U9800 Series Impulse Winding Tester
Programming Guide

Programming guide provides guidance for user to program this impulse winding 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 impulse winding 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:

Impulse VOLTage, whose long form would be IVOLtage, abbreviates to IVOL.

2 Command System

U9800 series support the following subsystem commands:

◆Common Commands
◆DISPlay Subsystem Commands
◆Impulse VOLtage Subsystem Commands
◆SAMPle Subsystem Commands
◆STATistic Subsystem Commands
◆TRIGger Subsystem Commands
◆COMParator Subsystem Commands
◆Standard WAVeform Subsystem Commands
◆Multi WINding Subsystem Commands
◆MEASure Subsystem Commands
◆WAVeform Subsystem Commands
◆FETCh Subsystem Commands
◆ABORt Subsystem Commands
◆Mass MEMory Subsystem Commands
◆KEY Subsystem Commands
◆SYSTem 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., U9815, 502-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>= {single channel :1 to 300;double channels: 1 to 150; four chanels: 1 to 120}.
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>= {single channel :1 to 300;double channels: 1 to 150; four chanels: 1 to 120}.
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.

12. ***LRN?**

The*LRN? query returns the settings of the instrument.

Query Syntax: *LRN?

Return Format: <data block><NL>

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.

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.

Note: Using the command SYSTem: the FORMat {ASCii | BIN} to set the data back to FORMat.

2.2 DISPlay Subsystem Commands

DISPlay commands are used to control the display system. Figure 2-2-1 shows the DISPlay system command tree.

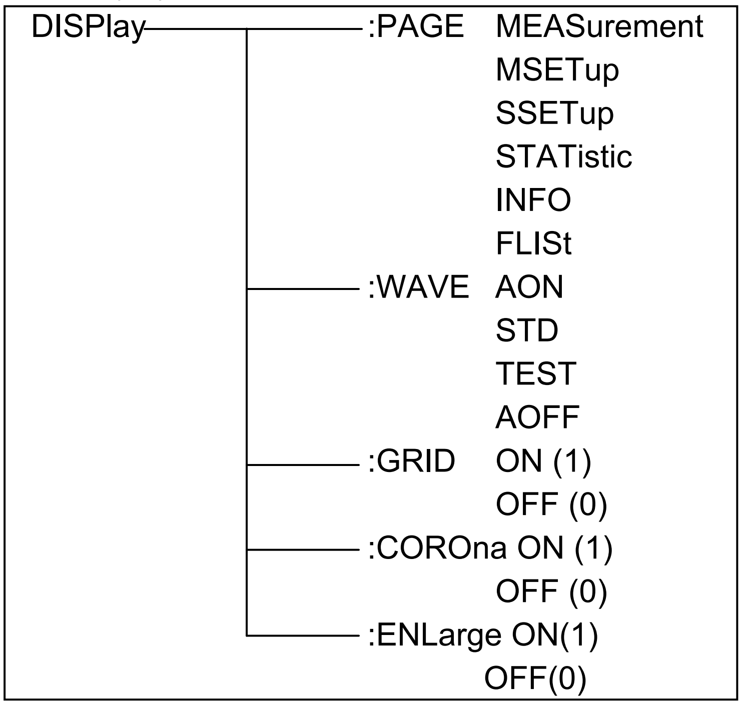


Figure 2-2-1

:PAGE

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

Command Syntax：DISPlay:PAGE <page name>

<page name> as follows:

MEASurement set the instrument display page to Measurement display page.
MSETup set the instrument display page to Measurement Setup page
SSETup set the instrument display page to System Setup page
STATistic set the instrument display page to Statistics 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： {MEAS | MSET | SSET | STAT | INFO | FLIS}<NL>
Note: During the measuring, the query command is ignored.

:WAVeform
The :WAVeform command sets the waveform display mode. The :WAVeform? Query returns the current waveform display mode.
Command Syntax: DISPlay:WAVeform {AON | STD | TSET | AOFF}
Where,
ON Display both the standard and tested waveforms on the LCD screen.
STD Only display the standard waveform on the LCD screen.
TSET Only display the tested waveform on the LCD screen.
AOFF No waveform will be displayed on the LCD screen.
Query Syntax: DISPlay:WAVeform?
Return format: {AON | STD | TSET | AOFF}<NL>

:GRID
The :GRID command sets the grid display mode. The :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>

:COROna
The :COROna command sets the corona display mode. The :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>

:ENLarge
The :ENLarge command sets the test waveform enlarge display on MEAS DISP page. The :ENLarge? query returns the current status of the test waveform on MEAS DISP page.
Command Syntax: DISPlay:ENLarge { {1 | ON} | {0 | OFF}}
Query Syntax: DISPlay:ENLarge?
Return format: {{1 | ON} | {0 | OFF}}<NL>

Note：Where,
1 (decimal49) is equal to ON; 0 (decimal48) is equal to OFF.

2.3 Impulse VOLtage subsystem commands

The Impulse VOLtage subsystem commands set the impulse voltage, average times, impulse voltage auto adjust and delay, etc. Figure 2-3-1 shows the Impulse VOLtage subsystem command tree.

```
Impulse VOLtage──┬─:VOLTage[<n>]    <value>
                 │                   MIN
                 │                   MAX
                 ├─:Break VOLtage    <start>,<stop>,<step>
                 ├─:Break MODe       FAIL
                 │                   END
                 ├─:Break IMPulse    <value>
                 ├─:Test IMPulse     <value>
                 ├─:Dummy IMPulse    <value>
                 └─:Voltage ADJust──── [:STATe]    ON (1)
                                                   OFF (0)
```

Figure 2-3-1

:VOLTage
The :VOLTage command sets the impulse voltage for testing. The :VOLTage? query returns the current impulse voltage valure.
Command Syntax: IVOLtage:VOLTage   {<value> | MAX | MIN}

Where, <value> can be NR1, NR2 or NR3 format followed by KV or V and the value must be between100V to 5000V.

MIN	Set the impulse voltage to 100V
MAX	Set the impulse voltage to 5000V

Query Syntax: IVOLtage:VOLTage?
Return format: <NR1><NL>
NOTE: The query returns no unit, and the default voltage unit is V.

For U9845 impulse winding tester，The IVOLtage:VOLTage<n> command sets the impulse voltage for the specify winding. Where n is from 1 ~ 4.

:Break VOLtage
The :Break VOLtage command sets parameters for Break Test. The :Break VOLtage? query returns the current voltage for Break Test.
Command Syntax: IVOLtage:BVOLtage <start>,<stop>,<step>
Where
<Start> The start voltage for Break Test can be NR1, NR2 or NR3 format followed by unit. The set start voltage ranges from 100V to 5000V and is less than <stop>.
<stop> The end voltage for Break Test can be NR1, NR2 or NR3 format followed by unit. The set end voltage ranges from 100V to 5000V and is larger than <start>.
<step> The voltage step for Break Test can be NR1, NR2 or NR3 format. The set voltage step ranges from 1% to 50% of the <start>.
For example: WrtCmd(“IVOL:BVOL 1000V,2KV,15”); set the start voltage to 1000V, end voltage to 2000V and voltage step to 1000*0.15 =150V.
Query Syntax: IVOLtage:BVOLtage?
Return format: <start>,<stop>,<step><NL>
NOTE: The query for Break VOLtage returns no unit and the default voltage unit is V.

:Break MODe
The :Break MODe command sets the stop mode of the break test. The :Break MODe？query returns the current stop mode of break test.
Command Syntax: IVOLtage:BMODe {FAIL | END}
Query Syntax: IVOLtage:BMODe?
Return format: {FAIL | END}<NL>

:Break IMPulse

The :Break IMPulse command sets the number of the break test impulse.
The :Break IMPluse？ query returns the current number of the break test impulse.
Command Syntax: IVOLtage:BIMPulse {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}
Query Syntax: IVOLtage:BIMPulse?
Return format: {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}<NL>

:Test IMPulse

The :Test IMPulse command sets the average times for testing. The :Test IMPulse? query returns the current average times
Command Syntax: IVOLtage:TIMPulse <value>
Where,
<value1> is NR1 format ranging from 1 to 32 without unit.
Query Syntax: IVOLtage:TIMPulse?
Return format: <NR1><NL>

:Dummy IMPulse

The :Dummy IMPulse command the times of dummy impulses for testing.
The :Dummy IMPulse? query returns the current times of dummy impulses.
Command Syntax: IVOLtage:DIMPulse <value>
Where,
<value> is NR1 format ranging from 0 to 8 without unit.
Command Syntax: IVOLtage:DIMPulse?
Query Syntax: <NR1><NL>

:Voltage ADJust

The :Voltage ADJust command sets the impulse auto adjust function to ON or OFF.
The :Voltage ADJust? query returns the current state of the impulse auto adjust function.
Command Syntax: IVOLtage:VADJust {{1 | ON} | {0 | OFF}}
Query Syntax: IVOLtage:VADJust?
Return format: {{1 | ON} | {0 | OFF}}<NL>

2.4 SAMPle rate subsystem commands

The SAMPle rate subsystem commands set the sample rate and the time base zoom. Figure 2-4-1 shows the SAMPle rate subsystem command tree.

```
SAMPle──┬──:FREQuency[<n>]   <string>
        ├──:EXTend          MIN
        │                   MAX
        │                   <n>
        └──:MODE            NORMal
                            STD
                            QUICk
                            BTEST
```

Figure 2-4-1

:FREQuency

The :FREQuency command sets the sample rate. The :FREQuency? query returns the current sample rate.

