

Keysight Technologies B2961A/B2962A Low Noise Power Source

Notices

Copyright Notice

© Keysight Technologies 2012-2016

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

B2961-90020

Edition

Edition 1, August 2012
Edition 2, October 2012
Edition 3, October 2014
Edition 4, May 2016

Printed in:

Printed in Malaysia

Published by:

Keysight Technologies International Japan
G.K.
9-1, Takakura-cho, Hachioji-shi, Tokyo
192-0033 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 CONSE-

QUENTIAL 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.

About Customer Feedback

We love hearing from you. Please take a few moments and let us know how we can improve this manual. [Click here to open the Keysight B2900 manual feedback form.](#)

We respect your privacy. Be assured that Keysight will never sell or rent your information. Nor will Keysight share this information with other companies without your expressed consent. We make a commitment to you that we will respect and protect your privacy. Please see the details of this commitment in our Privacy Statement. [Click here to open the statement.](#)

In This Manual

This manual provides the information for controlling the Keysight Technologies B2961A/B2962A by using an external computer, and consists of the following chapters.

1. “Controlling the Keysight B2961A/B2962A”

Describes how to control the B2961A/B2962A on a task basis.

2. “Programming Examples”

Introduces example programs for controlling the B2961A/B2962A.

See *Keysight B2961A/B2962A User's Guide* for information about the B2961A/B2962A itself.

Refer to *Keysight B2961A/B2962A SCPI Command Reference* for the SCPI messages and conventions, data output format, error code, and the details on Keysight B2961A/B2962A SCPI commands.

1. Controlling the Keysight B2961A/B2962A

Before Starting	1-3
Software Requirements	1-3
Connecting to the Interface	1-3
Starting the Instrument Control	1-4
Controlling Various Functions	1-5
Setting the Power Frequency	1-5
Resetting to the Initial Settings	1-6
Setting the Beeper	1-6
Setting the Date and Time	1-6
Performing the Self-Test	1-6
Performing the Self-Calibration	1-6
Setting the Operations at Power On	1-7
Reading an Error Message	1-7
Clearing the Error Buffer	1-7
Reading Timestamp	1-8
Clearing Timestamp	1-8
Setting the Automatic Clear of Timestamp	1-8
Confirming the Firmware Revision	1-8
Setting the Remote Display Mode	1-9
Making a Screen Dump	1-9
Performing a File Operation	1-9
Controlling the Source Output	1-10
Enabling the Source Output	1-11
Setting the Source Output Mode	1-11
Applying the DC Voltage/Current	1-11
Stopping the Source Output	1-11
Setting the Limit/Compliance Value	1-11
Setting the Output Range	1-12
Setting the Pulse Output	1-12

Contents

Setting the Arbitrary Waveform Output	1-12
Setting the Sweep Operation	1-15
Setting the Sweep Output	1-16
Setting the Ranging Mode of the Sweep Source	1-16
Setting the List Sweep Output	1-16
Setting the Source Output Trigger	1-17
Setting the Source Wait Time	1-18
Setting the Output Filter	1-18
Setting the External Filter	1-18
Setting the Connection Type	1-19
Setting the Low Terminal State	1-19
Enabling or Disabling the High Capacitance Mode	1-19
Enabling or Disabling the Over Voltage/Current Protection	1-19
Specifying the Output-Off Status	1-20
Enabling or Disabling the Automatic Output-On Function	1-20
Enabling or Disabling the Automatic Output-Off Function	1-20
Using the Programmable Output Resistance Function	1-20
Controlling the Measurement Function	1-21
Enabling the Measurement Channel	1-21
Setting the Measurement Mode	1-21
Enabling or Disabling the Resistance Compensation	1-21
Performing Spot Measurement	1-22
Setting the Measurement Speed	1-22
Setting the Measurement Trigger	1-22
Setting the Measurement Wait Time	1-24
Performing Sweep Measurement	1-24
Stopping Measurement	1-24
Using the Math Function	1-25
Defining a Math Expression	1-25
Deleting an User Defined Math Expression	1-25
Enabling or Disabling the Math Function	1-25

Contents

Reading Math Result Data	1-25
Using the Trace Buffer	1-26
Setting the Trace Buffer	1-27
Reading the Trace Data	1-27
Using Program Memory	1-28
Defining a Memory Program	1-28
Deleting a Program	1-28
Controlling the Program Operation	1-28

2. Programming Examples

Preparations	2-3
To Create Your Project Template	2-3
To Create Control Program	2-4
DC Output	2-7
Pulse Output	2-9
Exponential Wave Output	2-13
Ramp Wave Output	2-17
Sinusoidal Wave Output	2-21
Square Wave Output	2-25
Trapezoidal Wave Output	2-29
Triangle Wave Output	2-33
User Defined Waveform Output	2-37
Staircase Sweep Output	2-41
Pulsed Sweep Output	2-45
List Sweep Output	2-49
Pulsed List Sweep Output	2-53

Contents

Using Program Memory 2-57

Reading Binary Data 2-60

1

Controlling the Keysight B2961A/B2962A

This chapter describes basic information to control the Keysight B2961A/B2962A, and consists of the following sections.

- “Before Starting”
- “Controlling Various Functions”
- “Controlling the Source Output”
- “Controlling the Measurement Function”
- “Using the Math Function”
- “Using the Trace Buffer”
- “Using Program Memory”

The following conventions are 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 must be placed there.
Square brackets []	Items within square brackets are optional. The representation [SOURce:]VOLTage means that SOURce: 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 B2961A/B2962A supports GPIB, LAN, and USB interfaces. All three interfaces are live at power-on. Select the interface used for controlling the B2961A/B2962A. Connect your interface cable to the appropriate interface connector.

For the information on configuring the interfaces, see *Keysight B2961A/B2962A 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 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 More > I/O > USB function keys.

Example:

```
address = "USB0::2391::12345::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 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 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
```

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 2012,7,1") '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.Binary
Type_UI1, False, True)

Dim dataSize As Integer = data.Length
Dim dumpname As String = "C:/temp/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 memory connected to the front panel USB connector, and performed by the :MMEM commands. Error occurs if an USB memory 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
```

Controlling the Source Output

This section describes how to control the source output of Keysight B2961A/B2962A.

- “Enabling the Source Output”
- “Setting the Source Output Mode”
- “Applying the DC Voltage/Current”
- “Stopping the Source Output”
- “Setting the Limit/Compliance Value”
- “Setting the Output Range”
- “Setting the Pulse Output”
- “Setting the Arbitrary Waveform Output”
- “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 Source Wait Time”
- “Setting the Output Filter”
- “Setting the External Filter”
- “Setting the Connection Type”
- “Setting the Low Terminal State”
- “Enabling or Disabling the High Capacitance Mode”
- “Enabling or Disabling the Over Voltage/Current Protection”
- “Specifying the Output-Off Status”
- “Enabling or Disabling the Automatic Output-On Function”
- “Enabling or Disabling the Automatic Output-Off Function”
- “Using the Programmable Output Resistance Function”

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")
```

Setting the Source Output Mode

Source output mode is set by the :SOUR:FUNC:MODE command.

Example

```
ioObj.WriteString(":SOUR:FUNC:MODE CURR") 'Current output
ioObj.WriteString(":SOUR:FUNC:MODE VOLT") 'Voltage output
```

Applying the DC Voltage/Current

DC current/voltage is immediately applied by the :SOUR:<CURR|VOLT> command during the source output is enabled.

If you want to control the DC current/voltage output timing using a trigger, use the :SOUR:<CURR|VOLT>:TRIG command. See [Figure 1-4](#).

Example

```
ioObj.WriteString(":SOUR:FUNC:MODE CURR")
ioObj.WriteString(":SOUR:CURR 1E-3") 'Outputs 1 mA immediately

ioObj.WriteString(":SOUR:FUNC:TRIG:CONT 1")
ioObj.WriteString(":SOUR:FUNC:MODE VOLT")
ioObj.WriteString(":SOUR:VOLT:MODE FIX")
ioObj.WriteString(":SOUR:VOLT:TRIG 1") 'Outputs 1 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 Limit/Compliance Value

Limit/compliance is set by the :SENS:<CURR|VOLT>:PROT command.

Example

```
ioObj.WriteString(":SENS:CURR:PROT 0.1") '100 mA compliance
```

```
ioObj.WriteString(":SENS:VOLT:PROT 10") '10 V compliance
```

NOTE**To set the positive limit and the negative limit individually**

Use the :SENS:<CURR|VOLT>:PROT:POS command to set the positive limit and the :SENS:<CURR|VOLT>:PROT:NEG command to set the negative limit. Do not use the :SENS:<CURR|VOLT>:PROT command.

Setting the Output Range

Output range is set by the :SOUR:<CURR|VOLT>:RANG command. And the auto range operation is enabled/disabled by the :SOUR:<CURR|VOLT>:RANG:AUTO command. The lower limit for the auto range operation is set by the :SOUR:<CURR|VOLT>:RANG:AUTO:LLIM command.

Example

```
ioObj.WriteString(":SOUR:VOLT:RANG:AUTO OFF")  
ioObj.WriteString(":SOUR:VOLT:RANG 20") '20 V range fix  
ioObj.WriteString(":SOUR:VOLT:RANG:AUTO ON")  
ioObj.WriteString(":SOUR:VOLT:RANG:AUTO:LLIM 2") '2 V range limit
```

Setting the Pulse Output

Pulse output is set by the :SOUR:FUNC:SHAP PULS, :SOUR:PULS:DEL, and :SOUR:PULS:WIDT commands. See [Figure 1-4](#).

Pulse base and peak values are set by the :SOUR:<CURR|VOLT> command and the :SOUR:<CURR|VOLT>:TRIG command respectively.

Example

