

Keysight B2980B Series
Femto/Picoammeter
Electrometer/High Resistance Meter

Notices

Copyright Notice

© Keysight Technologies 2021

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies as governed by United States and international copyright laws.

Manual Part Number

B2980-90120

Edition

Edition 1, April 2021

Published by:

Keysight Technologies Japan K.K.
9-1, Takakura-cho, Hachioji-shi, Tokyo
192-8550 Japan

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S. Government Rights

The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement (“DFARS”) 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at <http://www.keysight.com/find/sweula>. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set

forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS,” AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR OF ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE

WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT SHALL CONTROL.

Open Software License

A portion of the software in this product is licensed under terms of the General Public License Version 2 (“GPLv2”). The text of the license and source code can be found at:

www.keysight.com/find/GPLV2

Declaration of Conformity

Declarations of Conformity for this product and for other Keysight products may be downloaded from the Web. Go to <http://www.keysight.com/go/conformity>. You can then search by product number to find the latest Declaration of Conformity.

Latest Information

To get the latest firmware/software/electronic manuals/specifications/support information, go to www.keysight.com and type in the product number in the Search field at the top of the page.

In This Manual

This manual provides the information for controlling Keysight Technologies B2980 by using an external computer, and consists of the following chapters.

- [Chapter 1, “Controlling Keysight B2980”](#)
- [Chapter 2, “Programming Examples”](#)

See *Keysight B2980B User's Guide* for information about the B2980 itself.

Refer to *Keysight B2980B SCPI Command Reference* for the SCPI messages and conventions, data output format, error code, and the details on Keysight B2980 SCPI commands.

Contents

1 Controlling Keysight B2980

Before Starting	10
Software Requirements	10
Connecting to the Interface.	10
Starting the Instrument Control	10
Controlling Various Functions.	12
Setting the Power Frequency	12
Resetting to the Initial Settings.	13
Setting the Beeper.	13
Setting the Date and Time.	13
Performing the Self-Test	13
Performing the Self-Calibration	14
Setting the Operations at Power On	14
Reading an Error Message	14
Clearing the Error Buffer	14
Reading Timestamp	15
Clearing Timestamp	15
Setting the Automatic Clear of Timestamp.	15
Confirming the Firmware Revision	15
Setting the Remote Display Mode.	16
Making a Screen Dump	16
Performing a File Operation.	16
Setting the Display Format of Measured Value	17
Controlling the Measurement Function	18
Enabling the Current/Charge Measurement.	18
Setting the Measurement Mode	18
Performing Spot Measurement	18
Setting the Measurement Speed.	19
Setting the Measurement Range.	19
Setting the Measurement Trigger	19

Setting the Resistance Measurement Result to Absolute Format.	20
Setting the Attenuation Level for Voltage Input	20
Setting the Measurement Wait Time	21
Performing Sweep Measurement	21
Stopping Measurement	22
Controlling the Source Output	23
Enabling the Source Output	23
Applying the DC Voltage	23
Stopping the Source Output	24
Setting the Output Range.	24
Setting the Sweep Operation	24
Setting the Sweep Output	25
Setting the Ranging Mode of the Sweep Source	25
Setting the List Sweep Output	25
Setting the Source Output Trigger.	26
Setting the Low Terminal State	26
Specifying the Output-Off Status.	26
Setting the Source Wait Time.	27
Using the Math Function	28
Defining a Math Expression	28
Deleting an User Defined Math Expression	28
Enabling or Disabling the Math Function.	28
Reading Math Result Data	28
Performing the Limit Test.	30
Setting the Composite Limit Test.	31
Setting Individual Limit Tests	31
Reading Limit Test Result.	31
Using the Trace Buffer	32
Setting the Trace Buffer	33
Reading the Trace Data	33

Using Program Memory	34
Defining a Memory Program	34
Deleting a Program	34
Controlling the Program Operation	34

2 Programming Examples

Preparations	38
To Create Your Project Template	38
To Create Measurement Program	39
Spot Measurements	42
Staircase Sweep Measurements	44
List Sweep Measurements	48
Sampling Measurements	52
Pass/Fail Judgment and Math Function	56
Using Program Memory	62
Reading Binary Data	65
Reading Data During Measurement	67

1 Controlling Keysight B2980

- Before Starting 10
- Controlling Various Functions 12
- Controlling the Measurement Function 18
- Controlling the Source Output 23
- Using the Math Function 28
- Performing the Limit Test 30
- Using the Trace Buffer 32
- Using Program Memory 34

This chapter describes basic information to control Keysight B2980.

Table 1-1

Conventions Used in This Document for Expressing SCPI Commands

Convention	Description
Angle brackets < >	Items within angle brackets are parameter abbreviations. For example, <NR1> indicates a specific form of numerical data.
Vertical bar	Vertical bars separate alternative parameters. For example, VOLT CURR indicates that either “VOLT” or “CURR” can be used as a parameter.
Square brackets []	Items within square brackets are optional. The representation [SENSe:]VOLTage means that SENSe: may be omitted.
Parentheses ()	Items within parentheses are used in place of the usual parameter types to specify a channel list. The notation (@1:3) specifies a channel list that includes channels 1, 2, and 3. The notation (@1,3) specifies a channel list that includes only channels 1 and 3.
Braces { }	Braces indicate parameters that may be repeated zero or more times. It is used especially for representing arrays. The notation <A>{,} shows that parameter “A” must be entered, while parameter “B” omitted or may be entered one or more times.

Before Starting

This section describes the information needed before starting programming.

- “Software Requirements”
- “Connecting to the Interface”
- “Starting the Instrument Control”

Software Requirements

Programming examples described in this manual use the following software. Install the software to your computer to execute the programming examples.

- Keysight IO Libraries Suite software
- Microsoft Visual Basic .NET software

Connecting to the Interface

Keysight B2980 supports GPIB, LAN, and USB interfaces. All three interfaces are live at power-on. Select the interface used for controlling the B2980. Connect your interface cable to the appropriate interface connector.

For the information on configuring the interfaces, see *Keysight B2980 Series User's Guide*.

Starting the Instrument Control

The following program code is one of the simple program template for starting and ending the communication between the computer and the instrument. For using the code, the instrument address must be set to the `address` variable correctly.

```
Sub Main()  
    Dim rm As Ivi.Visa.Interop.ResourceManager  
    Dim ioObj As Ivi.Visa.Interop.FormattedIO488  
    Dim address As String = "enter address of your instrument"  
  
    rm = New Ivi.Visa.Interop.ResourceManager  
    ioObj = New Ivi.Visa.Interop.FormattedIO488  
    ioObj.IO = rm.Open(address)  
        'insert your code for instrument control  
    ioObj.IO.Close()  
End Sub
```

The **address** value depends on the interface as shown below.

- For using the GPIB interface

The **address** value is the VISA GPIB Connect String displayed on the GPIB Configuration dialog box opened by pressing the System Menu > More > I/O > GPIB function keys.

Example:

address = "GPIB0::23::INSTR"

- For using the USB interface

The **address** value is the VISA USB Connect String displayed on the USB Status dialog box opened by pressing the System Menu > More > I/O > USB function keys.

Example:

address = "USB0::10893::39681::XY00001234::0::INSTR"

- For using the LAN interface

The **address** value is as follows.

address = "TCPIP0::xxx.yyy.zzz.aaa::5025::SOCKET"

Where, *xxx.yyy.zzz.aaa* is the IP Address displayed on the LAN Configuration dialog box opened by pressing the System Menu > More > I/O > LAN > Config function keys.

Example:

address = "TCPIP0::192.168.0.1::5025::SOCKET"

Controlling Various Functions

This section describes how to control various functions apart from the source output and measurement functions.

- “Setting the Power Frequency”
- “Resetting to the Initial Settings”
- “Setting the Beeper”
- “Setting the Date and Time”
- “Performing the Self-Test”
- “Performing the Self-Calibration”
- “Setting the Operations at Power On”
- “Reading an Error Message”
- “Clearing the Error Buffer”
- “Reading Timestamp”
- “Clearing Timestamp”
- “Setting the Automatic Clear of Timestamp”
- “Confirming the Firmware Revision”
- “Setting the Remote Display Mode”
- “Making a Screen Dump”
- “Performing a File Operation”
- “Setting the Display Format of Measured Value”

Setting the Power Frequency

Power line frequency is set by the :SYST:LFR command.

Example

```
ioObj.WriteString(":SYST:LFR 50") '50 Hz  
ioObj.WriteString(":SYST:LFR 60") '60 Hz  
ioObj.WriteString(":SYST:LFR:DET:AUTO") 'Automatic Detection
```

Resetting to the Initial Settings

The initial settings are applied by the *RST command

Example `ioObj.WriteString("*RST")`

For the initial settings, see *SCPI Command Reference*.

Setting the Beeper

Beeper is enabled/disabled by the :SYST:BEEP:STAT command. And a beep sound of the specified frequency and duration is generated by the :SYST:BEEP command.

Example `ioObj.WriteString(":SYST:BEEP:STAT ON")` 'Enables beep
`ioObj.WriteString(":SYST:BEEP 200,1")` '200 Hz, 1 s

Setting the Date and Time

Date is set by the :SYST:DATE command. And time is set by the :SYST:TIME command.

Example `ioObj.WriteString(":SYST:DATE 2011,1,1")` 'Y,M,D
`ioObj.WriteString(":SYST:TIME 23,59,59")` 'H,M,S

Performing the Self-Test

Self-test is performed by the *TST? command. The *TST? command also returns the execution result. Before performing the self-test, disconnect test leads and cables from the channel terminals.

Example `ioObj.WriteString("*TST?")`
`Dim d As String = ioObj.ReadString()`
`If d = 0 Then`
 `Console.WriteLine("PASS")`
`Else`
 `Console.WriteLine("FAIL")`
`End If`

This example performs the self-test, and displays the test result, pass or fail.

Performing the Self-Calibration

Self-calibration is performed by the `*CAL?` command. The `*CAL?` command also returns the execution result. Before performing the self-calibration, disconnect test leads and cables from the channel terminals.

Example

```
ioObj.WriteString("*CAL?")
Dim d As String = ioObj.ReadString()
If d = 0 Then
    Console.WriteLine("PASS")
Else
    Console.WriteLine("FAIL")
End If
```

This example performs the self-calibration, and displays the result, pass or fail.

