made to update display.

Note that :CALC2:LIM:DISP NUMBER will only show the raw result if the command :DISP:TEXT:FEED 'CALC2' is also issued. These commands must be issued in pairs.

Commands for Displaying the Limit Graph

The following lines enable limit testing and show the limit graph.
If the Math is enabled, the scale and offset will be applied to the measurement result before being tested for the limit graph.

:DISP:MENU OFF	Clear any menu items that may be on display.
:DISP:TEXT:FEED 'CALC2'	Show the non-statistical result.
:CALC2:LIM:STATE ON	Enable limit testing.
:CALC2:LIM:DISP GRAPH	Display the limit test result graph.
:CALC:IMM	Cause a calculation to be made to update display.

Commands for Displaying Statistics Results

The following lines enable Statistics. The default value displayed is Mean.

:DISP:MENU OFF	Clear any menu items that may be on display.
:DISP:TEXT:FEED 'CALC3'	Show statistical results.
:CALC3:AVER ON	Enable statistics.

Commands for Enabling and Disabling the Display

The Counter display can be turned on or off. The normal condition is for the display to be on. To achieve maximum GPIB throughput, the display must be disabled.

:DISP:ENABLE OFF	Disable the display, all segments off.
:DISP:ENABLE ON	Normal display mode.

How to Program the Counter to Synchronize Measurements

Synchronizing Measurement Completion

The Counter has three different methods for synchronizing the end of a measurement and computer transfer of data. The three methods are:

1. Using the *WAI command
2. Using the *OPC? command
3. Using the *OPC command to assert SRQ

The following discussion shows how to use all three methods.

Resetting the Counter and Clearing the GPIB Interface

Before attempting any programming, it is a good idea to set the Counter to a known state. The following command grouping illustrates how to reset the Counter. Before issuing these commands, execute a device clear to reset the interface and Counter. Consult your interface card's documentation for how to issue a device clear since the device clear command will be specific to the interface you are using. Perform the following:

1. Issue a Device Clear. (See your computer or interface card documentation for how to issue this command.)
2. Issue the following commands:

```
*RST
*CLS
*SRE 0
*ESE 0
:STAT:PRES
```

Using the *WAI Command

This method is most useful when only the Counter is on the bus and you want the Counter to send the data when it is ready. In this example, the Counter is instructed to take 50 measurements and return the statistics for these 50 measurements. After the :INIT command is issued, the Counter is instructed to hold off execution of any more commands by the *WAI command. When the Counter has completed the 50 measurements and statistics, it executes the :CALC3:AVERAGE:ALL? command, which asks for the results.

:CALC3:AVERAGE ON	Enable statistics.
:CALC3:AVERAGE:COUNT 50	Base statistics on 50 measurements.
:TRIG:COUNT:AUTO ON	On INIT, take 50 measurements.
:INIT	Start 50 measurements.
*WAI	Wait until 50 measurements are complete before Counter executes another command. At this point, commands could be issued to other instruments.
:CALC3:AVERAGE:ALL?	Asks for the statistics. This command, will not be executed until the 50th measurement is complete.

Using the *OPC? Command

This method is useful if you want to hold off execution of the program while you wait for the Counter to complete any pending activity. In the *WAI example above, the line following the *WAI command is accepted by the Counter. However, the Counter does not execute the command because of the preceding *WAI command. If this line had been a command to address another instrument, it would be immediately executed. If you had wanted to hold off the command to another instrument, you would use the *OPC? command instead of the *WAI command.

:CALC3:AVERAGE ON	Enable statistics.
:CALC3:AVERAGE:COUNT 50	Base statistics on 50 measurements.
:TRIG:COUNT:AUTO ON	On INIT, take 50 measurements.
:INIT	Start making measurements.
*OPC?	Tell Counter to put a 1 in the output buffer when 50th measurement is complete.

Read the Counter. The program will wait here until the Counter returns a 1.

:CALC3:AVERAGE:ALL?	Ask for statistics.

Using the *OPC Command to Assert SRQ

This method is recommended when the Counter is on the GPIB with many other instruments, any of which can assert SRQ. The commands *OPC, *ESE 1 and *SRE 32 are used to assert the SRQ line to alert the computer that the Counter has completed a measurement. It is up to the computer to use the serial poll command to determine which of the instruments on the bus requested service.

Of the three procedures discussed here, this is the most flexible, but also the most complex.

:CALC3:AVERAGE ON	Enable statistics.
:CALC3:AVERAGE:COUNT 50	Base statistics on 50 measurements.
:TRIG:COUNT:AUTO ON	On INIT, take N measurements.
*ESE 1	Summarize OPC bit for Status Byte Register.
*SRE 32	SRQ when event summary bit is 1.

Set up program to specify service routine and enable interrupt when SRQ is asserted.

:INIT	Start measurements.
*OPC	Enable OPC bit.

Program could be doing other things while waiting for SRQ.

When SRQ occurs and the Counter has been identified as the cause of the SRQ, ask for the data:

:CALC3:AVERAGE:ALL?	Ask for statistics.

How to Program the Counter for Math/Limit Operations

Updating Math and Limit Results Over GPIB

When using the Limits or Math capabilities from the front panel, the default (power-up) operation is for results to be automatically updated whenever a value is updated in either the Limit or Scale&Offset menu. For example, entering a scale value automatically enables Math and updates the result in the display to reflect the changes. Similarly, entering either an upper or lower limit automatically enables Limit Testing. If, after entering a value in either of these menus, you do not want limits or math, you must go to the appropriate menu item and turn off limits or math.

When the Counter is programmed, there are additional issues that must be addressed.

The first thing done in most programs is to put the Counter in a known state using *RST. The *RST command resets the Counter. **One of the things this command does is disable the *automatic* post-processing of Limit and Math operations**. **What this means is, that if you set a limit, scale or offset value, and enable Limits or Math, the answer will not be *automatically* updated to reflect the Limit or Math values**. Whenever a *new* measurement is made, the result *will* be updated, but, if the Counter is in Single mode, changing the Limits or Math will not result in an automatic re-calculation.

There are two things you can do to make sure the results are updated. One is to send the command :CALC:IMM:AUTO ON after the *RST command. This will cause the results to be updated whenever a limit, scale or offset value is changed. The benefit with this command is that you only have to send this command once and the Counter will always return data that reflects the current limit or scale/offset settings.
One potential drawback is that results you may not care about can occur, possibly causing an unexpected event (like an SRQ or out-of-limit condition). For example, if you wanted to change the scale and offset, you might first send the scale value. With :CALC:IMM:AUTO ON, the scale value will be immediately applied, before the offset value is received. You may not care what this scaled-only value is,
but it may cause an out-of-limit condition, which may in turn cause an SRQ, neither of which you might have expected.

The other option is to program the Counter to update post-processed results only when you tell it to. This is accomplished by sending the command :CALC:IMM after you send all of the limits or scale/offset values. This way, no intermediate

results are calculated. The only drawback with this command is that you must always send it when you change the limits or scale/offset values.

The section in this chapter titled “How to Program the Counter to Display Results” uses the :CALC:IMM technique to make sure the results are properly displayed.

Using the Scale and Offset Over GPIB

Using the scale and offset values over the bus is different from setting any other value.

The commands for setting the scale and offset are in the TRACE subsystem. For example, if you make a frequency measurement and want to set the scale to 5.0 and the offset to 100 Hz, send the following commands:

:TRACE SCALE, 5	Set scale value.
:TRACE OFFSET, 100	Set offset value.

The above commands just set the values. To enable them, Math must be turned on, and the results processed as described in the previous section:

:CALC1:MATH:STATE ON	Enable math.
:CALC:IMM	Process results using scale and offset.

If you then wanted the to have the Counter’s display show the processed results, the following commands must be issued:

:DISP:MENU OFF	Clear any menu items that may be on display.
:DISP:TEXT:FEED ‘CALC2’	Show the non-statistical result.
:CALC2:LIM:DISP NUMBER	Use the numeric display mode.

If you need to query the scale and offset values, you need to know if you are in ASCII or REAL data format. The values returned from the following query will be sent using the format that is currently defined (:FORMat[:DATA]) in the box. To query the scale, use the following command:

```
:TRACE? SCALE
```

Then, enter the data, keeping in mind how it will be formatted (ASCII or REAL).

How to Program the Counter to Define Macros

A macro is a user defined command that can be used to replace one or many Counter commands. There are two good reasons to use macros in place of other commands:

1. They provide a mnemonic for long or complex commands.
2. They reduce the overhead associated with sending long commands.

For example, if you often want the Counter to display the limit graph, you can replace the following string of commands with a macro called 'limitresult' (you can provide any name you wish).

```
:DISP:MENU OFF;:DISP:TEXT:FEED 'CALC2';
:CALC2:LIM:STATE  ON;:CALC2:LIM:DISP GRAPH
```

Anytime you wanted to display the limit graph, you would just send the command 'limitresult'.

To assign the macro 'limitresult' to the above command sequence, you would send the following:

```
*DMC 'limitresult',#280:DISP:MENU OFF;
:DISP:TEXT:FEED 'CALC2';:CALC2:LIM:STATE  ON;
:CALC2:LIM:DISP GRAPH
```

The #2 indicates that the next two characters contain the length of the command, in this example, 80 characters. To program a macro, you need to know the length of characters in the command. This can be tedious and is prone to users counting incorrectly. The "To Use Macros" sample programs on pages 3-75 and 3-88, can be used to help set up macros and perform the counting for you.

A macro also lets you send variable parameters along with the name. For example, you could have a macro that sets up a measurement channel. One of the variables may be the input impedance, either 50 Ohms or 1 Megaohm. To program this, you would send the macro name along with the impedance value. To assign a variable inside the macro definition, you would replace the normal parameter with a $ followed by a number from 1 to 9. Up to 9 variables can be assigned. When sending the macro, the first parameter would be assigned to the $1 and all occurrences of $1 in the macro. The second parameter would be assigned to $2 and so on. Here is what the macro called ‘setimp’ would look like. It changes the impedance on channel 1 to the value assigned to $1 in the macro command.

```
*DMC ‘setimp’,#212:INP1:IMP $1
```

To change the impedance to 50 ohms, send:

```
setimp 50
```

The above is a very simple example. Macros are best used for a long sequence of commands. A good use for macros is changing the display from one format to another. To change to the limit graph, the following commands must be sent:

```
:DISP:MENU OFF
:DISP:TEXT:FEED ‘CALC2’
:CALC2:LIM:STATE ON
:CALC2:LIM:DISP GRAPH
:CALC:IMM
```

These commands can all be replaced by a macro called ‘limitdisplay’, defined as follows:

```
*DMC ‘limitdisplay’,#290:DISP:MENU OFF;
:DISP:TEXT:FEED ‘CALC2’;:CALC2:LIM:STATE ON;
:CALC2:LIM:DISP GRAPH;:CALC:IMM
```

There is a finite amount of memory available in the Counter for storing macros. If you find that you are running low on memory,
you can shorten the commands as follows:

1. Do not send the complete path unless it is necessary.
2. Use 1 and 0 instead of ON and OFF for <Boolean> parameters.
3. Use the short form for keywords, INP for INPut, FUNC for FUNCtion and so on.

The above example for switching to the limit display can be significantly decreased in length using these shortcuts:

```
*DMC 'limitdisplay',#268
:DISP:MENU 0;TEXT:FEED 'CALC2';:CALC2:LIM:STAT 1;DISP
GRAP;
:CALC:IMM
```

Programming examples using macros are provided in the following section titled "Programming Examples." The first macro program listing (starting on page 3-75) uses BASIC for an HP 9000 series 300 computer. The second Macro program listing (starting on page 3-88) is for an IBM PC (or clone) and Agilent 82335A/B card. Both are softkey driven and can be used to define macros, enable or disable macros, determine what macros are available and purge macros. There is little error trapping in the programs, if you misspell a command, the Counter will give an error message.

Writing SCPI Programs

Figure 3-11 is a general summation of how to write SCPI programs.
It shows a typical sequence you might go through in the process of writing a program. You do not have to follow this exact sequence,
but it will help you to become familiar with the Counter's capabilities and to direct you to sections of the guide which will be useful while writing programs.

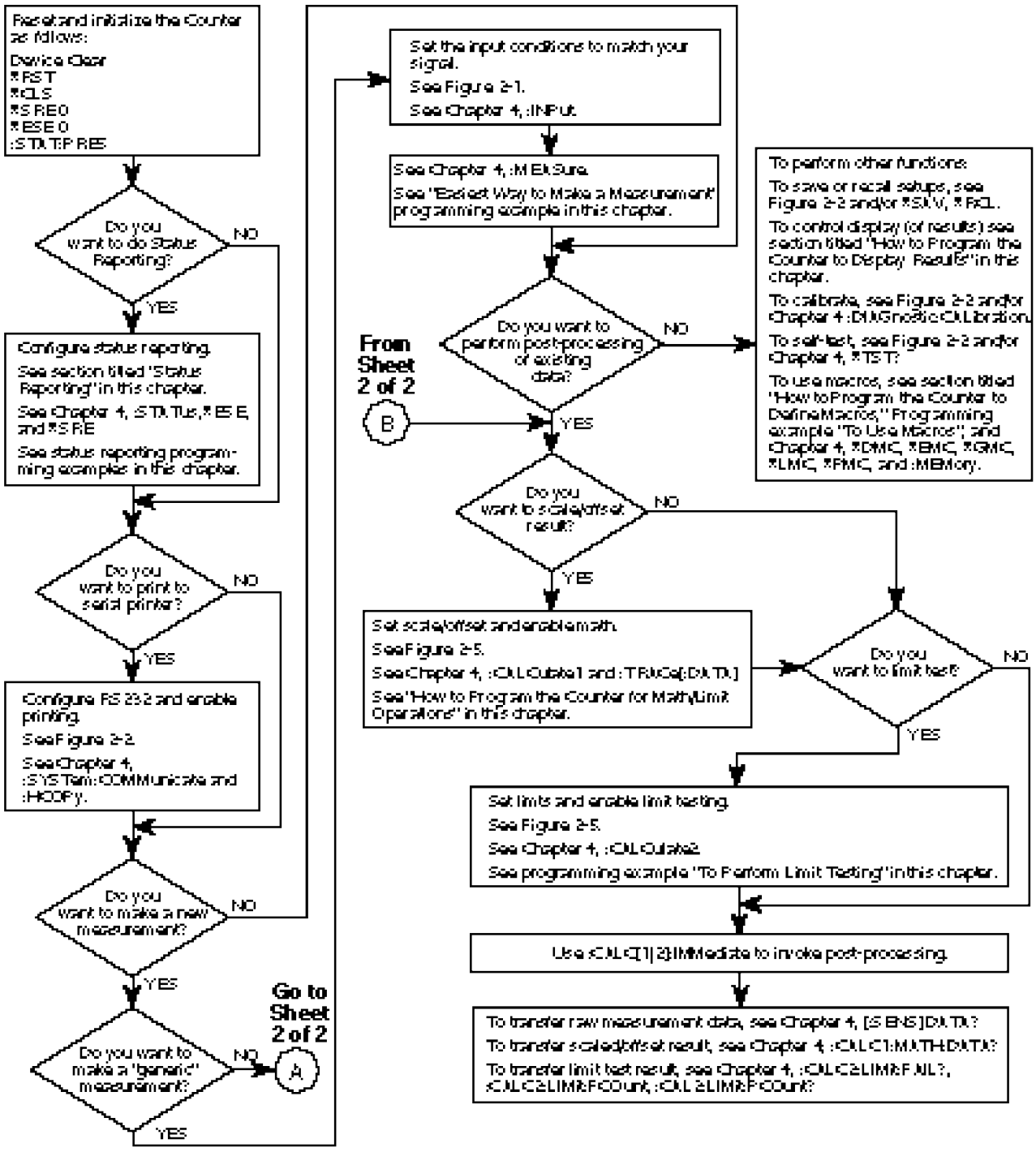


Figure 3-11. SCPI Programming Flowchart (Sheet 1 of 2)

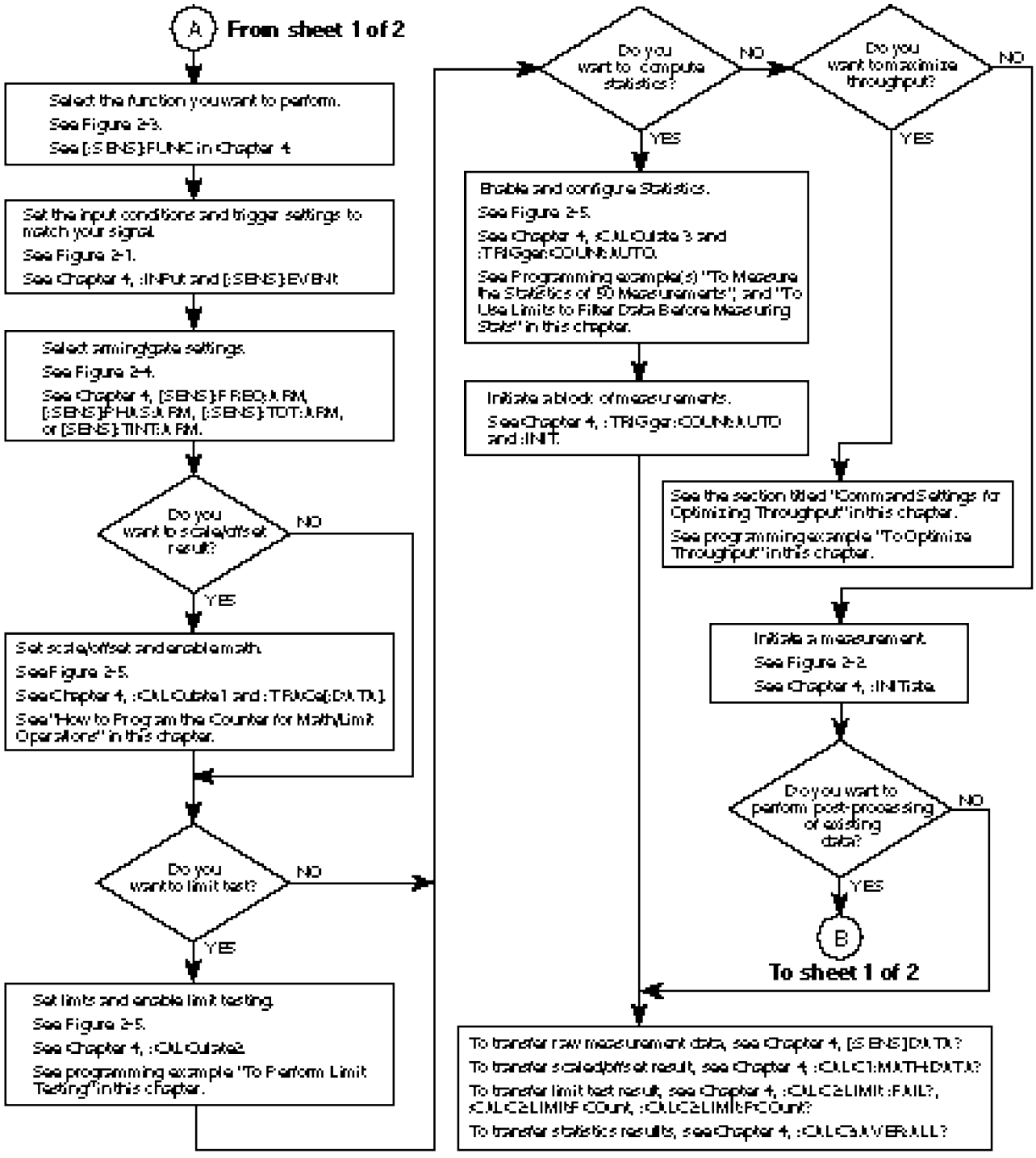


Figure 3-11. SCPI Programming Flowchart (Sheet 2 of 2)

Programming Examples

In this section, you will see how to program the Agilent 53131A/132A to make many common measurements. Examples are provided in the following programming languages:

- BASIC
- MicrosoftÒ QuickBASIC (version 4.5)*
- BorlandÒ Turbo C**

Using BASIC

This guide uses doubles quotes to enclose string parameters in syntax descriptions, but uses single quotes in the BASIC programming examples for readability.

The Counter allows string parameters to be enclosed by either double or single quotes. Each method is discussed in the following sub-sections.

To Send a Double-Quoted String

For the BASIC OUTPUT statements, remember that strings enclosed in double quotes need special consideration. For example, send the FUNC “FREQ 1” command with the following:

```
OUTPUT 703;"FUNC ""FREQ 1"""
```

Note that a pair of double quotes (as shown in bold) is required by BASIC to embed a double quote within an BASIC string.

To Send a Single-Quoted String

For more readable BASIC OUTPUT statements, you may send, for example, the following:

```
OUTPUT 703;"FUNC 'FREQ 1'"
```

Note the pair of single quotes (as shown in bold) is more readable.

---

*Microsoft is a U.S. registered trademark of Microsoft Corporation.

**Turbo C is a product of Borland International, Inc.

Using QuickBASIC

The QuickBASIC examples assume you have an Agilent 82335A GPIB Interface card inside your IBM PC or compatible.

Using Turbo C

The Turbo C examples assume you have an Agilent 82335A GPIB Interface card inside your IBM PC or compatible.

List of the Programming Examples

The following examples are provided:

1. Easiest Way to Make a Measurement (BASIC only)
2. To Make a Frequency Measurement
3. To Perform Limit Testing (BASIC and QuickBASIC only)
4. To Measure the Statistics of 50 Measurements (BASIC and QuickBASIC only)
5. To Use Limits to Filter Data Before Measuring Statistics.
6. To Read and Store Calibration Data—this program is useful if you plan to calibrate your Agilent 53131A/132A and want to be able to return to the original calibration at a later date.
   (BASIC and QuickBASIC only)
7. To Perform a Time Interval Calibration—this program shows how to perform both types of time interval calibrations.
   (The “fine” time interval calibration uses the Agilent 59992A J06 Time Interval Calibrator.) This program also shows how to use the basic calibration security capabilities. Note, early versions of the Counter cannot execute this program since they do not support calibration security or “fine” time interval calibration. (BASIC only)
8. To Optimize Throughput; that is, to set up the Counter to transfer data at the fastest possible rate.
9. To Use Macros

NOTE

All programming examples use the ASCII format to transfer data from the Counter to the computer. The ASCII format is the default format when *RST is used.

Easiest Way to Make a Measurement (BASIC)

```
10 ! This program shows how to use the MEASure group of instructions to
20 ! quickly and easily make any of the counter's measurements.
30 ! In this program, time interval, frequency and period  will be measured.
40 ! However, the MEASure group can make measurements using any of the other
50 ! counter functions.
60 ! The program is composed of three subroutines. The first uses only
70 ! the MEAS:TINT? (@1),(@2) command to make a time interval measurement.
80 ! The second subroutine uses CONF:FREQ and READ? to make a measurement.
90 ! The third uses CONF:FREQ, INIT and FETCH? to make a measurement.
100! The comments at the start of each subroutine explain the benefits of
110! each method.
120   INTEGER I                       ! Declare variables
130   DIM Freq$[22]                   ! Declare string to enter data
140   DIM Period$[22]                 ! Using strings to enter ASCII format
150   DIM Tint$[22]                   ! data yields results formatted to the
160                                   ! correct resolution. ASCII is the
170                                   ! default format for the counter.
180 ! The following commands reset the counter
190   ASSIGN @Count TO 703            ! Assign I/O path for counter
200   CLEAR 703                       ! Clear the counter and interface
210   OUTPUT @Count;"*RST"            ! Reset the counter
220   OUTPUT @Count;"*CLS"            ! Clear event registers and error queue
230   OUTPUT @Count;"*SRE 0"          ! Clear service request enable register
240   OUTPUT @Count;"*ESE 0"          ! Clear event status enable register
250   OUTPUT @Count;":STAT:PRES"      ! Preset enable registers and
260                                   ! transition filters for operation and
270                                   ! questionable status structures.
280   GOSUB Measure                   ! Call subroutines to make measurements
290   LINPUT "Press RETURN  for CONF and READ",A$
300   GOSUB Conf_read
310   LINPUT "Press RETURN for CONF, INIT and FETC",A$
320   GOSUB Init_fetc
330   STOP
340 !
350 !
360 Measure:!   Use the MEAS:TINT? (@1),(@2) command
370 ! The MEAS:TINT? (@1),(@2) query initiates a complete measurement
380 ! sequence. It configures the counter for a 2 channel time interval
390 ! measurement, starts the measurement and asks for the data. The MEAS
400 ! command is the simplest (and least flexible) way to make a measurement
410 ! and collect data.
420 ! Make sure there are signals at both counter inputs.
430   PRINT "Time Interval from 1 to 2 measured using MEAS:TINT? (@1),(@2)"
440   OUTPUT @Count;"MEAS:TINT? (@1),(@2)"  ! Configure for time interval
450                                         ! and query counter for results.
460   ENTER @Count;Tint$
470   PRINT
480   PRINT "Time Interval 1 to 2 = ";Tint$
490   PRINT
500   RETURN
510 !
520 !
530 Conf_read:!  Use CONF and READ? command
```

Easiest Way to Make a Measurement (BASIC) (Continued)

```
540 ! The following commands will measure the frequency on channel 1.
550 ! The MEAS? query can be broken down into CONF and READ? commands.
560 ! The CONF and READ? allow more flexibility than the MEAS? query.
570 ! CONF can be used to configure a measurement. Additional commands
580 ! can then be issued to fine tune the measurement setup. The READ?
590 ! command than reads the result. In the following example, a frequency
600 ! measurement is configured, then, as an example for changing the setup
610 ! created by the CONF command, the counter is programmed for a trigger
620 ! level of 50 mV. (The CONF command tells the counter to use the AUTO
630 ! trigger level mode) Finally, the data is read using the READ? command.
640   PRINT
650   PRINT "Frequency measured using CONF:FREQ (@1) and READ?"
660   OUTPUT @Count;"CONF:FREQ (@1)"    ! Configure for frequency measurement
670   OUTPUT @Count;":EVENT1:LEVEL .05" ! Set trigger level to 50 mV
680   OUTPUT @Count;"READ?"             ! Ask for data
690   ENTER @Count;Freq$
700   PRINT
710   PRINT "Frequency = ";Freq$
720   PRINT
730   RETURN
740 !
750 !
760 Init_fetc:! Use INIT and FETCH to read frequency and period
770 ! The READ? command can be broken down into INIT and FETCH?, providing
780 ! even more measurement flexibility. By using FETCH?, you can retrieve
790 ! results based on already acquired data. For example, period can be
800 ! derived from a frequency measurement, without a new acquisition.
810 ! The following example uses CONF to set up a frequency measurement.
820 ! The trigger level is then changed to -50 millivolts and an INIT is
830 ! performed, starting the measurement process. The data is read using
840 ! the FETCH:FREQUENCY? command. The period can then be read by sending
850 ! FETCH:PERIOD?, this time asking for the period.
860   PRINT
870   PRINT "Frequency and Period measured using CONF:FREQ (@1), INIT, FETCH?"
880   OUTPUT @Count;"CONF:FREQ (@1)"    ! Configure for frequency measurement
890   OUTPUT @Count;":EVENT1:LEVEL -.05"! Change trigger level
900   OUTPUT @Count;"INIT"              ! Start a measurement
910   OUTPUT @Count;"FETCH:FREQUENCY?"  ! Ask for frequency result
920   ENTER 703;Freq$
930   PRINT
940   PRINT "Frequency = ";Freq$
950   OUTPUT @Count;"FETCH:PERIOD?"     ! Ask for period result derived from
960                                     ! frequency measurement. Note that
970                                     ! another measurement was not made.
980   ENTER @Count;Period$
990   PRINT
1000  PRINT "Period = ";Period$
1010  PRINT
1020  RETURN
1030  END
```

To Make a Frequency Measurement (BASIC)

```
10 ! This program sets up the counter to make 10 frequency
20 ! measurements on channel 1, using a 0.1 second gate time.
30 ! The results are displayed on the computer CRT.
40 ! ASCII format is used to preserve resolution.
50 !
60    INTEGER I                          ! Declare variables
70    DIM Freq$(10)[22]                  ! Declare string to enter data
80                                       ! Using strings to enter ASCII format
90                                       ! data yields results formatted to the
100                                      ! correct resolution. ASCII is the
110                                      ! default format for the counter.
120   Samples=10                         ! Take 10 measurements
130    !
140   ASSIGN @Count TO 703               ! Assign I/O path for counter
150   CLEAR 703                          ! Clear the counter and interface
160   OUTPUT @Count;"*RST"               ! Reset the counter
170   OUTPUT @Count;"*CLS"               ! Clear event registers and error queue
180   OUTPUT @Count;"*SRE 0"             ! Clear service request enable register
190   OUTPUT @Count;"*ESE 0"             ! Clear event status enable register
200   OUTPUT @Count;":STAT:PRES"         ! Preset enable registers and
210                                      ! transition filters for operation and
220                                      ! questionable status structures.
230   OUTPUT @Count;":FUNC 'FREQ 1'"     ! Measure frequency on channel 1
240   OUTPUT @Count;":FREQ:ARM:STAR:SOUR IMM" ! These three lines enable
250   OUTPUT @Count;":FREQ:ARM:STOP:SOUR TIM" ! Using time arming, with a
260   OUTPUT @Count;":FREQ:ARM:STOP:TIM .100" ! 0.1 second gate time
270    !
280   CLEAR SCREEN                                ! Clear the computer display
290   FOR I=1 TO Samples                          ! Start making measurements
300     OUTPUT @Count;"READ:FREQ?"                ! Start a measurement and
310                                               ! fetch the data
320     ENTER @Count;Freq$(I)                     ! Enter the frequency
330     PRINT USING "11A,DD,4A,22A,3A";"Frequency (";I;") = ";Freq$(I);" Hz"
340   NEXT I
350   LOCAL 703                                   ! Return counter to local
360   END
```

To Perform Limit Testing (BASIC)

```
10 ! This program sets up the counter to make period measurements
20 ! indefinitely until an out of limits measurement occurs.
30 ! The upper limit is 1 usec and the lower limit is 500 nsec.
40 ! If a measurement falls outside of these limits, the counter will
50 ! stop measuring and report the out of limits value to the computer
60 ! ASCII format is used to preserve resolution.
70    Lower=5.00E-7                    ! Lower limit for period
80    Upper=1.E-6                      ! Upper limit for period
90    DIM Result$[22]                  ! Read out of limit period into Result$
100   ASSIGN @Count TO 703             ! Assign I/O path for the counter
110   CLEAR 703                        ! Clear the counter and interface
120   OUTPUT @Count;"*RST"             ! Reset the counter
130   OUTPUT @Count;"*CLS"             ! Clear event registers and error queue
140   OUTPUT @Count;"*SRE 0"           ! Clear service request enable register
150   OUTPUT @Count;"*ESE 0 "          ! Clear event status enable register
160   OUTPUT @Count;":STAT:PRES"       ! Preset enable registers and
170                                    ! transition filters for Operation and
180                                    ! Questionable status structures
190   OUTPUT @Count;":FUNC ""PER 1"""  ! Measure period on channel 1
200    ! Note that the function selected must be a quoted string.
210    ! The actual string sent to the counter is "PER 1".
220    !
230   OUTPUT @Count;":FREQ:ARM:STAR:SOUR IMM" ! These two lines enable
240   OUTPUT @Count;":FREQ:ARM:STOP:SOUR IMM" ! Automatic arming.
250    !
260   OUTPUT @Count;":CALC2:LIM:STAT ON"      ! Enable limit testing
270   OUTPUT @Count;":CALC2:LIM:DISP GRAP"    ! Show the analog limit graph
280   OUTPUT @Count;":CALC2:LIM:LOWER ";Lower ! Set lower limit to 500 ns
290   OUTPUT @Count;":CALC2:LIM:UPPER ";Upper ! Set upper limit to 1 us
300   OUTPUT @Count;":INIT:AUTO ON"           ! Stop when out of limit
310   OUTPUT @Count;":STAT:QUES:ENAB 1024"    ! 1024 is out of limit bit
320   OUTPUT @Count;"*SRE 8"                  ! Enable SRQ on questionable
330                                           ! data register event
340   ON INTR 7 GOTO Out_of_limits            ! If SRQ goto out_of_limits
350   ENABLE INTR 7;2                         ! Enable the interrupt
360   OUTPUT @Count;":INIT:CONT ON"           ! Start making measurements
370 Loop_here:GOTO Loop_here                  ! Loop while in limits
380   !
390 Out_of_limits:                            ! Here because SRQ line
400   Status_byte=SPOLL(703)                  ! asserted Serial poll counter
410   OUTPUT @Count;"FETCH:PERIOD?"           ! Query the counter
420   ENTER @Count;Result$                    ! Read the period
430   PRINT "Out of limits measurement is ";Result$;" seconds"
440   PRINT "Status byte is ";Status_byte     ! Should be 72 (64+8)
450   LOCAL 703                               ! Return counter to local
460   END
```