```
ioObj.WriteString(":SOUR:FUNC:SHAP PULS")  
ioObj.WriteString(":SOUR:PULS:DEL 1E-3") 'Delay time 1 ms  
ioObj.WriteString(":SOUR:PULS:WIDT 1E-3") 'Pulse width 1 ms  
ioObj.WriteString(":SOUR:VOLT 0") 'Base 0 V  
ioObj.WriteString(":SOUR:VOLT:TRIG 1") 'Peak 1 V
```

NOTE**Outputting the pulse voltage/current**

Execute the :OUTP ON command to start outputting the pulse base value.

Execute the :INIT to perform the specified pulse output and measurement.

Setting the Arbitrary Waveform Output

Arbitrary waveform output is enabled by the :<CURR|VOLT>:MODE ARB command. And the output waveform is set by the :SOUR:ARB commands. See [Figures 1-1](#) and [1-2](#) for various waveforms and associated setup commands.

Waveform count is set by the :SOUR:ARB:COUN command.

Example

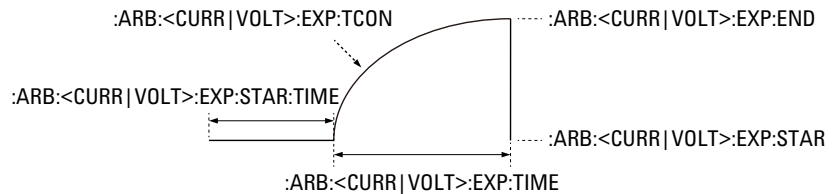
```
ioObj.WriteString(":SOUR:VOLT:MODE ARB")
ioObj.WriteString(":SOUR:ARB:FUNC SIN")
ioObj.WriteString(":SOUR:ARB:VOLT:SIN:AMPL 1")
ioObj.WriteString(":SOUR:ARB:VOLT:SIN:OFFS 0")
ioObj.WriteString(":SOUR:ARB:VOLT:SIN:FREQ 1")
```

'Sinusoidal wave
'Amplitude 1 V
'Offset 0 V
'Frequency 1 Hz

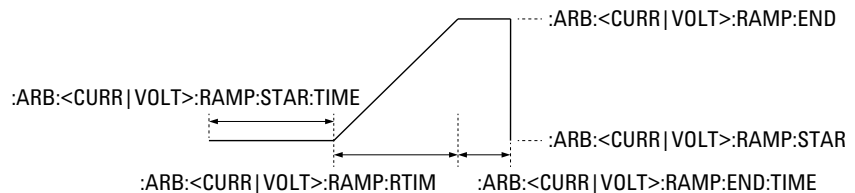
Figure 1-1

Variety of Arbitrary Waveforms, EXP, RAMP, SIN

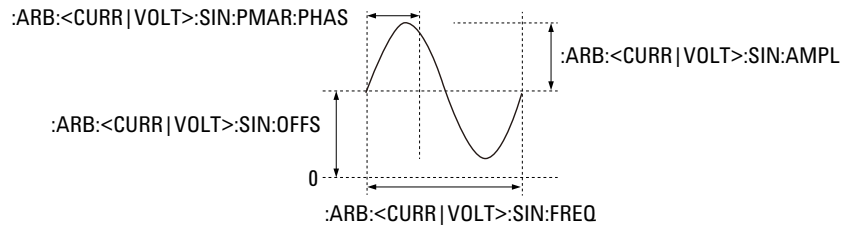
Exponential waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC EXP



Ramp waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC RAMP



Sinusoidal waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC SIN

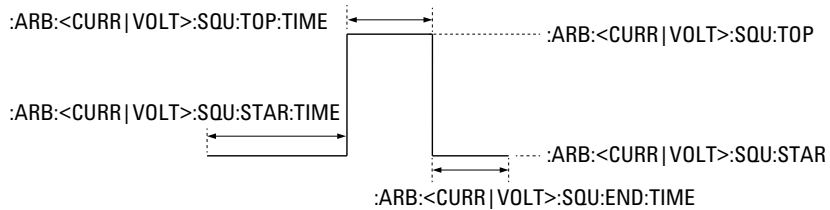


For all waveform, period must not exceed 1000 seconds.

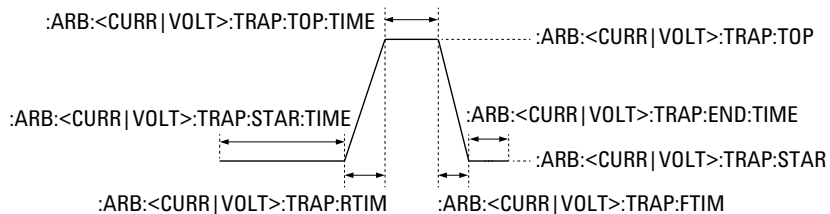
Figure 1-2

Variety of Arbitrary Waveforms, SQU, TRAP, TRI

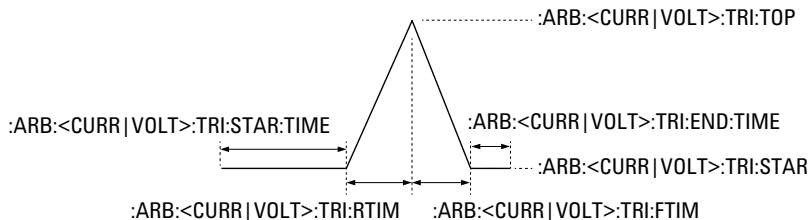
Square waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC SQU



Trapezoidal waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC TRAP



Triangle waveform :<CURR|VOLT>:MODE ARB, :ARB:FUNC TRI



For all waveform, period must not exceed 1000 seconds.

NOTE

To define your desired waveform

Use the :SOUR:ARB:<CURR|VOLT>:UDEF commands.

The :SOUR:ARB:<CURR|VOLT>:UDEF[:LEV] commands set the output level.

The :SOUR:ARB:<CURR|VOLT>:UDEF:TIME command sets the step time between adjacent points in a waveform.

Setting the Sweep Operation

For the variety of sweep output operation, see [Figure 1-3](#).

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.

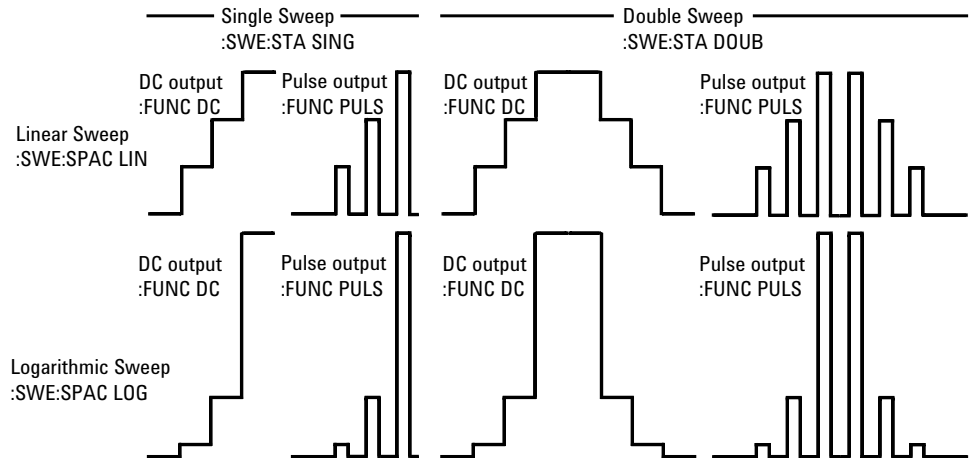
Sweep spacing, linear or log is set by the :SOUR:SWE:SPAC command.

Example

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

Figure 1-3

Variety of Sweep Outputs



Setting the Sweep Output

Staircase sweep output is set by the :SOUR:<CURR|VOLT>:MODE SWE command, the :SOUR:<CURR|VOLT>:<POIN|STEP> or :SOUR:SWE:POIN command, and the :SOUR:<CURR|VOLT>:<STAR|STOP> or :SOUR:<CURR|VOLT>:<CENT|SPAN> command. See [Figure 1-5](#).

Before performing the pulsed sweep output, it is necessary to set the staircase sweep output and pulse output. For details on setting the pulse output, see [“Setting the Pulse Output” on page 1-12](#). Also see [Figure 1-6](#).

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/current

Execute the :OUTP ON command to start outputting the value set by the :SOUR:<CURR|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  
ioObj.WriteString(":SOUR:SWE:RANG AUTO") 'Auto for each step
```

Setting the List Sweep Output

List sweep output is set by the :SOUR:<CURR|VOLT>:MODE LIST command and the :SOUR:LIST:<CURR|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:<CURR|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 1-4](#).

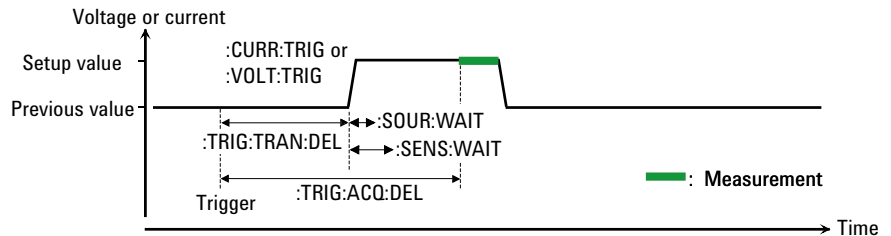
Example

```
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
```

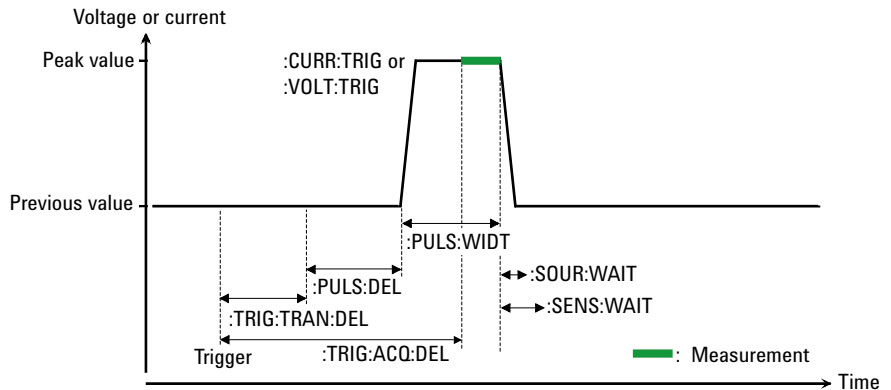
Figure 1-4

To Perform DC and Pulse Output and Spot Measurement

Constant source :FUNC DC, :CURR:MODE FIX or :VOLT:MODE FIX



Pulse source :FUNC PULS, :CURR:MODE FIX or :VOLT:MODE FIX



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*.

NOTE

If source channels are set as shown below, the source output starts simultaneously.

- Trigger source is set to the same mode.
 - Delay time is set to the same value.
 - Source output ranging mode is set to the fixed mode.
 - Source wait time control is set to OFF.
 - Measurement wait time control is set to OFF.
 - Measurement ranging mode is set to the fixed mode.
-

Setting the Source Wait Time

Source wait time is set by the :SOUR:WAIT commands. See Figures 1-5 and 1-6 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
```

Setting the Output Filter

Output filter is set by the :OUTP:FILT[:LPAS] commands.

Example

```
ioObj.WriteString(":OUTP:FILT ON")
ioObj.WriteString(":OUTP:FILT:AUTO OFF")
ioObj.WriteString(":OUTP:FILT:FREQ 10E+3") '10 kHz
```

Setting the External Filter

To use the N1294A-020/021/022 external filter, specify the filter type HCULNF (N1294A-020 high current ultra low noise filter), ULNF (N1294A-021 ultra low noise filter), or LNF (N1294A-022 low noise filter) by using the :OUTP:FILT:EXT:TYPE command and enable the external filter by using the :OUTP:FILT:EXT:STAT command.

Example

```
ioObj.WriteString(":OUTP:FILT:EXT:TYPE LNF") 'N1294A-022 filter  
ioObj.WriteString(":OUTP:FILT:EXT:STAT ON") 'Enables filter
```

Setting the Connection Type

Connection type, 2-wire or 4-wire is set by the :SENS:REM command.

Example

```
ioObj.WriteString(":SENS:REM ON") '4-wire
```

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 GRO") 'Ground  
ioObj.WriteString(":OUTP ON")
```

Enabling or Disabling the High Capacitance Mode

High capacitance mode is set by the :OUTP:HCAP command.

