U2832 Series LCR Meter Programming Guide

CHANGZHOU EUCOL ELECTRONIC TECHNOLOGY CO.,LTD.
NO.1,North Qingyang Road, Tianning District, Changzhou, Jiangsu Province, China
Tel: +86-519-85505199
Fax: +86-519-85505169
E-mail: sales@eucol.com.cn
www.eucol.com.cn

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:
Save TYPe, whose long form would be STPYe, abbreviates to STYP.

2 Command System

U2832 series support the following subsystem commands：

- Common Commands
- DISPlay Subsystem Commands
- ABORt Subsystem Commands
- INITiate Subsystem Commands
- TRIGger Subsystem Commands
- FUNCtion Subsystem Commands
- FREQuency Subsystem Commands
- VOLTage Subsystem Commands
- APERture Subsystem Commands
- Source RESister Subsystem Commands
- FETCh Subsystem Commands
- CORRection Subsystem Commands
- COMParator Subsystem Commands
- LIST Subsystem Commands
- MMEMory 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 the instrument Settings

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., U2832, 120F14105, V1.23.01.14B2

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 range is from 1 to 100.
Query Syntax: *RCL <value>
Example:
*RCL 1 Restore the state of the instrument from the specified Setup01.
The MMEMory command has the same function.

4. *SAV <value>

The *SAV command stores the current state of the instrument to the specified setup file position. <value> is from 1 to 100.
Command Syntax: *SAV <value>[,“name”], name is the file name,the length of name should be less then 17 charactors.
Example:
*SAV 1 Store the current state of the instrument to the specified Setup01.
The MMEMory command has the same function.

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
Note：This command should be used with INITiate Subsystem command.

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	Operation is complete.

		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. *TST?

The *TST? Query returns the test results of the instruments.

Query Syntax: *TST?

Return Format: {1 | 0}<NL>

2.2 DISPlay Subsystem Commands

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

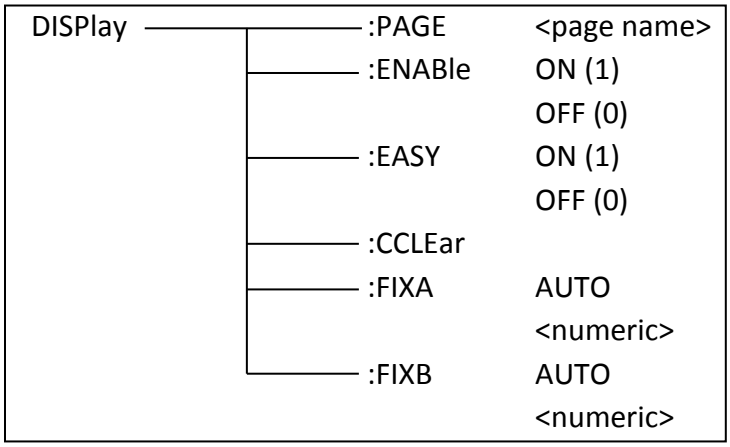


Figure 2-2-1

: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.
BDISplay	set the instrument display page to Bin display page.
LIST	set the instrument display page to LIST display page.
MSETup	set the instrument display page to Measurement Setup page.
LTABle	set the instrument display page to Limit Table page.
LSETup	set the instrument display page to List Setup page.
SYSTem	set the instrument display page to System 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：{MEAS | BDIS | LIST | MSET | LTAB | LSET | SYST | INFO | FLIS}<NL>

:ENABle

The :ENABle command set up the readings displayed or undisplayed.

The :ENABle? Command returns the current state of the measurement display page.

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

Query Command：DISPlay:ENABle?

Return Format：{1 | 0}<NL>

:EASY

The :EASY command set up the easy test mode as ON or OFF.

The :EASY? Command returns the current test mode.

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

Query Command：DISPlay:EASY?

Return Format：{1 | 0}<NL>

Note：Where,

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

:CCLear

The :CCLear command clear the error information displayed on the measurement display page.

Command Syntax：DISPlay:CCLear

:FIXA

The :FIXA command sets the decimal lock value for the first parameter.

Command Syntax：DISPlay:FIXA {<numeric> | AUTO}

The value of numeric ranges from 0 to 5, 0 and AUTO indicate automatic.

:FIXB

The :FIXB command sets the decimal lock value for the second parameter.

Command Syntax：DISPlay:FIXB {<numeric> | AUTO}

The value of numeric ranges from 0 to 5, 0 and AUTO indicate automatic.

2.3 ABORt Subsystem Commands

U2832 will abort the current measurement as soon as the ABORt command is received.