To Measure the Statistics of 50 Measurements (BASIC)

```
10    ! This program instructs the counter to take 50 period measurements.
20    ! The counter is put into SINGLE measurement mode. The number of
30    ! measurements to take is programmed using ":CALC3:AVER:COUNT 50"
40    ! The counter is told to stop after 50 measurements using
50    ! ":TRIG:COUNT:AUTO ON"
60    ! At the end of 50 measurements, the statistics are calculated and
70    ! sent to the computer.
80    ! ASCII format is used to preserve resolution.
90    ! In this example, the status reporting structure is used to alert
100   ! the program that the statistics are ready.
110   ! The "*OPC" command and the "*ESE 1 " command are used together
120   ! to generate an output from the Event Status Register when
130   ! the measurement is complete. The output of this register is
140   ! used as an input to the Service Request Register. In order for the
150   ! Service Request Register to be able to use that input, the "*SRE 32"
160   ! command must be used. This enables the Service Request Register to
170   ! assert the SRQ line when the measurement is complete.
180   ! Note, that the *OPC command must be sent prior to every measurement
190   ! in order to enable the OPC bit.  See Line # 520.
200   INTEGER I,Num_meas
210   DIM Sdev$[22],Mean$[22],Minimum$[22],Maximum$[22]
220   Num_meas=50             ! Statistics based on Num_meas measurements
230   ASSIGN @Count TO 703
240   CLEAR 703                 ! Clear the counter and interface
250   OUTPUT @Count;"*RST"      ! Reset the counter
260   OUTPUT @Count;"*CLS"      ! Clear event registers and error queue
270   OUTPUT @Count;"*SRE 0 "   ! Clear service request enable register
280   OUTPUT @Count;"*ESE 0 "   ! Clear event status enable register
290   OUTPUT @Count;":STAT:PRES" ! Preset enable registers and transition
300                             ! filters for operation and questionable
310                             ! status structures.
320   OUTPUT @Count;":FUNC 'PER 1'"  ! Measure Period on channel 1.
330                                  ! Note that the functions must be
340                                  ! a quoted string. The actual
350                                  ! string sent to the counter
360                                  ! is 'PER 1'.
370   OUTPUT @Count;":FREQ:ARM:STAR:SOUR IMM" ! These three lines enable
380   OUTPUT @Count;":FREQ:ARM:STOP:SOUR TIM" ! time arming with a 0.01
390   OUTPUT @Count;":FREQ:ARM:STOP:TIM .01"  ! second gate time.
400   OUTPUT @Count;":DISP:TEXT:FEED 'CALC3'" ! Display statistics
410   OUTPUT @Count;":CALC3:AVER:TYPE SDEV"   !Display the standard deviation
420   OUTPUT @Count;":CALC3:AVER ON"              ! Enable statistics
430   OUTPUT @Count;":CALC3:AVER:COUNT ";Num_meas ! Do statistics on num_meas
440                                               ! measurements.
450   OUTPUT @Count;":TRIG:COUNT:AUTO ON "        !Take Num_meas measurements
460   OUTPUT @Count;"*ESE 1"      ! "*ESE 1" is used so bit 5
470                               !of the service request register will allow
480   OUTPUT @Count;"*SRE 32"     ! an SRQ when measurement complete.
490   ON INTR 7 GOTO Get_averages ! Goto Get_averages on interrupt.
500   ENABLE INTR 7;2             ! Enable interrupt on counter SRQ.
510   PRINT "Waiting for measurement to complete"
```

To Measure the Statistics of 50 Measurements (BASIC) (Continued)

```
520   OUTPUT @Count;":INIT;*OPC"  ! Enable OPC bit and starts measurement
530 Loop_here:GOTO Loop_here      ! Wait here until measurement complete.
540   !
550 Get_averages:                 ! Data ready
560   Serial_poll=SPOLL(703)
570   OUTPUT @Count;":CALC3:AVERAGE:TYPE MAX;:CALC3:DATA?"
580   ENTER @Count;Maximum$
590   OUTPUT @Count;":CALC3:AVERAGE:TYPE MIN;:CALC3:DATA?"
600   ENTER @Count;Minimum$
610   OUTPUT @Count;":CALC3:AVERAGE:TYPE MEAN;:CALC3:DATA?"
620   ENTER @Count;Mean$
630   OUTPUT @Count;":CALC3:AVERAGE:TYPE SDEV;:CALC3:DATA?"
640   ENTER @Count;Sdev$
650   PRINT
660   PRINT "Serial Poll = ";Serial_poll          ! Should be 96
670   PRINT
680   PRINT USING "21A,22A,X,8A";"Minimum Period     = ";Minimum$;" seconds"
690   PRINT USING "21A,22A,X,8A";"Maximum Period     = ";Maximum$;" seconds"
700   PRINT USING "21A,22A,X,8A";"Mean Period        = ";Mean$;" seconds"
710   PRINT USING "21A,22A,X,8A";"Standard Deviation = ";Sdev$;" seconds"
720   LOCAL 703
730   END
```

To Use Limits to Filter Data Before Measuring Stats (BASIC)

```
10    ! This program instructs the counter to determine the statistics of
20    ! 50 Period measurements that are within the limits defined by the
30    ! variables "Upper" and "Lower". Periods that are outside of
40    ! the limits are not included in the statistics. The Limit graph is
50    ! displayed so you can see if measurements are in limit.
60    ! To alert the program that the statistics are ready, bit 8 in the
70    ! Operation Status register is used. When statistics are being
80    ! calculated, this bit is high, when they are complete, the bit goes
90    ! low. By using the transition filters, an SRQ can be generated when
100   ! statistics are complete.
110   ! ASCII format is used to preserve resolution.
120   !
130   INTEGER I,Num_meas
140   DIM Sdev$[22],Mean$[22],Minimum$[22],Maximum$[22]
150   Num_meas=50             ! Statistics based on num_meas measurements
160   Upper=1.10E-7           ! Upper period to be included in statistics
170   Lower=1.00E-7           ! Lower period to be included in statistics
180   CLEAR SCREEN
190   ASSIGN @Count TO 703
200   CLEAR 703                 ! Clear the counter and interface
210   OUTPUT @Count;"*RST"      ! Reset the counter
220   OUTPUT @Count;"*CLS"      ! Clear event registers and error queue
230   OUTPUT @Count;"*SRE 0"    ! Clear service request enable register
240   OUTPUT @Count;"*ESE 0"    ! Clear event status enable register
250   OUTPUT @Count;":STAT:PRES" ! Preset enable registers and transition
260                              ! filters for operation and questionable
270                              ! status structures.
280   OUTPUT @Count;":FUNC 'PER'"          ! Measure frequency on channel 1
290                                        ! Note that the function must be
300                                        ! a quoted string. The actual
310                                        ! string sent to the counter
320                                        ! is "PER".
330   OUTPUT @Count;":FREQ:ARM:STAR:SOUR IMM"  ! These three lines enable
340   OUTPUT @Count;":FREQ:ARM:STOP:SOUR TIM"  ! time arming with a 0.01
350   OUTPUT @Count;":FREQ:ARM:STOP:TIM .01"   ! second gate time.
360   OUTPUT @Count;":STAT:OPER:ENABLE 256"    ! Computing Statistics bit in
370                                            ! Operation status register
380   OUTPUT @Count;":STAT:OPER:NTR 256"  ! When statistics are complete,
390   OUTPUT @Count;":STAT:OPER:PTR 0"    ! the bit will go from high to low
400                                       ! so a negative transition is
410                                       ! needed to enable the bit that
420                                       ! is summarized in the Status Byte
430                                       ! Register.
440   OUTPUT @Count;"*SRE 128"            ! This is the bit from the
450                                       ! Operation Status register that is
460                                       ! summarized in the Status Byte
470                                       ! Register.
480                                       ! When it goes high, SRQ will be
490                                       ! asserted.
```

To Use Limits to Filter Data Before Measuring Stats (BASIC) (Continued)

```
500   OUTPUT @Count;":CALC3:LFIL:STATE ON"       ! Enable statistics filter
510   OUTPUT @Count;":CALC3:LFIL:LOWER ";Lower   ! Set the lower limit
520   OUTPUT @Count;":CALC3:LFIL:UPPER ";Upper   ! Set the upper limit
530   OUTPUT @Count;":CALC3:AVER ON"             ! Enable statistics
540   OUTPUT @Count;":CALC3:AVER:COUNT ";Num_meas! Set number of
550                                              ! measurements for stats.
560   OUTPUT @Count;":CALC2:LIM:STATE ON"        ! Enable limit testing. This
570                                              ! must happen in order to
580                                              ! see the limit graph.
590   OUTPUT @Count;":CALC2:LIM:LOWER ";Lower    ! Set the limits lower limit
600   OUTPUT @Count;":CALC2:LIM:UPPER ";Upper    ! Set the limits upper limit
610   OUTPUT @Count;":CALC2:LIM:DISP GRAPH"      ! Display the limit graph
620   !
630   ON INTR 7 GOTO Stats_ready        ! Where to go when statistics ready
640   ENABLE INTR 7;2                   ! Enable interrupt on SRQ
650   PRINT "Waiting for measurement to complete"
660   OUTPUT @Count;":INIT:CONT ON"     ! Set counter to RUN
670 Loop_here:  !WAITING FOR STATISTICS TO COMPLETE
680   GOTO Loop_here
690             !
700 Stats_ready:            !Statistics are ready
710   S=SPOLL(703)          ! Serial poll to see if correct bit is set.
720   OUTPUT @Count;":INIT:CONT OFF"    ! Stop making new measurements
730   OUTPUT @Count;":CALC3:AVERAGE:TYPE MAX;:CALC3:DATA?"
740   ENTER @Count;Maximum$
750   OUTPUT @Count;":CALC3:AVERAGE:TYPE MIN;:CALC3:DATA?"
760   ENTER @Count;Minimum$
770   OUTPUT @Count;":CALC3:AVERAGE:TYPE MEAN;:CALC3:DATA?"
780   ENTER @Count;Mean$
790   OUTPUT @Count;":CALC3:AVERAGE:TYPE SDEV;:CALC3:DATA?"
800   ENTER @Count;Sdev$
810   PRINT
820   PRINT "Serial Poll Result = ";S    ! Should be 192
830   PRINT
840   PRINT USING "21A,22A,X,8A";"Minimum Period     = ";Minimum$;" seconds"
850   PRINT USING "21A,22A,X,8A";"Maximum Period     = ";Maximum$;" seconds"
860   PRINT USING "21A,22A,X,8A";"Mean Period        = ";Mean$;" seconds"
870   PRINT USING "21A,22A,X,8A";"Standard Deviation = ";Sdev$;" seconds"
880   LOCAL 703                         ! Put counter is local
890   END
```

To Read and Store Calibration Information (BASIC)

```
10 !This program reads the calibration data for the counter into an array.
20 !Before calibrating the counter, it is a good idea to read
30 !and store the current values in case something goes wrong with the
40 !calibration.
50 !In this program, the calibration values are stored in the array cal_data.
60 !Normally, you would store the calibration data on a disk for safe
70 !keeping. The calibration values should only be changed by running the
80 ! calibration diagnostics.
90 !
100   DIM Cal_data$[57]              ! Array to hold calibration data
110   DIM Err_string$[255]           ! Array to hold error message
120   CLEAR SCREEN
130   ASSIGN @Count TO 703           ! Assign I/O path for Agilent 53131A
140   CLEAR @Count
150   OUTPUT @Count;"*RST"           ! Reset the Agilent 53131A
160   OUTPUT @Count;"*CLS"           ! Clear event registers and error queue
170   OUTPUT @Count;"*SRE 0"         ! Clear service request enable register
180   OUTPUT @Count;"*ESE 0"         ! Clear event status enable register
190   OUTPUT @Count;":STAT:PRES"     ! Preset enable registers and
200                                  ! transition filters for operation and
210                                  ! questionable status structures.
220   OUTPUT @Count;":CAL:DATA?"     ! Ask for data
230   ENTER @Count USING "#,4A";Head1$
240   ENTER @Count USING "%,K";Cal_data$
250   PRINT "Calibration data now in array Cal_data"
260   ! You may want to store Cal_data$ and Head1$ on a disk.
270   ! If, at some later point, you need to send the calibration data
280   ! back to the counter, you would use the following command:
290   ! OUTPUT @Count;":CAL:DATA ";Head1$&Cal_data$  ! Send calibration data
300   ! REPEAT
310   !   OUTPUT @Count;"SYST:ERR?"
320   !   ENTER @Count;Err_num,Err_string$
330   !   IF Err_num<>0 THEN
340   !     PRINT Err_num,Err_string$
350   !   END IF
360   ! UNTIL Err_num=0
370  END
```

To Perform a Time Interval Calibration (BASIC)

NOTE Early versions of the Counter cannot execute this program since they do not support calibration security or “fine” time interval calibration.

```
10   ! This program shows you how to perform a TIME INTERVAL calibration
20   ! using GPIB.   There are two types of Time Interval calibration:
30   ! a quick calibration, requiring a simple calibration signal, and
40   ! a FINE calibration, requiring a more complex signal--but producing
50   ! a better result.  Both calibrations are coded below.
60   !   When you perform a calibration, the additional steps required
70   ! include: storing a backup copy of old calibration data, using
80   ! the security system to UNSECURE the counter prior to calibration,
90   ! using the security system to SECURE the counter after calibration.
100  !   These tasks are coded below.
110  !
120  ! PROGRAM SHOWS:
130  !   FINE Time Interval Calibration
140  !   QUICK Time Interval Calibration
150  !
160  !   Storing current calibration values before recalibrating--as a backup
170  !   Restoring the backup calibration values
180  !
190  !   SECURING and UNSECURING the counter
200  !
210  !
220   DIM Cal_data$[57]               ! Array to hold calibration data
230   DIM Err_string$[255]            ! Array to hold error message
240   CLEAR SCREEN
250   ASSIGN @Count TO 703            ! Assign I/O path for Counter
260   ASSIGN @Cal TO 708              ! Assign I/O path for Calibrator
270                                   ! used only for the FINE calibration
280   !
290   GOSUB Init
300   GOSUB Store_cal
310   GOSUB Unsecure
320   !
330   INPUT "Which calibration: QUICK or FINE?",Answer$
340   IF Answer$[1,1]="f" OR Answer$[1,1]="F" THEN
350     GOSUB Cal_fine
360   ELSE
370     GOSUB Cal_quick
380   END IF
390   !
400   GOSUB Restore_cal
410   GOSUB Resecure
420   GOTO End
430 Init: !
```

To Perform a Time Interval Calibration (BASIC) (Continued)

```
440   CLEAR @Count
450   OUTPUT @Count;"*RST"            ! Reset the Agilent 53131A
460   OUTPUT @Count;"*CLS"            ! Clear event registers and error queue
470   OUTPUT @Count;"*SRE 0"          ! Clear service request enable register
480   OUTPUT @Count;"*ESE 0"          ! Clear event status enable register
490   OUTPUT @Count;":STAT:PRES"      ! Preset enable registers and
500                                   ! transition filters for operation and
510                                   ! questionable status structures.
520   RETURN
530   !
540 Unsecure: ! Routine UNSECURES the device
550   OUTPUT @Count;":CAL:SEC:STAT?"   ! Query the current security status
560   ENTER @Count;Device_secure       ! Check: SECURED=1, UNSECURED=0
570   WHILE Device_secure=1            ! If device is SECURED
580     INPUT "Enter security code to unsecure counter.",Code$
590     OUTPUT @Count;":CAL:SEC:STAT OFF,";Code$  ! UNSECURE the device
600     OUTPUT @Count;":CAL:SEC:STAT?"            ! verify it is SECURED
610     ENTER @Count;Device_secure
620     IF Device_secure=1 THEN                 ! On operator error, loop
630       PRINT "Wrong code, try again."
640       GOSUB Read_errors
650     END IF
660   END WHILE
670   RETURN
680   !
690 Resecure: ! Routine SECURES the device
700   OUTPUT @Count;":CAL:SEC:STAT?"   ! Query the current security status
710   ENTER @Count;Device_secure       ! Check: SECURED=1, UNSECURED=0
720   WHILE Device_secure=0            ! If device is UNSECURED
730     INPUT "Enter security code to secure counter.",Code$
740     OUTPUT @Count;":CAL:SEC:STAT ON,";Code$  ! SECURE the device
750     OUTPUT @Count;":CAL:SEC:STAT?"           ! verify it is SECURED
760     ENTER @Count;Device_secure
770     IF Device_secure=0 THEN                  ! On operator error, loop
780       PRINT "Wrong code, try again."
790       GOSUB Read_errors
800     END IF
810   END WHILE
820   RETURN
830   !
840 Store_cal: ! Routine collects the calibration data
850   OUTPUT @Count;":CAL:DATA?"        Ask for calibration data
860   ENTER @Count USING "#,4A";Head1$ ! Collect the data.  Store Head1$ and
870   ENTER @Count USING "%,K";Cal_data$ ! Cal_data$ on a disk as your backup
880   RETURN
890   !
900   !
```

To Perform a Time Interval Calibration (BASIC) (Continued)

```
910 Cal_quick: ! Calibrates using the QUICK TI calibration.
920   PRINT "Connect square wave signal to Channel 1"
930   GOSUB Wait_for_input
940   PRINT "Recalibrating"
950   OUTPUT @Count;"DIAG:CAL:TINT:QUICK"
960   OUTPUT @Count;"DIAG:CAL:STATUS?"
970   ENTER @Count;Status
980   IF Status<>0 THEN
990     PRINT "Quick TI calibration failed."
1000  END IF
1010  RETURN
1020  !
1030  !
1040 Cal_fine:! Calibrates using the FINE TI calibration.
1050          ! Code below includes the commands required to drive
1060          ! the Agilent 59992 JO6 Time Interval Calibrator.
1070  PRINT "Connect 10-MHz signal to calibrator back-panel input."
1080  PRINT "Connect calibrator Channel A,B outputs to 53131 Channel 1,2"
1090  GOSUB Wait_for_input
1100  PRINT "Recalibrating"
1110  OUTPUT @Cal;"B1"                          ! B1 = Button 1 on calibrator
1120  OUTPUT @Count;":DIAG:CAL:TINT:FINE1"  ! Counter processes B1 signal.
1130  OUTPUT @Count;":DIAG:CAL:STATUS?"     ! Query status only to force
1140                                        ! program to wait until prior
1150                                        ! command has completed.
1160  ENTER @Count;Status                   ! Response comes back when box
1170                                        ! is ready for next CAL command.
1180       !
1190  OUTPUT @Cal;"B2"                          ! B2 = Button 2 on calibrator
1200  OUTPUT @Count;":DIAG:CAL:TINT:FINE2"  ! Counter processes B2 signal.
1210  OUTPUT @Count;":DIAG:CAL:STATUS?"     ! Query status only to wait
1220  ENTER @Count;Status
1230       !
1240  OUTPUT @Cal;"B3"                          ! B3 = Button 3 on calibrator
1250  OUTPUT @Count;":DIAG:CAL:TINT:FINE3"  ! Counter processes B3 signal.
1260  OUTPUT @Count;":DIAG:CAL:STATUS?"     ! Query status only to wait
1270  ENTER @Count;Status
1280       !
1290  OUTPUT @Cal;"B4"                          ! B4 = Button 4 on calibrator.
1300  OUTPUT @Count;":DIAG:CAL:TINT:FINE4"  ! Counter processes B4 signal.
1310  OUTPUT @Count;":DIAG:CAL:STATUS?"     ! NOW query status to assess
1320  ENTER @Count;Status                   !   PASS/FAIL status of the
1330       !                                !   calibration.
1340                                        !   0=PASS, 1=FAIL
1350  IF Status<>0 THEN
1360    PRINT "Fine TI calibration failed."
1370  END IF
1380  RETURN
1390  !
```

To Perform a Time Interval Calibration (BASIC) (Continued)

```
1400 Restore_cal:! Restores the calibration data previously saved.
1410  IF Status=0 THEN
1420    INPUT "Calibration successful, restore backup anyway?",Answer$
1430  ELSE
1440    INPUT "Calibration unsuccessful, restore backup?",Answer$
1450  END IF
1460  IF Answer$[1,1]="y" OR Answer$[1,1]="Y" THEN
1470    PRINT "Restoring saved data"
1480    OUTPUT @Count;":CAL:DATA ";Head1$&Cal_data$! Send calibration data
1490  END IF
1500  GOSUB Read_errors
1510  RETURN
1520  !
1530 Wait_for_input:!
1540  INPUT "Input signal ready?",Answer$
1550  WHILE Answer$[1,1]<>"Y" AND Answer$[1,1]<>"y"
1560    INPUT "Input signal ready?",Answer$
1570  END WHILE
1580  RETURN
1590  !
1600 Read_errors: !
1610  REPEAT
1620    OUTPUT @Count;"SYST:ERR?"
1630    ENTER @Count;Err_num,Err_string$
1640    IF Err_num<>0 THEN
1650      PRINT Err_num,Err_string$
1660    END IF
1670  UNTIL Err_num=0
1680  RETURN
1690 End:  !
1700  END
```

To Optimize Throughput (BASIC)

```
10    ! This program shows how to set up the counter to transfer data at the
20    ! fastest possible rate. Note that the arming mode is AUTO. This mode
30    ! provides the least resolution of all arming modes.
40    ! The program comments discuss the meaning of each command.
50    ! ASCII result format is to preserve resolution.
60    !
70    CLEAR SCREEN
80    INTEGER I
90    DIM A$(200)[22],Dummy$[22]
100   ASSIGN @Count TO 703
110   CLEAR 703                     ! Clear the counter and interface
120   OUTPUT @Count;"*RST"          ! Reset the counter
130   OUTPUT @Count;"*CLS"          ! Clear event registers and error queue
140   OUTPUT @Count;"*SRE 0"        ! Clear service request enable register
150   OUTPUT @Count;"*ESE 0"        ! Clear event status enable register
160   OUTPUT @Count;":STAT:PRES"    ! Preset enable register and transition
170                                 ! filters for operation and questionable
180                                 ! status structures.
190   ! The following lines will provide the highest throughput, regardless
200   ! of the state of the counter before these lines are executed.
210   OUTPUT @Count;":FORMAT ASCII"  ! ASCII format for fastest throughput
220   OUTPUT @Count;":FUNC 'FREQ 1'" ! Select frequency
230   OUTPUT @Count;":EVENT1:LEVEL 0"     ! Set Ch 1 trigger level to 0 volts
240   OUTPUT @Count;":FREQ:ARM:STAR:SOUR IMM" ! These two lines enable the
250   OUTPUT @Count;":FREQ:ARM:STOP:SOUR IMM" ! AUTO arming mode.
260   OUTPUT @Count;":ROSC:SOUR INT"          ! Use internal oscillator. If
270                                           ! you want to use an external
280                                           ! timebase, you must select it
290                                           ! and turn off the automatic
300                                           ! detection using:
310                                           ! :ROSC:EXT:CHECK OFF
320                                           !
330   OUTPUT @Count;":DIAG:CAL:INT:AUTO OFF" ! Disable automatic interpolater
340                                         ! calibration. The most recent
350                                         ! calibration values are used in
360                                         ! the calculation of frequency
370   OUTPUT @Count;":DISP:ENAB OFF"        ! Turn off the counter display
380                                         ! This greatly increases
390                                         ! measurement throughput.
400   OUTPUT @Count;":CALC:MATH:STATE OFF"  ! Disable any post processing.
410   OUTPUT @Count;":CALC2:LIM:STATE OFF"
420   OUTPUT @Count;":CALC3:AVER:STATE OFF"
430   OUTPUT @Count;":HCOPY:CONT OFF"       ! Disable any printing operation
440   OUTPUT @Count;"*DDT #15FETC?"         ! Define the Trigger command
450                                         ! This means the command FETC?
460                                         ! does not need to be sent for
470                                         ! every measurement, decreasing
480                                         ! the number of bytes
490                                         ! transferred over the bus.
```

To Optimize Throughput (BASIC) (Continued)

```
500   OUTPUT @Count;":INIT:CONT ON"          ! Put counter in Run mode
510   OUTPUT @Count;"FETCH:FREQ?"            ! Fetch the frequency to be used
520   ENTER @Count USING "#,K";Dummy$        ! for the expected frequency.
530   OUTPUT @Count;":FREQ:EXP1 ";VAL(Dummy$)!Tell the counter what frequency
540                                          ! to expect on Ch 1. This number
550                                          ! must be within 10% of the input
560                                          ! frequency. Using this greatly
570                                          ! increases throughput. When
580                                          ! high throughput is not needed,
590                                          ! the expected value is not
600                                          ! required.
610   FOR I=1 TO 200
620     TRIGGER @Count                       ! Trigger the counter and read
630     ENTER @Count;A$(I)
640   NEXT I
650   FOR I=1 TO 10                          ! Print first 10 measurements
660     PRINT A$(I),
670   NEXT I
680   END
```

To Use Macros (BASIC)

```
10    USER 1 KEYS
20    ON KEY 1 LABEL " Macro   Free  ",1 CALL Macro_free
30    ON KEY 2 LABEL " Enable  Macros",1 CALL Macro_enable
40    ON KEY 3 LABEL " Display Macros",1 CALL Display_macros
50    ON KEY 4 LABEL " Macro   Query",1 CALL Macro_query
60    ON KEY 5 LABEL " Define  Macro",1 CALL Define_macro
70    ON KEY 6 LABEL " Delete  Macro",1 CALL Delete_macro
80    ON KEY 7 LABEL " Send    Macro",1 CALL Send_macros
90    ON KEY 8 LABEL " Disable Macros",1 CALL Disable_macro
100 Loop_h:GOTO Loop_h
110   END
120   SUB Macro_free                  ! Display memory available for macros.
130     OUTPUT 703;":MEM:FREE:MACRO?"
140     ENTER 703;Macro_free
150     DISP "Macro memory free = ";Macro_free
160     LOCAL 703
170   SUBEND
180   SUB Macro_enable                ! Enable macros. Default is disabled
190     OUTPUT 703;"*EMC 1"
200     DISP "Macros Enabled!"
210     LOCAL 703
220   SUBEND
230   SUB Disable_macro               ! Disable macros.
240     OUTPUT 703;"*EMC 0"
250     DISP "Macros Disabled!"
260   SUBEND
270   SUB Display_macros              ! Display available macros.
280     CLEAR SCREEN
290     DIM Macros$[6500]
300     OUTPUT 703;"*LMC?"
310     ENTER 703;Macros$
320     PRINT
330     PRINT "The following macros are available:"
340     PRINT
350     PRINT Macros$
360   SUBEND
370   SUB Send_macros                 ! Send a macro command to the counter.
380     CLEAR SCREEN                  ! A list of macros to choose from is
390     CALL Display_macros           ! shown on the computer.
400     DIM Name$[25],Macro$[200],Send$[255]
410     LINPUT "Enter the name of the macro",Name$
420     IF Name$="" THEN SUBEXIT
430     OUTPUT 703;"*GMC? "&CHR$(39)&Name$&CHR$(39)
440     ENTER 703;Macro$
450     PRINT
460     PRINT "Macro ";Name$;" is defined as follows:"
470     PRINT
480     PRINT Macro$
490     LINPUT "Enter the macro name and commands to be sent",Send$
500     OUTPUT 703;Send$
510   SUBEND
```

To Use Macros (BASIC) (Continued)

```
520   SUB Define_macro                ! Define a macro for the counter
530     DIM Name$[25],Macro$[200],Send$[255],Header$[2]
540     CLEAR SCREEN
550     LINPUT "Enter the name of the macro",Name$
560     LINPUT "Enter the counter commands",Macro$
570     Length=LEN(Macro$)
580     Num_char=INT(LGT(Length))+1   ! Determine # of characters for header
590     Header$="#"&VAL$(Num_char)
600     Send$="*DMC "&CHR$(39)&Name$&CHR$(39)&","&Header$&VAL$(Length)&Macro$
610     OUTPUT 703;Send$
620   SUBEND
630   SUB Macro_query                 ! Ask for the definition of a macro.
640     DIM Name$[25],Macro$[255]
650     CLEAR SCREEN
660     CALL Display_macros
670     LINPUT "Enter the name of the macro you want to see",Name$
680     IF Name$="" THEN SUBEXIT
690     OUTPUT 703;"*GMC? "&CHR$(39)&Name$&CHR$(39)
700     ENTER 703;Macro$
710     PRINT
720     PRINT "Macro ";Name$;" is defined as follows:"
730     PRINT
740     PRINT Macro$[(VAL(Macro$[2,2])+3)] ! Display command portion of macro
750   SUBEND
760   SUB Delete_macro                ! Delete a macro.
770     DIM Name$[25]
780     CALL Display_macros
790     LINPUT "Enter the name of the macro you want to delete",Name$
800     IF Name$="" THEN SUBEXIT
810     OUTPUT 703;":MEM:DELETE:MACRO ";CHR$(39)&Name$&CHR$(39)
820   SUBEND
```

To Make a Frequency Measurement (QuickBASIC)

```
'This program sets up the counter to make 10 frequency measurements
'on channel 1 using a 0.1 second gate time.
'The results are printed on the computer CRT.
'Data is sent in ASCII format to preseve resolution.
'
'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'                    'Required by Agilent 82335A
DIM SHARED source AS LONG                      'Address and select code
DIM i AS INTEGER                               'i is used for loops
DIM samples AS INTEGER
samples = 10                                   'Number of measurements
DIM freqs(10) AS STRING * 23                   'String to be read
                                               'Reading ASCII formatted data
                                               'gives results to the correct
                                               'resolution. Must be read into
                                               'a string. The maximum number
                                               'of characters that can ever be
                                               'sent is 20 per measurement.
source& = 703                                  'Counter at address 3
isc& = 7                                       'Select code 7
state% = 1                                     'Used in IOEOI

CLS                                            'Clear screen
CALL IOEOI(isc&, state%)              'Make sure EOI enabled
CALL IOCLEAR(source&)                 'Clear the counter and interface
CALL sendhp("*RST")                   'Reset counter
CALL sendhp("*CLS")                   'Clear event registers and error queue
CALL sendhp("*SRE 0")                 'Clear service request enable register
CALL sendhp("*ESE 0")                 'Clear event status enable register
CALL sendhp(":STAT:PRES")             'Preset enable registers and transition
                                      'filters for operation and questionable
                                      'status structures
CALL sendhp(":func " + CHR$(34) + "FREQ 1" + CHR$(34)) 'Measure frequency
CALL sendhp(":FREQ:ARM:STAR:SOUR IMM")    'These 3 lines enable using
CALL sendhp(":FREQ:ARM:STOP:SOUR TIM")    'time arming with a 0.1 second
CALL sendhp(":FREQ:ARM:STOP:TIM .1")      'gate time
CLS                                       'Clear computer screen
FOR i = 1 TO samples
CALL sendhp("READ:FREQ?")                    'Initiate a measurement and
                                             'get the result
CALL IOENTERS(source&, freqs(i), 23, actf%)  'Read the ASCII characters
PRINT "Frequency"; i; "= "; freqs(i)
NEXT i

END

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Perform Limit Testing (QuickBASIC)

```
'This program sets up the counter to make period measurements
'indefinitely until an out of limits measurement occurs. The upper
'limit is set to 1 us and the lower limit is set to 500 ns.
'If a measurement falls outside of these limits, the counter will
'stop measuring and send the out of limits period to the computer.
'The out of limit period is sent in ASCII format to preserve resolution.
'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'                'Required by Agilent 82335A
DIM SHARED source AS LONG                  'Address and select code

DIM period AS STRING * 23                  'Period string, the maximum number
                                           'of characters that can ever be
                                           'sent is 23
DIM complete AS INTEGER                    '
DIM statusbyte AS INTEGER                  'Status byte variable
upper = .000001                            'Upper period
lower = .0000005                           'lower period
source& = 703                              'Counter at address 3
isc& = 7                                   'Select code 7
complete = 0                               'Used to check if stats received
state% = 1                                 'Used in IOEOI
priority% = 1                              'Used in IOPEN
CLS                                        'Clear the screen
CALL IOEOI(isc&, state%)          'Make sure EOI enabled
CALL IOCLEAR(source&)             'Clear the counter and interface
CALL sendhp("*RST")               'Reset counter
CALL sendhp("*CLS")               'clear event registers and error queue
CALL sendhp("*SRE 0")             'clear service request enable register
CALL sendhp("*ESE 0")             'clear event status enable registers
CALL sendhp(":STAT:PRES")         'preset filters for operation and
                                  'questionable status structures
CALL sendhp(":FUNC " + CHR$(34) + "PER 1" + CHR$(34)) 'Measure period
'The function must be a quoted string. The actual string sent to the
'counter is "PER 1"

CALL sendhp(":FREQ:ARM:STAR:SOUR IMM")    'These 2 lines enable using
CALL sendhp(":FREQ:ARM:STOP:SOUR IMM")    'automatic arming

CALL sendhp(":CALC2:LIM:STAT ON")         'Enable limit testing
CALL sendhp(":CALC2:LIM:DISP GRAP")       'Show the analog limit graph
CALL sendhp(":CALC2:LIM:LOWER " + STR$(lower)) 'Set lower limit
CALL sendhp(":CALC2:LIM:UPPER " + STR$(upper)) 'Set upper limit
CALL sendhp(":INIT:AUTO ON")              'Stop when out of limit
CALL sendhp("*SRE 8")                     'Enable SRQ on questionable data
                                          'register event
CALL sendhp(":STAT:QUES:ENAB 1024")       '1024 is out of limit bit
```

To Perform Limit Testing (QuickBASIC) (Continued)

```
ON PEN GOSUB limitfail                    'When SRQ happens, go get out of
PEN ON                                    'limit result
CALL IOPEN(isc&, priority%)

CALL sendhp(":INIT:CONT ON")              'Set counter to run

PRINT "Making Period measurements"

Loophere:                                 'Wait here until out of limit
IF complete THEN GOTO endprogram          'If already serviced out of limit
GOTO Loophere                             'then end program
limitfail:
complete = 1                              'Test bit
CALL IOSPOLL(source&, statusbyte)         'Check status byte. Should be 72
PRINT "Status byte = ", statusbyte
CALL sendhp("FETCH:PERIOD?")              'Fetch the out of limits period
CALL IOENTERS(source&, period, 23, actf%) 'Read the out of limit period
PRINT "Out of limits period is ", period  'Print results
RETURN

endprogram:
END

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Measure the Statistics of 50 Measurements (QuickBASIC)

```
'This program instructs the counter to take 50 period measurements
'and return the mean, minimum, maximum and standard deviation.
'The counter is put into SINGLE measurement mode.
'The number of measurements is programmed using ":CALC3:AVER:COUNT 50"
'The counter is set up to take 50 measurements and then stop
'using the ":TRIG:COUNT:AUTO ON" command.
'At the end of the 50 measurements, the statistics are sent to the
'computer. The data is sent in ASCII format to preserve resolution.
'
'When the program has completed, the statistics will be displayed on
'the computer and the standard deviation will be displayed on the
'counter
'
'In this example, the status reporting structure is used to alert the
'program that the statistics are ready.
'The "*OPC" and "*ESE 1" command are used together to generate an output
'from the Event Status Register when the measurement is complete. The
'output of this register is summarized in the Status Byte Register
'In order for the Service Request Register to summarize that input
'the "*SRE 128" command must be used. This enables the Service
'Request Register to assert the SRQ line when the measurement is complete.
'Note that the *OPC command must be sent at the start of every measurement.
'
'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'            'Required by Agilent 82335A
DIM SHARED source AS LONG              'Address and select code
DIM samples AS INTEGER                 'Number of measurements
DIM maximum AS STRING * 23             'Strings for statistics
DIM minimum AS STRING * 23             'The maximum number of characters that
DIM mean AS STRING * 23                'can ever be sent is 23
DIM sdev AS STRING * 23
maxelem% = 23                          'Maximum number of characters expected
actual% = 0                            'Returns actual characters received
samples = 50                           'Number of statistics measurements
source = 703                           'Counter at address 3
isc& = 7                               'Select code 7
state% = 1                             'Used in IOEOI
priority% = 1                          'Used in IOPEN

CLS
CALL IOEOI(isc&, state%)             'Make sure EOI enabled
CALL IOCLEAR(source&)                'Reset the counter and interface
CALL sendhp("*RST")                  'Reset the counter
CALL sendhp("*CLS")                  'Clear event registers and error queue
CALL sendhp("*SRE 0")                'Clear service request enable register
CALL sendhp("*ESE 0")                'Clear event status enable register
CALL sendhp(":STAT:PRES")            'Preset enable registers and transition
                                     'filters for operation and questionable
                                     'status structures.
```

To Measure the Statistics of 50 Measurements (QuickBASIC) (Continued)

```
CALL sendhp(":FUNC " + CHR$(34) + "PER 1" + CHR$(34)) 'Measure Period

'The function must be a quoted string. The actual string sent to the
'counter is "PER 1"

CALL sendhp(":FREQ:ARM:STAR:SOUR IMM")      'These 3 lines enable using
CALL sendhp(":FREQ:ARM:STOP:SOUR TIM")      'time arming with a 0.01 second
CALL sendhp(":FREQ:ARM:STOP:TIM .01")       'gate time

CALL sendhp(":DISP:TEXT:FEED " + CHR$(34) + "CALC3" + CHR$(34)) 'Display stats
CALL sendhp(":CALC3:AVER:TYPE SDEV")        'Display the standard deviation
CALL sendhp(":CALC3:AVER ON")               'Enable statistics
CALL sendhp(":CALC3:AVER:COUNT " + STR$(samples)) 'Do stats on samples
CALL sendhp(":TRIG:COUNT:AUTO ON")          'Take samples measurements
CALL sendhp("*ESE 1")                       '"*ESE 1" is used so the
                                            'correct bit is summarized
CALL sendhp("*SRE 32")                      'in the Status Byte Register
                                            'when the measurement is complete
PRINT "Waiting for measurement to complete"
ON PEN GOSUB statsready                     'Wait for interrupt
PEN ON
CALL IOPEN(isc&, priorty%)

CALL sendhp(":INIT;*OPC")                   'Enable OPC bit and start meas

loophere:
IF complete THEN GOTO endprogram            'Wait here, if already made
GOTO loophere                               'stats measurements, then goto
                                            'endprogram.

statsready:                                 'Ready to read statistics

CALL sendhp(":CALC3:AVERAGE:TYPE MIN;:CALC3:DATA?")  'Read them individually
CALL IOENTERS(source&, minimum, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE MAX;:CALC3:DATA?")
CALL IOENTERS(source&, maximum, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE MEAN;:CALC3:DATA?")
CALL IOENTERS(source&, mean, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE SDEV;:CALC3:DATA?")
CALL IOENTERS(source&, sdev, maxelem%, actual%)
PRINT
PRINT "Minimum Period     = ", minimum
PRINT "Maximum Period     = ", maximum
PRINT "Mean Period        =  ", mean
PRINT "Standard Deviation = ", sdev
complete = 1
RETURN

endprogram:                                 'All done!

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Use Limits to Filter Data Before Measuring Stats (QuickBASIC)

```
'This program sets up the counter to determine the statistics of
'50 period measurements that are within limits defined by the variables
''UPPER' and 'LOWER'. Periods that are outside of the limits are not
'included in the statistics. The Limit graph is displayed so you can see if
'measurements are in limit.
'To alert the program that the statistics are ready, bit 8 in the Operation
'Status register is used. When statistics are being calculated, this bit
'is high, when they are complete, the bit goes low. By using the transition
'filters, an SRQ can be generated when statistics are complete.

