Agilent B2900 Series
Precision Source/Measure Unit

Programming Guide

Agilent Technologies

Notices

Manual Part Number

B2910-90020

Edition

Edition 1, May 2011
Edition 2, July 2011

Agilent Technologies, Inc.
5301 Stevens Creek Blvd
Santa Clara, CA 95051 USA

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, Agilent 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. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent 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.

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.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

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.agilent.com/find/GPLV2

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 Agilent B2900 manual feedback form.

We respect your privacy. Be assured that Agilent will never sell or rent your information. Nor will Agilent 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 Agilent Technologies B2900 by using an external computer, and consists of the following chapters.

- "Controlling the Agilent B2900"

  Describes how to control the B2900 on a task basis.

- "Programming Examples"

  Introduces example programs for controlling the B2900.

See *Agilent B2900 User's Guide* for information about the B2900 itself.

Refer to *Agilent B2900 SCPI Command Reference* for the SCPI messages and conventions, data output format, error code, and the details on Agilent B2900 SCPI commands.

Contents

Contents

Contents

Contents

1 Controlling the Agilent B2900

This chapter describes basic information to control the Agilent B2900, and consists of the following sections.

- “Before Starting”
- “Controlling Various Functions”
- “Controlling the Source Output”
- “Controlling the Measurement Function”
- “Using the Math Function”
- “Performing the Limit Test”
- “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” can be used as a parameter.
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>{,<B>} 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.

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

Connecting to the Interface

Agilent B2900 supports GPIB, LAN, and USB interfaces. All three interfaces are live at power-on. Select the interface used for controlling the B2900. Connect your interface cable to the appropriate interface connector.

For the information on configuring the interfaces, see *Agilent B2900 Series User's Guide*.

Starting the Instrument Control

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

```
Sub Main()
  Dim rm As Ivi.Visa.Interop.ResourceManager
  Dim ioObj As Ivi.Visa.Interop.FormattedIO488
  Dim address As String = "enter address of your instrument"

  rm = New Ivi.Visa.Interop.ResourceManager
  ioObj = New Ivi.Visa.Interop.FormattedIO488
  ioObj.IO = rm.Open(address)
      ' insert your code for instrument control
  ioObj.IO.Close()
End Sub
```

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

- For using the GPIB interface

  The `address` value is the VISA GPIB Connect String displayed on the GPIB Configuration dialog box opened by pressing the 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 2011,1,1")   'Y,M,D
ioObj.WriteString(":SYST:TIME 23,59,59")   'H,M,S
```

Performing the Self-Test

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

Example

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

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

Performing the Self-Calibration

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

Example

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

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

Setting the Operations at Power On

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

Example

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

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

Reading an Error Message

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

Example

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

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

Clearing the Error Buffer

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

Example

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

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

Reading Timestamp

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

Example

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

Clearing Timestamp

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

Example

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

Setting the Automatic Clear of Timestamp

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

Example

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

Confirming the Firmware Revision

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

Example

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

The returned value will be as follows.

`Agilent 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 Agilent B2900.

- “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 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 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”

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 1-2.

Example

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

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
```

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 1-2.

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 Sweep Operation

For the variety of sweep output operation, see Figure 1-1.

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

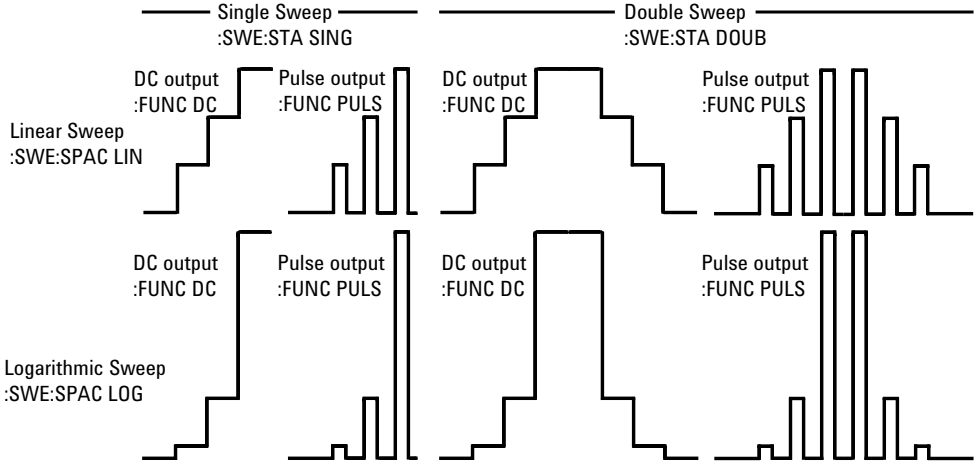
Sweep mode, single or double is set by the :SOUR:SWE:STA command.

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 1-1 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 1-3.

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 1-4.

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.

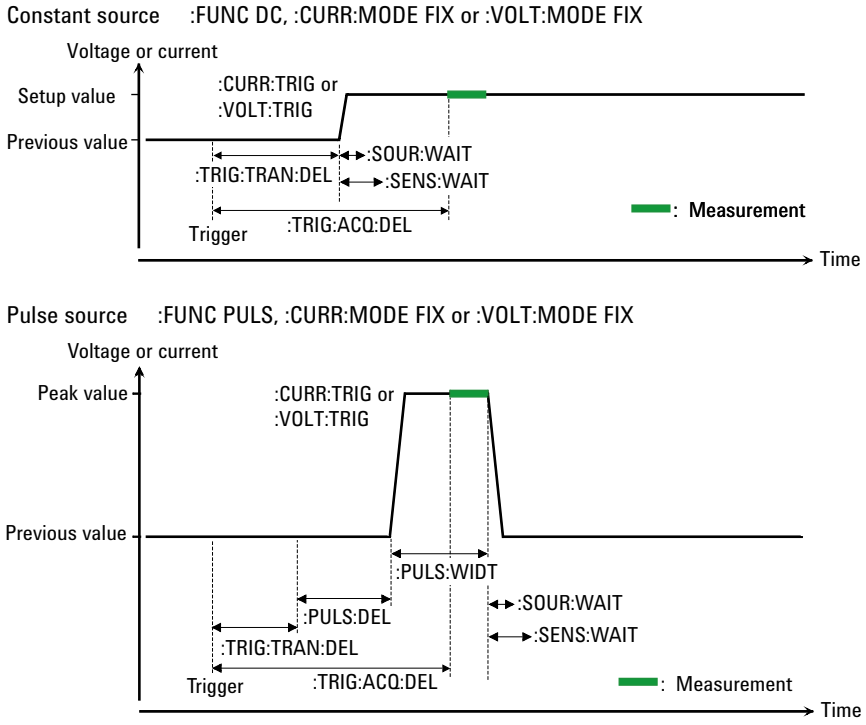
Setting the Source Output Trigger

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 1-2.