Setting the Operations at Power On

Operations at power-on are decided by the memory program specified by the `:PROG:PON:COPY` command. And the power-on program execution is enabled/disabled by the `:PROG:PON:RUN` command. The specified program must be previously defined in the program memory.

Example

```
ioObj.WriteString(":PROG:PON:COPY ""program1""")
ioObj.WriteString(":PROG:PON:RUN ON")
```

This example sets *program1* to the power-on program and enables the function.

Reading an Error Message

Error message is read one by one by using the `:SYST:ERR?` command. This command reads and removes the top item in the error buffer, and returns the code and message.

Example

```
ioObj.WriteString(":SYST:ERR?")
Dim d As String = ioObj.ReadString()
Console.WriteLine(d)
```

If the error buffer is empty, the response is `+0,"No error"`.

Clearing the Error Buffer

Error buffer is cleared by the `:SYST:ERR:ALL?` command. This command reads and returns all items in the error buffer, and clears the buffer.

Example

```
ioObj.WriteString(":SYST:ERR:ALL?")
Dim d As String = ioObj.ReadString()
Console.WriteLine(d)
```

If the error buffer is empty, the response is +0, "No error".

Reading Timestamp

Timestamp is read by the :SYST:TIME:TIM:COUN? command.

Example

```
ioObj.WriteString(":SYST:TIME:TIM:COUN?")
Dim d As String = ioObj.ReadString()
Console.WriteLine(d)
```

Clearing Timestamp

Timestamp is cleared by the :SYST:TIME:TIM:COUN:RES command.

Example

```
ioObj.WriteString(":SYST:TIME:TIM:COUN:RES")
```

Setting the Automatic Clear of Timestamp

Automatic clear of timestamp is enabled/disabled by the :SYST:TIME:TIM:COUN:RES:AUTO command. If this function is enabled, the timestamp is cleared when the initiate action occurs.

Example

```
ioObj.WriteString(":SYST:TIME:TIM:COUN:RES:AUTO ON")
```

Confirming the Firmware Revision

Instrument's (mainframe) identification and firmware revision are read by the *IDN? command.

Example

```
ioObj.WriteString("*IDN?")
Dim d As String = ioObj.ReadString()
Console.WriteLine(d)
```

The returned value will be as follows.

Keysight Technologies,model,serial,revision

model: mainframe model number

serial: mainframe serial number

revision: firmware revision number

Setting the Remote Display Mode

Front panel display under remote operation is enabled or disabled by the :DISP:ENAB command.

Example `ioObj.WriteString(":DISP:ENAB ON")`

Making a Screen Dump

Screen dump of the front panel display is made by the :HCOP:SDUM commands.

Example

```
ioObj.WriteString(":DISP:ENAB ON")
ioObj.WriteString(":DISP:VIEW GRAP")
ioObj.WriteString(":HCOP:SDUM:FORM JPG")
ioObj.WriteString("*OPC?") : s = ioObj.ReadString()
ioObj.WriteString(":HCOP:SDUM:DATA?")

Dim data As Object
data = ioObj.ReadIEEEBlock(Ivi.Visa.Interop.IEEEBinaryType.BinaryType_UI1,
False, True)

Dim dataSize As Integer = data.Length
Dim dumpname As String =
"../../../../execution result/screendump1.jpg"
Using stream As New FileStream(dumpname, FileMode.Create, FileAccess.Write)
    stream.Write(data, 0, dataSize)
End Using
```

Performing a File Operation

File operation is effective for the USB flash drive connected to the front panel USB connector, and performed by the :MMEM commands. Error occurs if an USB flash drive is not connected.

Example

```
ioObj.WriteString(":MMEM:CAT?") 'Gets file catalog
s = ioObj.ReadString()

ioObj.WriteString(":MMEM:STOR:DATA ""test.dat""") 'Saves data
ioObj.WriteString("*OPC?") : s = ioObj.ReadString()

ioObj.WriteString(":MMEM:STOR:STAT ""test.sta""") 'Saves status
ioObj.WriteString("*OPC?") : s = ioObj.ReadString()

ioObj.WriteString(":MMEM:LOAD:STAT ""test.sta""") 'Loads status
```


Setting the Display Format of Measured Value

Display format of measured value to be displayed on Meter view is set by the :DISP:VIEW:SING:FORM command.

Example `ioObj.WriteString(":DISP:VIEW:SING:FORM EXP")` 'Exponential format

Controlling the Measurement Function

This section describes how to control the measurement function of Keysight B2980.

- “Enabling the Current/Charge Measurement”
- “Setting the Measurement Mode”
- “Performing Spot Measurement”
- “Setting the Measurement Speed”
- “Setting the Measurement Range”
- “Setting the Measurement Trigger”
- “Setting the Resistance Measurement Result to Absolute Format”
- “Setting the Attenuation Level for Voltage Input”
- “Setting the Measurement Wait Time”
- “Performing Sweep Measurement”
- “Stopping Measurement”

Enabling the Current/Charge Measurement

The current/charge measurement is enabled by the `:INP ON` command.

Example `ioObj.WriteString(":INP ON")`

Setting the Measurement Mode

Measurement mode is set by the `:SENS:FUNC` commands.

Example `ioObj.WriteString(":SENS:FUNC:ON ""VOLT"", ""CURR"", ""RES"")`
`ioObj.WriteString(":SENS:FUNC:ON ""VOLT"", ""CHAR"")`

Performing Spot Measurement

Spot measurement is performed by the `:MEAS:<CURR|CHAR|VOLT|RES>?` command or the `:MEAS?` command. See [Figure 1-3](#) for the spot measurement.

Example `ioObj.WriteString(":MEAS:RES?")`
`ioObj.WriteString(":FORM:ELEM:SENS RES,STAT")`
`ioObj.WriteString(":MEAS?")`

NOTE

The `:FORM:ELEM:SENS` command is used to specify the data to obtain.

Setting the Measurement Speed

Measurement speed is set by the `:SENS:<CURR|CHAR|VOLT|RES>:APER` or `:SENS:<CURR|CHAR|VOLT|RES>:NPLC` command.

Example `ioObj.WriteString(":SENS:CURR:APER:AUTO ON")` 'Automatic
`ioObj.WriteString(":SENS:CURR:APER 1E-4")` '0.1 ms
`ioObj.WriteString(":SENS:CURR:NPLC 1")` '1 Power Line Cycle

Setting the Measurement Range

Measurement range is set by the `:SENS:<CURR|CHAR|VOLT|RES>:RANG` command. And the auto range operation is enabled/disabled by the `:SENS:<CURR|CHAR|VOLT|RES>:RANG:AUTO` command. The lower limit for the auto range operation is set by the `:SENS:<CURR|VOLT|RES>:RANG:AUTO:LLIM` command. The upper limit for the auto range operation is set by the `:SENS:<CURR|VOLT|RES>:RANG:AUTO:ULIM` command. For charge measurement, the auto range operation is set by the `:SENS:CHAR:RANG:AUTO:GRO` command.

Example `ioObj.WriteString(":SENS:CURR:RANG:AUTO OFF")`
`ioObj.WriteString(":SENS:CURR:RANG 1E-12")` '1 pA fixed

`ioObj.WriteString(":SENS:RES:RANG:AUTO ON")`
`ioObj.WriteString(":SENS:RES:RANG:AUTO:LLIM 1E+9")` '1G ohm limit
`ioObj.WriteString(":SENS:RES:RANG:AUTO:ULIM 1E+12")` '1T ohm limit

Setting the Measurement Trigger

Measurement trigger is simply set by the `:TRIG<:ACQ | [:ALL]>:SOUR`, `:TRIG<:ACQ | [:ALL]>:TIM`, `:TRIG<:ACQ | [:ALL]>:COUN`, and `:TRIG<:ACQ | [:ALL]>:DEL` commands. See [Figures 1-1](#), [1-2](#), and [1-3](#).

```

Example   ioObj.WriteString(":TRIG:SOUR TIM")
             ioObj.WriteString(":TRIG:TIM 4E-3")           'Interval 4 ms
             ioObj.WriteString(":TRIG:COUN 11")            '11 points
             ioObj.WriteString(":TRIG:ACQ:DEL 2E-3")       'Meas delay 2 ms
    
```

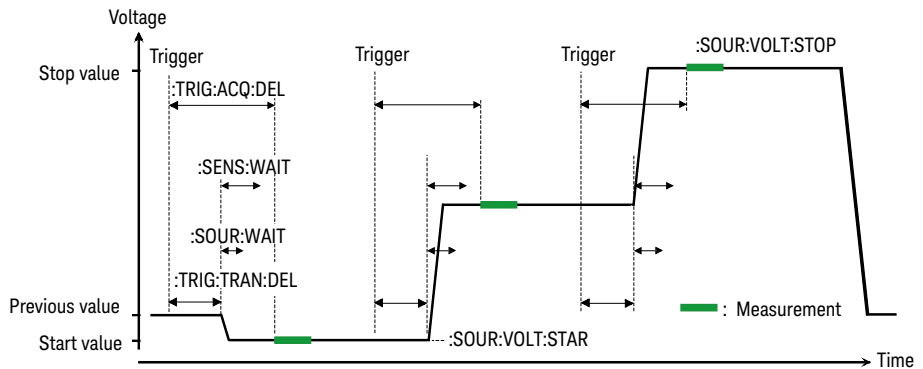
NOTE

If you want to use arm trigger, use the :ARM<:ACQ | [:ALL]>:SOUR, :ARM<:ACQ | [:ALL]>:TIM, :ARM<:ACQ | [:ALL]>:COUN, and :ARM<:ACQ | [:ALL]>:DEL commands. For more details, see *SCPI Command Reference*.

Figure 1-1

To Perform Staircase Sweep Output and Measurement

Staircase sweep source :SOUR:FUNC DC, :SOUR:VOLT:MODE SWE



Setting the Resistance Measurement Result to Absolute Format

Resistance measurement result is set to the absolute format by the :SENG:RES:AVAL command.

```

Example   ioObj.WriteString(":SENG:RES:AVAL ON")
             ioObj.WriteString(":SENG:RES:AVAL OFF")
    
```

Setting the Attenuation Level for Voltage Input

Attenuation level for the voltage input is set by the :SENG:VOLT:ATT command.

When performing voltage measurement with the Keysight N1413A/N1414A adapter, you can set the proper attenuation level for the adapter using this command, in order to return actual voltage value.