Example

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

Enabling or Disabling the Over Voltage/Current Protection

Over voltage/current protection is set by the :OUTP:PROT command.

Example

```
ioObj.WriteString(":OUTP:PROT 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
```

Enabling or Disabling the Automatic Output-On Function

Automatic output-on function is set by the :OUTP:ON:AUTO command.

Example

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

Enabling or Disabling the Automatic Output-Off Function

Automatic output-off function is set by the :OUTP:OFF:AUTO command.

Example

```
ioObj.WriteString(":OUTP:OFF:AUTO ON")
```

Using the Programmable Output Resistance Function

The programmable output resistance function is set by the :OUTP:RES commands.

Example

```
ioObj.WriteString(":OUTP:RES:MODE FIX") 'Sets fix mode
ioObj.WriteString(":OUTP:RES:SER 1")   'Sets 1 ohm series R
ioObj.WriteString(":OUTP:RES:STAT ON")  'Enables function

ioObj.WriteString(":OUTP:RES:MODE FIX") 'Sets fix mode
ioObj.WriteString(":OUTP:RES:SHUN 50")  'Sets 50 ohm shunt R
ioObj.WriteString(":OUTP:RES:STAT ON")  'Enables function

ioObj.WriteString(":OUTP:RES:MODE EMUL") 'Sets emulation mode
ioObj.WriteString(":OUTP:RES:EMUL:VOLT 0.1,0.2,0.3,0.4,0.5,0.6,0.7")
                                     'Sets voltage list data for emulation table
ioObj.WriteString(":OUTP:RES:EMUL:CURREN 0.075,0.075,0.07,0.06,0.05,0.03,0.01")
                                     'Sets current list data for emulation table
ioObj.WriteString(":OUTP:RES:EMUL:MODE CUREN,CURREN,CURREN,VOLT,VOLT,VOLT")
                                     'Sets source type list data for emulation table
ioObj.WriteString(":OUTP:RES:STAT ON")  'Enables function
```

Controlling the Measurement Function

This section describes how to control the measurement function of Keysight B2961A/B2962A.

- “Enabling the Measurement Channel”
- “Setting the Measurement Mode”
- “Enabling or Disabling the Resistance Compensation”
- “Performing Spot Measurement”
- “Setting the Measurement Speed”
- “Setting the Measurement Trigger”
- “Setting the Measurement Wait Time”
- “Performing Sweep Measurement”
- “Stopping Measurement”

Enabling the Measurement Channel

Measurement channel is enabled by the :OUTP ON command.

Example

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

Setting the Measurement Mode

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

Example

```
ioObj.WriteString(":SENS:FUNC:ALL")  
ioObj.WriteString(":SENS:FUNC:OFF " "RES" " ")  
  
ioObj.WriteString(":SENS:FUNC:OFF:ALL")  
ioObj.WriteString(":SENS:FUNC " "RES" " ")
```

Enabling or Disabling the Resistance Compensation

Resistance compensation is set by the :SENS:RES:OCOM command.

Example

```
ioObj.WriteString(":SENS:RES:OCOM ON")      'Enables compensation
```

Performing Spot Measurement

Spot measurement is performed by the :MEAS:<CURR|VOLT|RES>? command or the :MEAS? command. See [Figure 1-4](#) for the spot measurement.

Example

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

NOTE

For the :MEAS? command, the measurement parameters are specified by :SENS:FUNC and the returned data is specified by :FORM:ELEM:SENS.

Setting the Measurement Speed

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

Example

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

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 1-4](#), [1-5](#), and [1-6](#).

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 measurement channels are set as shown below, the measurement starts simultaneously.

- Trigger source is set to the same mode.
 - Delay time is set to the same value.
 - Measurement wait time control is set to OFF.
 - Measurement ranging mode is set to the fixed mode.
-

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 1-5 To Perform Staircase Sweep Output and Measurement

Staircase sweep source :FUNC DC, :CURR:MODE SWE or :VOLT:MODE SWE

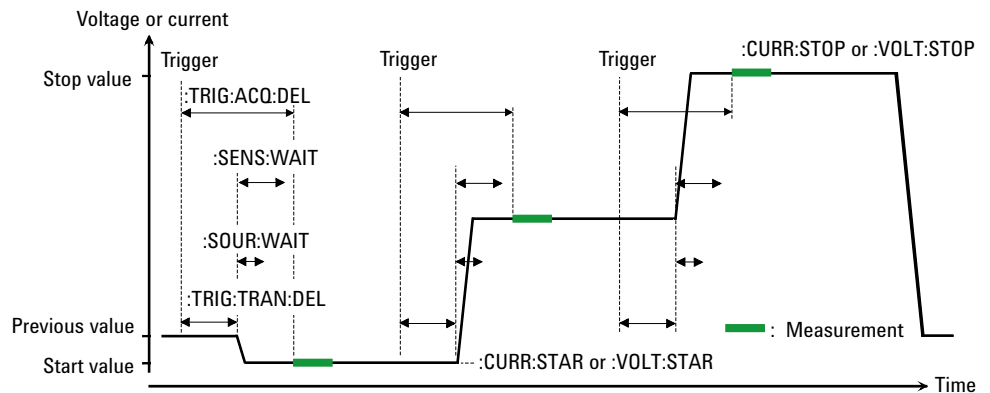
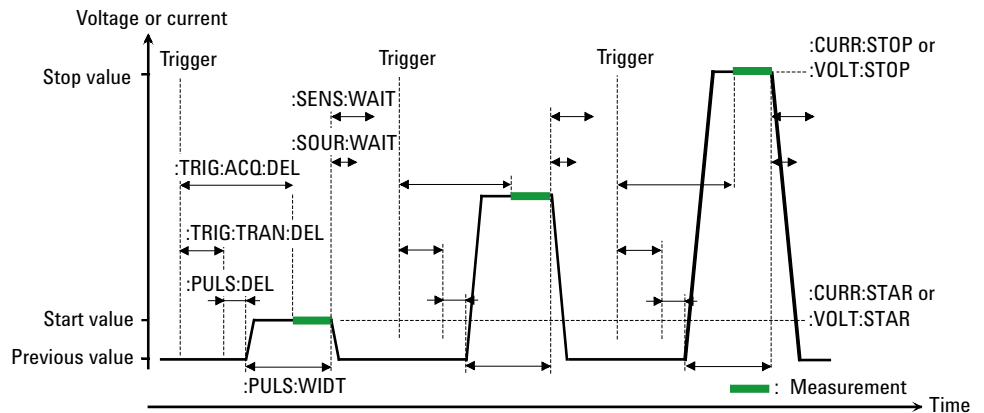


Figure 1-6 To Perform Pulsed Sweep Output and Measurement

Pulsed sweep source :FUNC PULS, :CURR:MODE SWE or :VOLT:MODE SWE



Setting the Measurement Wait Time

Measurement wait time is set by the :SENS:WAIT commands. See Figures 1-5 and 1-6 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 follows.

3. Set the staircase sweep source and the required source functions. For details, see [“Controlling the Source Output” on page 1-10](#).
4. Set the required measurement functions. For details, see previous topics in this section.
5. Set the trigger condition. See [“Setting the Source Output Trigger” on page 1-17](#) and [“Setting the Measurement Trigger” on page 1-22](#).
6. Enable the channel. See [“Enabling the Measurement Channel” on page 1-21](#).

The channel starts output set by the :SOUR:<CURR|VOLT> command.

7. Execute the :INIT command to start measurement.

For the programming example, see [“Staircase Sweep Output” on page 2-41](#).

NOTE

To get measurement result data, use a :FETC subsystem command. For example, the :FETC:CURR? command returns the latest current measurement data. The :FETC?command returns the latest data for the parameters specified by the :FORM:ELEM:SENS command.

For details on the :FETC subsystem commands, see *SCPI Command Reference*.

Stopping Measurement

Measurement is stopped by the :OUTP OFF command.

Example

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


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.

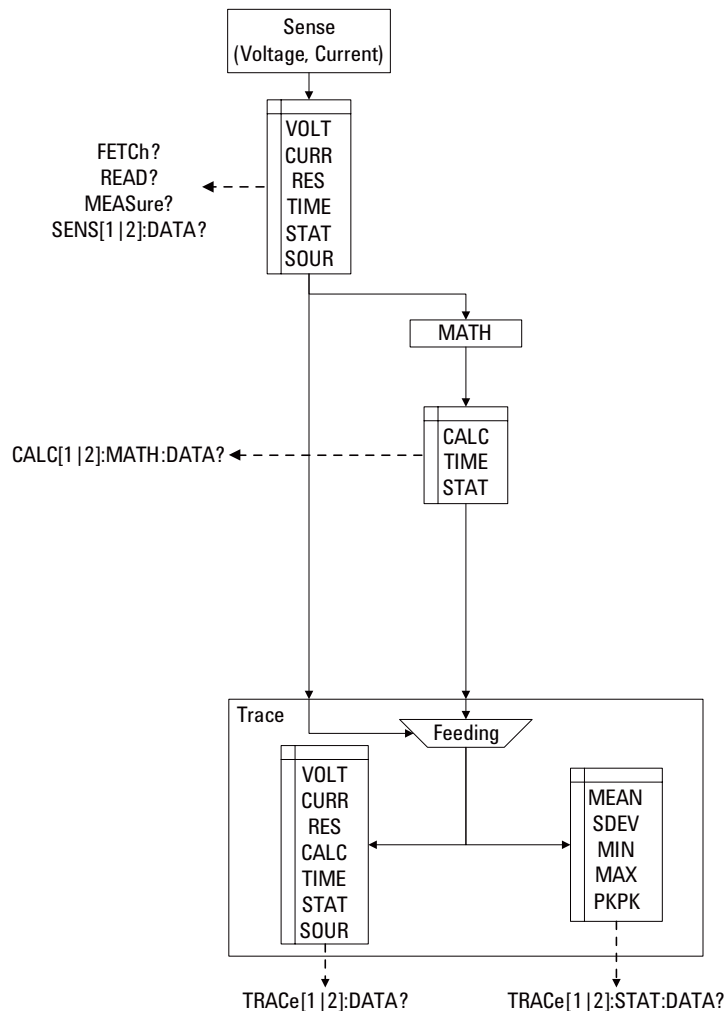
Using the Trace Buffer

This section describes how to use the trace buffer.