Command Syntax：ABORt

2.4 INITiate Subsystem Commands

The INITiate subsystem commands is used to trigger the instrument together with *TRG command. Figure 2-4-1 shows the commands tree.

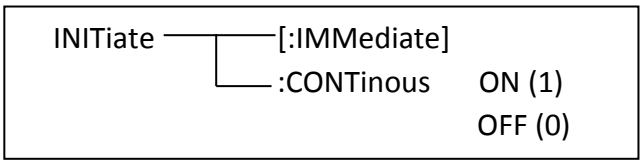


Figure 2-4-1

[:IMMediate]

The [:IMMediate] command sets the instrument triggered only one time after receive the *TRG command.

Command Syntax：INITiate [:IMMediate]

: CONTinous

The [:CONTinous] command sets the instrument triggered continuously after receive the *TRG command.

Command Syntax：INITiate:CONTinous { {1 | ON} | {0 | OFF}}

Query Command：INITiate:CONTinous?

Return Format：{1 | 0}<NL>

2.5 TRIGger Subsystem Commands

The TRIGger subsystem command group is used to trigger a measurement or to set the trigger mode. Figure 2-5-1 shows the TRIGger subsystem command tree.

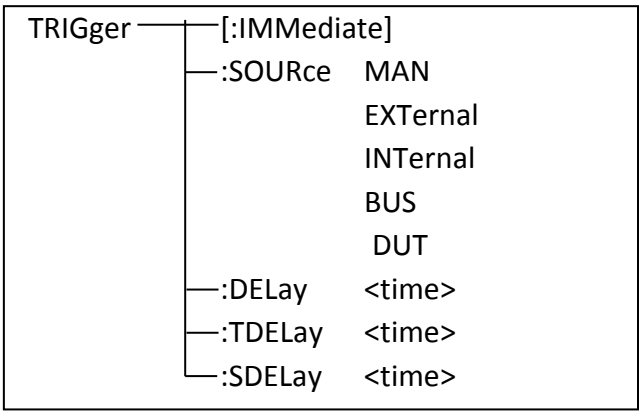


Figure 2-5-1

[:IMMediate]

The [:IMMediate] command triggers a measurement.

Command Syntax：TRIGger[:IMMediate]

ⓘ NOTE: The TRIGger[:IMMediate] command, is available only in the <MEAS DISP> ,<Bin DISP> and <LIST>page, will be ignored when U2832 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 | HOLD | EXTernal | INTernal | BUS}

Where,

MANual or HOLD Triggered by pressing the TRIGGER button.

EXTernal Triggered by the HANDLER interface.

INTernal Automatically triggered by the instrument.

BUS Triggered by RS232 interface, USB interface.

Query Command：TRIGger:SOURce?

Return Format：{MAN | HOLD | EXT | INT | BUS}<NL>

:DELay or **:TDELay**

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

The :DELay? or :TDELay query returns the current delay time. Delay time range from 0 to 60s with step of 1ms.

Command Syntax：TRIGger:DELay <time>　where　<time> is NR1, NR2 or NR3 format.
Query Command：TRIGger:DELay?
Return Format：<NR3><NL>

:SDELay
The :SDELay command sets the delay time between two steps. The :SDELay? query returns the current delay time. Delay time range from 0 to 60s with step of 1ms.
Command Syntax：TRIGger:SDELay <time>　where　<time> is NR1, NR2 or NR3 format.
Query Command：TRIGger:SDELay?
Return Format：<NR3><NL>

2.6　FUNCtion Subsystem Commands

The FUNCtion subsystem commands are mainly used to set “function”,”range”,”self-calibration mode”,etc.. Figure 2-6-1 shows the function subsystem commands tree.