Command Syntax: SAMPle:FREQuency {100Msa/s | 50Msa/s | 25Msa/s | 10Msa/s | 5Msa/s | 2.5Msa/s | 1Msa/s | 500Ksa/s | 250Ksa/s | 250Ksa/s | 100Ksa/s}

Query Syntax: SAMPle:FREQuency?

Return format: {100Msa/s | 50Msa/s | 25Msa/s | 10Msa/s | 5Msa/s | 2.5Msa/s | 1Msa/s | 500Ksa/s | 250Ksa/s | 250Ksa/s | 100Ksa/s}<NL>

For U9845，The SAMPle:FREQuency<n> command sets the sample rate of the specify windings. The rang for n is from 1 to 4.

☞**Note:**

1. When the tested device is under test, this command will be ignored.

2. When the standard waveform is tested in the mode of SINGLE CYCLE, if the test has not put an end, this command will be ignored; if the test is nearly finished, this command will select the standard waveforms under different sample rate.

:EXTend

The :EXTend command sets the time base zoom function. The :EXTend? query returns the current value of the time base zoom.

Command Syntax: SAMPle:EXTend {MIN | MAX | <n>}

Where ,

<n> is NR1 format ranging from 0 to 3,

MIN Set to display all 6500 points of the waveform, which means the timer shaft is not extended.

MAX Set to display the first 650 points of the waveform, which means the timer shaft is extended to ten times the original one.

Query Syntax: SAMPle:EXTend?

Return format: {0 | 1 | 2 | 3}<NL>

:MODE

The :MODE command sets the methods for standard wave sampling.

The :MODE? query returns the current methods for standard wave sampling.

Command Syntax: SAMPle:MODE {NORMal | STD | QUICk | BTESt}

Where,

NORMal	sets the standard wave sampling methods to normal mode.
STD	sets the standard wave sampling methods to standard mode.
QUICk	sets the test mode to quick mode.
BTESt	sets the test mode to break test mode.

Query Syntax: SAMPle:MODE?

Return format: {NORMal | STD | QUICk | BTES}<NL>

2.5 STATistic subsystem commands

The STATistic subsystem commands set the statistic function to ON or OFF, clear or save statistic data. Figure 2-5-1 shows the STATistic subsystem command tree.

```
STATistic ─┬─[:STATe]      ON(1)
           │               OFF(0)
           ├─:CLEar
           └─:RESult?
```

Figure 2-5-1

[:STATe]

The [:STATe] command sets the statistic function to ON or OFF. The [:STATe]? query returns the current state of the statistic function.

Command Syntax: STATistic[:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: STATistic[:STATe]?

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

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

:RESult? The :RESult? query returns the current winding’s statistic data.
Return format:
<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>,<NR1>
<NL>
Respectively total count, total passed count, area-size count, area-size passed count, Dif-area count, Dif-area passed count, Corona count, Corona passed count, Dif-phase count, Dif-phase passed count.

2.6 TRIGger subsystem commands

The TRIGger subsystem command group is used to trigger a measurement or to set the trigger mode. Figure 2-6-1 shows the TRIGger subsystem command tree.

```
TRIGger──┬─[:IMMediate]
         ├─:SOURce   MAN
         │          EXTernal
         │          INTernal
         │          BUS
         └─:DELay    <value>
```

Figure 2-6-1

[:IMMediate]
The [: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 U9800 is in the testing state.

:SOURce
The :SOURce command sets the trigger mode. The :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>

:DELay

The :DELay command sets the delay time between two triggers. The :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>

2.7 COMParator subsystem commands

The COMParator subsystem commands set the compare conditions, such as Areasize, Diffzone, Corona and Phasediff. Figure 2-7-1 shows the command tree of the COMParator subsystem commands.

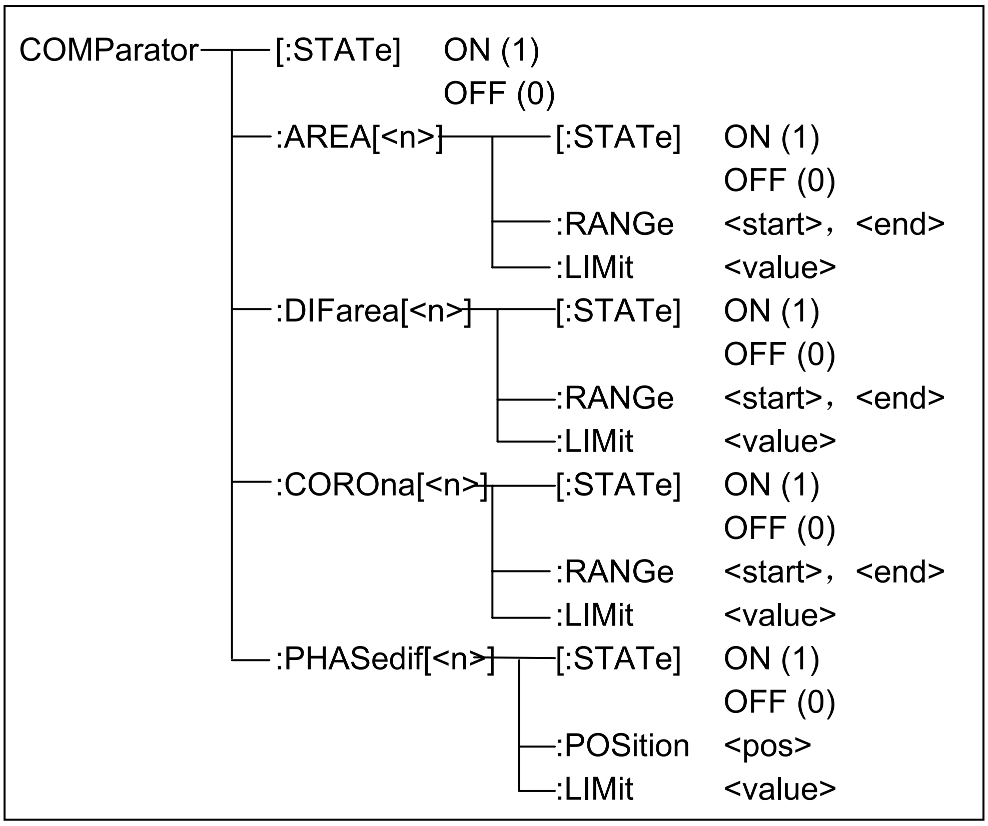


Figure 2-7-1

[:STATe]

The [:STATe] command sets the comparator function to ON or OFF. The [:STATe]? query returns the current comparator state.

Command Syntax: COMParator[:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: COMParator[:STATe]?

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

:AREA[:STATe]

The :AREA[:STATe] command sets the Areasize comparator function to ON or OFF. The :AREA[:STATe]? query returns the current Areasize comparator state.

Command Syntax: COMParator:AREA[:STATe] {{1 | ON} | {0 | OFF}}
Query Syntax: COMParator:AREA[:STATe]?
Return format: {{1 | ON} | {0 | OFF}}<NL>

:AREA:RANGe
The :AREA:RANGe command sets the comparison range of Areasize comparator. The :AREA:RANGe? query returns the current comparison range of Areasize comparator.
Command Syntax: COMParator:AREA:RANGe <start>,<end>
Where,
<start> start point of comparison range. NR1 format, range from 1 to 6500, without unit.
<end> end point of comparison range. NR1 format, range from 1 to 6500, without unit.
NOTE: The end point value should be larger than that of the start point; otherwise an error message will be displayed on the system message line.
Command Syntax: COMParator:AREA:RANGe?
Query Syntax: <start>,<end><NL > start and end are NR1 format.