Example `ioObj.WriteString(":SENC:VOLT:ATT 100") 'attenuation = x100`
`ioObj.WriteString(":SENC:VOLT:ATT 1") 'attenuation = x1`

Setting the Measurement Wait Time

Measurement wait time is set by the :SENS:WAIT commands. See [Figures 1-1 and 1-3](#) for the wait time.

Example `ioObj.WriteString(":SENS:WAIT OFF") 'Wait = 0 s`
`ioObj.WriteString(":SENS:WAIT ON")`
`ioObj.WriteString(":SENS:WAIT:AUTO OFF")`
`ioObj.WriteString(":SENS:WAIT:OFFS 10E-3") 'Wait = 10 ms`

`ioObj.WriteString(":SENS:WAIT ON")`
`ioObj.WriteString(":SENS:WAIT:AUTO ON")`
`ioObj.WriteString(":SENS:WAIT:OFFS 10E-3")`
`ioObj.WriteString(":SENS:WAIT:GAIN 1")'Wait = 10 ms + initial wait`

Performing Sweep Measurement

Staircase sweep measurement is performed as shown below.

1. Set the staircase sweep source and the required source functions. For details, see [“Controlling the Source Output” on page 23](#).
2. Set the required measurement functions. For details, see previous topics in this section.
3. Set the trigger condition. See [“Setting the Source Output Trigger” on page 26](#) and [“Setting the Measurement Trigger” on page 19](#).
4. Enable the channel. See [“Enabling the Current/Charge Measurement” on page 18](#).

The channel starts output set by the :SOUR:VOLT command.

5. Execute the :INIT command to start measurement.

For the programming example, see [“Staircase Sweep Measurements” on page 44](#).

NOTE

To specify the data to obtain, use the :FORM:ELEM:SENS command for the measurement data or the :FORM:ELEM:CALC command for the calculation data.

Stopping Measurement

Measurement is stopped by the :ABOR command.

Example `ioObj.WriteString(":ABOR")`

Controlling the Source Output

This section describes how to control the source output of Keysight B2985B and B2987B.

- “Enabling the Source Output”
- “Applying the DC Voltage”
- “Stopping the Source Output”
- “Setting the Output Range”
- “Setting the Sweep Operation”
- “Setting the Sweep Output”
- “Setting the Ranging Mode of the Sweep Source”
- “Setting the List Sweep Output”
- “Setting the Source Output Trigger”
- “Setting the Low Terminal State”
- “Specifying the Output-Off Status”
- “Setting the Source Wait Time”

NOTE

The string `:SOUR` in the command string described in this manual can be omitted. For example, `:SOUR:VOLT` can be `:VOLT`.

Enabling the Source Output

Source output is enabled by the `:OUTP ON` command.

Example `ioObj.WriteString(":OUTP ON")`

Applying the DC Voltage

DC voltage is immediately applied by the `:SOUR:VOLT` command during the source output is enabled.

If you want to control the DC voltage output timing using a trigger, use the :SOUR:VOLT:TRIG command. See [Figure 1-3](#).

Example `ioObj.WriteString(":SOUR:VOLT 20")` 'Outputs 20 V immediately
`ioObj.WriteString(":SOUR:FUNC:TRIG:CONT 1")`
`ioObj.WriteString(":SOUR:VOLT:MODE FIX")`
`ioObj.WriteString(":SOUR:VOLT:TRIG -5")` 'Outputs -5 V by a trigger

Stopping the Source Output

Source output is stopped and disabled by the :OUTP OFF command.

Example `ioObj.WriteString(":OUTP OFF")`

Setting the Output Range

Output range is set by the :SOUR:VOLT:RANG command.

Example `ioObj.WriteString(":SOUR:VOLT:RANG -1000")` '-1000 V range

Setting the Sweep Operation

For the variety of sweep output operation, see [Figure 1-2](#).

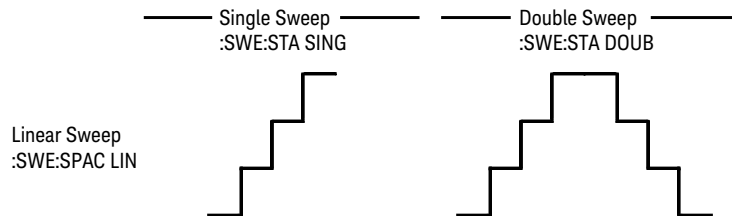
Sweep direction, upward or downward is set by the :SOUR:SWE:DIR command.

Sweep mode, single or double is set by the :SOUR:SWE:STA command.

Example `ioObj.WriteString(":SOUR:SWE:DIR DOWN")`
`ioObj.WriteString(":SOUR:SWE:STA DOUB")`

Figure 1-2

Variety of Sweep Outputs



Setting the Sweep Output

Staircase sweep output is set by the `:SOUR:VOLT:MODE SWE` command, the `:SOUR:VOLT:<POIN|STEP>` or `:SOUR:SWE:POIN` command, and the `:SOUR:VOLT:<STAR|STOP>` or `:SOUR:VOLT:<CENT|SPAN>` command. See [Figure 1-1](#).

Example

```
ioObj.WriteString(":SOUR:VOLT:MODE SWE")
ioObj.WriteString(":SOUR:VOLT:STAR 0") 'Start 0 V
ioObj.WriteString(":SOUR:VOLT:STOP 1") 'Stop 1 V
ioObj.WriteString(":SOUR:VOLT:POIN 11") '11 points
```

NOTE

Outputting the sweep voltage

Execute the `:OUTP ON` command to start outputting the value set by the `:SOUR:VOLT` command.

Execute the `:INIT` to perform the specified sweep output and measurement.

Setting the Ranging Mode of the Sweep Source

Ranging mode of sweep source is set by the `:SOUR:SWE:RANG` command.

Example

```
ioObj.WriteString(":SOUR:SWE:RANG BEST") 'Covers all LIN steps
ioObj.WriteString(":SOUR:SWE:RANG FIX") 'Not change
```

Setting the List Sweep Output

List sweep output is set by the `:SOUR:VOLT:MODE LIST` command and the `:SOUR:LIST:VOLT` command.

Example

```
ioObj.WriteString(":SOUR:VOLT:MODE LIST")
ioObj.WriteString(":SOUR:LIST:VOLT 0,2,4,6,8,10,0")
```

NOTE

Outputting the list sweep voltage/current

Execute the `:OUTP ON` command to start outputting the value set by the `:SOUR:VOLT` command.

Execute the `:INIT` to perform the specified list sweep output and measurement.

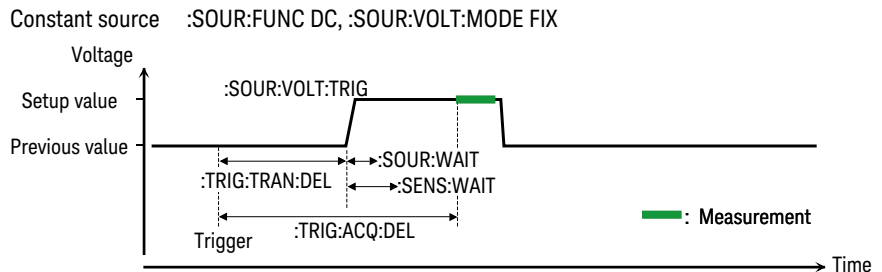
Setting the Source Output Trigger

Source output trigger is simply set by the `:TRIG<:TRAN | [:ALL]>:SOUR`, `:TRIG<:TRAN | [:ALL]>:TIM`, `:TRIG<:TRAN | [:ALL]>:COUN`, and `:TRIG<:TRAN | [:ALL]>:DEL` commands. See [Figure 1-3](#).

Example

```
ioObj.WriteString(":TRIG:SOUR TIM")
ioObj.WriteString(":TRIG:TIM 4E-3")      'Interval 4 ms
ioObj.WriteString(":TRIG:COUN 11")      '11 points
ioObj.WriteString(":TRIG:TRAN:DEL 1E-3") 'Source delay 1 ms
```

Figure 1-3 To Perform DC and Spot Measurement



NOTE

If you want to use arm trigger, use the `:ARM<:TRAN | [:ALL]>:SOUR`, `:ARM<:TRAN | [:ALL]>:TIM`, `:ARM<:TRAN | [:ALL]>:COUN`, and `:ARM<:TRAN | [:ALL]>:DEL` commands. For more details, see *SCPI Command Reference*.

Setting the Low Terminal State

Low terminal state, ground or floating is set by the `:OUTP:LOW` command.

Example

```
ioObj.WriteString(":OUTP OFF")
ioObj.WriteString(":OUTP:LOW COMM")      'Common
ioObj.WriteString(":OUTP ON")
```

Specifying the Output-Off Status

Output-off status is set by the `:OUTP:OFF:MODE` command.

Example

```
ioObj.WriteString(":OUTP:OFF:MODE ZERO") 'Zero volt
ioObj.WriteString(":OUTP:OFF:MODE HIZ") 'High impedance
```

```
ioObj.WriteString(":OUTP:OFF:MODE NORM") 'Normal
```

Setting the Source Wait Time

Source wait time is set by the :SOUR:WAIT commands. See [Figures 1-1](#) and [1-3](#) for the wait time.

Example

```
ioObj.WriteString(":SOUR:WAIT OFF") 'Wait = 0 s

ioObj.WriteString(":SOUR:WAIT ON")
ioObj.WriteString(":SOUR:WAIT:AUTO OFF")
ioObj.WriteString(":SOUR:WAIT:OFFS 10E-3") 'Wait = 10 ms

ioObj.WriteString(":SOUR:WAIT ON")
ioObj.WriteString(":SOUR:WAIT:AUTO ON")
ioObj.WriteString(":SOUR:WAIT:OFFS 10E-3")
ioObj.WriteString(":SOUR:WAIT:GAIN 1")'Wait = 10 ms + initial wait
```

Using the Math Function

This section describes how to use the math function.

- [“Defining a Math Expression”](#)
- [“Deleting an User Defined Math Expression”](#)
- [“Enabling or Disabling the Math Function”](#)
- [“Reading Math Result Data”](#)

Defining a Math Expression

Math expression is defined by the `:CALC:MATH[:EXPR]` commands.

Example

```
ioObj.WriteString(":CALC:MATH:NAME ""DiffV""")
ioObj.WriteString(":CALC:MATH:DEF (SOUR-VOLT)")
ioObj.WriteString(":CALC:MATH:UNIT ""V""")
```

Deleting an User Defined Math Expression

Math expression is deleted by the `:CALC:MATH[:EXPR]:DEL` commands. The commands do not delete the predefined math expression.

