U9036 Motor Stator Tester Programming Guide

CHANGZHOU EUCOL ELELCTRONIC TECHNOLOGY CO.,LTD.
NO.1,NORTH QINGYANG ROAD,TIANNING DISTRICT,CHANGZHOU
TEL: +86-519-85505199
FAX: +86-519-85505169
WWW.EUCOL.COM.CN
SALES@EUCOL.COM.CN

Programming guide provides guidance for user to program this motor stator tester with existing commands, mainly dealing with notation conventions and definitions, short-form rules of command and parameter, commands introduction and appendix.
You can further program this motor stator tester with the commands mentioned in this guide.

1 Command Introduction

1.1 Notation Conventions and Definitions

：	A colon is used to separate the higher level commands and the lower level commands.
?	A question mark is used to generate a query for the command in front of it.
；	The semicolon can be used as a separator to execute multiple commands on a single line.
*	Asterisk is used to indicate that the command followed is a common command.
，	Comma is used to separate the multi-parameters in the command.
-	White space is used as a separator between a command and a parameter.
<>	Words or characters enclosed in angle brackets symbolize a program code parameter.
[]	Items that enclosed in square brackets are optional.
{}	When several items are enclosed by brace, only one of these elements may be selected.
NR1	Specify integer data (For example: 12)
NR2	Specify fixed-point data (For example: 12.3)
NR3	Specify exponential data in floating point format (For example: 2.000000e-03)
NL	New Line character (ACSII decimal 10) is the end of the input/output string.

Note: behind every command string must be enclosed in NL (ASCII is 10) as command terminator.

1.2 Short-form Rules of Command and Parameter

For memory and writing conveniences to long-form commands or parameters, we will use the following rules to shorten the long-form commands or parameters.
If the length of the command word is four letters or less, no short form version exists.

Example:
TYPE=TYPE
These following rules apply to command words that exceed four letters:

1. If the fourth letter of the command word is a vowel, delete it and all the letters after it.
2. If the fourth letter of the command word is a consonant, retain it but drop all the letters after it.

Examples:
POSition abbreviates to POS.
DISPlay abbreviates to DISP.
If the long-form mnemonic is defined as a phrase rather than a single word, then the long-form mnemonic is the first character of the first word followed by the entire last word. The above rules, when the long-form mnemonic is a single word, are then applied to the resulting long-form mnemonic to obtain the short form.
Example:
Test SETup, whose long form would be TSETup, abbreviates to TSET.

2 Command System

U9036 support the following subsystem commands:

- ◆ Common Commands
- ◆ DISPlay Subsystem Commands
- ◆ SYSTem Subsystem Commands
- ◆ TRIGger Subsystem Commands
- ◆ ABORt Subsystem Commands
- ◆ DCR Subsystem Commands
- ◆ DCR BAL Subsystem Commands
- ◆ IW Subsystem Commands
- ◆ IW BAL Subsystem Commands
- ◆OS Subsystem Commands
- ◆ IR Subsystem Commands
- ◆ HIPOT Subsystem Commands
- ◆ L Subsystem Commands
- ◆ L BAL Subsystem Commands
- ◆ SEQ Subsystem Commands
- ◆ FETCh Subsystem Commands
- ◆ STATistic Subsystem Commands
- ◆ Mass MEMory Subsystem Commands
- ◆ KEY Subsystem Commands

2.1 Common Commands

The common commands defined by the IEEE 488.2-1987 standard are basic commands in instrument command system which can work with other commands as a command set and also can execute special functions independently.
Common commands used in instrument command system are shown as table 2-1-1.

Command	**Query**	**Return Format**
N/A	*IDN?	Eucol Electronic Technology Co., Ltd.,<model>,< serial number>,< software version>
*RST	N/A	N/A
*RCL <value>	N/A	N/A
*SAV <value>	N/A	N/A
*TRG	N/A	N/A
*CLS	N/A	N/A
*ESE <0-255>	*ESE?	Event Status Enable Register
N/A	*ESR?	Event Status Register
*OPC	*OPC?	Returns 1
*SRE <0-255>	*SRE?	Service Request Enable Register
N/A	*STB?	Service Request Register
N/A	*LRN?	Returns instrument settings

Table 2-1-1

1. ***IDN?**

The *IDN ? query returns the instrument information, including company name, instrument model, instrument serial number and software version.

Query Syntax: *IDN?

Return Format: Eucol Electronic Technology Co., Ltd., <model>, <serial number>, <software revision><NL>

Example:

*IDN? Eucol Electronic Technology Co., Ltd., U9036, 903-A13-105, VER1.0 10 070623A

2. ***RST (Reset)**

The *RST command places the instrument in a known state—factory default state.

Query Syntax: *RST

3. ***RCL <value>**

The *RCL **<value>** command restores the state of the instrument from the specified setup file position, <value>= { 1 to 180 }.
Query Syntax: *RCL <value>
Example:
*RCL 1   Restore the state of the instrument from the specified Setup01.

4. ***SAV <value>**

The *SAV command stores the current state of the instrument to the specified setup file position. <value>= { 1 to 180}.
Command Syntax: *SAV <value>[,“name”], name is the file name,the length of name should be less then 20 charactors.
Example:
*SAV 1   Store the current state of the instrument to the specified Setup01.

5. ***TRG**

The *TRG command generates forcible triggering signal. When an acquisition is completed, the instrument is stopped (similar to single+force trig).
Command Syntax: *TRG

6. ***CLS**

The *CLS command clears the status register, output buffer data and the Request-for-OPC flag.
Command Syntax: *CLS

7. ***ESE <0-255>**

*ESE common command sets the bits in the Standard Event Status Enable Register. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A "1" in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register.

ESE (Event Status Enable Register)

PON		CME	EXE		QYE		OPC

Event Descriptions

Bit	**Name**	**Description**	**When Set to 1, Enables**
7	PON	Power on	Event when an OFF to ON transition occurs.
5	CME	Command Error	Event when a command error is detected.
4	EXE	Execution Error	Event when an execution error is detected.
2	QYE	Output data loss	Event when data and command in output buffer
0	OPC	Operation complete	Event when an operation is complete.

Command Syntax: *ESE <0-255>
Query Syntax: *ESE?
Return Format：<NR1><NL>   Return the ESE register value.

8. ***ESR?**

The *ESR? query returns the contents of the Standard Event Status Register. When you read the Event Status Register, the value returned is the total bit weights of all of the bits that are high at the time you read the byte. Reading the register clears the Event Status Register.

ESR（Event Status Register）

Bit	Name	Description	When Set to 1, Indicates:
7	PON	Power on	An OFF to ON transition has occurred.
5	CME	Command error	A command error has been detected.
4	EXE	Execution error	An execution error has been detected.
2	QYE	Output data loss	Output data loss has been detected
0	OPC	Operation complete	Operation is complete.

Query Syntax: *ESR?
Return Format: <NR1><NL>   Return the current status.

9. ***OPC**

The *OPC command places an ASCII "1" in the output queue when all pending device operations have completed.
Command Syntax: *OPC
Query Syntax: *OPC?
Return Format: <1><NL>
Note: The interface hangs until this query returns.

10. ***SRE <0-255>**

The *SRE command sets the bits in the Service Request Enable Register. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A "1" in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A "0" disables the bit.
SRE（Service Request Enable Register）

		ESB	MAV				

Event Descriptions

Bit	**Name**	**Description**	**When Set to 1, Enables:**
5	ESB	Event Status Bit	Interrupts when enabled conditions in the Standard Event Status Register (ESR) occur.
4	MAV	Message Available	Interrupts when messages are in the Output Queue.

Command Syntax: *SRE <0-255>

Query Syntax: *SRE?

Return Format: <NR1><NL>   Return the current value of the Service Request Enable Register.

11. ***STB?**

The*STB? query returns the current value of the instrument's status byte.

Status Byte Register (STB)

	RQS	ESB	MAV				

Event Descriptions

Bit	**Name**	**Description**	**When Set to 1, Indicates:**
6	RQS	Request Service	When polled, indicates that the device is requesting service or not.
5	ESB	Event Status Bit	Indicates that an enabled condition in the Standard Event Status Register (ESR) has occurred.
4	MAV	Message Available	Indicates that there are messages in the Output Queue.

Query Syntax: *STB?

Return Format: <NR1><NL>   Return Service Request Register value.

2.2 *DISPlay Subsystem Commands*

DISPlay commands are used to control the display system.

DISPlay:PAGE

The DISPlay:PAGE command set up the display page of instrument. The DISPlay:PAGE? Command returns the abbreviated page name currently displayed on the LCD screen.

Command Syntax： DISPlay:PAGE <page name>

<page name> as follows:

TEST	set the instrument display page to Test DDisplay page.
IWINding	set the instrument display page to Wave display page.
TSETup	set the instrument display page to Test Setup page
STATistic	set the instrument display page to Statistics page
HISTory	set the instrument display page to History Data page
IWSETup	set the instrument display page to IW(Surge) Setup page
IBSETup	set the instrument display page to IW Banlance Setup page
RSETup	set the instrument display page to DCR Setup page
RBSETup	set the instrument display page to R Balance Setup page
OSETup	set the instrument display page to OS Setup page
IRSETup	set the instrument display page to IR Setup page
HSETup	set the instrument display page to Hipot Setup page
LSETup	set the instrument display page to L Setup page
LBSETup	set the instrument display page to L Balance Setup page
SEQ	set the instrument display page to Sequence Setup page
SCONfig	set the instrument display page to System Config page
TCONfig	set the instrument display page to Test Config page
TCSetup	set the instrument display page to Temp.Corr.Setup page
INFO	set the instrument display page to System Information page
FLISt	set the instrument display page to File list page

Query Command: DISPlay:PAGE?

Return Format： {TEST| IWIN | TSET | STAT | HIST | IWSET | IBSET | RSET | RBSET | OSET | IRSET | HSET | LSET | LBSET | SEQ | SCON | TCON | TCS |INFO | FLIS}<NL>

Note: During the measuring, the query command is ignored.

DISPlay:GRID

The DISPlay:GRID command sets the grid display mode. The DISPlay:GRID? Query returns the current grid display mode.

Command Syntax: DISPlay:GRID { {1 | ON} | {0 | OFF}}

Query Syntax: DISPlay:GRID?

Return format: {{1 | ON} | {0 | OFF}}<NL>

Note： Where,

1 (decimal49) is equal to ON; 0 (decimal48) is equal to OFF.

DISPlay:COROna

The DISPlay:COROna command sets the corona display mode. The DISPlay:COROna? query returns the current corona display mode.

Command Syntax: DISPlay:COROna { {1 | ON} | {0 | OFF}}

Query Syntax: DISPlay:COROna?

Return format: {{1 | ON} | {0 | OFF}}<NL>

DISPlay:ENLarge

The DISPlay:ENLarge command sets the test waveform enlarge display on Wave Display page. The DISPlay:ENLarge? query returns the current status of the test waveform on Wave Display page.

Command Syntax: DISPlay:ENLarge { {1 | ON} | {0 | OFF}}

Query Syntax: DISPlay:ENLarge?

Return format: {{1 | ON} | {0 | OFF}}<NL>

DISPlay:MASK

The DISPlay:MASK command sets the test waveform compare function on Wave Display page. The DISPlay:MASK? query returns the current status of the waveform-compare function on Wave Display page.

Command Syntax: DISPlay:MASK { {1 | ON} | {0 | OFF}}

Query Syntax: DISPlay:MASK?

Return format: {{1 | ON} | {0 | OFF}}<NL>

2.3 SYSTem subsystem commands

The SYSTem subsystem commands are used for system setups.

SYSTem:DATE

The SYSTem:DATE command sets the date of system. The SYSTem:DATE? query returns the current date.

Command Syntax: SYSTem:DATE <year>,<month>,<day> year,month,day is NR1 format.

Where, month can be string format：{JANuary | FEBruary | MARch |APRil |MAY | JUNe | JULy | AUGust | SEPtember | OCTober | NOVember | DECember}

Query Syntax: SYSTem:DATE?

Return format: <NR1>,<NR1>,<NR1><NL>

SYSTem:TIME

The SYSTem:TIME command sets the time of system. The SYSTem:TIME? query returns the current time.

Command Syntax: SYSTem:TIME <hour>,<minute>,<second> hour, minute, second is NR1 format.

Query Syntax: SYSTem:TIME?

Return format: <NR1>,<NR1>,<NR1><NL>

SYSTem:BEEPer:PASS

The SYSTem:BEEPer:PASS command sets the alarm of passed judgment. The SYSTem:BEEPer:PASS? query returns the current status.

Command Syntax: SYSTem:BEEPer:PASS {OFF | LHIGh | LLOW | SSHort | DSHort}

Query Syntax: SYSTem:BEEPer:PASS?

Return format: {OFF | LHIG | LLOW | SSH | DSH}<NL>

SYSTem:BEEPer:FAIL

The SYSTem:BEEPer:FAIL command sets the alarm of fail judgment. The SYSTem:BEEPer:FAIL? query returns the current status.

Command Syntax: SYSTem:BEEPer:FAIL {OFF | LHIGh | LLOW | SSHort | DSHort}

Query Syntax: SYSTem:BEEPer:FAIL?

Return format: {OFF | LHIG | LLOW | SSH | DSH}<NL>

SYSTem:BOFail

The SYSTem:BOFail command sets the stop testing when on Fail test result. The SYSTem:BOFail? query returns the current status.

Command Syntax: SYSTem:BOFail { {1 | ON} | {0 | OFF}}

Query Syntax: SYSTem:BOFail?

Return format: {1 | 0}<NL>

SYSTem:STWave

The SYSTem:STWave command sets display the test wavform of IW(Surge) in the screen when in comprehensive testing. The SYSTem:STWave? query returns the current status.

Command Syntax: SYSTem:STWave { {1 | ON} | {0 | OFF}}

Query Syntax: SYSTem:STWave?

Return format: {1 | 0}<NL>

SYSTem:TDATa:RECord

The SYSTem:TDATa:RECord command sets recording the test results. The SYSTem:TDATa:RECord? query returns the current status.

Command Syntax: SYSTem:TDATa:RECord {OFF | FAIL | PASS | ALL}

where,

OFF	the test results is not be recorded when testing finished
FAIL	record the test result in FAIL only when testing finished
PASS	record the test result in PASS only when testing finished
ALL	record all the test result when testing finished

Query Syntax: SYSTem:TDATa:RECord?

Return format: {OFF | FAIL | PASS | ALL}

SYSTem:TDATa:UPLoad

The SYSTem:TDATa:UPLoad command sets uploading the test results. The SYSTem:TDATa:UPLoad? query returns the current status.

Command Syntax: SYSTem:TDATa:UPLoad {OFF | FAIL | PASS | ALL}

where,

OFF	the test results is not be uploaded when testing finished
FAIL	upload the test result in FAIL only when testing finished
PASS	upload the test result in PASS only when testing finished
ALL	upload all the test result when testing finished

	data block format	Description
DCR	1,1,2,DCR,1.2345ohm,HI	DUT No,CH+,CH-,DCR,value,judgment
DCR Balance	1,12,34,RBAL,0.10mohm,OK	DUT No,RefA,RefB,RBAL,value,judgment
IW(Surge)	1,1,2,IW,,OK	DUT No,CH+,CH-,IW,,judgment
IW Balance	1,12,34,IWBAL,,NG	DUT No,RefA,RefB,IWBAL,,judgment
OS	1,1,2,OS,,OK	DUT No,CH+,CH-,OS,,judgment
IR	1,12,34,IR,1.234Gohm,OK	DUT No,CH+,CH-,IR,value,judgment
HIPOT	1,1,2,HIPOT,1.234mA,OK	DUT No,CH+,CH-,HIPOT,value,judgment
Ls	1,1,2,Ls,1.2345uH,2.2358,OK	DUT No,CH+,CH-,Ls-Q,value, value,judgment
L Balance	1,(1-2),(3-4),LBAL,10.234uH,OK	DUT No,RefA,RefB,LBAL,value,judgment

Query Syntax: SYSTem:TDATa:UPLoad?
Return format: {OFF | FAIL | PASS | ALL}

SYSTem:TCORrection
The SYSTem:TCORrection command sets the temperature correction function. The SYSTem:TCORrection? query returns the current status.
Command Syntax: SYSTem:TCORrection { {1 | ON} | {0 | OFF}}
Query Syntax: SYSTem:TCORrection?
Return format: {1 | 0}<NL>

SYSTem:TCORrection:COEFficient
The SYSTem:TCORrection:COEFficient command sets the temperature coefficient of the material. The SYSTem:TCORrection:COEFficient? query returns the current temperature coefficient of material.
Command Syntax: SYSTem:TCORrection:COEFficient <coef>   NR3 format, from -999 to 999
Query Syntax: SYSTem:TCORrection:COEFficient?
Return format: <NR3><NL>

SYSTem:TCORrection:REFerence
The SYSTem:TCORrection:REFerence command sets the reference temperature. The SYSTem:TCORrection:REFerence? query returns the reference temperature.