'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'              'Required by Agilent 82335A
DIM SHARED source AS LONG                'Address and select code
DIM status AS INTEGER                    'Status byte variable
DIM complete AS INTEGER                  'Variable used in the program
DIM statusbyte AS INTEGER                'Status Byte variable
DIM maximum AS STRING * 23               'Strings used to enter stats
DIM minimum AS STRING * 23
DIM mean AS STRING * 23
DIM sdev AS STRING * 23
DIM nummeas AS INTEGER                   'Number of measurements
DIM lower AS SINGLE                      'Lower limit
DIM upper AS SINGLE                      'Upper limit
nummeas = 50                             'Number of statistics measurements
lower = .0000005                         'Limit values
upper = .000001
actual% = 0                              'Used in IOENTERS
maxelem% = 23                            'Used in IOENTERS
source& = 703                            'Counter at address 3
isc& = 7                                 'Select code 7
complete = 0                             'Used to check if stats received
state% = 1                               'Used in IOEOI
priority% = 1                            'Used in IOPEN
CLS
CALL IOEOI(isc&, state%)        'Make sure EOI enabled
CALL IOCLEAR(source&)           'Reset counter and interface
CALL sendhp("*RST")             'Reset counter
CALL sendhp("*CLS")             'Clear event registers and error queue
CALL sendhp("*SRE 0")           'Clear service request enable register
CALL sendhp("*ESE 0")           'Clear event status enable registers
CALL sendhp(":STAT:PRES")       'Preset filters for Operation and
                                'Questionable Status structures
CALL sendhp(":FUNC " + CHR$(34) + "PER 1" + CHR$(34)) 'Measure period
'The function must be a quoted string. The actual string sent to the
'counter is "PER 1"
```

To Use Limits to Filter Data Before Measuring Stats (QuickBASIC) (Continued)

```
CALL sendhp(":FREQ:ARM:STAR:SOUR IMM")          'These 3 lines enable time
CALL sendhp(":FREQ:ARM:STOP:SOUR TIM")          'arming with a 0.01 second
CALL sendhp(":FREQ:ARM:STOP:TIM .01")           'gate time.

CALL sendhp(":STAT:OPER:ENABLE 256")            'Computing statistics bit in
                                                'Operation Status Register.

CALL sendhp(":STAT:OPER:NTR 256")       'When stats are complete, the bit
CALL sendhp(":STAT:OPER:PTR 0")         'will go from high to low, so a
                                        'negative transition is needed to
                                        'enable the bit that is summarized
                                        'in the Status Byte Register.
CALL sendhp("*SRE 128")                 'This is the bit from the Operation
                                        'Status register that is summarized
                                        'in the Status Byte Register. When
                                        'it goes high, SRQ will be asserted.

CALL sendhp(":CALC3:LFIL:STATE ON")             'Enable statistics filter
CALL sendhp(":CALC3:LFIL:LOWER " + STR$(lower)) 'Set lower stats limit
CALL sendhp(":CALC3:LFIL:UPPER " + STR$(upper)) 'Set upper stats limit
CALL sendhp(":CALC3:AVER ON")                   'Enable statistics
CALL sendhp(":CALC3:AVER:COUNT " + STR$(nummeas))'Set number of measurements
                                                'to use in statistics
                                                'calculation

CALL sendhp(":CALC2:LIM:STATE ON")              'Enable limit testing. Must
                                                'do this to see graph
CALL sendhp(":CALC2:LIM:LOWER " + STR$(lower))  'Set lower limit
CALL sendhp(":CALC2:LIM:UPPER " + STR$(upper))  'Set upper limit
CALL sendhp(":CALC2:LIM:DISP GRAPH")            'Display limit graph

ON PEN GOSUB getstats                           'When SRQ happens, go get
PEN ON                                          'statistics
CALL IOPEN(isc&, priority%)                     'Watch for interrupts
PRINT "Making Period measurements"
CALL sendhp(":INIT:CONT ON")                    'Set counter to run

Loophere:                                       'Wait here until complete
IF complete THEN GOTO endprogram                'If stats received, then end
GOTO Loophere

getstats:
complete = 1                                    'Test bit
CALL IOSPOLL(source&, statusbyte)               'Check status byte
                                                'Should be 192
CALL sendhp(":INIT:CONT OFF")                   'Put counter in single
```

To Use Limits to Filter Data Before Measuring Stats (QuickBASIC) (Continued)

```
PRINT "Status byte = ", statusbyte
CALL sendhp(":CALC3:AVERAGE:TYPE MIN;:CALC3:DATA?") 'Ask for all the stats
CALL IOENTERS(source&, minimum, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE MAX;:CALC3:DATA?")
CALL IOENTERS(source&, maximum, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE MEAN;:CALC3:DATA?")
CALL IOENTERS(source&, mean, maxelem%, actual%)
CALL sendhp(":CALC3:AVERAGE:TYPE SDEV;:CALC3:DATA?")
CALL IOENTERS(source&, sdev, maxelem%, actual%)
PRINT
PRINT "Minimum Period     = ", minimum
PRINT "Maximum Period     = ", maximum
PRINT "Mean Period        =  ", mean
PRINT "Standard Deviation = ", sdev
RETURN

endprogram:
END

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Read and Store Calibration Data (QuickBASIC)

```
'Before calibrating the counter, it is a good idea to read
'and store the current calibration values in case something goes wrong with
'the calibration.
'This program reads the cal values, and stores them in a file on the computer
'hard drive. It then reads the data from the file and sends it back to
'the counter.
'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'                    'Required by Agilent 82335A
DIM SHARED source AS LONG                      'Address and select code
DIM CALDATA AS STRING * 61                     'Strings to be read
source& = 703                                   'Counter at address 3
isc& = 7                                       'Select code 7
state% = 1

CLS 0
CALL IOEOI(isc&, state%)           'Make sure EOI enabled
CALL IOCLEAR(source&)              'Clear the counter and interface
CALL sendhp("*RST")                'Reset Agilent counter
CALL sendhp("*CLS")                'Clear event registers and error queue
CALL sendhp("*SRE 0")              'Clear service request enable register
CALL sendhp("*ESE 0")              'Clear event status enable register
CALL sendhp(":STAT:PRES")          'Preset enable registers and transition
                                   'filters for operation and questionable
                                   'status structures.
PRINT "Reading Calibration Data"
CALL sendhp(":CAL:DATA?")
CALL ioenters(source&, CALDATA, 61, actf%)  'Read the ASCII characters

OPEN "CALDATA.DAT" FOR BINARY AS #1       'Store the cal data in a file
PUT #1, 1, CALDATA
CLOSE #1

'The following lines show how to open a file with calibration data
'and send it back to the counter.
PRINT
PRINT "Sending calibration data to counter"
OPEN "CALDATA.DAT" FOR BINARY AS #1
GET #1, 1, CALDATA
CLOSE #1
CALL sendhp(":CAL:DATA " + CALDATA)    'Send the data just read to counter
END

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Optimize Throughput (QuickBASIC)

```
'This program sets up the counter make 1000 frequency as fast as possible.
'Note that the arming is set to AUTO. This allows measurements to be taken
'quickly, but at the least resolution the counter can provide.
'See the program comments for details.
'Requires an Agilent 82335A/B GPIB interface card to a PC.
'The data is sent in ASCII format to preserve resolution.
'
'The SUB sendhp sends commands to the counter

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'            'Required by Agilent 82335A
DIM SHARED source AS LONG              'Address and select code
DIM i AS INTEGER                       'i is used for loops
DIM samples AS INTEGER
samples = 1000                         'Number of measurements
DIM freqstring(1000) AS STRING * 23    'String to be read
                                       'Reading ASCII formatted data
                                       'gives results to the correct
                                       'resolution. Must be read into
                                       'a string. Also, provides the
                                       'fastest data transfer.
maxelem% = 22                          'Maximum number of characters expected
actual% = 0                            'Returns actual characters received
source& = 703                          'Counter at address 3
isc& = 7                               'Select code 7
state% = 1                             'Used in IOEOI

CLS 0
CALL IOEOI(isc&, state%)             'Make sure EOI enabled
CALL IOCLEAR(source&)                'Clear the counter and interface
CALL sendhp("*RST")                  'Reset counter
CALL sendhp("*CLS")                  'Clear event registers and error queue
CALL sendhp("*SRE 0")                'Clear service request enable register
CALL sendhp("*ESE 0")                'Clear event status enable register
CALL sendhp(":STAT:PRES")            'Preset enable registers and transition
                                     'filters for operation and questionable
                                     'status structures
'The following commands will provide the fastest measurement throughput,
'independent of the state of the counter prior to these commands.
CALL sendhp(":FORMAT ASCII")         'ASCII give fastest throughput
CALL sendhp(":FUNC " + CHR$(34) + "FREQ 1" + CHR$(34)) 'Measure frequency
'The function must be a quoted string. The actual string sent to the
'counter is "FREQ 1"
```

To Optimize Throughput (QuickBASIC) (Continued)

```
'The following lines will provide the fastest throughput, regardless of
'the state of the counter before these lines are executed.
CALL sendhp(":FREQ:ARM:STAR:SOUR IMM")    'These 3 lines enable using
CALL sendhp(":FREQ:ARM:STOP:SOUR IMM")    'time arming with a 0.1 second
CALL sendhp(":EVENT1:LEVEL 0")            'Set trigger level on channel 1
                                          'This also disables auto trigger
CALL sendhp(":CALC:MATH:STATE OFF")       'Make sure all post-processing
CALL sendhp(":CALC2:LIM:STATE OFF")       'is turned off.
CALL sendhp(":CALC3:AVER:STATE OFF")
CALL sendhp(":HCOPY:CONT OFF")            'Do not update printing operations
CALL sendhp(":ROSC:SOUR INT")
CALL sendhp(":ROSC:EXT:CHECK OFF")
CALL sendhp(":DIAG:CAL:INT:AUTO OFF")     'Disable automatic interpolator
                                          'calibration. The most recent
                                          'calibration factors will be
                                          'used in the calculation for
                                          'frequency
CALL sendhp("*DDT #15FETC?")              'Define trigger as fetc?
CALL sendhp(":DISP:ENABLE OFF")           'Turn off the display
CALL sendhp("READ:FREQUENCY?")            'Read the expected frequency
CALL IOENTERS(source&, freqstring(1), maxelem%, actual%)
CALL sendhp(":FREQ:EXP1 " + freqstring(1))  'Send the expected frequency
CALL sendhp(":INIT:CONT ON")                'Start making measurements
PRINT "Making measurements"

FOR i = 1 TO samples
CALL IOTRIGGER(source)                              'Query the counter for data
CALL IOENTERS(source&, freqstring(i), 22, actual%) 'Read the ASCII characters
NEXT i

PRINT "Measurements complete"

END

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Use Macros (QuickBASIC)

```
'This program is useful for writing macros for the counter. Softkeys
'are available at the bottom of the computer screen to help determine
'the status of the macros.
'The SUB sendhp sends commands to the Agilent 53131A

DECLARE SUB sendhp (code$)
REM $INCLUDE: 'QBSETUP.BAS'                    'Required by Agilent 82335A
DIM SHARED source AS LONG                      'Address and select code
DIM maxlength AS INTEGER
DIM actual AS INTEGER
DIM length AS INTEGER
maxlength = 6400
DIM answer AS STRING
DIM namemacro AS STRING
DIM commandmacro AS STRING
DIM results AS STRING * 6400
DIM macros AS STRING
source = 703                                   'Agilent 53131A at address 3
isc& = 7                                       'Select code 7
state% = 1

CLS
CALL IOEOI(isc&, state%)             'Make sure EOI enabled
CALL sendhp("*RST")                  'Reset the counter
CALL sendhp("*CLS")                  'Clear event registers and error queue
CALL sendhp("*SRE 0")                'Clear service request enable register
CALL sendhp("*ESE 0")                'Clear event status enable register
CALL sendhp(":STAT:PRES")            'Preset enable registers and transition
                                     'filters for operation and questionable
                                     'status structures.

CALL sendhp(":INIT:CONT OFF")        'Put counter in Single

KEY 1, "Free"
KEY 2, "Enable"
KEY 3, "Display"
KEY 4, "Query"
KEY 5, "Define"
KEY 6, "Delete1"
KEY 7, "Purge"
KEY 8, "Disable"
KEY 9, "Send"
KEY 10, "QUIT"
FOR i = 1 TO 10
KEY(i) ON
NEXT i

KEY ON
```

To Use Macros (QuickBASIC) (Continued)

```
ON KEY(1) GOSUB availablememory
ON KEY(2) GOSUB enablemacro
ON KEY(3) GOSUB displaymacro
ON KEY(4) GOSUB querymacro
ON KEY(5) GOSUB definemacro
ON KEY(6) GOSUB deletemacro
ON KEY(7) GOSUB purgemacro
ON KEY(8) GOSUB disablemacro
ON KEY(9) GOSUB sendmacro
ON KEY(10) GOSUB quit
loophere: GOTO loophere                'Wait for function key to be pressed

availablememory:                       'Display available macro memory
CALL sendhp(":MEM:FREE:MACRO?")
CALL IOENTER(source&, freemacro)
PRINT "Available macro memory = "; freemacro
RETURN

enablemacro:                           'Enable all macros
sendhp ("*EMC 1")
PRINT "Macros Enabled"
RETURN

displaymacro:                          'Display macros available in counter
CLS
sendhp ("*LMC?")
CALL IOENTERS(source, results$, maxlength, actual)
macros$ = LEFT$(results$, actual)
PRINT "The following macros are available:"
PRINT macros$
RETURN

querymacro:                            'Ask for definition of a macro
CLS
GOSUB displaymacro
INPUT "Enter the name of the macro you want to see ", namemacro$
IF namemacro$ = "" THEN RETURN
sendhp ("*GMC? " + CHR$(39) + namemacro$ + CHR$(39))
CALL IOENTERS(source, results, maxlength, actual)
macroname$ = LEFT$(results, actual)
PRINT namemacro$; " is defined as:"
PRINT macroname$
RETURN

deletemacro:                           'Delete a macro
GOSUB displaymacro
INPUT "Enter the name of the macro you want to delete ", namemacro$
IF namemacro$ = "" THEN RETURN
sendhp ("MEM:DELETE:MACRO " + CHR$(39) + namemacro$ + CHR$(39))
RETURN
```

To Use Macros (QuickBASIC) (Continued)

```
purgemacro:                          'Purge all macros
INPUT "Are you sure you want to purge all macros? ", answer$
answer$ = UCASE$(answer$)
IF answer$ = "Y" THEN
sendhp ("*PMC")
PRINT "All macros purged"
END IF
RETURN

disablemacro:                        'Disable macros, but do not purge
sendhp ("*EMC 0")
PRINT ("Macros Disabled")
RETURN

sendmacro:
CLS
GOSUB displaymacro
INPUT "Enter the name of the macro to send ", namemacro$
IF namemacro$ = "" THEN RETURN
sendhp (namemacro$)
RETURN

definemacro:                         'Define a macro
CLS
INPUT "Enter the name of the macro to be defined ", namemacro$
INPUT "Enter the commands to be sent ", commandmacro$
length = LEN(commandmacro$)
numchar = INT(LOG(length) / LOG(10#)) + 1
header$ = "#" + LTRIM$(STR$(numchar))
PRINT header$
macrocommand$ = header$ + LTRIM$(STR$(length)) + commandmacro$
code$ = "*DMC " + CHR$(39) + namemacro$ + CHR$(39) + "," + macrocommand$
PRINT code$
CALL iooutputs(source, code$, LEN(code$))
RETURN


quit:
PRINT "End of Program"
STOP
RETURN

SUB sendhp (code$)
CALL iooutputs(source, code$, LEN(code$))
END SUB
```

To Make a Frequency Measurement (Turbo C)

```
/* This program sets up the counter to make 10 freqeuncy measurements
   on channel 1, using a 0.1 second gate time.
   The results are displayed on the computer CRT
   The program comments discuss the meaning of each command.
   ASCII result format is used to preserve resolution. */

#include <stdio.h>
#include <string.h>
#include "CHPIB.H"
#include "CFUNC.H"

void sendhp(char *);   /* function to send command to counter */

/* global data */

long ctr=703;         /* Counter is at address 03. GPIB is at select code 7 */
int  error;

void main()
{   long isc=7;              /* Select code 7 */
    int state=1;             /* Used in IOEOI    */
    int i;                   /* Used for loop counter */
    int samples=10;          /* Number of measurements to take */
    int length=23;           /* Max number of bytes per measurements */
    char freq[23];           /* Array to hold frequency string */
    IORESET(isc);            /* Clear the GPIB interface */
    sendhp("*RST");          /* Reset the counter */
    sendhp("*CLS");          /* Clear event registers and error queue */
    sendhp("*SRE 0");        /* Clear service request enable register */
    sendhp("*ESE 0");        /* Clear event status enable register */
    sendhp(":STAT:PRES");    /* Preset enable registers and transition
                                filters for operation and questionable
                                status structures */
    IOEOI(isc,state);        /* Enable use of EOI */

    sendhp(":FUNC 'FREQ 1'");             /* Measure frequency on channel 1
                                             Note that the function must
                                             be a quoted string. The actual
                                             string sent to the counter is
                                             'FREQ 1'. */
    sendhp(":FREQ:ARM:STAR:SOUR IMM");    /* These 3 lines enable the */
    sendhp(":FREQ:ARM:STOP:SOUR TIM");    /* time arming mode with a */
    sendhp(":FREQ:ARM:STOP:TIM .1");      /* 0.1 second gate time */
```

To Make a Frequency Measurement (Turbo C) (Continued)

```
    for (i=1; i<=samples ;i++)
        {
          sendhp("INIT");                 /* Start a measurement */
          sendhp("FETCH:FREQUENCY?");
          IOENTERS(ctr,freq,&length);     /* fetch the data */
          length=strlen(freq);            /* Get length of result so */
          freq[length-1]='\0';            /* the linefeed can be removed */
          printf ("Frequency %d = %s Hz\n",i,freq);
        }
    printf("Press a key to continue\n");
    getch();

}


/* Function to send command to Agilent 53131A */

void sendhp(hpib_cmd)
char *hpib_cmd;
{

char hpcmd[80];                     /* Variables used by function */
int length;
strcpy(hpcmd,hpib_cmd);
length=strlen(hpcmd);
error=IOOUTPUTS(ctr,hpcmd,length);  /* Send command to Agilent 53131A */
if (error!=0)
printf("Error during GPIB: %d Command %s\n",error,hpcmd);

}
```

To Use Limits to Filter Data Before Measuring Statistics (Turbo C)

```
/* This program instructs the counter to determine the statistics of
   50 Period measurements that are within programmed test limit values.
   Periods that are outside of the limits are not included in the statistics.
   The Limit graph is displayed so you can see if measurements are in limit.
   To alert the program that the statistics are ready, bit 8 in the Operation
   Status Register is used. When statistics are being calculated, this bit is
   high, when they are complete, this bit goes low. By using the transition
   filters, an SRQ can be generated when statistics are complete (the
   transition from high to low of bit 8 in the Operation Status register.)*/


#include <stdio.h>     /* used for printf() */
#include <dos.h>       /* used for delay() */

#include "CHPIB.H"     /* GPIB library constant declarations */
#include "CFUNC.H"     /* GPIB library function prototypes */

void sendhp(char *);   /* function to send command to counter */

/* global data */

long ctr=703;   /* Counter is at address 03. GPIB is at select code 7 */
int  error;

void main()
{
    long isc=7;              /* Select code 7 */
    int condition=1;         /* Used in IOSTATUS */
    int status;              /* Used in IOSTATUS */
    int state=1;             /* Used in IOEOI    */
    char mean[23];           /* mean variable */
    char minimum[23];        /* minimum variable */
    char maximum[23];        /* maximum variable */
    char sdev[23];           /* standard deviation variable */
    int length=23;           /* Used in IOENTERS */
    clrscr();                /* Clear the computer CRT */

    IORESET(isc);            /* Clear the GPIB interface */
    sendhp("*RST");          /* Reset the counter */
    sendhp("*CLS");          /* Clear event registers and error queue */
    sendhp("*SRE 0");        /* Clear service request enable register */
    sendhp("*ESE 0");        /* Clear event status enable register */
    sendhp(":STAT:PRES");    /* Preset enable registers and transition
                                filters for operation and questionable
                                status structures */
```

To Use Limits to Filter Data Before Measuring Statistics (Turbo C) (Continued)

```
IOEOI(isc,state);                    /* Enable use of EOI */
sendhp(":FUNC 'FREQ 1'");            /* Make a frequency measurement */
sendhp(":FREQ:ARM:STAR:SOUR IMM");   /* These 3 lines enable */
sendhp(":FREQ:ARM:STOP:SOUR TIM");   /* time arming with a .001 */
sendhp(":FREQ:ARM:STOP:TIM .001");   /* second gate time */
sendhp(":STAT:OPER:ENABLE 256");     /* Computing statistics bit in
                                        Operation Status register */
sendhp(":STAT:OPER:NTR 256");        /* When statistics are complete, */
sendhp(":STAT:OPER:PTR 0");          /* the bit will go from high to low
                                        so a negative transition is
                                        needed to enable the bit that is
                                        summarized in the Status Byte
                                        register. */
sendhp("*SRE 128");                  /* This is the bit from the Operation
                                        Status register that is summarized
                                        in the Status Byte Register */
sendhp(":CALC3:LFIL:STAT ON");       /* Enable statistics filter */
sendhp(":CALC3:LFIL:LOWER 1 MHz");   /* Set lower limit to 1 MHz */
sendhp(":CALC3:LFIL:UPPER 2 MHz");   /* Set upper limit to 2 MHz */
sendhp(":CALC3:AVER ON");            /* Enable statistics */
sendhp(":CALC3:AVER:COUNT 50");      /* Use 50 measurements for stats */

sendhp(":CALC2:LIM:STAT ON");        /* Enable limit testing. This must
                                        happen in order to see limit
                                        graph */
sendhp(":CALC2:LIM:LOWER 1 MHz");    /* Set lower limit */
sendhp(":CALC2:LIM:UPPER 2 MHz");    /* Set upper limit */
sendhp(":CALC2:LIM:DISP GRAPH");     /* Display limit graph */

/* Waiting for the measurement to complete */

printf("Waiting for measurement to complete\n");
sendhp(":INIT:CONT ON");             /* Start making measurements */
do
{
IOSTATUS(isc,condition,&status);     /* Check status byte */
  delay(200);                        /* Wait 200 milliseconds */
}
while (status!=1);                   /* if =1 then measurement complete */

/* Measurement is complete, get the data */
```

To Use Limits to Filter Data Before Measuring Statistics (Turbo C) (Continued)

```
IOSPOLL(ctr,&status);               /* Serial poll counter for status */
    puts("Transferring and processing data");
    sendhp(":INIT:CONT OFF");           /* Set counter to Run */
    sendhp(":CALC3:AVERAGE:TYPE MIN;:CALC3:DATA?");
    IOENTERS(ctr,minimum,&length);      /* Get the data from the counter */
    sendhp(":CALC3:AVERAGE:TYPE MAX;:CALC3:DATA?");
    IOENTERS(ctr,maximum,&length);      /* Get the data from the counter */
    sendhp(":CALC3:AVERAGE:TYPE MEAN;:CALC3:DATA?");
    IOENTERS(ctr,mean,&length);         /* Get the data from the counter */
    sendhp(":CALC3:AVERAGE:TYPE SDEV;:CALC3:DATA?");
    IOENTERS(ctr,sdev,&length);         /* Get the data from the counter */

    printf("Mean frequency     = %s",mean);
    printf("Minimum frequency  = %s",minimum);
    printf("Maximum frequency  = %s",maximum);
    printf("Standard deviation = %s",sdev);
    printf("Press a key to continue\n");
    getch();

}


/* Function to send command to Agilent 53131A */

void sendhp(hpib_cmd)
char *hpib_cmd;
{

char hpcmd[80];                     /* Variables used by function */
int length;
strcpy(hpcmd,hpib_cmd);
length=strlen(hpcmd);
error=IOOUTPUTS(ctr,hpcmd,length);  /* Send command to Agilent 53131A */
if (error!=0)
printf("Error during GPIB: %d Command %s\n",error,hpcmd);

}
```

To Optimize Throughput (Turbo C)

```
/* This program sets up the counter to transfer data at the fastest
   possible rate. Note that the arming mode is AUTO. This mode provides
   the least resolution of all the arming modes.
   The program comments discuss the meaning of each command.
   ASCII result format is used to preserve resolution.
   For optimal performance, compile for best speed. */


#include <stdio.h>   /* used for printf() */
#include <string.h>  /* used for strlen() */
#include "CHPIB.H"   /* GPIB library constant declarations */
#include "CFUNC.H"   /* GPIB library function prototypes */

void sendhp(char *); /* function to send command to counter */

/* global data */

long ctr=703;        /* Counter is at address 03. GPIB is at select code 7 */
int  error;

void main()
{
    long isc=7;              /* Select code 7 */
    int status;              /* Used in IOSTATUS */
    int state=1;             /* Used in IOEOI    */
    int i;                   /* Used for loop counter */
    float exp_freq;          /* Expected frequency value */
    int readings = 1000;     /* Number of measurements to take */
    int length=23;           /* Max number of bytes per measurement */
    char freq[1001][23];     /* Array to hold measurements */
    char destination[130];   /* Used for expected frequency */
    IORESET(isc);            /* Clear the GPIB interface */
    sendhp("*RST");          /* Reset the counter */
    sendhp("*CLS");          /* Clear event registers and error queue */
    sendhp("*SRE 0");        /* Clear service request enable register */
    sendhp("*ESE 0");        /* Clear event status enable register */
    sendhp(":STAT:PRES");    /* Preset enable registers and transition
                                filters for operation and questionable
                                status structures */
    IOEOI(isc,state);        /* Enable use of EOI */

    sendhp(":FUNC 'FREQ 1'");             /* Make a frequency measurement */
    sendhp(":FREQ:ARM:STAR:SOUR IMM");    /* These 2 lines enable the */
    sendhp(":FREQ:ARM:STOP:SOUR IMM");    /* AUTO arming mode*/
    sendhp(":EVENT1:LEVEL 0");            /* Set a trigger level for
                                             channel 1. This disables the
                                             auto trigger, increasing
                                             throughput */
```

To Optimize Throughput (Turbo C) (Continued)

```
    sendhp(":ROSC:SOURCE INT");           /* Use internal oscillator. If
                                             you want to use an external
                                             timebase, you must select it
                                             and turn off the automatic
                                             detection using:
                                             :ROSC:EXT:CHECK OFF */
    sendhp(":DIAG:CAL:INT:AUTO OFF");     /* Disable automatic interpolater
                                             calibration. The most recent
                                             calibration values are used in
                                             the calculation of frequency */
    sendhp(":DISP:ENABLE OFF");           /* Turn off the counter display */
    sendhp(":HCOPY:CONT OFF");
    sendhp(":CALC:MATH:STATE OFF");       /* Disable any post processing */
    sendhp(":CALC2:LIM:STATE OFF");
    sendhp(":CALC3:AVER:STATE OFF");
    sendhp("*DDT #15FETC?");              /* Define the Trigger command. This
                                             means the command FETC? does not
                                             need to be sent for every
                                             measurement, decreasing the
                                             number of bytes transferred over
                                             the bus */
    sendhp(":INIT:CONT ON");              /* Put the counter in Run mode */
    sendhp("FETCH:FREQ?");                /* Fetch the frequency to be */
    IOENTER(ctr,&exp_freq);               /* for the expected frequency */
    strcpy(destination,":FREQ:EXP1 ");    /* Copy string */
    sprintf(&destination[strlen(destination)],"%e",exp_freq); /* Append
                                             expected frequency value */
    sendhp(destination);                  /* Send the expected frequency */
                                          /* This number must be within 10%
                                             of the Ch 1 input frequency.
                                             Using this greatly increases
                                             throughput, but is not
                                             recommended for signals that
                                             change by more than 10% */
 puts("Transferring and processing data\n");
 for (i=1; i<=readings ;i++)
{
 IOTRIGGER(ctr);                          /* Trigger the counter and */
 IOENTERS(ctr,freq[i],&length);           /* read the data */
 }

    printf("Measurement complete.  Press a key to continue.\n");
    getch();

}
```

To Optimize Throughput (Turbo C) (Continued)

```
/* Function to send command to Agilent 53131A */

void sendhp(hpib_cmd)
char *hpib_cmd;
{

char hpcmd[80];                     /* Variables used by function */
int length;
strcpy(hpcmd,hpib_cmd);
length=strlen(hpcmd);
error=IOOUTPUTS(ctr,hpcmd,length);  /* Send command to Agilent 53131A */
if (error!=0)
printf("Error during GPIB: %d Command %s\n",error,hpcmd);

}
```

4

Command Reference

A Dictionary

Introduction

This chapter describes the SCPI Subsystem commands and the IEEE 488.2 Common commands for the Agilent 53131A/132A 225 MHz Universal Counter. The information in this chapter will help you program the Counter over the GPIB.

The commands are presented in alphabetical order.

- SCPI Subsystem commands are described on pages 4-4 thru 4-121.
- IEEE 488.2 Common command descriptions start on page 4-122.
- Device Clear and Group Execute Trigger descriptions are also included on pages 4-31 and 4-42, respectively.

For each command description:

- where the phrase “Sets or queries” is used, the command setting can be queried by omitting the parameter and appending a “?” to the last command keyword.

  For example,

  ```
  :INPut:COUPling [AC | DC]
  ```

  can be queried with

  ```
  :INPut:COUPling?
  ```

- unless otherwise noted, a command described as an *event* cannot be queried.
- unless otherwise noted, the command setting is affected by save/recall.
- the square brackets, [ ], are used to indicate that the element(s) within the brackets are optional. Note, the brackets are NOT part of the command and should not be sent to the Counter.
- the vertical bar, |, is used to mean “OR” and is used to separate alternative options.

- the short form of keywords is shown in uppercase.

- quotation marks may be part of the command's parameter; the quotation marks shown must be sent to the Counter.

- unless otherwise noted, the command is sequential (not overlapped).

See Chapter 3 in this guide for details regarding command syntax, parameter types, and query response types.

See the *Agilent* 531331A/*132A Operating Guide*, Table 2-6, for power-up values.

:ABORt Command

:ABORt

This command is an event that causes the Counter to abort, as quickly as possible, any measurement in progress.

The :ABORt command is not complete until the current measurement is stopped. The execution of an ABORt command sets false any Pending Operation Flags that were set true by initiation of measuring.

Comments

- If :ABORt is issued while the measurement cycle is idle (:INIT:CONT OFF and pending operation flag is false), the command will be ignored.