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 1-2 To Perform DC and Pulse Output and Spot Measurement



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

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 1-3 and 1-4 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 command, the :OUTP:FILT:AUTO command, and the :OUTP:FILT:<FREQ|TCON> command.

Example

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

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")
```

Controlling the Measurement Function

This section describes how to control the measurement function of Agilent B2900.

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""")
```

Setting the Resistance Measurement Mode

Resistance measurement mode, manual or auto is set by the :SENS:RES:MODE command.

Example

```
ioObj.WriteString(":SENS:RES:MODE AUTO")      'Auto mode
ioObj.WriteString(":SENS:RES:MODE MAN")       'Manual mode
```

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 1-2 for the spot measurement.

Example

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

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

Setting the Measurement Speed

Measurement speed is set by the :SENS:<CURR|VOLT|RES>:APER or :SENS:<CURR|VOLT|RES>: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 Range

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

For current measurement and voltage measurement, the maximum measurement range depends on the limit/compliance value.

Example

```
ioObj.WriteString(":SENS:CURR:RANG:AUTO OFF")
ioObj.WriteString(":SENS:CURR:RANG 0.1")          '100 mA fixed

ioObj.WriteString(":SENS:RES:RANG:AUTO ON")
ioObj.WriteString(":SENS:RES:RANG:AUTO:LLIM 2")  '2 ohm limit
ioObj.WriteString(":SENS:RES:RANG:AUTO:ULIM 200") '200 ohm limit
```

Setting the Measurement Auto Range Operation

Automatic measurement ranging operation mode, normal, resolution, or speed is set by the :SENS:<CURR|VOLT>:RANG:AUTO:MODE command. And the threshold rate for the operation is set by the :SENS:<CURR|VOLT>:RANG:AUTO:THR command.

Example

```
ioObj.WriteString(":SENS:CURR:RANG:AUTO:MODE NORM") 'Normal
ioObj.WriteString(":SENS:CURR:RANG:AUTO:THR 80")

ioObj.WriteString(":SENS:CURR:RANG:AUTO:MODE RES")  'Resolution
ioObj.WriteString(":SENS:CURR:RANG:AUTO:THR 80")

ioObj.WriteString(":SENS:CURR:RANG:AUTO:MODE SPE")  'Speed
ioObj.WriteString(":SENS:CURR:RANG:AUTO:THR 80")
```

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 1-2, 1-3, and 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: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 1-3 To Perform Staircase Sweep Output and Measurement

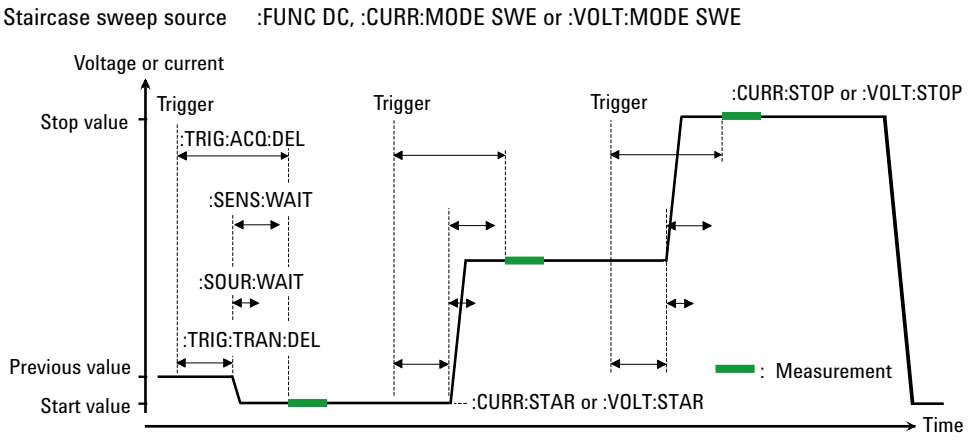
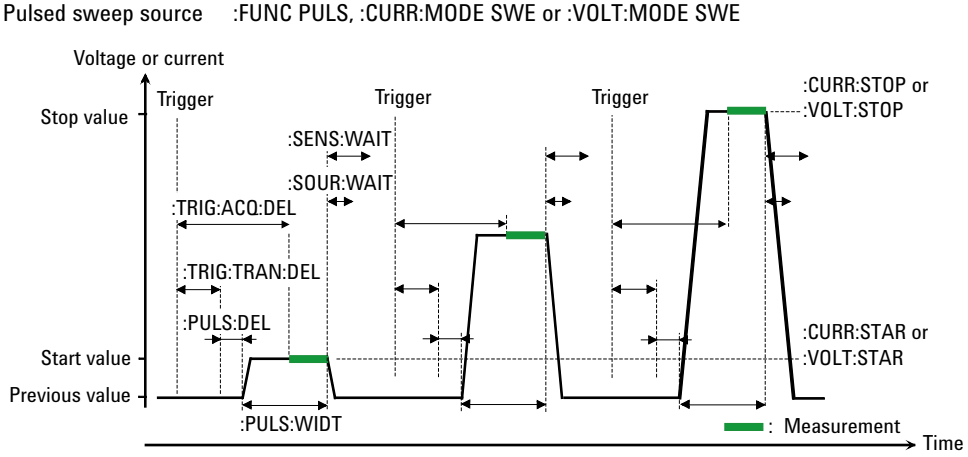


Figure 1-4 To Perform Pulsed Sweep Output and Measurement



Setting the Measurement Wait Time

Measurement wait time is set by the :SENS:WAIT commands. See Figures 1-3 and 1-4 for the wait time.