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

Figure 1-7

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 #213:OUTP:STAT ON") 'Definite length
ioObj.WriteString(":PROG:NAME " "sample1"")
ioObj.WriteString(":PROG:DEF #0:OUTP: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")
```


This chapter provides the following sections which explain programming example.

- “Preparations”
- “DC Output”
- “Pulse Output”
- “Exponential Wave Output”
- “Ramp Wave Output”
- “Sinusoidal Wave Output”
- “Square Wave Output”
- “Trapezoidal Wave Output”
- “Triangle Wave Output”
- “User Defined Waveform Output”
- “Staircase Sweep Output”
- “Pulsed Sweep Output”
- “List Sweep Output”
- “Pulsed List Sweep Output”
- “Using Program Memory”
- “Reading Binary Data”

NOTE**About Numeric Suffix**

Command header may be accompanied by a numeric suffix *c* for specifying the instrument channel. *c* must be 1 for using the channel 1, or 2 for using the channel 2. Abbreviating *c* gives the same result as specifying 1.

For example, the :OUTP ON command and the :OUTP1 ON command enable the channel 1, and the :OUTP2 ON command enables the channel 2.

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 B2960control subprogram in the template.

Preparations

This section provides the basic information for programming of the automatic measurement using the Keysight B2961A/B2962A, Keysight IO Libraries, and Microsoft Visual Basic .NET.

- “To Create Your Project Template”
- “To Create Control 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 B2961A/B2962A (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 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. \B2960\my_temp).

NOTE

To Start Program

If you create the control 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.

To Create Control Program

Create the control 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:

- Source mode, voltage or current
- Source function

Arbitrary waveform, DC output, pulsed output, staircase sweep, your desired waveform, and so on.

- Number of waves/repetitions, and trigger timing
- Device under test and parameters/characteristics to measure, optional

Step 2. Make a copy of your project template (e.g. \B2960\my_temp to \B2960\source\my_temp).

Step 3. Rename the copy (e.g. \B2960\source\my_temp to \B2960\source\wave1).

Step 4. Launch Visual Basic .NET.

Step 5. Open the project (e.g. \B2960\source\wave1).

Step 6. Open the module that contains the template code as shown in [Table 2-1](#). On the code window, complete the B2960control subprogram.

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

Step 8. Save the project (e.g. \B2960\source\wave1).

Table 2-1 Example Template Program Code

<pre> Module Module1 Sub Main() 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 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 B2960control(ioObj, s, filename) Console.Write(filedata + s) MsgBox("Click OK to close the console window.", vbOKOnly, "") FileOpen(1, filename, OpenMode.Output, OpenAccess.Write, OpenShare.LockReadW rite) Print(1, filedata + s) FileClose(1) ioObj.IO.Close() 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 B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As Stri ng, ByRef filename As String) filename = "C:/temp/exdata1.txt" End Sub End Module </pre>	
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 instrument specified by the GPIB address ifAddress=23 on the interface GPIB0.

Programming Examples Preparations

```

Module Module1

    Sub Main()
        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
            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

            B2960control(ioObj, s, filename)
            Console.Write(filedata + s)
            MsgBox("Click OK to close the console window.", vbOKOnly, "")

            FileOpen(1, filename, OpenMode.Output, OpenAccess.Write, OpenShare.LockReadW
rite)
            Print(1, filedata + s)
            FileClose(1)

            ioObj.IO.Close()
            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 B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As Stri
ng, ByRef filename As String)
            filename = "C:/temp/exdata1.txt"
        End Sub

    End Module

```

Line	Description
21 to 23	Calls the B2960control subprogram. And displays the example data in a console window.
25 to 27	Saves the data to a file specified by filename.
29 to 35	Breaks the connection with the instrument specified by ifAddress=23.
37 to 39	B2960control subprogram. Instrument control program code should be entered here.

DC Output

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

DC output and measurement can be performed by using the following commands.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Sets source output range	<code>[[:SOUR[c]]:v-or-c:RANG:AUTO <ON OFF></code>
	<code>[[:SOUR[c]]:v-or-c:RANG <i>value</i></code>
Sets source output value	<code>[[:SOUR[c]]:v-or-c <i>value</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:<i>func</i>2:APER <i>time</i></code>
	<code>:SENS[c]:<i>func</i>2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates measurement and reads result data (latest data)	<code>:MEAS? [<i>chanlist</i>]</code>
	<code>:MEAS:<i>func</i>? [<i>chanlist</i>]</code>

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current measurement.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Programming Examples

DC Output

Table 2-2 DC Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/FixedDcl.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage output to 0.1 V ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt 0.1") ' 6 ' Set 100 mA fixed-range current measurement ioObj.WriteString(":sens:func "curr"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' 11 Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try ' Turn on output switch ioObj.WriteString(":outp on") ' 20 Try ' Initiate measurement and retrieve measurement result ioObj.WriteString(":meas:curr? (@1)") s = ioObj.ReadString() ' 22 Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try End Sub </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 8	Sets the voltage source function. And sets the source value to 0.1 V.
11 to 13	Sets the current measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
20	Enables the channel. And starts DC output.
22 to 24	Performs measurement and reads the measurement result data.

Measurement Result Example

Result: +9.000000E-05

Pulse Output

A program example of pulse output is shown in [Table 2-3](#). This example is used to apply pulsed voltage and measure current three times.

Pulse output and measurement can be performed by using the following commands.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Sets pulse output	<code>[[:SOUR[c]]:FUNC[:SHAP] PULS</code>
Sets source output range	<code>[[:SOUR[c]]:<i>v-or-c</i>:RANG:AUTO <ON OFF></code>
	<code>[[:SOUR[c]]:<i>v-or-c</i>:RANG <i>value</i></code>
Sets source output value	<code>[[:SOUR[c]]:<i>v-or-c</i> <i>value</i></code>
Sets pulse peak value	<code>[[:SOUR[c]]:<i>v-or-c</i>:TRIG <i>value</i></code>
Sets pulse delay time	<code>[[:SOUR[c]]:PULS:DEL <i>time</i></code>
Sets pulse width	<code>[[:SOUR[c]]:PULS:WIDT <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:<i>func</i>2:APER <i>time</i></code>
	<code>:SENS[c]:<i>func</i>2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:<i>v-or-c</i>:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>

Programming Examples

Pulse Output

Function	Command
Reads result data (latest data)	:FETC[:SCAL]? [<i>chanlist</i>]
	:FETC[:SCAL]: <i>type</i> ? [<i>chanlist</i>]
Reads result data (array data)	:FETC:ARR? [<i>chanlist</i>]
	:FETC:ARR: <i>type</i> ? [<i>chanlist</i>]

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-3 Pulse Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/FixedPulse1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage pulse output ioObj.WriteString(":sour:func:mode volt") ' 7 ioObj.WriteString(":sour:func:shap puls") ' Set base/peak voltages to 0.0/0.1 V ioObj.WriteString(":sour:volt 0") ' 11 ioObj.WriteString(":sour:volt:trig 0.1") ' Set delay/width to 500 us/1 ms ioObj.WriteString(":sour:puls:del 0.5e-3") ' 15 ioObj.WriteString(":sour:puls:width 1.0e-3") ' Set 100 mA fixed-range current measurement ioObj.WriteString(":sens:func ""curr""") ' 19 ioObj.WriteString(":sens:curr:aper 1e-4") ioObj.WriteString(":sens:curr:prot 0.1") ' Adjust trigger timing parameters ioObj.WriteString(":trig:tran:del 1.5e-3") ' 24 ioObj.WriteString(":trig:acq:del 2.9e-3") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
7 to 8	Sets the voltage source function. And sets the pulse output function.
11 to 12	Sets the pulse base voltage and the pulse peak voltage.
15 to 16	Sets the pulse delay time and the pulse width.
19 to 21	Sets the current measurement function. And sets the aperture time to 0.1 ms and the current limit (compliance) value to 0.1 A.
24 to 25	Sets the transient (source) delay time and the acquire (measurement) delay time.

Programming Examples

Pulse Output

```

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

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

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

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")

Try ' Retrieve measurement result
    ioObj.WriteString(":fetc:arr:curr? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
28 to 30	Sets the timer trigger source. And sets the trigger interval to 4 ms, and the trigger count to 3. The B2961A/B2962A will perform the pulsed spot measurement three times.
37	Enables the channel. And starts source output.
40	Starts pulse output and pulsed spot measurement.
42 to 44	Reads the measurement result data.

Measurement Result Example

Result: +9.000000E-05,+9.000000E-05,+9.000000E-05

Exponential Wave Output

A program example of exponential wave output is shown in [Table 2-4](#). This example is used to apply exponential wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the exponential wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects exponential wave output	<code>[[:SOUR[c]]:ARB:FUNC EXP</code>
Sets exponential wave start level	<code>[[:SOUR[c]]:ARB:v-or-c:EXP:STAR <i>level</i></code>
Sets exponential wave end level	<code>[[:SOUR[c]]:ARB:v-or-c:EXP:END <i>level</i></code>
Sets exponential wave start time	<code>[[:SOUR[c]]:ARB:v-or-c:EXP:STAR:TIME <i>time</i></code>
Sets time constant	<code>[[:SOUR[c]]:ARB:v-or-c:EXP:TCON <i>value</i></code>
Sets exponential wave output time	<code>[[:SOUR[c]]:ARB:v-or-c:EXP:TIME <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>
Reads result data (array data)	<code>:FETC:ARR:<i>type</i>? [<i>chanlist</i>]</code>

Programming Examples

Exponential Wave Output

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-4 Exponential Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/ExponentialWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set exponential wave voltage output ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func exp") ioObj.WriteString(":sour:arb:volt:exp:star 0") ioObj.WriteString(":sour:arb:volt:exp:end 5") ioObj.WriteString(":sour:arb:volt:exp:star:time 0.1") ioObj.WriteString(":sour:arb:volt:exp:tcon 0.2") ioObj.WriteString(":sour:arb:volt:exp:time 0.9") ' Set voltage measurement ioObj.WriteString(":sens:func " & "volt") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate triggers ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 14	Sets the exponential wave output voltage from 0 V to 5 V and the time parameters. See Figure 1-1 for the relation between the commands and the waveform.
16 to 19	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
22 to 23	Sets the transient trigger. Source output will be triggered once.
24 to 26	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Exponential Wave Output

```

Catch ex As Exception                                     '28
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             '33

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                           '36

Try ' Retrieve measurement result                         '38
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
33	Enables the channel. And starts source output (0 V with the default setting).
36	Starts the exponential wave output and monitor.
38 to 40	Reads the measurement result data.

Measurement Result Example

Result:

```

+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+1.800000E-02,+2.610000E-01
,+4.920000E-01,+7.120000E-01,+9.210000E-01,+1.120000E+00,+1.30900
0E+00,+1.489000E+00,+1.660000E+00,+1.823000E+00,+1.978000E+00,+2.
125000E+00,+2.266000E+00,+2.399000E+00,+2.526000E+00,+2.646000E+0
0,+2.761000E+00,+2.870000E+00,+2.974000E+00,+3.073000E+00,+3.1670
00E+00,+3.256000E+00,+3.341000E+00,+3.422000E+00,+3.499000E+00,+3
.572000E+00,+3.642000E+00,+3.708000E+00,+3.771000E+00,+3.831000E+
00,+3.888000E+00,+3.942000E+00,+3.994000E+00,+4.043000E+00,+4.090
000E+00,+4.134000E+00,+4.176000E+00,+4.217000E+00,+4.255000E+00,+
4.291000E+00,+4.326000E+00,+4.359000E+00,+4.390000E+00,+4.420000E
+00,+4.448000E+00,+4.475000E+00,+4.500000E+00,+4.525000E+00,+4.54
8000E+00,+4.570000E+00,+4.591000E+00,+4.611000E+00,+4.630000E+00,+
4.648000E+00,+4.665000E+00,+4.681000E+00,+4.697000E+00,+4.712000
E+00,+4.726000E+00,+4.739000E+00,+4.752000E+00,+4.764000E+00,+4.7
76000E+00,+4.786000E+00,+4.797000E+00,+4.807000E+00,+4.816000E+00
,+4.825000E+00,+4.834000E+00,+4.842000E+00,+4.850000E+00,+4.85700
0E+00,+4.864000E+00,+4.870000E+00,+4.877000E+00,+4.883000E+00,+4.
889000E+00,+4.894000E+00,+4.899000E+00,+4.904000E+00,+4.909000E+0
0,+4.913000E+00,+4.917000E+00,+4.921000E+00,+4.925000E+00,+4.9290
00E+00,+4.932000E+00,+4.936000E+00

```

Ramp Wave Output

A program example of ramp wave output is shown in [Table 2-5](#). This example is used to apply ramp wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the ramp wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects ramp wave output	<code>[[:SOUR[c]]:ARB:FUNC RAMP</code>
Sets ramp wave start level	<code>[[:SOUR[c]]:ARB:v-or-c:RAMP:STAR <i>level</i></code>
Sets ramp wave end level	<code>[[:SOUR[c]]:ARB:v-or-c:RAMP:END <i>level</i></code>
Sets ramp wave start time	<code>[[:SOUR[c]]:ARB:v-or-c:RAMP:STAR:TIME <i>time</i></code>
Sets ramp wave ramp time	<code>[[:SOUR[c]]:ARB:v-or-c:RAMP:RTIM <i>time</i></code>
Sets ramp wave end time	<code>[[:SOUR[c]]:ARB:v-or-c:RAMP:END:TIME <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>
Reads result data (array data)	<code>:FETC:ARR:<i>type</i>? [<i>chanlist</i>]</code>

Programming Examples

Ramp Wave Output

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-5 Ramp Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/RampWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set ramp wave voltage output ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func ramp") ioObj.WriteString(":sour:arb:volt:ramp:star 0") ioObj.WriteString(":sour:arb:volt:ramp:end 5") ioObj.WriteString(":sour:arb:volt:ramp:star:time 0.2") ioObj.WriteString(":sour:arb:volt:ramp:rtime 0.4") ioObj.WriteString(":sour:arb:volt:ramp:end:time 0.4") ' Set voltage measurement ' 16 ioObj.WriteString(":sens:func \"volt\"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate triggers ' 21 ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 14	Sets the ramp wave output voltage from 0 V to 5 V and the time parameters. See Figure 1-1 for the relation between the commands and the waveform.
16 to 19	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
22 to 23	Sets the transient trigger. Source output will be triggered once.
24 to 26	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Ramp Wave Output

```

Catch ex As Exception                                     '28
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             '33

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                          '36

Try ' Retrieve measurement result                        '38
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
33	Enables the channel. And starts source output (0 V with the default setting).
36	Starts the ramp wave output and monitor.
38 to 40	Reads the measurement result data.

Measurement Result Example

Result:

```

+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+8.
100000E-02,+2.060000E-01,+3.310000E-01,+4.560000E-01,+5.810000E-0
1,+7.060000E-01,+8.310000E-01,+9.560000E-01,+1.081000E+00,+1.2060
00E+00,+1.331000E+00,+1.456000E+00,+1.581000E+00,+1.706000E+00,+1.
831000E+00,+1.956000E+00,+2.081000E+00,+2.206000E+00,+2.331000E+
00,+2.456000E+00,+2.580000E+00,+2.705000E+00,+2.830000E+00,+2.955
000E+00,+3.080000E+00,+3.205000E+00,+3.330000E+00,+3.455000E+00,+
3.580000E+00,+3.705000E+00,+3.830000E+00,+3.955000E+00,+4.080000E
+00,+4.205000E+00,+4.330000E+00,+4.455000E+00,+4.580000E+00,+4.70
5000E+00,+4.830000E+00,+4.955000E+00,+5.000000E+00,+5.000000E+00,+
5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000
E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.0
00000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00
,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.00000
0E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.
000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+0
0,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.0000
00E+00,+5.000000E+00,+5.000000E+00

```


Sinusoidal Wave Output

A program example of sinusoidal wave output is shown in [Table 2-6](#). This example is used to apply sinusoidal wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the sinusoidal wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects sinusoidal wave output	<code>[[:SOUR[c]]:ARB:FUNC SIN</code>
Sets sinusoidal wave signal level	<code>[[:SOUR[c]]:ARB:v-or-c:SIN:AMPL <i>level</i></code>
Sets sinusoidal wave frequency	<code>[[:SOUR[c]]:ARB:v-or-c:SIN:FREQ <i>frequency</i></code>
Sets offset value	<code>[[:SOUR[c]]:ARB:v-or-c:SIN:OFFS <i>value</i></code>
Sets phase marker	<code>[[:SOUR[c]]:ARB:v-or-c:SIN:PMAR:PHAS <i>value</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>
Reads result data (array data)	<code>:FETC:ARR:<i>type</i>? [<i>chanlist</i>]</code>

Programming Examples

Sinusoidal Wave Output

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-6 Sinusoidal Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/SinusoidalWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set sinusoidal wave voltage output ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func sin") ioObj.WriteString(":sour:arb:volt:sin:ampl 1") ioObj.WriteString(":sour:arb:volt:sin:freq 1") ' 6 ' Set voltage measurement ioObj.WriteString(":sens:func \"volt\"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' 13 ' Generate triggers ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") ' 18 </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 11	Sets the sinusoidal wave output with the signal level 1 V and the frequency 1 Hz. See Figure 1-1 for the relation between the commands and the waveform.
13 to 16	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
19 to 20	Sets the transient trigger. Source output will be triggered once.
21 to 23	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Sinusoidal Wave Output

```

Catch ex As Exception                                     '25
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             '30

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                           '33

Try ' Retrieve measurement result                         '35
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
30	Enables the channel. And starts source output (0 V with the default setting).
33	Starts the sinusoidal wave output and monitor.
35 to 37	Reads the measurement result data.

Measurement Result Example

```

Result:
+0.000000E+00,+0.000000E+00,+0.000000E+00,+3.450000E-02,+9.720000
E-02,+1.595000E-01,+2.212000E-01,+2.820000E-01,+3.417000E-01,+4.0
00000E-01,+4.568000E-01,+5.117000E-01,+5.647000E-01,+6.154000E-01
,+6.637000E-01,+7.093000E-01,+7.522000E-01,+7.921000E-01,+8.28800
0E-01,+8.623000E-01,+8.924000E-01,+9.190000E-01,+9.419000E-01,+9.
612000E-01,+9.766000E-01,+9.882000E-01,+9.958000E-01,+9.996000E-0
1,+9.994000E-01,+9.953000E-01,+9.872000E-01,+9.752000E-01,+9.5940
00E-01,+9.398000E-01,+9.165000E-01,+8.896000E-01,+8.591000E-01,+8
.253000E-01,+7.882000E-01,+7.480000E-01,+7.049000E-01,+6.590000E-
01,+6.104000E-01,+5.595000E-01,+5.064000E-01,+4.512000E-01,+3.943
000E-01,+3.358000E-01,+2.760000E-01,+2.151000E-01,+1.534000E-01,+
9.100000E-02,+2.830000E-02,-3.450000E-02,-9.720000E-02,-1.595000E
-01,-2.212000E-01,-2.820000E-01,-3.417000E-01,-4.000000E-01,-4.56
7000E-01,-5.117000E-01,-5.646000E-01,-6.153000E-01,-6.636000E-01,
-7.093000E-01,-7.521000E-01,-7.920000E-01,-8.288000E-01,-8.623000
E-01,-8.924000E-01,-9.190000E-01,-9.419000E-01,-9.611000E-01,-9.7
66000E-01,-9.882000E-01,-9.958000E-01,-9.996000E-01,-9.994000E-01
,-9.953000E-01,-9.872000E-01,-9.752000E-01,-9.594000E-01,-9.39800
0E-01,-9.165000E-01,-8.896000E-01,-8.592000E-01,-8.253000E-01,-7.
883000E-01,-7.481000E-01,-7.049000E-01,-6.590000E-01,-6.105000E-0
1,-5.595000E-01,-5.064000E-01,-4.512000E-01,-3.943000E-01,-3.3580
00E-01,-2.760000E-01,-2.151000E-01

```

Square Wave Output

A program example of square wave output is shown in [Table 2-7](#). This example is used to apply square wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the square wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects square wave output	<code>[[:SOUR[c]]:ARB:FUNC SQU</code>
Sets square wave start level	<code>[[:SOUR[c]]:ARB:v-or-c:SQU:STAR <i>level</i></code>
Sets square wave top level	<code>[[:SOUR[c]]:ARB:v-or-c:SQU:TOP <i>level</i></code>
Sets square wave start time	<code>[[:SOUR[c]]:ARB:v-or-c:SQU:STAR:TIME <i>time</i></code>
Sets square wave top time	<code>[[:SOUR[c]]:ARB:v-or-c:SQU:TOP:TIME <i>time</i></code>
Sets square wave end time	<code>[[:SOUR[c]]:ARB:v-or-c:SQU:END:TIME <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>
Reads result data (array data)	<code>:FETC:ARR:<i>type</i>? [<i>chanlist</i>]</code>

Programming Examples

Square Wave Output

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-7 Square Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/SquareWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set square wave voltage output ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func squ") ioObj.WriteString(":sour:arb:volt:squ:star 0") ioObj.WriteString(":sour:arb:volt:squ:top 5") ioObj.WriteString(":sour:arb:volt:squ:star:time 0.2") ioObj.WriteString(":sour:arb:volt:squ:top:time 0.2") ioObj.WriteString(":sour:arb:volt:squ:end:time 0.2") ' Set voltage measurement ' 16 ioObj.WriteString(":sens:func \"volt\"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate triggers ' 21 ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 14	Sets the square wave output voltage and the time parameters. See Figure 1-2 for the relation between the commands and the waveform.
16 to 19	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
22 to 23	Sets the transient trigger. Source output will be triggered once.
24 to 26	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Square Wave Output

```

Catch ex As Exception                                     '28
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             '33

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                           '36

Try ' Retrieve measurement result                         '38
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
33	Enables the channel. And starts source output (0 V with the default setting).
36	Starts the square wave output and monitor.
38 to 40	Reads the measurement result data.