- If :ABORt is issued while a single measurement is in progress (:TRIG:COUN:AUTO OFF or :CALC3:AVER OFF, :INIT:CONT OFF, and pending operation flag is true), the measurement will be aborted and pending operation flag set false.

- If :ABORt is issued while repetitive measurement are being made (:INIT:CONT ON), the current measurement in progress will be aborted and the pending operation flag set false. Then, a new measurement will automatically be initiated and the pending operation flag set true.

- If :ABORt is issued while a block of measurements is in progress (:TRIG:COUN:AUTO ON and :CALC3:AVER ON, :INIT:CONT OFF, and pending operation flag is true), the measurement block will be aborted and the pending operation flag set false.

- When a measurement or block of measurements is aborted, the Measuring bit in the Operation Status Register will be set false.

- Aborting a measurement in progress invalidates the result, with one exception. :ABORt is the appropriate way to terminate an automatically armed Totalize measurement.

Related Front-Panel Keys

Stop/Single

:CALCulate Subsystems

Three :CALCulate subsystems (:CALCulate[1], :CALCulate2, and :CALCulate3) perform post-acquisition data processing and data transfer of the corresponding results. Functions in the SENSe subsystem are related to data acquisition, while the :CALCulate systems operate on the data acquired by a SENSe function as shown in Figure 4-1.

The :CALCulate subsystems are logically between the :SENSe subsystem and the data output to either the bus or display. When a measurement is initiated (by a :MEASure, :READ, or an :INITiate command), the :SENSe subsystem collects data. This data is transformed by :CALCulate[1|2|3], as specified, and then passed on to the selected output. In effect, the collection of new data "initiates" the :CALCulate subsystems. The :CALCulate subsystems may also be directed by command to transform, making it possible to change the configuration of :CALCulate and consequently derive a different set of results from the same SENSe data set without re-acquiring SENSe data.

Calculated results are available (valid) until new results are computed or until relevant instrument state is changed.

The :CALCulate3 subsystem consists of two sub-blocks as shown in Figure 4-1. The data flows through the sub-blocks in a serial fashion. The manner in which these sub-blocks are arranged is specified in the :CALC3:PATH? query.

The :CALCulate[1|2|3] settings are not used when measuring Totalize or Voltage Peaks (voltage minimum, maximum, or peak-to-peak).

NOTE

Not until :CALCulate[1]:MATH:STATe is set to ON will any of the :CALCulate[1] settings or :TRACe[:DATA] settings be used.

Not until :CALCulate2:LIMit:STATe is set to ON will any of the :CALCulate2 settings be used.

Not until :CALCulate3:LFILter:STATe is set to ON will any of the :CALCulate3:LFILter settings be used.

Not until :CALCulate3:AVERage:STATe is set to ON will any of the :CALCulate3:AVERage settings be used.

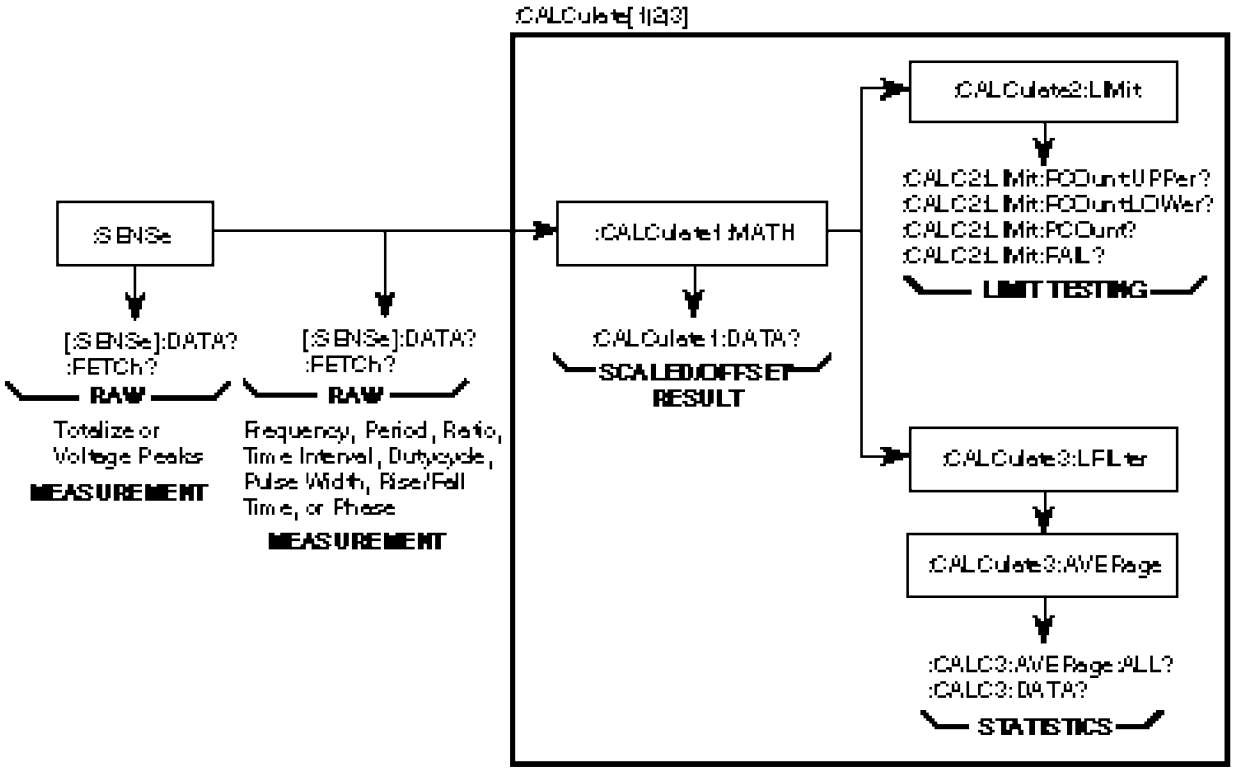


Figure 4-1. The CALCulate Subsystems

:CALCulate[1] Subsystem

Performs post-acquisition math (scale/offset) processing (on the data acquired by a SENSe function) and data transfer of the scaled/offset result. See the :TRACe subsystem for commands used to set the scale and offset.

NOTE Not until :CALCulate[1]:MATH:STATe is set to ON will any of the :CALCulate[1] settings or :TRACe[:DATA] settings be used.

:CALCulate[1]:DATA?

Queries the current scaled and offset measurement result.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.
- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15, depending on the measurement resolution. Only significant digits will be returned.
- If no valid result exists, Not a Number **9.91E37** is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, Not a Number **9.91E37** is returned and error -221 is generated.

Comments

- Query only.
- If this command is issued when math is enabled and while a measurement is in progress, no response will be produced until the measurement completes.
- This command holds off subsequent commands from being processed until a measurement completes. This holdoff action can only be canceled by the measurement completing, a device clear, or power cycle.
- The last calculated result remains valid until a new computation is made or a relevant instrument state is modified.

:CALCulate[1]:FEED "[:]SENSe[1]"

Sets or queries the data flow to be fed into the CALCulate[1] block.

Since the Counter can only sense one function at a time, there is only one valid parameter.

Query Response The string "SENS" is returned.

Comments *RST: "SENSe[1]"

:CALCulate[1]:IMMediate

This command is an event that causes the Counter to recalculate existing data without re-acquiring data. (This recalculation also happens automatically when any change is made to the
:CALCulate[1|2] subsystems while :CALC:IMM:AUTO is ON.)

:CALC:IMM? is semantically equivalent to :CALC:IMM;DATA?.
The query form outputs the results of the new calculation.

This command will not affect:

:CALC2:LIM:FCO
:CALC2:LIM:PCO
:CALC3: ...

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.

- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15, depending on the measurement resolution.
Only significant digits will be returned.

- If no valid result exists, Not a Number **9.91E37** is returned and error -230 is generated.

- If the current measurement is Totalize or Voltage Peaks, Not a Number **9.91E37** is returned and error -221 is generated.

Comments This command causes post-processing to occur in the :CALCulate2 subsystem, as well as the CALCulate subsystem.

:CALCulate[1]:IMMediate:AUTO <Boolean>

Sets or queries whether post-processing (recalculation) will automatically occur whenever any changes are made to the :CALCulate[1|2] subsystems.

With :CALC:IMM:AUTO set to OFF, :CALCulate[1|2] only produces new results when new SENSe data is acquired or when the :CALCulate:IMMediate command is received.

Once :CALC:IMM:AUTO is set to ON, the CALCulate[1|2] subsystems produce new results when any CALCulate[1|2] command is processed, even when new SENSe data is not being acquired.
This allows the user to make configuration changes in the CALCulate[1|2] subsystems and immediately have new CALCulate[1|2] results on the same SENSe data.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- This command affects all of the post-processing subsystems settings (:CALC:IMM:AUTO).
- Note that the Counter powers up with :CALC:IMM:AUTO set to ON, but *RST sets it to OFF.

:CALCulate[1]:MATH Subtree

This subtree collects together the commands related to math (scale/offset) processing. See the :TRACe subsystem for commands used to set the scale and offset.

Many of these commands are query-only because the Counter has only one fixed math operation.

:CALCulate[1]:MATH[:EXPRession]:CATalog?

Queries defined equation name.

Query Response The string “SCALE_OFFSET” is returned.

Comments Query only.

:CALCulate[1]:MATH[:EXPRession][:DEFine]?

Queries equation used for math operation.

Query Response A sequence of ASCII-encoded bytes:

("SENS" * SCALE + OFFSET)

terminated with a new line and EOI.

Comments

- Query only.
- This query should be the last query in a terminated program message; otherwise, error -440 is generated.

:CALCulate[1]:MATH[:EXPRession]:NAME SCALE_OFFSET
or
:CALCulate[1]:MATH[:EXPRession]:SELect SCALE_OFFSET

Sets or queries the name of the expression selected for math processing.

Query Response A sequence of ASCII-encoded bytes: SCALE_OFFSET

Comments *RST: SCALE_OFFSET

:CALCulate[1]:MATH:STATe <Boolean>

Sets or queries the math enable.

This enable specifies whether or not measurement (SENSe) data will be scaled and offset.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- Updating the math enable causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.

Related Front-Panel Keys Scale & Offset

:CALCulate2 Subsystem

This subsystem performs post-acquisition limit testing and data transfer.

NOTE Not until :CALCulate2:LIMit:STATe is set to ON will any of the :CALCulate2 settings be used.

:CALCulate2:FEED "[:]CALCulate[1]"

Sets or queries the data flow to be fed into the CALCulate2 block.

Query Response The string "CALC" is returned.

Comments *RST: "CALCulate[1]"

:CALCulate2:IMMediate

This command is an event that causes the Counter to recalculate existing data without re-acquiring data. (This recalculation also happens automatically when any change is made to the :CALCulate[1|2] subsystems while :CALC2:IMM:AUTO is ON.)

The only limit result that can be truly post-processed is :CALC2:LIM:FAIL?. The limit counts (:CALC2:LIM:FCO and :CALC2:LIM:PCO) reflect measurements that were limit-tested at time of data acquisition.

This command will not affect:

:CALC2:LIM:FCO
:CALC2:LIM:PCO
:CALC3: ...

Comments This command causes post-processing to occur in the :CALCulate[1] subsystem, as well as the :CALCulate2 subsystem.

:CALCulate2:IMMediate:AUTO <Boolean>

Sets or queries whether post-processing (recalculation) will automatically occur whenever any changes are made to the :CALCulate[1|2] subsystems.

With :CALC2:IMM:AUTO set to OFF, CALCulate[1|2] only produces new results when new SENSe data is acquired or when the CALCulate2:IMMediate command is received.

Once :CALC2:IMM:AUTO is set to ON, the CALCulate[1|2] subsystems produce new results when any CALCulate[1|2] command is processed, even when new SENSe data is not being acquired.
This allows the user to make configuration changes in the CALCulate[1|2] subsystems and immediately have new CALCulate[1|2] results on the same SENSe data.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- This command affects all of the post-processing subsystems settings (:CALC[1]:IMM:AUTO).
- Note that the Counter powers up with :CALC2:IMM:AUTO set to ON, but *RST sets it to OFF.
- The only limit result that can be truly post-processed is :CALC2:LIM:FAIL?. The limit counts (:CALC2:LIM:FCO and :CALC2:LIM:PCO) reflect measurements that were limit-tested at
time of data acquisition.

:CALCulate2:LIMit Subtree

This subtree collects together the commands associated with controlling and getting reports from a single LIMit test. The limit test is defined as both an upper and lower limit test.

If the measurement cycle is aborted or terminates abnormally, the limit test status will be unaffected. That is, an aborted or abnormally terminated measurement does not get limit tested and has no effect on the limit test results.

:CALCulate2:LIMit:CLEar:AUTO <Boolean>

Sets or queries if the limit test results are to be cleared with each :INITiate[:IMMediate] and :INITiate:CONTinuous ON operation.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: ON
- When AUTO is ON, the Counter will perform the following whenever :INIT[:IMM] or :INIT:CONT ON is executed:
  - Invalidate the limit data.
  - Clear :CALC2:LIM:FAIL, :CALC2:LIM:FCOunt, and :CALC2:LIM:PCOunt information.
  - Turn off the front-panel display's **Limit** annunciator.
  - Set the Limit-Detect output of the RS-232 connector to the in-limit voltage level.
- When AUTO is OFF, the only way to clear the limit-test results is to send :CALC2:LIM:CLE[:IMM].

:CALCulate2:LIMit:CLEar[:IMMediate]

This command is an event that causes the Counter to

- immediately invalidate the limit data,
- clear the information in :CALC2:LIM:FAIL, :CALC2:LIM:FCOunt, and :CALC2:LIM:PCOunt,
- turn off the front-panel display's Limit annunciator, and
- set the Limit-Detect output to its in-limit voltage level.

Comments

If :CALC2:LIM:STAT is OFF, error -221 is generated.

:CALCulate2:LIMit:DISPlay GRAPh | NUMBer

Sets or queries whether the measurement display is numeric or symbolic (on a graph).

When :CALC2:LIM:DISP is NUMBer, the measurement results are displayed numerically. When :CALC2:LIM:DISP is GRAPh,
the measurement results are displayed symbolically on a graph;
the measurement result is represented by an asterisk (*), while the upper and lower limits are each represented by a colon (:).

Query Response A sequence of ASCII-encoded bytes: GRAP or NUMB

Comments

- RST: NUMBer
- This command updates the display mode immediately. The display update is independent of :CALC2:IMM:AUTO state.
- See the section titled “How to Program the Counter to Display Results” in Chapter 3 for programming examples.

Related Front-Panel Keys Limit Modes

:CALCulate2:LIMit:FAIL?

Queries the status of the last measurement that was limit tested.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of zero indicates the last tested measurement passed the limit test. A value of one indicates the last tested measurement failed.
- If no valid result exists, 0 is returned and error -230 is generated.

Comments

- Query only.
- If this command is issued when limit testing is enabled and while a measurement is in progress, no response will be produced until the measurement completes.

? - This command holds off subsequent commands from being processed until a measurement completes. This holdoff action can only be canceled by the measurement completing, a device clear, or power cycle.

- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

:CALCulate2:LIMit:FCOunt:LOWer?

Queries the number of limit test failures (that is, the Fail COunt) at the lower limit.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- If CALC2:LIM:STATe is OFF, 0 is returned and error -221 is generated.
- If no valid result exists, 0 is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

Comments Query only.

:CALCulate2:LIMit:FCOunt[:TOTal]?

Queries the total Fail COunt (that is, the number of measurements that have failed the limit test). No failures is indicated by 0.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- If CALC2:LIM:STATe is OFF, 0 is returned and error -221 is generated.
- If no valid result exists, 0 is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

Comments Query only.

:CALCulate2:LIMit:FCOunt:UPPer?

Queries the number of limit test failures (that is, the Fail COunt) at the upper limit.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- If CALC2:LIM:STATe is OFF, 0 is returned and error -221 is generated.
- If no valid result exists, 0 is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

Comments Query only.

:CALCulate2:LIMit:LOWer[:DATA] <numeric_value> [HZ | S |DEG]

Sets or queries the lower limit used for limit testing.

When the result is less than the lower limit, a fail is reported; when the result is equal to the lower limit, a fail is not reported.

If math is enabled (:CALC:MATH:STATe ON), the limit value specified should take into account that the limit testing is on measurements that have been scaled and offset.

<numeric_value> Range -9.9999990000E+12 to -1.0000000000E-13, 0.0000000000, +1.0000000000E-13 to +9.9999990000E+12.

<numeric_value> Resolution 11 digits

Query Response Numeric data transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 0.0000000000
- This command couples :CALC3:LFIL:LOW to the same value.
- Updating the lower limit value causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel Enter key, while this value is in the 11-digit display, will update the actual value to the displayed (rounded) value.

Related Front-Panel Keys Uppr & Lower

:CALCulate2:LIMit:PCOunt[:TOTal]?

Queries the total Pass COunt (that is, the number of measurements that passed the limit test).

Query Response

- Numerical data transferred as ASCII bytes in <NR1> format.
- If CALC2:LIM:STATe is OFF, 0 is returned and error -221 is generated.
- If no valid result exists, 0 is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

Comments Query only.

:CALCulate2:LIMit:STATe <Boolean>

Sets or queries the limit test enable.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- When :CALC2:LIM:STAT OFF is sent, it causes the Counter to:
  - Invalidate the limit data and clear the information in :CALC2:LIM:FAIL, :CALC2:LIM:FCOunt, and :CALC2:LIM:PCOunt.
  - Turn off the front-panel display's **Limit** annunciator.
  - Set the Limit-Detect output to the in-limit voltage level.

Related Front-Panel Keys Limit Modes

:CALCulate2:LIMit:UPPer[:DATA] <numeric_value> [HZ | S | DEG]

Sets or queries the upper limit used for limit testing.

When the result is greater than the upper limit, a fail is reported; when the result is equal to the upper limit, a fail is not reported.

If math is enabled (:CALC:MATH:STATe ON), the limit value specified should take into account that the limit testing is on measurements that have been scaled and offset.

<numeric_value> Range -9.9999990000E+12 to -1.0000000000E-13, 0.0000000000, +1.0000000000E-13 to +9.9999990000E+12.

<numeric_value> Resolution 11 digits

Query Response Numeric data transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 0.0000000000
- This command couples :CALC3:LFIL:UPP to the same value.
- Updating the upper limit value causes the limits counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel **Enter** key, while this value is in the 11-digit display, will update the actual value to the displayed (rounded) value.

Related Front-Panel Keys Uppr & Lower

:CALCulate3 Subsystem

This subsystem performs post-acquisition statistics computation and data transfer.

NOTE

Not until :CALCulate3:LFILter:STATe is set to ON will any of the :CALCulate3:LFILter settings be used.

Not until :CALCulate3:AVERage[:STATe] is set to ON will any of the :CALCulate3:AVERage settings be used.

The statistics results are unaffected by post-processing invoked with :CALC[1|2]:IMM.

:CALCulate3:AVERage Subtree

This subtree collects together the commands associated with the statistics capabilities.

The statistics results combine successive measurements to produce a composite result

Not until :CALCulate3:AVERage[:STATe] is set to ON will any of one :CALCulate3:AVERage settings be used.

:CALCulate3:AVERage:ALL?

This query returns all four statistics (i.e., mean, standard deviation, maximum, and minimum).

Statistics should be enabled (:CALC3:AVER[:STATe] ON) before attempting to query results.

Query Response

- Numeric data transferred as ASCII bytes in <NR3> format.

The number of significant digits will range from 1 to 15, depending on the measurement resolution.

- Numbers are separated by commas. The ordering of numbers within the response is mean, standard deviation, minimum,
and maximum.

- If :CALC3:AVER[:STATe] is OFF, four comma-separated Not a Number **9.91E37** values are returned and error -221 is generated.

- If no valid result exists, four comma-separated Not a Number **9.91E37** values are returned and error -230 is generated.

- If the current measurement is Totalize or Voltage Peaks, Not a Number **9.91E37** is returned and error -221 is generated.

Comments

- Query only.
- The last calculated result remains valid until a new computation is made or a relevant instrument state is modified.

Related Front-Panel keys

Stats

:CALCulate3:AVERage:CLEar

This command is an event that causes the Counter to:

- invalidate the statistics results,
- clear the statistics current count to 0, and
- report the negative status condition (NOT Computing Statistics) to bit 8 of the Operation Status Register.

Comments

If :CALC3:AVER[:STATe] is OFF, error -221 is generated.

:CALCulate3:AVERage:COUNt <numeric_value>

Sets or queries the number of measurements to combine for statistics processing.

After :CALC3:AVER:COUNt measurements is reached, a new set of :CALC3:AVER:COUNt measurements must be acquired before another statistics computation will occur.

<numeric_value> Range

2 to 1,000,000

<numeric_value> Resolution

1

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 100

Related Front-Panel keys Stats

:CALCulate3:AVERage:COUNt:CURRent?

Queries the current count (that is, the number of data values collected for statistical computation).

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 1,000,000.
- If :CALC3:AVER[:STATe] is OFF, error -221 is generated.
- If the current measurement is Totalize or Voltage Peaks, 0 is returned and error -221 is generated.

Comments

- Query only.
- No statistics results exist until the :CALC3:AVER:COUN:CURR? is equal to the specified :CALC3:AVER:COUN.

Related Front-Panel keys Stats

:CALCulate3:AVERage[:STATe] <Boolean>

Sets or queries the statistics post-processing enable.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- When this enable is ON, and :TRIG:COUN:AUTO is ON, and [:SENS]:FUNC[:ON] is not Totalize or Voltage Peak function, then :INIT[:IMM] initiates a complete block of measurements. See :TRIG:COUN:AUTO in this chapter for specifics.
- When this enable is OFF, :INIT[:IMM] always initiates a single measurement.

Related Front-Panel keys Stats

:CALCulate3:AVERage:TYPE MAXimum | MINimum | SDEViation | SCALar or MEAN

Selects which statistical result will appear:

- in the :CALC3:DATA? response, and
- on the front-panel display when :DISP[:WIND]:TEXT:FEED is set to "CALC3".

Query Response A sequence of ASCII-encoded bytes: MAX, MIN, SDEV, or MEAN

Comments

- *RST: MEAN
- If :DISP[:WIND]:TEXT:FEED is "CALC3", then this command updates the display immediately.

Related Front-Panel keys Stats

:CALCulate3:DATA?

Queries the statistical result specified by :CALC3:AVER:TYPE.

Enable statistics (:CALC3:AVER[:STATe] ON) before attempting to query results.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.
- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15, depending on the measurement resolution. Only significant digits will be returned.
- If :CALC3:AVER[:STATe] is OFF, Not a Number **9.91E37** is returned and error -221 is generated.
- If no valid result exists, Not a Number **9.91E37** is returned and error -230 is generated.
- If the current measurement is Totalize or Voltage Peaks, Not a Number **9.91E37** is returned and error -221 is generated.

Comments

- Query only.
- The last calculated result remains valid until a new computation is made or a relevant instrument state is modified.

:CALCulate3:FEED "[:]CALCulate[1]"

Sets or queries the data flow to be fed into the CALCulate3 block.

Query Response The string "CALC" is returned.

Comments *RST: "CALCulate[1]"

:CALCulate3:LFILter Subtree

This subtree collects together the commands used to specify *which* measurements will be used in computing statistics; out-of-limit measurements can be filtered out of the statistics processing.

NOTE Not until :CALCulate3:LFILter:STATe is set to ON will any of the :CALCulate3:LFILter settings be used.

:CALCulate3:LFILter:LOWer[:DATA] <numeric_value> [HZ | S |DEG]

Sets or queries the statistics filter lower limit.

If limit filtering is enabled (:CALC3:LFIL:STAT ON), any measurements below this value will not be combined into the statistics computation.

If math is enabled (:CALC:MATH:STATe ON), the limit value specified should take into account that the filtering is on measurements that have been scaled and offset.

<numeric_value> Range -9.9999990000E+12 to -1.0000000000E-13, 0.0000000000, +1.0000000000E-13 to +9.9999990000E+12.

<numeric_value> Resolution 11 digits

Query Response Numeric data transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 0.0000000000
- This command couples :CALC2:LIM:LOW to the same value.
- Updating the lower limit value causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel **Enter** key, while this value is in the 11-digit display, will update the actual value to the displayed (rounded) value.

:CALCulate3:LFILter:STATe <Boolean>

Sets or queries the statistics filter enable. When set to ON, only measurements (scaled and offset if math is enabled) which are within the filter limits are combined into the statistics processing. When set to OFF, all measurements, whether they are within or without the filter limits are combined into the statistics processing.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments *RST: OFF

Related Front-Panel Keys Stats

:CALCulate3:LFILter:UPPer[:DATA] <numeric_value> [HZ | S | DEG]

Sets or queries the statistics filter upper limit.

If limit filtering is enabled (:CALC3:LFIL:STAT ON), any measurements above this value will not be combined into the statistics computation.

If math is enabled (:CALC:MATH:STATe ON), the limit value specified should take into account that the filtering is on measurements that have been scaled and offset.

<numeric_value> Range

-9.9999990000E+12 to -1.0000000000E-13, 0.0000000000, +1.0000000000E-13 to +9.9999990000E+12.

<numeric_value> Resolution

11 digits

Query Response

Numeric data transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 0.0000000000
- This command couples :CALC2:LIM:UPP to the same value.
- Updating the upper limit value causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel **Enter** key, while this value is in the 11-digit display, will update the actual value to the displayed (rounded) value.

:CALCulate3:PATH?

Queries the order in which CALCulate3 sub-blocks are to be processed.

For the Counter, this sequence is fixed to be LFILter followed by AVERage.

Query Response

A sequence of ASCII-encoded bytes: LFIL, AVER

Comments

Query only.

:CALibration Subsystem

:CALibration[:ALL]?

This query causes an internal interpolator self-calibration.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- A value of zero indicates the calibration completed without error. A value of one indicates the calibration completed with error.

Comments Query only.

:CALibration:COUNt?

Queries the number of times the Counter has been calibrated.

By monitoring the calibration count, you can determine whether an unauthorized calibration has been performed.

The following commands (as well as the front-panel invoked calibrations) increment the count upon the completion of a successful calibration:

```
:DIAG:CAL:INP[1|2]:GAIN:AUTO  ONCE
:DIAG:CAL:INP[1|2]:OFFS:AUTO  ONCE
:DIAG:CAL:ROSC:AUTO  ONCE
:DIAG:CAL:TINT:FINE[1|2|3|4]
:DIAG:CAL:TINT:QUICk
```

The :CAL:DATA command also increments the calibration count.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- Query only.
- The calibration count is stored in non-volatile memory, thus cycling power will not reset value.
- The calibration count is unaffected by power-on, save/recall, and *RST.
- The calibration count increments up to a maximum of 32,767 after which it wraps around to 1. (A value of 0 indicates no calibration has been performed since the last reset of the non-volatile memory.)
- Your Counter was calibrated before it left the factory. When you receive your Counter, read the calibration count to determine its initial value.

- Early versions of the Counter do not support this query.

Related Front-Panel Key Scale & Offset / POWER (Calibration Menu)

:CALibration:DATA <arbitrary block>

Sets or queries the calibration data (input gain, input offset, reference oscillator, and time interval).

Before performing calibration, it is a good idea to query (:CAL:DATA?) and store the current calibration values in your program or on a disk in case an error occurs during the calibration process. See the sample program "How to Read and Store Calibration Information" in Chapter 3, "Programming Your Universal Counter for Remote Operation."

Query Response Definite Length Block.

Comments

- The query response will be **#256<56 calibration-data bytes>** terminated with a new line and EOI.

- This command does not affect the interpolator calibration data.

- If the <arbitrary block> command parameter has the incorrect number of bytes or does not checksum, error -220 is generated.

- If the update to EEPROM fails, error +2013 is generated.

- The calibration data (updated by this command) is stored in non-volatile memory, so cycling power will not reset these values. The only way to update the calibration data is through this command or by initiating the individual calibrations (see :DIAG:CAL: ...).

- The calibration data (updated by this command) is unaffected by power-on, save/recall, and *RST.

:CALibration:SECurity Subtree

This subtree provides capabilities related to the security of the Counter's calibration factors.

Note, early versions of the Counter do not support any of the :CALibration:SECurity commands.

:CALibration:SECurity:CODE <NRf>

Sets the calibration security code.

To change the security code, the Counter must first be unsecured. To unsecure the Counter, use the :CALibration:SECurity:STATe command.

<NRf> Range 0 to 9999999

<NRf> Resolution 1

Comments

- No query.
- The calibration code is stored in non-volatile memory, and is unaffected by power-on, save/recall, and *RST.

Related Front-Panel key Scale & Offset / POWER (Calibration Menu)

:CALibration:SECurity:STATe <Boolean>, <NRf>

Sets and queries the calibration security state.

To unsecure for calibration, specify OFF with the present security code. When the Counter is unsecure, any calibration can be performed.

To secure against calibration, specify ON with the present security code. When the Counter is secure, no calibration can or will be performed (except for interpolator calibration).

<NRf> Range 0 to 9999999

<NRf> Resolution 1

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates the Counter is unsecure; a value of 1 indicates the Counter is secure.

Comments

- The calibration state is stored in non-volatile memory, and is unaffected by power-on, save/recall, and *RST.

- The security code is set to 53131 or 53132 (depending on which model you have) when the Counter is shipped from the factory.
If you forget your security code, you can reset the security code to the model-number default by resetting all of the non-volatile memory to a default state. See the Assembly-Level Service Guide for more information.

Related Front-Panel Keys

Scale & Offset / POWER (Calibration Menu)

:CONFigure Subsystem

Refer to the Measurement Instructions section on page 4-52 in this chapter for a description of :CONFigure.

Device Clear

Device Clear

The full capability of the Device Clear IEEE 488.1 interface function is implemented in the Counter. This function allows a device to be initialized to a cleared state. The device-dependent effect is described below.

In response to either the Device Clear message or the Selected Device Clear message, the Counter:

- clears the input buffer and Output Queue,
- resets the parser, execution control, and response formatter,
- clears any command that would prevent processing a *RST or other commands,
- disables the effect of a prior *OPC command, and
- terminates the holdoff action of a *WAI, *OPC?, or data query (:MEASure query, :READ query, :FETCh query, :CALC:DATA?, :CALC2:LIM:FAIL?) waiting for pending operation to complete.

Also, a front-panel initiated diagnostic or calibration may be aborted (for example, if the front-panel diagnostic or calibration is waiting for user input).

:DIAGnostic Subsystem

This subsystem controls the remote calibration of the Counter.

All of the calibration values, with the exception of the interpolator values, are stored in non-volatile memory and are unaffected by power-on, save/recall, and *RST.

Any of the commands which perform a calibration, with the exception of the interpolator calibration, will generate error -221 if the user tries to execute a calibration while the Counter is secured. (Note, this will not occur in early revisions of the Counter because calibration security does not exist.) Please refer to the :CALibration:SECurity subtree for command specifics regarding calibration security.

:DIAGnostic:CALibration:INPut[1|2]:GAIN: AUTO ONCE | OFF

Calibrates the channel 1 or 2 input trigger GAIN when the ONCE parameter is used.

Before sending this command, connect a +5V source to the appropriate input.

Query Response

A sequence of ASCII-encoded bytes: OFF

Comments

- The calibration values are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.
- Use :DIAG:CAL:STAT? to check for successful calibration.
- After calibration is completed, the state of this command's parameter is OFF.

Related Front-Panel Keys

Scale and Offset/POWER (Calibration menu).

:DIAGnostic:CALibration:INPut[1|2]:OFFSet: AUTO ONCE | OFF

Calibrates the channel 1 or 2 input trigger OFFSet when the ONCE parameter is used.

Before sending this command, BE SURE to disconnect any input signal from the appropriate input.

Query Response A sequence of ASCII-encoded bytes: OFF

Comments

- The calibration values are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.
- Use :DIAG:CAL:STAT? to check for successful calibration.
- After calibration is completed, the state of this command's parameter is OFF.

Related Front-Panel Keys Scale and Offset/POWER (Calibration menu).

:DIAGnostic:CALibration:INTerpolator:AUTO ONCE | OFF |ON

Calibrates the interpolator circuit in the Counter when the ONCE parameter is used.

AUTO ON enables automatic interpolator calibration on every measurement. AUTO OFF disables automatic interpolator calibration.

Query Response A sequence of ASCII-encoded bytes: OFF or ON

Comments

- *RST: ON
- Use :DIAG:CAL:STAT? to check for successful calibration.
- After ONCE calibration is completed, the state of this command's parameter is OFF.
- When :DIAG:CAL:INT is set to OFF, the Counter reports the positive status condition (questionable Time, Frequency, and Phase) to bits 2, 5, and 6 of the Questionable Status Register. When :DIAG:CAL:INT is set to ON, the Counter reports the negative status condition (NOT questionable Time, Frequency, and Phase) to bits 2, 5, and 6 of the Questionable Status Register.
- This enable is unaffected by save/recall.

:DIAGnostic:CALibration:ROSCillator:AUTO ONCE | OFF

Calibrates the reference oscillator when ONCE parameter is used.

Before sending this command, connect 10 MHz to channel 1.

Query Response A sequence of ASCII-encoded bytes: OFF

Comments

- This command is available only if the instrument contains the medium or high stability oscillator option; otherwise, error -241 is generated.
- The calibration values are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.
- Use :DIAG:CAL:STAT? to check for successful calibration.
- After calibration is completed, the state of this command's parameter is OFF.

Related Front-Panel Keys Scale and Offset/POWER (Calibration menu).

:DIAGnostic:CALibration:STATus?

Queries pass/fail status of the last calibration. It can be used after any calibration to determine if the calibration was successful.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- A value of zero indicates that calibration completed without error. A value of one indicates the calibration completed with error.

Comments Query only.

:DIAGnostic:CALibration:TINTerval:FINE[1|2|3|4]

These event commands, when performed in the appropriate order and with the appropriate calibration signals supplied to both channels, calibrate out the differences in electrical path length between Channel 1 and Channel 2.

Before sending any of these commands, the appropriate calibration signal ***must*** be supplied to both channels. Please see the section titled "To Initiate the Calibration Routine" in Chapter 2 of the Operating Guide for details involved in providing the calibration signals. The Operating Guide also discusses the difference between this calibration and the alternate (simpler) choice, :DIAG:CAL:TINT:QUICk.