Example

```
ioObj.WriteString(":CALC:MATH:DEL ""DiffV""")      'Deletes DiffV
ioObj.WriteString(":CALC:MATH:DEL:ALL")           'Deletes all
```

Enabling or Disabling the Math Function

Math function is set by the `:CALC:MATH:STAT` command.

Example

```
ioObj.WriteString(":CALC:MATH:STAT ON")
```

Reading Math Result Data

Math result data is read by the `:CALC:MATH:DATA?` commands.

Example

```
ioObj.WriteString(":CALC:MATH:DATA:LAT?")        'Latest data
ioObj.WriteString(":CALC:MATH:DATA?")           'All data
```

NOTE

To specify the data to obtain, use the :FORM:ELEM:CALC command.

Performing the Limit Test

Limit test is performed as shown below.

1. Set the required source condition. For details, see [“Controlling the Source Output” on page 23](#).
2. Set the required measurement condition. For details, see [“Controlling the Measurement Function” on page 18](#).
3. Set the composite limit test. See [“Setting the Composite Limit Test” on page 31](#).
4. Set the individual limit tests. See [“Setting Individual Limit Tests” on page 31](#).
5. Set the trigger condition. See [“Setting the Source Output Trigger” on page 26](#) and [“Setting the Measurement Trigger” on page 19](#).
6. Enable the channel. See [“Enabling the Current/Charge Measurement” on page 18](#).
7. Execute the :INIT command to start limit test.

Note that the DC source channel applies the value set by the :SOUR:VOLT:TRIG command when it is triggered, even if the channel is applying the value set by the :SOUR:VOLT command.

8. Read limit test result. See [“Reading Limit Test Result” on page 31](#).

For the programming example, see [“Pass/Fail Judgment and Math Function” on page 56](#).

Setting the Composite Limit Test

Composite limit test is set by the :CALC:CLIM commands and enabled by the :CALC:CLIM:STAT ON command. To perform a limit test, at least one individual limit test must be set and enabled.

Example

```
ioObj.WriteString(":CALC:CLIM:CLE")      'Clears result now
ioObj.WriteString(":CALC:CLIM:MODE GRAD") 'Sets grading mode
ioObj.WriteString(":CALC:CLIM:UPD END")  'Sends result at the end
ioObj.WriteString(":CALC:CLIM:STAT ON")  'Composite limit test on
```

NOTE

If you want to use the GPIO port for sending a pass/fail bit pattern, use the :CALC:DIG commands to specify the output port. See *SCPI command reference*.

If you want to use the null offset function for cancelling the offset value from the measurement data automatically, use the :CALC:OFFS commands. See *SCPI command reference*.

Setting Individual Limit Tests

Individual limit test is set by the :CALC:LIM commands and the :CALC:FEED command. And each limit test is enabled by the :CALC:LIM:STAT ON command.

Example

```
ioObj.WriteString(":CALC:FEED CURR")    'Specifies feed data
ioObj.WriteString(":CALC:LIM:STAT ON")  'Limit test on
ioObj.WriteString(":CALC:LIM:FUNC LIM") 'Selects limit test
ioObj.WriteString(":CALC:LIM:UPP +3.5E-2") 'Sets upper limit
ioObj.WriteString(":CALC:LIM:LOW +3E-2") 'Sets lower limit
```

Reading Limit Test Result

Limit test result is read by the :CALC:DATA? commands.

Fail status of the individual limit test is read by the :CALC:LIM:FAIL? command.

Example

```
ioObj.WriteString(":CALC:DATA:LAT?")    'Latest data
ioObj.WriteString(":CALC:DATA?")       'All data
ioObj.WriteString(":CALC:LIM:FAIL?")    '1:failed, 0:passed
```

NOTE

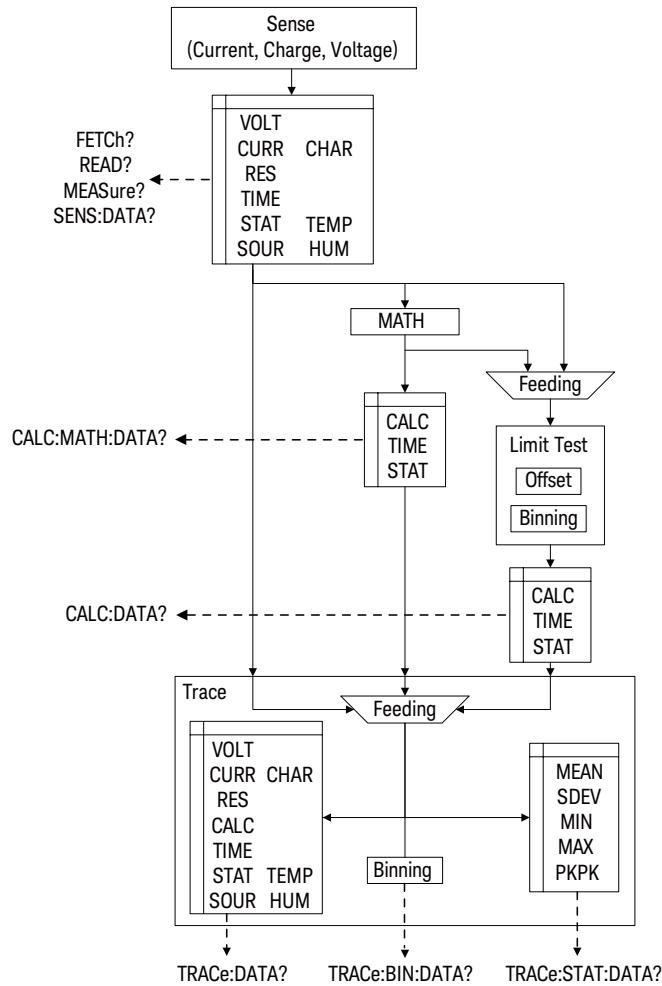
To specify the data to obtain, use the :FORM:ELEM:CALC command.

Using the Trace Buffer

This section describes how to use the trace buffer.

- “Setting the Trace Buffer”
- “Reading the Trace Data”

Figure 1-4 Trace Buffer and Data Flow



Setting the Trace Buffer

Trace buffer is set by the :TRAC commands.

Example

```
ioObj.WriteString(":TRAC:CLE")           'Clears trace buffer
ioObj.WriteString(":TRAC:POIN 1000")     'Sets buffer size
ioObj.WriteString(":TRAC:FEED SENS")     'Specifies data to feed
ioObj.WriteString(":TRAC:FEED:CONT NEXT") 'Enables write buffer
ioObj.WriteString(":TRAC:TST:FORM DELT")
```

NOTE

The :TRAC:TST:FORM command is used to specify the timestamp data format, delta (DELT) or absolute (ABS).

To specify the data to collect, use the :FORM:ELEM:SENS command for the measurement data or the :FORM:ELEM:CALC command for the calculation data.

Reading the Trace Data

All data in the trace buffer is read by the :TRAC:DATA? command.

Statistical data of the data stored in the trace buffer is read by the :TRAC:STAT:DATA? command. Previously, the type of the statistical data to read must be selected by the :TRAC:STAT:FORM command.

The :TRAC:STAT:FORM command selects one from the following statistical data.

- MEAN: Mean value
- SDEV: Standard deviation
- PKPK: Peak to peak value
- MIN: Minimum value
- MAX: Maximum value

Example

```
ioObj.WriteString(":TRAC:DATA?")           'Reads all data
ioObj.WriteString(":TRAC:STAT:FORM MEAN")
ioObj.WriteString(":TRAC:STAT:DATA?")     'Reads statistical data
```

Using Program Memory

This section describes how to use program memory.

- “Defining a Memory Program”
- “Deleting a Program”
- “Controlling the Program Operation”

Defining a Memory Program

Memory program is defined by the :PROG:NAME and :PROG:DEF commands.

Example

```
ioObj.WriteString(":PROG:NAME ""sample""")
ioObj.WriteString(":PROG:DEF #212:INP:STAT ON") 'Definite length

ioObj.WriteString(":PROG:NAME ""sample1""")
ioObj.WriteString(":PROG:DEF #0:INP:STAT ON") 'Indefinite length
```

Deleting a Program

Memory program is deleted by the :PROG:DEL commands.

Example

```
ioObj.WriteString(":PROG:DEL:ALL") 'Deletes all

ioObj.WriteString(":PROG:NAME ""sample1""")
ioObj.WriteString(":PROG:DEL") 'Deletes sample1
```

Controlling the Program Operation

Memory program is controlled by the :PROG:NAME command and the :PROG:EXEC or :PROG:STAT command. The :PROG[:SEL]:STAT command needs a parameter used to control the operation or change the status. The parameter must be RUN to change the status to running, PAUS to change it to paused, CONT to change it from paused to running, STOP to change it to stopped, or STEP to perform step execution.

Example

```
ioObj.WriteString(":PROG:NAME ""sample""")
ioObj.WriteString(":PROG:EXEC")
ioObj.WriteString("*OPC?") : s = ioObj.ReadString()
```

```
ioObj.WriteString(":PROG:NAME ""sample""")  
ioObj.WriteString(":PROG:STAT RUN")  
ioObj.WriteString("*OPC?") : s = ioObj.ReadString()  
ioObj.WriteString(":PROG:STAT STOP")
```

Controlling Keysight B2980 Using Program Memory

2 Programming Examples

Preparations	38
Spot Measurements	42
Staircase Sweep Measurements	44
List Sweep Measurements	48
Sampling Measurements	52
Pass/Fail Judgment and Math Function	56
Using Program Memory	62
Reading Binary Data	65
Reading Data During Measurement	67

This chapter explains the programming examples.

NOTE

About Example Program Code

Example programs described in this section have been written in the Microsoft Visual Basic .NET language. The examples are provided as a subprogram that can be run with the project template shown in [Table 2-1](#). To run the program, insert the example subprogram or your subprogram instead of the B2900control subprogram in the template.

NOTE

To Start Program

If you create the measurement program by using the example code shown in [Table 2-1](#), the program can be run by clicking the Run button on the Visual Basic main window.

Preparations

This section provides the basic information for programming of the automatic measurement using the Keysight B2980, Keysight IO Libraries, and Microsoft Visual Basic .NET.