Command Syntax: SYSTem:TCORrection:REFerence <ref>   NR3 format, from -999 to 999
Query Syntax: SYSTem:TCORrection:REFerence?
Return format: <NR3><NL>

2.4 *TRIGger subsystem commands*

The TRIGger subsystem command group is used to trigger a measurement or to set the trigger mode.

TRIGger[:IMMediate]
The TRIGger[:IMMediate] command triggers a measurement.
Command Syntax: TRIGger[:IMMediate]
NOTE: The TRIGger[:IMMediate] command, available only in the <MEAS DISP> page, will be ignored when U9036 is in the testing state.

TRIGger:SOURce
The TRIGger:SOURce command sets the trigger mode. The TRIGger:SOURce? query returns the current trigger mode.
Command Syntax: TRIGger:SOURce {MANual | EXTernal | INTernal | BUS}

Where,

MANual	Triggered by pressing the $\boxed{\text{START}}$ button or using foot control switch.
EXTernal	Triggered by the HANDLER interface.
INTernal	Automatically triggered after pressing the $\boxed{\text{START}}$ button.
BUS	Triggered by RS232 interface, USB interface.

Query Syntax: TRIGger:SOURce?
Return format: {MAN | EXT | INT | BUS}<NL>

TRIGger:DELay
The TRIGger:DELay command sets the delay time between two triggers. The TRIGger:DELay? query returns the current delay time. Delay time range from 0 to 60s with step of 1ms.
Command Syntax: TRIGger:DELay <value>   where   <value> is NR3 format.
Query Syntax: TRIGger:DELay?
Return format: <NR3><NL>

2.5 ABORt subsystem command

U9036 will abort the current measurement as soon as the ABORt command is received.
Command Syntax: ABORt

2.6 DCR subsystem commands

The DCR subsystem command group is used to set the DC resistance test parameters.

DCR:STEPSN? Query returns the number of DCR test steps.

DCR:STEP<n>:DELete
The DCR:STEP<n>:DELete command is used to delete the DCR test steps.

DCR:STEP<n>:SET
The DCR:STEP<n>:SET commands set the DCR test parameters.
Command Syntax: DCR:STEP<n>:SET <ch+>,<ch->,<std>,<high>,<low>,<speed>,
<delay>,<dev>,<dut no.>

where

<ch+> and <ch-> NR1 format, from 1 to 12.
<std>,<high> and <low> NR3 format, from 0 to 100k.
<speed> used to set test speed, { SLOW | MEDium | FAST }
<delay> used to set delay time, NR3 format, from 0 to 60s.
<dev> used to set deviation value, NR3 format, from -100k to 100k.
<dut no.> used to set DUT no., NR1 format, from 1 to 6.

Query Syntax: DCR:STEP<n>:SET?
Return format:
<NR1>,<NR1>,<NR3>,<NR3>,<NR3>,<{SLOW|MED|FAST}>,<NR3>,<NR3>,
<NR1><NL>
If the query steps is empty, return format: <0><NL>

DCR:STEP<n>:CHH
The DCR:STEP<n>:CHH command sets CH+. The DCR:STEP<n>:CHH? Query returns CH+.

Command Syntax: DCR:STEP<n>:CHH <ch> where <n> is NR1 format, from 1 to 12
Query Syntax: DCR:STEP<n>:CHH?
Return format: <NR1><NL> 0 meas the step is empty.

DCR:STEP<n>:CHL
The DCR:STEP<n>:CHL command sets CH-. The DCR:STEP<n>:CHL? Query returns CH-.
Command Syntax: DCR:STEP<n>:CHL <ch> where <n> is NR1 format, from 1 to 12
Query Syntax: DCR:STEP<n>:CHL?
Return format: <NR1><NL> 0 meas the step is empty.

DCR:STEP<n>:STD
The DCR:STEP<n>:STD command sets the nominal value. The DCR:STEP<n>:STD? Query returns the nominal value.
Command Syntax: DCR:STEP<n>:STD <value> where <value> is NR3 format, from 0 to 100k.
Query Syntax: DCR:STEP<n>:STD?
Return format: <NR3><NL>

DCR:STEP<n>:HIGH
The DCR:STEP<n>:HIGH command sets the upper limit . The DCR:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: DCR:STEP<n>:HIGH <value> where <value> is NR3 format, from 0 to 100k.
Query Syntax: DCR:STEP<n>:HIGH?
Return format: <NR3><NL>

DCR:STEP<n>:LOW
The DCR:STEP<n>:LOW command sets the lower limit . The DCR:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: DCR:STEP<n>:LOW <value> where <value> is NR3 format, from 0 to 100k.
Query Syntax: DCR:STEP<n>:LOW?
Return format: <NR3><NL>

DCR:STEP<n>:SPEed
The DCR:STEP<n>:SPEed command sets the test speed . The DCR:STEP<n>:SPEed? Query returns the current test speed.
Command Syntax: DCR:STEP<n>:SPEed { SLOW | MEDium | FAST }
Query Syntax: DCR:STEP<n>:SPEed?
Return format: {SLOW | MED | FAST}<NL>

DCR:STEP<n>:DELAY
The DCR:STEP<n>:DELAY command sets the delay time . The DCR:STEP<n>:DELAY? Query returns the delay time.
Command Syntax: DCR:STEP<n>:DELAY <value> where <value> is NR3 format, from 0 to 60s.
Query Syntax: DCR:STEP<n>:DELAY?
Return format: <NR3><NL>

DCR:STEP<n>:DUTNo
The DCR:STEP<n>:DUTNo command sets the number of test object . The DCR:STEP<n>:DUTNo? Query returns the current number of test object.
Command Syntax: DCR:STEP<n>:DUTNo <value> where <value> is NR1 format, from 1 to 6.
Query Syntax: DCR:STEP<n>:DUTNo?
Return format: <NR1><NL>

DCR:STEP<n>:CLEar
The DCR:STEP<n>:CLEar command used to perform clear operation.
Command Syntax: DCR:STEP<n>:CLEar
Note: After success of clear operation, return format:
<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3>
After fail of clear operation, retur format: <FAIL><NL>

DCR:STEP<n>:CLEar:DATA
The DCR:STEP<n>:CLEar:DATA command used to upload the data of clear operation. The DCR:STEP<n>:CLEar:DATA? Query returns the data of clear operation.
Command Syntax: DCR:STEP<n>:CLEar:DATA
<d1>,<d2>,<d3>,<d4>,<d5>,<d6>,<d7>,<d8><NL>
Query Syntax: DCR:STEP<n>:CLEar:DATA?
Return format: <NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR3><NL>

DCR:STEP<n>:MEASure

The DCR:STEP<n>:MEASure command used to triggers a measurement and return the measurement data.

Command Syntax: DCR:STEP<n>:MEASure

Return format: <NR3><NL>

2.7 RBAL subsystem commands

The RBAL subsystem command group is used to set the DC resistance balance test parameters.

RBAL:STEPSN? Query returns the number of DCR balance test steps.

RBAL:STEP<n>:DELete
The RBAL:STEP<n>:DELete command is used to delete the DCR balance test steps.

RBAL:STEP<n>:SET
The RBAL:STEP<n>:SET commands set the DCR balance test parameters.
Command Syntax: RBAL:STEP<n>:SET <ref a>,<ref b>,<high>,<low>
where

<ref a> set A balance winding	NR1 format, from 1 to max.
<ref b> set B balance winding	NR1 format, from 1 to max.
<high>	used to set the upper limit, NR3 format, from 0 to 100k.
<low>	used to set the lower limit, NR3 format, from 0 to 100k.

Query Syntax: RBAL:STEP<n>:SET?
Return format: <NR1>,<NR1>,<NR3>,<NR3><NL>
If the query steps is empty, return format: <0><NL>

RBAL:STEP<n>:REFA
The RBAL:STEP<n>:REFA command sets A balance winding . The RBAL:STEP<n>:REFA? Query returns the A balance winding.
Command Syntax: RBAL:STEP<n>:REFA <refa> where <refa> is NR1 format, from 1 to max.
Query Syntax: RBAL:STEP<n>:REFA?
Return format: <NR1><NL>
If the query steps is empty, return format: <0><NL>

RBAL:STEP<n>:REFB
The RBAL:STEP<n>:REFB command sets B balance winding . The RBAL:STEP<n>:REFB? Query returns the B balance winding.

Command Syntax: RBAL:STEP<n>:REFB <refb>   where <refb> is NR1 format, from 1 to max.
Query Syntax: RBAL:STEP<n>:REFB?
Return format: <NR1><NL>

RBAL:STEP<n>:HIGH
The RBAL:STEP<n>:HIGH command sets the upper limit . The RBAL:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: RBAL:STEP<n>:HIGH <value>   where <value> is NR3 format, from 0 to 100k.
Query Syntax: RBAL:STEP<n>:HIGH?
Return format: <NR3><NL>

RBAL:STEP<n>:LOW
The RBAL:STEP<n>:LOW command sets the lower limit . The RBAL:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: RBAL:STEP<n>:LOW <value>   where <value> is NR3 format, from 0 to 100k.
Query Syntax: RBAL:STEP<n>:LOW?
Return format: <NR3><NL>

2.8 IW subsystem commands

The IW subsystem command group is used to set impulse winding test parameters.

IW:VADJust

The IW:VADJust command sets the voltage adjustment switch. The IW:VADJust? query returns the status of the voltage adjustment switch.
Command Syntax: IW:VADJust {{1 | ON} | {0 | OFF}}
Query Syntax: IW:VADJust?
Return format: {1 | 0}<NL>

IW:CMODe

The IW:CMODe command sets the corona mode. The IW:CMODe? query returns the current corona mode.
Command Syntax: IW:CMODe {PEAK | TOTals | FLUTters}
Query Syntax: IW:CMODe?
Return format: {PEAK | TOT | FLUT}<NL>

IW:SCOMp

The IW:SCOMp command sets the comparator switch. The IW:SCOMp? query returns the current status of the comparator.
Command Syntax: IW:SCOMp {{1 | ON} | {0 | OFF}}
Query Syntax: IW:SCOMp?
Return format: {1 | 0}<NL>

IW:FORMat

The IW:FORMat command sets the returned waveform data format. The IW:FORMat? query returns the current data format.
Command Syntax: IW:FORMat {ASCii | BIN}
Query Syntax: IW:FORMat?
Return format: {ASC | BIN}<NL>

IW:STEPSN? Query returns the number of IW test steps.

IW:STEP<n>:DELete

The IW:STEP<n>:DELete command is used to delete the IW test steps.

IW:STEP<n>:SET

The IW:STEP<n>:SET commands set the IW test parameters.

Command Syntax: IW:STEP<n>:SET <ch+>,<ch->,<mode>,<volt>,<samp rate>,<dummy imp>,<test imp>,<area on/off>,<area start pos>,<area end pos>,<area limit>,<diff on/off>,<diff start pos>,<diff end pos>,<diff limit>,<coro on/off>,<coro start pos>,<coro end pos>,<coro limit>,<phase dif on/off>,<phase dif pos>,<phase dif limit>,<wave comp on/off>,<wave comp start pos>,<wave comp end pos>,<wave comp T limit>,<wave comp V limit>,<dut no.>

where

<ch+> and <ch-> NR1 format, from 1 to 12.

<mode> fixed as NORMal

<volt> used to set test voltage, NR1 format, from 100 to 5000

<samp rate> used to set the sampling rate, { 100Msa/s | 50Msa/s | 25Msa/s | 10Msa/s | 5Msa/s | 2.5Msa/s | 1Msa/s | 500ksa/s | 250ksa/s | 100ksa/s}

<dummy imp> used to set the number of dummy impulse, NR1 format, from 0 to 8.

<test imp> used to set the number of test impulse, NR1 format, from 1 to 32.

<area on/off> used to set area size ON , {ON | 1}.

<area start pos> used to set the start position of area size, NR1 format, from 1 to 650.

<area end pos> used to set the end position of area size, NR1 format, from 1 to 650.

<area limit> used to set the limit of area size, NR3 format, from 0.1 to 99.9.

Note: limit value is only part of the data before %. Such as setting 2.5% ,please input 2.5. Below the limit of the inout is same.

<dif area on/off> used to set dif-area size ON or OFF, {{1 | ON} | {0 | OFF}}.

<dif area start pos> used to set the start position of dif-area size, NR1 format, from 1 to 650.

<dif area end pos> used to set the end position of dif-area size, NR1 format, from 1 to 650.

<dif area limit> used to set the limit of dif-area size, NR3 format, from 0.1 to 99.9.

<coro on/off>   used to set corona ON or OFF ,   {{1 | ON} | {0 | OFF}}.

<coro start pos>   used to set the start position of corona,   NR1 format, from 1 to 650.

<coro end pos>   used to set the end position of corona,   NR1 format, from 1 to 650.

<coro limit>   used to set the limit of corona,   NR1 format, from 1 to 9999.

<phase dif on/off>   used to set phase-dif ON or OFF ,   {{1 | ON} | {0 | OFF}}.

<phase dif pos>   used to set the zero position of phase difference,   NR1 format, from 2 to 20.

<phase dif limit>   used to set the limit of phase-dif,   NR3 format, from 0.1 to 99.9.

<wave comp on/off>   used to set waveform comparison ON or OFF ,   {{1 | ON} | {0 | OFF}}.

<wave comp start pos>   used to set the start position of waveform comparison, NR1 format, from 1 to 650.

<wave comp end pos>   used to set the end position of waveform comparison, NR1 format, from 1 to 650.

<wave comp T limit>   used to set the limit of time orientation error,   NR1 format, from 2 to 50.

<wave comp V limit>   used to set the limit of voltage direction error,   NR1 format, from 2 to 50.

<dut no.>   used to set DUT no.,   NR1 format, from 1 to 6.

Query Syntax: IW:STEP<n>:SET?

Return format: <NR1>,<NR1>,<NORM>,<NR1>,<{100M | 50M | 25M | 10M | 5M | 2.5M | 1M | 500k | 250k | 100k }>,<NR1>,<NR1>,<1>,<NR1>,<NR1>,<NR3>,<{1 | 0}>,<NR1>,<NR1>,<NR3>,<{1 | 0}>,<NR1>,<NR1>,<NR1>,<{1 | 0}>,<NR1>,<NR3>, <{1 | 0}>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NL>

If the query steps is empty, return format: <0><NL>

IW:STEP<n>:CHH

The IW:STEP<n>:CHH command sets CH+. The IW:STEP<n>:CHH? Query returns CH+.

Command Syntax: IW:STEP<n>:CHH <ch>   where <n> is NR1 format, from 1 to 12

Query Syntax: IW:STEP<n>:CHH?

Return format: <NR1><NL>   0 meas the step is empty.

IW:STEP<n>:CHL

The IW:STEP<n>:CHL command sets CH-. The IW:STEP<n>:CHL? Query returns CH-.
Command Syntax: IW:STEP<n>:CHL <ch> where <n> is NR1 format, from 1 to 12
Query Syntax: IW:STEP<n>:CHL?
Return format: <NR1><NL> 0 meas the step is empty.