The command :DIAG:CAL:TINT:FINE1 invokes the first step of the calibration, :DIAG:CAL:TINT:FINE2 invokes the second, and so on. Each step corresponds to the calibration signals produced by each of the four numbered buttons on the Agilent 59992A J06 Time Interval Calibrator.

NOTE

This is a four-step calibration—all four steps must be successfully completed in sequence, starting with step one and ending with step four. To restart the calibration (from step one), power cycle the Counter. A good precaution is to use an GPIB program (see the programming example titled "To Read and Store Calibration Data" in Chapter 3 of the Programming Guide) to read and store the calibration factors prior to initiating ***any*** calibration(s).

Comments

- No query.
- The calibration values are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.
- Use :DIAG:CAL:STAT? to check for successful calibration, after all four steps have been completed.
- Note, early versions of the Counter do not support these commands.

Related Front-Panel Keys

Scale & Offset / POWER (Calibration Menu)

:DIAGnostic:CALibration:TINTerval:QUICk

This event command calibrates out the differences in electrical path length between channels 1 and 2.

Before sending this command, connect to channel 1 a square wave of approximate frequency 10 MHz, but more importantly with a rapid rise time.

Comments

- No query.
- The calibration values are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.
- Use DIAG:CAL:STAT? to check for successful calibration.

Related Front-Panel Keys

Scale and Offset/POWER (Calibration menu).

:DISPlay Subsystem

This subsystem controls the selection and presentation of textual information on the Counter's display. This information includes measurement results. :DISPlay is independent of, and does not modify, how data is returned to the controller.

See the section titled "How to Program the Counter to Display Results" in Chapter 3 of this guide.

:DISPlay:ENABle <Boolean>

Sets or queries whether the whole display (text area, annunciators, and indicators—with the exception of Remote and SRQ) is visible.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: ON
- This value is unaffected by save/recall.

:DISPlay:MENU[:STATe] OFF

This command, which only allows the OFF parameter, disables the menu display. When the menu display is disabled, the results display appears.

The query indicates whether the menu display or result display is enabled.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates the menu display is disabled (the result display is enabled). A value of 1 indicates the menu display is enabled (the result display is disabled).

Comments

- *RST: OFF
- To enable the menu display, use either the front-panel keys or the :SYST:KEY command.
- This value is unaffected by save/recall.

:DISPlay[:WINDow]:TEXT:FEED "[:]CALCulate2" | "[:]CALCulate3"

Sets or queries what data flow is fed into the display.

Choose from the following <data_handle> strings:

- "[:]CALCulate2"—should be used to direct any result *other than the statistics* to the result display

  Specifically this would select one of the following results for the result display:

  – raw measurement (if math is disabled—:CALC:MATH:STAT OFF)

  – the scaled/offset measurement (if math is enabled—:CALC:MATH:STAT ON)

  – the limit graph (if limit testing is enabled with the graphic display—:CALC2:LIM:STAT ON, :CALC2:LIM:DISP GRAPh)

- "[:]CALCulate3"—should be used to direct the statistical result (if statistics are enabled, :CALC3:AVER[:STAT] ON) to the result display; the particular statistic displayed is determined by :CALC3:AVER:TYPE.

Query Response A string is returned: "CALC2" or "CALC3."

Comments

- *RST: ":CALCulate2"
- Refer to the section titled "How to Program the Counter to Display Results" in Chapter 3 of this guide.

Related Front-Panel Keys Stats

:DISPlay[:WINDow]:TEXT:RADix COMMa | DPOint

Sets or queries the character used to separate integral and fractional portions of a displayed number.

To conform to the numerical convention used in the USA, specify decimal point with DPOint. To conform to the numerical convention used in many other countries, specify COMMa.

For example:

With DPOint, one thousand is displayed as 1,000.0
With COMMa, one thousand is displayed as 1.000,0

Query Response A sequence of ASCII-encoded bytes: DPO or COMM

Comments This value is stored in non-volatile memory. It is unaffected by power-on, save/recall, and *RST.

Related Front-Panel Keys Utility/POWER

:FETCh Subsystem

Refer to the Measurement Instructions section on page 4-52 in this chapter for a description of :FETCh.

:FORMat Subsystem

This subsystem sets the data format for transferring numeric information. This data format is used for response data by those commands that are specifically designated to be affected by the :FORMat subsystem.

:FORMat[:DATA] ASCii | REAL

Sets or queries the data format type. Valid types are ASCii and REAL.

When ASCii type is selected, numeric response data is transferred as ASCII bytes in <NR3> format. The numbers are separated by commas as specified in IEEE 488.2. To indicate that no response data exists, Not a Number **9.91E37** is returned.

When REAL type is selected, response data is transferred in a <definite length block> as a 64-bit IEEE 754 floating point number. To indicate that no response data exists, Not a Number **9.91E37** is returned in the <definite length block>.

Query Response A sequence of ASCII-encoded bytes: ASC or REAL

Comments

- *RST: ASCii
- This command affects the response format of the following commands:

  :CALCulate:DATA?
  :CALCulate3:DATA?
  :FETCh?
  :MEASure query
  :READ?
  [:SENSe]:DATA?
  :TRACe[:DATA] query

Group Execute Trigger (GET)

The full capability of the Group Execute Trigger IEEE 488.1 interface function is implemented in the Counter. This function permits the Counter to have its operation initiated over the Bus. The device-dependent result of this triggering is described in the following paragraph.

In response to the IEEE 488.1 Group Execute Trigger (GET) remote interface message (while the Counter is addressed to listen), the Counter performs the action defined by the *DDT command (see page 4-124).

:HCOPy Subsystem

:HCOPy:CONTinuous <Boolean>

Enables or disables printing results.

When :HCOPy:CONTinuous is enabled (:HCOP:CONT ON), the Counter prints each measurement.

If statistics is enabled (:CALC3:AVER[:STAT] ON), all statistics (standard deviation, mean, minimum, and maximum) will be printed in addition to the individual measurements. If limit testing is enabled (:CALC2:LIMit:STAT ON), an indication will be printed for the measurements that fail the limit test.

Refer to the sections titled “Using the Print Menu,” and “To Configure the RS-232 Serial Port for Printing” in the Operating Guide for more details on printing.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF, a value of 1 indicates ON.

Comments

*RST: OFF

Related Front-Panel Keys

Save & **Print**

:INITiate Subsystem

This subsystem controls the initiation of a measurement.

:INITiate:AUTO <Boolean>

Sets or queries if the Counter should stop measurements or continue measuring (go on) when a measurement exceeds the user-entered limits.

AUTO ON configures the Counter to automatically stop measuring (set :INIT:CONT to OFF) on a limit test failure (that is, out-of-limit results are detected). AUTO OFF configures the Counter to continue measuring (leave :INIT:CONT unaffected) when the limit test fails.

The AUTO ON capability is only meaningful when the Counter is limit testing (:CALC2:LIM:STAT is ON) and :INIT:CONT is ON.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments *RST: OFF

Related Front-Panel Keys Limit Modes

:INITiate:CONTinuous <Boolean>

Sets or queries the enable for continuously initiated measurements.

With CONTinuous set to OFF, no measurements are made until CONTinuous is set to ON or :INITiate[:IMMediate] is received.
Once CONTinuous is set to ON, a new measurement is initiated.
On the completion of each measurement, with CONTinuous ON, another measurement immediately commences.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF

- When the :INIT:CONT ON command is sent, the Counter:

  – invalidates the statistics results,
  – clears the statistics current count to 0,
  – reports the negative status condition (NOT Computing Statistics) to bit 8 of Operation Status Register.

- When :CALC2:LIM:CLE:AUTO is ON, the Counter performs the following whenever the :INIT:CONT ON command is sent:

  – invalidates the limit data,
  – clears :CALC2:LIM:FAIL, :CALC2:LIM:FCOunt, and :CALC2:LIM:PCOunt information,
  – turns off the front-panel display's **Limit** annunciator, and
  – sets the Limit-Detect output to the in-limit voltage level.

- When [:SENS]:EVEN:LEV[:ABS]:AUTO is ON, the Counter performs an auto-trigger on the measurement channel(s) whenever the :INIT:CONT ON command is executed, and also at the beginning of each measurement cycle while :INIT:CONT is ON.

- The state of :TRIG:COUN:AUTO has no affect on the operation of :INIT:CONT ON.

- :INIT:CONT ON operates as if :TRIG:COUN was 1.

- The commencement of the first measurement due to setting :INITiate:CONTinuous to ON sets the Pending Operation Flag to true. The Pending Operation Flag is set false by aborting of a measurement, or by the completion of the last measurement after :INITiate:CONTinuous is set OFF.

- With the measurements being made continuously, the :ABORt command shall abort the current measurement in progress, however, the value of :INITiate:CONTinuous is unaffected.
If CONTinuous was set to ON prior to receiving :ABORt, it remains ON and a new measurement begins.

- When a single measurement is in progress (:INIT:CONT is OFF):
  - Error -213 (Init ignored) is generated and the state of INIT:CONT is unaffected by :INIT:CONT ON.
  - Error -210 (Trigger error) is generated by INIT:CONT OFF.
- Note that the Counter powers up with :INIT:CONT set to ON, but *RST sets :INIT:CONT to OFF.

Related Front-Panel Keys Run

:INITiate[:IMMediate]

This event command causes the instrument to initiate either a single measurement or a block of measurements.

When

:TRIG:COUN:AUTO is OFF, or
:CALC3:AVER[:STAT] is OFF, or
[:SENS]:FUNC[:ON] is Totalize function, or
[:SENS]:FUNC[:ON] is any Voltage Peaks function,

then :INIT[:IMM] initiates a single measurement.

When

:TRIG:COUN:AUTO is ON, and
:CALC3:AVER[:STAT] is ON, and
[:SENS]:FUNC[:ON] is not Totalize or Voltage Peak function,

then :INIT[:IMM] initiates a complete block of measurements. See :TRIG:COUN:AUTO for specifics.

Comments

● When :TRIG:COUN:AUTO is ON and :CAL3:AVER[:STAT] is ON, the Counter clears the statistics results and the statistics current count on :INIT[:IMM].

● If the instrument is already in the process of making a measurement or if INITiate:CONTinuous is set to ON, an :IMMediate command has no affect, and an error -213
(Init ignored) is generated.

● When :CALC2:LIM:CLE:AUTO is ON, the Counter performs the following whenever the :INIT[:IMM] command is sent:

– invalidates the limit data,
– clears :CALC2:LIM:FAIL, :CALC2:LIM:FCOunt, and :CALC2:LIM:PCOunt information,
– turns off the front-panel display's **Limit** annunciator, and
– sets the Limit-Detect output to the in-limit voltage level.

● This command is an overlapped command (see IEEE 488.2, Section 12). Beginning a measurement or block of measurements with an :INITiate[:IMMediate] sets the Pending Operation Flag to true. Completing the measurement or block of measurements (normally or by aborting) sets Pending Operation Flag to false.

● For an automatically armed Totalize measurement, use :ABORt to terminate the measurement initiated with :INIT[:IMM].

● When [:SENS]:EVEN:LEV[:ABS]:AUTO is ON, the Counter performs an auto-trigger on the measurement channel(s) whenever the :INIT[:IMM] command is executed.

Related Front-Panel Keys

Stop/**Single**

:INPut[1|2] Subsystem

This subsystem controls the characteristics of the Counter's input ports. :INPut1 corresponds to channel 1 input port and :INPut2 corresponds to channel 2 input port.

:INPut[1|2]:ATTenuation 1 | 10

Sets or queries the input attenuation.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments *RST: 1

Related Front-Panel Keys X10 Attenuate

:INPut[1|2]:COUPling AC | DC

Sets or queries the input coupling.

Query Response A sequence of ASCII-encoded bytes: AC or DC

Comments *RST: AC

Related Front-Panel Keys DC/AC

:INPut[1|2]:FILTer[:LPASs][:STATe] <Boolean>

Sets or queries the state of the low-pass filter.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments *RST: OFF

Related Front-Panel Keys 100kHz Filter

:INPut[1|2]:FILTer[:LPASs]:FREQuency?

Queries the cutoff frequency of the low-pass filter.

Query Response

- Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.
- A value of 100E+3 is returned.

Comments

Units are Hertz.

:INPut[1|2]:IMPedance <numeric_value> [OHM]

Sets or queries the input impedance (50Ω or 1MΩ).

<numeric_value> Range

50 or 1E6

Query Response

Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

Comments

- *RST: 1E6 OHM
- Units are Ohms.

Related Front-Panel Keys

50Ω/1MΩ

:INPut3 Subsystem

This subsystem queries the characteristics of the Counter's channel 3 input port. These commands are only available if Option 030/050 is installed.

:INPut3:COUPling?

Queries the channel 3 input coupling.

Query Response A sequence of ASCII-encoded bytes: AC

Comments This command is only available if Option 030/050 is installed.

:INPut3:IMPedance?

Queries the channel 3 input impedance.

Query Response Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

The value returned is **50**, or Not a Number **9.91E37** if Option 030/050 Channel 3 is not installed.

Comments

- Units are Ohms.
- This command is only available if Option 030/050 is installed.

:MEASure Subsystem

Refer to the Measurement Instructions section on page 4-52 in this chapter for a description of :MEASure.

Measurement Instructions (:CONFigure, :FETCh, :MEASure, :READ)

The purpose of these commands is to acquire data using a set of high-level instructions. These commands are structured to allow you to trade off interchangeability with fine control of the measurement process. The :MEASure query provides a complete capability where the instrument is configured, a measurement is taken, and the results are stored in the Output Queue in one operation.

When more precise control of the measurement is required, the :CONFigure and :READ? commands can be used. :CONFigure performs the configuration portion of the measurement. :READ? performs the data acquisition and post processing (if any), and then it places the results in the Output Queue. This allows generic configuration of the instrument using :CONFigure, and then customization of the measurement with other commands (for example, from the [:SENSe] subsystem). :READ? completes the measurement process.

The :READ? command, in turn, is composed of the :INITiate[:IMMediate] and :FETCh? commands. :INITiate[:IMMediate] performs the data acquisition. :FETCh? performs the post-processing function (if any) and places the result in the Output Queue. This allows more than one FETCh? on a single set of acquired data.

Summary of the Measurement Instruction Commands	
:MEASure query	This command is the simplest to use, but allows few additional possibilities. This command lets the Counter configure itself for an optimal measurement, initiate measurement, and return the result; that is, it provides complete measurement sequence (:MEAS query is equivalent to the :CONF, :INIT, :FETC? command sequence, but with no flexibility.)
:CONFigure :READ?	The combined use of these two commands allows for more control when the Counter performs measurement, initiates measurement, and returns the result. Use this command sequence if you are planning for the Counter to perform something in between the measurement setup and acquisition.
:CONFigure :INITiate :FETCh?	This combination of commands allows for the most flexibility . This command sequence configures the Counter, initiates the measurement as specified, and returns the result.

The <source_list> parameter has the same syntax as SCPI <channel_list> syntax. For example, a one-channel function (such as Frequency, Period, etc.) would use (@1) to specify channel 1, whereas a two-channel function (such as Time Interval, Phase, and Ratio) would use (@1), (@2) to specify a measurement between channel 1 and channel 2.

If the instrument receives a parameter which is unexpected, it shall process the command, ignoring the unexpected parameter, and set the "Command Warning" bit of the Data Questionable status reporting structure.

The response format for :MEASure query, :READ?, and :FETCh? is determined by the :FORMat subsystem. If no valid data is available, error -230 (Data corrupt or stale) is generated.

See the programming example "Easiest Way to Make a Measurement" in Chapter 3 of this guide.

:CONFigure[:SCALar]:<function> <parameters> [,<source_list>]

Configures the instrument to perform the specified function, but does not initiate the measurement. Use :INITiate;FETCh? or :READ? to make and query a measurement.

Parameters (other than <source_list>) may be defaulted from the right by omitting them, or anywhere by substituting the keyword DEFault. The <source_list> parameter may be defaulted by omitting it. The default values are specified by the particular function description.

Note, this command defaults several Counter settings. To simply change the function, while leaving all other Counter settings as they are, use [:SENS]:FUNC[:ON] instead.

Comments

- Refer to the sub-section in this section titled "Descriptions of the Measurement Functions" for descriptions of each measurement function.

- Refer to Table 4-1 in this sub-section for a summary of the <function>, <parameters>, and <source_list> for each of the measurement functions.

- This command disables math, statistics, and limit-testing.

- If an absolute trigger level is not specified in the <parameters>, then when this command executes, for functions other than Voltage Peaks (maximum, minimum, peak-to-peak) or Totalize,

  – auto-trigger is enabled,
  – auto-trigger level(s) are set,
  – auto-trigger is invoked on measurement channel(s).

:CONFigure?

Queries the function configured by the last :CONFigure or :MEASure query.

If the instrument state has changed through commands other than :CONFigure or :MEASure query, the instrument will not track these changes, and the query response will not reflect these changes.

Query Response

- A string of the form: "<function> <parameters>[,<source_list>]", omitting the leading colon from the <function>.

- The Rise Time, Fall Time, and Duty Cycle functions each have two ways of being specified. This query response will use RTIM, FTIM, or DCYC to indicate these functions.

- The response is unaffected by *RST, recall, and [:SENS]:FUNC.

- At power-on, this query generates an error and returns an empty string.

Comments

- Refer to the sub-section in this section titled "Descriptions of Measurement Functions" for descriptions of each measurement function.

- Refer to Table 4-1 in this section for a summary of the <function>, <parameters>, and <source_list> for each of the measurement functions.

FETCh[[:SCALar]:<function>]?

This query returns the measurement taken by the :INITiate (or :MEASure query or :READ?) commands.

When [:SCALar]:<function> is specified, the instrument will retrieve the specified result if it matches the current measurement type or can be derived from the current measurement type. The only functions which can be derived from a different measurement type are:

- frequency to/from period,
- voltage minimum to/from voltage maximum,
- voltage minimum to/from voltage peak-to-peak, and
- voltage maximum to/from voltage peak-to-peak.

When [:SCALar]:<function> is omitted, the function specified/used by the last :CONFigure, :MEASure, :READ, or FETCh will be used, if possible. This behavior is apparent when switching between frequency and period, or when switching among the voltage peaks functions.

Issuing this query while a measurement is in progress has the effect of holding off further commands from being processed until the measurement completes. This hold-off action can only be canceled by the measurement completing, Device Clear, or power-on.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.

- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15, depending on the measurement resolution.

- If no valid results exists, Not a Number **9.91E37** is returned and error -230 is generated.

- If :FETCh? is executed while an automatically armed Totalize measurement *is in progress*, Not a Number **9.91E37** is returned and error -221 is generated. :ABORt should be used to terminate the measurement and avoid this error.

Comments

- Refer to the sub-section in this chapter titled "Descriptions of the Measurement Functions" for descriptions of each measurement function.

- Refer to Table 4-1 in this section for a summary of the <function>, <parameters>, and <source_list> for each of the measurement functions.

:MEASure[:SCALar]:<function>? <parameters> [,<source_list>]

This query provides a complete measurement sequence: configuration, measurement initiation, and query for result. It is used when the generic measurement is acceptable and fine adjustment of Counter settings is unnecessary.

Parameters (other than <source_list>) may be defaulted from the right by omitting them, or anywhere by substituting the keyword DEFault. The <source_list> parameter may be defaulted by omitting it. The default values are specified by the particular function description.

Issuing this query while a measurement is in progress will result in this query aborting the current measurement before initiating the desired measurement, and then waiting for the measurement to complete. Consequently, this has the effect of holding off further commands from being processed until the desired measurement completes. This hold-off action can only be canceled by the measurement completing, Device Clear, or power-on.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.

- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range form 1 to 15, depending on the measurement resolution.

Comments

- Refer to the sub-section in this section titled "Descriptions of the Measurement Functions" for descriptions of each measurement function.

- Refer to Table 4-1 in this section for a summary of the <function>, <parameters>, and <source_list> for each of the measurement functions.

- This command disables math, statistics, and limit-testing.

- If an absolute trigger level is not specified in the <parameters>, then when this command executes, for functions other than Voltage Peaks (maximum, minimum, peak-to-peak) or Totalize,

  – auto-trigger is enabled,
  – auto-trigger level(s) are set,
  – auto-trigger is invoked on measurement channel(s).

:READ[[:SCALar]:<function>]?

This query provides a method of performing a :FETCh? on *fresh* data.

A common application is to use this command in conjunction with a :CONFigure to provide a capability like :MEASure? in which the application programmer is allowed to provide fine adjustments to the instrument state by issuing the corresponding commands between the :CONFigure and :READ?.

When [:SCALar]:<function> is specified, the instrument will retrieve the specified result if it matches the current measurement type or can be derived from the current measurement type. The only functions which can be derived from a different measurement type are:

- frequency to/from period,
- voltage minimum to/from voltage maximum,
- voltage minimum to/from voltage peak-to-peak, and
- voltage maximum to/from voltage peak-to-peak.

When [:SCALar]:<function> is omitted, the function specified/used by the last :CONFigure, :MEASure, :READ, or FETCh will be used,
if possible. This behavior is apparent when switching between frequency and period, or when switching among the voltage peaks functions.

Issuing this query while a measurement is in progress will result in this query aborting the current measurement and idling the measurement cycle before initiating the desired measurement,
and then waiting for the measurement to complete. Consequently, this has the effect of holding off further commands from being processed until the desired measurement completes. This hold-off action can only be canceled by the measurement completing, Device Clear, or power-on.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.

- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15, depending on the measurement resolution.

Comments

- Refer to the sub-section in this section titled “Descriptions of Measurement Functions” for descriptions of each measurement function.

- Refer to Table 4-1 for a summary of the <function>, <parameters>, and <source_list> for each of the measurement functions.

- If :READ? is executed while the Counter is configured to make an automatically armed Totalize measurement, Not a Number **9.91E37** is returned and error -221 is generated. :INITiate and :ABORt should be used to begin and terminate the measurement; :FETCh? should be used to retrieve the terminated measurement result.

Table 4-1. The <function>, associated <parameters> and <source_list> for the Measure Instruction Commands

<function> *	**<parameters>**	**[,<source_list>]****
[:VOLTage]:DCYCle or [:VOLTage]:PDUTycycle	[<reference>]	[(@1)]
[:VOLTage]:FALL:TIME or [:VOLTage]:FTIMe	[<lower_reference> [,upper_reference>]]	[(@1)]
[:VOLTage]:FREQuency	[<expected_value>[,<resolution>]]	[(@1) | (@2) | (@3)]
[:VOLTage]:FREQuency:RATio	[<expected_value> [,<resolution>]]	[(@1), (@2 | @3)] | [(@2 | @3), (@1)]
[:VOLTage]:MAXimum		[(@1) | (@2)]
[:VOLTage]:MINimum		[(@1) | (@2)]
[:VOLTage]:NWIDth	[<reference>]	[(@1)]
[:VOLTage]:PERiod	[<expected_value>[,<resolution>]]	[(@1) | (@2) | (@3)]
[:VOLTage]:PHASe		[(@1), (@2)]
[:VOLTage]:PTPeak		[(@1) | (@2)]
[:VOLTage]:PWIDth	[<reference>]	[(@1)]
[:VOLTage]:RISE:TIME or [:VOLTage]:RTIMe	[<lower_reference> [,upper_reference>]]	[(@1)]
[:VOLTage]:TINTerval		[(@1), (@2)]
[:VOLTage]:TOTalize:CONTinuous***		[(@1)]
[:VOLTage]:TOTalize:TIMed	[<gate_time>]	[(@1)]

---

*The only functions which can be derived (using FETC? or READ?) from the stored data are period to/from frequency, maximum to/from minimum, maximum to/from peak-to-peak and minimum to/from peak-to-peak. All other functions require an acquisition of a new type.

**<source_list> has the same syntax as SCPI <channel _list> syntax. For example, a single-channel function (e.g., frequency, period, etc.) would use (@1) to specify channel 1, where as a two-channel function (e.g., time interval, phase, and frequency ratio) would use (@1), (@2) to specify a measurement between channel 1 and channel 2.

***This <function> is only allowed with :CONFigure.

Descriptions of the Measurement Functions—<function>

This sub-section provides a description of each measurement function (that is, [:VOLTage]:FREQuency, [:VOLTage]:FREQuency:RATio, [:VOLTage]:PERiod, etc.) that can be used with either the :MEASure query or :CONFigure command.

CAUTION

If your Agilent 53131A contians **Firmware Revision 3413**, refer to the subsection titled "Firmware Revision Work-Around Commands" on page 4-77. It is important to refer to this subsection if your counter contains Firmware Revision 3413 because several measurement functions commands will not work with this firmware revision; thus, work-around commands must be used.

The commands that required work-around commands are:

:MEASure[:SCALar][:VOLTage]:MAXimum? <parameters>[<source_list>]

:MEASure[:SCALar][:VOLTage]:MINimum? <parameters>[<source_list>]
:MEASure[:SCALar][:VOLTage]:PTPeak? <Parameters>[<source_list>]

:READ[:SCALar][:VOLTage]:MAXimum?
:READ[:SCALar][:VOLTage]:MINimum?
:READ[:SCALar][:VOLTage]:PTPeak?

:READ?

In each of the following command lines, the MEASure command is used with the measurement function commands.

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:DCYCle? [<reference>][,(@1)]

Measures Duty Cycle.

The measurement arming is coupled to “auto.”

The <reference> specifies the point on the pulse where the duty cycle is determined. This point can be specified as either a percentage or an absolute voltage. The default units are percent.

The <reference> parameter is used to configure the trigger settings. When the <reference> is specified as a percentage (or defaulted), auto-trigger is enabled and auto-trigger levels are set to the specified (or defaulted) reference value. When the <reference> is specified in volts, auto-trigger is disabled and the absolute trigger levels are set to the specified reference value.

<reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	50 PCT

<source_list>

range:	(@1)
default:	(@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:FALL:TIME?
[<lower_reference>[,upper_reference>]][,(@1)]

Measures Fall Time.

The measurement arming is coupled to "auto."

The Counter uses the <lower_reference> and <upper_reference> to select the lower and upper points on the falling edge of an input signal applied to channel 1. These reference points can be specified as either a percentage or an absolute voltage. The default units are percent.

The <lower_reference> and <upper_reference> parameters are used to configure the Channel 1 and Channel 2 trigger settings, respectively. When a reference is specified as a percentage (or defaulted), auto-trigger is enabled and auto-trigger level is set to the specified (or defaulted) value. When a reference is specified in volts, auto-trigger is disabled and the absolute trigger level is set to the specified value.

<lower_reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	10 PCT

<upper_reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	90 PCT

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:FALL:TIME? ***(Cont.)***

<source_list>

range: (@1)

default: (@1)

:MEASure[:SCALar][:VOLTage]:FREQuency?
[<expected_value>[,<resolution>]][, (@1)|(@2)|(@3)]

Measures Frequency.

The measurement arming mode is set to “digits.” The Counter uses the <expected_value> and <resolution> parameters to configure the number of digits of resolution arming setting.

The Channel 1 or Channel 2 trigger settings are coupled so that the measurement channel has auto-trigger enabled at 50% with a positive slope.

Ch1, Ch2
<expected_value>

range: .100 Hz to 225 MHz

resolution: <expected value> should be within 10% of input frequency for optimum arming configuration

default: 10 MHz

Ch1, Ch2 <resolution>

description: value indicates decade corresponding to least significant digit of the result

range: 1E-16 to 1E6 Hz

value which indicates 3 to 15 digits of resolution for the specified <expected value>

resolution: <resolution> should use a mantissa of 1.0 and be an even power of 10

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:FREQuency? ***(Cont.)***

default: value which indicates 4 digits of resolution for the specified <expected_value>

Ch3 <expected_value>

range: 100 MHz to 3.00 GHz

resolution: <expected_value> should be within 10% of input frequency for optimum arming configuration

default: 500 MHz

Ch3 <resolution>

description: value indicates decade corresponding to least significant digit of the result

range: 1E-7 to1E7 Hz

value which indicates 3 to 15 digits of resolution for the specified <expected value>

resolution: <resolution> should use a mantissa of 1.0 and be an even power of 10

default: value which indicates 4 digits of resolution for the specified <expected_value>

<source_list>

range: (@1) | (@2) | (@3)

default: (@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:FREQuency:RATio?
[<expected_value>[,<resolution>]]
[, (@1), (@2) | (@1), (@3) | (@2), (@1) | (@3), (@1)]

Measures Frequency Ratio between two inputs.

The measurement arming mode is set to “digits.” The Counter uses the <expected_value> and <resolution> parameters to configure the number of digits of resolution arming setting.

The Channel 1 and possibly Channel 2 trigger settings are coupled so that the measurement channels have auto-trigger enabled at 50% with a positive slope.

<expected_value>

range for $\frac{Ch1}{Ch2}$, $\frac{Ch1}{Ch3}$:	1.00E-10 to 1.00E11
range for $\frac{Ch2}{Ch1}$, $\frac{Ch3}{Ch1}$:	1.00E-11 to 1.00E10
resolution:	<expected_value> should be within 10% of ratio for optimum arming configuration
default:	1

<resolution>

description:	value indicates decade corresponding to least significant digit of the result
range for $\frac{Ch1}{Ch2}$, $\frac{Ch1}{Ch3}$:	1E-25 to 1E8 value which indicates 3 to 15 digits of resolution for the specified <expected>
range for $\frac{Ch2}{Ch1}$, $\frac{Ch3}{Ch1}$:	1E-26 to 1E7 value which indicates 3 to 15 digits of resolution for the specified <expected>

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:FREQuency:RATio? ***(Cont.)***

resolution: <resolution> should use a mantissa of 1.0 and be an even power of 10

default: value which indicates 4 digits of resolution for the specified <expected_value>

<source_list>

range: (@1), (@2) | (@1), (@3) | (@2), (@1) | (@3), (@1)

default: (@1), (@2)

:MEASure[:SCALar][:VOLTage]:MAXimum? [(@1)|(@2)]

Measures Voltage Maximum.

<source_list>

range: (@1) | (@2)

default: (@1)

:MEASure[:SCALar][:VOLTage]:MINimum? [(@1)|(@2)]

Measures Voltage Minimum.

<source_list>

range: (@1) | (@2)

default: (@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:NWIDth? [<reference>][,(@1)]

Measures Negative Pulse Width.

The measurement arming is coupled to “auto.”

The <reference> specifies the point on the pulse where the negative pulse width is determined. This point can be specified as either a percentage or an absolute voltage. The default units are percent.

The <reference> parameter is used to configure the trigger settings. When the <reference> is specified as a percentage (or defaulted), auto-trigger is enabled and auto-trigger levels are set to the specified (or defaulted) reference value. When the <reference> is specified in volts, auto-trigger is disabled and the absolute trigger levels are set to the specified reference value.

<reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	50 PCT

<source_list>

range:	(@1)
default:	(@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:PERiod?
[<expected_value>[,<resolution>]][, (@1)|(@2)|(@3)]

Measures Period.

The measurement arming mode is set to "digits." The Counter uses the <expected_value> and <resolution> parameters to configure the number of digits of resolution arming setting.

The Channel 1 or Channel 2 trigger settings are coupled so that the measurement channel has auto-trigger enabled at 50% with a positive slope.

Ch1, Ch2
<expected_value>

range:	4.4 ns to 10.0 sec
resolution:	<expected value> should be within 10% of input period for optimum arming configuration
default:	100 ns

Ch1, Ch2 <resolution>

description:	value indicates decade corresponding to least significant digit of the result
range:	1E-23 to 1E-2 sec value which indicates 3 to 15 digits of resolution for the specified <expected value>
resolution:	<resolution> should use a mantissa of 1.0 and be an even power of 10
default:	value which indicates 4 digits of resolution for the specified <expected_value>

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:PERiod? *(Cont.)*

Ch3<expected_value>

range: 0.33 ns to 10.0 ns

resolution: <expected_value> should be within 10% of input period for optimum arming configuration

default: 2 ns

Ch3 <resolution>

description: value indicates decade corresponding to least significant digit of the result

range: 1E-24 to 1E-11 sec

value which indicates 3 to 15 digits of resolution for the specified <expected value>

resolution: <resolution> should use a mantissa of 1.0 and be an even power of 10

default: value which indicates 4 digits of resolution for the specified <expected_value>

<source_list>

range: (@1) | (@2) | (@3)

default: (@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:PHASe? [(@1),(@2)]

Measures Phase.

The measurement arming is coupled to "auto."

The trigger settings are coupled so that both channels 1 and 2 have auto-trigger enabled at 50% with a positive slope.

<source_list>

range:	(@1), (@2)
default:	(@1), (@2)

:MEASure[:SCALar][:VOLTage]:PTPeak? [(@1)|(@2)]

Measures Peak-to-Peak Voltage.

<source_list>

range:	(@1) | (@2)
default:	(@1)

:MEASure[:SCALar][:VOLTage]:PWIDth? [<reference>][,(@1)]

Measures Positive Pulse Width.

The measurement arming is coupled to "auto."

The <reference> specifies the point on the pulse where the positive pulse width is determined. This point can be specified as either a percentage or an absolute voltage. The default units are percent.

The <reference> parameter is used to configure the trigger settings. When the <reference> is specified as a percentage (or defaulted), auto-trigger is enabled and auto-trigger levels are set to the specified (or defaulted) reference value. When the <reference> is specified in volts, auto-trigger is disabled and the absolute trigger levels are set to the specified reference value.

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:PWIDth? [<reference>][,(@1)] ***(Cont.)***

<reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	50 PCT

<source_list>

range:	(@1)
default:	(@1)

:MEASure[:SCALar][:VOLTage]:RISE:TIME? [<lower_reference>[,upper_reference>]][(,@1)]

Measures Rise Time.

The measurement arming is coupled to “auto.”

The Counter uses the <lower_reference> and <upper_reference> to select the lower and upper points on the rising edge of an input signal applied to channel 1. These reference points can be specified as either a percentage or an absolute voltage. The default units are percent.

The <lower_reference> and <upper_reference> parameters are used to configure the Channel 1 and Channel 2 trigger settings, respectively. When a reference is specified as a percentage (or defaulted), auto-trigger is enabled and auto-trigger level is set to the specified (or defaulted) value. When a reference is specified in volts, auto-trigger is disabled and the absolute trigger level is set to the specified value.

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:RISE:TIME? *(Cont.)*

<lower_reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	10 PCT

<upper_reference>

percent range:	0 to 100 [PCT]
percent resolution:	10%
voltage range:	For volts if X1 Attenuation: -5.125V to +5.125V For volts if X10 Attenuation: -51.25V to +51.25V
voltage resolution:	For volts if X1 Attenuation: .005V For volts if X10 Attenuation: .05V
default:	90 PCT

<source_list>

range:	(@1)
default:	(@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:TINTerval? [(@1),(@2)]

Measures Time Interval.