- “To Create Your Project Template”
- “To Create Measurement Program”

NOTE

To execute the example programs in this chapter, you need to install Keysight GPIB interface, Keysight IO Libraries Suite, and Microsoft Visual Basic .NET on your computer.

To Create Your Project Template

Before starting programming, create your project template, and keep it as your reference. It will remove the conventional task in the future programming. This section explains how to create a project template.

- Step 1. Connect Keysight B2980 (e.g. GPIB address 23) to the computer via GPIB.
- Step 2. Launch Visual Basic .NET and create a new project. The project type should be Console Application to simplify the programming.
- Step 3. Add the following references to the project.
 - VISA COM 3.0 (or later) Type Library
 - Ivi.Visa.Interop
 - System.IO
- Step 4. Open a module (e.g. Module1.vb) in the project. And enter a program code as template. See [Table 2-1](#) for example.
- Step 5. Save the project as your template (e.g. \test\my_temp).

To Create Measurement Program

Create the measurement program as shown below. The following procedure needs your project template. If the procedure does not fit your programming environment, arrange it to suit your environment.

Step 1. Plan the automatic measurements. Then decide the following items:

- Measurement devices
Discrete, packaged, on-wafer, and so on.
- Parameters/characteristics to be measured
 h_{FE} , V_{th} , sheet resistance, and so on.
- Measurement method
Spot measurement, staircase sweep measurement, and so on.

Step 2. Make a copy of your project template (e.g. `\test\my_temp` to `\test\dev_a\my_temp`).

Step 3. Rename the copy (e.g. `\test\dev_a\my_temp` to `\test\dev_a\spot1V`).

Step 4. Launch Visual Basic .NET.

Step 5. Open the project (e.g. `\test\dev_a\spot1V`).

Step 6. Open the module that contains the template code as shown in [Table 2-1](#). On the code window, complete the B2900control subprogram.

Step 7. Insert the code to display, store, or calculate data into the subprogram.

Step 8. Save the project (e.g. `\test\dev_a\spot1V`).

Table 2-1 Example Template Program Code

```

Module Module1
    Sub Main() '1
        Dim rm As Ivi.Visa.Interop.ResourceManager
        Dim ioObj As Ivi.Visa.Interop.FormattedIO488
        Dim ifAddress As String = "23"
        Dim filename As String = ""
        Dim filedata As String = "Result: "
        Dim s As String = ""

        Try '9
            rm = New Ivi.Visa.Interop.ResourceManager
            ioObj = New Ivi.Visa.Interop.FormattedIO488
            Try
                ioObj.IO = rm.Open("GPIB0::" + ifAddress + "::INSTR")
                ioObj.IO.Timeout = 60000
                ioObj.IO.TerminationCharacter = 10
                ioObj.IO.TerminationCharacterEnabled = True
            Catch ex As Exception
                Console.WriteLine("An error occurred: " + ex.Message)
            End Try

            B2900control(ioObj, s, filename) '21
            Console.Write(filedata + s)
            MsgBox("Click OK to close the console window.", vbOKOnly, "")

            FileOpen(1, filename, OpenMode.Output, OpenAccess.Write,
            OpenShare.LockReadWrite) '25
            Print(1, filedata + s)
            FileClose(1)

            ioObj.IO.Close() '29
            System.Runtime.InteropServices.ReleaseComObject(ioObj)
            System.Runtime.InteropServices.ReleaseComObject(rm)
            Catch ex As Exception
                Console.WriteLine("An error occurred: " + ex.Message)
            End Try
        End Sub

        Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
        ByRef filename As String) '37
            filename = ".././../execution result/resultdata1.txt"
        End Sub
    End Module

```

Line	Description
1 to 8	Beginning of the Main subprogram. And defines the variables used in this program.
9 to 20	Establishes the connection with the B2980 specified by the GPIB address ifAddress = 23 on the interface GPIB0.


```

Module Module1

Sub Main() '1
    Dim rm As Ivi.Visa.Interop.ResourceManager
    Dim ioObj As Ivi.Visa.Interop.FormattedIO488
    Dim ifAddress As String = "23"
    Dim filename As String = ""
    Dim filedata As String = "Result: "
    Dim s As String = ""

    Try '9
        rm = New Ivi.Visa.Interop.ResourceManager
        ioObj = New Ivi.Visa.Interop.FormattedIO488
        Try
            ioObj.IO = rm.Open("GPIB0::" + ifAddress + "::INSTR")
            ioObj.IO.Timeout = 60000
            ioObj.IO.TerminationCharacter = 10
            ioObj.IO.TerminationCharacterEnabled = True
        Catch ex As Exception
            Console.WriteLine("An error occurred: " + ex.Message)
        End Try

        B2900control(ioObj, s, filename) '21
        Console.Write(filedata + s)
        MsgBox("Click OK to close the console window.", vbOKOnly, "")

        FileOpen(1, filename, OpenMode.Output, OpenAccess.Write, '25
        OpenShare.LockReadWrite)
        Print(1, filedata + s)
        FileClose(1)

        ioObj.IO.Close() '29
        System.Runtime.InteropServices.Marshal.ReleaseComObject(ioObj)
        System.Runtime.InteropServices.Marshal.ReleaseComObject(rm)
        Catch ex As Exception
            Console.WriteLine("An error occurred: " + ex.Message)
        End Try
    End Sub

    Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, '37
    ByRef filename As String)
        filename = "../../execution result/resultdata1.txt"
    End Sub

End Module

```

Line	Description
21 to 23	Calls the B2900control subprogram. And displays the result data in a console window.
25 to 27	Saves the result data to a file specified by filename.
29 to 35	Breaks the connection with the B2980.
37 to 39	B2900control subprogram. Measurement program code should be entered here.

Spot Measurements

A program example of spot measurements is shown in [Table 2-3](#). This example is used to apply voltage and measure current.

Table 2-2 Spot Measurement Commands

Function	Command
Selects source function	:SOUR:FUNC:MODE <i>VOLT</i>
Sets source output range	:SOUR:VOLT:RANG <i>value</i>
Sets source output value	:SOUR:VOLT <i>value</i>
Sets measurement function	[:SENS:]FUNC " <i>func</i> ", " <i>func</i> ", " <i>func</i> "]
Sets measurement range	[:SENS:] <i>func</i> :RANG:AUTO <ON OFF> [:SENS:] <i>func</i> :RANG <i>value</i>
Sets aperture time in seconds or by using NPLC value	[:SENS:] <i>func</i> :APER <i>time</i> [:SENS:] <i>func</i> :NPLC <i>value</i>
Enables/disables Ammeter input	:INP <ON OFF>
Enables/disables Voltage Source output	:OUTP <ON OFF>
Initiates measurement and reads result data (latest data)	:MEAS? :MEAS: <i>func</i> ?

Measurement Result Example

Result: +3.034160E-02

Table 2-3 Spot Measurement Example

```

Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef
filename As String)
    filename = "../.././execution result/FixedDc1.txt" '2

    ioObj.WriteString("*RST") ' Reset

    Try ' Set voltage output to 0.1 V '6
        ioObj.WriteString(":sour:func:mode volt")
        ioObj.WriteString(":sour:volt 0.1")

        ' Set auto-range current measurement
        ioObj.WriteString(":sens:func ""curr""") '11
        ioObj.WriteString(":sens:curr:rang:auto on")
        ioObj.WriteString(":sens:curr:nplc:auto off")
        ioObj.WriteString(":sens:curr:nplc 0.1")

    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try

    ' Turn on output switch
    ioObj.WriteString(":outp on") '21

    ' Turn on input switch
    ioObj.WriteString(":inp on") '24

    Try ' Initiate measurement and retrieve measurement result '26
        ioObj.WriteString(":meas:curr?")
        s = ioObj.ReadString()

    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
End Sub

```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
6 to 8	Sets the voltage source function. And sets the source value to 0.1 V.
11 to 14	Sets the current measurement function and the auto range measurement. And sets the aperture time to 0.1 PLC.
21	Enables the output and starts source output.
24	Enables the input.
26 to 28	Reads the measurement result data.

Staircase Sweep Measurements

A program example of staircase sweep measurements is shown in [Table 2-5](#). This example is used to apply sweep voltage and measure current at each sweep step.

Table 2-4 Staircase Sweep Measurement Commands

Function	Command
Selects source function	:SOUR:FUNC:MODE <i>VOLT</i>
Sets sweep output	:SOUR:VOLT:MODE SWE
Sets output range when starting sweep	:SOUR:VOLT:RANG <i>value</i>
Sets source output value	:SOUR:VOLT <i>value</i>
Sets sweep start or stop value	:SOUR:VOLT:<STAR STOP> <i>value</i>
Sets sweep center or span value	:SOUR:VOLT:<CENT SPAN> <i>value</i>
Sets sweep step value	:SOUR:VOLT:STEP <i>value</i>
Sets number of sweep steps	:SOUR:VOLT:POIN <i>value</i> :SOUR:SWE:POIN <i>value</i>
Selects sweep source ranging mode	:SOUR:SWE:RANG <BEST FIX>
Selects sweep direction	:SOUR:SWE:DIR <UP DOWN>
Selects sweep single or double	:SOUR:SWE:STA <SING DOUB>
Sets measurement function	[:SENS:]FUNC " <i>func</i> ", " <i>func</i> ", " <i>func</i> "]
Sets measurement range	[:SENS:] <i>func</i> :RANG:AUTO <ON OFF> [:SENS:] <i>func</i> :RANG <i>value</i>
Sets aperture time in seconds or by using NPLC value	[:SENS:] <i>func</i> :APER <i>time</i> [:SENS:] <i>func</i> :NPLC <i>value</i>
Selects trigger source	:TRIG<:ACQ :TRAN [:ALL]>:SOUR <i>source</i>
Sets interval of timer trigger	:TRIG<:ACQ :TRAN [:ALL]>:TIM <i>time</i>

Function	Command
Sets trigger count	:TRIG<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables Ammeter input	:INP <ON OFF>
Enables/disables Voltage Source output	:OUTP <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]>
Reads result data (array data)	:FETC:ARR? :FETC:ARR: <i>type</i> ?

func is VOLT for voltage measurement, CURR for current measurement, CHAR for charge measurement, or RES for resistance measurement.

source is AINT for the automatic trigger, BUS for the remote interface trigger command, TIM for the internal timer, INT n for a signal from the internal bus ($n = 1$ or 2), EXT m for a signal from the GPIO pin m ($m = 1$ to 7), or LAN for the LXI trigger.

type is VOLT for voltage data, CURR for current data, CHAR for charge data, RES for resistance data, SOUR for source output data, STAT for status data, or TIME for time data.