IW:STEP<n>:MODE

The IW:STEP<n>:MODE command sets the IW mode as normal.
Command Syntax: IW:STEP<n>:MODE NORMal
Query Syntax: IW:STEP<n>:MODE?
Return format: <NORM><NL>

IW:STEP<n>:VOLTage

The IW:STEP<n>:VOLTage command sets test voltage. The IW:STEP<n>:VOLTage? Query returns the current test voltage.
Command Syntax: IW:STEP<n>:VOLTage <value> where <value> is NR1 format, from 100 to 5000.
Query Syntax: IW:STEP<n>:VOLTage?
Return format: <NR1><NL>

IW:STEP<n>:SRATe

The IW:STEP<n>:SRATe command sets sampling rate. The IW:STEP<n>:SRATe? Query returns the current sampling rate.
Command Syntax: IW:STEP<n>:SRATe {100Msa/s | 50Msa/s | 25Msa/s | 10Msa/s | 5Msa/s | 2.5Msa/s | 1Msa/s | 500ksa/s | 250ksa/s | 100ksa/s }
Query Syntax: IW:STEP<n>:SRATe?
Return format: {100M | 50M | 25M | 10M | 5M | 2.5M |
1M | 500k | 250k | 100k }<NL>

IW:STEP<n>:DIMPulse

The IW:STEP<n>:DIMPulse command sets the numbers of dummy impulse. The IW:STEP<n>:DIMPulse? Query returns the current numbers of dummy impulse.
Command Syntax: IW:STEP<n>:DIMPulse <value> where <value> is NR1 format, from 0 to 8.
Query Syntax: IW:STEP<n>:DIMPulse?
Return format: <NR1><NL>

IW:STEP<n>:TIMPulse
The IW:STEP<n>:TIMPulse command sets the numbers of test impulse. The IW:STEP<n>:TIMPulse? Query returns the current numbers of test impulse.
Command Syntax: IW:STEP<n>:TIMPulse <value> where <value> is NR1 format, from 1 to 32.
Query Syntax: IW:STEP<n>:TIMPulse?
Return format: <NR1><NL>

IW:STEP<n>:AREAsize[:STATe]
The IW:STEP<n>:AREAsize[:STATe] command sets area size comparator. The IW:STEP<n>:AREAsize[:STATe]? Query returns the current status of area size comparator.
Command Syntax: IW:STEP<n>:AREAsize[:STATe] {1 | ON}
Query Syntax: IW:STEP<n>:AREAsize[:STATe]?
Return format: <1><NL>

IW:STEP<n>:AREAsize:RANGe
The IW:STEP<n>:AREAsize:RANGe command sets the range of area size. The IW:STEP<n>:AREAsize:RANGe? Query returns the current range of area size.
Command Syntax: IW:STEP<n>:AREAsize:RANGe <start>,<end>
where,
<start> used to set the start position of area size, NR1 format, from 1 to 650.
<end> used to set the end position of area size, NR1 format, from 1 to 650.
Note: <end> value must be larger than <start> value.
Query Syntax: IW:STEP<n>:AREAsize:RANGe?
Return format: <NR1>,<NR1><NL >

IW:STEP<n>:AREAsize:LIMit
The IW:STEP<n>:AREAsize:LIMit command sets the limit of area size. The IW:STEP<n>:AREAsize:LIMit? Query returns the current limit of area size.
Command Syntax: IW:STEP<n>:AREAsize:LIMit <value>
where,
<value> is NR1,NR2 or NR3 format, from 0.1 to 99.9.
Query Syntax: IW:STEP<n>:AREAsize:LIMit?
Return format: <NR3><NL>

IW:STEP<n>:DIFarea[:STATe]

The IW:STEP<n>:DIFarea[:STATe] command sets dif-area comparator. The IW:STEP<n>:DIFarea[:STATe]? Query returns the current status of dif-area comparator.

Command Syntax: IW:STEP<n>:DIFarea [:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IW:STEP<n>:DIFarea[:STATe]?

Return format: {1 | 0} <NL>

IW:STEP<n>:DIFarea:RANGe

The IW:STEP<n>:DIFarea:RANGe command sets the range of dif-area. The IW:STEP<n>:DIFarea:RANGe? Query returns the current range of dif-area.

Command Syntax: IW:STEP<n>:DIFarea:RANGe <start>,<end>

where,

<start>   used to set the start position of dif-area,   NR1 format, from 1 to 650.

<end>   used to set the end position of dif-area,   NR1 format, from 1 to 650.

Note: <end> value must be larger than <start> value.

Query Syntax: IW:STEP<n>:DIFarea:RANGe?

Return format: <NR1>,<NR1><NL >

IW:STEP<n>:DIFarea:LIMit

The IW:STEP<n>:DIFarea:LIMit command sets the limit of dif-area. The IW:STEP<n>:DIFarea:LIMit? Query returns the current limit of dif-area.

Command Syntax: IW:STEP<n>:DIFarea:LIMit <value>

where,

<value> is NR1,NR2 or NR3 format, from 0.1 to 99.9.

Query Syntax: IW:STEP<n>:DIFarea:LIMit?

Return format: <NR3><NL>

IW:STEP<n>:COROna[:STATe]

The IW:STEP<n>:COROna[:STATe] command sets corona comparator. The IW:STEP<n>:COROna[:STATe]? Query returns the current status of corona comparator.

Command Syntax: IW:STEP<n>:COROna [:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IW:STEP<n>:COROna[:STATe]?

Return format: {1 | 0} <NL>

IW:STEP<n>:COROna:RANGe

The IW:STEP<n>:COROna:RANGe command sets the range of corona. The IW:STEP<n>:COROna:RANGe? Query returns the current range of corona.

Command Syntax: IW:STEP<n>:COROna:RANGe <start>,<end>

where,

<start> used to set the start position of dif-area, NR1 format, from 1 to 650.

<end> used to set the end position of dif-area, NR1 format, from 1 to 650.

Note: <end> value must be larger than <start> value.

Query Syntax: IW:STEP<n>:COROna:RANGe?

Return format: <NR1>,<NR1><NL >

IW:STEP<n>:COROna:LIMit

The IW:STEP<n>:COROna:LIMit command sets the limit of corona. The IW:STEP<n>:COROna:LIMit? Query returns the current limit of corona.

Command Syntax: IW:STEP<n>:COROna:LIMit <value>

where,

<value> is NR1 format, from 1 to 255.

Query Syntax: IW:STEP<n>:COROna:LIMit?

Return format: <NR1><NL>

IW:STEP<n>:PHASedif[:STATe]

The IW:STEP<n>:PHASedif[:STATe] command sets phase-dif comparator. The IW:STEP<n>:PHASedif[:STATe]? Query returns the current status of phase-dif comparator.

Command Syntax: IW:STEP<n>:PHASedif [:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IW:STEP<n>:PHASedif[:STATe]?

Return format: {1 | 0} <NL>

IW:STEP<n>:PHASedif:POSition

The IW:STEP<n>:PHASedif:POSition command sets the zero position of phase difference. The IW:STEP<n>:PHASedif:POSition? Query returns the current zero position of phase difference.

Command Syntax: IW:STEP<n>:PHASediff:POSition <value>

where,

<value> NR1 format, from 0 to 20.

Query Syntax: IW:STEP<n>:PHASediff:POSition?

Return format: <NR1><NL>

IW:STEP<n>:PHASedif:LIMit

The IW:STEP<n>:PHASedif:LIMit command sets the limit of phase-dif. The IW:STEP<n>:PHASedif:LIMit? Query returns the current limit of phase-dif.

Command Syntax: IW:STEP<n>:PHASedif:LIMit <value>

where,

<value> is NR1,NR2 or NR3 format, from 0.1 to 99.9.

Query Syntax: IW:STEP<n>:PHASedif:LIMit?

Return format: <NR3><NL>

IW:STEP<n>:WCOMp[:STATe]

The IW:STEP<n>:WCOMp[:STATe] command sets waveform comparison ON or OFF. The IW:STEP<n>:WCOMp[:STATe]? Query returns the current status of waveform comparison.

Command Syntax: IW:STEP<n>:WCOMp[:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IW:STEP<n>:WCOMp[:STATe]?

Return format: {1 | 0} <NL>

IW:STEP<n>:WCOMp:RANGe

The IW:STEP<n>:WCOMp:RANGe command sets the range of waveform comparison. The IW:STEP<n>:WCOMp:RANGe? Query returns the current range of waveform comparison.

Command Syntax: IW:STEP<n>:WCOMp:RANGe <start>,<end>

where,

<start> used to set the start position of waveform comparison, NR1 format, from 1 to 650.

<end> used to set the end position of waveform comparison, NR1 format, from 1 to 650.

Note: <end> value must be larger than <start> value.

Query Syntax: IW:STEP<n>:WCOMp:RANGe?

Return format: <NR1>,<NR1><NL >

IW:STEP<n>:WCOMp:TLIMit

The IW:STEP<n>:WCOMp:TLIMit command sets the limit of time orientation error. The IW:STEP<n>:WCOMp:TLIMit? Query returns the current limit of time orientation error.

Command Syntax: IW:STEP<n>:WCOMp:TLIMit <value>

where,

<value> is NR1 format, from 2 to 50.

Query Syntax: IW:STEP<n>:WCOMp:TLIMit?

Return format: <NR1><NL>

IW:STEP<n>:WCOMp:VLIMit

The IW:STEP<n>:WCOMp:VLIMit command sets the limit of voltage direction error. The IW:STEP<n>:WCOMp:VLIMit? Query returns the current limit of voltage direction error.

Command Syntax: IW:STEP<n>:WCOMp:VLIMit <value>

where,

<value> is NR1 format, from 2 to 50.

Query Syntax: IW:STEP<n>:WCOMp:VLIMit?

Return format: <NR1><NL>

IW:STEP<n>:DUTNo

The IW:STEP<n>:DUTNo command sets the DUT No.. The IW:STEP<n>:DUTNo? Query returns the current DUT No..

Command Syntax: IW:STEP<n>:DUTNo <value>

where,

<value> is NR1 format, from 1 to 6.

Query Syntax: IW:STEP<n>:DUTNo?

Return format: <NR1><NL>

IW:STEP<n>:TWAVeform[:DATA]?

The IW:STEP<n>:TWAVeform[:DATA]? query returns the data of test waveform.

Query Syntax: IW:STEP<n>:TWAVeform[:DATA]?

Return format: <data block><NL>

IW:STEP<n>:SWAVeform[:DATA]:GET

The IW:STEP<n>:SWAVeform[:DATA]:GET command used to perform a standard waveform sampling and return the data of standard waveform.

Command Syntax: IW:STEP<n>:SWAVeform[:DATA]:GET

Return format: <data block><NL>

IW:STEP<n>:SWAVeform[:DATA]?

The IW:STEP<n>:SWAVeform[:DATA]? query returns the data of standard waveform.

Query Syntax: IW:STEP<n>:SWAVeform[:DATA]?

Return format: <data block><NL>

IW:STEP<n>:SWAVeform[:DATA] <data block>

The IW:STEP<n>:SWAVeform[:DATA] <data block> command used to load the data of standard waveform.

Note: IW:STEP<n>:SWAVeform[:DATA]:GET and IW:STEP<n>:SWAVeform[:DATA]? returns the same data block.

Using the command IW:FORMat {ASCii | BIN} to set the data block format.

data block format：

1. #8xxxxxxxx Followed by the binary data, no Spaces, XXXXXXXX is data length.
2. $8xxxxxxxx Followed by the string data, no Spaces, XXXXXXXX is data length.

For example, header = #800001200， where #8 means that the following 8 bytes represent the data length of waveform, and 00001200 means that the valid length of waveform data is 1200.

String format is to convert binary data to the string data, each binary data into two ASCII character data (a valid data is composed of two bytes), these two characters represent binary data corresponding to the high four and low four of hexadecimal data. For example the 0x01 data is converted to "0" and "1" two characters.

IW:STEP<n>:SWAVeform:VOLTage

The IW:STEP<n>:SWAVeform:VOLTage command sets the actual test voltage of standard waveform. The IW:STEP<n>:SWAVeform:VOLTage? Returns the actual test voltage of standard waveform.

Command Syntax: IW:STEP<n>:SWAVeform:VOLTage <volt>

where,

<vol> is NR1 format, from 100 to 5000.

Query Syntax: IW:STEP<n>:SWAVeform:VOLTage?
Return format: <NR1><NL>
Note: After performing standard waveform, query the standard waveform data and query the actual control voltage should be the same time.
Load standard waveform data and load the actual control voltage should be the same time.

IW:STEP<n>:XINcrement?
The IW:STEP<n>:XINcrement? command returns the x-increment value for the currently selected source. The value is the time between consecutive sampling points in seconds.
Query Syntax: IW:STEP<n>:XINcrement?
Return format: <NR3><NL>

IW:STEP<n>:YINcrement?
The IW:STEP<n>:YINcrement? command returns the vertical voltage value.
I.e., vertical scale/25.
Query Syntax: IW:STEP<n>:YINcrement?
Return format: <NR3><NL>

2.9 IW BAL subsystem commands

The IWBAL subsystem command group is used to set the impulse winding balance test parameters.

IWBAL:STEPSN?   Query returns the number of IW balance test steps.

IWBAL:STEP<n>:DELete
The IWBAL:STEP<n>:DELete command is used to delete the IW balance test steps.

IWBAL:STEP<n>:SET
The IWBAL:STEP<n>:SET commands set the IW balance test parameters.
Command Syntax: IWBAL:STEP<n>:SET <ref a>,<ref b>,,<area on/off>,<area start pos>,<area end pos>,<area limit>,<diff on/off>,<diff start pos>,<diff end pos>,<diff limit>,<phase dif on/off>,<phase dif pos>,<phase dif limit>

where

<ref a> set A balance winding NR1 format, from 1 to max.

<ref b> set B balance winding NR1 format, from 1 to max.

<area on/off> used to set area size ON , {ON | 1}.

<area start pos> used to set the start position of area size, NR1 format, from 1 to 650.

<area end pos> used to set the end position of area size, NR1 format, from 1 to 650.

<area limit> used to set the limit of area size, NR3 format, from 0.1 to 99.9.

Note: limit value is only part of the data before %. Such as setting 2.5% ,please input 2.5. Below the limit of the inout is same.

<dif area on/off> used to set dif-area size ON or OFF, {{1 | ON} | {0 | OFF}}.

<dif area start pos> used to set the start position of dif-area size, NR1 format, from 1 to 650.

<dif area end pos> used to set the end position of dif-area size, NR1 format, from 1 to 650.

<dif area limit> used to set the limit of dif-area size, NR3 format, from 0.1 to 99.9.

<phase dif on/off> used to set phase-dif ON or OFF , {{1 | ON} | {0 | OFF}}.

<phase dif pos> used to set the zero position of phase difference, NR1 format, from 2 to 20.

<phase dif limit> used to set the limit of phase-dif, NR3 format, from 0.1 to 99.9.

Query Syntax: IWBAL:STEP<n>:SET?

Return format: NR1>,<NR1>,<1>,<NR1>,<NR1>,<NR3>,<{1|0}>,<NR1>,<NR1>,<NR3>,
<{1|0}>,<NR1>,<NR3>,<NL>

If the query steps is empty, return format: <0><NL>

IWBAL:STEP<n>:REFA

The IWBAL:STEP<n>:REFA command sets A balance winding . The IWBAL:STEP<n>:REFA? Query returns the A balance winding.

Command Syntax: IWBAL:STEP<n>:REFA <refa> where <refa> is NR1 format, from 1 to max.

Query Syntax: IWBAL:STEP<n>:REFA?

Return format: <NR1><NL>

If the query steps is empty, return format: <0><NL>

IWBAL:STEP<n>:REFB

The IWBAL:STEP<n>:REFB command sets B balance winding . The IWBAL:STEP<n>:REFB? Query returns the B balance winding.

Command Syntax: IWBAL:STEP<n>:REFB <refb> where <refb> is NR1 format, from 1 to max.

Query Syntax: IWBAL:STEP<n>:REFB?

Return format: <NR1><NL>

IWBAL:STEP<n>:AREAsize[:STATe]

The IWBAL:STEP<n>:AREAsize[:STATe] command sets area size comparator. The IWBAL:STEP<n>:AREAsize[:STATe]? Query returns the current status of area size comparator.

Command Syntax: IWBAL:STEP<n>:AREAsize[:STATe] {1 | ON}

Query Syntax: IWBAL:STEP<n>:AREAsize[:STATe]?

Return format: <1><NL>

IWBAL:STEP<n>:AREAsize:RANGe

The IWBAL:STEP<n>:AREAsize:RANGe command sets the range of area size. The IWBAL:STEP<n>:AREAsize:RANGe? Query returns the current range of area size.

Command Syntax: IWBAL:STEP<n>:AREAsize:RANGe <start>,<end>

where,

<start> used to set the start position of area size, NR1 format, from 1 to 650.

<end> used to set the end position of area size, NR1 format, from 1 to 650.

Note: <end> value must be larger than <start> value.

Query Syntax: IWBAL:STEP<n>:AREAsize:RANGe?

Return format: <NR1>,<NR1><NL >

IWBAL:STEP<n>:AREAsize:LIMit

The IWBAL:STEP<n>:AREAsize:LIMit command sets the limit of area size. The IWBAL:STEP<n>:AREAsize:LIMit? Query returns the current limit of area size.

Command Syntax: IWBAL:STEP<n>:AREAsize:LIMit <value>

where,

<value> is NR3 format, from 0.1 to 99.9.

Query Syntax: IWBAL:STEP<n>:AREAsize:LIMit?

Return format: <NR3><NL>

IWBAL:STEP<n>:DIFarea[:STATe]

The IWBAL:STEP<n>:DIFarea[:STATe] command sets dif-area comparator. The IWBAL:STEP<n>:DIFarea[:STATe]? Query returns the current status of dif-area comparator.

Command Syntax: IWBAL:STEP<n>:DIFarea [:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IWBAL:STEP<n>:DIFarea[:STATe]?