The first channel in the channel list is the start channel and the second is the stop channel.

The trigger settings are coupled so that both channels 1 and 2 have auto-trigger enabled at 50% with a positive slope.

The measurement arming is coupled to "auto" with no delay.

The input routing is coupled to separate inputs.

<source_list>

range:	(@1), (@2)
default:	(@1), (@2)

:CONFigure[:SCALar][:VOLTage]:TOTalize:CONTinuous [(@1)]

Configures the Counter for Totalize with "auto" arming.

The trigger settings are coupled so that the measurement channel has auto-trigger disabled with a positive slope.

Use :ABORt to terminate the measurement before querying for the totalize result.

This function (continuous totalize, also known as automatically-armed totalize) cannot be used with the :MEASure or :READ queries.
The :FETCh? query, without an explicit function specifier, will retrieve the result if the measurement has been properly terminated with :ABORT.

<source_list>

range:	(@1)
default:	(@1)

Descriptions of the Measurement Functions—<function> (Cont.)

:MEASure[:SCALar][:VOLTage]:TOTalize:TIMed? [<gate_time>][,(@1)]

Measures Totalize during the specified <gate time>.

The totalize gate mode is set to "time," and the totalize gate time is defaulted or set to the specified value.

The trigger settings are coupled so that the measurement channel has auto-trigger disabled with a positive slope.

<gate_time>

range:	For short gate time: 100E-5 to 999E-5 seconds For long gate time: 10E-3 to 1000.000 seconds
resolution:	For short gate time: 1E-5 seconds For long gate time: 1E-3 seconds
default:	.100 second

<source_list>

range:	(@1)
default:	(@1)

How to Use the Measurement Instruction Commands

The Measure Instruction commands have a different level of compatibility and flexibility than other commands. The parameters used with commands from the Measure Instruction describe the signal you are going to measure. This means that the Measure Instructions give compatibility between instruments since you do not need to know anything about the instrument you are using.

Using :MEAsure

This is the simplest Measurement Instruction command to use, but it does not offer much flexibility. :MEASure causes the Counter to configure itself for a default measurement, starts the measurement, and queries the result. The following example shows how to use query to measure frequency. Use

```
:MEASURE:FREQ?
```

to execute a default frequency measurement and have the result sent to the controller. The Counter will select settings and carry out the required measurement; moreover, it will automatically start the measurement and send the result to the controller.

You may add parameters to give more details about the signal you are going to measure. Use

```
:MEASURE:FREQ?  50 MHZ, 1 HZ
```

where 50 MHz is the expected value, which can of course also be sent as 50E6 HZ, and 1Hz is the required resolution.

Also the channel numbers can be specified if you send, for example:

```
:MEASURE:FREQ?  (@2)
```

```
:MEASURE:FREQ?  50 MHz, 1 HZ, (@2)
```

How to Use the Measurement Instruction Commands (Cont.)

Using :CONFigure with :READ?

The :CONFigure command causes the instrument to choose default settings for the specified measurement. :READ? starts the measurement and queries the result.

This sequence operates in the same way as the :MEASure query, but now it is possible to insert commands between :CONFigure and :READ? to specify a particular setting. For example, use

```
:CONF:FREQ  5 MHZ, 1HZ
```

to configure a default frequency measurement where 1 Hz is the required resolution and 5 MHz is the expected value.

Use

```
:SENS:EVEN:LEV  0V
```

to set the trigger level to 0 Volts.

Use

```
:READ?
```

to start the measurement and query the result.

Using :CONFigure with :INITiate and :FETCh?

The :READ? query is composed of the :INITiate command, which starts the measurement, and the :FETCh? command, which returns the results to the controller. For example, use

```
:CONF:FREQ  50 MHZ, 1 HZ
```

to configure for a default frequency measurement where 1 Hz is the required resolution and 50 MHz is the expected value.

How to Use the Measurement Instruction Commands (Cont.)

Use

:SENS:EVEN:LEV 0V

to set the trigger level to 0 Volts.

Use

:INITIATE

to start the measurement.

Use

:FETCH?

to query for result.

Firmware Revision Work-Around Commands

The following applies to Agilent 53131A instruments with **Firmware Revision 3413**:

The three commands listed below should NOT be used.

:MEASure[:SCALar][:VOLTage]:MAXimum? <parameters>[<source_list>]

:MEASure[:SCALar][:VOLTage]:MINimum? <parameters>[<source_list>]

:MEASure[:SCALar][:VOLTage]:PTPeak? <parameters>[<source_list>]

The work-around commands which should be substituted are listed below.

:CONFigure[:SCALar][:VOLTage]:MAXimum <parameters>[<source_list>]

:INIT; *WAI; FETCh?

:CONFigure[:SCALar][:VOLTage]:MINimum <parameters>[<source_list>]

:INIT; *WAI; FETCh?

:CONFigure[:SCALar][:VOLTage]:PTPeak <parameters>[<source_list>]

:INIT; *WAI; FETCh?

The three commands listed below should NOT be used.

:READ[:SCALar][:VOLTage]:MAXimum?

:READ[:SCALar][:VOLTage]:MINimum?

:READ[:SCALar][:VOLTage]:PTPeak?

The work-around commands which should be substituted are listed below.

:INIT; *WAI

:FETCh:MAXimum?

:INIT; *WAI

:FETCh:MINimum?

:INIT; *WAI

:FETCh:PTPeak?

The command listed below should NOT be used when the function choice is Voltage Peaks.

:READ?

The work-around command sequence which should be substituted is listed below.

:INIT; *WAI

:FETCh?

When the function is set to Voltage Peaks, and the Counter is in SINGLE mode, the GPIB command :FETCh? will hang the GPIB if the measurement has not completed.

The work around inserts a command which enforces correct timing. The work-around command sequence is listed below.

:INIT; *WAI

:FETCh?

When the function is set to Voltage Peaks, and the Counter is in RUN mode, the GPIB command :FETCh? should not be used.

The work around puts the box into SINGLE mode, and inserts a command which enforces correct timing. The work-around command sequence is listed below.

:INIT:CONTinuous OFF

:INIT; *WAI

:FETCh?

:MEMory Subsystem

This subsystem manages the instrument's memory. The MEMory capabilities of an instrument are not part of the instrument state, and are not affected by reset (*RST) or recall (*RCL). In this instrument, the macro capabilities will not survive a power cycle, but the *SAV/*RCL states will.

:MEMory:DELete:MACRo <string>

Deletes the macro with the name specified by the string parameter.

The new IEEE 488.2-1992 command *RMC (Remove Macro Command) may also be used; it performs exactly the same action as :MEMory:DELete:MACRo. Note, however, that the Counter complies with IEEE 488.2-1987.

Comments

- Event; no query.
- See *PMC (page 4-135) if you want to delete all macros.

:MEMory:FREE:MACRo?

Queries the memory usage and availability corresponding to macro data. A total of 6500 bytes is dedicated to macro memory.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Two numbers transferred as ASCII bytes in <NR1> format and comma-separated: <bytes available>, <bytes in use>.

:MEMory:NSTates?

Queries the Number of available *SAV/*RCL STates in the instrument.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- The value returned is 21.
- The response value is one greater than the maximum which can be sent as a parameter to the *SAV and *RCL commands.

[:SENSe] Subsystem

The [:SENSe] subsystem commands are divided into several sections. Each section or subtree deals with controls that directly affect instrument-specific settings and not those related to the signal-oriented characteristics.

[:SENSe]:DATA? ["[:]SENSe[1]"]

Queries the current measurement result data of the :SENSe subsystem (no scale or offset applied).

If this query executes while a measurement is in progress, then the prior measurement result will be returned, if the prior result has not been invalidated.

Query Response

- Result will be formatted according to :FORMat[:DATA] ASCii | REAL setting.

- When ASCii format is used, numeric data is transferred as ASCII bytes in <NR3> format. The number of significant digits will range from 1 to 15 depending on the measurement resolution.

- If no valid result exists, Not a Number **9.91E37** is returned and error -230 is generated.

- If this query is executed while an automatically armed Totalize measurement *is in progress*, Not a Number **9.91E37** is returned and error -221 is generated. :ABORt should be used to terminate the measurement and avoid this error.

Comments

Query only.

[:SENSe]:EVENt[1|2] Subtree

This subtree defines the "trigger event."

For Frequency, this is the event which is counted; for Time Interval, this is the start/stop edge.

These trigger events are fed to the sensor function. In separate, the EVENt1 subtree is fed the INPut1 signal and the :EVENt2 subtree is fed the INPut2 signal; in common, the INPut1 signal is fed to both the EVENt1 and :EVENt2 subtrees.

When you are measuring Voltage Peaks, none of the :SENSe:EVENt settings are used.

[:SENSe]:EVENt2:FEED "[:]INPut[1] | [:]INPut2"

Sets or queries the common/separate enable.

Feeding the :INPut2 subsystem to the [:SENSe]:EVENt2 subsystem corresponds to separate; feeding the :INPut1 subsystem to the [:SENSe]:EVENt2 subsystem corresponds to common.

Query Response The string “INP” or “INP2” is returned.

Comments

- *RST: “INPut2”
- Only the Time Interval function allows *either* the separate or common setting. All of the other functions have a fixed input route setting.

Related Front-Panel Keys Trigger/Sensitivity

[:SENSe]:EVENt[1|2]:HYSTeresis:RELative <numeric_value> [PCT]

The operation of this command is determined, in part, by the firmware revision of the Counter.

Current Firmware Revision

Sets or queries the size of the hysteresis window as a percentage of the allowable hysteresis. For example, 0% is the minimum hysteresis setting and 100% is the maximum hysteresis setting.

Specifying 100% or MAXimum provides the greatest noise immunity (lowest sensitivity), while specifying 0% or MINimum provides the least noise immunity (most sensitive).

Prior Firmware Revisions: 3317, 3335, and 3402

Sets or queries the sensitivity of the input channel as a percentage of the allowable sensitivity. For example, 0% is the minimum sensitivity setting and 100% is the maximum sensitivity setting.

Specifying 100% or MAXimum provides the most sensitivity (least noise immunity), while specifying 0% or MINimum provides the lowest sensitivity (greatest noise immunity).

<numeric_value> Range 0, 50, or 100 PCT

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments Current firmware revision—*RST: 0 PCT (least noise immunity)
Prior firmware revisions—*RST: 100 PCT (maximum sensitivity) (3317, 3335, and 3402)

Related Front-Panel Keys Trigger/Sensitivity

[:SENSe]:EVENt[1|2]:LEVel[:ABSolute] <numeric_value> [V]

Sets or queries the level at the center of the hysteresis window.

The actual trigger event is at the top of the hysteresis window (for POSitive slope) or at the bottom of the hysteresis window (for NEGative slope).

<numeric_value> Range

- X1 Attenuation: -5.125 to +5.125V
- X10 Attenuation: -51.25 to +51.25V

Resolution

- X1 Attenuation: .005V
- X10 Attenuation: .05V

Query Response Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

Comments

- Execution of this command turns [:SENS]:EVEN[1|2]:LEV:AUTO to OFF.
- The query can be used to determine the current trigger level when auto-trigger is enabled ([:SENS]:EVEN[1|2]:LEV[:ABS]:AUTO ON). That is, the query response will indicate what level has automatically been selected.

Related Front-Panel Keys Trigger/Sensitivity

[:SENSe]:EVENt[1|2]:LEVel[:ABSolute]:AUTO <Boolean>

Sets or queries the “auto-trigger“ enable.

When AUTO is set to ON, the Counter automatically measures and computes a trigger level which corresponds to the auto-trigger percentage (specified with [:SENS]:EVEN[1|2]:LEV:REL) of the specified channel.

While the enable is set to ON, the Counter will measure and compute the measurement channel(s) trigger level(s) each time :INIT or :INIT:CONT ON is executed. Also, for each measurement, while the enable is set to ON, the Counter will check that the measurement signal(s) are triggering—if no triggering is found, the Counter will measure and compute new trigger level(s).

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: ON
- Explicitly selecting a trigger level (with [:SENS]:EVEN[1|2]:LEV[:ABS]) turns AUTO OFF.

Related Front-Panel Keys

Trigger/Sensitivity

[:SENSe]:EVENt[1|2]:LEVel:RELative <numeric_value> [PCT]

Sets or queries the percentage of the peak-to-peak range of the signal at which the instrument auto triggers.

If [:SENS]:EVEN[1|2]:LEV[:ABS]:AUTO is ON, then when this command executes, the Counter automatically measures and computes a trigger level corresponding to the specified percentage of the specified channel.

<numeric_value> Range

0 to 100 PCT

Resolution

10 PCT

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 50 PCT
- Only applies when [:SENS]:EVEN[1|2]:LEV[:ABS]:AUTO is ON.

Related Front-Panel Keys Trigger/Sensitivity

[:SENSe]:EVENt[1|2]:SLOPe POSitive | NEGative

Sets or queries which edge of the input signal will be considered an event for Frequency, Period, Frequency Ratio, Time Interval, Totalize, and Phase measurements.

With the POSitive slope selected, a signal going from one voltage level to a more positive level, regardless of polarity, will define the event at the upper hysteresis limit. With the NEGative slope selected,
the negative going edge of the signal will define an event at the lower hysteresis limit.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- The slopes used for Rise/Fall Time, Pulse Width, and Duty Cycle measurements are independent of this setting.

Related Front-Panel Keys Trigger/Sensitivity

[:SENSe]:EVENt3 Subtree

This subtree queries the characteristics of the “trigger event“ for channel 3 input port.

[:SENSe]:EVENt3:LEVel[:ABSolute]?

Queries the trigger level of channel 3 input port.

Query Response

- Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.
- The value returned is 0.

Comments Units are Volts.

[:SENSe]:EVENt3:SLOPe?

Queries which edge of channel 3 input port will be considered an event.

Query Response A sequence of ASCII-encoded bytes: POS

[:SENSe]:FREQuency Subtree

This subtree controls the Frequency, Frequency Ratio, and Period measuring capabilities of the instrument.

[:SENSe]:FREQuency:ARM Subtree

This subtree is used to synchronize the Frequency, Frequency Ratio, and Period start and stop arm with events. The following combination of start/stop arming sources are valid:

STARt:SOURce	**STOP:SOURce**	**Front-Panel Gating Settings**	
		GATE	**STOP**
IMMediate	IMMediate	AUTO	_____
IMMediate	TIMer	TIME	_____
IMMediate	DIGits	DIGITS	_____
EXTernal	EXTernal	EXTERNAL	POS, NEG
EXTernal	TIMer	EXTERNAL	TIME
EXTernal	IMMediate	EXTERNAL	AUTO

[:SENSe]:FREQuency:ARM[:STARt]:SLOPe POSitive | NEGative

Sets or queries the slope of the external start arm signal used in external arming Frequency, Frequency Ratio, and Period measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- Only applies when [:SENS]:FREQ:ARM[:STAR]:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:ARM[:STARt]:SOURce IMMediate |EXTernal

Sets or queries the start arm for Frequency, Frequency Ratio, and Period measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or EXT

Comments *RST: IMMediate

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:ARM:STOP:DIGits <numeric_value>

Sets or queries the resolution in terms of digits used in arming Frequency, Period, and Ratio measurements.

<numeric_value> Range 3 to 15

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 4
- Only applies when [:SENS]:FREQ:ARM:STOP:SOUR DIG is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:ARM:STOP:SLOPe POSitive | NEGative

Sets or queries the slope of the external stop arm signal used in external arming Frequency, Frequency Ratio, and Period measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: NEGative
- Only applies when [:SENS]:FREQ:ARM:STOP:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:ARM:STOP:SOURce IMMediate | EXTernal | TIMer | DIGits

Sets or queries the stop arm for Frequency, Frequency Ratio, and Period measurements.

Query Response A sequence of ASCII-encoded bytes: IMM, EXT, TIM, or DIG

Comments *RST: TIMer

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:ARM:STOP:TIMer <numeric_value> [S]

Sets or queries the gate time used in arming Frequency, Frequency Ratio, and Period measurements.

<numeric_value> Range

- For short gate time: 1E-3 to 99.99E-3 seconds
- For long gate time: 100E-3 to 1000.000 seconds

<numeric_value Resolution

- For short gate time: 0.01E-3 seconds
- For long gate time: 1E-3 seconds

Query Response Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

Comments

- *RST: 100E-3 S
- Only applies when [:SENS]:FREQ:ARM:STOP:SOUR TIM is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:FREQuency:EXPected[1|2|3] <numeric_value> [HZ]

Sets or queries the approximate frequency of a signal you expect to measure. Providing this value enables the Counter to eliminate a pre-measurement step, saving measurement time and enabling more accurate arming. This applies to the following measurement functions: Frequency, Period, Ratio, Phase, and Duty Cycle.

Note that the actual frequency of the input signal must be within **10 %** of the expected frequency value you entered.

<numeric_value> Range

- For channels 1 and 2, the frequency range is 0.1 to 225E6 HZ
- For channel 3 (Option 030), the frequency range is 100E6 to 3E9 HZ
- For channel 3 (Option 050), the frequency range is 200E6 to 5E9 HZ

Query Response

- Numeric data transferred as ASCII bytes in <NR3> format with fifteen significant digits.
- If [:SENS]:FREQ:EXP[1|2|3]:AUTO is ON, Not a Number **9.91E37** is returned and error -221 is generated.

Comments

This value is unaffected by save/recall.

[:SENSe]:FREQuency:EXPected[1|2|3]:AUTO ON

The command, which only allows the ON parameter, configures the Counter to perform, as necessary, a pre-measurement step to automatically determine the approximate frequency of the measurement signal(s). This applies to the following measurement functions: Frequency, Period, Ratio, Phase, and Duty Cycle.

The query indicates whether or not the above described pre-measurement step is enabled.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: ON
- This value is unaffected by save/recall.
- While the Counter is configured to ON, representative CW signal(s) must be present at the measurements input(s).
- The ON setting causes the Counter to disregard any previously set “expected frequency” ([:SENS]:FREQ:EXP[1|2|3]).
- The only mechanism for disabling the above described pre-measurement step is to specify an expected frequency with [:SENS]:FREQ:EXP[1|2|3].

[:SENSe]:FUNCtion[:ON] <sensor_function>

Sets or queries the <sensor_function> to be sensed by the Counter.

The <sensor_function> strings are:

“[:][XNONe:]DCYCle [1]”
or
“[:][XNONe:]PDUTycycle [1]”
“[:][XNONe:]FALL:TIME [1]”
or
“[:][XNONe:]FTIMe [1]”
“[:][XNONe:]FREQuency [1|2|3]”
“[:][XNONe:]FREQuency:RATio [1,2 | 1,3 | 2,1 | 3,1]”
“[:][XNONe:]NWIDth [1]”
“[:][XNONe:]PERiod [1|2|3]”
“[:][XNONe:]PHASe [1,2]”
“[:][XNONe:]PWIDth [1]”
“[:][XNONe:]TINTerval [1,2]”
“[:][XNONe:]TOTalize [1]”
“[:][XNONe:]RISE:TIME [1]”
or
“[:][XNONe:]RTIMe [1]”
“[:][XNONe:]VOLTage:MAXimum [1|2]”
“[:][XNONe:]VOLTage:MINimum [1|2]”
“[:][XNONe:]VOLTage:PTPeak [1|2]”

Query Response

- The string "<function> <channel>[,<channel>]" is returned.

- The string omits default nodes (XNONe) and uses short form mnemonics. If the channel specifier(s) are set to default value(s), no channel specifier is returned in response. If the channel specifier(s) are not set to default value(s), they will be returned in the response with a single space separating the first channel specifier from the function name.

  For example:

  – "FREQ" would be returned for frequency on Channel 1.

  – "FREQ 2" would be returned for frequency on Channel 2.

  – "FREQ:RAT" would be returned for frequency ratio of Channel 1 to Channel 2.

  – "FREQ:RAT 1,3" would be returned for frequency ratio of Channel 1 to Channel 3.

Comments

- *RST: "FREQuency 1"

- If the optional channel specification is omitted from the <sensor_function>, a default channel selection is made. For Frequency, Period, Rise Time, Fall Time, Pulse Width, Duty Cycle, Totalize, and Voltage Peaks, the default is Channel 1. For Time Interval, Frequency Ratio, and Phase, the default is Channel 1 to Channel 2.

- When the sensor function is changed to [:][XNONe:]TINTerval [1,2], the input routing is set to separate ([:SENS]:EVEN2:FEED "INP2"). Use [:SENS]:EVEN2:FEED "INP1" to select common-mode Time Interval.

- When the sensor function is changed to Rise Time, the Channel 1 trigger level is set to 10%, the Channel 2 trigger level is set to 90%, and auto-triggering is enabled.

- When the sensor function is changed to Fall Time, the Channel 1 trigger level is set to 90%, the Channel 2 trigger level is set to 10%, and auto-triggering is enabled.

- When the sensor function is changed to Duty Cycle or Pulse Width, the Channel 1 trigger level is set to 50%, the Channel 2 trigger level is set to 50%, and auto-triggering is enabled.

- When the sensor function is changed resulting in auto-trigger being enabled, the Counter will wait until a measurement is initiated before performing the first "auto-trigger" and updating the absolute level(s).

- This command has no direct effect on :FETCh?, :READ?, or :CONFigure?.

- When the sensor function is Totalize, Voltage Minimum, Voltage Maximum, or Voltage Peak-to-Peak, then [:INIT]:IMM always initiates a single measurement.

Related Front-Panel Keys

Freq & Ratio, Time & Period, and Other Meas

[:SENSe]:PHASe Subtree

This subtree controls the Phase measuring capabilities of the instrument.

[:SENSe]:PHASe:ARM Subtree

This subtree is used to synchronize the Phase start arm with events.

STARt:SOURce	Front-Panel Arming Mode
IMMediate	AUTO
EXTernal	EXTERNAL

[:SENSe]:PHASe:ARM[:STARt]:SLOPe POSitive | NEGative

Sets or queries the slope of the external start arm signal used in external arming Phase measurements.

Query Response

A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive

- Only applies when [:SENS]:PHAS:ARM[:STAR]:SOUR EXT is selected.

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:PHASe:ARM[:STARt]:SOURce IMMediate | EXTernal

Sets or queries the start arm for Phase measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or EXT

Comments *RST: IMMediate

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:ROSCillator Subtree

This subtree controls the Reference Oscillator.

[:SENSe]:ROSCillator:EXTernal:CHECk ON | OFF | ONCE

Sets or queries the enable for "checking" the validity and presence of the external reference.

When CHECk is ON and external has been explicitly selected ([:SENS]:ROSC:SOUR is EXT and [:SENS]:ROSC:SOUR:AUTO is OFF), the Counter checks the external reference signal to ensure that the frequency is 1, 5, or 10 MHz and that the reference is present at measurement completion. Note that the Agilent 53132A only accepts 10 MHz. When CHECk is OFF, the external reference signal is not checked at all.

CHECk ONCE is an event which invokes the external reference check at the time the command is executed. ONCE is only permitted if [:SENS]:ROSC:SOUR is EXT; otherwise, error -221 is generated.
If the CHECk ONCE does not detect a valid timebase, error +2009 is generated. After the check is completed, this command's parameter is set to OFF.

Query Response A sequence of ASCII-encoded bytes: ON or OFF

Comments

- *RST: ON
- Use this command when [:SENS]:ROSC:SOUR EXT has been sent.
- This value is unaffected by save/recall.

[:SENSe]:ROSCillator:EXTernal:FREQuency?

Queries the frequency value of the external reference oscillator.

Query Response

- Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.
- Range is 1E6 to 10E6.
- Units are Hertz.
- If the current reference timebase is external but the frequency is not known (because it is not 1, 5, or 10 MHz and [:SENS]:ROSC:SOUR:AUTO is OFF), Not a Number **9.91E37** is returned.
- If the current reference timebase is internal, Not a Number **9.91E37** is returned.

Comments Query only.

[:SENSe]:ROSCillator:SOURce INTernal | EXTernal

Sets or queries current reference timebase.

INTernal indicates the timebase is the internal reference. EXTernal indicates the signal at the external reference input (located on the rear panel of the Counter; **Ref In** connector) is the reference timebase.

Query Response A sequence of ASCII-encoded bytes: INT or EXT.

Comments

- Execution of the command (that is, explicitly selecting internal or external timebase) sets [:SENS]:ROSC:SOUR:AUTO to OFF.
- The query can be used to determine the current reference timebase when [:SENS]:ROSC:SOUR:AUTO is ON. That is, the query response will indicate which timebase (internal or external) has automatically been selected.
- This value is unaffected by save/recall.

Related Front-Panel Keys

Utility/POWER

[:SENSe]:ROSCillator:SOURce:AUTO <Boolean>

Sets or queries the enable for automatically selecting a reference timebase.

When AUTO is ON, the Counter will automatically select the external reference signal as the reference timebase when a valid signal (1, 5, or 10 MHz) is present at the **Ref In** rear-panel connector. Note that the Agilent 53132A only accepts 10 MHz. The internal timebase is used when an invalid signal is present at this connector.

When AUTO is OFF, the reference timebase is selected with [:SENS]:ROSC:SOUR.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: ON
- Explicitly selecting a reference oscillator (with [:SENS]:ROSC:SOUR INT|EXT) sets AUTO to OFF.
- This value is unaffected by save/recall.

Related Front-Panel Keys

Utility/POWER

[:SENSe]:TINTerval Subtree (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646)

This subtree controls the time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measuring capabilities of the instrument. *Refer to page 4-98, for Time Interval arming capabilities of the Agilent 53132A with a serial number prefix 3646 and above.*

[:SENSe]:TINTerval:ARM Subtree (Agilent 53131A and Agilent 53132A With S/N Prefix Below 3646)

This subtree is use to synchronize the time interval start and stop arm with events. The following combination of start/stop arming sources are valid:

STARt:SOURce	STOP:SOURce	Front-Panel Arming Settings	
		ARM	DELAY
IMMediate	IMMediate	AUTO	NONE
IMMediate	TIMer	AUTO	TIME
EXTernal	IMMediate	EXTERNAL	NONE
EXTernal	TIMer	EXTERNAL	TIME

[:SENSe]:TINTerval:ARM[:STARt]:SLOPe POSitive | NEGative

Sets or queries the slope of the external start arm signal used in external arming time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- Only applies when [:SENS]:TINT:ARM[:STAR]:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM[:STARt]:SOURce IMMediate | EXTernal

Sets or queries the start arm for time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or EXT

Comments *RST: IMMediate

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM:STOP:SOURce IMMediate | TIMer

Sets or queries the stop arm (that is, the delay) for Time Interval measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or TIM

Comments

- *RST: IMMediate
- This [:SENS]:TINT:ARM:STOP:SOUR command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM:STOP:TIMer <numeric_value> [S]

Sets or queries the time used to delay the stop arm for Time Interval measurements.

<numeric_value> Range

- For short gate time: 0.1E-3* to 99.99E-3 seconds
- For long gate time: 100E-3 to 10.000 seconds

<numeric-value> Resolution

- For short gate time: 0.01E-3 seconds
- For long gate time: 1E-3 seconds

*For Agilent 53131A with firmware revisions below 3427, the minimum gate time is 1 ms.
*For Agilent 53132A with firmware revisions below 3646, the minimum gate time is 1 ms.

Query Response

Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

Comments

- *RST: 10E-3 S
- Only applies when [:SENS]:TINT:ARM:STOP:SOUR TIM is selected.
- This [:SENS]:TINT:ARM:STOP:TIM command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:TINTerval Subtree (Agilent 53132A With S/N Prefix 3646 and Above)

This subtree controls the time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measuring capabilities of the instrument. *Refer to page 4-95, for Time Interval arming capabilities of the Agilent 53131A (and the Agilent 53132A with a serial number prefix below 3646).*

[:SENSe]:TINTerval:ARM:ESTART and :ESTOP Subtrees (Agilent 53132A With S/N Prefix 3646 and Above)

These subtrees are used to synchronize the time interval start and stop arm with events.

The following table defines the valid combinations of start/stop arming for Time Interval only. Agilent recommends that SCPI programs issue arming commands in the order shown in the following table ; that is, from left to right, beginning with setting up ESTART and concluding by setting up ESTOP arming.

ESTART:LAY2 :SOURce	ESTART:LAY1 :SOURce	ESTOP:LAY2 :SOURce	ESTOP:LAY1 :SOURce	**Front-Panel Arming Settings**			
				START?	DELAY	STOP?	DELAY?
IMMediate	IMMediate	IMMediate	IMMediate	AUTO	_____	AUTO	NONE
IMMediate	IMMediate	IMMediate	TIMer	AUTO	_____	AUTO	TIME
IMMediate	IMMediate	IMMediate	INTernal2	AUTO	_____	AUTO	EVENT
EXTernal	IMMediate	IMMediate	IMMediate	EXT	NONE	AUTO	NONE
EXTernal	IMMediate	IMMediate	TIMer	EXT	NONE	AUTO	TIME
EXTernal	IMMediate	IMMediate	INTernal2	EXT	NONE	AUTO	EVENT
EXTernal	IMMediate	EXTernal	IMMediate	EXT	NONE	EXT	NONE
EXTernal	IMMediate	EXTernal	TIMer	EXT	NONE	EXT	TIME
EXTernal	IMMediate	EXTernal	INTernal2	EXT	NONE	EXT	EVENT
EXTernal	TIMer	IMMediate	IMMediate	EXT	TIME	AUTO	NONE
EXTernal	TIMer	IMMediate	TIMer	EXT	TIME	AUTO	TIME
EXTernal	TIMer	IMMediate	INTernal2	EXT	TIME	AUTO	EVENT
EXTernal	TIMer	EXTernal	IMMediate	EXT	TIME	EXT	NONE
EXTernal	TIMer	EXTernal	TIMer	EXT	TIME	EXT	TIME
EXTernal	TIMer	EXTernal	INTernal2	EXT	TIME	EXT	EVENT
EXTernal	INTernal1	IMMediate	IMMediate	EXT	EVENT	AUTO	NONE
EXTernal	INTernal1	IMMediate	TIMer	EXT	EVENT	AUTO	TIME
EXTernal	INTernal1	IMMediate	INTernal2	EXT	EVENT	AUTO	EVENT
EXTernal	INTernal1	EXTernal	IMMediate	EXT	EVENT	EXT	NONE
EXTernal	INTernal1	EXTernal	TIMer	EXT	EVENT	EXT	TIME
EXTernal	INTernal1	EXTernal	INTernal2	EXT	EVENT	EXT	EVENT

The following table is for Risetime, Falltime, Duty Cycle, and Pulse Width.

ESTART:LAY2:SOURce	**Front-Panel Arming Setting** ARM
IMMediate	AUTO
EXTernal	EXTERNAL

[:SENSe]:TINTerval:ARM:ESTART:LAYer2:SLOPe POSitive | NEGative

Sets or queries the slope of the external start arm signal used in external arming time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response

A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- Only applies when [:SENS]:TINT:ARM:ESTART:LAY2:SOUR EXT is selected.

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTART:LAYer2:SOURce IMMediate | EXTernal

Sets or queries the start arm for time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response

A sequence of ASCII-encoded bytes: IMM or EXT

Comments

*RST: IMMediate

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:ECOunt <numeric_value>

Sets or queries the number of events used to delay the start arm for Time Interval measurements.

<numeric_value> Range

1 to 99,999,999

<numeric_value> Resolution

1

Query Response

Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 1
- Only applies when [:SENS]:TINT:ARM:ESTART[:LAYer[1]]:SOUR INTernal1 is selected.

- The [:SENS]:TINT:ARM:ESTART[:LAYer[1]]:ECO command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:SOURce IMMediate | TIMer | INTernal[1]

Sets or queries the start arm delay for Time Interval measurements.

Query Response

A sequence of ASCII-encoded bytes: IMM or TIM or INT[1]

Comments

- *RST: IMMediate
- The [:SENS]:TINT:ARM:ESTART[:LAYer[1]]:SOUR command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTART[:LAYer[1]]:TIMer <numeric_value> [S]

Sets or queries the time used to delay the start arm for Time Interval measurements.

<numeric_value> Range

100E-9 to 0.999,999,9 seconds

<numeric_value> Resolution

100E-9 seconds

Query Response

Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 100E-9 seconds
- Only applies when [:SENS]:TINT:ARM:ESTART[:LAYer[1]]:SOUR TIM is selected.
- [:SENS]:TINT:ARM:ESTART[:LAYer[1]]:TIM command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys

Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SLOPe POSitive | NEGative

Sets or queries the slope of the external stop arm signal used in external arming time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- Only applies when [:SENS]:TINT:ARM:ESTOP:LAY2:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTOP:LAYer2:SOURce IMMediate | EXTernal

Sets or queries the stop arm for time interval (including Time Interval, Risetime, Falltime, Duty Cycle, and Pulse Width functions) measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or EXT

Comments *RST: IMMediate

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:ECOunt <numeric_value>

Sets or queries the number of events used to delay the stop arm for Time Interval measurements.

<numeric_value> Range 1 to 99,999,999

<numeric_value> Resolution 1

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 1
- Only applies when [:SENS]:TINT:ARM:ESTOP[:LAYer[1]]:SOUR INTernal2 is selected.

[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:SOURce IMMediate | TIMer | INTernal2

Sets or queries the stop arm for Time Interval measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or TIM or INT2

Comments

- *RST: IMMediate
- The [:SENS]:TINT:ARM:ESTOP[:LAYer[1]]:SOUR command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TINTerval:ARM:ESTOP[:LAYer[1]]:TIMer <numeric_value> [S]

Sets or queries the time used to delay the stop arm for Time Interval measurements.

<numeric_value> Range For short delay: 100E-9 to 0.999,999,9 seconds

For long delay: 1.000 to 10.000 seconds

<numeric_value> Resolution For short delay: 100E-9 seconds

For long delay: 1E-3 seconds

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- *RST: 100E-9 S
- Only applies when [:SENS]:TINT:ARM:ESTOP[:LAYer[1]]:SOUR TIM is selected.
- The [:SENS]:TINT:ARM:ESTOP[:LAYer[1]]:TIM command has no affect on the following measurements: Risetime, Falltime, Duty Cycle, and Pulse Width.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TOTalize Subtree

This subtree controls the Totalize measuring capabilities of the instrument.