:AREA:LIMit
The :AREA:DIFFerence command sets the difference limit value of Areasize comparator. The :AREA:DIFFerence? query returns the current difference limit value of Areasize comparator.
Command Syntax: COMParator:AREA:LIMit <value> value ranges from 0.1 to 99.9
Where
<value> can be NR1, NR2 or NR3 format without unit.
For example: WrtCmd(“COMP:AREA:DIFF 2.5”); set the difference limit value to 2.5%.
NOTE: The <value> here is a percent value, For example, input 2.5 for2.5%.
Command Syntax: COMParator:AREA:LIMit?
Query Syntax: <NR3><NL>

:DIFarea[:STATe]

The :DIFarea[:STATe] command sets the Diffzone comparator to ON or OFF. The :DIFarea[:STATe]? query returns the current Diffzone comparator state.
Command Syntax: COMParator:DIFarea[:STATe] {{1 | ON} | {0 | OFF}}
Query Syntax: COMParator:DIFarea[:STATe]?
Return format: {{1 | ON} | {0 | OFF}}<NL>

:DIFarea:RANGe
The :DIFarea:RANGe command sets the comparison range of the Diffzone comparator. The :DIFarea:RANGe? query returns the current comparison range of the Diffzone comparator.
Command Syntax: COMParator:DIFarea:RANGe <start>,<end>
Where,
<start> start point of comparison range. NR1 format, range from 1 to 6500, without unit.
<end> end point of comparison range. NR1 format, range from 1 to 6500, without unit.
NOTE: The end point value should be larger than that of the start point; otherwise an error message will be displayed on the system message line.
Query Syntax: COMParator:DIFarea:RANGe?
Return format: <start>,<end><NL > start and end are NR1 fromat.

:DIFarea:LIMit
The :DIFarea:LIMit command sets the difference limit value of Diffzone comparator. The :DIFarea:LIMit? query returns the current difference limit value of the Diffzone comparator.
Command Syntax: COMParator:DIFarea:LIMit <value> value is from 0.1 to 99.9
Where,
<value> can be NR1, NR2 or NR3 format without unit.
For example: WrtCmd(“COMP:DIF:LIM 2.5”); set the difference limit value to 2.5%.
NOTE: Here <value> is a percent value, for example, input 2.5 for 2.5%.
Query Syntax: COMParator:DIFarea:LIMit?
Return format: <NR3><NL>

:**COROna[:STATe]**

The :COROna[:STATe] command sets the Corona comparator to ON or OFF. The :COROna[:STATe]? query returns the current Corona comparator state.

Command Syntax: COMParator:COROna[:STATe] {{1 | ON} | {0 | OFF}}

Query Syntax: COMParator:COROna[:STATe]?

Return format: <NR1><NL>

:COROna:RANGe

The :COROna:RANGe command sets the comparison range of the Corona comparator. The :COROna:RANGe? query returns the current comparison range of the Corona comparator.

Command Syntax: COMParator:COROna:RANGe <start>，<end>

Where,

<start> start point of comparison range. NR1 format, range from 1 to 6500, without unit.

<end>   end point of comparison range. NR1 format, range from 1 to 6500, without unit.

NOTE: The end point value should be larger than that of the start point; otherwise an error message will be displayed on the LCD screen.

Query Syntax: COMParator:COROna:RANGe？

Return format: <start>,<end><NL>   start,end is NR1 format.

:COROna:LIMit

The :COROna:LIMit command sets the difference limit value of the Corona comparator. The :COROna:LIMit? query returns the current difference limit value of the Corona comparator.

Command Syntax: COMParator:COROna:LIMit <value>

Where,

<value> is NR1 format, range from 1 to 255 without unit.

Query Syntax: COMParator:COROna:LIMit?

Return format: <NR1><NL>

:**PHASedif[:STATe]**

The :PHASediff[:STATe] command sets the Phasediff comparator to ON or OFF. The :PHASediff[:STATe]? query returns the current Phasediff comparator state.

Command Syntax: COMParator:PHASedif[:STATe] {{1 | ON} | {0 | OFF}}
Query Syntax: COMParator:PHASediff[:STATe]?
Return format: {{1 | ON} | {0 | OFF}}<NL>

:PHASedif:POSition

The :PHASediff:POSItion command sets which zero-crossing point is used in the Phasediff comparator. The :PHASediff:POSItion? query returns the zero-crossing position value of the Phasediff comparator.
Command Syntax: COMParator:PHASediff:POSition <value>
 Where,
<value> is Zero-crossing position value, NR1 format, range from 2 to 20 without unit.
Query Syntax: COMParator:PHASediff:POSition?
Return format: <NR1><NL>

:PHASedif:LIMit

The :PHASedif:LIMit command sets the difference limit value of the Phasediff comparator. The :PHASedif:LIMit? query returns the current difference limit value of the Phasediff comparator.
Command Syntax: COMParator:PHASedif:LIMit <value>
 Where,
 <value> can be NR1、NR2 or NR3 format without unit.
For example: WrtCmd(“COMP:PHAS:LIM 2.5”); set the difference limit value to 2.5%.
NOTE: Here <value> is a percent value, for example, input 2.5 for 2.5%.
Query Syntax: COMParator:PHASedif:LIMit?
Return format: <NR3><NL>

For multi-channels instrument, the COMParator:AREA<n>, COMParator:DIFarea<n>, COMParator:COROna<n> and COMParator:PHASedif<n> commands set the specified winding parameters. n range from 1 to 4.

2.8 Standard WAVE subsystem commands

The Standard WAVE subsystem commands load a sampled standard waveform and choosing a sampled standard waveform. Figure 2-8-1 shows the Standard WAVE subsystem command tree.

```
Standard WAVeform ─┬─:LOAD       <data block>
                   ├─:CHOose
                   └─:VOLTage       <value>
```

Figure 2-8-1

:LOAD

The :LOAD command load a sampled standard waveform from PC to the instrument.

Command Syntax: SWAVeform:LOAD <data block>

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.

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.

Note: Using the command WAVeform:FORMat {ASCii | BIN} to set the data back to FORMat.

:CHOose

The :CHOose command is used to select the required standard waveform.

Command Syntax: SWAVeform:CHOose

NOTE: 1. This command is available only on <MEAS DISP> page.
2. This command is valid only when sampling test is finished.

:VOLTage

The :VOLTage command set the impulse voltage value of the loaded standard waveform.

Command Syntax: SWAVeform:VOLTage <volt>

Volt is NR1 format with unti of kV or V

Query Syntax: SWAVeform:VOLTage?
Return format: <NR1><NL>   if there is no standard waveform, it returns the current impulse voltage value of the setting.

2.9 Multi WINding subsystem commands(Only for multi-channels)

The Multi WINding subsystem commands sets the winding type, working mode,绕 standard waveform mode and choose the current test winding. Figure 2-9-1 shows the Multi WINding subsystem commands tree.

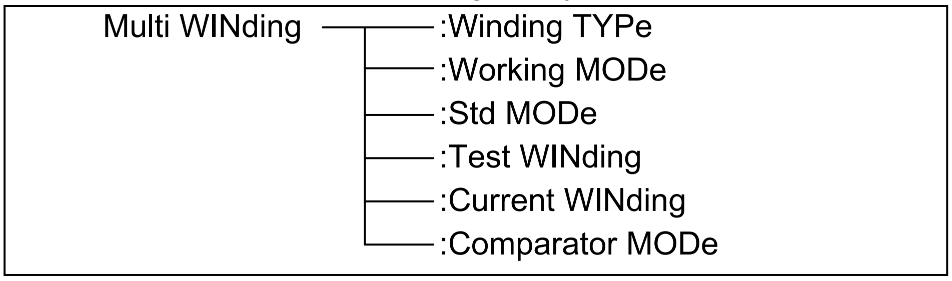


Figure 2-9-1

:Winding TYPe

The :Winding TYPe command sets the tested winding type. The :Winding TYPe? query returns the current winding type.
Command Syntax: MWINding:WTYPe {1COIL | 2COIL | 3COIL | 4COIL | 3P3W | 3P4W}

Where,

1COIL set the tested winding type to single coil,
2COIL set the tested winding type to two coils,
3COIL set the tested winding type to three coils,
4COIL set the tested winding type to four coils,
3P3W set the tested winding type to three-phase three-wire motor
3P4W set the tested winding type to three-phase four-wire motor

Query Syntax: MWINding:WTYPe?
Return format: {1COIL | 2COIL | 3COIL | 4COIL | 3P3W | 3P4W}<NL>

:Working MODe

The :Working MODe command sets the working mode of the instrument. The :Working MODe? query returns the current working mode of the instruement.
Command Syntax: MWINding:WMODe {NORMal | BALance}

Where,

NORMal set the test mode to normal test.

BALance set the test mode to balance test.

Query Syntax: MWINding:WMODe?

Return format: {NORM | BAL}<NL>

:Std MODe

The :Std MODe command sets the mode of standard waveform. The :Std MODe? query returns the current mode of the standard waveform.

Command Syntax: MWINding:SMODe {Single STD | Multi STD}

Where,

Single STD is single standard waveform.

Multi STD is multi standard waveform.

Query Syntax: MWINding:SMODe?

Return format: {SSTD | MSTD}<NL>

:Test WINding

The :Test WINding command sets the test sequence of coil. The :Test WINding? query returns the current test sequence.

Command Syntax: MWINding:TWINding <string>

Query Syntax: MWINding:TWINding?