Return format: {1 | 0} <NL>

IWBAL:STEP<n>:DIFarea:RANGe

The IWBAL:STEP<n>:DIFarea:RANGe command sets the range of dif-area. The IWBAL:STEP<n>:DIFarea:RANGe? Query returns the current range of dif-area.

Command Syntax: IWBAL:STEP<n>:DIFarea:RANGe <start>,<end>

where,

<start> used to set the start position of dif-area, NR1 format, from 1 to 650.

<end> used to set the end position of dif-area, NR1 format, from 1 to 650.

Note: <end> value must be larger than <start> value.

Query Syntax: IWBAL:STEP<n>:DIFarea:RANGe?

Return format: <NR1>,<NR1><NL >

IWBAL:STEP<n>:DIFarea:LIMit

The IWBAL:STEP<n>:DIFarea:LIMit command sets the limit of dif-area. The IWBAL:STEP<n>:DIFarea:LIMit? Query returns the current limit of dif-area.

Command Syntax: IWBAL:STEP<n>:DIFarea:LIMit <value>

where,

<value> is NR3 format, from 0.1 to 99.9.

Query Syntax: IWBAL:STEP<n>:DIFarea:LIMit?

Return format: <NR3><NL>

IWBAL:STEP<n>:PHASedif[:STATe]

The IWBAL:STEP<n>:PHASedif[:STATe] command sets phase-dif comparator. The IWBAL:STEP<n>:PHASedif[:STATe]? Query returns the current status of phase-dif comparator.

Command Syntax: IWBAL:STEP<n>:PHASedif [:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: IWBAL:STEP<n>:PHASedif[:STATe]?

Return format: {1 | 0} <NL>

IWBAL:STEP<n>:PHASedif:POSition

The IWBAL:STEP<n>:PHASedif:POSition command sets the zero position of phase difference. The IWBAL:STEP<n>:PHASedif:POSition? Query returns the current zero position of phase difference.

Command Syntax: IWBAL:STEP<n>:PHASediff:POSition <value>

where,

<value> NR1 format, from 0 to 20.

Query Syntax: IWBAL:STEP<n>:PHASediff:POSition?

Return format: <NR1><NL>

IWBAL:STEP<n>:PHASedif:LIMit

The IWBAL:STEP<n>:PHASedif:LIMit command sets the limit of phase-dif. The IWBAL:STEP<n>:PHASedif:LIMit? Query returns the current limit of phase-dif.

Command Syntax: IWBAL:STEP<n>:PHASedif:LIMit <value>

where,

<value> is NR3 format, from 0.1 to 99.9.

Query Syntax: IWBAL:STEP<n>:PHASedif:LIMit?

Return format: <NR3><NL>

2.10 OS subsystem commands

The OS subsystem command group is used to set the OS resistance test parameters.

OS:STEPSN? Query returns the number of OS test steps.

OS:STEP<n>:DELete

The OS:STEP<n>:DELete command is used to delete the OS test steps.

OS:STEP<n>:SET

The OS:STEP<n>:SET commands set the OS test parameters.

Command Syntax: OS:STEP<n>:SET <ch+>,<ch->,<std>,<dut no.>

where

<ch+> and <ch-> NR1 format, from 1 to 12.

<std> NR3 format, from 0 to 100k.

<dut no.> used to set DUT no., NR1 format, from 1 to 6.

Query Syntax: OS:STEP<n>:SET?
Return format: <NR1>,<NR1>,<NR3><NR1><NL>
If the query steps is empty, return format: <0><NL>

OS:STEP<n>:CHH
The OS:STEP<n>:CHH command sets CH+. The OS:STEP<n>:CHH? Query returns CH+.
Command Syntax: OS:STEP<n>:CHH <ch>   where <n> is NR1 format, from 1 to 12
Query Syntax: OS:STEP<n>:CHH?
Return format: <NR1><NL>   0 meas the step is empty.

OS:STEP<n>:CHL
The OS:STEP<n>:CHL command sets CH-. The OS:STEP<n>:CHL? Query returns CH-.
Command Syntax: OS:STEP<n>:CHL <ch>   where <n> is NR1 format, from 1 to 12
Query Syntax: OS:STEP<n>:CHL?
Return format: <NR1><NL>   0 meas the step is empty.

OS:STEP<n>:STD
The OS:STEP<n>:STD command sets the nominal value. The OS:STEP<n>:STD? Query returns the nominal value.
Command Syntax: OS:STEP<n>:STD <value>   where <value> is NR3 format, from 0 to 100k.
Query Syntax: OS:STEP<n>:STD?
Return format: <NR3><NL>

OS:STEP<n>:DUTNo
The DCR:STEP<n>:DUTNo command sets the number of test object . The DCR:STEP<n>:DUTNo? Query returns the current number of test object.
Command Syntax: DCR:STEP<n>:DUTNo <value>   where <value> is NR1 format, from 1 to 6.
Query Syntax: DCR:STEP<n>:DUTNo?
Return format: <NR1><NL>

2.11 IR subsystem commands

The IR subsystem command group is used to set the insulation resistance test parameters.

IR:STEPSN? Query returns the number of IR test steps.

IR:STEP<n>:DELete
The IR:STEP<n>:DELete command is used to delete the IR test steps.

IR:STEP<n>:SET
The IR:STEP<n>:SET commands set the IR test parameters.
Command Syntax: IR:STEP<n>:SET
<ch+>,<ch->,<volt>,<time>,<ramp>,<high>,<low>,<dut no.>
where

<ch+> and <ch->		(@(ch1,ch2,ch3)) format, from 1 to 12.
<volt>	used to set test voltage,	NR3 format, from 100 to 1000V.
<time>	used to set test time,	NR3 format, from 0.3 to 999.9s.
<ramp>	used to set ramp time,	NR3 format, from 0.1 to 999.9s.
<high>	use to set the high limit	NR3 format, from 0 to 50G.
<low>	use to set the low limit	NR3 format, from 1M to 50G.
<dut no.>	used to set DUT no.,	NR1 format, from 1 to 6.

Query Syntax: IR:STEP<n>:SET?
Return format: (@(1,2)), (@(3,4)),<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,<NR1><NL>
If the query steps is empty, return format: <0><NL>

IR:STEP<n>:CHH
The IR:STEP<n>:CHH command sets CH+. The IR:STEP<n>:CHH? Query returns CH+.
Command Syntax: IR:STEP<n>:CHH (@(ch1,ch2,ch3)) where <n> is NR1 format, from 1 to 12
Query Syntax: IR:STEP<n>:CHH?
Return format: (@(1,2,3))<NL> (@()) meas the step is empty.

IR:STEP<n>:CHL
The IR:STEP<n>:CHL command sets CH-. The IR:STEP<n>:CHL? Query returns CH-.
Command Syntax: IR:STEP<n>:CHL <ch> where <n> is NR1 format, from 1 to 12
Query Syntax: IR:STEP<n>:CHL?
Return format: (@(1,2,3))<NL> (@()) meas the step is empty.

IR:STEP<n>:VOLTage
The IR:STEP<n>:VOLTage command sets the test voltage. The IR:STEP<n>:VOLTage? Query returns the test voltage value.
Command Syntax: IR:STEP<n>:VOLTage <value> where <value> is NR3 format, from 100 to 1000V.
Query Syntax: IR:STEP<n>:VOLTage?
Return format: <NR3><NL>

IR:STEP<n>:TIME
The IR:STEP<n>:TIME command sets the test time . The IR:STEP<n>:TIME? Query returns the test time.
Command Syntax: IR:STEP<n>:TIME <value> where <value> is NR3 format, from 0.3 to 999.9s.
Query Syntax: IR:STEP<n>:TIME?
Return format: <NR3><NL>

IR:STEP<n>:RAMP
The IR:STEP<n>:RAMP command sets the voltage ramp time . The IR:STEP<n>:RAMP? Query returns the voltage ramp time.
Command Syntax: IR:STEP<n>:RAMP <value> where <value> is NR3 format, from 0.1 to 999.9s.
Query Syntax: IR:STEP<n>:RAMP?
Return format: <NR3><NL>

IR:STEP<n>:HIGH
The IR:STEP<n>:HIGH command sets the upper limit . The IR:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: IR:STEP<n>:HIGH <value> where <value> is NR3 format, from 0 to 50G, 0 means shut off the upper limit.
Query Syntax: IR:STEP<n>:HIGH?
Return format: <NR3><NL>

IR:STEP<n>:LOW
The IR:STEP<n>:LOW command sets the lower limit . The IR:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: IR:STEP<n>:LOW <value>   where <value> is NR3 format, from 1M to 50G.
Query Syntax: IR:STEP<n>:LOW?
Return format: <NR3><NL>

IR:STEP<n>:DUTNo
The IR:STEP<n>:DUTNo command sets the number of test object . The IR:STEP<n>:DUTNo? Query returns the current number of test object.
Command Syntax: IR:STEP<n>:DUTNo <value>   where <value> is NR1 format, from 0 to 6, only step 1 can be set as 0.
Query Syntax: IR:STEP<n>:DUTNo?
Return format: <NR1><NL>

2.12 HIPOT subsystem commands

The HIPOT subsystem command group is used to set the hipot test parameters.

HIPot:STEPSN? Query returns the number of HIPOT test steps.

HIPot:STEP<n>:DELete
The HIPot:STEP<n>:DELete command is used to delete the hipot test steps.

HIPot:STEP<n>:SET
The HIPot:STEP<n>:SET commands set the HIPOT test parameters.
Command Syntax: HIPot:STEP<n>:SET
<ch+>,<ch->,<freq>,<volt>,<time>,<ramp>,<high>,<low>,<arc>,<offset>,<dut no.>
where
<ch+> and <ch->   (@(ch1,ch2,ch3)) format, from 1 to 12.
<freq>   used to set frequency of test voltage,   {AC50 | AC60 | DC}
<volt>   used to set test voltage,
NR3 format, AC from 50 to 5000V DC: from 50 to 6000V.
<time>   used to set test time,   NR3 format, from 0.3 to 999.9s.

<ramp>   used to set ramp time,   NR3 format, from 0.1 to 999.9s.

<high>   use to set the upper limit

NR3 format, AC: from 0.001mA to 30.000mA   DC: from 0.001mA to 10.000mA.

<low>   use to set the low limit

NR3 format, AC: from 0.001mA to 30.000mA   DC: from 0.001mA to 10.000mA.

0 means shut off the lower limit,

<arc>   use to set the arc limit

NR3 format, AC: from 0.0mA to 15.0mA   DC: from 0.0mA to 10.0mA.

0 means shut off the arc detection.

<offset>   use to set the offset current

NR3 format, from 0.000mA to 10.000mA

<dut no.>   used to set DUT no.,   NR1 format, from 1 to 6.

Query Syntax: HIPot:STEP<n>:SET?

Return format:

(@(1,2)), (@(3,4)),{AC50|AC60|DC},<NR3>,<NR3>,<NR3>,<NR3>,<NR3>,
<NR3>,<NR3>,<NR1><NL>

If the query steps is empty, return format: <0><NL>

HIPot:STEP<n>:CHH

The HIPot:STEP<n>:CHH command sets CH+. The HIPot:STEP<n>:CHH? Query returns CH+.

Command Syntax: HIPot:STEP<n>:CHH (@(ch1,ch2,ch3))   where (@(ch1,ch2,ch3)) is from ch1 to ch12

Query Syntax: HIPot:STEP<n>:CHH?

Return format: (@(1,2,3))<NL>   (@()) meas the step is empty.

HIPot:STEP<n>:CHL

The HIPot:STEP<n>:CHL command sets CH-. The HIPot:STEP<n>:CHL? Query returns CH-.

Command Syntax: HIPot:STEP<n>:CHL (@(ch1,ch2,ch3))   where @(ch1,ch2,ch3) is from ch1 to ch12

Query Syntax: HIPot:STEP<n>:CHL?

Return format: (@(1,2,3))<NL>   (@()) means the step is empty.

HIPot:STEP<n>:FREQuency
The HIPot:STEP<n>:FREQuency command sets the frequency of test voltage. The HIPot:STEP<n>:FREQuency? Query returns the test voltage freqency.
Command Syntax: HIPot:STEP<n>:FREQuency {AC50|AC60|DC}
Query Syntax: HIPot:STEP<n>:FREQuency?
Return format: {AC50|AC60|DC}<NL>

HIPot:STEP<n>:VOLTage
The HIPot:STEP<n>:VOLTage command sets the test voltage. The HIPot:STEP<n>:VOLTage? Query returns the test voltage value.
Command Syntax: HIPot:STEP<n>:VOLTage <value> where <value> is NR3 format, AC: from 50 to 5000V, DC: from 50 to 6000V.
Query Syntax: HIPot:STEP<n>:VOLTage?
Return format: <NR3><NL>

HIPot:STEP<n>:TIME
The HIPot:STEP<n>:TIME command sets the test time . The HIPot:STEP<n>:TIME? Query returns the test time.
Command Syntax: HIPot:STEP<n>:TIME <value> where <value> is NR3 format, from 0.3 to 999.9s.
Query Syntax: HIPot:STEP<n>:TIME?
Return format: <NR3><NL>

HIPot:STEP<n>:RAMP
The HIPot:STEP<n>:RAMP command sets the voltage ramp time . The HIPot:STEP<n>:RAMP? Query returns the voltage ramp time.
Command Syntax: HIPot:STEP<n>:RAMP <value> where <value> is NR3 format, from 0.1 to 999.9s.
Query Syntax: HIPot:STEP<n>:RAMP?
Return format: <NR3><NL>

HIPot:STEP<n>:HIGH
The HIPot:STEP<n>:HIGH command sets the upper limit . The HIPot:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: HIPot:STEP<n>:HIGH <value> where <value> is NR3 format, AC: from 0.001mA to 30.000mA, DC: from 0.001mA to 10.000mA.
Query Syntax: HIPot:STEP<n>:HIGH?

Return format: <NR3><NL>

HIPot:STEP<n>:LOW

The HIPot:STEP<n>:LOW command sets the lower limit . The HIPot:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: HIPot:STEP<n>:LOW <value> where <value> is NR3 format, AC:from 0.000mA to 30.000mA, DC: from 0.000mA to 10.000mA.
Query Syntax: HIPot:STEP<n>:LOW?
Return format: <NR3><NL>

HIPot:STEP<n>:ARC

The HIPot:STEP<n>:ARC command sets the arc detection current . The HIPot:STEP<n>:ARC? Query returns the arc detection current.
Command Syntax: HIPot:STEP<n>:ARC <value> where <value> is NR3 format, AC:from 0.0mA to 15.0mA, DC: from 0.0mA to 10.0mA, 0 means shut off the arc detection function.
Query Syntax: HIPot:STEP<n>:ARC?
Return format: <NR3><NL>

HIPot:STEP<n>:OFFSet

The HIPot:STEP<n>:OFFSet command sets the offset value . The HIPot:STEP<n>:OFFSet? Query returns the offset value.
Command Syntax: HIPot:STEP<n>:OFFSet <value> where <value> is NR3 format, from 0.000mA to 10.000mA.
Query Syntax: HIPot:STEP<n>:OFFSet?
Return format: <NR3><NL>

HIPot:STEP<n>:OFFSet:GET

The HIPot:STEP<n>:OFFSet:GET command executes offset operation and returns the offset value.
Command Syntax: HIPot:STEP<n>:OFFSet:GET

HIPot:STEP<n>:DUTNo

The HIPot:STEP<n>:DUTNo command sets the number of test object . The HIPot:STEP<n>:DUTNo? Query returns the current number of test object.
Command Syntax: HIPot:STEP<n>:DUTNo <value> where <value> is NR1 format, from 0 to 6, only step 1 can be set as 0.

Query Syntax: HIPot:STEP<n>:DUTNo?
Return format: <NR1><NL>

2.13 L subsystem commands

The L subsystem command group is used to set the inductance test parameters.

L:STEPSN? Query returns the number of L test steps.

L:STEP<n>:DELete
The DCR:STEP<n>:DELete command is used to delete the L test steps.

L:STEP<n>:SET
The DCR:STEP<n>:SET commands set the inductance test parameters.
Command Syntax: L:STEP<n>:SET <ch+>,<ch->,<para>,<freq>,<level>,<L std>,<L high>,<L low>,<speed>,<delay>,<L dev>,<Q comp>,<Q high>,<Q low>,<Q dev>,<dut no.>

where
<ch+> and <ch-> NR1 format, from 1 to 12.
<freq> used to set test frequency,
{50hz|60hz|100hz|120hz|1Khz|10Khz|20Khz|40Khz|50Khz|100Khz}
<level> used to set test level, {0.1V | 0.3V | 1.0V}
<L std>,<L high> and <L low> NR3 format, from 0 to 100M.
<speed> used to set test speed, { SLOW | MEDium | FAST }
<delay> used to set delay time, NR3 format, from 0 to 60s.
<L dev> used to set deviation value, NR3 format, from -100M to 100M.
<Q comp> used to set the compare for Q {{1 | ON} | {0 | OFF}}.
<Q high> and <Q low> NR3 format, from 0 to 100M.
<dut no.> used to set DUT no., NR1 format, from 1 to 6.