Example

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

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

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

Performing Sweep Measurement

Staircase sweep measurement is performed as shown below.

1. Set the staircase sweep source and the required source functions. For details, see “Controlling the Source Output” on page 1-10.
2. Set the required measurement functions. For details, see previous topics in this section.
3. Set the trigger condition. See “Setting the Source Output Trigger” on page 1-15 and “Setting the Measurement Trigger” on page 1-20.
4. Enable the channel. See “Enabling the Measurement Channel” on page 1-18.

   The channel starts output set by the :SOUR:<CURR|VOLT> command.
5. Execute the :INIT command to start measurement.

For the programming example, see “Staircase Sweep Measurements” on page 2-13.

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

Stopping Measurement

Measurement is stopped by the :OUTP OFF command.

Example

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

Using the Math Function

This section describes how to use the math function.

- “Defining a Mass Expression”
- “Deleting an User Defined Mass Expression”
- “Enabling or Disabling the Mass Function”
- “Reading Mass Result Data”

Defining a Mass Expression

Mass 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 Mass Expression

Mass expression is deleted by the :CALC:MATH[:EXPR]:DEL commands. The commands do not delete the predefined mass expression.

Example

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

Enabling or Disabling the Mass Function

Mass function is set by the :CALC:MATH:STAT command.

Example

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

Reading Mass Result Data

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

Example

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

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

Performing the Limit Test

Limit test is performed as shown below.

1. Set the required source condition. For details, see “Controlling the Source Output” on page 1-10.
2. Set the required measurement condition. For details, see “Controlling the Measurement Function” on page 1-18.
3. Set the composite limit test. See “Setting the Composite Limit Test” on page 1-25.
4. Set the individual limit tests. See “Setting Individual Limit Tests” on page 1-25.
5. Set the trigger condition. See “Setting the Source Output Trigger” on page 1-15 and “Setting the Measurement Trigger” on page 1-20.
6. Enable the channel. See “Enabling the Measurement Channel” on page 1-18.

   The channel starts output set by the :SOUR:<CURR|VOLT> command.
7. Execute the :INIT command to start limit test.

   Note that the DC source channel applies the value set by the :SOUR:<CURR|VOLT>:TRIG command when it is triggered, even if the channel is applying the value set by the :SOUR:<CURR|VOLT> command.
8. Read limit test result. See “Reading Limit Test Result” on page 1-25.

For the programming example, see “Pass/Fail Judgement and Math Function” on page 2-33.

Setting the Composite Limit Test

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

Example

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

NOTE

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

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

Setting Individual Limit Tests

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

Example

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

Reading Limit Test Result

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

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

Example

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

NOTE

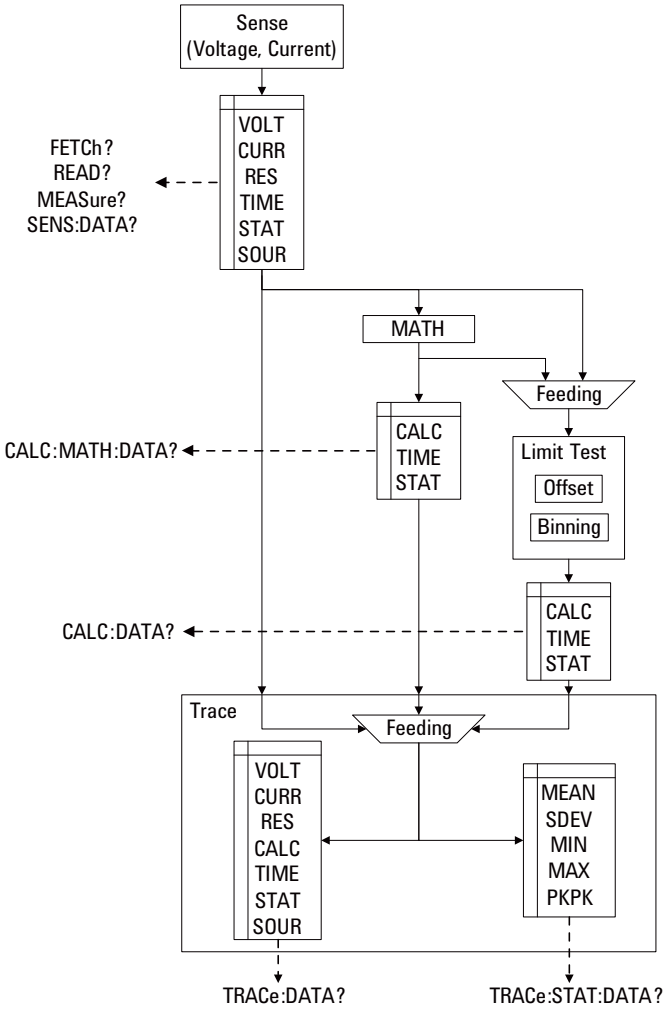
To specify the data to obtain, use the :FORM:ELEM:CALC command.

Using the Trace Buffer

This section describes how to use the trace buffer.

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

Figure 1-5 **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")
```

2 Programming Examples

This chapter provides the following sections which explain programming example.

- “Preparations”
- “Spot Measurements”
- “Pulsed Spot Measurements”
- “Staircase Sweep Measurements”
- “Pulsed Sweep Measurements”
- “List Sweep Measurements”
- “Pulsed List Sweep Measurements”
- “Sampling Measurements”
- “Pass/Fail Judgement and Math Function”
- “Using Program Memory”
- “Reading Binary Data”
- “Performing MOSFET Id-Vd Measurements”

NOTE

About Numeric Suffix

Command header may be accompanied by a numeric suffix *c* for specifying the SMU (source/measure unit) 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 B2900control subprogram in the template.

NOTE

To Start Program

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

Preparations

This section provides the basic information for programming of the automatic measurement using the Agilent B2900, Agilent IO Libraries, and Microsoft Visual Basic .NET.

- "To Create Your Project Template"
- "To Create Measurement Program"

NOTE To execute the example programs in this chapter, you need to install Agilent GPIB interface, Agilent 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 Agilent B2900 (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. \test\my_temp).

To Create Measurement Program

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

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

- Measurement devices

  Discrete, packaged, on-wafer, and so on.

- Parameters/characteristics to be measured

  h_{FE}, Vth, sheet resistance, and so on.

- Measurement method

  Spot measurement, staircase sweep measurement, and so on.

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

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

Step 4. Launch Visual Basic .NET.

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

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

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

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

Table 2-1 Example Template Program Code

```
Module Module1

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

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

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

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

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

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

End Module
```

Line	Description
1 to 8	Beginning of the Main subprogram. And defines the variables used in this program.
9 to 20	Establishes the connection with the B2900 specified by the GPIB address ifAddress=23 on the interface GPIB0.