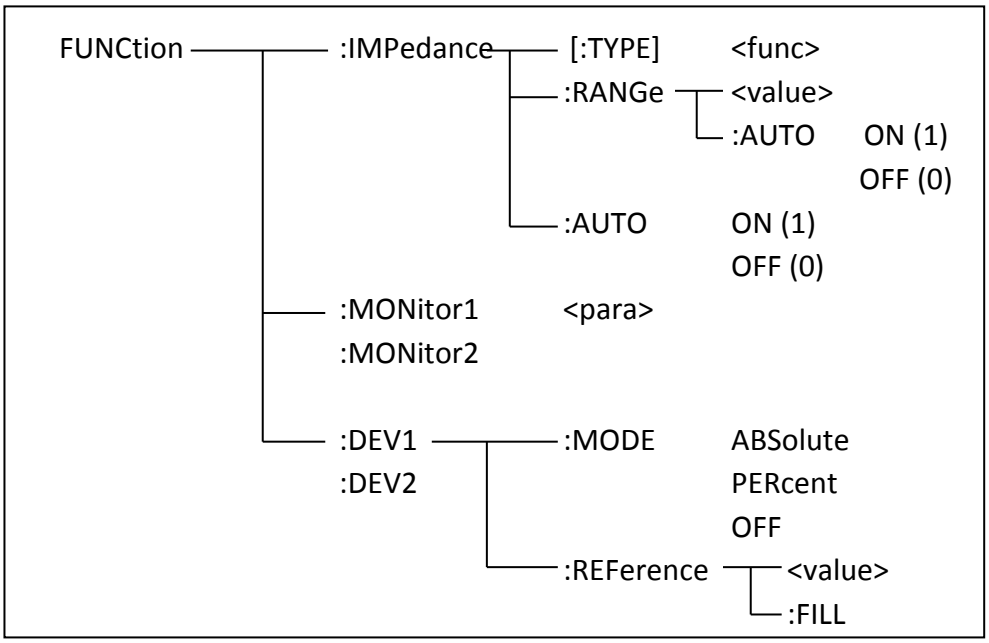


Figure 2-6-1

IMPedance[:TYPE]

The FUNCtion:IMPedance command is used to set instrument functions. The FUNCtion:IMPedance? query returns the current function parameters.

Command Syntax：FUNCtion:IMPedance[:TYPE] {CPD | CPRP | CSD | CSRS | LSQ | LSRS |LPQ | LPRP | RX | ZTD | ZTR | GB| DCR|LSDCR|LPDCR}

<function> can be one of the following items.

CPD Set the function as Cp-D CPRP Set the function as Cp-Rp

CSD Set the function as Cs-D CSRS Set the function as Cs-Rs

LSQ Set the function as Ls-Q LSRS Set the function as Ls-Rs

LPQ Set the function as Lp-Q LPRP Set the function as Lp-Rp

RX Set the function as R-X ZTD Set the function as Z-θ°

ZTR Set the function as Z-θr GB Set the function as G-B

DCR Set the function as DC resistance LSDCR Set the function as Ls-DCR

LPDCR Set the function as Lp-DCR

Query Command：FUNCtion:IMPedance[:TYPE]?

Return Format：{CPD|CPRP|CSD|CSRS|LSQ|LSRS|LPQ|LPRP|RX|ZTD |ZTR|GB| DCR|LSDCR|LPDCR }<NL>

IMPedance:RANGe

The FUNCtion:IMPedance:RANGe command is used to set the range. The FUNCtion:IMPedance:RANGe? query returns the current range.

Command Syntax：FUNCtion:RANGe { 10 | 31.6 | 100 | 1000 | 10000 | 100000 | 10ohm | 31.6ohm | 100ohm |1kohm| 10kohm | 100kohm }

Query Command：FUNCtion:RANGe?

Return Format：{10|31.6|100|1000|10000|100000}<NL>

IMPedance:RANGe:AUTO

The FUNCtion:IMPedance:RANGe:AUTO command is used to set the automatic range selection status. The FUNCtion:IMPedance:RANGe:AUTO? query returns the current range status.

Command Syntax：FUNCtion:RANGe:AUTO { {1 | ON} | {0 | OFF}}

Query Command：FUNCtion:RANGe:AUTO?

Return Format：{1 | 0}<NL>

IMPedance:AUTO

The FUNCtion:IMPedance:AUTO command is used to set instrument function as Auto LCR. The FUNCCtion:IMPedance:AUTO? query returns the current status.

Command Syntax：FUNCtion:IMPedance:AUTO { {1 | ON} | {0 | OFF}}

Query Command：FUNCtion:IMPedance:AUTO?

Return Format：{1 | 0}<NL>

:MONitor<n>

The FUNCtion:MONitor command is used to set the function monitor. The FUNCTtion:MONitor? query returns the current function monitor.

Command Syntax：FUNCtion:MONitor<n> {CS | CP | LS | LP | RS | RP | R | D | Q | Z | RAD | DEG | X | Y | G | B | V | I| OFF }

Query Command：FUNCtion:MONitor<n>?

Return Format：{CS|CP|LS|LP|RS|RP|R|D|Q|Z|RAD|DEG|X|Y|G|B|V|I| OFF }<NL>

:DEV<n>:MODE

The FUNCtion:DEV<n>:MODE command is used to set the deviation measurement mode. The FUNCtion:DEV<n>:MODE? query returns the current deviation measurement mode.

Command Syntax：FUNCtion:DEV<n>:MODE { ABSolute | PERcent | OFF }

Where, ABSolute Absolute value deviation display
PERCent Percent deviation display
OFF Real value display

Where, <n> is

Character 1 (49) is equal to the nominal value of primary parameter.