**Measurement
Result Example**

Result: +1.842730E-07,+6.775910E-03,+1.519010E-02,+2.277660E-02,+3.036230E-02

Table 2-5 Staircase Sweep Measurement Example

<pre> Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "../../././execution result/StaircaseSweep1.txt" '2 ioObj.WriteString("*RST") ' Reset '4 Try ' Set voltage output from 0 V to 0.1 V, 5 steps ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode swe") ioObj.WriteString(":sour:volt:star 0") ioObj.WriteString(":sour:volt:stop 0.1") ioObj.WriteString(":sour:volt:poin 5") '7 ' Set 2nA limited auto ranging current measurement ioObj.WriteString(":sens:func ""curr""") ioObj.WriteString(":sens:curr:rang:auto:llim 2e-9") '14 ' Set measurement speed to 0.1 PLC ioObj.WriteString(":sens:curr:nplc:auto off") ioObj.WriteString(":sens:curr:nplc 0.1") '18 ' Generate 5 triggers by automatic internal algorithm ioObj.WriteString(":trig:sour aint") ioObj.WriteString(":trig:coun 5") '22 Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try ' Turn on output switch ioObj.WriteString(":outp on") '30 </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
7 to 11	Sets the voltage sweep output function. And sets the sweep output from 0 to 0.1 V in 0.02 V step (5 points).
14 to 15	Sets the current auto ranging to 2nA limited auto ranging.
18 to 19	Sets the aperture time to 0.1 PLC
22 to 23	Sets the trigger source to AINT (automatic trigger). And sets the trigger count to 5 to perform a 5-step staircase sweep measurement.
30	Enables the output and starts source output (0 V with the default setting).

```

    ' Turn on input switch
    ioObj.WriteString(":inp on")                                     ' 33

    ' Initiate transition and acquire
    ioObj.WriteString(":init")                                     ' 36

    Try ' Retrieve measurement result
        ioObj.WriteString(":fetc:arr:curr?")                       ' 39
        s = ioObj.ReadString()

    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
End Sub

```

Line	Description
33	Enabled the input for current measurement.
36	Starts staircase sweep measurement.
39 to 40	Reads the measurement result data.

List Sweep Measurements

A program example of list sweep measurements is shown in [Table 2-7](#). This example is used to apply sweep voltage and measure current at each sweep step.

Table 2-6 List Sweep Measurements Commands

Function	Command
Selects source function	:SOUR:FUNC:MODE <i>VOLT</i>
Sets list sweep output	:SOUR:VOLT:MODE LIST
Sets output range when starting sweep	:SOUR:VOLT:RANG <i>value</i>
Sets source output value	:SOUR:VOLT <i>value</i>
Sets list sweep output values	:SOUR:LIST:VOLT <i>values</i>
Adds list sweep output values to the end of the present setting	:SOUR:LIST:VOLT:APP <i>values</i>
Specifies the list sweep start point	:SOUR:LIST:VOLT:STAR <i>start_index</i>
Asks the number of sweep points	:SOUR:LIST:VOLT:POIN?
Sets measurement function	[:SENS:]FUNC " <i>func</i> "[, " <i>func</i> "[, " <i>func</i> "]]
Sets measurement range	[:SENS:] <i>func</i> :RANG:AUTO <ON OFF> [:SENS:] <i>func</i> :RANG <i>value</i>
Sets aperture time in seconds or by using NPLC value	[:SENS:] <i>func</i> :APER <i>time</i> [:SENS:] <i>func</i> :NPLC <i>value</i>
Selects trigger source	:TRIG<:ACQ :TRAN [:ALL]>:SOUR <i>source</i>
Sets interval of timer trigger	:TRIG<:ACQ :TRAN [:ALL]>:TIM <i>time</i>
Sets trigger count	:TRIG<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables Ammeter input	:INP <ON OFF>

Function	Command
Enables/disables Voltage Source output	:OUTP <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]>
Reads result data (array data)	:FETC:ARR? :FETC:ARR: <i>type</i> ?

func is VOLT for voltage measurement, CURR for current measurement, CHAR for charge measurement, or RES for resistance measurement.

source is AINT for the automatic trigger, BUS for the remote interface trigger command, TIM for the internal timer, INT n for a signal from the internal bus ($n = 1$ or 2), EXT m for a signal from the GPIO pin m ($m = 1$ to 7), or LAN for the LXI trigger.

type is VOLT for voltage data, CURR for current data, RES for resistance data, SOUR for source output data, STAT for status data, or TIME for time data.

Measurement Result Example

Result: +9.121600E-03,+1.822320E-02,+3.036130E-02

Table 2-7 List Sweep Measurement Example

<pre> Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "../.././././execution result/ListSweep1.txt" '2 ioObj.WriteString("*RST") ' Reset '4 Try ' Set voltage output to 0.03, 0.06, and 0.1 V ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode list") ioObj.WriteString(":sour:list:volt 0.03,0.06,0.1") '7 ' Set 2nA limited auto ranging ioObj.WriteString(":sens:func ""curr""") ioObj.WriteString(":sens:curr:rang:auto:llim 2e-9") '12 ' Set measurement speed to 0.1 PLC ioObj.WriteString(":sens:curr:nplc:auto off") ioObj.WriteString(":sens:curr:nplc 0.1") '16 ' Generate 3 triggers by automatic internal algorithm ioObj.WriteString(":trig:sour aint") ioObj.WriteString(":trig:coun 3") '20 Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
7 to 9	Sets the voltage list sweep output function. And sets the list sweep output 0.03 V, 0.06 V, and 0.1 V (3 points).
12 to 13	Sets the current auto ranging to 2nA limited auto ranging.
16 to 17	Sets the aperture time to 0.1 PLC
20 to 21	Sets the trigger source to AINT (automatic trigger). And sets the trigger count to 3to perform a 3-step list sweep measurement.

```

' Turn on output switch
ioObj.WriteString(":outp on")                                ' 28

' Turn on input switch
ioObj.WriteString(":inp on")                                ' 31

' Initiate transition and acquire
ioObj.WriteString(":init")                                  ' 34

Try ' Retrieve measurement result
    ioObj.WriteString(":fetc:arr:curr?")                    ' 37
    s = ioObj.ReadString()

Catch ex As Exception
    Console.WriteLine("An error occurred: " + ex.Message)
End Try
End Sub

```

Line	Description
28	Enables the channel. And starts source output (0 V with the default setting).
31	Enabled the input for current measurement.
33	Starts list sweep measurement.
37 to 38	Reads the measurement result data.

Sampling Measurements

A program example of sampling measurements is shown in [Table 2-9](#). This example is used to apply constant voltage and measure current six times.

Table 2-8 Sampling Measurements Commands

Function	Command
Selects source function	:SOUR:FUNC:MODE <i>VOLT</i>
Sets output range when starting sweep	:SOUR:VOLT:RANG <i>value</i>
Sets source output value	:SOUR:VOLT <i>value</i>
Sets triggered source output value	:SOUR:VOLT:TRIG <i>value</i>
Sets measurement function	[:SENS:]FUNC " <i>func</i> "[, " <i>func</i> "[, " <i>func</i> "]]
Sets measurement range	[:SENS:] <i>func</i> :RANG:AUTO <ON OFF> [:SENS:] <i>func</i> :RANG <i>value</i>
Sets aperture time in seconds or by using NPLC value	[:SENS:] <i>func</i> :APER <i>time</i> [:SENS:] <i>func</i> :NPLC <i>value</i>
Selects trigger source	:TRIG<:ACQ :TRAN [:ALL]>:SOUR <i>source</i>
Sets interval of timer trigger	:TRIG<:ACQ :TRAN [:ALL]>:TIM <i>time</i>
Sets trigger count	:TRIG<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables Ammeter input	:INP <ON OFF>
Enables/disables Voltage Source output	:OUTP <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]>
Reads result data (array data)	:FETC:ARR? :FETC:ARR: <i>type</i> ?

func is VOLT for voltage measurement, CURR for current measurement, CHAR for charge measurement, or RES for resistance measurement.

source is AINT for the automatic trigger, BUS for the remote interface trigger command, TIM for the internal timer, INT n for a signal from the internal bus ($n = 1$ or 2), EXT m for a signal from the GPIO pin m ($m = 1$ to 7), or LAN for the LXI trigger.

type is VOLT for voltage data, CURR for current data, RES for resistance data, SOUR for source output data, STAT for status data, or TIME for time data.

**Measurement
Result Example**

Result: +3.036250E-02,+3.036240E-02,+3.036190E-02,+3.036210E-02,+3.036300E-02,+3.036270E-02

Table 2-9 Sampling Measurement Example

```

Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef
filename As String)
    filename = "../../././execution result/Sampling1.txt" '2

    ioObj.WriteString("*RST") ' Reset '4

    Try ' Set fixed range current measurement
        ioObj.WriteString(":sens:func ""curr""") '7
        ioObj.WriteString(":sens:curr:rang:auto off")
        ioObj.WriteString(":sens:curr:rang 2e-9")

        ' Set measurement speed to 0.1 PLC
        ioObj.WriteString(":sens:curr:nplc:auto off") '12
        ioObj.WriteString(":sens:curr:nplc 0.1")

        ' Adjust trigger timing parameters
        ioObj.WriteString(":trig:acq:del 2.0e-3") '16

        ' Generate 6 triggers in 4 ms period
        ioObj.WriteString(":trig:sour tim") '19
        ioObj.WriteString(":trig:tim 4e-3")
        ioObj.WriteString(":trig:coun 6")

    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try

    ' Turn on input switch
    ioObj.WriteString(":inp on") '28

    ' Initiate acquire
    ioObj.WriteString(":init:acq") '31

    Try ' Retrieve measurement result
        ioObj.WriteString(":fetc:arr:curr?") '34
        s = ioObj.ReadString()

    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
End Sub

```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
7 to 9	Sets the current measurement function and the fixed range measurement.
12 to 13	Sets the aperture time to 0.1 PLC.
16	Sets the acquire (measurement) delay time.

```

Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef
filename As String)
    filename = ".././../execution result/Sampling1.txt" '2
    ioObj.WriteString("*RST") ' Reset '4
    Try ' Set fixed range current measurement
        ioObj.WriteString(":sens:func ""curr""") '7
        ioObj.WriteString(":sens:curr:rang:auto off")
        ioObj.WriteString(":sens:curr:rang 2e-9")
        ' Set measurement speed to 0.1 PLC
        ioObj.WriteString(":sens:curr:nplc:auto off") '12
        ioObj.WriteString(":sens:curr:nplc 0.1")
        ' Adjust trigger timing parameters
        ioObj.WriteString(":trig:acq:del 2.0e-3") '16
        ' Generate 6 triggers in 4 ms period
        ioObj.WriteString(":trig:sour tim") '19
        ioObj.WriteString(":trig:tim 4e-3")
        ioObj.WriteString(":trig:coun 6")
    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
    ' Turn on input switch
    ioObj.WriteString(":inp on") '28
    ' Initiate acquire
    ioObj.WriteString(":init:acq") '31
    Try ' Retrieve measurement result
        ioObj.WriteString(":fetc:arr:curr?") '34
        s = ioObj.ReadString()
    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
End Sub

```

Line	Description
19 to 21	Sets the timer trigger source. And sets the timer interval to 4 ms, and the trigger count to 6 to perform a 6-point sampling measurement.
28	Enables the Ammeter input.
31	Starts sampling measurement.
34 to 35	Reads the measurement result data.