```
Module Module1

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

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

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

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

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

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

End Module
```

Line	Description
21 to 23	Calls the B2900control subprogram. And displays the result data in a console window.
25 to 27	Saves the result data to a file specified by filename.
29 to 35	Breaks the connection with the B2900.
37 to 39	B2900control subprogram. Measurement program code should be entered here.

Spot Measurements

A program example of spot measurements is shown in Table 2-2. This example is used to apply voltage and measure current.

Spot measurements can be performed by using the following commands.

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

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.

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 2-2 Spot Measurement Example

```
Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
  filename = "C:/temp/FixedDc1.txt"                                          ’2

  ioObj.WriteString("*RST") ’ Reset

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

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

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

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

  Try ’ Initiate measurement and retrieve measurement result                 ’23
    ioObj.WriteString(":meas:curr? (@1)")
    s = ioObj.ReadString()

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

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
6 to 8	Sets the voltage source function. And sets the source value to 0.1 V.
11 to 14	Sets the current measurement function and the auto range measurement. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
21	Enables the channel. And starts source output.
23 to 25	Reads the measurement result data.

Measurement Result Example

```
Result: +3.034160E-02
```

Pulsed Spot Measurements

A program example of pulsed spot measurements is shown in Table 2-3. This example is used to apply pulsed voltage and measure current three times.

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

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

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

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.

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 Pulsed Spot Measurement Example

```
Sub B2900control(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:widt 1.0e-3")

    ’ Set 100 mA fixed-range current measurement
    ioObj.WriteString(":sens:func ""curr""")                                   ’19
    ioObj.WriteString(":sens:curr:rang:auto off")
    ioObj.WriteString(":sens:curr:rang 100e-3")
    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")                                 ’26
    ioObj.WriteString(":trig:acq:del 2.9e-3")
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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 23	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.
26 to 27	Sets the transient (source) delay time and the acquire (measurement) delay time.

```
    ' Generate 3 triggers in 4 ms period
    ioObj.WriteString(":trig:sour tim")                                    '30
    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")                                            '39

  ' Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                          '42

  Try ' Retrieve measurement result                                        '44
    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
30 to 32	Sets the timer trigger source. And sets the trigger interval to 4 ms, and the trigger count to 3. The B2900 will perform the pulsed spot measurement three times.
39	Enables the channel. And starts source output.
42	Starts pulse output and pulsed spot measurement.
44 to 46	Reads the measurement result data.

Measurement Result Example

```
Result: +3.033990E-02,+3.034060E-02,+3.034120E-02
```

Staircase Sweep Measurements

A program example of staircase sweep measurements is shown in Table 2-4. 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	[:SOUR[*c*]]:FUNC:MODE *v-or-c*
Sets sweep output	[:SOUR[*c*]]:*v-or-c*:MODE SWE
Sets output range when starting sweep	[:SOUR[*c*]]:*v-or-c*:RANG *value*
Sets source output value	[:SOUR[*c*]]:*v-or-c value*
Sets sweep start or stop value	[:SOUR[*c*]]:*v-or-c*:<STAR | STOP> *value*
Sets sweep center or span value	[:SOUR[*c*]]:*v-or-c*:<CENT | SPAN> *value*
Sets sweep step value	[:SOUR[*c*]]:*v-or-c*:STEP *value*
Sets number of sweep steps	[:SOUR[*c*]]:*v-or-c*:POIN *value*
	[:SOUR[*c*]]:SWE:POIN *value*
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 “*func*”[, “*func*”[, “*func*”]]
Sets measurement range	:SENS[*c*]:*func*:RANG:AUTO <ON | OFF>
	:SENS[*c*]:*func*:RANG *value*
Sets aperture time in seconds or by using NPLC value	:SENS[*c*]:*func*:APER *time*
	:SENS[*c*]:*func*:NPLC *value*

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

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.

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

Table 2-4 Staircase Sweep Measurement Example

```
Sub B2900control(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
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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.

```
Sub B2900control(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 Measurements

A program example of pulsed sweep measurements is shown in Table 2-5. 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 *v-or-c*
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 *value*
Sets source output value	[:SOUR[*c*]]:*v-or-c value*
Sets sweep start or stop value	[:SOUR[*c*]]:*v-or-c*:<STAR | STOP> *value*
Sets sweep center or span value	[:SOUR[*c*]]:*v-or-c*:<CENT | SPAN> *value*
Sets sweep step value	[:SOUR[*c*]]:*v-or-c*:STEP *value*
Sets number of sweep steps	[:SOUR[*c*]]:*v-or-c*:POIN *value*
	[:SOUR[*c*]]:SWE:POIN *value*
Sets pulse delay time	[:SOUR[*c*]]:PULS:DEL *time*
Sets pulse width	[:SOUR[*c*]]:PULS:WIDT *time*
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 “*func*”[, “*func*”[, “*func*”]]

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

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.

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 Pulsed Sweep Measurement Example

```
Sub B2900control(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")                                ’15
    ioObj.WriteString(":sour:puls:widt 1.0e-3")

    ’ Set 100 mA fixed-range current measurement
    ioObj.WriteString(":sens:func ""curr""")                                  ’19
    ioObj.WriteString(":sens:curr:rang:auto off")
    ioObj.WriteString(":sens:curr:rang 100e-3")
    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")                                ’26
    ioObj.WriteString(":trig:acq:del 2.9e-3")
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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 23	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.
26 to 27	Sets the transient (source) delay time and the acquire (measurement) delay time.