Or character 2 (50) is equal to the nominal value of the secondary parameter.

Query Command：FUNCtion:DEV<n>:MODE?

Return Format：{ABS | PER | OFF}<NL>

:DEV<n>:REFerence[:VALue]

The FUNCtion:DEV<n>:REFerence[:value] command is used to set the nominal value of the deviation. The FUNCtion:DEV<n>:REFernece[:value]? query returns the current nominal value of the deviation.

Command Syntax：FUNCtion:DEV<n>:REFerence <value>

Where,[:value] is NR1, NR2 or NR3 data format.

<n> is

Character 1 (49) is equal to the nominal value of primary parameter.
Or character 2 (50) is equal to the nominal value of the secondary parameter.
Query Command：FUNCtion:DEV<n>:REFerence?
Return Format：<NR3><NL>

:DEV<n>:REFerence:FILL

The FUNCtion:DEV<n>:REFerence:FILL command is used to set the nominal value of the deviation. This command directs the instrument to make a test and then copies the results of the primary and the secondary parameters as the nominal values of the deviation.
Command Syntax：FUNCtion:DEV<n>:REFerence:FILL

2.7 FREQuency subsystem commands

The FREQuency subsystem commands are mainly used to set the measurement frequency of the instrument. The :FREQuency query returns the current measurement frequency.
Command Syntax：FREQuency {<value> | MAX | MIN}
Where,
<value> NR1, NR2 or NR3 data format followed by Hz, kHz, MHz.
MIN Set the measurement frequency as 50Hz.
MAX Set the measurement frequency as 100kHz or 200kHz (The maximum frequency of U2832 is 200kHz)
Query Command： FREQuency?
Return Format：<NR3><NL>

2.8 VOLTage subsystem commands

The VOLTage subsystem commands are mainly used to set the measurement voltage. The VOLTage? query returns the current measurement voltage.
Command Syntax：VOLTage[:LEVel] {<value> | MAX | MIN}
Where,
<value> NR1, NR2 or NR3 data format followed by V.
MIN Set the measurement voltage as 0.1V.
MAX Set the measurement voltage as 1V or 2V.
Query Command： VOLTage[:LEVel]?

Return Format：<NR3><NL>

BIAS[:STATe] Enables or disables the BIAS function
Command Syntax： BIAS[:STATe] {ON|OFF|0|1}
Query Command： BIAS[:STATe]?
Return Format：{1 | 0}<NL>
Note: This feature requires instrument software support

BIAS:CURRent[:LEVel] Sets the BIAS current. The current unit can be uA, mA, or A
Command Syntax： BIAS:CURRent[:LEVel] {<Data>|Min|Max}
Query Command： BIAS:CURRent[:LEVel]?
Return Format：If the current mode is not biased, +9.90000E+37 is invalid
Note: This feature requires instrument software support

BIAS:VOLTage[:LEVel] Sets the BIAS voltage. The voltage unit and multiplier mV and V can be used
Command Syntax： BIAS:VOLTage[:LEVel] {<Data>|Min|Max}
Query Command：BIAS:VOLTage[:LEVel]?
Return Format：If the current mode is not biased, +9.90000E+37 is invalid
Note: This feature requires instrument software support

2.9 APERture subsystem commands

The APERture subsystem commands are mainly used to set the measurement speed, average times used in measurement. The APERture? query returns the current measurement speed, average times.
Command Syntax：APERture {FAST | MEDium | SLOW}[,<value>]
Where,
FAST: 40 times/sec
MEDium: 10 times/sec
SLOW: 2.5 times/sec
<value> 1 to 255 in NR1
Query Command：APERture?
Return Format：{FAST | MED | SLOW},<NR1><NL>

2.10 Source RESister subsystem commands

The Source RESister subsystem commands are mainly used to set the output internal resistor mode. The Source RESister? query returns the current output internal resistance status.
Command Syntax：SRESister {30 | 50 | 100}
Query Command：SRESister?
Return Format：{30 | 50 | 100}<NL>

2.11 FETCh? subsystem commands

The FETCh? subsystem commands are mainly used to direct U2832 to input a measurement result. Figure 2-12-1 shows the FETCh? subsystem commands tree.