Pass/Fail Judgment and Math Function

A program example of pass/fail judgements is shown in [Table 2-12](#). This example is used to perform power vs voltage sweep measurement and performs pass/fail judgement. Power is calculated by using the built-in math function.

Table 2-10

Math Function Commands

Function	Command
Specifies math expression	:CALC:MATH:NAME " <i>name</i> "
Defines math expression	:CALC:MATH:DEF <i>definition</i>
Sets unit for the math expression	:CALC:MATH:UNIT
Enables/disables math function	:CALC:MATH:STAT <ON OFF>

Table 2-11

Pass/Fail Judgment Commands

Function	Command
Sets composite limit test mode to grading or sorting	:CALC:CLIM:MODE <GRAD SORT>
Specifies test result output timing for the grading mode, end or immediate	:CALC:CLIM:UPD <END IMM>
Enables/disables composite limit test	:CALC:CLIM:STAT <ON OFF>
Specifies limit test feed data	:CALC:FEED <i>type</i>
Sets compliance check or limit test	:CALC:LIM:FUNC <COMP LIM>
Sets limit test lower or upper limit	:CALC:LIM:<LOW UPP> <i>value</i>
Sets compliance fail condition	:CALC:LIM:COMP:FAIL <OUT IN>
Enables/disables limit test	:CALC:LIM:STAT <ON OFF>
Asks limit test result pass or fail	:CALC:LIM:FAIL?

type is VOLT for voltage measurement data, CURR for current measurement data, CHAR for charge measurement data, RES for resistance measurement data, or MATH for math result data.

Table 2-12 Pass/Fail Judgment Example

<pre> Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "../../././execution result/Judgment1.txt" '2 ioObj.WriteString("*RST") ' Reset '4 Try ' Set voltage sweep from 0.0 V to 10.0V, 21 linear steps ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode swe") ioObj.WriteString(":sour:volt:star 0.0") ioObj.WriteString(":sour:volt:stop 10.0") ioObj.WriteString(":sour:volt:poin 21") ioObj.WriteString(":sour:swe:sta sing") '7 ' Set 2nA limited auto ranging ioObj.WriteString(":sens:curr:rang:auto:llim 2e-9") '15 ' Set measurement speed to 0.1 PLC ioObj.WriteString(":sens:curr:nplc:auto off") ioObj.WriteString(":sens:curr:nplc 0.1") '18 ' Enable predefined math expression power (voltage * current) ioObj.WriteString(":calc:math:name ""POWER""") ioObj.WriteString(":calc:math:stat on") '22 ' Set limit test mode to "Grading" ioObj.WriteString(":calc:clim:mode grad") ioObj.WriteString(":calc:clim:upd end") ioObj.WriteString(":calc:clim:stat on") '26 ' Feed math result and judge it is in +/-5 mW or not ioObj.WriteString(":calc:feed math") ioObj.WriteString(":calc:lim1:stat on") ioObj.WriteString(":calc:lim1:func lim") ioObj.WriteString(":calc:lim1:low -5.0e-3") ioObj.WriteString(":calc:lim1:upp +5.0e-3") '31 </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
7 to 12	Sets the voltage sweep output function. And sets the sweep output from 0 to 10 V, 21 points. Also sets the single sweep.
15	Sets the current auto ranging to 2nA limited auto ranging.
18 to 19	Sets the aperture time to 0.1 PLC.

```

Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef
filename As String)
    filename = "../.././execution result/Judgment1.txt" '2

    ioObj.WriteString("*RST") ' Reset '4

    Try ' Set voltage sweep from 0.0 V to 10.0V, 21 linear steps
        ioObj.WriteString(":sour:func:mode volt") '7
        ioObj.WriteString(":sour:volt:mode swe")
        ioObj.WriteString(":sour:volt:star 0.0")
        ioObj.WriteString(":sour:volt:stop 10.0")
        ioObj.WriteString(":sour:volt:poin 21")
        ioObj.WriteString(":sour:swe:sta sing")

        ' Set 2nA limited auto ranging
        ioObj.WriteString(":sens:curr:rang:auto:llim 2e-9") '15

        ' Set measurement speed to 0.1 PLC
        ioObj.WriteString(":sens:curr:nplc:auto off") '18
        ioObj.WriteString(":sens:curr:nplc 0.1")

        ' Enable predefined math expression power (voltage * current)
        ioObj.WriteString(":calc:math:name ""POWER""") '22
        ioObj.WriteString(":calc:math:stat on")

        ' Set limit test mode to "Grading"
        ioObj.WriteString(":calc:clim:mode grad") '26
        ioObj.WriteString(":calc:clim:upd end")
        ioObj.WriteString(":calc:clim:stat on")

        ' Feed math result and judge it is in +/-5 mW or not
        ioObj.WriteString(":calc:feed math") '31
        ioObj.WriteString(":calc:lim1:stat on")
        ioObj.WriteString(":calc:lim1:func lim")
        ioObj.WriteString(":calc:lim1:low -5.0e-3")
        ioObj.WriteString(":calc:lim1:upp +5.0e-3")
    
```

Line	Description
22 to 23	Enables POWER math function.
26 to 28	Enables composite limit test of grading mode.
31 to 35	Enables limit test 1 which judges if POWER is between -5 mW and $+5\text{ mW}$.

Programming Examples
Pass/Fail Judgment and Math Function

```

' Set arm count to 1, trigger count to 21
ioObj.WriteString(":arm:coun 1")
ioObj.WriteString(":trig:coun 21")
' 37

Catch ex As Exception
  Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")
' 45

' Turn on input switch
ioObj.WriteString(":inp on")
' 48

' Initiate transition and acquire
ioObj.WriteString(":init")
' 51

' Wait for operation complete
ioObj.WriteString("*OPC?")
' 54
Dim d As String = ioObj.ReadString()
Console.WriteLine("OPC?: " + d)
Console.WriteLine()

Try ' Retrieve measurement result (not bin numbers)
  ioObj.WriteString(":calc:data?")
  s = ioObj.ReadString()
' 59

  ioObj.WriteString(":calc:lim1:fail?")
  d = ioObj.ReadString()
  If d = 0 Then
    Console.WriteLine("PASS")
  Else
    Console.WriteLine("FAIL")
  End If
  Console.WriteLine()
' 63

Catch ex As Exception
  Console.WriteLine("An error occurred: " + ex.Message)
End Try
End Sub

```

Line	Description
37 to 38	Sets the arm count to 1, and the trigger count to 21 to perform a 21-step staircase sweep measurement.
45	Enables the output. And starts source output (0 V with the default setting).
48	Enables the input.
51	Starts staircase sweep measurement.

```

' Set arm count to 1, trigger count to 21
ioObj.WriteString(":arm:coun 1")
ioObj.WriteString(":trig:coun 21")
' 37

Catch ex As Exception
  Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")
' 45

' Turn on input switch
ioObj.WriteString(":inp on")
' 48

' Initiate transition and acquire
ioObj.WriteString(":init")
' 51

' Wait for operation complete
ioObj.WriteString("*OPC?")
' 54
Dim d As String = ioObj.ReadString()
Console.WriteLine("*OPC?: " + d)
Console.WriteLine()

Try ' Retrieve measurement result (not bin numbers)
  ioObj.WriteString(":calc:data?")
  s = ioObj.ReadString()
' 59

  ioObj.WriteString(":calc:lim1:fail?")
  d = ioObj.ReadString()
  If d = 0 Then
    Console.WriteLine("PASS")
  Else
    Console.WriteLine("FAIL")
  End If
  Console.WriteLine()
' 63

Catch ex As Exception
  Console.WriteLine("An error occurred: " + ex.Message)
End Try
End Sub

```

Line	Description
54 to 57	Waits for operation complete. And write "*OPC?: 1" on the console window when the operation is completed.
59 to 60	Reads the measurement result data.
63 to 70	Reads the judgement result. And write "PASS" or "FAIL" on the console window.

**Measurement
Result Example**

Result:
-1.172735E-12,+5.042879E-08,+1.943762E-07,+4.422486E-07,+7.842360E-07,+1.23
1724E-06,+1.758310E-06,+2.414372E-06,+3.137176E-06,+3.986594E-06,+4.897826E
-06,+5.953960E-06,+7.072061E-06,+8.307558E-06,+9.611092E-06,+1.106005E-05,+
1.258392E-05,+1.420136E-05,+1.591136E-05,+1.773358E-05,+1.967062E-05

Using Program Memory

A program example for using program memory is shown in [Table 2-14](#). This example is used to store a program in the program memory and execute it.

Table 2-13 Program Memory Commands

Function	Command
Returns the names of all programs defined in the program memory	:PROG:CAT?
Specifies memory program	:PROG:NAME "name"
Defines memory program ^a	:PROG:DEF <i>program_code</i>
Adds program code to the end of the memory program ^a	:PROG:APP <i>program_code</i>
Sets a value to the variable specified by <i>n</i> ^b	:PROG:VAR <i>n</i> "value"
Executes memory program ^a	:PROG:EXEC
Changes status of memory program ^a	:PROG:STAT <i>operation</i>
Blocks other commands until the program execution status changes to Paused or Stopped ^a	:PROG:WAIT? <i>timeout_in_seconds</i>
Deletes a memory program ^a	:PROG:DEL
Deletes all memory programs	:PROG:DEL:ALL

a. This function is effective for the memory program previously specified by the :PROG:NAME command.

b. Variables can be used in the memory program. They must be expressed as %*n*% (*n*: integer. 1 to 100) in the memory program.

operation is RUN to change to the running status, PAUS to change to the paused status, CONT to change to the running status, STOP to change to the stopped status, or STEP to perform step execution.