```
    ' Generate 5 triggers in 4 ms period
    ioObj.WriteString(":trig:sour tim")                                    '30
    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")                                            '39

  ' Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                          '42

  Try ' Retrieve measurement result                                        '44
    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
30 to 32	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.
39	Enables the channel. And starts source output (0 V with the default setting).
42	Starts pulsed sweep measurement.
44 to 46	Reads the measurement result data.

Measurement Result Example

```
Result: +1.520000E-05,+7.599400E-03,+1.519220E-02,+2.276880E-02,+
3.035860E-02
```

List Sweep Measurements

A program example of list sweep measurements is shown in Table 2-6. 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	[:SOUR[*c*]]:FUNC:MODE *v-or-c*
Sets list sweep output	[:SOUR[*c*]]:*v-or-c*:MODE LIST
Sets source output range	[:SOUR[*c*]]:*v-or-c*:RANG:AUTO <ON | OFF>
	[:SOUR[*c*]]:*v-or-c*:RANG *value*
Sets source output value	[:SOUR[*c*]]:*v-or-c value*
Sets list sweep output values	[:SOUR[*c*]]:LIST:*v-or-c values*
Adds list sweep output values to the end of the present setting	[:SOUR[*c*]]:LIST:*v-or-c*:APP *values*
Specifies the list sweep start point	[:SOUR[*c*]]:LIST:*v-or-c*:STAR *start_index*
Asks the number of sweep points	[:SOUR[*c*]]:LIST:*v-or-c*:POIN?
Sets measurement function	:SENS[*c*]:FUNC "*func*"[, "*func*"[, "*func*"]]
Sets measurement range	:SENS[*c*]:*func*:RANG:AUTO <ON | OFF>
	:SENS[*c*]:*func*:RANG *value*
Sets aperture time in seconds or by using NPLC value	:SENS[*c*]:*func*:APER *time*
	:SENS[*c*]:*func*:NPLC *value*
Sets limit (compliance) value	:SENS[*c*]:*v-or-c*:PROT *value*
Selects trigger source	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:SOUR *source*
Sets interval of timer trigger	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:TIM *time*

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

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.

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 List Sweep Measurement Example

```
Sub B2900control(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
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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.

```
Sub B2900control(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
```

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

Pulsed List Sweep Measurements

A program example of pulsed list sweep measurements is shown in Table 2-7. 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	[:SOUR[*c*]]:FUNC:MODE *v-or-c*
Sets pulse output	[:SOUR[*c*]]:FUNC[:SHAP] PULS
Sets list sweep output	[:SOUR[*c*]]:*v-or-c*:MODE LIST
Sets source output range	[:SOUR[*c*]]:*v-or-c*:RANG:AUTO <ON | OFF>
	[:SOUR[*c*]]:*v-or-c*:RANG *value*
Sets source output value	[:SOUR[*c*]]:*v-or-c value*
Sets list sweep output values	[:SOUR[*c*]]:LIST:*v-or-c values*
Adds list sweep output values to the end of the present setting	[:SOUR[*c*]]:LIST:*v-or-c*:APP *values*
Specifies the list sweep start point	[:SOUR[*c*]]:LIST:*v-or-c*:STAR *start_index*
Asks the number of sweep points	[:SOUR[*c*]]:LIST:*v-or-c*:POIN?
Sets pulse delay time	[:SOUR[*c*]]:PULS:DEL *time*
Sets pulse width	[:SOUR[*c*]]:PULS:WIDT *time*
Sets measurement function	:SENS[*c*]:FUNC “*func*”[, “*func*”[, “*func*”]]
Sets measurement range	:SENS[*c*]:*func*:RANG:AUTO <ON | OFF>
	:SENS[*c*]:*func*:RANG *value*
Sets aperture time in seconds or by using NPLC value	:SENS[*c*]:*func*:APER *time*
	:SENS[*c*]:*func*:NPLC *value*

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

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.

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 Pulsed List Sweep Measurement Example

```
Sub B2900control(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:rang:auto off")
    ioObj.WriteString(":sens:curr:rang 100e-3")
    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")
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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 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.

```
    ' Generate 3 triggers in 4 ms period
    ioObj.WriteString(":trig:sour tim")                                    '28
    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")                                            '37

  ' Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                          '40

  Try ' Retrieve measurement result                                        '42
    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 to perform a 3-point pulsed list sweep measurement.
37	Enables the channel. And starts source output (0 V with the default setting).
40	Starts pulsed list sweep measurement.
42 to 44	Reads the measurement result data.

Measurement Result Example

```
Result: +9.123000E-03,+1.822410E-02,+3.036380E-02
```

Sampling Measurements

A program example of sampling measurements is shown in Table 2-8. This example is used to apply constant voltage and measure current six times.

Sampling measurements can be performed by using the following commands.

Function	Command
Selects source function	[:SOUR[*c*]]:FUNC:MODE *v-or-c*
Sets source output range	[:SOUR[*c*]]:*v-or-c*:RANG:AUTO <ON | OFF>
	[:SOUR[*c*]]:*v-or-c*:RANG *value*
Sets source output value	[:SOUR[*c*]]:*v-or-c value*
Sets triggered source output value	[:SOUR[*c*]]:*v-or-c*:TRIG *value*
Sets measurement function	:SENS[*c*]:FUNC "*func*"[, "*func*"[, "*func*"]]
Sets measurement range	:SENS[*c*]:*func*:RANG:AUTO <ON | OFF>
	:SENS[*c*]:*func*:RANG *value*
Sets aperture time in seconds or by using NPLC value	:SENS[*c*]:*func*:APER *time*
	:SENS[*c*]:*func*:NPLC *value*
Sets limit (compliance) value	:SENS[*c*]:*v-or-c*:PROT *value*
Selects trigger source	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:SOUR *source*
Sets interval of timer trigger	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:TIM *time*
Sets trigger count	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:COUN *value*
Sets trigger delay time	:TRIG[*c*]<:ACQ | :TRAN | [:ALL]>:DEL *time*
Enables/disables channel	:OUTP[*c*] <ON | OFF>

Function	Command
Initiates specified action	:INIT<:ACQ | :TRAN | [:ALL]> [*chanlist*]
Reads result data (array data)	:FETC:ARR? [*chanlist*]
	:FETC:ARR:*type*? [*chanlist*]

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.

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 Sampling Measurement Example