Measurement Result Example

```

Result:
+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+4.154000E+00,+5.
000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+0
0,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.0000
00E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5
.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+
00,+8.440000E-01,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000
000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+
0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E
+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00
0000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+
0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.
000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+0
0,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0000
00E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.
000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00

```


Trapezoidal Wave Output

A program example of trapezoidal wave output is shown in [Table 2-8](#). This example is used to apply trapezoidal wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the trapezoidal wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects trapezoidal wave output	<code>[[:SOUR[c]]:ARB:FUNC TRAP</code>
Sets trapezoidal wave start level	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:STAR <i>level</i></code>
Sets trapezoidal wave top level	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:TOP <i>level</i></code>
Sets trapezoidal wave start time	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:STAR:TIME <i>time</i></code>
Sets trapezoidal wave rise time	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:RTIM <i>value</i></code>
Sets trapezoidal wave top time	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:TOP:TIME <i>time</i></code>
Sets trapezoidal wave fall time	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:FTIM <i>value</i></code>
Sets trapezoidal wave end time	<code>[[:SOUR[c]]:ARB:v-or-c:TRAP:END:TIME <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>

Programming Examples Trapezoidal Wave Output

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

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-8 Trapezoidal Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/TrapezoidalWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set trapezoidal wave voltage output ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func trap") ioObj.WriteString(":sour:arb:volt:trap:star 0") ioObj.WriteString(":sour:arb:volt:trap:top 5") ioObj.WriteString(":sour:arb:volt:trap:star:time 0.2") ioObj.WriteString(":sour:arb:volt:trap:rtime 0.2") ioObj.WriteString(":sour:arb:volt:trap:top:time 0.2") ioObj.WriteString(":sour:arb:volt:trap:ftim 0.2") ioObj.WriteString(":sour:arb:volt:trap:end:time 0.2") ' Set voltage measurement ' 18 ioObj.WriteString(":sens:func \"volt\"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate triggers ' 23 ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 16	Sets the trapezoidal wave output voltage and the time parameters. See Figure 1-2 for the relation between the commands and the waveform.
18 to 21	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
24 to 25	Sets the transient trigger. Source output will be triggered once.
26 to 28	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Trapezoidal Wave Output

```

Catch ex As Exception                                     ' 30
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             ' 35

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                           ' 38

Try ' Retrieve measurement result                         ' 40
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
35	Enables the channel. And starts source output (0 V with the default setting).
38	Starts the trapezoidal wave output and monitor.
40 to 42	Reads the measurement result data.

Measurement Result Example

Result:

```

+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+1.
730000E-01,+4.230000E-01,+6.730000E-01,+9.230000E-01,+1.173000E+0
0,+1.423000E+00,+1.673000E+00,+1.923000E+00,+2.173000E+00,+2.4230
00E+00,+2.673000E+00,+2.923000E+00,+3.173000E+00,+3.423000E+00,+3
.673000E+00,+3.923000E+00,+4.173000E+00,+4.423000E+00,+4.672000E+
00,+4.922000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000
000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+
5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E
+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.000000E+00,+5.00
0000E+00,+5.000000E+00,+5.000000E+00,+4.828000E+00,+4.578000E+00,
+4.328000E+00,+4.078000E+00,+3.828000E+00,+3.578000E+00,+3.328000
E+00,+3.078000E+00,+2.828000E+00,+2.578000E+00,+2.328000E+00,+2.0
78000E+00,+1.828000E+00,+1.578000E+00,+1.328000E+00,+1.078000E+00
,+8.280000E-01,+5.780000E-01,+3.290000E-01,+7.900000E-02,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.
000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+0
0,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0000
00E+00,+0.000000E+00,+0.000000E+00

```

Triangle Wave Output

A program example of triangle wave output is shown in [Table 2-9](#). This example is used to apply triangle wave voltage and monitor the output voltage.

The following commands are used to apply and monitor the triangle wave.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects triangle wave output	<code>[[:SOUR[c]]:ARB:FUNC TRI</code>
Sets triangle wave start level	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:STAR <i>level</i></code>
Sets triangle wave top level	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:TOP <i>level</i></code>
Sets triangle wave start time	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:STAR:TIME <i>time</i></code>
Sets triangle wave rise time	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:RTIM <i>value</i></code>
Sets triangle wave fall time	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:FTIM <i>value</i></code>
Sets triangle wave end time	<code>[[:SOUR[c]]:ARB:v-or-c:TRI:END:TIME <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:<i>func</i>2:APER <i>time</i></code>
	<code>:SENS[c]:<i>func</i>2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>

Programming Examples

Triangle Wave Output

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

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-9 Triangle Wave Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/TriangleWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set triangle wave voltage output ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func tri") ioObj.WriteString(":sour:arb:volt:tri:star 0") ioObj.WriteString(":sour:arb:volt:tri:top 5") ioObj.WriteString(":sour:arb:volt:tri:star:time 0.2") ioObj.WriteString(":sour:arb:volt:tri:rtim 0.2") ioObj.WriteString(":sour:arb:volt:tri:ftim 0.2") ioObj.WriteString(":sour:arb:volt:tri:end:time 0.2") ' Set voltage measurement ' 17 ioObj.WriteString(":sens:func \"volt\"") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate triggers ' 22 ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 15	Sets the triangle wave output voltage and the time parameters. See Figure 1-2 for the relation between the commands and the waveform.
17 to 20	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
23 to 24	Sets the transient trigger. Source output will be triggered once.
25 to 27	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

Triangle Wave Output

```

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

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

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")

Try ' Retrieve measurement result
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
34	Enables the channel. And starts source output (0 V with the default setting).
37	Starts the triangle wave output and monitor.
39 to 41	Reads the measurement result data.

Measurement Result Example

Result:

```

+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+2.
280000E-01,+4.780000E-01,+7.280000E-01,+9.780000E-01,+1.228000E+0
0,+1.478000E+00,+1.728000E+00,+1.977000E+00,+2.227000E+00,+2.4770
00E+00,+2.727000E+00,+2.977000E+00,+3.227000E+00,+3.477000E+00,+3
.727000E+00,+3.977000E+00,+4.227000E+00,+4.477000E+00,+4.727000E+
00,+4.977000E+00,+4.773000E+00,+4.523000E+00,+4.273000E+00,+4.023
000E+00,+3.773000E+00,+3.523000E+00,+3.273000E+00,+3.024000E+00,+
2.774000E+00,+2.524000E+00,+2.274000E+00,+2.024000E+00,+1.774000E
+00,+1.524000E+00,+1.274000E+00,+1.024000E+00,+7.740000E-01,+5.24
0000E-01,+2.740000E-01,+2.400000E-02,+0.000000E+00,+0.000000E+00,+
0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.
000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+0
0,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0000
00E+00,+0.000000E+00,+0.000000E+00

```


User Defined Waveform Output

A program example of user defined waveform output is shown in [Table 2-10](#). This example is used to apply voltage and monitor the output voltage.

The following commands are used to apply and monitor the user defined waveform.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Selects arbitrary waveform output	<code>[[:SOUR[c]]:v-or-c:MODE ARB</code>
Selects user defined waveform output	<code>[[:SOUR[c]]:ARB:FUNC UDEF</code>
Sets data list used to create an user defined waveform	<code>[[:SOUR[c]]:ARB:v-or-c:UDEF <i>list</i></code>
Adds data to the list	<code>[[:SOUR[c]]:ARB:v-or-c:UDEF:APP <i>list</i></code>
Gets the number of data in the list	<code>[[:SOUR[c]]:ARB:v-or-c:UDEF:POIN <i>value</i></code>
Sets interval between each list data	<code>[[:SOUR[c]]:ARB:v-or-c:UDEF:TIME <i>interval</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:<i>func</i>2:APER <i>time</i></code>
	<code>:SENS[c]:<i>func</i>2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>
Enables/disables channel	<code>:OUTP[c] <ON OFF></code>
Initiates specified action	<code>:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]</code>
Reads result data (array data)	<code>:FETC:ARR:<i>type</i>? [<i>chanlist</i>]</code>

Programming Examples

User Defined Waveform Output

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

list is data list used to create an user defined waveform. It must be a comma separated values, such as 0 , 1 , 0 , -1 , 0. This example contains five data.

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-10 User Defined Waveform Output Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/UserDefinedWaveform1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set user defined waveform voltage output ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode arb") ioObj.WriteString(":sour:arb:func udef") ioObj.WriteString(":sour:arb:volt:udef 0,1,0,-1,0") ioObj.WriteString(":sour:arb:volt:udef:time 0.2") ' 6 ' Set voltage measurement ioObj.WriteString(":sens:func " & "volt") ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' 13 ' Generate triggers ioObj.WriteString(":trig:tran:coun 1") ioObj.WriteString(":trig:tran:sour aint") ioObj.WriteString(":trig:acq:coun 100") ioObj.WriteString(":trig:acq:sour timer") ioObj.WriteString(":trig:acq:tim 0.01") ' 18 </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 11	<p>Sets the the user defined waveform voltage output.</p> <p>The <code>:sour:arb:volt:udef</code> command sets the list of the output levels separated by a comma.</p> <p>The <code>:sour:arb:volt:udef:time</code> command sets the interval for changing the output level.</p>
13 to 16	Sets the voltage measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
19 to 20	Sets the transient trigger. Source output will be triggered once.
21 to 23	Sets the acquire trigger. Output monitor will be triggered 100 times in 10 ms interval.

Programming Examples

User Defined Waveform Output

```

Catch ex As Exception                                     ' 25
    Console.WriteLine("An error occurred: " + ex.Message)
End Try

' Turn on output switch
ioObj.WriteString(":outp on")                             ' 30

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")                           ' 33

Try ' Retrieve measurement result                         ' 35
    ioObj.WriteString(":fetc:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
30	Enables the channel. And starts source output (0 V with the default setting).
33	Starts the user defined waveform output and monitor.
35 to 37	Reads the measurement result data.

Measurement Result Example

Result:

```

+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000
E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0
00000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00
,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+6.780000E-02,+1.
000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+0
0,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1.0000
00E+00,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1
.000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+00,+1.000000E+
00,+9.331000E-01,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000
000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+
0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E
+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.00
0000E+00,+0.000000E+00,-6.580000E-02,-1.000000E+00,-1.000000E+00,-
1.000000E+00,-1.000000E+00,-1.000000E+00,-1.000000E+00,-1.000000
E+00,-1.000000E+00,-1.000000E+00,-1.000000E+00,-1.000000E+00,-1.0
00000E+00,-1.000000E+00,-1.000000E+00,-1.000000E+00,-1.000000E+00
,-1.000000E+00,-1.000000E+00,-1.000000E+00,-9.353000E-01,+0.00000
0E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.
000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+0
0,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.000000E+00,+0.0000
00E+00,+0.000000E+00,+0.000000E+00

```

Staircase Sweep Output

A program example of staircase sweep measurements is shown in [Table 2-11](#). This example is used to apply sweep voltage and measure current at each sweep step.

Staircase sweep measurements can be performed by using the following commands.

Function	Command
Selects source function	<code>[[:SOUR[<i>c</i>]]:FUNC:MODE <i>v-or-c</i></code>
Sets sweep output	<code>[[:SOUR[<i>c</i>]]:v-or-c:MODE SWE</code>
Sets output range when starting sweep	<code>[[:SOUR[<i>c</i>]]:v-or-c:RANG <i>value</i></code>
Sets source output value	<code>[[:SOUR[<i>c</i>]]:v-or-c <i>value</i></code>
Sets sweep start or stop value	<code>[[:SOUR[<i>c</i>]]:v-or-c:<STAR STOP> <i>value</i></code>
Sets sweep center or span value	<code>[[:SOUR[<i>c</i>]]:v-or-c:<CENT SPAN> <i>value</i></code>
Sets sweep step value	<code>[[:SOUR[<i>c</i>]]:v-or-c:STEP <i>value</i></code>
Sets number of sweep steps	<code>[[:SOUR[<i>c</i>]]:v-or-c:POIN <i>value</i></code>
	<code>[[:SOUR[<i>c</i>]]:SWE:POIN <i>value</i></code>
Selects sweep source ranging mode	<code>[[:SOUR[<i>c</i>]]:SWE:RANG <BEST FIX AUTO></code>
Selects sweep direction	<code>[[:SOUR[<i>c</i>]]:SWE:DIR <UP DOWN></code>
Selects sweep linear or log	<code>[[:SOUR[<i>c</i>]]:SWE:SPAC <LIN LOG></code>
Selects sweep single or double	<code>[[:SOUR[<i>c</i>]]:SWE:STA <SING DOUB></code>
Sets measurement function	<code>:SENS[<i>c</i>]:FUNC "<i>func</i>", "<i>func</i>", "<i>func</i>"</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[<i>c</i>]:func2:APER <i>time</i></code>
	<code>:SENS[<i>c</i>]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[<i>c</i>]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[<i>c</i>]:<ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>

Programming Examples Staircase Sweep Output

Function	Command
Sets interval of timer trigger	:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i>
Sets trigger count	:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables channel	:OUTP[c] <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]
Reads result data (array data)	:FETC:ARR? [<i>chanlist</i>]
	:FETC:ARR: <i>type</i> ? [<i>chanlist</i>]

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Measurement Result Example

Result: +0.000000E+00,+2.000000E-05,+4.000000E-05,+6.000000E-05,+9.000000E-05

Table 2-11 Staircase Sweep Measurement Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/StaircaseSweep1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage output from 0 V to 0.1 V, 5 steps ' 6 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") ' Set auto-range current measurement ioObj.WriteString(":sens:func ""curr""") ' 14 ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate 5 triggers by automatic internal algorithm ioObj.WriteString(":trig:sour aint") ' 19 ioObj.WriteString(":trig:coun 5") Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try ' Turn on output switch ioObj.WriteString(":outp on") ' 27 ' Initiate transition and acquire ioObj.WriteString(":init (@1)") ' 30 Try ' Retrieve measurement result ' 32 ioObj.WriteString(":fetc:arr:curr? (@1)") s = ioObj.ReadString() Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try End Sub </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 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 16	Sets the current measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A. Auto range is ON with the default setting.

Programming Examples

Staircase Sweep Output

```

Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
    filename = "C:/temp/StaircaseSweep1.txt" '2

    ioObj.WriteString("*RST") ' Reset

    Try ' Set voltage output from 0 V to 0.1 V, 5 steps '6
        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")

        ' Set auto-range current measurement
        ioObj.WriteString(":sens:func "curr"") '14
        ioObj.WriteString(":sens:curr:nplc 0.1")
        ioObj.WriteString(":sens:curr:prot 0.1")

        ' Generate 5 triggers by automatic internal algorithm
        ioObj.WriteString(":trig:sour aint") '19
        ioObj.WriteString(":trig:coun 5")

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

    ' Turn on output switch
    ioObj.WriteString(":outp on") '27

    ' Initiate transition and acquire
    ioObj.WriteString(":init (@1)") '30

    Try ' Retrieve measurement result '32
        ioObj.WriteString(":fetc:arr:curr? (@1)")
        s = ioObj.ReadString()

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

```

Line	Description
19 to 20	Sets the trigger source to AINT (automatic trigger). And sets the trigger count to 5 to perform a 5-step staircase sweep measurement.
27	Enables the channel. And starts source output (0 V with the default setting).
30	Starts staircase sweep measurement.
32 to 34	Reads the measurement result data.

Pulsed Sweep Output

A program example of pulsed sweep measurements is shown in [Table 2-12](#). This example is used to apply pulsed sweep voltage and measure current at each sweep step.

Pulsed sweep measurements can be performed by using the following commands.

Function	Command
Selects source function	[:SOUR[c]]:FUNC:MODE <i>v-or-c</i>
Sets pulse output	[:SOUR[c]]:FUNC[:SHAP] PULS
Sets sweep output	[:SOUR[c]]:v-or-c:MODE SWE
Sets output range when starting sweep	[:SOUR[c]]:v-or-c:RANG <i>value</i>
Sets source output value	[:SOUR[c]]:v-or-c <i>value</i>
Sets sweep start or stop value	[:SOUR[c]]:v-or-c:<STAR STOP> <i>value</i>
Sets sweep center or span value	[:SOUR[c]]:v-or-c:<CENT SPAN> <i>value</i>
Sets sweep step value	[:SOUR[c]]:v-or-c:STEP <i>value</i>
Sets number of sweep steps	[:SOUR[c]]:v-or-c:POIN <i>value</i>
	[:SOUR[c]]:SWE:POIN <i>value</i>
Sets pulse delay time	[:SOUR[c]]:PULS:DEL <i>time</i>
Sets pulse width	[:SOUR[c]]:PULS:WIDT <i>time</i>
Selects sweep source ranging mode	[:SOUR[c]]:SWE:RANG <BEST FIX AUTO>
Selects sweep direction	[:SOUR[c]]:SWE:DIR <UP DOWN>
Selects sweep linear or log	[:SOUR[c]]:SWE:SPAC <LIN LOG>
Selects sweep single or double	[:SOUR[c]]:SWE:STA <SING DOUB>
Sets measurement function	:SENS[c]:FUNC " <i>func</i> ", " <i>func</i> ", " <i>func</i> "

Programming Examples Pulsed Sweep Output

Function	Command
Sets aperture time in seconds or by using NPLC value	:SENS[c]:func2:APER <i>time</i>
	:SENS[c]:func2:NPLC <i>value</i>
Sets limit (compliance) value	:SENS[c]:v-or-c:PROT <i>value</i>
Selects trigger source	:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i>
Sets interval of timer trigger	:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i>
Sets trigger count	:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables channel	:OUTP[c] <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]
Reads result data (array data)	:FETC:ARR? [<i>chanlist</i>]
	:FETC:ARR:~type? [<i>chanlist</i>]

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-12 Pulsed Sweep Measurement Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/StaircasePulsedSweep1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage output from 0 V to 0.1 V, 5 steps ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:func:shap puls") 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") ' Set delay/width to 500 us/1 ms ioObj.WriteString(":sour:puls:del 0.5e-3") ioObj.WriteString(":sour:puls:widt 1.0e-3") ' 15 ' Set 100 mA fixed-range current measurement ioObj.WriteString(":sens:func ""curr""") ioObj.WriteString(":sens:curr:aper 1e-4") ioObj.WriteString(":sens:curr:prot 0.1") ' 19 ' Adjust trigger timing parameters ioObj.WriteString(":trig:tran:del 1.5e-3") ioObj.WriteString(":trig:acq:del 2.9e-3") ' 24 </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 12	Sets the voltage pulse sweep output function. And sets the sweep output from 0 to 0.1 V in 0.02 V step (5 points).
15 to 16	Sets the pulse delay time and the pulse width.
19 to 21	Sets the current measurement function and the 100 mA fixed range measurement. And sets the aperture time to 0.1 ms and the current limit (compliance) value to 0.1 A.
24 to 25	Sets the transient (source) delay time and the acquire (measurement) delay time.

Programming Examples

Pulsed Sweep Output

```

' Generate 5 triggers in 4 ms period
ioObj.WriteString(":trig:sour tim")
ioObj.WriteString(":trig:tim 4e-3")
ioObj.WriteString(":trig:coun 5")

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

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

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")

Try ' Retrieve measurement result
    ioObj.WriteString(":fetc:arr:curr? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
28 to 30	Sets the timer trigger source. And sets the trigger interval to 4 ms, and the trigger count to 5 to perform a 5-step pulsed sweep measurement.
37	Enables the channel. And starts source output (0 V with the default setting).
40	Starts pulsed sweep measurement.
42 to 44	Reads the measurement result data.

Measurement Result Example

Result: +0.000000E+00,+2.000000E-05,+4.000000E-05,+6.000000E-05,+9.000000E-05

List Sweep Output

A program example of list sweep measurements is shown in [Table 2-13](#). This example is used to apply sweep voltage and measure current at each sweep step.

List sweep measurements can be performed by using the following commands.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Sets list sweep output	<code>[[:SOUR[c]]:v-or-c:MODE LIST</code>
Sets source output range	<code>[[:SOUR[c]]:v-or-c:RANG:AUTO <ON OFF></code>
	<code>[[:SOUR[c]]:v-or-c:RANG <i>value</i></code>
Sets source output value	<code>[[:SOUR[c]]:v-or-c <i>value</i></code>
Sets list sweep output values	<code>[[:SOUR[c]]:LIST:v-or-c <i>values</i></code>
Adds list sweep output values to the end of the present setting	<code>[[:SOUR[c]]:LIST:v-or-c:APP <i>values</i></code>
Specifies the list sweep start point	<code>[[:SOUR[c]]:LIST:v-or-c:STAR <i>start_index</i></code>
Asks the number of sweep points	<code>[[:SOUR[c]]:LIST:v-or-c:POIN?</code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>", "<i>func</i>", "<i>func</i>"</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:func2:APER <i>time</i></code>
	<code>:SENS[c]:func2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:v-or-c:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>
Sets interval of timer trigger	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:TIM <i>time</i></code>
Sets trigger count	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:COUN <i>value</i></code>
Sets trigger delay time	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:DEL <i>time</i></code>

Programming Examples
List Sweep Output

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

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-13 **List Sweep Measurement Example**

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/ListSweep1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage output to 0.03, 0.06, and 0.1 V ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:volt:mode list") ioObj.WriteString(":sour:list:volt 0.03,0.06,0.1") ' Set auto-range current measurement ioObj.WriteString(":sens:func ""curr""") ' 12 ioObj.WriteString(":sens:curr:nplc 0.1") ioObj.WriteString(":sens:curr:prot 0.1") ' Generate 3 triggers by automatic internal algorithm ioObj.WriteString(":trig:sour aint") ' 17 ioObj.WriteString(":trig:coun 3") Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try ' Turn on output switch ioObj.WriteString(":outp on") ' 25 ' Initiate transition and acquire ioObj.WriteString(":init (@1)") ' 28 Try ' Retrieve measurement result ' 30 ioObj.WriteString(":fetc:arr:curr? (@1)") s = ioObj.ReadString() Catch ex As Exception Console.WriteLine("An error occurred: " + ex.Message) End Try End Sub </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 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 14	Sets the current measurement function. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A. Auto range is ON with the default setting.