Return format: <string><NL>

Where, <string> as follows：

```
Balance Mode：   2COIL: AG-BG, CG-DG, AB-CD
                 3COIL: AG-BG-CG, AB-BC-AC
                 4COIL: AG-BG-CG-DG
             Three-phase three-wire: AB-BC-AC
             Three-phase four-wire: AG-BG-CG
Normal Mode：    1COIL: AG, BG, CG, DG
                 2COIL: AG-BG, CG-DG, AB-CD
                 3COIL: AG-BG-CG, AB-BC-AC
                 4COIL: AG-BG-CG-DG
             Three-phase three-wire: AB-BC-AC
             Three-phase four-wire: AG-BG-CG
```

:Current WINding

The :Current WINding command sets the current phase on the MEAS DISP page. The :Current WINding? query returns the current phase on the MEAS DISP page.
Command Syntax: MWINding:CWINding <string>
Query Syntax: MWINding:CWINding?
Return format: <string><NL>
Where, <string> is as follows：

Balance Mode： 2COIL: AG-BG, CG-DG, AB-CD
3COIL: AG-BG, BG-CG, CG -AG / AB-BC, BC-AC, AC -AB
4COIL: AG-BG, BG-CG, CG-DG, DG -AG
Three-phase three-wire: AB-BC, BC-AC, AC -AB
Three-phase four-wire: AG-BG, BG-CG, CG -AG

Normal Mode： 1COIL: AG/BG/CG/DG
2COIL: AG, BG / CG, DG / AB, CD
3COIL: AG, BG, CG / AB, BC, AC
4COIL: AG, BG, CG, DG
Three-phase three-wire: AB, BC, AC
Three-phase four-wire: AG, BG, CG

:Comparator MODe
The :Comparator MODe command sets the comparator mode of the instrument. The :Comparator MODe? query returns the current comparator mode.
Command Syntax: MWINding:CMODe {PUBLic | PRIVate}
Where,
PUBLic is all the coils using the same parameters and standard waveform.
PRIVate is all the coils can used the different parameters and standard waveform.
Query Syntax: MWINding:CMODe?
Return format: {PUBL | PRIV}<NL>

2.10 MEASure subsystem commands

The MEASure subsystem commands sets the measurement range of voltage, frequency and time. Figure 2-10-1 shows the MEASure subsystem command tree.

```
MEASure ─┬─:VOLTage <upper>,<lower>
         ├─:FREQuency <start>,<end>
         └─:TIME <start>,<end>
```

Figure 2-10-1

:VOLTage

The :VOLTage command sets the measurement range of voltage.
The :VOLTage? query returns the current voltage range.
Command Syntax: MEASure:VOLTage <upper>,<lower>

Where,

<upper> is the upper limit of voltage, NR1 format, ranging from -120 to 120 without unit.

< lower> is the lower limit of the voltage, NR1 format, ranging from -120 to 120 without unit.

NOTE: The lower limit must be less than the upper one; otherwise an error message will be displayed on the system message line.

Query Syntax: MEASure:VOLTage?

Return format: <upper>,<lower><NL> Both upper limit and lower limit are NR1 format.

:FREQuency

The :FREQuency command sets the frequency range. The :FREQuency? query returns the currently set frequency range.

Command Syntax: MEASure:FREQuency <start>,<end>

Where,

<start> is the start point of frequency, NR1 format and ranging from 1 to 650 without unit.

<end> is the end point of frequency, NR1 format and ranging from 1 to 650 without unit.

ⓘNOTE: The value of the end point must be larger than that of the start point; otherwise an error message will be displayed on the system message line.
☞NOTE: The measurement ranges of frequency and time are the same, thus the time range will vary with the frequency range.
Query Syntax: MEASure:FREQuency?
Return format: <start>,<end><NL>   Both <start> and <end > are NR1 format.

:TIME
The :TIME command sets the time range. The :TIME? query returns the current time range.
Command Syntax: MEASure:TIME <star>,<end>
Where,
<start> is the start point of time, NR1 format and ranging from 1 to 650 without unit.
<end> is the end point of time, NR1 format and ranging from 1 to 650 without unit.
ⓘNOTE: The value of the end point must be larger than that of the start point; otherwise an error message will be displayed on the system message line.
☞NOTE: The measurement ranges of frequency and time are the same, thus the time range will vary with the frequency range.
Query Syntax: MEASure:TIME?
Return format: <start>,<end><NL >   Both <start> and <end> are NR1 format.

2.11 *WAVeform subsystem commands*

The WAVeform subsystem commands are used to read the related parameters of test waveforms and standard waveforms. Figure 2-11-1 shows the WAVeform subsystem command tree.

```
WAVeform ──┬──:FORMat       ASCii
           │                BIN
           ├──:POINts       ALL
           │                PEAK
           ├──:SOURce       STD
           │                TEST
           ├──:XINcrement
           ├──:YINcrement
           ├──:READy
           └──:DATA
```

Figure 2-11-1

:FORMat

The :FORMat command set the format waveform data. The :FORMat? Query returns the current data format. ASCii means string format, BIN means binary string format.

Command Syntax: WAVeform:FORMat {ASCii | BIN}

Query Syntax: WAVeform:FORMat?

Return format: {ASC | BIN}<NL>

:POINts

The : POINts command set waveform data length to be transferred waveform.

The : POINts? Query returns the current data length.

Command Syntax: WAVeform:POINts {ALL | PEAK}

Where,

ALL return all of the data. It is 6500 point.

PEAK return the peak envelope data. It is 650 point.

Query Syntax: WAVeform:POINts?

Return format: {ALL | PEAK}<NL>

:SOURce

The :SOURce command sets the waveform data source. The :SOURce?
 query returns the current waveform data source.
Command Syntax: WAVeform:SOURce {STD | TEST}
 Where,
 STD sets the waveform data source to the standard waveform.
 TEST sets the waveform data source to the test waveform.
Query Syntax: WAVeform:SOURce?
Return format: {STD | TEST}<NL>

: XINcrement?
The :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: WAVeform:XINcrement?
Return format: <NR3><NL>

: YINcrement?
The:YINcrement? command returns the vertical voltage value of vertical scale/25.
Query Syntax: WAVeform:YINcrement?
Return format: <NR3><NL>

:READy?
Query Syntax: WAVeform:READy?
Return format: {0 | 1}<NL>

:DATA?
The :DATA? query returns the data for the waveform.
Query Syntax: WAVeform:DATA?
Return format: <data block><NL>

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.

2.12 FETCh subsystem commands

The FETCh subsystem commands are used to output waveform data, comparison results, output voltage, frequency and time measurement result. Figure 2-12-1 shows the FETCh subsystem command tree.

```
FETCh ┬──:Standard WAVeform?
      ├──:Test WAVeform?
      ├──:Comparator RESult?
      ├──:VOLTage?
      ├──:FREQuency?
      ├──:TIME?
      ├──:PEAK?
      └──:Fail WINding?
```

Figure 2-12-1

:Standard WAVeform?

The :Standard WAVeform? query outputs the current standard waveform data.
Query Syntax: FETCh SWAVeform?

:Test WAVeform?

The :Test WAVeform? Query outputs the latest tested waveform data.
Query Syntax: FETCh TWAVeform?

:Comparator RESult?

The :Compartor RESult? query returns the latest comparison result.
Query Syntax: FETCh CRESult?
Return format:
There are two cases as follows:

1. If the comparator function or four comparison methods is set to OFF, or no waveform data available, the query response will be <NR1><NL^END>, here NR1 is 2.
2. If the comparator function is set to on, the return format will be

<NR1, NR3, NR3, NR1, NR3><NL^END>, the first NR1 is the general comparison result: 1 (PASS) or 0 (FAIL). The following four data are the comparison results corresponding to each comparator: AREA SIZE, DIFF ZONE, CORONA and PHASE DIFF.

: VOLTage?

The :VOLTage? query returns the current voltage value in the set voltage range. Refer to MEASure subsystem commands for the voltage range settings.

Query Syntax: FETCh:VOLTage?

Return format: <NR1><NL> ☞ **NOTE: The unit of the returned voltage is V.**

:FREQuency?

The :FREQuency? query returns the current frequency result in the set frequency range. Refer to MEASure subsystem commands for the frequency range settings.

Query Syntax: FETCh:FREQuency?

Return format: <NR3><NL>

☞**NOTE: The unit of the returned frequency is Hz. If the start point is overlapped with the end cursor completely, then 9.9E37 will be returned.**

:TIME?

The :TIME? query returns the current time result in the set time range. Refer to MEASure subsystem commands for the time range settings.

Query Syntax: FETCh:TIME?

Return format: <NR3><NL>

☞ **NOTE: The unit of the returned time is s.**

:PEAK?

The :PEAK? query returns the current peak voltage value of the test waveform.

Query Syntax: FETCh:PEAK?

Return format: <NR3><NL>

:Fail WINding?
The :Fail WINding? query returns the phase of failed test winding. Query Syntax: FETCh:FWINding?
Return format: <string><NL>

2.13 ABORt subsystem command

U9800 will abort the current measurement as soon as the ABORt command is received.
Command Syntax: ABORt

2.14 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.

:LOAD:STATe
The :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 720 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 720, 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.

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

The :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 720 , 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，"#U9800*"")；

ⓘ NOTE: U9800 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.

:DELete:STATe

The :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 720, NR1 format without unit.