```
Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
  filename = "C:/temp/Sampling1.txt"                                          ’2

  ioObj.WriteString("*RST") ’ Reset

  Try ’ Set voltage output                                                    ’6
    ioObj.WriteString(":sour:func:mode volt")
    ’ Set triggered voltage to 0.1 V
    ioObj.WriteString(":sour:volt:trig 0.1")
    ’ Set auto-range current measurement
    ioObj.WriteString(":sens:func ""curr""")                                  ’11
    ioObj.WriteString(":sens:curr:nplc 0.1")
    ioObj.WriteString(":sens:curr:prot 0.1")
    ’ Adjust trigger timing parameters
    ioObj.WriteString(":trig:acq:del 2.0e-3")                                 ’15
    ’ Generate 6 triggers in 4 ms period
    ioObj.WriteString(":trig:sour tim")                                       ’17
    ioObj.WriteString(":trig:tim 4e-3")
    ioObj.WriteString(":trig:coun 6")

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

  ’ Turn on output switch
  ioObj.WriteString(":outp on")                                               ’26

  ’ Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                             ’29

  Try ’ Retrieve measurement result                                           ’31
    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
2	Defines the file name used for saving the result data.
4	Resets the B2900.
6 to 9	Sets the voltage output function. And sets the bias voltage to 0.1 V.
11 to 13	Sets the current measurement function and the auto range measurement. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 0.1 A.
15	Sets the acquire (measurement) delay time.

```
Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
  filename = "C:/temp/Sampling1.txt"                                          ’2

  ioObj.WriteString("*RST") ’ Reset

  Try ’ Set voltage output                                                    ’6
    ioObj.WriteString(":sour:func:mode volt")
    ’ Set triggered voltage to 0.1 V
    ioObj.WriteString(":sour:volt:trig 0.1")
    ’ Set auto-range current measurement
    ioObj.WriteString(":sens:func ""curr""")                                  ’11
    ioObj.WriteString(":sens:curr:nplc 0.1")
    ioObj.WriteString(":sens:curr:prot 0.1")
    ’ Adjust trigger timing parameters
    ioObj.WriteString(":trig:acq:del 2.0e-3")                                 ’15
    ’ Generate 6 triggers in 4 ms period
    ioObj.WriteString(":trig:sour tim")                                       ’17
    ioObj.WriteString(":trig:tim 4e-3")
    ioObj.WriteString(":trig:coun 6")

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

  ’ Turn on output switch
  ioObj.WriteString(":outp on")                                               ’26

  ’ Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                             ’29

  Try ’ Retrieve measurement result                                           ’31
    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 19	Sets the timer trigger source. And sets the timer interval to 4 ms, and the trigger count to 6 to perform a 6-point sampling measurement.
26	Enables the channel. And starts source output (0 V with the default setting).
29	Starts sampling measurement.
31 to 33	Reads the measurement result data.

Measurement Result Example

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

Pass/Fail Judgement and Math Function

A program example of pass/fail judgements is shown in Table 2-9. This example is used to perform power vs voltage sweep measurement and performs pass/fail judgement. Power is calculated by using the built-in math function.

Math function can be set by using the following commands.

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

Pass/fail judgements can be performed by using the following commands.

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

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

Table 2-9 Pass/Fail Judgement Example

```
Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
  filename = "C:/temp/Judgement1.txt"                                          ’2

  ioObj.WriteString("*RST") ’ Reset

  Try ’ Set voltage sweep from 0.0 V to 0.1 V, 201 linear steps                ’6
    ioObj.WriteString(":sour:func:mode volt")
    ioObj.WriteString(":sour:volt:mode swe")
    ioObj.WriteString(":sour:volt:star 0.0")
    ioObj.WriteString(":sour:volt:stop 0.1")
    ioObj.WriteString(":sour:volt:poin 201")
    ioObj.WriteString(":sour:swe:spac lin")
    ioObj.WriteString(":sour:swe:sta sing")

    ’ Set current limit to 100 mA
    ioObj.WriteString(":sens:curr:prot 0.1")                                   ’16

    ’ Enable predefined math expression power (voltage * current)
    ioObj.WriteString(":calc:math:name ""POWER""")                             ’19
    ioObj.WriteString(":calc:math:stat on")

    ’ Set limit test mode to "Grading"
    ioObj.WriteString(":calc:clim:mode grad")                                  ’23
    ioObj.WriteString(":calc:clim:upd end")
    ioObj.WriteString(":calc:clim:stat on")

    ’ Feed math result and judge it is in +/-5 mW or not
    ioObj.WriteString(":calc:feed math")                                       ’28
    ioObj.WriteString(":calc:lim1:stat on")
    ioObj.WriteString(":calc:lim1:func lim")
    ioObj.WriteString(":calc:lim1:low -5.0e-3")
    ioObj.WriteString(":calc:lim1:upp +5.0e-3")
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
6 to 13	Sets the voltage sweep output function. And sets the sweep output from 0 to 0.1 V, 201 points. Also sets the linear increment and single sweep.
16	Sets the current limit (compliance) value to 0.1 A.
19 to 20	Enables POWER math function.
23 to 25	Enables composite limit test of grading mode.
28 to 32	Enables limit test 1 which judges if POWER is between -5 mW and +5 mW.

```
    ' Set arm count to 1, trigger count to 201
    ioObj.WriteString(":arm:coun 1")                                    '35
    ioObj.WriteString(":trig:coun 201")

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

  ' Turn on output switch
  ioObj.WriteString(":outp on")                                         '43

  ' Initiate transition and acquire
  ioObj.WriteString(":init (@1)")                                       '46

  ' Wait for operation complete
  ioObj.WriteString("*OPC?")                                            '49
  Dim d As String = ioObj.ReadString()
  Console.Write("*OPC?: " + d)
  Console.WriteLine()

  Try ' Retrieve measurement result (not bin numbers)                   '54
    ioObj.WriteString(":calc:data?")
    s = ioObj.ReadString()

    ioObj.WriteString(":calc:lim1:fail?")                               '58
    d = ioObj.ReadString()
    If d = 0 Then
      Console.WriteLine("PASS")
    Else
      Console.WriteLine("FAIL")
    End If
    Console.WriteLine()

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

Line	Description
35 to 36	Sets the arm count to 1, and the trigger count to 201 to perform a 201-step staircase sweep measurement.
43	Enables the channel. And starts source output (0 V with the default setting).
46	Starts staircase sweep measurement.
49 to 52	Waits for operation complete. And write “*OPC?: 1” on the console window when the operation is completed.
54 to 56	Reads the measurement result data.
58 to 65	Reads the judgement result. And write “PASS” or “FAIL” on the console window.

Measurement Result Example