Query Syntax: L:STEP<n>:SET?
Return format: <NR1>,<NR1>,{LS|LP},{50|60|100|120|1k|10k|20k|40k|50k|100k},{0.1V|0.3V|1.0V},<NR3>,<NR3>,<NR3>,{SLOW|MED|FAST},<NR3>,<NR3>,<{1 | 0}>,<NR3>,<NR3>,<NR1><NL>
If the query steps is empty, return format: <0><NL>

L:STEP<n>:CHH

The L:STEP<n>:CHH command sets CH+. The L:STEP<n>:CHH? Query returns CH+.
Command Syntax: L:STEP<n>:CHH <ch> where <ch> is NR1 format, from 1 to 12
Query Syntax: L:STEP<n>:CHH?
Return format: <NR1><NL> 0 means the step is empty.

L:STEP<n>:CHL

The L:STEP<n>:CHL command sets CH-. The L:STEP<n>:CHL? Query returns CH-.
Command Syntax: L:STEP<n>:CHL <ch> where <ch> is NR1 format, from 1 to 12
Query Syntax: L:STEP<n>:CHL?
Return format: <NR1><NL> 0 means the step is empty.

L:STEP<n>:PARA

The L:STEP<n>:PARA command sets the test parameter . The L:STEP<n>:PARA? Query returns the current test parameter.
Command Syntax: L:STEP<n>:PARA { LS| LP }
Query Syntax: L:STEP<n>:PARA?
Return format: { LS| LP }<NL>

L:STEP<n>:FREQuency

The L:STEP<n>:FREQuency command sets the test frequency. The L:STEP<n>:FREQuency? Query returns the current test frequency.
Command Syntax: L:STEP<n>:FREQuency {50hz|60hz|100hz|120hz|1Khz|10Khz|20Khz|40Khz|50Khz|100Khz}
Query Syntax: L:STEP<n>:FREQuency?
Return format: {50|60|100|120|1k|10k|20k|40k|50k|100k}<NL>

L:STEP<n>:LEVel

The L:STEP<n>:LEVel command sets the test level. The L:STEP<n>:LEVel? Query returns the current test level.
Command Syntax: L:STEP<n>:LEVel {0.1V | 0.3V | 1.0V}
Query Syntax: L:STEP<n>:LEVel?
Return format: {0.1V | 0.3V | 1.0V}<NL>

L:STEP<n>:STD

The L:STEP<n>:STD command sets the nominal value. The L:STEP<n>:STD? Query returns the nominal value.

Command Syntax: L:STEP<n>:STD <value>   where <value> is NR3 format, from 0 to 100M.
Query Syntax: L:STEP<n>:STD?
Return format: <NR3><NL>

L:STEP<n>:HIGH
The L:STEP<n>:HIGH command sets the upper limit . The L:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: L:STEP<n>:HIGH <value>   where <value> is NR3 format, from 0 to 100M.
Query Syntax: L:STEP<n>:HIGH?
Return format: <NR3><NL>

L:STEP<n>:LOW
The L:STEP<n>:LOW command sets the lower limit . The L:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: L:STEP<n>:LOW <value>   where <value> is NR3 format, from 0 to 100M.
Query Syntax: L:STEP<n>:LOW?
Return format: <NR3><NL>

L:STEP<n>:SPEed
The L:STEP<n>:SPEed command sets the test speed . The L:STEP<n>:SPEed? Query returns the current test speed.
Command Syntax: L:STEP<n>:SPEed { SLOW | MEDium | FAST }
Query Syntax: L:STEP<n>:SPEed?
Return format: {SLOW | MED | FAST}<NL>

L:STEP<n>:DELAY
The L:STEP<n>:DELAY command sets the delay time . The L:STEP<n>:DELAY? Query returns the delay time.
Command Syntax: L:STEP<n>:DELAY <value>   where <value> is NR3 format, from 0 to 60s.
Query Syntax: L:STEP<n>:DELAY?
Return format: <NR3><NL>

L:STEP<n>:DEViation

The L:STEP<n>:DEViation command sets the deviation of L value . The L:STEP<n>:DEViation? Query returns the deviation of L value.

Command Syntax: L:STEP<n>:DEViation <value> where <value> is NR3 format, from -100M to 100M.

Query Syntax: L:STEP<n>:DEViation?

Return format: <NR3><NL>

L:STEP<n>:COMPQ

The L:STEP<n>:COMPQ command sets the comparison of Q . The L:STEP<n>:COMPQ? Query returns the status of Q comparison.

Command Syntax: L:STEP<n>:COMPQ {{1 | ON} | {0 | OFF}}

Query Syntax: L:STEP<n>:COMPQ?

Return format: {1 | 0} <NL>

L:STEP<n>:HIGHQ

The L:STEP<n>:HIGHQ command sets the upper limit of Q. The L:STEP<n>:HIGHQ? Query returns the upper limit of Q.

Command Syntax: L:STEP<n>:HIGHQ <value> where <value> is NR3 format, from 0 to 100M.

Query Syntax: L:STEP<n>:HIGHQ?

Return format: <NR3><NL>

L:STEP<n>:LOWQ

The L:STEP<n>:LOWQ command sets the lower limit of Q. The L:STEP<n>:LOWQ? Query returns the lower limit of Q.

Command Syntax: L:STEP<n>:LOWQ <value> where <value> is NR3 format, from 0 to 100M.

Query Syntax: L:STEP<n>:LOWQ?

Return format: <NR3><NL>

L:STEP<n>:DUTNo

The L:STEP<n>:DUTNo command sets the number of test object . The L:STEP<n>:DUTNo? Query returns the current number of test object.

Command Syntax: L:STEP<n>:DUTNo <value> where <value> is NR1 format, from 1 to 6.

Query Syntax: DCR:STEP<n>:DUTNo?

Return format: <NR1><NL>

L:STEP<n>:CLEar:OPEN
The L:STEP<n>:CLEar:OPEN command used to perform open clear operation.
Command Syntax: L:STEP<n>:CLEar:OPEN
Note: After success of clear operation, return format: <NR3>,<NR3><NL>
After fail of clear operation, return format: <FAIL><NL>

L:STEP<n>:CLEar:SHORt
The L:STEP<n>:CLEar:SHORt command used to perform short clear operation.
Command Syntax: L:STEP<n>:CLEar:SHORt
Note: After success of clear operation, return format: <NR3>,<NR3><NL>
After fail of clear operation, return format: <FAIL><NL>

L:STEP<n>:CLEar:DATA:OPEN
The L:STEP<n>:CLEar:DATA:OPEN command used to upload the data of open clear operation. The L:STEP<n>:CLEar:DATA:OPEN? Query returns the data of open clear operation.
Command Syntax: L:STEP<n>:CLEar:DATA:OPEN <d1>,<d2><NL>
Query Syntax: L:STEP<n>:CLEar:DATA:OPEN?
Return format: <NR3>,<NR3><NL>

L:STEP<n>:CLEar:DATA:SHORt
The L:STEP<n>:CLEar:DATA:SHORt command used to upload the data of short clear operation. The L:STEP<n>:CLEar:DATA:SHORt? Query returns the data of short clear operation.
Command Syntax: L:STEP<n>:CLEar:DATA:SHORt <d1>,<d2><NL>
Query Syntax: L:STEP<n>:CLEar:DATA:SHORt?
Return format: <NR3>,<NR3><NL>

L:STEP<n>:MEASure
The L:STEP<n>:MEASure command used to triggers a measurement and return the measurement data.
Command Syntax: L:STEP<n>:MEASure
Return format: <NR3><NL>

2.14 LBAL subsystem commands

The LBAL subsystem command group is used to set the inductance balance test parameters.

LBAL:STEPSN? Query returns the number of inductance balance test steps.

LBAL:STEP<n>:DELete
The RBAL:STEP<n>:DELete command is used to delete the inductance balance test steps.

LBAL:STEP<n>:SET
The LBAL:STEP<n>:SET commands set the inductance balance test parameters.
Command Syntax: LBAL:STEP<n>:SET <ref a>,<ref b>,<high>,<low>
where
<ref a> set A balance winding NR1 format, from 1 to max.
<ref b> set B balance winding NR1 format, from 1 to max.
<high> used to set the upper limit, NR3 format, from 0 to 100M.
<low> used to set the lower limit, NR3 format, from 0 to 100M.

Query Syntax: LBAL:STEP<n>:SET?
Return format: <NR1>,<NR1>,<NR3>,<NR3><NL>
If the query steps is empty, return format: <0><NL>

LBAL:STEP<n>:REFA
The LBAL:STEP<n>:REFA command sets A balance winding . The LBAL:STEP<n>:REFA? Query returns the A balance winding.
Command Syntax: LBAL:STEP<n>:REFA <refa> where <refa> is NR1 format, from 1 to max.
Query Syntax: LBAL:STEP<n>:REFA?
Return format: <NR1><NL>
If the query steps is empty, return format: <0><NL>

LBAL:STEP<n>:REFB
The LBAL:STEP<n>:REFB command sets B balance winding . The LBAL:STEP<n>:REFB? Query returns the B balance winding.
Command Syntax: LBAL:STEP<n>:REFB <refb> where <refb> is NR1 format, from 1 to max.
Query Syntax: LBAL:STEP<n>:REFB?
Return format: <NR1><NL>

LBAL:STEP<n>:HIGH
The LBAL:STEP<n>:HIGH command sets the upper limit . The LBAL:STEP<n>:HIGH? Query returns the upper limit.
Command Syntax: LBAL:STEP<n>:HIGH <value> where <value> is NR3 format, from 0 to 100M.
Query Syntax: LBAL:STEP<n>:HIGH?
Return format: <NR3><NL>

LBAL:STEP<n>:LOW
The LBAL:STEP<n>:LOW command sets the lower limit . The LBAL:STEP<n>:LOW? Query returns the lower limit.
Command Syntax: LBAL:STEP<n>:LOW <value> where <value> is NR3 format, from 0 to 100M.
Query Syntax: LBAL:STEP<n>:LOW?
Return format: <NR3><NL>

2.15 SEQ subsystem commands

The SEQ subsystem command group is used to set the sequence of the test project.

SEQ
The SEQ commands is used to set the sequence of test project. The SEQ? query returns the status and sequence of the test project.
Command Syntax: SEQ <item>,{{1 | ON} | {0 | OFF}},<item>,{{1 | ON} | {0 | OFF}}… where <item> is IW, DCR,L,HIPOT,IR or OS.
Query Syntax: SEQ?
Return format: <item>,{1|0},<item>,{1|0}…

SEQ:SEQ

The SEQ:SEQ command sets the sequence of the test project. The SEQ:SEQ? query return the sequence of the test project.

Command Syntax: SEQ:SEQ <item>,<item>…

Query Syntax: SEQ:SEQ?

Return format: <item>,<item><NL>

SEQ:TEST:DCR

The SEQ:TEST:DCR command sets the status of DCR. The SEQ:TEST:DCR? query returns the status of DCR.

Command Syntax: SEQ:TEST:DCR {{1 | ON} | {0 | OFF}}

Query Syntax: SEQ:TEST:DCR?

Return format: {1 | 0}<NL>

SEQ:TEST:IW

The SEQ:TEST:IW command sets the status of IW. The SEQ:TEST:IW? query returns the status of IW.

Command Syntax: SEQ:TEST:IW {{1 | ON} | {0 | OFF}}

Query Syntax: SEQ:TEST:IW?

Return format: {1 | 0}<NL>

SEQ:TEST:OS

The SEQ:TEST:OS command sets the status of OS. The SEQ:TEST:OS? query returns the status of OS.

Command Syntax: SEQ:TEST:OS {{1 | ON} | {0 | OFF}}

Query Syntax: SEQ:TEST:OS?

Return format: {1 | 0}<NL>

SEQ:TEST:IR

The SEQ:TEST:IR command sets the status of IR. The SEQ:TEST:IR? query returns the status of IR.

Command Syntax: SEQ:TEST:IR {{1 | ON} | {0 | OFF}}

Query Syntax: SEQ:TEST:IR?

Return format: {1 | 0}<NL>

SEQ:TEST:HIPot
The SEQ:TEST:HIPot command sets the status of IW. The SEQ:TEST:HIPot? query returns the status of HIPot.
Command Syntax: SEQ:TEST:HIPot {{1 | ON} | {0 | OFF}}
Query Syntax: SEQ:TEST:HIPot?
Return format: {1 | 0}<NL>

SEQ:TEST:L
The SEQ:TEST:L command sets the status of IW. The SEQ:TEST:L? query returns the status of L.
Command Syntax: SEQ:TEST:L {{1 | ON} | {0 | OFF}}
Query Syntax: SEQ:TEST:L?
Return format: {1 | 0}<NL>

2.16 FETCh subsystem commands

The FETCh subsystem commands are used to output test result and judgment results.

FETCh:AREPort
The FETCh:AREPort command sets whether returns the judgment results automatically after the end of comprehensive tesing. The FETCh:APERort? query returns the current status.
Command Syntax: FETCh:AREPort {{1 | ON} | {0 | OFF}}
Query Syntax: FETCh:AREPort?
Return format: {1 | 0}<NL>

FETCh:RESult[:JUDGment]? query returns the judgment result.
Return format: {PASS | FAIL}<NL>

FETCh:RESult:ALL? query returns all the test results.
Return format:

	data block format	Description
DCR	1,1,2,DCR,1.2345ohm,HI	DUT No,CH+,CH-,DCR,value,judgment
RCR Balance	1,(1-2),(3-4),RBAL,0.10mohm,OK	DUT No,RefA,RefB,RBAL,value,judgment
IW(Surge)	1,1,2,IW,0.1,OK,0.2,OK,3,OK,1.2,OK,OK,OK	DUT No,CH+,CH-,IW,AREAsize,judgment,DIFarea,judgment,COROna,judgment,PHASedif,judgment,judgment,judgment
IW Balance	1,12,34,IWBAL,,NG	DUT No,RefA,RefB,IWBAL,,judgment
OS	1,1,2,OS,,OK	DUT No,CH+,CH-,OS,,judgment
IR	1,12,34,IR,1.234Gohm,OK	DUT No,CH+,CH-,IR,value,judgment
HIPOT	1,12,34,HIPOT,1.234mA,OK	DUT No,CH+,CH-,HIPOT,value,judgment
L	1,1,2,Ls,123.45uH,OK	DUT No,CH+,CH-,Ls,value,judgment
L Balance	1,(1-2),(3-4),LBAL, 10.234uH,OK	DUT No,RefA,RefB,LBAL,value,judgment

2.17 STATistic subsystem commands

The STATistic subsystem commands is used to clear or save statistic data.

STATistic:CLEar
The STATistic:CLEAr command is used to clear the statistic data。
Command Syntax: STATistic:CLEar

STATistic:RESult? The STATistic:RESult? query returns the statistic data.
Return format: <NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,
<NR1>,<NR1>,<NR1>,<NR 1><NL>
Respectively total count, total passed count, IR count, IR passed count, HIPOT count, HIPOT passed count, IW count, IW passed count, DCR count, DCR passed count, L count, L passed count

STATistic:RESult:IR? The STATistic:RESult:IR? query returns the statistic data for Insulation Resistance test.
Return format: <NR1>,<NR1><NL>
Respectively IR count, IR passed count

STATistic:RESult:HIPot? The STATistic:RESult:HIPot? query returns the statistic data for HIPOT test.
Return format: <NR1>,<NR1><NL>
Respectively HIPOT count, HIPOT passed count

STATistic:RESult:IW? The STATistic:RESult:IW? query returns the statistic data for Impulse winding test.
Return format: <NR1>,<NR1><NL>
Respectively IW count, IW passed count

STATistic:RESult:DCR? The STATistic:RESult:DCR? query returns the statistic data for DC resistance test.
Return format: <NR1>,<NR1><NL>
Respectively DCR count, DCR passed count

STATistic:RESult:L? The STATistic:RESult:L? query returns the statistic data for Inductance test.
Return format: <NR1>,<NR1><NL>
Respectively L count, L passed count

2.18 Mass MEMory subsystem commands

The Mass MEMory subsystem commands load and store files. Figure 2-14-1 shows the Mass MEMory subsystem command tree.

```
Mass MEMory──┬ :LOAD    ────────────  :STATe    <file number>
             │                                 [<“filename”>]
             ├ :SAVE STORe────  :STATe    <flie number>
             │                                   [<“filename”>]
             └ :DELete  ───────────  :STATe   <flie number>
                                                   [<“filename”>]
```

Figure 2-14-1

ⓘ NOTE: The Mass MEMory subsystem commands will be ignored in the phase of testing.