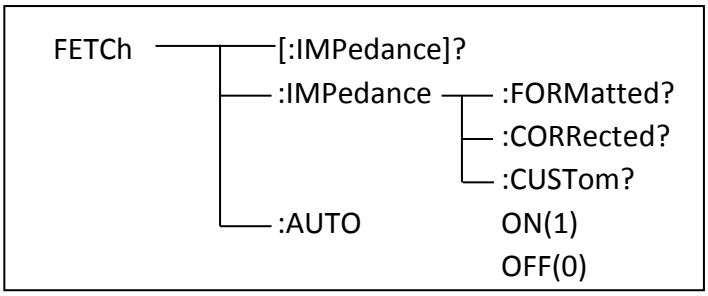


Figure 2-12-1

[:IMPedance]?
The FETCh[:IMPedance]? query directs U2832 to input the last measurement result to the output buffer zone.
Query Command：FETCh[:IMPedance]?
For example: WrtCmd (“TRIG:SOUR BUS”)
WrtCmd (“TRIG”)
WrtCmd (“FETC?”)
U2832 applies ASCII to delivery result, details are follows.
On measurement display page, bin NO. display page, bin count display page, ASCII data output format are described as below:

SN.NNNNNESNN	,	SN.NNNNNESNN	,	SN	,	SN or SNN	NL^END
<DATA A>		<DATA B>		<Status>		<BIN number>	

Where,
<DATA A>, <DATA B> format: <DATA A> (primary measurement data), <DATA B> (secondary measurement data)
12-digits ASCII format are as below:
SN.NNNNNESNN
(S:+/-,N: from 0 to 9, E: Exponent Sign)

Status	Description
-1	(In data buffer memory) no data
0	Common measurement data
+1	Analog LCR unbalance
+2	A/D converter is not working.
+3	Signal source is over loading.
+4	Constant voltage cannot be adjusted.

<status> format: When above measurement data is used, <status> data will display measurement status.

The output format of the <Status> display data uses 2-digits ASCII: SN (S: +/-, N: from 0 to 4)
NOTE: When <status> is -1, +1 or +2, the measurement data is 9.9E37. When <status> is 0, +3 or +4, the real measurement data is beyond the limits.

Data	Sort result
0	No sorting
+1	Bin 1
+2	Bin 2
+3	Bin 3
+10	Out of tolerance
+11	Auxiliary bin

<Bin No.> format: The data displays the sorting results of the displayed bin, shown as above.
Only when the instrument compare function is set as ON, <bin No.> data can be displayed.
The output format of <bin No.> data applies 2 to 3 digits ASCII: SN or SNN (S: +/_, N: from 0 to 9).

On list sweep display page, the ASCII data output format is shown as below, that is, the return-circuit replaces sweep point number.

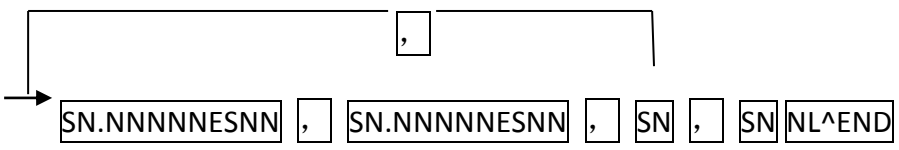


Figure 6 ASCII format 2 (list sweep)

Where,

Descriptions for <DATA A>, <DATA B>, <Status> are the same described before.

<Judge> format is as below:

<Input/Output> format: The data displays the compare result of the list sweep..

Data	Result
-1	low
0	pass
+1	high

When the compare function of the list sweep measurement is turned off, the output result of <Input/Output> is 0.

<Input/Output> data output format applies 2-digits ASCII format: SN (S: +/_, N: from 0 to 1)

:IMPedance:FORMatted?

Query Command：FETCh:IMPedance:FORMatted?

Return Format：same as FETCh[:IMPedance]?

:IMPedance:CORRected?

Query Command：FETCh:IMPedance:CORRected?

Return Format：<DATA A>,<DATA B><NL>

:IMPedance:CUSTom?

Query Command：FETCh:IMPedance:CUSTom?

Return Format：<DATA A>,<DATA B>,<MON 1>,<MON 2>,<Status>,<Bin No.><NL>

:AUTO

Command Syntax：FETCh:AUTO {{1 | ON} | {0 | OFF}}

Query Command：FETCh:AUTO?

Return Format：{1 | 0}<NL>

2.12 CORRection subsystem commands

The CORRection subsystem commands are mainly used to set the correction function, OPEN, SHORT, LOAD.

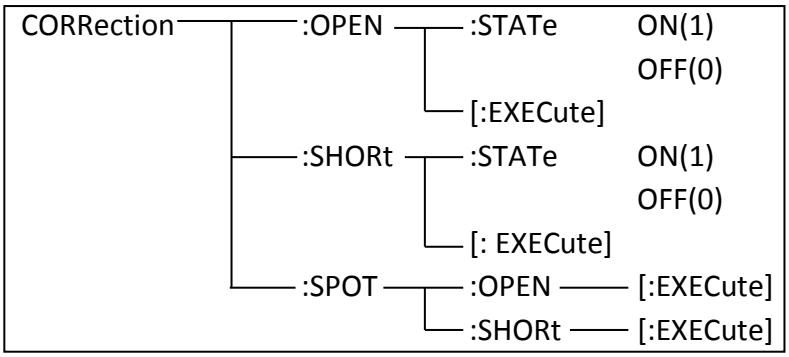


Figure 2-13-1

:OPEN[:EXECute]

The CORReciton:OPEN command is used to execute open correction for preset test points.