For example: WrtCmd("MMEM:DEL:STAT 1"); delete file 1.

ⓘ NOTE: U9800 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.

2.15 KEY subsystem commands

KEY commands are used to control the keys and knobs on the operation panel of U9800.

KEY:LOCal	enable the front panel operation
KEY:MEASure	enter into the <MEAS DISP> page
KEY:SETup	enter into the <MEAS 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:ADD	turn the knob clockwise
KEY:SUB	turn the know counter-clockwise
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 6

2.16 SYSTem subsystem commands

The SYSTem subsystem commands are used for system setups. Figure 2-16-1 shows the SYSTem subsystem commands tree.

```
SYSTem ──┬──:DATE
         ├──:TIME
         ├──:Save TYPe
         ├──:SETup
         └──:FORMat
```

Figure 2-12-1

:DATE

The :DATE command sets the date of system. The :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>

:TIME

The :TIME command sets the time of system. The: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>

:Save TYPe

The :Save TYPe command sets the file type when pressing SAVE key. The :Save TYPe? query returns the current file type.
Command Syntax: SYSTem:STYPe {CSV | GIF | BMP8 | BMP24 | PNG}
Query Syntax: SYSTem:STYPe?
Return format: {CSV | GIF | BMP8 | BMP24 | PNG}<NL>

:FORMat

The :FORMat command set the data format. The :FORMat? query returns the current data format.
Command Syntax: SYSTem:FORMat {ASCii | BIN}

Where, ASCii is string data format, BIN is binary data format.

Query Syntax: SYSTem:FORMat?
Return format: {ASCii | BIN}<NL>

:SETup

The :SETup command sets parameters for download(excluding standard waveform data). The :SETup? query returns the parameters.

Command Syntax: SYSTem:SETup <data block>

Where, data block is the parameters returned by SYSTem:SETup? query.

Query Syntax: SYSTem:SETup?   This command is equal to *LRN?

Return format: <data block><NL>

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.

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.

Note: Using the command SYSTem: the FORMat {ASCii | BIN} to set the data back to FORMat.

3 Error and warning message

The bus commands may have some spelling errors, syntax errors or wrong parameters. U9800 executes a command after the command is analyzed. If one of above errors occurs, U9800 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 MEAS should be DISP:PAGE MEAS
Data out of range	The data is out of range. For example: IVOLT 5500, the impulse voltage 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: IVOLT: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 MSET, where（.）should be（:）.
Trigger ignored	When U9800 is in the testing state, all trigger signals will be ignored.
Command ignored	Some command may be ignored. For example: DISP:PAGE MSET When U9800 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 U9800 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.

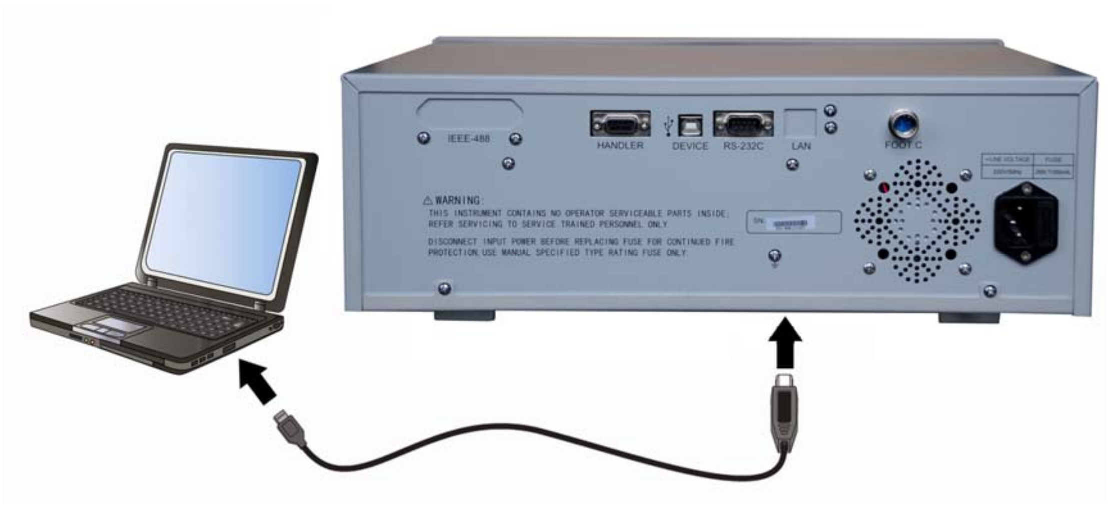
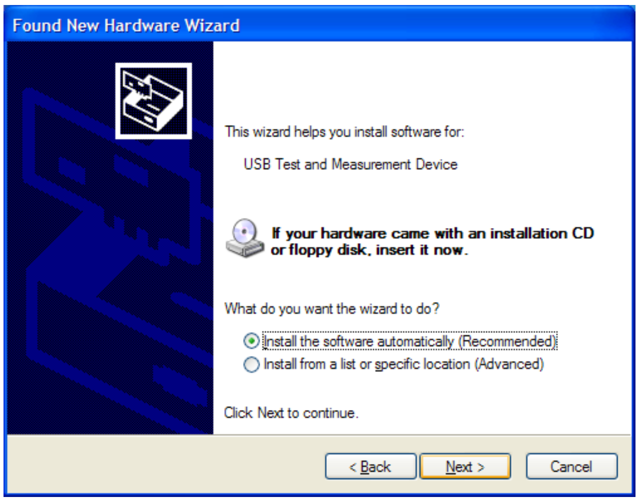


Figure 4-1

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)

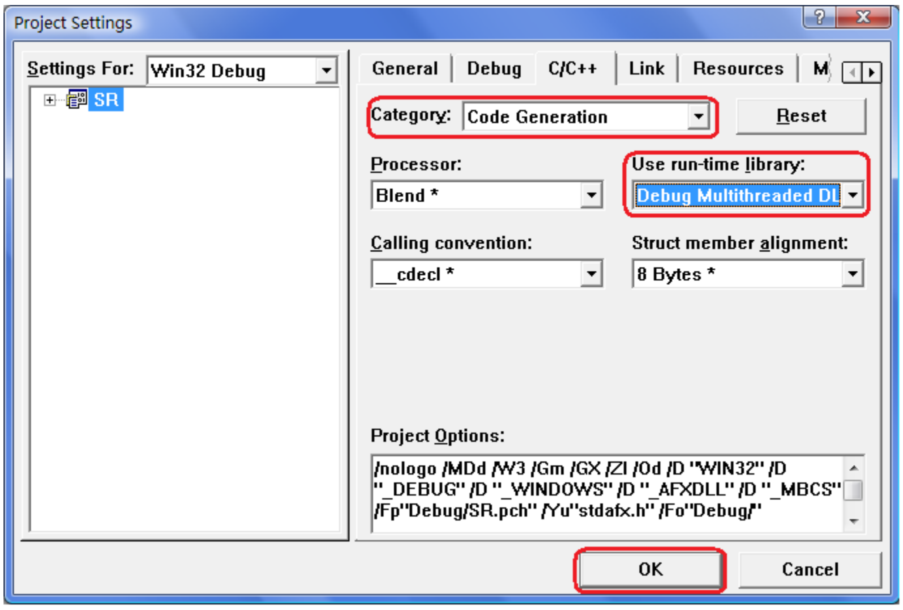
Click Next to continue.

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:

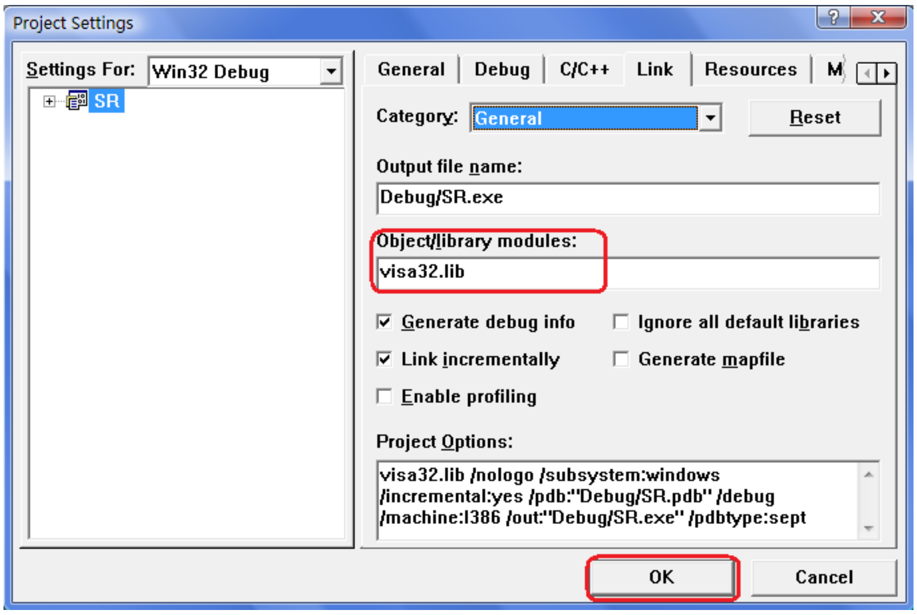
```
/nologo /MDd /W3 /Gm /GX /ZI /Od /D "WIN32" /D
"_DEBUG" /D "_WINDOWS" /D "_AFXDLL" /D "_MBCS"
/Fp"Debug/SR.pch" /Yu"stdafx.h" /Fo"Debug/"
```

Figure 4-1-1

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

4. 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.