MMEMory:LOAD:STATe

The MMEMory:LOAD:STATe command is used to load the stored file.

Command Syntax: MMEMory:LOAD:STATe <flie number>

Where,

<file number> is the file serial number ranging from 1 to 780 without unit.

For example: WrtCmd(“MMEM:LOAD:STAT 1”); load file 1.

ⓘNOTE: 1. If the file you want to load is not available, “File not exist” message will be displayed on the system message line.

2. If the input file number is out of 1 to 900, message “Out of file range” will be displayed on the system message line.

Command Syntax: MMEMory:LOAD:STATe <“filename”>

The command is used to find and load the file using the file name directly.

MMEMory:SAVE:STATe or **STORe:STATe**

The MMEMory:SAVE:STATe or STORe:STATe command is used to save the current setting data to a file.

Command Syntax: MMEMory:STORe:STATe <flie number> [,<“filename”>]

Where,

<file number> is the file serial number ranging from 1 to 780 , NR1 format without unit.
<"filename"> The file name consists of less than 20 ASCII characters. <Unnamed> will be the default name, if you don't input a file name.
For example: WrtCmd("MMEM:STOR:STAT 1， "#U9036*"")；
ⓘ NOTE: U9036 will not give a warning message when the existent file is to be over written.
☞NOTE： The file name assigned by bus will be quoted without any change, thus user can enter some special characters such as special symbols and letters in lower case that cannot be input on the panel of the instrument.

MMEMory:DELete:STATe
The MMEMory:DELete:STATe command deletes a file.
Command Syntax: MMEMory:DELete:STATe <file number>
Where,
<file number> is the file serial number ranging form 1 to 780, NR1 format without unit.
For example: WrtCmd("MMEM:DEL:STAT 1"); delete file 1.
ⓘ NOTE: U9036 will not give a warning message when a file is to be deleted.

Command Syntax: Mass MEMory:DELete:STATe "filename"
This command is used to delete a file using the filename directly.

2.19 KEY subsystem commands

KEY commands are used to control the keys and knobs on the operation panel of U9845R.

KEY:LOCal	enable the front panel operation
KEY:MEASure	enter into the <TEST DISP> page
KEY:SETup	enter into the <TEST SETUP> page
KEY:SYSTem	enter into the <SYSTEM > page
KEY:FILE	enter into the <FILE LIST> page
KEY:UPPer	upper the cursor
KEY:DOWN	down the cursor
KEY:LEFT	left the cursor
KEY:RIGHt	right the cursor
KEY:TAB	TAB key
KEY:NUM<n>	numeric key， n range from 0 to 9
KEY:DOT	decimal point key
KEY:SIGN	minus key
KEY:ENTer	enter key
KEY:BACKsapce	backspace key
KEY:ESC	escape key
KEY:STARt	start key
KEY:STOP	stop key
KEY:SAVE	save key
KEY:F<n>	soft key, n is from 1 to 7
KEY:ACDC	AC/DC key
KEY:IRIWT	IR/IWT key
KEY:LDCR	L/DCR key
KEY:U	Unit u key
KEY:M	Unit m key
KEY:KP	Unit k/p key
KEY:MN	Unit M/n key

3 Error and warning message

The bus commands may have some spelling errors, syntax errors or wrong parameters. U9036 executes a command after the command is analyzed. If one of above errors occurs, U9036 halts the command analysis, and the rest commands will be ignored. If a command (for example a trigger command is ignored.) is ignored, the rest commands will be executed. The error and warning messages will be displayed on the system message line.

The following table shows the common error and warning messages, which will be displayed on the message line when they occur.

Error message	Description
Undefined message	Unknown command is received. Usually there is a spelling error in the command. For example: TRG should be TRIG DISP:PAG TEST should be DISP:PAGE TEST
Data out of range	The data is out of range. For example: TEIG:DEL 100, the delay time is out of its range.
Invalid parameter	Unrecognizable parameter is used. For example: TRIG:SOUR INTER， INTER is not the correct short-form and should not used.
Invalid suffix	Units are unrecognizable, or the units are not correct. For example: TRIG:DEL 200us， us can not be the unit of the impulse voltage.
Data too long	Data is too long. For example: The number of characters for a file name can not exceed 20 characters and numeric parameter, 20 characters.
Syntax error	Error syntax, for example:DISP.PAGE TSET, where（.）should be （:）.
Trigger ignored	When U9036 is in the testing state, all trigger signals will be ignored.
Command ignored	Some command may be ignored. For example: DISP:PAGE TSET When U9036 is in the testing state, this command will be ignored.

4 Programming Examples

This chapter lists three programming examples in the development environments of Visual C++ 6.0, Visual Basic 6.0 and LabVIEW 8.5. All the examples are based on VISA (Virtual Instrument Software Architecture).
VISA is an API (Application Programming Interface) used for controlling instruments. It is convenient for users to develop testing applications which are independent of the types of instrument and interface. Note that "VISA" here we mention is NI (National Instrument)-VISA. NI-VISA is an API written by NI based on VISA standard. You can use NI-VISA to achieve the communication between the oscilloscope and PC via GPIB, USB, RS232, LAN and such instrument bus. As VISA has defined a set of software commands, users can control the instrument without understanding the working state of the interface bus. See NI-VISA User Manual and NI-VISA Programmmer Reference Manual for more information about NI-VISA API.

A typical application of VISA contains the following parts:
1. Set up the conversation for the existing resource
2. Configure the resource (such as: Baud rate)
3. Close the conversation

Preparation for Programming

Download NI-VISA software from http://www.ni.com to install it. The installing path is C:\Program Files\IVI Foundation\VISA.
Take U9036 as an example to show how to construct the communication between an impulse winding tester and a PC. Use a USB cable with one teminal connecting the DEVICE interface on the rear panel of the instrument and the other one connecting the USB interface of PC, as is shown in figure 4-1.

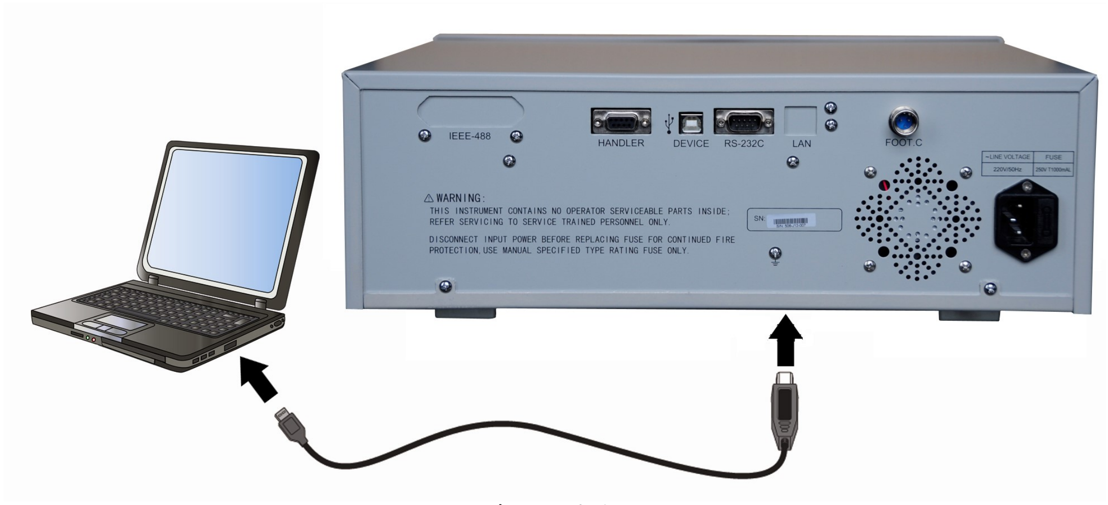
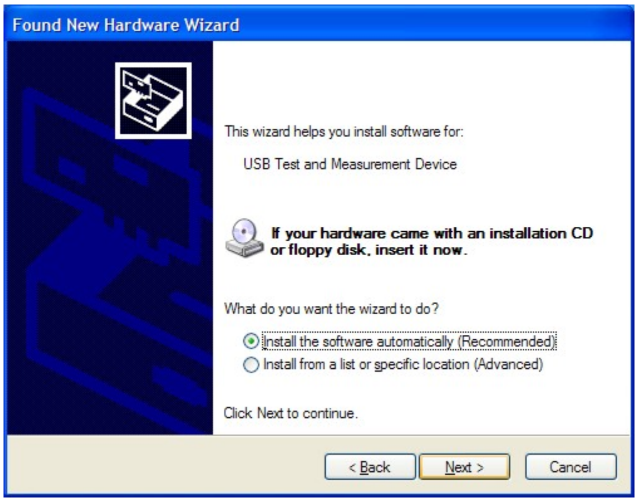

IEEE-488
HANDLER
DEVICE
RS-232C
LAN
FOOT.C
⚠ WARNING:
THIS INSTRUMENT CONTAINS NO OPERATOR SERVICEABLE PARTS INSIDE;
REFER SERVICING TO SERVICE TRAINED PERSONNEL ONLY.
DISCONNECT INPUT POWER BEFORE REPLACING FUSE FOR CONTINUED FIRE
PROTECTION,USE MANUAL SPECIFIED TYPE RATING FUSE ONLY.
SN
~LINE VOLTAGE
FUSE
220V/50Hz

Figure 4-1

Switch the instrument power on. An upgrading guide dialoge will pop up and you can install USB Test and Measurement Device software by prompt information.


Found New Hardware Wizard

This wizard helps you install software for:

USB Test and Measurement Device

If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

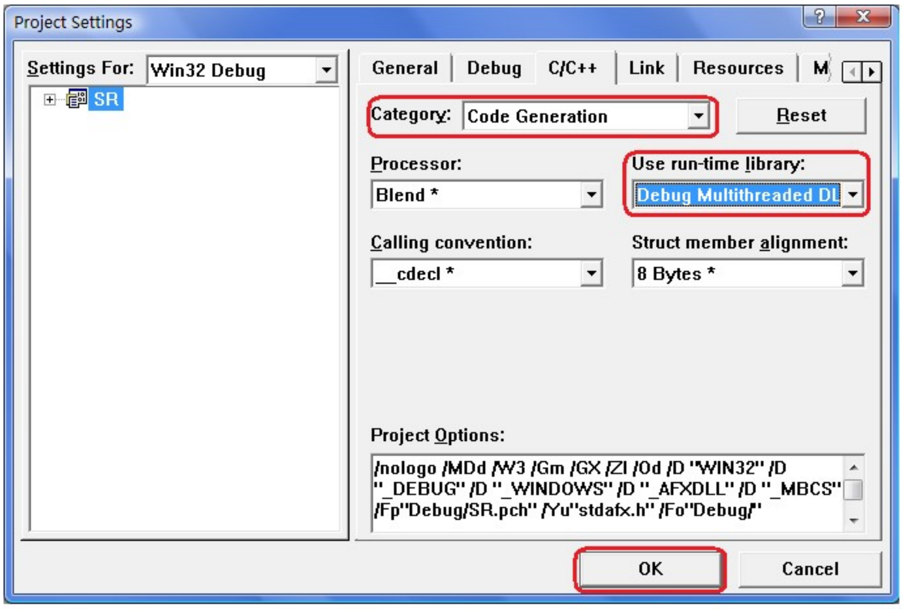
Click Next to continue.

4.1 Visual C++ 6.0 Programming Example

Open Visual C++ 6.0, take the following steps:

1. Create a project based on MFC.

2. Choose Project→Settings→C/C++; select “Code Generation” in Category and “Debug Multithreaded DLL” in Use run-time library; click OK; as is shown in figure 4-1-1.



Settings For: Win32 Debug

SR

Category: Code Generation

Processor: Blend *

Use run-time library: Debug Multithreaded DL

Calling convention: __cdecl *

Struct member alignment: 8 Bytes *

Project Options:

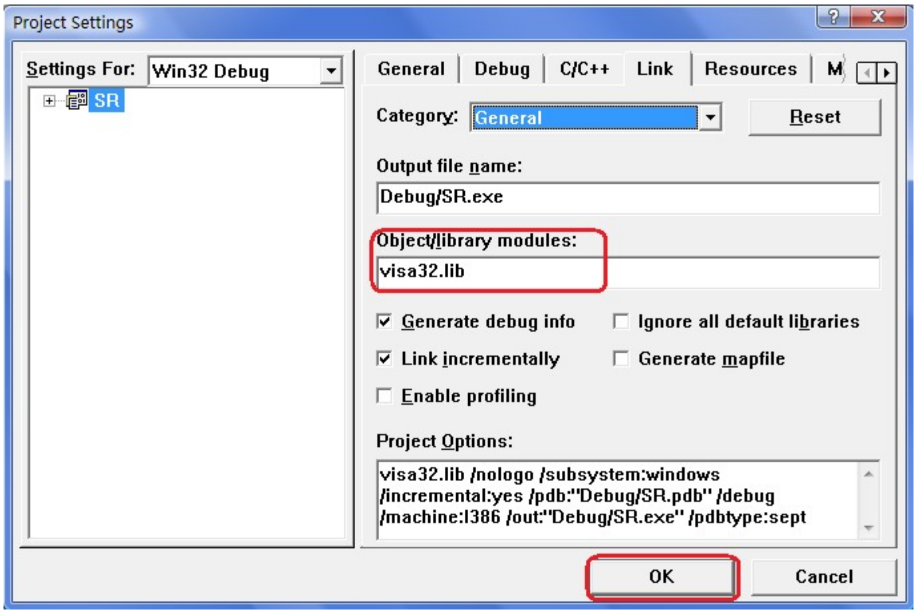
```
/nologo /MDd /W3 /Gm /GX /ZI /Od /D "WIN32" /D
"_DEBUG" /D "_WINDOWS" /D "_AFXDLL" /D "_MBCS"
/Fp"Debug/SR.pch" /Yu"stdafx.h" /Fo"Debug/"
```

Figure 4-1-1

1. Choose Project-> Settings-> Link, add the file visa32.lib manually in Object/library modules; click OK; as is shown in figure 4-1-2.

2. Choose Tools->Options ->Directories; select Include files in Show directories for, and then double click the blank in Directories to add the path of Include: C:\Program Files\IVI Foundation\VISA\WinNT\include, as is shown in figure 4-1-3.

Select Library files in Show directories for, and then double click the blank in Directories to add the path of Lib: C\Program Files\IVI Foundation\VISA\WinNT\lib\msc.