Command Syntax：CORRection:OPEN[:EXECute]

NOTE: The command from the BUS will be suspended during zero clearing. After zero clearing is finished, the subsequent command will be executed.

Remark:If adding a *OPC ? command after a command need to execute for a long time, "1" will be returned after the command is finished.

For example, CORR:OPEN:EXEC;*OPC?

:OPEN:STATe

The CORRection:OPEN:STATe command is used to set the open correction ON or OFF. The CORReciton：OPEN:STATe? query returns the current open correction status.

Command Syntax：CORRection:OPEN:STATe {{1 | ON} | {0 | OFF}}

Query Command：CORRection:OPEN:STATe?

Return Format：{1 | 0}<NL>

:SHORt[:EXECute]

The CORRection:SHORt command is used to execute short correction for preset test points.

Command Syntax：CORRection:SHORt[:EXECute]

NOTE: The command from the BUS will be suspended during zero clearing. After zero clearing is finished, the subsequent command will be executed.

Remark:If adding a *OPC ? command after a command need to execute for a long time, "1" will be returned after the command is finished.

For example, CORR:SHOR:EXEC;*OPC?

:SHORt:STATe

The CORRection:SHORt:STATe command is used to set the short correction status.

The CORRection:SHORt:STATe? query returns the current short correction status.

Command Syntax：CORRection:SHORt:STATe {{1 | ON} | {0 | OFF}}

Query Command：CORRection:SHORt:STATe?

Return Format：{1 | 0}<NL>

:SPOT:OPEN[:EXECute]

The CORRection:SPOT:OPEN[:EXECute] command is used to execute open correction for current test points.

Command Syntax：CORRection:SPOT:OPEN[:EXECute]

NOTE: The command from the BUS will be suspended during zero clearing. After zero clearing is finished, the subsequent command will be executed.

:SPOT:SHORt[:EXECute]

The CORRection:SPOT:SHORt[:EXECute] command is used to execute short correction for current test points.

Command Syntax：CORRection:SPOT:SHORt[:EXECute]

NOTE: The command from the BUS will be suspended during zero clearing. After zero clearing is finished, the subsequent command will be executed.

:LOAD:STATe Enables or disables the LOAD correction function

Command Syntax：CORRection:LOAD:STATe {1|0|ON|OFF}

Query Command：CORRection:LOAD:STATe?

:LOAD:TYPE Sets the parameter TYPE of the LOAD correction function
Command Syntax：CORRection:LOAD:TYPE {CPD |CSRS |LSQ|RX| ZTR }
Query Command：CORRection:LOAD:TYPE?
Note: Parameters are determined by the impedance class combination parameters supported by the instrument (excluding DCR parameters)

:SPOT<n>:STATe Sets the frequency point correction switch
Command Syntax：CORRection:SPOT<n>:STATe {1|0|ON|OFF}
Query Command：CORRection:SPOT<n>:STATe?
Note: <n> indicates the number 0 to 9

:SPOT<n>:FREQuency Sets the correction FREQuency of the specified FREQuency point
Command Syntax：CORRection:SPOT<n>:FREQuency <freq>
Query Command：CORRection:SPOT<n>:FREQuency?

:SPOT<n>:OPEN Performs open-circuit zero clearing at the specified frequency point (no query)
Command Syntax：CORRection:SPOT<n>:OPEN[:EXECute]

:SPOT<n>:SHORt Performs SHORt circuit clearing at the specified frequency point (no query)
Command Syntax：CORRection:SPOT<n>:SHORt[:EXECute]

:LOAD[:EXECute] Performs a LOAD measurement at a specified frequency point (no query)
Command Syntax：CORRection:SPOT<n>:LOAD[:EXECute]

:LOAD:STANdard Sets the STANdard reference value (primary and secondary parameters) for the specified frequency point.
Command Syntax：CORRection:SPOT<n>:LOAD:STANdard <REFA>,<REFB>
Query Command：CORRection:SPOT<n>:LOAD:STANdard?

2.13 COMParator subsystem commands

The COMParataor subsystem commands are used to set the bin comparator function including ON/OFF setting, Limit table setting. Figure 2-14-1 shows the COMParator subsystem commands tree.