Table 2-14 Example to Use Program Memory

<pre> Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "../.././execution result/ProgramMemory1.txt" '2 ioObj.WriteString("*RST") ' Reset '4 Try ' Build program '6 Dim program As String = "" program = ":sour:func:mode volt\n" program += ":sour:volt:mode swe\n" program += ":sour:volt:star -10.0\n" program += ":sour:volt:stop 10\n" program += ":sour:volt:poin 21\n" program += ":sens:func ""curr"", ""volt""\n" program += ":sens:curr:rang:auto:llim 2e-9\n" program += ":sens:curr:nplc:auto off\n" program += ":sens:curr:nplc 0.1\n" program += ":arm:coun 1\n" program += ":trig:coun 21\n" program += ":outp 1\n" program += ":inp 1\n" program += ":init (@1)\n" ' Get program length Dim sProgramLength As String = String.Format("{0:#}", program.Length) '24 ioObj.WriteString(":prog:name ""sample""") '26 ioObj.WriteString(":prog:def #" + sProgramLength.Length.ToString() + sProgramLength + program) Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2980.
6 to 21	Enters program code to the “program” variable. The program is for performing voltage source current/voltage measure sweep measurement from –10V to 10V, 21 points, with the aperture time 0.1 PLC.
24	Gets the program length (number of characters in the “program” variable).
26 to 27	Stores the program code to the program memory as the program name “sample.”

Programming Examples
Using Program Memory

```

' Run program
ioObj.WriteString(":prog:stat run")                                ' 34

' Wait for operation complete
ioObj.WriteString("*OPC?")                                        ' 37
s = ioObj.ReadString()
Console.WriteLine("*OPC?: " + s)
Console.WriteLine()

Try ' Retrieve measurement result
    ioObj.WriteString(":fetch:arr:curr?")                        ' 43
    s = ioObj.ReadString()

Catch ex As Exception
    Console.WriteLine("An error occurred: " + ex.Message)
End Try
End Sub

```

Line	Description
34	Executes the memory program.
37 to 40	Waits for operation complete. And write "*OPC?: 1" on the console window when the operation is completed.
43 to 44	Reads the measurement result data.

**Measurement
Result Example**

Result: -7.480000E-05,+6.524500E-03,+1.311680E-02,+1.971080E-02,+2.630190E-02,+3.289570E-02,+3.948990E-02,+4.607940E-02,+5.266580E-02,+5.926410E-02,+6.585490E-02,+7.244700E-02,+7.903250E-02,+8.562770E-02,+9.221940E-02,+9.880730E-02,+1.053949E-01,+1.119873E-01,+1.185849E-01,+1.251736E-01,+1.317632E-01

Reading Binary Data

A program example for reading binary data is shown in [Table 2-16](#). This example is used to read data in the ASCII format and the 8-byte binary format.

This example code can be replaced with the code used to read data in a measurement program. For example, it can be used instead of the lines from 38 to 44 of the staircase sweep measurement program shown in [Table 2-5](#).

Table 2-15 Data Output Format Commands

Function	Command
Sets the data output format	:FORM[:DATA] <i>format</i>
Sets byte order of binary data	:FORM:BORD <i>byte_order</i>

format is ASC for the ASCII data output format, REAL,32 for the IEEE-754 single precision format (4-byte data), or REAL,64 for the IEEE-754 double precision format (8-byte data).

byte_order is NORM for the normal byte order from byte 1 to byte 4 or 8, or SWAP for the reverse byte order from byte 4 or 8 to byte 1.

**Measurement
Result Example**

Result: V (V), I (A), Time (sec), Status: -5.88E-05,2.85297E-06,0.021938,21
120.0222637,0.006749,0.030997,36480.0499995,0.0151897,0.037096,41600.075000
8,0.022776,0.041071,41600.0999998,0.0303624,0.045048,4160

Table 2-16 Example to Read Binary Data

```

' Select measure data elements                                     '2
ioObj.WriteString(":form:elem:sens volt,curr,time,stat")

' Retrieve measurement result & Output measurement result(Ascii format)
ioObj.WriteString(":form asc")                                     '5
ioObj.WriteString(":fetch:arr?")
Dim numOfElem As Integer = 4 'V, I, Time, Status
Dim data(numOfElem * trigCount - 1)
data = ioObj.ReadList(Ivi.Visa.Interop.IEEEASCIIType.ASCIIType_Any, ",")

Dim value As String = "V (V), I (A), Time (sec), Status: "
s = value
Console.WriteLine("ASCII format")
Console.WriteLine(value)
For i = LBound(data) To UBound(data)
    If (i + 1) Mod numOfElem = 0 Then
        Console.WriteLine(data(i).ToString())
        s = s + data(i).ToString()
    Else
        Console.Write(data(i).ToString() + ",")
        s = s + data(i).ToString() + ","
    End If
Next
Console.WriteLine()

' Retrieve measurement result & Output measurement result(Real64 format)
Console.WriteLine("REAL64 format")                                 '28
Console.WriteLine(value)
ioObj.WriteString(":form real,64")
ioObj.WriteString(":fetch:arr?")
Dim data64
data64 = ioObj.ReadIEEEBlock(Ivi.Visa.Interop.IEEEBinaryType.BinaryType_R8, False, True)
For i = LBound(data64) To UBound(data64)
    If (i + 1) Mod numOfElem = 0 Then
        Console.WriteLine(data64(i).ToString())
    Else
        Console.Write(data64(i).ToString() + ",")
    End If
Next
Console.WriteLine()

```

Line	Description
2	Specifies the data to return. This example selects voltage measurement data, current measurement data, time data, and status data.
5 to 23	Reads the measurement result data in the ASCII format.
28 to 44	Reads the measurement result data in the REAL,64 format.

Reading Data During Measurement

A program example for reading data is shown in [Table 2-18](#). This example is used to read data during measurement. This example returns data in ASCII format.

For example, this example code can be used instead of the lines from 4 to 39 of the sampling measurement program shown in [Table 2-9](#).

Table 2-17 Measurement Data Command

Function	Command
Read sense data	:SENS:DATA? [<i>offset</i> [, <i>size</i>]]

offset indicates the beginning of the data received *n* | *CURRent* | *START* (default).
size is the number of data to be received.

**Measurement
Result Example**

Data(1) -1.282000E-12,+4.999999E+00
Data(2) -6.000000E-14,+1.500000E+01
Data(3) +7.760000E-13,+2.500000E+01

Table 2-18 Example to Read Data during Measurement

<pre> Dim buffer As String ioObj.WriteString("*RST") ' Reset Try ' Set fixed-range current measurement ioObj.WriteString(":sens:func ""curr""") '5 ioObj.WriteString(":sens:curr:rang:auto off") ioObj.WriteString(":sens:curr:rang 2e-9") ' Set measurement speed to 0.1 PLC ioObj.WriteString(":sens:curr:nplc:auto off") '10 ioObj.WriteString(":sens:curr:nplc 0.1") ' Adjust trigger timing parameters ioObj.WriteString(":trig:acq:del 5.0") '14 ' Generate 3 triggers in 10 s period ioObj.WriteString(":trig:acq:sour tim") '17 ioObj.WriteString(":trig:acq:tim 10.0") ioObj.WriteString(":trig:acq:coun 3") Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try ' Turn on input switch ioObj.WriteString(":inp on") '26 ' Initiate acquire trigger with timestamp reset ioObj.WriteString(":sys:time:tim:coun:res:auto on") '29 ioObj.WriteString(":init:acq") '30 ' Query current and timestamp ioObj.WriteString(":form:elem:sens CURR,TIME") '31 ' Retrieving measurement result For count As Integer = 1 To 3 Step 1 Try ioObj.WriteString(":sens:data? curr,1") '36 buffer = ioObj.ReadString() Console.Write("Data(" + count.ToString() + ") " + buffer) Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try Next </pre>	
Line	Description
5 to 7	Set 2 nA fixed ranging current measurement.
10 to 11	Set 0.1 PLC measurement speed.
17 to 19	Set 3 points measurement with 10 seconds period.
26	Turn on input switch for current measurement.

```

Dim buffer As String
ioObj.WriteString("*RST") ' Reset

Try ' Set fixed-range current measurement
    ioObj.WriteString(":sens:func ""curr""") '5
    ioObj.WriteString(":sens:curr:rang:auto off")
    ioObj.WriteString(":sens:curr:rang 2e-9")

    ' Set measurement speed to 0.1 PLC
    ioObj.WriteString(":sens:curr:nplc:auto off") '10
    ioObj.WriteString(":sens:curr:nplc 0.1")

    ' Adjust trigger timing parameters
    ioObj.WriteString(":trig:acq:del 5.0") '14

    ' Generate 3 triggers in 10 s period
    ioObj.WriteString(":trig:acq:sour tim") '17
    ioObj.WriteString(":trig:acq:tim 10.0")
    ioObj.WriteString(":trig:acq:coun 3")

Catch ex As Exception
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on input switch
ioObj.WriteString(":inp on") '26

' Initiate acquire trigger with timestamp reset
ioObj.WriteString(":sys:time:tim:coun:res:auto on") '29
ioObj.WriteString(":init:acq") '30

' Query current and timestamp
ioObj.WriteString(":form:elem:sens CURR,TIME") '31

' Retrieving measurement result
For count As Integer = 1 To 3 Step 1
    Try
        ioObj.WriteString(":sens:data? curr,1") '36
        buffer = ioObj.ReadString()
        Console.WriteLine("Data(" + count.ToString() + ") " + buffer)
    Catch ex As Exception
        Console.WriteLine("An error occurred: " + ex.Message)
    End Try
Next

```

Line	Description
29	Set timestamp to be reset automatically by INITiate command.
30	Initiate acquire trigger system.
31	Set data query format to current and timestamp.
36	Query one measurement result without waiting operation complete.

This information is subject to change without notice.
© Keysight Technologies 2021
Edition 1, April 2021



B2980-90120
www.keysight.com