Programming Examples

List Sweep Output

```

Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
    filename = "C:/temp/ListSweep1.txt" '2

    ioObj.WriteString("*RST") ' Reset

    Try ' Set voltage output to 0.03, 0.06, and 0.1 V '6
        ioObj.WriteString(":sour:func:mode volt")
        ioObj.WriteString(":sour:volt:mode list")
        ioObj.WriteString(":sour:list:volt 0.03,0.06,0.1")

        ' Set auto-range current measurement '12
        ioObj.WriteString(":sens:func ""curr""")
        ioObj.WriteString(":sens:curr:nplc 0.1")
        ioObj.WriteString(":sens:curr:prot 0.1")

        ' Generate 3 triggers by automatic internal algorithm
        ioObj.WriteString(":trig:sour aint") '17
        ioObj.WriteString(":trig:coun 3")

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

    ' Turn on output switch
    ioObj.WriteString(":outp on") '25

    ' Initiate transition and acquire
    ioObj.WriteString(":init (@1)") '28

    Try ' Retrieve measurement result '30
        ioObj.WriteString(":fetc:arr:curr? (@1)")
        s = ioObj.ReadString()

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

```

Line	Description
17 to 18	Sets the trigger source to AINT (automatic trigger). And sets the trigger count to 3 to perform a 3-point list sweep measurement.
25	Enables the channel. And starts source output (0 V with the default setting).
28	Starts list sweep measurement.
30 to 32	Reads the measurement result data.

Measurement Result Example

Result: +2.000000E-05,+5.000000E-05,+9.000000E-05

Pulsed List Sweep Output

A program example of pulsed list sweep measurements is shown in [Table 2-14](#). This example is used to apply pulsed sweep voltage and measure current at each sweep step.

Pulsed list sweep measurements can be performed by using the following commands.

Function	Command
Selects source function	<code>[[:SOUR[c]]:FUNC:MODE <i>v-or-c</i></code>
Sets pulse output	<code>[[:SOUR[c]]:FUNC[:SHAP] PULS</code>
Sets list sweep output	<code>[[:SOUR[c]]:<i>v-or-c</i>:MODE LIST</code>
Sets source output range	<code>[[:SOUR[c]]:<i>v-or-c</i>:RANG:AUTO <ON OFF></code>
	<code>[[:SOUR[c]]:<i>v-or-c</i>:RANG <i>value</i></code>
Sets source output value	<code>[[:SOUR[c]]:<i>v-or-c</i> <i>value</i></code>
Sets list sweep output values	<code>[[:SOUR[c]]:LIST:<i>v-or-c</i> <i>values</i></code>
Adds list sweep output values to the end of the present setting	<code>[[:SOUR[c]]:LIST:<i>v-or-c</i>:APP <i>values</i></code>
Specifies the list sweep start point	<code>[[:SOUR[c]]:LIST:<i>v-or-c</i>:STAR <i>start_index</i></code>
Asks the number of sweep points	<code>[[:SOUR[c]]:LIST:<i>v-or-c</i>:POIN?</code>
Sets pulse delay time	<code>[[:SOUR[c]]:PULS:DEL <i>time</i></code>
Sets pulse width	<code>[[:SOUR[c]]:PULS:WIDT <i>time</i></code>
Sets measurement function	<code>:SENS[c]:FUNC "<i>func</i>"[, "<i>func</i>"[, "<i>func</i>"]]</code>
Sets aperture time in seconds or by using NPLC value	<code>:SENS[c]:<i>func</i>2:APER <i>time</i></code>
	<code>:SENS[c]:<i>func</i>2:NPLC <i>value</i></code>
Sets limit (compliance) value	<code>:SENS[c]:<i>v-or-c</i>:PROT <i>value</i></code>
Selects trigger source	<code>:TRIG[c]<:ACQ :TRAN [:ALL]>:SOUR <i>source</i></code>

Function	Command
Sets interval of timer trigger	:TRIG[<i>c</i>]<:ACQ :TRAN [:ALL]>:TIM <i>time</i>
Sets trigger count	:TRIG[<i>c</i>]<:ACQ :TRAN [:ALL]>:COUN <i>value</i>
Sets trigger delay time	:TRIG[<i>c</i>]<:ACQ :TRAN [:ALL]>:DEL <i>time</i>
Enables/disables channel	:OUTP[<i>c</i>] <ON OFF>
Initiates specified action	:INIT<:ACQ :TRAN [:ALL]> [<i>chanlist</i>]
Reads result data (array data)	:FETC:ARR? [<i>chanlist</i>]
	:FETC:ARR: <i>type</i> ? [<i>chanlist</i>]

v-or-c is VOLT for voltage source or limit (compliance), or CURR for current source or limit (compliance).

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

func2 is VOLT for voltage measurement or CURR for current 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 14), 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.

chanlist is (@1) for selecting the channel 1 only, (@2) for selecting the channel 2 only, or (@1,2), (@1:2), (@2,1), or (@2:1) for selecting both channels 1 and 2. Abbreviating this parameter sets *chanlist*=(@1) for the 1-channel models, and *chanlist*=(@1,2) for the 2-channel models.

Table 2-14 Pulsed List Sweep Measurement Example

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/ListPulsedSweep1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Set voltage output to 0.03, 0.06, and 0.1 V ' 6 ioObj.WriteString(":sour:func:mode volt") ioObj.WriteString(":sour:func:shap puls") ioObj.WriteString(":sour:volt:mode list") ioObj.WriteString(":sour:list:volt 0.03,0.06,0.1") ' Set delay/width to 500 us/1 ms ioObj.WriteString(":sour:puls:del 0.5e-3") ' 13 ioObj.WriteString(":sour:puls:widt 1.0e-3") ' Set 100 mA fixed-range current measurement ioObj.WriteString(":sens:func "curr"") ' 17 ioObj.WriteString(":sens:curr:aper 1e-4") ioObj.WriteString(":sens:curr:prot 0.1") ' Adjust trigger timing parameters ioObj.WriteString(":trig:tran:del 1.5e-3") ' 22 ioObj.WriteString(":trig:acq:del 2.9e-3") </pre>	
Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2961A/B2962A.
6 to 10	Sets the voltage pulse list sweep output function. And sets the pulsed list sweep output 0.03 V, 0.06 V, and 0.1 V (3 points). Pulse base value is 0 V with the default setting.
13 to 14	Sets the pulse delay time and the pulse width.
17 to 19	Sets the current measurement function and the 100 mA fixed range measurement. And sets the aperture time to 0.1 ms and the current limit (compliance) value to 0.1 A.
22 to 23	Sets the transient (source) delay time and the acquire (measurement) delay time.

Programming Examples

Pulsed List Sweep Output

```

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

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

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

' Initiate transition and acquire
ioObj.WriteString(":init (@1)")

Try ' Retrieve measurement result
    ioObj.WriteString(":fetc:arr:curr? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
26 to 28	Sets the timer trigger source. And sets the trigger interval to 4 ms, and the trigger count to 3 to perform a 3-point pulsed list sweep measurement.
35	Enables the channel. And starts source output (0 V with the default setting).
38	Starts pulsed list sweep measurement.
40 to 42	Reads the measurement result data.

Measurement Result Example

Result: +2.000000E-05,+5.000000E-05,+9.000000E-05

Using Program Memory

A program example for using program memory is shown in [Table 2-15](#). This example is used to store a program in the program memory and execute it.

Program memory can be set and controlled by using the following 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.

Programming Examples Using Program Memory

Table 2-15 **Example to Use Program Memory**

<pre> Sub B2960control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String, ByRef filename As String) filename = "C:/temp/ProgramMemory1.txt" ' 2 ioObj.WriteString("*RST") ' Reset Try ' Build program ' 6 Dim program As String = "" program = ":sour:func:mode curr\n" program += ":sour:curr:mode swe\n" program += ":sour:curr:star 0.0\n" program += ":sour:curr:stop 40e-3\n" program += ":sour:curr:poin 21\n" program += ":sens:func "volt"\n" program += ":sens:curr:nplc 0.1\n" program += ":arm:coun 1\n" program += ":trig:coun 21\n" program += ":outp 1\n" program += ":init (@1)\n" ' Get program length Dim sProgramLength As String = String.Format("{0:#}", program.Length) ' 21 ioObj.WriteString(":prog:name "sample"") ' 23 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 B2961A/B2962A.
6 to 18	Enters program code to the “program” variable. The program is for performing current source voltage measure sweep measurement from 0 A to 40 mA, 21 points, with the aperture time 0.1 PLC.
21	Gets the program length (number of characters in the “program” variable).
23 to 24	Stores the program code to the program memory as the program name “sample”.

```

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

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

Try ' Retrieve measurement result
    ioObj.WriteString(":fetch:arr:volt? (@1)")
    s = ioObj.ReadString()

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

```

Line	Description
31	Executes the memory program.
34 to 37	Waits for operation complete. And write “*OPC?: 1” on the console window when the operation is completed.
39 to 41	Reads the measurement result data.

Measurement Result Example

Result: +5.200000E-03,+6.643000E-01,+1.931000E+00,+2.000000E+00,+
2.000000E+00,+2.000000E+00,+2.000000E+00,+2.000000E+00,+2.000000E
+00,+2.000000E+00,+2.000000E+00,+2.000000E+00,+2.000000E+00,+2.00
0000E+00,+2.000000E+00,+2.000000E+00,+2.000000E+00,+2.000000E+00,
+2.000000E+00,+2.000000E+00,+2.000000E+00

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.

For performing a staircase sweep measurement, replace the program code from lines 32 to 38 shown in [Table 2-11](#) with the code shown in [Table 2-16](#).

Data output format can be controlled by using the following 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: 0,0,0.022718,41600.025,  
2E-05,0.025817,41600.05,4E-05,0.02878,41600.075,6E-05,0.031722,41  
600.1,9E-05,0.034668,4160
```


Table 2-16 **Example to Read Binary Data**

<pre> ' Select measure data elements ioObj.WriteString(":form:elem:sens volt,curr,time,stat") '2 ' Retrieve measurement result & Output measurement result(Ascii format) '4 ioObj.WriteString(":form asc") ioObj.WriteString(":fetch:arr? (@1)") 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) '27 Console.WriteLine("REAL64 format") Console.WriteLine(value) ioObj.WriteString(":form real,64") ioObj.WriteString(":fetch:arr? (@1)") 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() </pre>	
Line	Description
2	Specifies the data to return. This example selects voltage measurement data, current measurement data, time data, and status data.
4 to 23	Reads the measurement result data in the ASCII format.
27 to 40	Reads the measurement result data in the REAL,64 format.

Programming Examples
Reading Binary Data