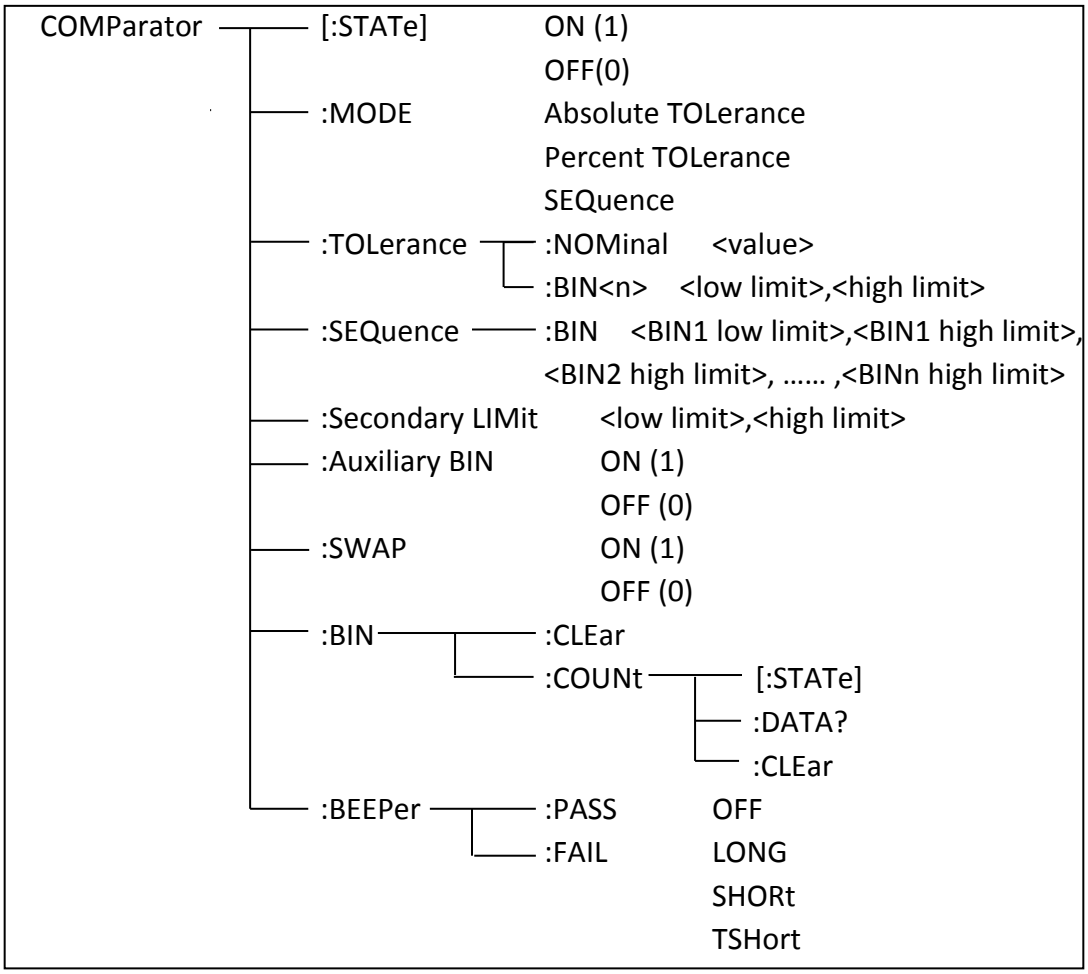


Figure 2-14-1

[:STATe]

The COMParator[STATe] command is used to set the comparator function as ON or OFF. The COMParator[STATe]? query returns the current comparator state.

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

Query Command：COMParator[:STATe]?

Return Format：{1 | 0}<NL>

:MODE

The COMParator:MODE command is used to set the comparator mode. The COMParator:MODE? query returns the current mode.

Command Syntax：COMParator:MODE {ATOLerance | PTOLerance | SEQuence}

Where, ATOLerance means absolute tolerance mode.

PTOLerance means proportional tolerance mode.

SEQuence means sequential tolerance mode.

For example: WrtCmd ("COMP:MODE ATOL")

Query Command：COMParator:MODE?

Return Format：{ATOL | PTOL | SEQ}<NL>

:TOLerance:NOMinal

The COMParator:TOLerance:NOMinal command is used to set the nominal value (this function is valid only when the limit mode is set as deviation mode).

The COMParator:TOLerance:NOMinal? query returns the current nominal value.

Command Syntax：COMParator:TOLerance:NOMinal <value>

Where,

<value> is a nominal value in NR1, NR2 or NR3 data format.

For example: WrtCmd ("COMP:TOL:NOM 100E-12")

Query Command：COMParator:TOLerance:NOMinal?