```
Result: -2.341916E-10,+8.687035E-08,+3.241116E-07,+7.134239E-07,+
1.258200E-06,+1.947869E-06,+2.790977E-06,+3.793277E-06,+4.929483E
-06,+6.228402E-06,+7.677595E-06,+9.279485E-06,+1.103210E-05,+1.29
3916E-05,+1.499512E-05,+1.721203E-05,+1.957131E-05,+2.207869E-05,
+2.467268E-05,+2.755153E-05,+3.052130E-05,+3.363800E-05,+3.691760
E-05,+4.033103E-05,+4.390036E-05,+4.762392E-05,+5.151573E-05,+5.5
53805E-05,+5.972507E-05,+6.405285E-05,+6.851899E-05,+7.315973E-05
,+7.794760E-05,+8.288631E-05,+8.796509E-05,+9.320610E-05,+9.86047
9E-05,+1.041754E-04,+1.098613E-04,+1.157115E-04,+1.215744E-04,+1.
278577E-04,+1.341738E-04,+1.406063E-04,+1.472060E-04,+1.539637E-0
4,+1.608716E-04,+1.679319E-04,+1.751681E-04,+1.825232E-04,+1.9004
83E-04,+1.977296E-04,+2.055358E-04,+2.135138E-04,+2.216522E-04,+2
.298863E-04,+2.383193E-04,+2.468891E-04,+2.556029E-04,+2.645003E-
04,+2.735255E-04,+2.826991E-04,+2.919260E-04,+3.015675E-04,+3.112
219E-04,+3.209985E-04,+3.309458E-04,+3.410295E-04,+3.512634E-04,+
3.616291E-04,+3.720819E-04,+3.827504E-04,+3.936007E-04,+4.045787E
-04,+4.157298E-04,+4.270301E-04,+4.384635E-04,+4.501420E-04,+4.61
8862E-04,+4.738464E-04,+4.859040E-04,+4.981206E-04,+5.104463E-04,
+5.229651E-04,+5.356430E-04,+5.484076E-04,+5.614119E-04,+5.744722
E-04,+5.877991E-04,+6.011935E-04,+6.148205E-04,+6.285388E-04,+6.4
24530E-04,+6.564842E-04,+6.706918E-04,+6.850603E-04,+6.995474E-04
,+7.141454E-04,+7.289117E-04,+7.438255E-04,+7.589229E-04,+7.74165
5E-04,+7.895723E-04,+8.051391E-04,+8.208203E-04,+8.367055E-04,+8.
526876E-04,+8.688672E-04,+8.852325E-04,+9.016407E-04,+9.182453E-0
4,+9.349851E-04,+9.518914E-04,+9.689457E-04,+9.861478E-04,+1.0035
23E-03,+1.021063E-03,+1.038796E-03,+1.056637E-03,+1.074605E-03,+1
.092708E-03,+1.110964E-03,+1.129434E-03,+1.147996E-03,+1.166662E-
03,+1.185524E-03,+1.204595E-03,+1.223783E-03,+1.243249E-03,+1.262
697E-03,+1.282301E-03,+1.302063E-03,+1.321965E-03,+1.342132E-03,+
1.362347E-03,+1.382697E-03,+1.403252E-03,+1.424084E-03,+1.444900E
-03,+1.465962E-03,+1.487102E-03,+1.508430E-03,+1.529923E-03,+1.55
1453E-03,+1.573165E-03,+1.595088E-03,+1.617097E-03,+1.639334E-03,
+1.661768E-03,+1.684316E-03,+1.707018E-03,+1.729902E-03,+1.752875
E-03,+1.776006E-03,+1.799316E-03,+1.822720E-03,+1.846251E-03,+1.8
69993E-03,+1.893788E-03,+1.917864E-03,+1.941986E-03,+1.966472E-03
,+1.990904E-03,+2.015598E-03,+2.040341E-03,+2.065370E-03,+2.09054
9E-03,+2.115705E-03,+2.141095E-03,+2.166588E-03,+2.192289E-03,+2.
218138E-03,+2.244184E-03,+2.270309E-03,+2.296767E-03,+2.323176E-0
3,+2.349848E-03,+2.376582E-03,+2.403580E-03,+2.430636E-03,+2.4578
34E-03,+2.485222E-03,+2.512773E-03,+2.540283E-03,+2.568170E-03,+2
.596157E-03,+2.624402E-03,+2.652645E-03,+2.681054E-03,+2.709752E-
03,+2.738454E-03,+2.767308E-03,+2.796405E-03,+2.825400E-03,+2.854
952E-03,+2.884357E-03,+2.914004E-03,+2.943753E-03,+2.973740E-03,+
3.003983E-03,+3.034236E-03
```

Using Program Memory

A program example for using program memory is shown in Table 2-10. 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 *program_code*
Adds program code to the end of the memory program [a]	:PROG:APP *program_code*
Sets a value to the variable specified by *n* [b]	:PROG:VAR*n* “*value*”
Executes memory program [a]	:PROG:EXEC
Changes status of memory program [a]	:PROG:STAT *operation*
Blocks other commands until the program execution status changes to Paused or Stopped [a]	:PROG:WAIT? *timeout_in_seconds*
Deletes a memory program [a]	:PROG:DEL
Deletes all memory programs	:PROG:DEL:ALL

a. This function is effective for the memory program previously specified by the :PROG:NAME command.
b. Variables can be used in the memory program. They must be expressed as %*n*% (*n*: integer. 1 to 100) in the memory program.

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

Table 2-10 Example to Use Program Memory

```
Sub B2900control(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
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
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")                                    '31

  ' Wait for operation complete
  ioObj.WriteString("*OPC?")                                             '34
  s = ioObj.ReadString()
  Console.Write("*OPC?: " + s)
  Console.WriteLine()