[:SENSe]:TOTalize:ARM Subtree

This subtree is used to synchronize the Totalize start and stop arm with events. The following combination of start/stop arming sources are valid:

STARt:SOURce	**STOP:SOURce**	**Front-Panel Gating Settings**	
		GATE	**STOP**
IMMediate	IMMediate	AUTO	_____
IMMediate	TIMer	TIME	_____
EXTernal	EXTernal	EXTERNAL	POS, NEG
EXTernal	TIMer	EXTERNAL	TIME

[:SENSe]:TOTalize:ARM[:STARt]:SLOPe POSitive | NEGative

Sets or queries the slope of the external start arm signal used in external arming Totalize measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: POSitive
- Only applies when [:SENS]:TOT:ARM[:STAR]:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TOTalize:ARM[:STARt]:SOURce IMMediate | EXTernal

Sets or queries the start arm for Totalize measurements.

Query Response A sequence of ASCII-encoded bytes: IMM or EXT

Comments

- *RST: IMMediate
- When both start and stop arm are set to IMMediate, use :INIT[:IMM] to begin totalizing and :ABORt to terminate. No valid totalize result will exist until the "auto" totalize measurement is terminated.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TOTalize:ARM:STOP:SLOPe POSitive | NEGative

Sets or queries the slope of the external stop arm signal used in external arming Totalize measurements.

Query Response A sequence of ASCII-encoded bytes: POS or NEG

Comments

- *RST: NEGative
- Only applies when [:SENS]:TOT:ARM:STOP:SOUR EXT is selected.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TOTalize:ARM:STOP:SOURce IMMediate | EXTernal TIMer

Sets or queries the stop arm for Totalize measurements.

Query Response A sequence of ASCII-encoded bytes: IMM, EXT or TIM

Comments

- *RST: TIMer
- When both start and stop arm are set to IMMediate, use :INIT[:IMM] to begin totalizing and :ABORt to terminate. No valid totalize result will exist until the "auto" totalize measurement is terminated.

Related Front-Panel Keys Gate & ExtArm

[:SENSe]:TOTalize:ARM:STOP:TIMer <numeric_value> [S]

Sets or queries the gate time used in arming Totalize measurements.

<numeric_value> Range

- For short gate time: 1E-3 to 99.99E-3 seconds
- For long gate time: 100E-3 to 1000.000 seconds

<numeric_value Resolution

- For short gate time: 0.01E-3 seconds
- For long gate time: 1E-3 seconds

Query Response Numeric data transferred as ASCII bytes in <NR3> format with six significant digits.

Comments

- *RST: 100E-3 S
- Only applies when [:SENS]:TOT:ARM:STOP:SOUR TIM is selected.

Related Front-Panel Keys Gate & ExtArm

:STATus Subsystem

The :STATus subsystem commands allow you to specify or examine the status of the Operation Status Register group and the Questionable Data/Signal Register group.

:STATus:OPERation Subtree

The :STATus:OPERation subtree commands allow you to examine the status of the Counter monitored by the Operation Status Register group, shown in Figure 4-2. The Operation Status Register group consists of a condition register, two transition registers, an event register, and an enable register. The commands in this subtree allow you to control and monitor these registers.

See the section titled “Operation Status Register Group and Questionable Data/Signal Status Register Group” on page 3-32 in Chapter 3 for a detailed description of the Operation Status Register Group.

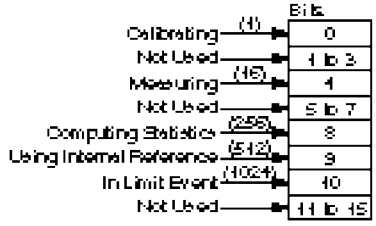


Figure 4-2. The Operation Status Register Group

:STATus:OPERation:CONDition?

Queries the status of the Operation Condition Status Register.

Bits are not cleared when read.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 65,535.
- The query response value is an integer formed by the binary-weighting of the bits. The value of unused bits is zero.

Comments

The Operation Condition Status Register is cleared at power-on.

:STATus:OPERation:ENABle <non-decimal numeric> | <NRf>

Sets or queries the Operation Event Status Enable Register.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Operation Event Status Enable Register. The value of unused bits is zero when queried and ignored when set.

This register is used to enable a single or inclusive OR group of Operation Event Status Register events to be summarized in the Status Byte Register (bit 7).

Range The range for the <non-decimal numeric> or <NRf> parameter is 0 to 65,535.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and :STAT:PRES, the Operation Event Status Enable Register is cleared (value is 0).
- This value is unaffected by *RST and save/recall.

:STATus:OPERation[:EVENt]?

Queries the status of the Operation Event Status Register.

The Operation Event Status Register captures changes in conditions by having each event bit correspond to a specific condition bit in the Operation Condition Status Register. An event becomes TRUE when the associated condition makes the transition specified by the transition filters. The event bits, once set, are "sticky." That is, they cannot be cleared, even if they do not reflect the current status of a related condition, until they are read.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 65,535.
- The query response value is an integer formed by the binary-weighting of bits. The value of unused bits is zero.

Comment The Operation Event Status Register is cleared by *CLS, by :STAT:OPER[:EVEN]?, and at power-on.

:STATus:OPERation:NTRansition <non-decimal numeric> | <NRf>

Sets or queries the negative transition filter for the Operation status reporting structure.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the negative transition filter. The value of unused bits is zero when queried and ignored when set.

A TRUE bit (in the negative transition filter) specifies that a negative (TRUE to FALSE) transition of the corresponding bit in the Operation Condition Status Register generates the corresponding event in the Operation Event Status Register.

Range The range of the <non-decimal numeric> or <NRf> parameter is 0 to 65,535.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and STAT:PRES, the negative transition filter is preset such that each bit is a 0 (FALSE).
- This value is unaffected by *RST and save/recall.

:STATus:OPERation:PTRansition <non-decimal numeric> | <NRf>

Sets or queries the positive transition filter for the Operation status reporting structure.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the positive transition filter. The value of unused bits is zero when queried and ignored when set.

A TRUE bit (in the positive transition filter) specifies that a positive (FALSE to TRUE) transition of the corresponding bit in the Operation Condition Status Register generates the corresponding event in the Operation Event Status Register.

Range The range of the <non-decimal numeric> or <NRf> parameters is 0 to 65,535.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and STAT:PRES, the positive transition filter is preset such that each bit is a 1 (TRUE).
- This value is unaffected by *RST and save/recall.

:STATus:PRESet

This event command presets the enable registers and transition filters associated with the Operation and Questionable status reporting structures. The enable registers and negative transition filters are preset such that each bit is a 0 (FALSE). The positive transition filters are preset such that each bit is a 1 (TRUE).

:STATus:QUEStionable Subtree

The :STATus:QUEStionable subtree commands allow you to examine the status of the Counter monitored by the Questionable Data/Signal Status Register group, shown in Figure 4-3. The Questionable Status group consists of a condition register, two transition registers, an event register, and an enable register. The commands in this subtree allow you to control and monitor these registers.

See the section titled “Operation Status Register Group and Questionable Data/Signal Status Register Group” on page 3-32 in Chapter 3 for a detailed description of the Questionable Data/Signal Status Register Group.

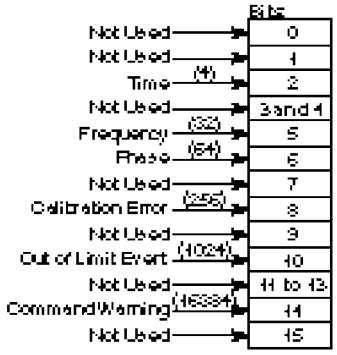


Figure 4-3. The Questionable Data/Signal Status Register Group

:STATus:QUEStionable:CONDition?

Queries the status of the Questionable Data Condition Status Register.

Bits are not cleared when read.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 65,535.
- The query response value is an integer formed by the binary-weighting of the bits. The value of unused bits is zero.

Comments The Questionable Data Condition Status Register is cleared at power-on.

:STATus:QUEStionable:ENABle <non-decimal numeric> | <NRf>

Sets or queries the Questionable Data Event Status Enable Register.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Questionable Data Event Status Enable Register. The value of unused bits is zero when queried and ignored when set.

This register is used to enable a single or inclusive OR group of Questionable Data Event Status Register events to be summarized in the Status Byte Register (bit 3).

Range The range of the <non-decimal numeric> or <NRf> parameter is 0 to 65,535.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and :STAT:PRES, the Questionable Data Event Status Enable Register is cleared (value is 0).
- This value is unaffected by *RST and save/recall.

:STATus:QUEStionable[:EVENt]?

Queries the status of the Questionable Data Event Status Register.

The Questionable Data Event Status Register captures changes in conditions by having each event bit correspond to a specific condition bit in the Questionable Data Condition Status Register. An event becomes TRUE when the associated condition makes the transition specified by the transition filters. The event bits, once set, are "sticky." That is, they cannot be cleared, even if they do not reflect the current status of a related condition, until they are read.

The Questionable Data Event Status Register is cleared by *CLS, by :STAT:QUES[:EVEN]?, and at power-on.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 65,535.
- The query response value is an integer formed by the binary-weighting of bits. The value of unused bits is zero.

:STATus:QUEStionable:NTRansition <non-decimal numeric> | <NRf>

Sets or the negative transition filter for the Questionable Data status reporting structure.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the negative transition filter. The value of unused bits is zero when queried and ignored when set.

A TRUE bit (in the negative transition filter) specifies that a negative (TRUE to FALSE) transition of the corresponding bit in the Questionable Data Condition Status Register generates the corresponding event in the Questionable Data Event Status Register.

Range

The range of the <non-decimal numeric> or <NRf> parameter is 0 to 65,535.

Query Response

Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and :STAT:PRES, the negative transition filter is preset such that each bit is a 0 (FALSE).
- This value is unaffected by *RST and save/recall.

:STATus:QUEStionable:PTRansition <non-decimal numeric> | <NRf>

Sets or queries the positive transition filter for the Questionable Data status reporting structure.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the positive transition filter. The value of unused bits is zero when queried and ignored when set.

A TRUE bit (in the positive transition filter) specifies that a positive (FALSE to TRUE) transition of the corresponding bit in the Questionable Data Condition Status Register generates the corresponding event in the Questionable Data Event Status Register.

Range The range of the <non-decimal numeric> or <NRf> parameter is 0 to 65,535.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

- At power-on and :STAT:PRES, the positive transition filter is preset such that each bit is a 1 (TRUE).
- This value is unaffected by *RST and save/recall.

:SYSTem Subsystem

This subsystem collects together the capabilities that are not related to instrument performance.

:SYSTem:COMMunicate Subtree

The :SYSTem:COMMunicate subtree collects together the configuration of the control/communication interfaces.

The :SYSTem:COMMunicate:SERial subtree controls the physical configuration of the RS-232C port. Any command to change the settings takes effect immediately upon receipt of the "program message termination." These settings are stored in non-volatile memory, and are unaffected by power-on, save/recall, and *RST.

The :SYSTem:COMMunicate:SERial:TRANsmit subtree controls parameters associated with transmission.

The Counter will always use one start bit and one stop bit.

:SYSTem:COMMunicate:SERial:CONTrol:DTR IBFull | ON | LIMit

Sets or queries the hardware pacing scheme.

The ON parameter (which is equivalent to ***DTR: HIGH*** in the front-panel Utility menu) indicates that the DTR (Data Terminal Ready) line, which is pin 4 of the RS-232 connector, is always asserted (HIGH) to always be ready to allow data to be sent to the printer. Choose the ON parameter when the printer or cable you are using does not support handshaking.

The IBFull parameter (which is equivalent to ***DTR: HW PACE*** in the front-panel Utility menu) sets the RS-232 DTR line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the Counter reaches a stop thresholdrhhhe the Counter will de-assert the DTR line. When the number of bytes has been reduced to a start threshold, the Counter will assert DTR, indicating that it can receive input again. The Counter will also monitor the state of the DSR (Data Set Ready) line, which is pin 6 of the RS-232 connector, and will stop transmission if either of those lines becomes de-asserted. Choose the IBFull parameter when the printer and cable you are using require handshaking for counter-to-printer communication.

The LIMit parameter (which is equivalent to ***DTR: LIMIT*** in the front-panel Utility menu) indicates that the RS-232 DTR line will be used to indicate out of limit. The LIMit parameter will force the DTR line HIGH if the measurement is in limit, and LOW if the measurement is out of limit.

Query Response A sequence of ASCII-encoded bytes: IBF, ON, or LIM

Comments
- This value is stored in non-volatile memory. It is unaffected by power-on, save/recall, and *RST.
- The start and stop thresholds are not user configurable.

Related Front-Panel Keys Utility/POWER

:SYSTem:COMMunicate:SERial:TRANsmit:BAUD <numeric_value>

Sets or queries the baud rate.

<numeric_value> Range The possible BAUD rate values that can be entered for the <numeric_value> parameter are: 300, 1200, 2400, 9600, 19200.

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments This value is stored in non-volatile memory. It is unaffected by power-on, save/recall, and *RST.

Related Front-Panel Keys Utility/POWER

:SYSTem:COMMunicate:SERial:TRANsmit:PACE XON | NONE

Sets or queries the software pacing scheme.

Query Response A sequence of ASCII-encoded bytes: XON or NONE

Comments This value is stored in non-volatile memory. It is unaffected by power-on, save/recall, and *RST.

Related Front-Panel Keys Utility/POWER

:SYSTem:COMMunicate:SERial:TRANsmit:PARity[:TYPE] EVEN | ODD | NONE

Sets or queries the parity scheme.

Query Response A sequence of ASCII-encoded bytes: EVEN, ODD, or NONE

Comments

- This value is stored in non-volatile memory. It is unaffected by power-on, save/recall, and *RST.
- If parity is enabled, the Counter sends/receives 7 data bits plus 1 parity bit. If parity is disabled, the Counter sends/receives 8 data bits.

Related Front-Panel Keys Utility/POWER

:SYSTem:ERRor?

Queries the oldest error in the Error Queue and removes that error from the queue (first in, first out).

See page 5-2 in Chapter 5, “Errors,” for detailed error information

Query Response

- The response is in the following form: <error_number>, “<error_description>”.
- The <error_number> is an integer in the range [-32768, 32767]. The negative error numbers are defined by the SCPI standard; positive error numbers are particular to this Counter. An error number value of zero indicates that the Error Queue is empty.
- The maximum length of the <error_description> is 255 characters.

Comments

- The queue is cleared (emptied) on *CLS, power-on, or upon reading the last error from queue.
- If the Error Queue overflows, the last error in the queue is replaced with the error -350, “Queue overflow”. Any time the queue overflows, the least recent errors remain in the queue and the most recent error is discarded. The maximum length of the Error Queue is 30.
- This query clears any displayed error message from the front-panel display.
- The Error Queue is unaffected by *RST and save/recall.

:SYSTem:KEY <numeric_value>

This command simulates the pressing or a front-panel key.
The <numeric_value> is a key code value.

This command puts an entry in the Key Queue (just as any front-panel key press does). The length of the Key Queue is 500.

The keys and their corresponding key codes are listed in the following table.

Key	Key Code	Key	Key Code
Freq & Ratio	1	d	25
Time & Period	4	f	28
Other Meas	2	s	26
Gate & ExtArm	5	g	27
Uppr & Lower	7	Channel 1 Trigger/Sensitivity	13
Limit Modes	8	Channel 1 50W/1MW	14
Scale & Offset	10	Channel 1 DC/AC	15
Stats	11	Channel 1 X10 Attenuate	16
Recall(Utility)	3	Channel 1 100kHz Filter	17
Save & Print	6	Channel 2 Trigger/Sensitivity	19
Run	9	Channel 2 50W/1MW	20
Stop/Single	12	Channel 2 DC/AC	21
+/-	29	Channel 2 X10 Attenuate	22
Enter	30	Channel 2 100kHz Filter	23

<numeric_value> Range

1 to 30, except 18 and 24

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.

- The query returns the key code for the last key pressed. A value of -1 indicates the queue (last in, first out) is empty.

Comments

- At*RST and power-on, the Key Queue is cleared (emptied).
- The Key Queue is unaffected by save/recall.
- Key commands are sequential, but only in terms of processing other key commands or getting into the Key Queue. The operation performed by the key command is not guaranteed to be complete before processing of the next non-key command. Be aware of this when intermixing key commands and non-key commands.

:SYSTem:KEY:LOG?

This query returns a comma separated list of integers representing all of the entries in the Key Queue.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- A value of -1 indicates the queue is empty.
- Numbers (representing key codes) are separated by commas.
The key codes appear in a last in, first out sequence.
The maximum number of key codes is 500. Each key code has a range of 1 to 30.

Comments

- At *RST and power-on, the Key Queue is cleared (emptied).
- The Key Queue is unaffected by save/recall.

:SYSTem:VERSion?

Queries the SCPI version number with which the Counter complies.

Query Response

- Numeric data transferred as ASCII bytes in <NR2> format.
- The response is an <NR2> formatted numeric value which has the form YYYY.V, where YYYY represents the year (1992) and the V represents the approved version for that year (0).

Comments

The instrument complies with SCPI Standard 1992.0 and returns this value as the response to this query.

:TRACe Subsystem

This subsystem provides access to the scale and offset values.
The :TRACe subsystem used in conjunction with the :CALCulate[1] subsystem, scales and offsets measurement results.

:TRACe:CATalog?

Queries list of intrinsic constants. The Counter has two constants, scale and offset.

Query Response A comma-separated list of strings: “SCALE”, “OFFSET”

:TRACe[:DATA] OFFSET, <numeric_value> [HZ | S | DEG]

or

:TRACe[:DATA] OFFSET, <arbitrary block>

:TRACe[:DATA]? OFFSET

Sets or queries the offset value.

<numeric_value> Range -9.9999990000E+12 to -1.0000000000E-13, 0.0000000000, +1.0000000000E-13 to +9.9999990000E+12.

<numeric_value> Resolution 11 digits

Query Response

- Response will be formatted according to :FORMat[:DATA] ASCii | REAL setting.
- When ASCii format is used, the numeric data is transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 0.0000000000
- Updating the offset causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel **Enter** key, while this value is in the 11-digit display, will update the actual value to the displayed (rounded) value.

Related Front-Panel Keys Scale & Offset

:TRACe[:DATA] SCALE, <numeric_value>
or
:TRACe[:DATA] SCALE, <arbitrary block>
:TRACe[:DATA]? SCALE

Sets or queries the scale value.

<numeric_value> Range -9.999999E+12 to -1.000000E-13, 0.000000, +1.000000E-13 to +9.999999E+12.

<numeric_value> Resolution 7 digits

Query Response

- Response will be formatted according to :FORMat[:DATA] ASCii | REAL setting.
- When ASCii format is used, the numeric data is transferred as ASCII bytes in <NR3> format with eleven significant digits.

Comments

- *RST: 1.000000
- Updating the scale causes the limit counts (:CALC2:LIM:FCO, :CALC2:LIM:PCO) to be cleared.
- The front panel menu item is not always able to display all of the significant digits of this value. When this is the case, the displayed value is different from the actual value in that the displayed value has been rounded. However, using the front panel **Enter** key, while this value is displayed, will update the actual value to the displayed (rounded) value.

Related Front-Panel Keys Scale & Offset

:TRIGger Subsystem

This subsystem enables synchronization of instrument actions with specified internal or external events.

:TRIGger:COUNt:AUTO <Boolean>

Sets or queries the control over the number of measurements made when :INITiate[:IMMediate] is performed.

When :TRIG:COUN:AUTO is OFF, then :INIT[:IMM] initiates a single measurement.

When :TRIG:COUN:AUTO is ON and [:SENS]:FUNC[:ON] is not Totalize or Voltage Peak function, then:

- when statistics are enabled (:CALC3:AVER[:STAT] is ON), :INIT[:IMM] will initiate a complete block of N (:CALC3:AVER:COUN) valid measurements.

- when statistics are enabled (:CALC3:AVER[:STAT] is ON), and limit-filtering is enabled (:CALC3:LFIL:STAT is ON), :INIT[:IMM] will initiate a complete block of N (:CALC3:AVER:COUN) valid, in-limit measurements.

- when statistics are disabled (:CALC3:AVER[:STAT] is OFF), :INIT[:IMM] initiates a single measurement.

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of 0 indicates OFF; a value of 1 indicates ON.

Comments

- *RST: OFF
- The :TRIG:COUN:AUTO setting has no affect on the operation of :INIT:CONT ON, which always behaves as if :TRIG:COUN is set to 1.

Related Front-Panel Keys

Stats

*CAL?
(Calibration Query)

*CAL?

This query causes an internal interpolator self-calibration.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- A value of zero indicates the calibration completed without error. A value of one indicates the calibration completed with error.

*CLS
(Clear Status Command)

*CLS

Clears all event registers summarized in the status byte (Standard Event Status Register, Operation Event Status Register, and Questionable Data Event Status Register) and clears the Error Queue. The *CLS command will not clear data memories or any other settings.

It also places the instrument in “Operation Complete Idle State” and “Operation Complete Query Idle State” (IEEE 488.2). This results in the disabling of any prior *OPC command.

If *CLS immediately follows a program message terminator, the output queue and the MAV bit are cleared because “any” new program message after a program message terminator clears the output queue.

This command will clear any displayed error message from the front panel.

*DDT <arbitrary block>
(Define Device Trigger Command)

*DDT <arbitrary block>

Sets or queries the command that the device will execute when it receives the IEEE 488.1 Group Execute Trigger (GET) interface message (page 4-42) or a *TRG common command.

There are only three valid commands that the Counter will accept: :INITiate[:IMMediate], :READ?, or :FETCh?; otherwise, error -224 is generated. If a zero-length <arbitrary block> is specified as the parameter, the Counter will do nothing when it receives a GET or *TRG command.

Query Response

- Definite length block
- The query response will be one of the following:

  #14INIT
  #15FETC?
  #15READ?
  #0

  terminated with a new line and EOI.

Comments

- *RST: #14INIT
- When defining the device trigger to :FETCh? or :READ?, note that these definitions do not allow the specification of a particular function. This lack of a function specification results in each *DDT using the function specified/used by the last :CONFigure, :FETCh, :READ, or :MEASure command, if possible.

*DMC <string>, <arbitrary block> (Define Macro Command)

*DMC <string>, <arbitrary block>

This command assigns a sequence of zero or more commands/queries to a macro label. The sequence is executed when the label is received as a command or query.

The <string> parameter specifies the macro label. The macro label may not be a common command/query header. It may be the same as an instrument-specific command/query header; in this case, provided macros are enabled, the macro expansion is executed and the instrument-specific command/query may be executed by disabling macros.

The <arbitrary block> contains the sequence of commands/queries being labeled.

Parameters may be passed to the sequence during execution. Placeholders for parameters appear in the sequence as a dollar sign followed by a single digit in the range one to nine inclusive. The first parameter following the macro label is substituted for the parameter placeholder labeled $1, and so on up to nine parameters.

See the section titled “How to Program the Counter to Define Macros” in Chapter 3 of this guide.

Comments

- The maximum macro label length is 12 characters.
- Redefining an existing macro causes an execution error.
- The Counter allows up to four levels of recursion.
- There is no query form. Use *GMC? (see page 4-129) to query the current definition of a macro label.

*EMC <NRf>
(Enable Macro Command)
*EMC?
(Enable Macro Query)

*EMC <NRf>

*EMC?

Sets or queries the Enable for defined macros.

Macro definitions are not affected by this command. One use of this command is to turn off macro expansion in order to execute an instrument-specific command with the same name as a macro.

The value of the numeric parameter determines whether the defined macros are enabled or disabled. A value that rounds to an integer value of zero disables any defined macros. A value that rounds to an integer value not equal to zero enables any defined macros.

<NRf> Range -32767 to +32767

<NRf> Resolution 1

Query Response

- Single ASCII-encoded byte, 0 or 1.
- A value of zero indicates that macros are disabled and a value of one indicates that macros are enabled.

Comments

- *RST: 0 (disabled)
- This value is unaffected by save/recall.
- Macros are disabled at power-on.

*ESE <NRf>
(Standard Event Status Enable Command)
*ESE?
(Standard Event Status Enable Query)

*ESE <NRf>

*ESE?

Sets or queries the Standard Event Status Enable Register, shown in Figure 4-4.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Standard Event Status Enable Register. The value of unused bits is zero when queried and ignored when set.

This register is used to enable a single or inclusive OR group of Standard Event Status Register events to be summarized in the Status Byte Register (bit 5).

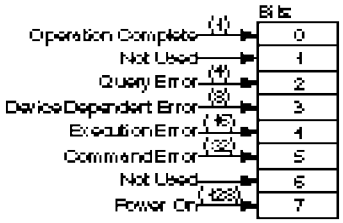


Figure 4-4. The Standard Event Status Enable Register

See the section titled “Standard Event Status Register Group,” page 3-24, in Chapter 3 of this guide for a detailed description of the Standard Event Status Register.

<NRf> Range 0 to 255

<NRf> Resolution 1

Query Response Numeric data transferred as ASCII bytes in <NR1> format.

Comments

1. At power-on, the Standard Event Status Enable Register is cleared (value is 0).

- This value is unaffected by *RST and save/recall.

*ESR?
(Event Status Register Query)

*ESR?

Queries the Standard Event Status Register, shown in Figure 4-5.

This event register captures changes in conditions, by having each event bit correspond to a specific condition in the instrument.
An event becomes TRUE when the associated condition makes the defined transition. The event bits, once set, are “sticky.” That is,
they cannot be cleared even if they do not reflect the current status of a related condition, until they are read.

This register is cleared by *CLS, by *ESR?, and at power-on.
Note that the instrument’s power-on sequence initially clears the register, but then records any subsequent events during the power-on sequence including setting the PON (power on) bit.

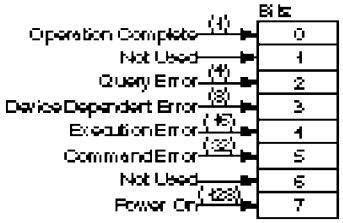


Figure 4-5. Standard Event Status Register

See the section titled “Standard Event Status Register Group,” page 3-24, in Chapter 3 of this guide for a detailed description of the Standard Event Status Register.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 255.
- The query response is an integer formed by the binary-weighting of the bits. The value of unused bit is zero.

*GMC? <string>
(Get Macro Contents Query)

*GMC? <string>

Queries the current definition of a macro.

The <string> parameter must be a currently defined macro label.

Query Response

- Definite length block.
- The query response is a <definite length block> containing the command/query sequence which is executed when the macro label is received.
- A zero-length block response indicates that no command sequence is stored by the specified label.

*IDN?
(Identification Query)

*IDN?

Queries the Counter identification.

Query Response A sequence of ASCII-encoded bytes:

HEWLETT-PACKARD, 53131A,0,XXXX
or
HEWLETT-PACKARD, 53132A,0,XXXX

terminated with a new line and EOI.

XXXX represents the firmware date code.

Comments This query should be the last query in a terminated program message; otherwise, error -440 is generated.

*LMC?
(Learn Macro Query)

*LMC?

Queries the currently defined macro labels.

Query Response

- A sequence of one or more strings separated by commas.
- If no macros are defined, the response is a null string (two consecutive double quote marks).

*OPC
(Operation Complete Command)

*OPC

This event command enables the OPC bit (bit 0) in the Standard Event Status Register to be set upon the transition of the measurement cycle from measuring to idle. (**Note —** For Agilent 53131A/132A with serial number prefix 3646 and above — If the measurement cycle is already idle, this command will immediately set the OPC bit.) See the section titled “Standard Event Status Register Group,” page 3-24, in Chapter 3 of this guide for a detailed description of the Standard Event Status Register’s Operation Complete bit.

This event command is “disabled” by *CLS, *RST, Device Clear (page 4-31), power-on, or upon the transition of the measurement cycle from measuring to idle.

This event command has no query form.

See the section titled “Using the *OPC Command to Assert SRQ,” page 3-48, in Chapter 3 for an example using this command.

*OPC?
(Operation Complete Query)

*OPC?

This query produces a response upon the transition of the measurement cycle from measuring to idle. This allows synchronization between a controller and the instrument using the MAV bit in the Status Byte Register or a read of the Output Queue. (Note that this query does not actually “read” a state, as most queries do.)

Since this query will not respond until the measurement cycle transitions from measuring to idle, the only way to cancel the query “holdoff” is by Device Clear (page 4-31) or power-on.

See the section titled “Using the *OPC? Command,” page 3-47, in Chapter 3 for an example using this command.

Query Response Single ASCII-encoded byte, 1.

NOTE The *OPC? query does not in any way affect the OPC bit in the Standard Event Status Register.

*OPT?
(Option Identification Query)

*OPT?

Queries the instrument to identify any installed options.

The following options can be installed in the instrument:

- Option 001, Medium Stability Oven Timebase
- Option 010, High Stability Oven Timebase
- Option 012, Ultra High Stability Oven Timebase
- Option 030, 3.0 GHz RF Input Channel (Channel 3)
- Option 050, 5.0 GHz RF Input Channel (Channel 3)

Query Response

- A sequence of ASCII-encoded bytes, indicating <timebase option>, <3.0 GHz RF Input option> terminated with a new line and EOI.

  – The <timebase option> is 001, 010, or 0.

  Note that the counter responds with 010 for either the high stability or ultra high stability oven timebase.

  – The <3.0 GHz RF Input option> is 030 or 0.

  Note that the counter responds with 030 for either the 3.0 GHz or 5.0 GHz RF Input option.

  – A missing option is identified by an ASCII 0 (zero).

For example, if only the medium stability timebase option is detected, the response would be: 001,0.

Comments

This query should be the last query in a terminated program message; otherwise, error -440 is generated.

*PMC
(Purge Macro Command)

*PMC

The Purge MaCros command deletes all macros previously defined using the *DMC command.

*RCL <NRf>
(Recall Command)

*RCL <NRf>

This command restores the state of the instrument from a copy stored in local non-volatile memory. Before the recall occurs, the current state of the instrument is automatically saved to register 0.

<NRf> Range 0 to 20

<NRf> Resolution 1

Comments The following commands/states are *unaffected* by *RCL:

*EMC
*ESE
*OPC
*OPC?
*SRE
*WAI
:CALibration:COUNt?
:CALibration:DATA
:CALibration:SECurity:CODE
:CALibration:SECurity:STATe
:CONFigure?
:DIAGnostic:CALibration:INTerpolator:AUTO
:DISPlay:ENABle
:DISPlay:MENU[:STATe]
:DISPLay[:WINDow]:TEXT:RADix
[:SENSe]:FREQuency:EXPected[1|2|3]
[:SENSe]:FREQuency:EXPected[1|2|3]:AUTO
[:SENSe]:ROSCillator:EXTernal:CHECk
[:SENSe]:ROSCillator:SOURCe
[:SENSe]:ROSCillator:SOURCe:AUTO
:STATus:OPERation:ENABle
:STATus:OPERation:NTRansition
:STATus:OPERation:PTRansition
:STATus:QUEStionable:ENABle
:STATus:QUEStionable:NTRansition
:STATus:QUEStionable:PTRansition
:SYSTem:COMMunicate:SERial:CONTrol:DTR
:SYSTem:COMMunicate:SERial:TRANsmit:BAUD
:SYSTem:COMMunicate:SERial:TRANsmit:PACE
:SYSTem:COMMunicate:SERial:TRANsmit:PARity[:TYPE]
:SYSTem:ERRor? (error queue)
:SYSTem:KEY? (key queue)
:SYSTem:KEY:LOG? (key queue)
GPIB Address

*RST
(Reset Command)

*RST

This event command performs an instrument reset.

The reset performs the following:

- sets instrument settings to their *RST states,
- disables macros,
- places instrument in “Operation Complete Idle State” and “Operation Complete Query Idle State,” and
- clears (empties) the Key Queue

The reset does not affect:

- the macros defined with *DMC,
- the calibration data,
- the Service Register Enable or the Standard Event Status Enable,
- the Output Queue, and
- the IEEE 488.1 address or the state of the IEEE 488.1 interface.

See the section titled “*RST Response,” page 2-36, in Chapter 2 of this guide for a complete listing of the *RST state.

Each command description in this chapter (Chapter 4) includes the *RST state in the “Comment” portion of the definition.

*SAV <NRf>
(Save Command)

*SAV <NRf>

This command stores the current state of the instrument in local non-volatile memory.

The current instrument state is saved in register 0 when *RCL or front-panel recall is executed.

<NRf> Range 1 to 20

<NRf> Resolution 1

Comments The following states are *not* saved:

*EMC
*ESE
*OPC
*SRE
:CALibration:COUNt?
:CALibration:DATA
:CALibration:SECurity:CODE
:CALibration:SECurity:STATe
:CONFigure? response
:DIAGnostic:CALibration:INTerpolator:AUTO
:DISPlay:ENABle
:DISPlay:MENU[:STATe]
:DISPLay[:WINDow]:TEXT:RADix
:FETCh? implied function
:READ? implied function
[:SENSe]:FREQuency:EXPected[1|2|3]
[:SENSe]:FREQuency:EXPected[1|2|3]:AUTO
[:SENSe]:ROSCillator:EXTernal:CHECk
[:SENSe]:ROSCillator:SOURCe
[:SENSe]:ROSCillator:SOURCe:AUTO
:STATus:OPERation:ENABle
:STATus:OPERation:NTRansition
:STATus:OPERation:PTRansition
:STATus:QUEStionable:ENABle
:STATus:QUEStionable:NTRansition
:STATus:QUEStionable:PTRansition
:SYSTem:COMMunicate:SERial:CONTrol:DTR
:SYSTem:COMMunicate:SERial:TRANsmit:BAUD
:SYSTem:COMMunicate:SERial:TRANsmit:PACE
:SYSTem:COMMunicate:SERial:TRANsmit:PARity[:TYPE]
GPIB Address
Error Queue
Key Queue

*SRE <NRf>
(Service Request Enable Command)
*SRE?
(Service Request Enable Query)

*SRE <NRf>

*SRE?

Sets or queries the Service Request Enable Register, shown in Figure 4-6.

The parameter and query response value, when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Service Request Enable Register.

This register is used to enable a single or inclusive OR group of Status Byte Register events to generate an SRQ.

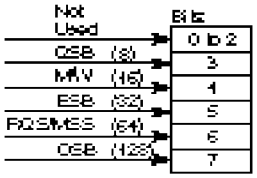


Figure 4-6. The Service Request Enable Register

See the section titled “Status Byte Register and Service Request Enable Register,” page 3-21, in Chapter 3 of this guide for a detailed description of the Service Request Enable Register.