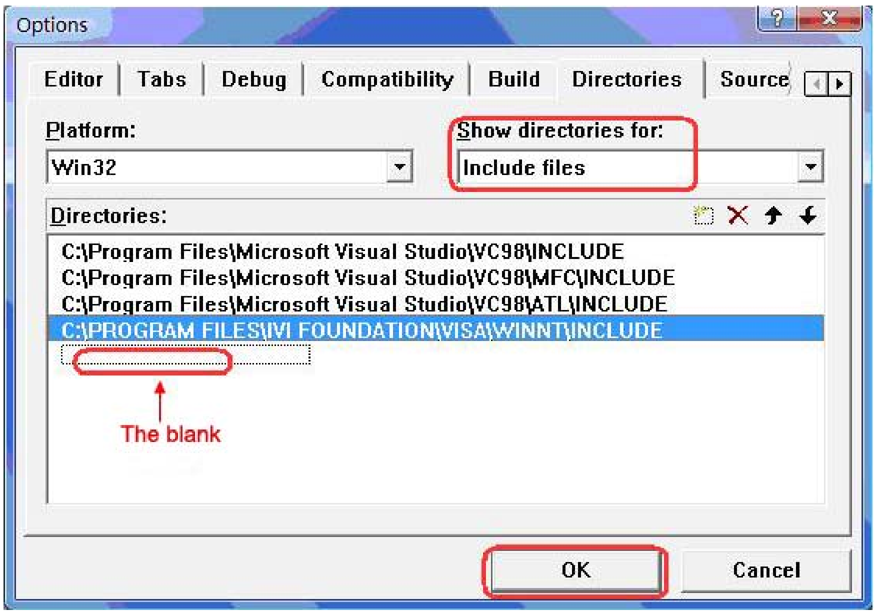
Output file name: Debug/SR.exe

Object/library modules: visa32.lib

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.

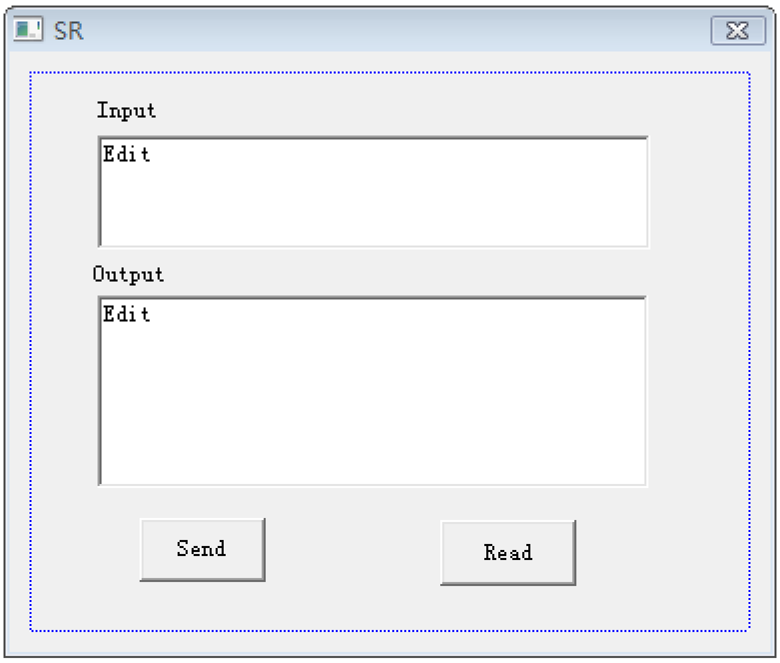


Figure 4-1-4

(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.

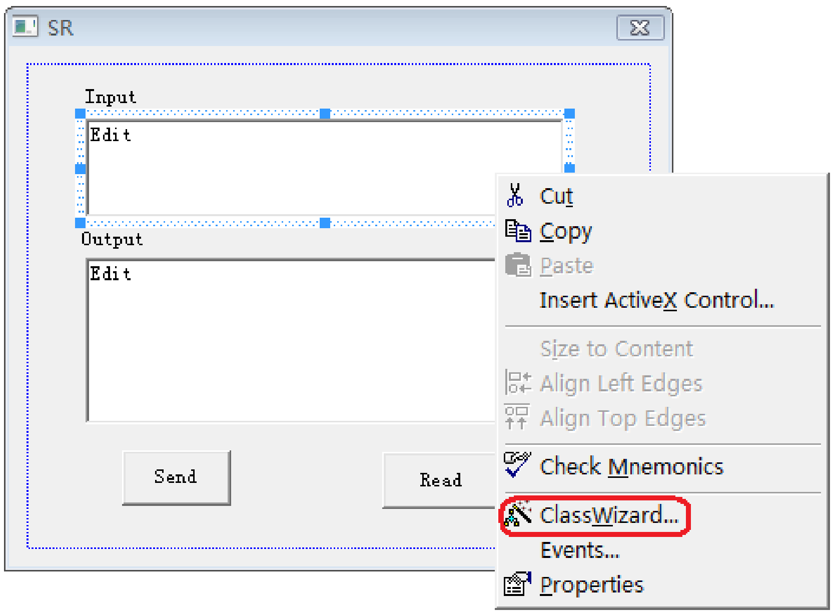
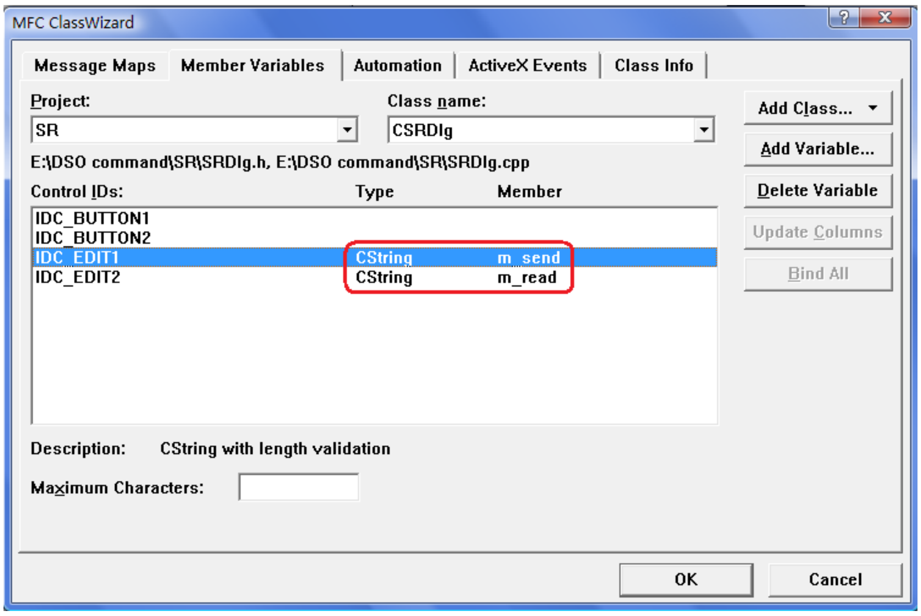


Figure 4-1-5


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);
```

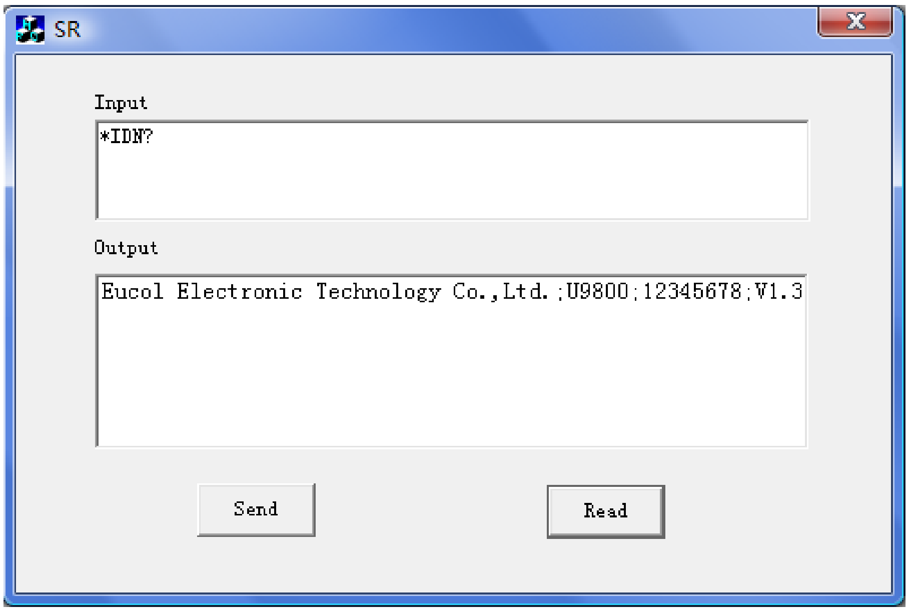
(6) Add the following codes in Read.