Category: General

Output file name: Debug/SR.exe

Object/library modules: visa32.lib

☑ Generate debug info ☐ Ignore all default libraries

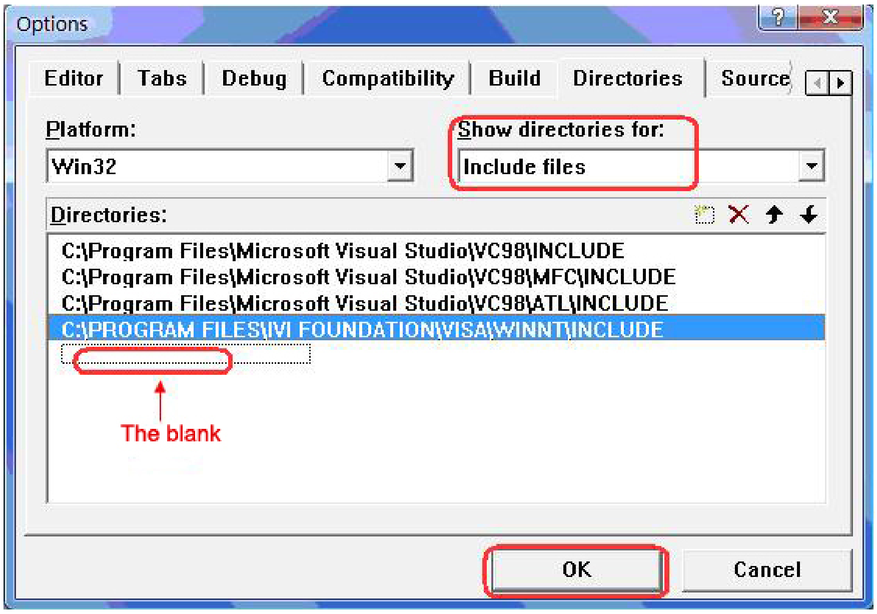
☑ Link incrementally ☐ Generate mapfile

☐ Enable profiling

Project Options:

```
visa32.lib /nologo /subsystem:windows
/incremental:yes /pdb:"Debug/SR.pdb" /debug
/machine:I386 /out:"Debug/SR.exe" /pdbtype:sept
```

Figure 4-1-2



Platform: Win32

Show directories for: Include files

Directories:

```
C:\Program Files\Microsoft Visual Studio\VC98\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE
C:\PROGRAM FILES\IVI FOUNDATION\VISA\WINNT\INCLUDE
```

Figure 4-1-3

5. Add controls: Static Text, Edit and Button. See figure 4-1-4.

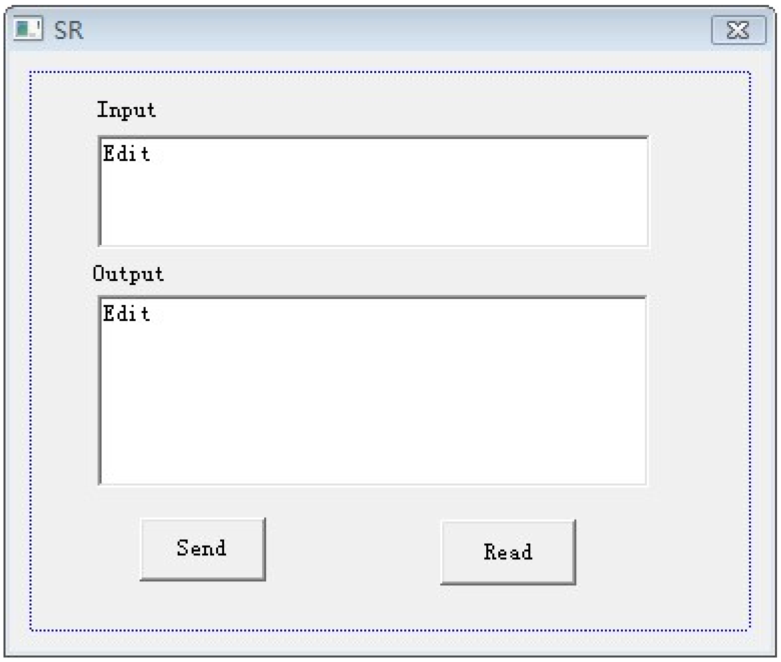


Figure 4-1-4

(1) Add two Static Text controls respectively named as Input and Output.

(2) Add two Edit controls, and then add two variables--m_send and m_read to them respectively. See figure 4-1-5 and figure 4-1-6.

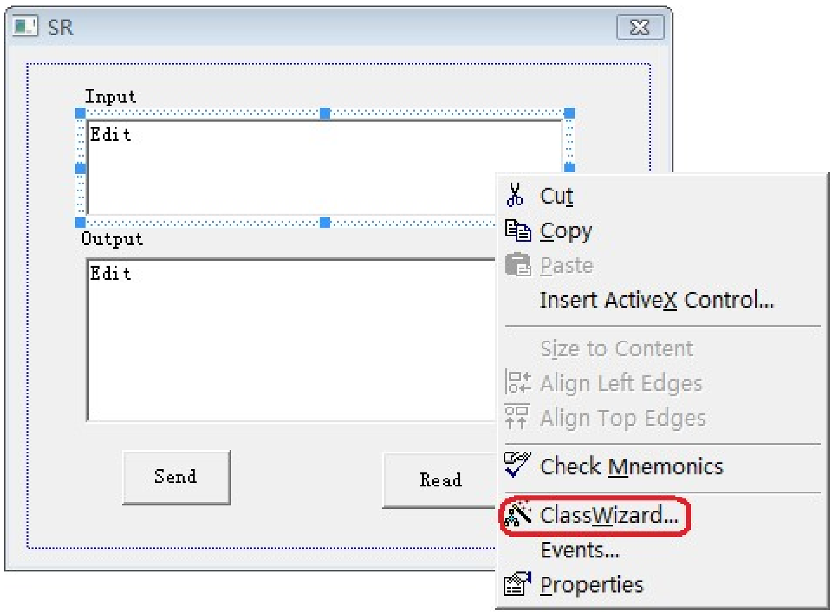
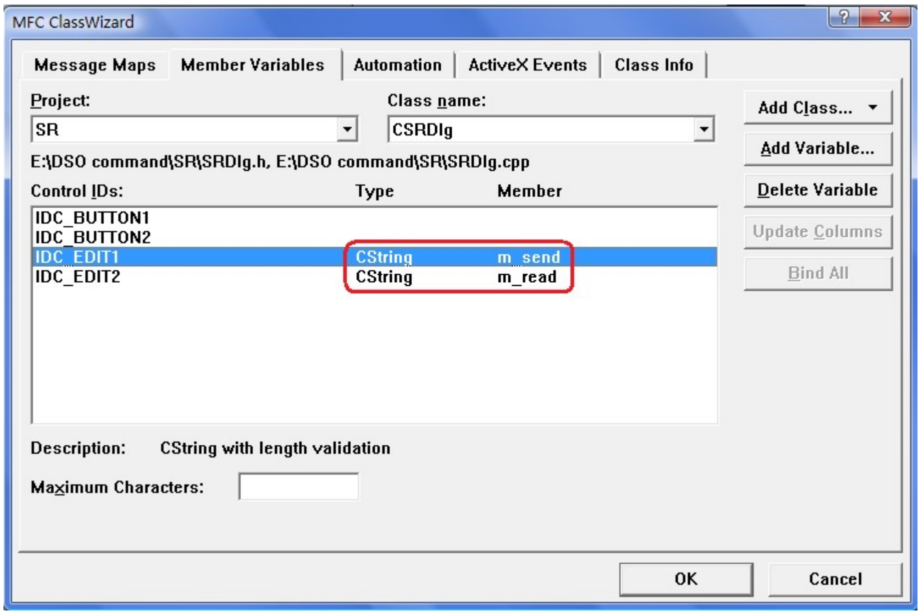


Figure 4-1-5


MFC ClassWizard

Project: SR

Class name: CSRDlg

E:\DSO command\SR\SRDlg.h, E:\DSO command\SR\SRDlg.cpp

Control IDs:	Type	Member
IDC_BUTTON1		
IDC_BUTTON2		
IDC_EDIT1	CString	m_send
IDC_EDIT2	CString	m_read

Description: CString with length validation

Maximum Characters:

Figure 4-1-6

(3) Add two Button controls named as Send and Read respectively.

6. Double click Send, enter the programming environment.

(1) Declare "#include"visa.h"" in header file.

(2) Define relative variables and then add the following codes:

```
ViSession defaultRM, vi;
char buf [256] = {0};
CString s,strTemp;
char* stringTemp;
ViChar buffer [VI_FIND_BUFLEN];
ViRsrc matches=buffer;
ViUInt32 nmatches;
ViFindList list;
```

(3) In ::CSRDlg(CWnd* pParent /*=NULL*/)

: CDialog(CSRDlg::IDD, pParent), order m_send = _T("*IDN?\n");

(4) Add the following codes to ::OnInitDialog().

```
viOpenDefaultRM (&defaultRM);
//acquire USB resource of visa
viFindRsrc(defaultRM, "USB?*", &list,&nmatches, matches);
viOpen (defaultRM,matches,VI_NULL,VI_NULL,&vi);
```

(5) Add the following codes in Send.

```
//send the receiving commands
UpdateData (TRUE);
strTemp = m_send + "\n";
stringTemp = (char *)(LPCTSTR)strTemp;
viPrintf (vi,stringTemp);
```

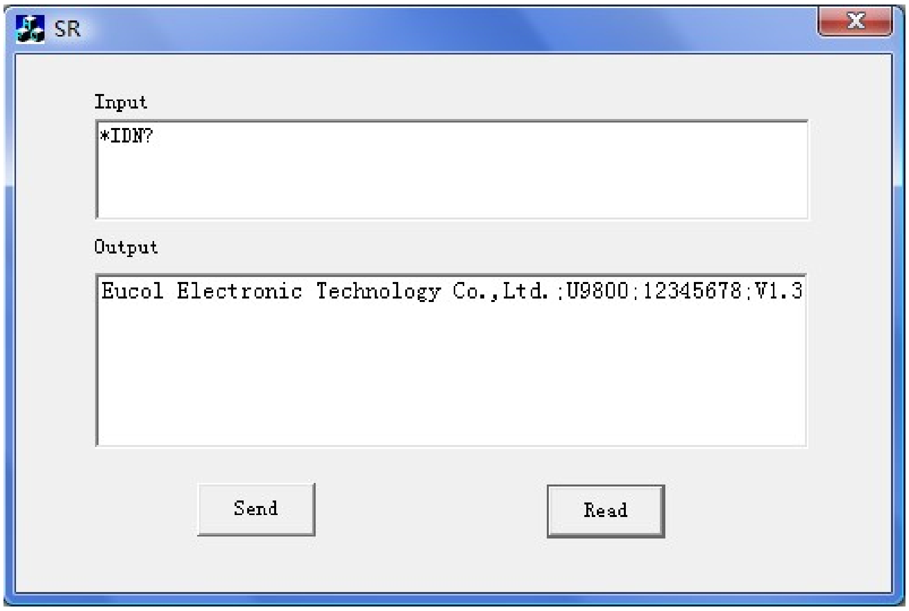
(6) Add the following codes in Read.

```
//read the result
viScanf (vi, "%t\n", &buf);
//display the results
m_read = buf;
UpdateData (FALSE);
```

(7) Add the following codes in ::OnQueryDragIcon().

```
//close resource.
viClose (vi);
viClose (defaultRM);
```

7. Save, build and run the project, you will get an EXE file. When the oscilloscope has been successfully connected with PC, input a command such as *IDN? (the default input command) in Input edit box and cilck Send and Read successively, the oscilloscope will return the result which will be displayed in Output edit box. See figure 4-1-7.



```
*IDN?
```

```
Eucol Electronic Technology Co.,Ltd.;U9800;12345678;V1.3
```

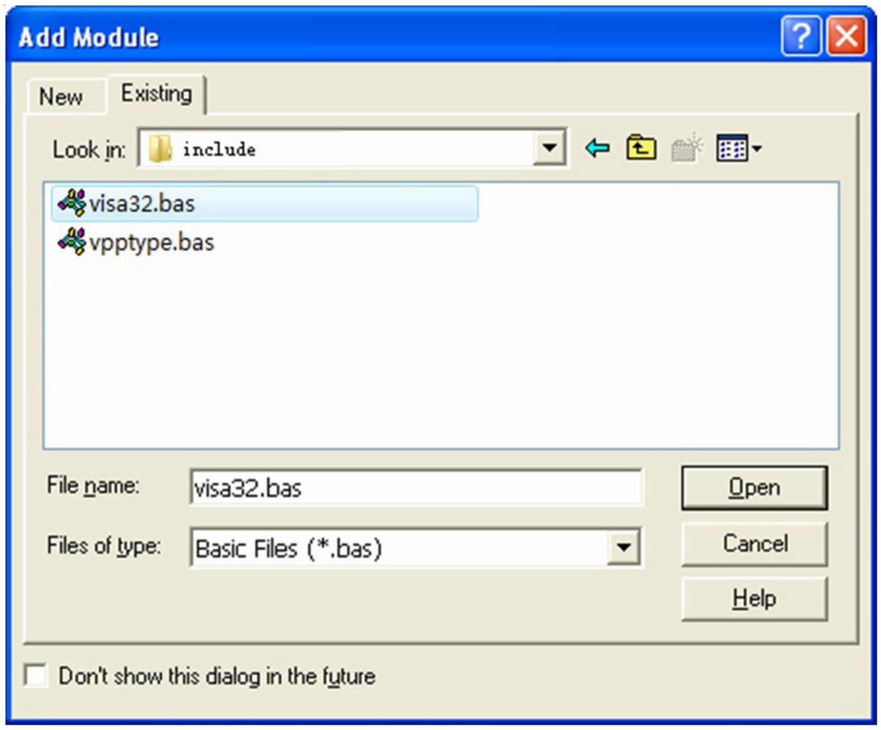
Figure 4-1-7

4.2 Visual Basic 6.0 Programming Examples

Open Visual Basic6 6.0, take the following setps:

1. Create a Standard EXE project.

2. Choose Project-> Add Module->Existing; find the visa32.bas file in the Add Module under the path of NI-VISA: C:\Program Files\IVI Foundation\VISA\WinNT\include, and then add it. See figure 4-2-1.


Add Module

New | Existing

Look in: include

visa32.bas
vpptype.bas

File name: visa32.bas

Files of type: Basic Files (*.bas)

Figure 4-2-1

3. Add two Lables respectively named as Input and Output, two TexBox and two CommandButtons named as Send and Read seperately. Set Text in the attribute of TextBox under Input as *IDN?. See figure 4-2-2.

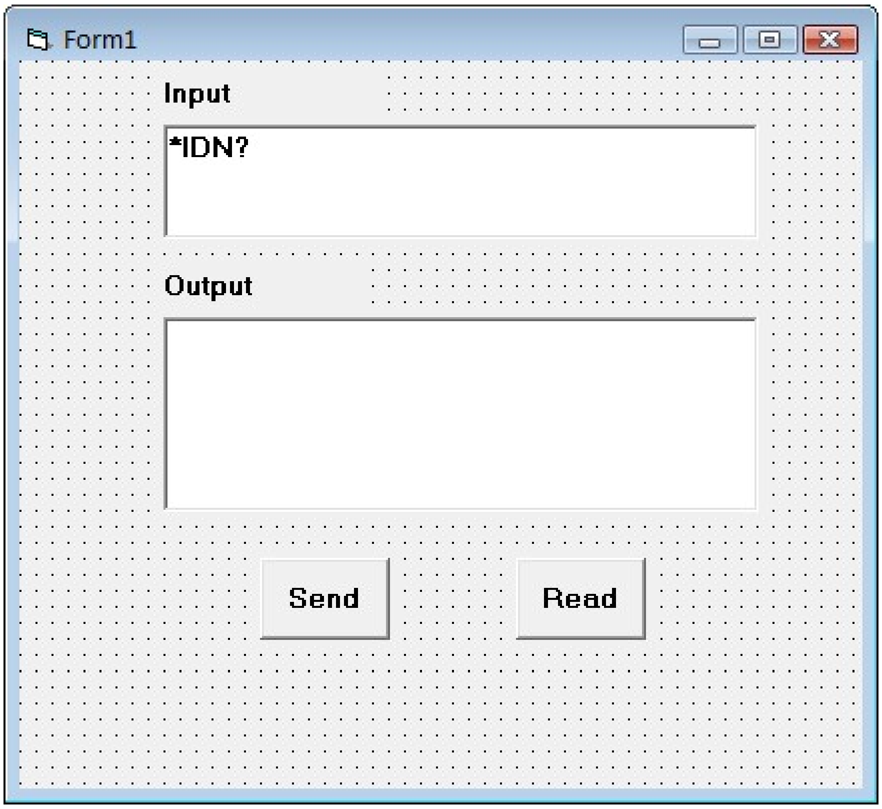


Figure 4-2-2

4. Choose Project->Project1 Properties->General, Select Form1 form the drop down box of Startup Object.

5. Double click Send, enter the programming environment and add the following codes:

```
Dim defrm As Long
Dim vi As Long
Dim list As Long
Dim nmatches As Long
Dim matches As String * 200 ' reserves to acquire the equipment ID.
Dim strRes As String * 200

Private Sub Cmd_Read_Click()
' acquire the command return state
Call viVScanf(vi, "%t", strRes)
```

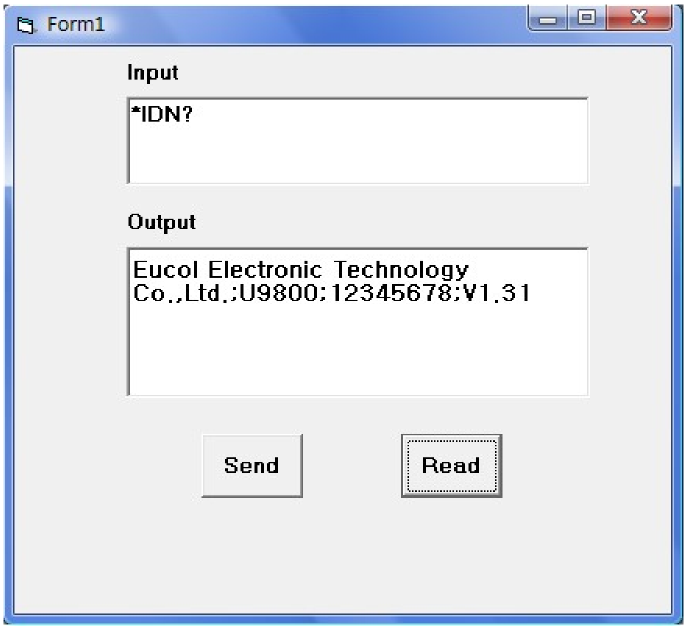
```
Txt_output.Text = strRes
End Sub

Private Sub Cmd_Send_Click()
' send the command to query
Call viVPrintf(vi, Txt_input.Text + Chr$(10), 0)
End Sub

Private Sub Form_Load()
' acquire the usb source of visa
Call viOpenDefaultRM(defrm)
Call viFindRsrc(defrm, "USB?*", list, nmatches, matches)
' open the device
Call viOpen(defrm, matches, 0, 0, vi)
End Sub

Private Sub Form_Unload(Cancel As Integer)
' close the resource
Call viClose(vi)
Call viClose(defrm)
End Sub
```

6. Save and run the project, you will get a single executable program. When the oscilloscope has been successfully connected with PC, you can input a command such as *IDN? (the default input command) in Input edit box and cilck Send and Read successively, the oscilloscope will return the result which will be displayed in Output edit box. See figure 4-2-3.



```
*IDN?
```

```
Eucol Electronic Technology
Co.,Ltd.;U9800;12345678;V1.31
```

Figure 4-2-3

4.3 LabVIEW 8.5 Programming Examples

Run LabVIEW8.5, take the following steps.