<NRf> Range
- to 255
- The value of bit 6 and unused bits is ignored when set.

<NRf> Resolution
1

Query Response
- Numeric data transferred as ACSII bytes in <NR1> format.
- The value of bit 6 and unused bits is zero when queried.

Comments
- At power-on, this value is cleared (set to 0).
- This value is unaffected by *RST and save/recall.

*STB?
(Status Byte Query)

*STB?

Queries the Status Byte Register, shown in Figure 4-7.

This register is cleared at power-on.

This query does not directly alter the Status Byte Register (including the MSS/RQS bit) or anything related to the generation of SRQ.

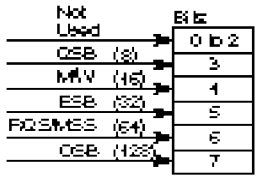


Figure 4-7. The Status Byte Register

See the section titled “Status Byte Register and Service Request Enable Register,” page 3-21, in Chapter 3 of this guide for a detailed description of the Status Byte Register.

Query Response

- Numeric data transferred as ASCII bytes in <NR1> format.
- Range is 0 to 255.
- The response value when rounded to an integer value and expressed in base 2 (binary), represents the bit values of the Status Byte Register.
- The value of unused bits is zero when queried.
- The Master Summary Status, not the RQS message, is reported on bit 6. Master Summary Status indicates that the Counter has at least one reason for requesting service. (The Master Summary Status is not sent in response to a serial poll; the IEEE 488.1 RQS message is sent instead.) It is the inclusive OR of the bitwise combination (excluding bit 6) of the Status Byte Register and the Service Request Enable Register.

*TRG
(Trigger Command)

*TRG

This command is the device-specific analog of the IEEE 488.1 Group Execute Trigger (GET) interface message (page 4-42), and has exactly the same effect.

The *TRG command will perform the action defined by the *DDT command (page 4-124).

*TST?
(Self-Test Query)

*TST?

This query causes an internal self-test and the response indicates whether any errors were detected.

Error -330 is generated when the self-test fails.

Query Response

- Numeric data transferred as ACSII bytes in <NR1> format.

- A response value of zero indicates the self-test has completed without errors detected, while a non-zero value indicates the self-test was not completed or was completed with errors detected.

Comments

The following are tested:

CPU,
ROM,
RAM,
EEPROM,
QSPI,
FPGA,
Front End,
Measurement hardware, and
Interpolator hardware.

*WAI
(Wait-to-Continue Command)

***WAI**

This command prevents the instrument from executing any further commands or queries until the measurement cycle transitions from measuring to idle. The only way to cancel this "holdoff" is by device clear or power-on. (*RST and *CLS have no affect on *WAI operation.)

See the section titled "Using the *WAI Command," page 3-46, in Chapter 3 for an example using this command.

5

Errors

Introduction

This chapter explains how to read any errors from the Counter, discusses the types of errors, and provides a table of all of the Counter's errors and their probable causes.

Displaying Errors

When an GPIB error is detected, the **HP-IB XXX** message will appear on the front-panel display, where XXX indicates the error number found in Table 5-2.

The front-panel error messages are most easily seen if the Counter is in Single (:INIT:CONT OFF) or in the menu display mode.
(When the Counter is configured to display measurement results,
the measurements will overwrite the GPIB error messages.)

Reading an Error

Executing the :SYSTem:ERRor? command reads the oldest error from the error queue and erases that error from the queue. The :SYST:ERR? response has the form:

<**error number**>, <**error string**>

An example response is:

-113, "Undefined header"

Positive error numbers are specific to the Counter. Negative error numbers are command language related and are discussed later in this chapter.

All errors set a corresponding bit in the Standard Event Status Register (see the section titled "Standard Event Status Register Group" on page 3-24 of Chapter 3).

The following short program reads all errors (one at a time, oldest to newest) from the error queue. After each error is read, it is automatically erased from the error queue. When the error queue is empty (that is, all errors have been read from the queue),
further queries return the **+0, "No error"** response.

```
10  ASSIGN @Cntr TO 703
20  !Assign path name
30  DIM Err_string$[255]
40  !Creates array for error string
50  REPEAT
60  !Repeats until error queue is empty
70      OUTPUT @Cntr;“SYST:ERR?”
80      !Read error number and string
90      ENTER @Cntr;Err_num,Err_string$
100     !Enter error number and string
110     PRINT Err_num,Err_string$
120     !Print error number and string
130 UNTIL Err_num = 0
140 END
```

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. That is, if there has been more than one error,
the first one in the queue is read out with :SYST:ERR?. Subsequent responses continue until the queue is empty.

If the error queue overflows, the last error in the queue is replaced with error **-350, “Queue overflow”.** Any time the queue overflows, the least recent errors remain in the queue, and the most recent error is discarded. The length of the Counter’s error queue is 30
(29 positions for the error messages, and 1 position for the “Queue overflow” error). Reading an error from the head of the queue removes that error from the queue, and opens a position at the tail of the queue for a new error, if one is subsequently detected.

When all errors have been read from the queue, further error queries return **+0, “No error”.**

The error queue is cleared when any of the following occur:

- Upon power-on.
- Upon receipt of a *CLS command.
- Upon reading the last item from the queue.

Error Types

Error numbers are categorized by type as shown in Table 5-1.
Each and every error is listed in Table 5-2.

Table 5-1. Error Types

Error Number	Error Type
+0	No Error
-100 to -199	Command Errors
-200 to -299	Execution Errors
-300 to -350	Device-Specific Errors
-400 to -499	Query Errors
+2000 to +2013	Counter-Specific Errors

The first error described in each class (for example, -100, -200, -300, -400) is a "generic" error.

No Error

The :SYST:ERR? response **+0, "No error"** indicates that the Counter has no errors. The error queue is empty when every error in the queue has been read (:SYST:ERR? query) or the queue was cleared by power-on or *CLS.

Command Error

An <error number> in the range [-100 to -199] indicates that an IEEE 488.2 syntax error has been detected by the Counter's parser. The occurrence of any error in this class causes the command error bit (bit 5) in the Event Status Register to be set. One of the following events has occurred:

- An IEEE 488.2 syntax error has been detected by the parser.
That is, a controller-to-Counter message was received that is in violation of the IEEE 488.2 Standard. Possible violations include a data element that violates the Counter listening formats or whose type is unacceptable to the Counter.

- An unrecognized header was received. Unrecognized headers include incorrect Counter-specific headers and incorrect or unimplemented IEEE 488.2 Common Commands.

- A Group Execute Trigger (GET) was entered into the input buffer inside of an IEEE 488.2 program message.

Events that generate command errors do not generate execution errors, device-specific errors, or query errors.

Execution Error

An <error number> in the range [-200 to -299] indicates that an error has been detected by the Counter's execution control block.
The occurrence of any error in this class causes the execution error bit (bit 4) in the Event Status Register to be set. One of the following events has occurred:

- A <PROGRAM DATA> element following a header was evaluated by the Counter as outside of its legal input range or is otherwise inconsistent with the Counter's capabilities.

- A valid program message could not be properly executed due to some Counter condition.

Execution errors are reported by the Counter after rounding and expression evaluation operations have been taken place. Rounding a numeric data element, for example, is not reported as an execution error. Events that generate execution errors do not generate command errors, device-specific errors, or query errors.

Device- or Counter-Specific Error

An <error number> in the range [-300 to -399] or [+1 to +32767] indicates that the Counter has detected an error that is not a command error, a query error, or an execution error; some Counter operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for
self-test response errors. The occurrence of any error in this class causes the device-specific error bit (bit 3) in the Event Status Register to be set.

Query Error

An <error number> in the range [-400 to -499] indicates that the output queue control of the Counter has detected a problem with the message exchange protocol. The occurrence of any error in this class should cause the query error bit (bit 2) in the Event Status Register to be set. One of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending.

- Data in the output queue has been lost.

Table 5-2. Errors

Number	Error String	Cause
+0	No error	The error queue is empty. Every error in the queue has been read (:SYSTem:ERRor? query) or the queue was cleared by power-on or *CLS.
-100	Command error	This is the generic syntax error used if the Counter cannot detect more specific errors.
-101	Invalid character	A syntactic element contains a character that is invalid for that type. For example, a header containing an ampersand, :INP:COUP& AC.
-102	Syntax error	An unrecognized command or data type was encountered.
-103	Invalid separator	The parser was expecting a separator and encountered an illegal character.
-104	Data type error	The parser recognized a data element different than one allowed. For example, numeric or string data was expected, but block data was received.
-105	GET not allowed	A Group Execute Trigger was received within a program message.
-108	Parameter not allowed	More parameters were received than expected for the header.
-109	Missing parameter	Fewer parameters were received than required for the header.
-112	Program mnemonic too long	The header or character data element contains more than twelve characters.
-113	Undefined header	The header is syntactically correct, but it is undefined for the Counter. For example, *XYZ is not defined for the Counter.
-120	Numeric data error	This error, as well as errors -121 through -129, are generated when parsing a data element which appears to be numeric, including the non-decimal numeric types. This particular error message is used when the Counter cannot detect a more specific error.
-121	Invalid character in number	An invalid character for the data type being parsed was encountered. For example, a “9” in octal data.
-123	Exponent too large	Numeric overflow.
-124	Too many digits	The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros.
-128	Numeric data not allowed	A legal numeric data element was received, but the Counter does not accept one in this position for the header.
-131	Invalid suffix	The suffix does not follow the syntax described in IEEE 488.2 or the suffix is inappropriate for the Counter.
-134	Suffix too long	The suffix contained more than 12 characters.
-138	Suffix not allowed	A suffix was encountered after a numeric element that does not allow suffixes.
-141	Invalid character data	The character data element contains an invalid character.
-148	Character data not allowed	A legal character data element was encountered where prohibited by the Counter.
-150	String data error	This error can be generated when parsing a string data element. This particular error message is used if the Counter cannot detect a more specific error.

Table 5-2. Errors (Continued)

Number	Error String	Cause
-151	Invalid string data	A string data element was expected but was invalid for some reason. For example, an END message was received before the terminal quote character.
-158	String data not allowed	A string data element was encountered but was not allowed by the Counter at this point in parsing.
-160	Block data error	This error can be generated when parsing a block data element. This particular error message is used if the Counter cannot detect a more specific error.
-161	Invalid block data	A block data element was expected, but it was not allowed by the Counter at this point in parsing.
-168	Block data not allowed	A legal block data element was encountered but was not allowed by the Counter at this point in parsing.
-170	Expression error	This error can be generated when parsing an expression data element. It is used if the Counter cannot detect a more specific error.
-171	Invalid expression	The expression data element was invalid (see IEEE 488.2). For example, unmatched parentheses or an illegal character.
-178	Expression data not allowed	Expression data was encountered but was not allowed by the Counter at this point in parsing.
-181	Invalid outside macro definition	Indicates that a macro parameter placeholder ($<number>) was encountered outside of a macro definition.
-183	Invalid inside macro definition	Indicates that the program message unit sequence, sent with a *DMC command, is syntactically invalid.
-200	Execution error	This is the generic syntax error if the Counter cannot detect more specific errors. This code indicates only that an Execution Error has occurred.
-210	Trigger error	Used if the Counter cannot detect a more specific error from the :INIT, :TRIG, or :ABOR subsystems.
-211	Trigger ignored	Indicates that a GET or *TRG was received and recognized by the Counter but was ignored.
-213	Init ignored	Indicates that a request for a measurement initiation was ignored as another measurement was in progress.
-220	Parameter error	Indicates that a program data element related error occurred. This error is used when the Counter cannot detect more specific errors.
-221	Settings conflict	Indicates that a legal program data element was parsed but could not be executed due to the current Counter state.
-222	Data out of range	Indicates that a legal program data element was parsed but could not be executed because the interpreted value is outside the legal range defined by the Counter. Typically, the value is clipped to legal limit.
-223	Too much data	Indicates that a legal program data element of block, expression, or string type was received that contained more data than the Counter could handle due to memory or related Counter-specific requirements.

Table 5-2. Errors (Continued)

Number	Error String	Cause
-224	Illegal parameter value	Used where exact value, from a list of possible values, was expected.
-230	Data corrupt or stale	No valid data available. New measurement started but not completed.
-240	Hardware error	Indicates that a legal program command or query could not be executed because of a hardware problem in the Counter.
-241	Hardware missing	Indicates that a legal program command or query could not be executed because of missing Counter hardware. For example, the Channel 3 option was not installed.
-272	Macro execution error	Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition.
-273	Illegal macro label	Indicates that the macro label defined in the *DMC command was a legal string syntax, but it could not be accepted by the Counter. For example, the label was too long, the same as a common command header, or contained invalid header syntax.
-276	Macro recursion error	Indicates that a syntactically legal macro program data sequence could not be executed because the Counter found the maximum recursion level of four was exceeded.
-277	Macro redefinition not allowed	Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined (see IEEE 488.2).
-278	Macro header not found	Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.
-300	Device-specific error	This is the generic device-dependent error.
-310	System error	Indicates that a system error occurred.
-321	Out of memory	Indicates that the Counter has detected that insufficient memory is available. For example, this error will eventually occur on a *DMC, once the macro memory is filled with previously defined macros.
-330	Self-test failed	Indicates at least one failure occurred when *TST? was executed.
	Self-test failed; CPU failure	Power-on self test detected this hardware failure.
	Self-test failed; ROM failure	Power-on self test detected this hardware failure.
	Self-test failed; RAM failure	Power-on self test detected this hardware failure.
	Self-test failed; EEPROM failure	Power-on self test detected this hardware failure.
	Self-test failed; GPIB failure	Power-on self test detected this hardware failure.
	Self-test failed; QSPI failure	Power-on self test detected this hardware failure.
	Self-test failed; FPGA failure	Power-on self test detected this hardware failure.
	Self-test failed; front-end failure	Power-on self test detected this hardware failure.
	Self-test failed; measurement failure	Power-on self test detected this hardware failure.
	Self-test failed; interpolator failure	Power-on self test detected this hardware failure.
-350	Queue overflow	Indicates that there is no room in the error queue and an error occurred but was not recorded.

Table 5-2. Errors (Continued)

Number	Error String	Cause
-400	Query error	This is the generic query error.
-410	Query INTERRUPTED	Indicates that a condition causing an INTERRUPTED Query error occurred. For example, a query followed by DAB or GET before a response was completely sent.
-420	Query UNTERMINATED	Indicates that a condition causing an UNTERMINATED Query error occurred. For example, the Counter was addressed to talk and an incomplete program message was received.
-430	Query DEADLOCKED	Indicates that a condition causing a DEADLOCKED Query error occurred. For example, both input buffer and output buffer are full and the Counter cannot continue.
-440	Query UNTERMINATED after indefinite response	Indicates that a query was received in the same program message after a query requesting an indefinite response (for example, *IDN? or *OPT?) was executed.
+2000	Offset calibration on channel 1 failed	:DIAGnostic:CALibration:INP:OFFS:AUTO ONCE failed.
+2001	Offset calibration on channel 2 failed	:DIAGnostic:CALibration:INP2:OFFS:AUTO ONCE failed.
+2002	Gain calibration on channel 1 failed	:DIAGnostic:CALibration:INP:GAIN:AUTO ONCE failed.
+2003	Gain calibration on channel 2 failed	:DIAGnostic:CALibration:INP2:GAIN:AUTO ONCE failed.
+2004	Interpolator calibration failed	:DIAGnostic:CALibration:INTerpolator:AUTO ONCE, *CAL?, or :CALibration[:ALL]? failed.
+2005	Oscillator calibration failed	:DIAGnostic:CALibration:ROSCillator:AUTO ONCE failed.
+2006	Time interval calibration failed	:DIAGnostic:CALibration:TINTerval:QUICk failed.
+2007	Measurement hardware calibration failed	A measurement calibration failed on the last measurement.
+2008	Measurement interpolator calibration failed	Interpolator calibration failed on the last measurement; therefore, no valid measurement was taken.
+2009	No valid external timebase	You have selected external reference and there is no external reference applied to the rear-panel Ref In connector, or the external signal is not an allowed frequency.
+2010	External timebase failed during measurement	You have selected the auto reference mode and the Counter detected that the external reference became invalid *during* the measurement. Therefore, the current result is not valid, and the Counter switches to using the internal reference.
+2011	Recall setup failed; hardware failure Recall setup failed; empty register	*RCL failed. *RCL failed because the register specified for recall is empty.
+2012	Save setup failed	*SAV failed, or *RCL couldn't save to register 0.
+2013	EEPROM failed	A request to update a setting which is stored in the EEPROM (:SYST:COMM:SER:TRAN:BAUD, :SYST:COMM:SER:TRAN:PAR, :SYST:COMM:SER:TRAN:PACE, :SYST:COMM:SER:CONT:DTR, :DISP[:WIND]:TEXT:RAD, :CAL[:DATA], or :DIAG:CAL: ...) resulted in a hardware failure.

Index

*****

*RST, 4-137
  affected setup, 2-40
  unaffected setup, 2-40, 2-43
*RST Response, 2-40
*RST summary list, 2-40

:

:CALibration
  SECurity, 4-28

<

<numeric value>, 3-12

9

9.91E37, 3-17

A

abbreviated commands, 3-9
address, GPIB, 3-4
Agilent 53131A/132A SCPI Subsystem Commands, 2-24
Applications, 1-8
arbitrary block, 3-11
ASCII, 4-41
ASCII format, 3-59
Assumptions, 1-9
auto-trigger, 4-44, 4-45, 4-83

B

Basic, using, 3-58
Boolean, 3-11

C

calibrating status bit, 3-30
calibration, 4-26
  reference oscillator, 4-34
  remote, 4-32
  security code, 4-28
  security state, 4-28
  unsecure, 4-28
calibration error status bit, 3-32, 3-33
clear, 4-123
CME, 3-24
comma, 3-8, 3-12, 3-16
command
  abbreviated, 3-9
  keyword, 3-7
  parameter, 3-11
  short form, 3-53
  syntax, 3-9
  terminator, 3-13
  terminators, 3-8
command error
  definition, 5-4
  status bit, 3-26
command warning status bit, 3-32, 3-34
commands to set Counter for optimal throughput, 3-35
common command
  format, 3-7
  syntax, 3-8
common commands
  definition, 2-21
  summary list, 2-22
  syntax, 2-21
Common Commands Summary Table, 2-22
Common Commands, IEEE 488.2, 4-122
  *CAL?, Calibration, 4-122
  *CLS, Clear Status, 4-123
  *DDT, Define Device Trigger Command, 4-124
  *DMC, Define Macro Command, 4-125
  *EMC, Enable Macro Command, 4-126
  *EMC?, Enable Macro Command, 4-126
  *ESE, Standard Event Status Enable, 4-127
  *ESE?, Standard Event Status Enable Query, 4-127
  *GMC?, Get Macro Contents Query, 4-129
  *IDN?, Identificarion Query, 4-130
  *LMC?, Learn Macro Query, 4-131
  *OPC, Operation Complete, 4-132
  *OPC?, Operation Complete Query, 4-133
  *OPT?, Option Identification, 4-134
  *PMC, Purge Macro Command, 4-135
  *RCL, Recall, 4-136
  *RST, Reset, 4-137
  *SAV, Save, 4-138
  *SRE, Service Request Enable, 4-139
  *SRE?, Service Request Enable Query, 4-139
  *STB?, Status Byte Query, 4-140
  *TRG, Trigger, 4-141
  *TST?, Self-Test Query, 4-142
  *WAI, Wait-to-Continue, 4-143
  ESR?, Event Status Register Query, 4-128
common/separate, 4-81
computing statistics status bit, 3-30, 3-31

condition register, 3-27, 3-28
CONFigure with INITiate and FETCh?
  using, 4-76
CONFigure with READ
  using, 4-76
configuring the GPIB, 3-4
conformance
  IEEE488.1, 3-7
  IEEE488.2, 2-20, 2-21, 3-7
  SCPI, 2-20, 3-7
connecting the Counter to a computer, 3-6
connector
  RS-232, 4-114

D

data, measurement, 4-80
date code, firmware, 4-130
DCL, 4-31
DDE, 3-24
DDT, 4-124
decimal point, 3-11, 3-17
DEG, 3-12
Device Clear, 4-31
device trigger, 4-124
device-dependent error
  definition, 5-5
device-specific error
  status bit, 3-26
differences between firmware revisions, 1-3
display, 4-38
  enable, 4-37
display results, 3-43
displaying errors, 5-2
double-quoted string
  sending a double-quoted string, 3-58
duty cycle, 4-61, 4-89, 4-96

E

EOI, 3-13
error
  command, 5-4
  displaying, 5-2
  execution, 5-5
  how to query, 5-2
  list, 5-7
  messages, 5-7
  query, 3-25, 4-116, 5-6
  queue, 5-3
  type, 5-4
errors list, 5-7
ESB, 3-22, 3-23
event enable register, 3-27, 3-29
event register, 3-27, 3-29
EXE, 3-24
execution error
  definition, 5-5
  status bit, 3-26

F

fail limit test, 4-14
fall time, 4-62, 4-89, 4-96
firmware
  date code, 4-130
  revision, 4-130
firmware revision, 2-32, 4-81
firmware revisions, 1-3, 2-31, 2-34
format
  ASCII, 4-41
  REAL, 4-41
frequency, 4-63, 4-85, 4-89
frequency ratio, 4-65, 4-89
front panel to SCPI command maps, 2-3
function, 4-60, 4-89

G

GET, 4-42
Getting Started, 1-6
GPIB
  configuration, 3-4
  operating modes, 3-4
GPIB operating modes
  Addressed (talk/listen), 3-4
  Talk-only, 3-4
group execute trigger, 4-42
Group Execute Trigger, GET, 4-42

H

How to Use This Guide, 1-6
hysteresis, 4-81
HZ, 3-12

I

IEEE 488.2
  summary list, 2-22
IEEE 488.2 Common Commands, 2-21
IEEE488.1
  conformance, 3-7
  obtaining copy of standard, 1-10
IEEE488.2
  common commands, 2-21
  conformance, 2-20, 2-21
  obtaining copy of standard, 1-11
  syntax, 3-8
implied channel, 3-10
in limit event status bit, 3-30, 3-31
initiate, 4-44
initiate measurements, 4-44
input
  attenuation, 4-48
  coupling, 4-48, 4-50

impedance, 4-49, 4-50
low-pass filter, 4-48
interpolator
automatic calibration, 4-32
calibration, 4-26, 4-33

K

key queue, 4-117
keyword, 3-9
optional, 3-9
separator, 3-9

L

Learning to Program the Counter, 1-7
level, 4-83
limit
fail, 4-14
fail count, 4-15
graph, 4-14
lower, 4-16
pass count, 4-17
test, 4-11
enable, 4-17
upper, 4-18
limit testing, 4-11
list of errors, 5-7
literal, 3-11, 3-18
local, 3-6

M

macros, 3-52
define, 3-52, 4-125
get contents, 4-129
labels, 4-131
memory, 4-79
purge, 4-135
math, 4-7
enable, 4-10
offest, scale, 4-7
offset, 4-10
scale, 4-10
math/limit operations, 3-49
MAV, 3-22
MAXimum, 3-12, 4-19
maximum value, 3-11
mean, 4-19
measure, 4-52
using, 4-75
measurement functions, 4-89
See function, 4-60
measurement instructions commands
definition, 4-52
measuring status bit, 3-30, 3-31
memory, 4-79
messages
program, 3-14
response, 3-16
MINimum, 3-12, 4-19
minimum value, 3-11
multipliers, 3-13

N

N, 4-20
negative pulse width, 4-67, 4-89
negative transition filter register, 3-27
new line, 3-11, 3-18
non-decimal numeric, 3-11
Not a Number
9.91E37, 3-17
NR1, 3-17
NR2, 3-17
NR3, 3-17
NRf, 3-11

O

offset/scale, 4-7
OHM, 3-12
OPC, 3-24
operation complete, 4-132
operation complete status bit, 3-24, 3-25
operation status register group, 3-27, 3-30
optimizing throughput, 3-35
Optimizing Throughput Results for Different Computers, 3-37
optional keyword, 3-9
options
identifying, 4-134
OSB, 3-22, 3-23
oscillator
reference, 4-92
out of limit event status bit, 3-32, 3-34

P

parameter separator, 3-12
parameter types, 3-11
Boolean, 3-11
literal, 3-11
string, 3-11
pass limit test, 4-17
PCT, 3-12
peak-to-peak voltage, 4-70, 4-89
period, 4-68, 4-89
phase, 4-70, 4-89, 4-91
PON, 3-24
positive pulse width, 4-70, 4-89
positive transition filter register, 3-27
post-processing, 4-9, 4-19
power on status bit, 3-24, 3-26
power-on, 3-23, 3-26, 3-30
preset, 4-110

print, 4-43
prior firmware revisions
  3317, 3335, and 3402, 2-32
  3317, 3335,and 3402, 4-81
program messages
  definition, 3-14
  syntax, 3-14
program the Counter for math/limit operations, 3-49
program the Counter for status reporting, 3-38
program the Counter to define macros, 3-52
program the Counter to display results, 3-43
Program the Counter to Synchronize Measurements, 3-46
programming examples, 3-58
programming for
  display results, 3-43
  macros, 3-52
  math/limit operations, 3-49
  status reporting, 3-38
  synchronizing measurements, 3-46
Programming Guide Contents, 1-9
programs
  examples, 3-59
  writing SCPI (reference flowchart), 3-55
pulse width, 4-96

Q

QSB, 3-22
query, 3-12, 3-16
query error, 5-6
query parameters
  <numeric value>, 3-12
  MAXimum, 3-12
  MINimum, 3-12
questionable data/signal status register group, 3-27, 3-32, 4-110
queue
  error, 5-2
  key, 4-117
  output, 4-133
QuickBASIC, using, 3-59
QYE, 3-24

R

radix, 3-17
ratio, 4-89
ratio, frequency, 4-65
reading an error, 5-2
REAL, 4-41
recall, 4-136
reference oscillator
  calibration, 4-34
reference, oscillator, 4-92
Related Documentation, 1-10
remote, 3-6
reset, 4-137
response message syntax, 3-16
response messages, 3-16
  data types, 3-17
results
  display, 4-38
  math, 4-7
  query, 4-7, 4-14, 4-19
  raw, 4-38
  statistics, 4-19
revision
  current firmware, 2-32, 4-81
  firmware, 1-3, 2-31, 2-34, 4-130
  prior firmware (3317, 3335, and 3402), 2-32
revisions
  prior (3317,3335, and 3402), 4-81
rise time, 4-71, 4-89, 4-96
RQS/MSS, 3-22, 3-23
RS-232, 4-43
RS-232 connector, 4-114

S

S, 3-12
save, 4-138
scale, 4-7
scale/offset, 4-7
SCPI
  command format, 3-7
  conformance, 2-20
  obtaining copy of standard, 1-10
  query format, 3-7
  version, 1-2, 2-20, 4-118
SCPI Command Summary Table, 2-25
SCPI programs, how to write, 3-55
SDCL, 4-31
security code, 4-28
security state, 4-28
selected device clear, 4-31
self-calibration, 4-26
self-test, internal, 4-142
sensitivity, 4-81
separate/common, 4-81
separator
  keyword, 3-9
  parameter, 3-12
serial control, 4-114
serial port, 4-43
service request enable register, 3-23, 4-139
short form, 3-53
single-quoted string
  sending a single-quoted string, 3-58
slope, 4-84

standard deviation, 4-19
standard event status enable register, 3-26, 4-127
standard event status register, 3-24, 4-128
statistics, 4-19
  enable, 4-21
  filter, 4-23
  maximum, 4-22
  mean, 4-22
  minimum, 4-22
  N, 4-20
  results, 4-19
  standard deviation, 4-22
stats, 4-19
status
  operation, 4-107
  preset, 4-110
status byte register, 3-21, 4-140
status reporting, 3-38
  flowchart, 3-41
  summary of all registers, 3-19
stop, 3-10
string, 3-11, 3-18
  BASIC, 3-58
  parameters, 3-58
subsystem command
  syntax, 3-8
Subsystem Commands, 4-4
  \:ABORt, 4-4
  \:CALCulate, 4-5
  \:CALCulate[1], 4-7
    \:CALCulate[1]\:DATA?, 4-7
    \:CALCulate[1]\:FEED, 4-8
    \:CALCulate[1]\:IMMediate, 4-8
    \:CALCulate[1]\:IMMediate\:AUTO, 4-9
    \:CALCulate[1]\:MATH, 4-9
  \:CALCulate2
    \:CALCulate2\:FEED, 4-11
    \:CALCulate2\:IMMediate, 4-11
    \:CALCulate2\:IMMediate\:AUTO, 4-12
    \:CALCulate2\:LIMit, 4-12
  \:CALCulate3, 4-19
    \:CALCulate3\:AVERage, 4-19
    \:CALCulate3\:DATA?, 4-22
    \:CALCulate3\:FEED, 4-23
    \:CALCulate3\:LFILter, 4-23
    \:CALCulate3\:PATH?, 4-25
  \:CALibration, 4-26
    \:CALibration\:COUNt?, 4-26
    \:CALibration\:DATA, 4-27
    \:CALibration\:SECurity, 4-28
    \:CALibration[\:ALL]?, 4-26
  \:CONFigure, 4-30
  \:DIAGnostic, 4-32
    \:DIAGnostic\:CALibration\:INPut[1|2]\:GAIN\:AUTO, 4-32
    \:DIAGnostic\:CALibration\:INPut[1|2]\:OFFSet\:AUTO, 4-33
    \:DIAGnostic\:CALibration\:INTerpolator\:AUTO, 4-33
    \:DIAGnostic\:CALibration\:ROSCillator\:AUTO, 4-34
    \:DIAGnostic\:CALibration\:STATus?, 4-34
    \:DIAGnostic\:CALibration\:TINTerval\:FINE, 4-35
    \:DIAGnostic\:CALibration\:TINTerval\:QUICk, 4-36
  \:DISPlay, 4-37
    \:DISPlay\:[WINDow]\:TEXT\:FEED, 4-38
    \:DISPlay\:ENABle, 4-37
    \:DISPlay\:MENU[\:STATe], 4-37
    \:DISPlay[\:WINDow]\:TEXT\:RADix, 4-39
  \:FETCh, 4-40
  \:FORMat, 4-41
    \:FORMat[\:DATA], 4-41
  \:HCOPy, 4-43
  \:INITiate, 4-44
    \:INITiate\:AUTO, 4-44
    \:INITiate\:CONTinuous, 4-44
    \:INITiate[\:IMMediate], 4-46
  \:INPut[1|2], 4-48
    \:INPut[1|2]\:ATTenuation, 4-48
    \:INPut[1|2]\:COUPling, 4-48
    \:INPut[1|2]\:FILTer[\:LPASs]\:FREQuency?, 4-49
    \:INPut[1|2]\:FILTer[\:LPASs][\:STATe], 4-48
    \:INPut[1|2]\:IMPedance, 4-49
  \:INPut3, 4-50
    \:INPut3\:COUPling?, 4-50
    \:INPut3\:IMPedance?, 4-50
  \:MEASure, 4-51
  \:MEMory, 4-79
    \:MEMory\:DELete\:MACRo, 4-79
    \:MEMory\:FREE\:MACRo?, 4-79
    \:MEMory\:NSTates?, 4-79
  \:STATus, 4-107
    \:STATus\:OPERation, 4-107
    \:STATus\:PRESet, 4-110
    \:STATus\:QUEStionable, 4-110
  \:SYSTem, 4-114
    \:SYSTem\:COMMunicate, 4-114
    \:SYSTem\:ERRor?, 4-116
    \:SYSTem\:KEY, 4-117
    \:SYSTem\:KEY\:LOG?, 4-118

\:SYSTem\:VERSion?, 4-118
\:TRACe, 4-119
\:TRACe\:CATalog?, 4-119
\:TRACe[\:DATA] OFFSET, 4-119
\:TRACe[\:DATA] SCALE, 4-120
\:TRIGger, 4-121
\:TRIGger\:TRIGger\:COUNt\:AUTO, 4-121
[\:SENSe]
[\:SENSe]\:DATA?, 4-80
[\:SENSe]\:EVENt[1|2], 4-80
[\:SENSe]\:EVENt3, 4-84
[\:SENSe]\:FREQuency, 4-85
[\:SENSe]\:FREQuency\:EXPected[1|2|3], 4-88
[\:SENSe]\:FREQuency\:EXPected[1|2|3]\:AUTO ON, 4-88
[\:SENSe]\:FUNCtion[\:ON], 4-89
[\:SENSe]\:PHASe, 4-91
[\:SENSe]\:ROSCillator, 4-92
[\:SENSe]\:TINTerval, 4-95, 4-98
[\:SENSe]\:TOTalize, 4-104
Measurement Instructions, 4-52
\:CONFigure, 4-53, 4-54
\:CONFigure?, 4-54
\:FETCh?, 4-55
\:MEASure query, 4-56
\:READ?, 4-57
Descriptions of the Measurement Functions—<function>, 4-60
How to Use the Measurement Instruction Commands, 4-75
Subsystem Commands:CALCulate2, 4-11
Subsystem Commands[\:SENSe], 4-80
suffix
elements, 3-12
multiplers, 3-13
suffix, multipler, 3-13
suffixes, 3-12
summary bits, 3-21
synchronizing measurements, 3-46
syntax
program messages, 3-14
response messages, 3-16

T

terminator
command, 3-13
throughput, 3-35
time interval, 4-73, 4-89, 4-95, 4-98
totalize, 4-73, 4-74, 4-89, 4-104
transition filter, 3-28
negative, 4-112
positive, 4-113
trigger, 4-84
device, 4-124
levels, 2-5, 2-6, 4-82, 4-83
sensitivity, 2-5, 2-6, 4-81
slope, 2-5, 2-6, 4-84
trigger command, 4-141
trigger events, 4-80
Turbo C, using, 3-59

U

Unaffected by *RST, 2-43
units, 3-12
unsecure, 4-28
using internal reference status bit, 3-30, 3-31
using the scale and offset over GPIB, 3-50

V

V, 3-12
version, SCPI, 4-118
voltage
maximum, 4-89
minimum, 4-89
voltage maximum, 4-66
voltage minimum, 4-66

W

writing programs, general, 3-55

X

x1, 4-48
x10, 4-48