```
//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.

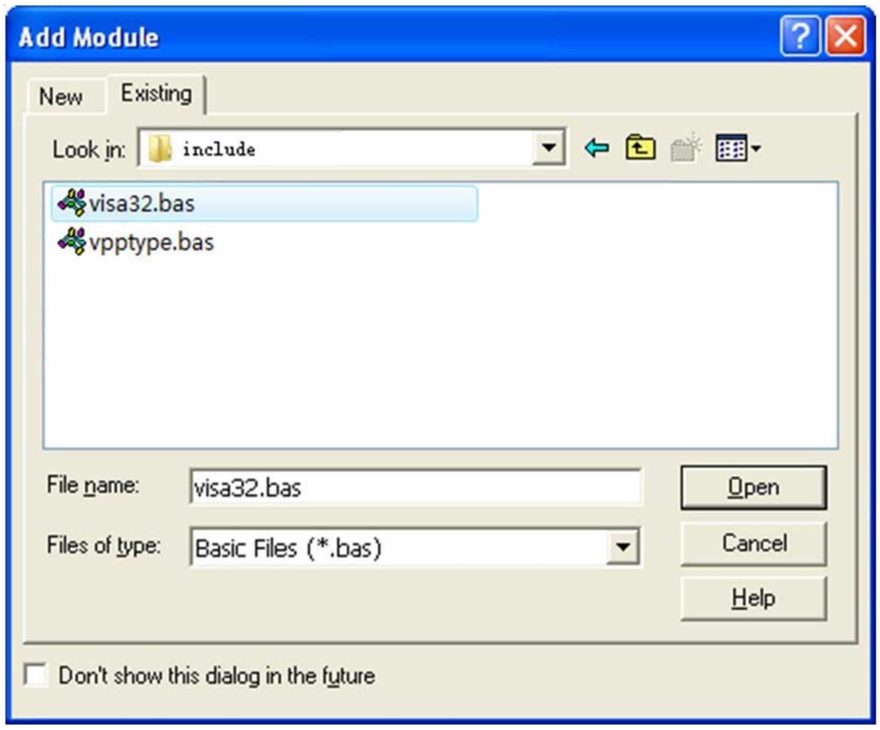


Figure 4-2-1

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.

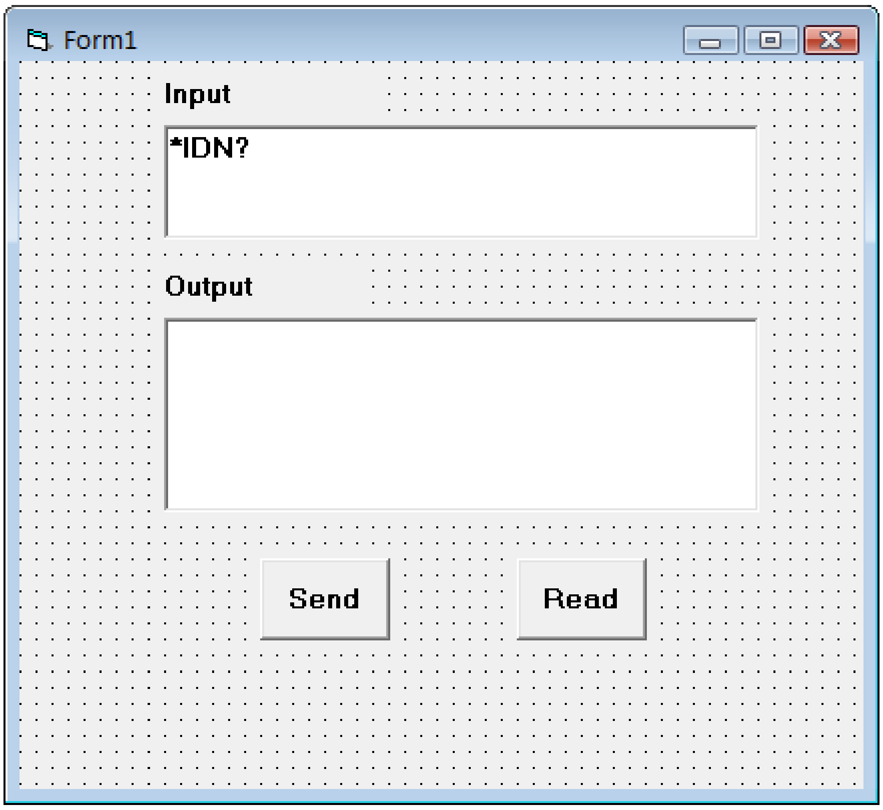


Figure 4-2-2

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)
```

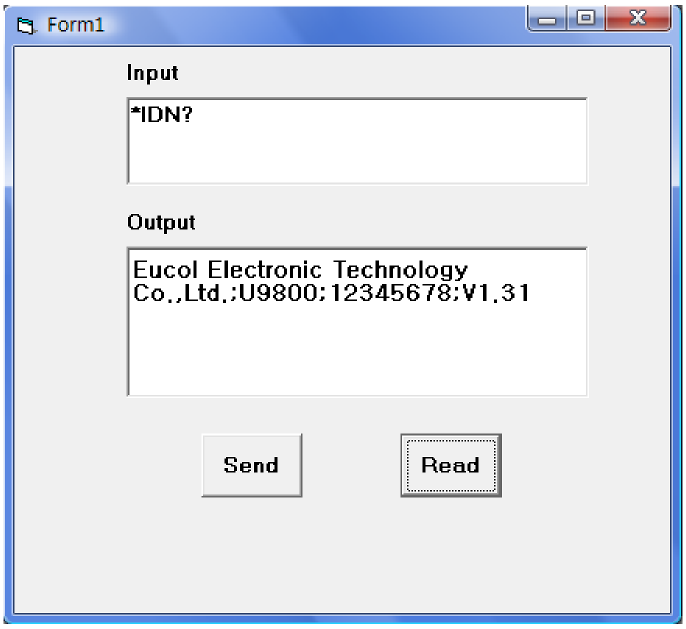
```
Txt_output.Text = strRes
End Sub

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.

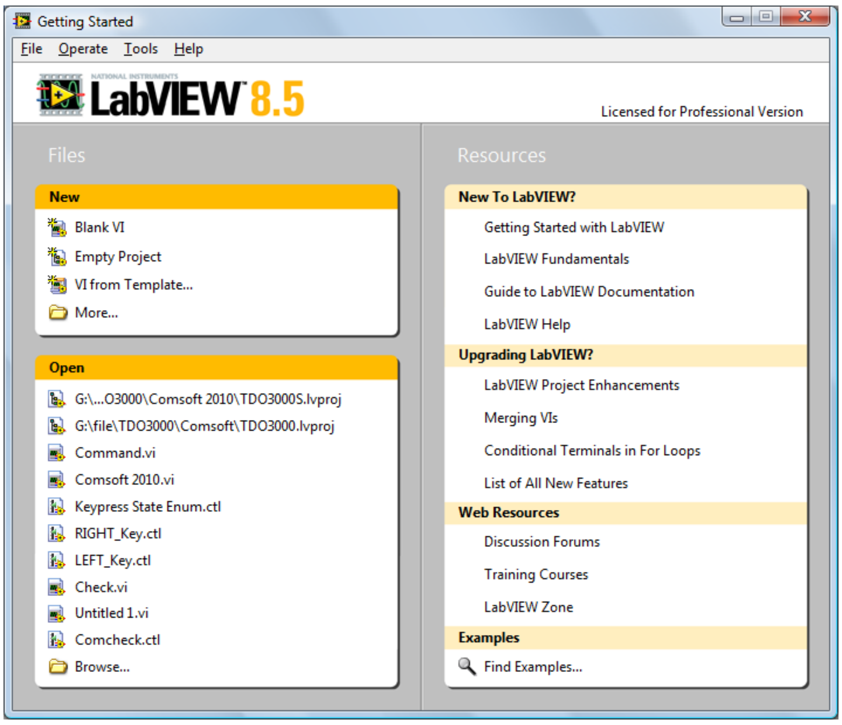


Figure 4-3-1

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.