1. Enter Getting Started, choose New>>Blank VI to create a new VI.

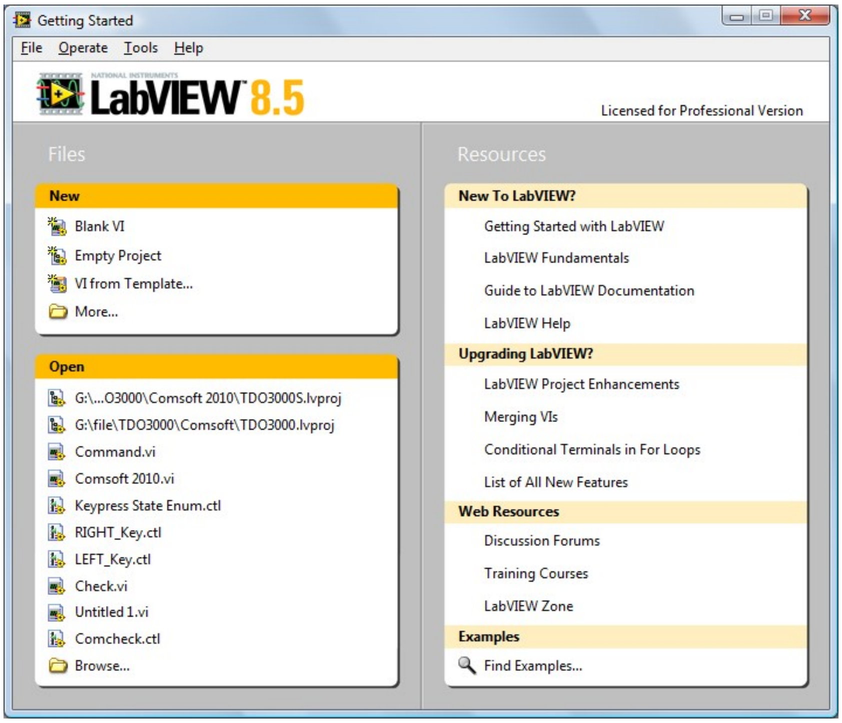


Figure 4-3-1

2. Right-click the Front Panel to choose Controls>>Modern>>Boolean>>OK Button; add three buttons and respectively define them as Write, Read and Stop. See figure 4-3-2.

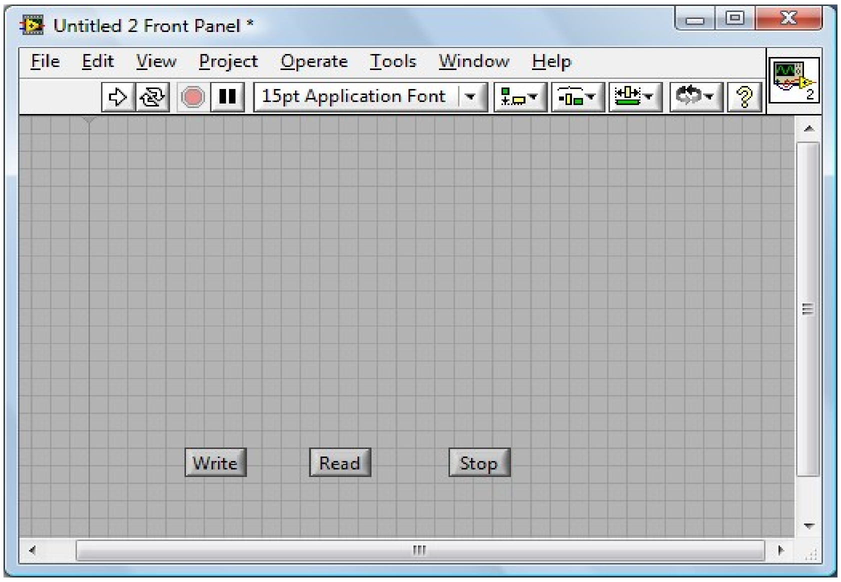


Figure 4-3-2

3. Open the Block Diagram, right-click it and choose Functions>> Programming>> Structure>> Event Structure to add an event structure.

4. Open the Block Diagram; right-click the event structure to choose Add Event Case…; Add the Value Change event for each control; drag all terminals into their own event structure.

5. Choose the Value Change event structure of the Write terminal; right-click the blank of the Block Diagram to select Functions>>Instrument I/O>>VISA>>VISA Write; add a VISA Write function for the Value Change event structure of the Write terminal.

6. Right-click the Block Diagram to choose Functions>>Instrument I/O>>VISA>>VISA Advanced>> VISA Open; add a VISA Open function on the left side of the Write structure event.

7. Right-click the VISA resource name terminal of the VISA Open function; click the shortcut menu and select Create>>Control to create a VISA resource name.

8. Wire the VISA resource out terminal of the VISA Open function to the VISA resource name terminal of the VISA Write function in the event structure; Connect the error out terminal of the VISA Open function with the error in terminal of the VISA Write function.

9. Right-click the write buffer terminal of the VISA Writer function; click the shortcut menu and choose Create>>Control to create a write buffer as shown in figure 4-3-3.

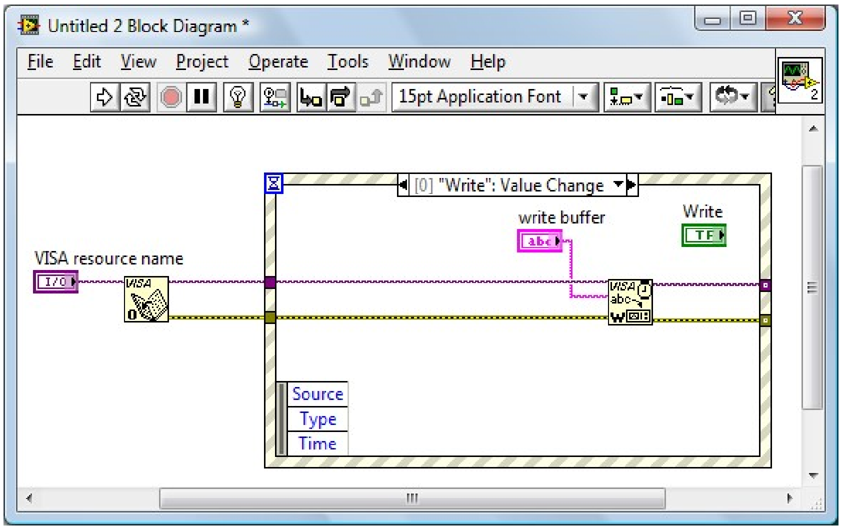

Untitled 2 Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
[0] "Write": Value Change
write buffer
Write
VISA resource name
Source
Type
Time

Figure 4-3-3

10. Select the Value Change event structure of the Read terminal; right-click Functions>> Instrument I/O>> VISA>> VISA Read to add a VISA Read function into the "Read": Value Change event structure.

11. Right-click the read buffer terminal of the VISA Read function; click the shortcut menu and choose Create>>Indicator to create a read butter.

12. Right-click the byte count terminal of the VISA Read function; click the shortcut menu and choose Create>>Constant to create a constant as 1024.

13. Wire the VISA resource out terminal of the VISA Open function to the VISA resource name terminal of the VISA Read function in the event structure; connect the error out terminal of the VISA Open function with the error in terminal of the VISA Read function shown as figure 4-3-4.

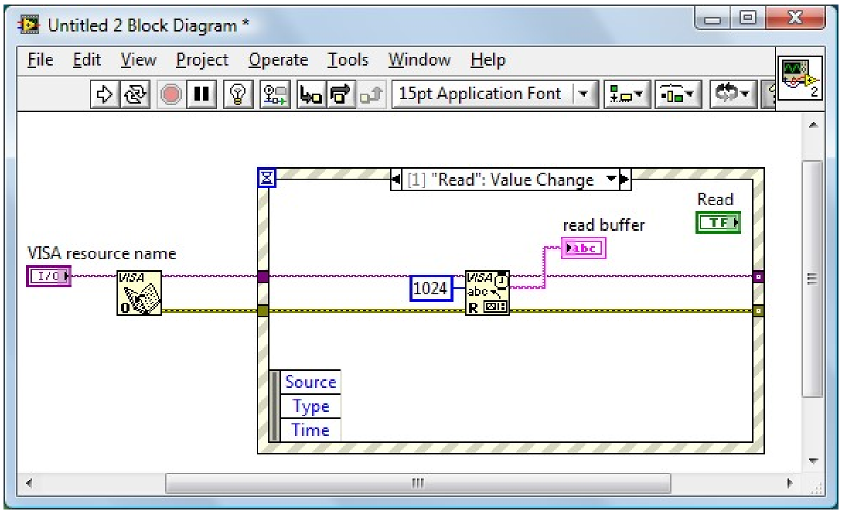


Figure 4-3-4

14. Select the Value Change event structure of the Stop terminal; right-click the blank of the Block Diagram and choose Functions>>Instrument I/O>>VISA>>VISA Advanced>>VISA Close to add a VISA Close function for the “Stop”:Value Change event structure.

15. Wire the VISA resource out terminal of the VISA Open function to the VISA resource name terminal of the VISA Close function in the event structure; connect the error out terminal of the VISA Open function with the error in terminal of the VISA Close function shown as figure 4-3-5.

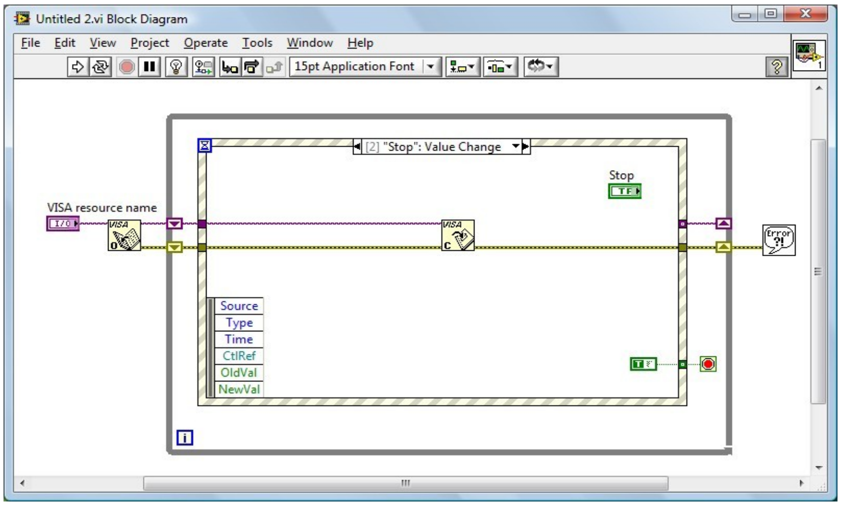


Figure 4-3-5

16. Right-click the blank of the Block Diagram and choose Functions>>Programming>> Structures >>While Loop to add a While Loop structure outside the event structure.

17. Click the Functions palette and choose Functions>>Programming>>Boolean>>True Constant to add a True Constant for the "Stop": Value Change event structure. Wire the True Constant to the stop terminal of the While Loop structure.

18. Click the Functions palette and choose Functions>> Programming>>Dialog& User Interface>> Simpel Error Handler to add a Simple Error Handler function. Wire the error out terminal of the VISA Close function to the error in terminal of the Simple Error Handler function.

19. Right-click the Loop Tunnel terminal where the While Loop structure and the error wire intersected; click the shortcut menu and choose Replace with Shift Register to create a Loop Shift Register Pair with the purpose of replacing the Loop Tunnel. Similarly with a Loop Shift Register Pair to replace the Loop Tunnel where the VISA resource out terminal of the VISA Open function and the VISA resource name terminal of the VISA Close function interested.

20. Adjust the style of the Front Panel shown as figure 4-3-6.

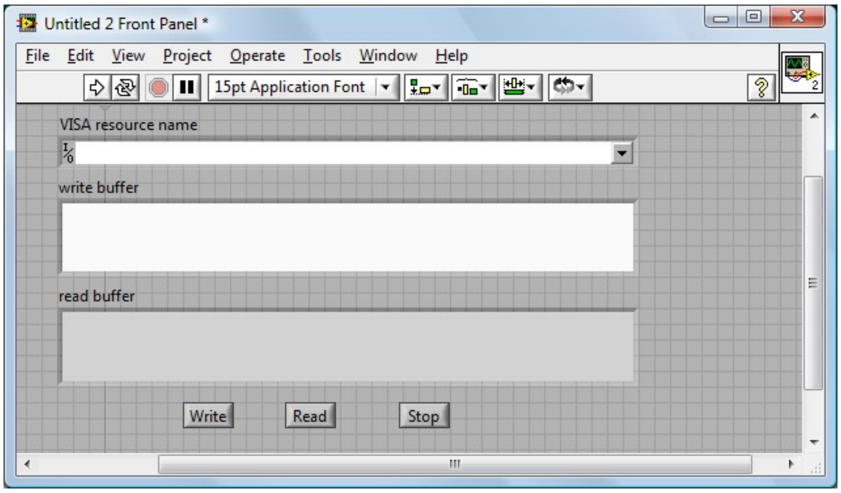
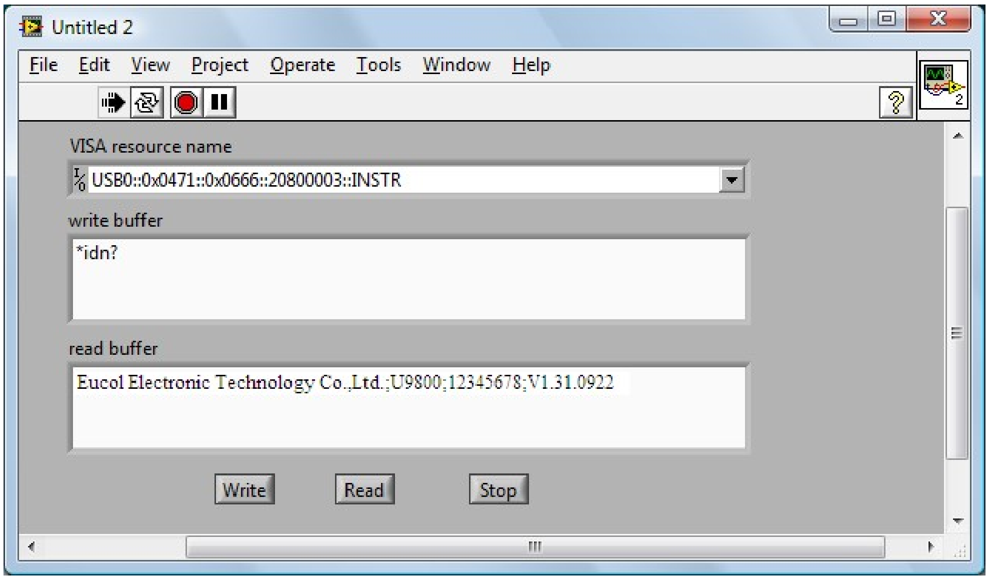


Figure 4-3-6

21. Save the current VI. Before running this VI, select the correct VISA resource name form the VISA resource name pull-down menu.

22. Run the current VI. Input your command or query in the writer buffer, for instance*idn?; click the Write control to send the command or query; then click the Read control to read the returned information. The execution result is shown as figure 4-3-7.



VISA resource name
USB0::0x0471::0x0666::20800003::INSTR

write buffer
*idn?

read buffer
Eucol Electronic Technology Co.,Ltd.;U9800;12345678;V1.31.0922

Figure 4-3-7

23. Click the Stop control to exit this program.