  Try ' Retrieve measurement result                                      '39
    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: -7.480000E-05,+6.524500E-03,+1.311680E-02,+1.971080E-02,+
2.630190E-02,+3.289570E-02,+3.948990E-02,+4.607940E-02,+5.266580E
-02,+5.926410E-02,+6.585490E-02,+7.244700E-02,+7.903250E-02,+8.56
2770E-02,+9.221940E-02,+9.880730E-02,+1.053949E-01,+1.119873E-01,
+1.185849E-01,+1.251736E-01,+1.317632E-01
```

Reading Binary Data

A program example for reading binary data is shown in Table 2-11. 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 2-4 with the code shown in Table 2-11.

Data output format can be controlled by using the following commands.

Function	Command
Sets the data output format	:FORM[:DATA] *format*
Sets byte order of binary data	:FORM:BORD *byte_order*

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

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

Measurement Result Example

```
Result: V (V), I (A), Time (sec), Status: -5.88E-05,2.85297E-06,0
.021938,21120.0222637,0.006749,0.030997,36480.0499995,0.0151897,0
.037096,41600.0750008,0.022776,0.041071,41600.0999998,0.0303624,0
.045048,4160
```

Table 2-11 Example to Read Binary Data

```
' 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()
```

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.

Performing MOSFET Id-Vd Measurements

Table 2-12 shows a programming example for measuring MOSFET Id-Vd characteristics. This example uses two channels.

The channel 1 performs the primary sweep output to apply the drain voltage, and measures the drain current at each sweep step.

The channel 2 performs the secondary sweep output to apply the gate voltage. The secondary sweep is realized by using the list sweep function.

The channels should be connected to the device as shown below.

Channel	Terminal	Connection
1	High Force	MOSFET drain terminal
	High Sense	
	Low Force	MOSFET source and substrate terminals
	Low Sense	
2	High Force	MOSFET gate terminal
	High Sense	
	Low Force	MOSFET source and substrate terminals
	Low Sense	

Measurement Result Example

```
Result:
-2.102110E-06,+1.149380E-03,+1.827990E-03,+2.150970E-03,+2.272340
E-03,+2.313980E-03,+2.333310E-03,+2.346400E-03,+2.356810E-03,+2.3
65510E-03,+2.373160E-03,-1.332470E-05,+2.465270E-03,+4.501590E-03
,+6.113420E-03,+7.326210E-03,+8.180630E-03,+8.736170E-03,+9.07385
0E-03,+9.283210E-03,+9.427320E-03,+9.536790E-03,-2.731010E-05,+3.
513460E-03,+6.642130E-03,+9.363240E-03,+1.167520E-02,+1.359510E-0
2,+1.513140E-02,+1.631180E-02,+1.717590E-02,+1.778500E-02,+1.8216
90E-02
```

Table 2-12 MOSFET Id-Vd Measurement Example

```
Sub B2900control(ByVal ioObj As Ivi.Visa.Interop.FormattedIO488, ByRef s As String,
ByRef filename As String)
  filename = "C:/temp/MOSFETId-Vd1.txt"                                          '2

  ioObj.WriteString("*RST") ' Reset

  Try ' Set primary sweep 0 V to 3 V, 11 points                                  '6
    ioObj.WriteString(":sour1:func:mode volt")
    ioObj.WriteString(":sour1:volt:mode swe")
    ioObj.WriteString(":sour1:volt:star 0")
    ioObj.WriteString(":sour1:volt:stop 3")
    ioObj.WriteString(":sour1:volt:poin 11")

    ' Set secondary sweep 1 V to 3 V, 3 steps, 33 points
    Dim vg As String = ""                                                        '14
    vg = "1,1,1,1,1,1,1,1,1,1,1,"    ' for 1st staircase sweep measurement
    vg += "2,2,2,2,2,2,2,2,2,2,2,"   ' for 2nd staircase sweep measurement
    vg += "3,3,3,3,3,3,3,3,3,3,3"    ' for 3rd staircase sweep measurement
    ioObj.WriteString(":sour2:func:mode volt")
    ioObj.WriteString(":sour2:volt:mode list")
    ioObj.WriteString(":sour2:list:volt " + vg)

    ' Set auto-range current measurement
    ioObj.WriteString(":sens1:func ""curr""")                                    '23
    ioObj.WriteString(":sens1:curr:nplc 0.1")
    ioObj.WriteString(":sens1:curr:prot 0.1")
    ioObj.WriteString(":sens2:curr:prot 0.01")                                   '26
```

Line	Description
2	Defines the file name used for saving the result data.
4	Resets the B2900.
6 to 11	Sets the channel 1 to the primary sweep source. And sets the sweep output from 0 to 3 V in 0.3 V step (11 points).
14 to 20	Sets the channel 2 to the secondary sweep source. And sets the sweep output from 1 to 3 V in 1 V step (3 steps, 33 points to perform 11-step staircase sweep measurement three times).
23 to 25	Sets the channel 1 measurement function to the current measurement. And sets the aperture time to 0.1 PLC and the current limit (compliance) value to 100 mA. Auto range is ON with the default setting.
26	Sets the channel 2 current limit (compliance) value to 10 mA.

```
    ' Generate 33 triggers by automatic internal algorithm
    ioObj.WriteString(":trig1:sour aint")                                    '29
    ioObj.WriteString(":trig1:coun 33")
    ioObj.WriteString(":trig2:sour aint")
    ioObj.WriteString(":trig2:coun 33")

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

  ' Turn on output switch
  ioObj.WriteString(":outp1 on")                                             '39
  ioObj.WriteString(":outp2 on")

  ' Initiate transition and acquire
  ioObj.WriteString(":init (@1,2)")                                          '43

  Try ' Retrieve measurement result                                          '45
    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
29 to 32	For the channels 1 and 2, sets the trigger source to AINT (automatic trigger). And sets the trigger count to 33 to perform the 11-step staircase sweep measurement three times.
39 to 40	Enables the channels 1 and 2. And starts source output (0 V with the default setting).
43	Starts MOSFET Id-Vd measurement.
45 to 47	Reads the measurement result data.