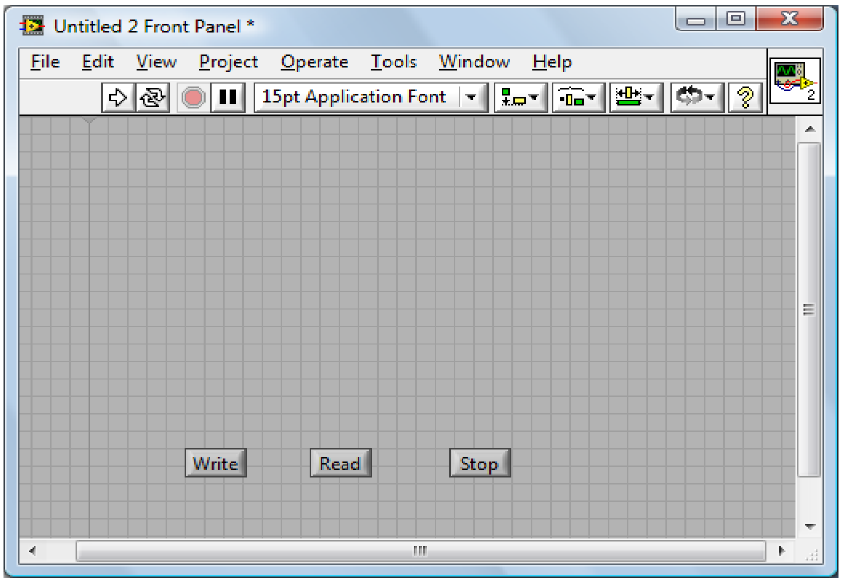


Figure 4-3-2

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.

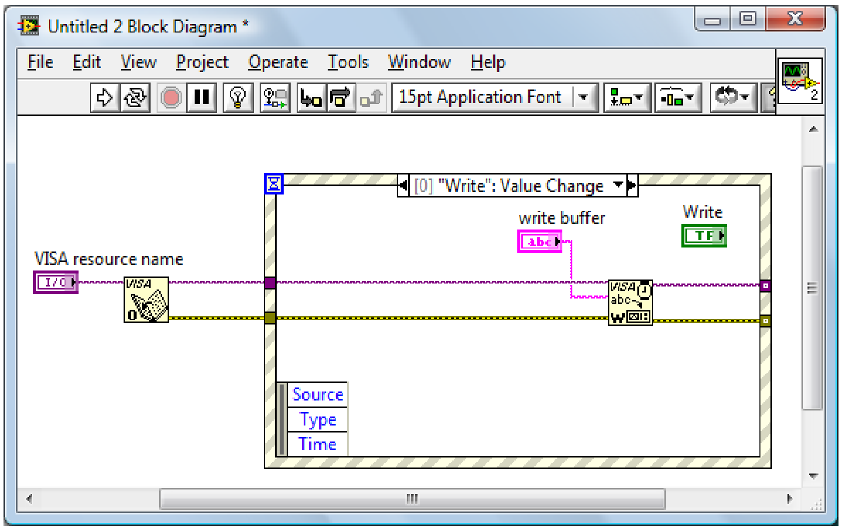

Untitled 2 Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
[0] "Write": Value Change
write buffer
Write
VISA resource name
Source
Type
Time

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.

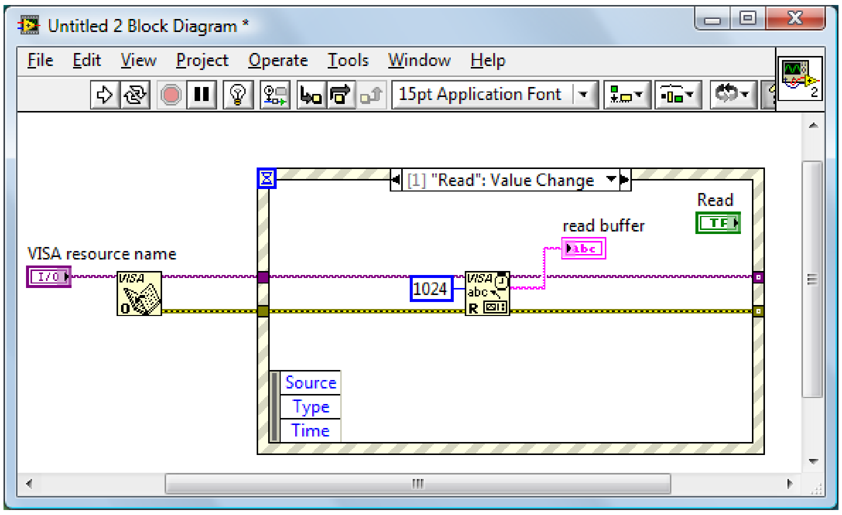


Figure 4-3-4

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.

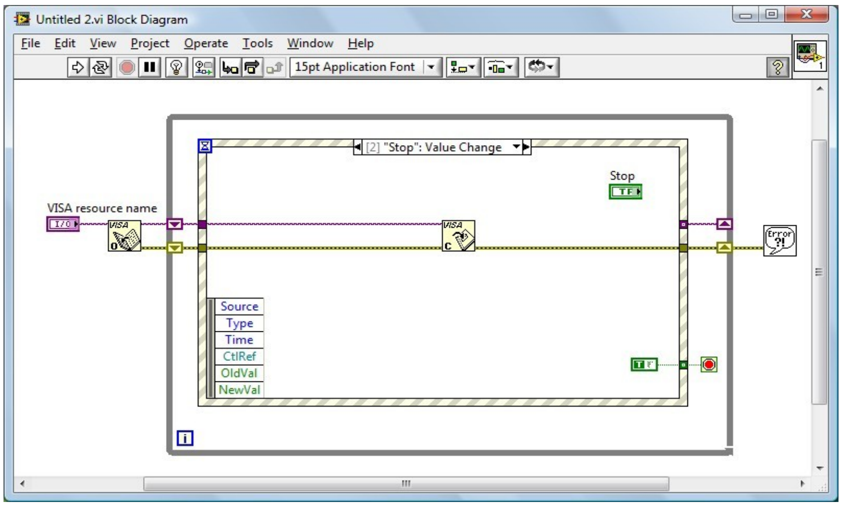


Figure 4-3-5

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.

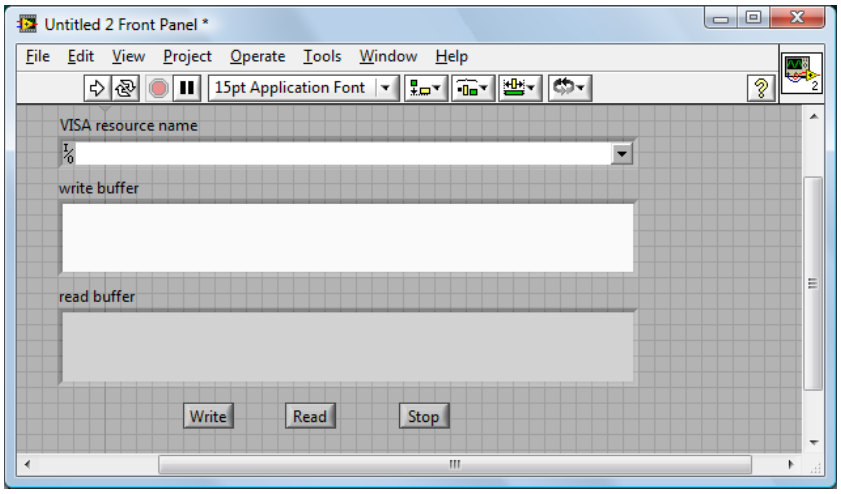
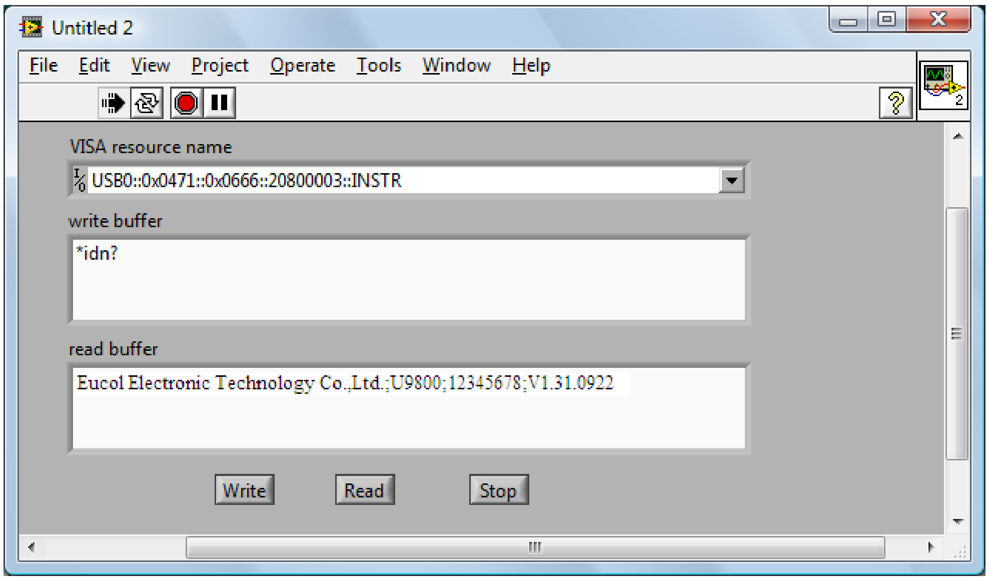


Figure 4-3-6

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.