Return Format：<NR3><NL>

:TOLerance:BIN<n>

The COMParator:TOLerance:BIN<n> command is used to set the high and the low limits of each bin (this function is valid only when the limit mode is set as deviation mode).

The COMParator:TOLeance:BIN<n>? query returns the current high and the low limits of each bin.

Command Syntax：COMParator:TOLerance:BIN<n><low limit>,<high limit>

Where,

<n> is the bin number from 1 to 4.

<low limit> is the low limit in NR1, NR2 or NR3 data format.

<high limit> is the high limit in NR1, NR2 or NR3 data format.

NOTE: The low limit should be smaller than the high limit, or error information will be reported.

For example: WrtCmd (“COMP:TOL:BIN1 -5,5”)
WrtCmd (“COMP:TOL:BIN2 -10,10”)

Query Command：COMParator:TOLerance:BIN<n>?
Return Format：<low limit>,<high limit><NL> NR3 data format.

:SEQuence:BIN

The COMParator:SEQuence:BIN command is used to set the high and the low limits of sequential mode (this function is valid only when the limit mode is set as the sequential mode.). The COMParator:SEQuence:BIN? query returns the current high and the low limits of each bin.

Command Syntax：COMParator:SEQuence:BIN <BIN1 low limit>,<BIN1 high limit>, <BIN2 high limit>,…, <BINn high limit>

Where,

<BIN1 low limit> is the low limit of BIN 1 in NR1, NR2 or NR3 data format.
<BIN1 high limit> is the high limit of BIN1 in NR1, NR2 or NR3 data format.
<BINn high limit> is the high limit of BINn (the miximum of n is 9) in NR1, NR2 or NR3 data format.

NOTE: The low limit should be smaller than the high limit, or error information will be reported.

For example: WrtCmd (“COMP:SEQ:BIN 10, 20, 30, 40, 50”)

Query Command：COMParator:SEQuence:BIN?
Return Format：<BIN1 low limit>,<BIN1 high limit>,<BIN2 high limit>,…, <BINn high limit><NL>

:Secondary LIMit

The COMParator:Secondary LIMit command is used to set the high and the low limits of the secondary parameter.

The COMParator:Secondary LIMit query returns the current high and the low limits of the secondary parameter.

Command Syntax：COMParator:SLIMit <low limit>,<high limit>

Where,

<low limit> is the low limit in NR1, NR2 or NR3 data format.
<high limit> is the high limit in NR1, NR2 or NR3 data format.

NOTE: The low limit should be smaller than the high limit, or error information will be reported.

For example: WrtCmd (“COMP:SLIM 0.001, 0.002”)

Query Command：COMParator:SLIMit?
Return Format：<NR3>,<NR3><NL>

:Auxiliary BIN

The COMParator:Auxiliary BIN command is used to set the auxiliary bin as ON or OFF. The COMParator:Auxiliary BIN? query returns the current auxiliary bin state.
Command Syntax：COMParator:ABIN {{1 | ON} | {0 | OFF}}
Query Command：COMParator:ABIN?
Return Format：{1 | 0}<NL>

:SWAP

The COMParator:SWAP command is used to set the swap mode ON or OFF. For example: the original function parameter is Cp-D, after the SWAP mode is set as ON, the function parameter will be changed as D-Cp. In this case, the limits from BIN1 to BIN9 become the high and the low limits of D, the original secondary limits become that of Cp. That is to say, this function is to make swap comparison between the primary and the secondary parameters. On the contrary, If OFF is selected, the comparison will be made according to the original sequence. The COMParator:SWAP? query returns the current state of the swap function.
Command Syntax：COMParator:SWAP {{1 | ON} | {0 | OFF}}
Query Command：COMParator:SWAP?
Return Format：{1 | 0}<NL>

:BIN:CLEar

The COMParator:BIN:CLEar command is used to clear all limits on limit table setup page.
Command Syntax：COMParator:BIN:CLEar

:BIN:COUNT[:STATe]

The COMParator:BIN:COUNt[:STATe] command is used to set the bin count function as ON or OFF. The COMParator:BIN:COUNt[:STATe]? query returns the current state of the bin count function.
Command Syntax：COMParator:BIN:COUNt[:STATe] {{1 | ON} | {0 | OFF}}
Query Command：COMParator:BIN:COUNt[STATe]?
Return Format：{1 | 0}<NL>

:BIN:COUNt:DATA?

The COMParator:BIN:COUNt:DATA? query returns the current comparison result of the bin count.

Query Command：COMParator:BIN:COUNt:DATA?

Return Format：<BIN1 count>,<BIN2 count>,…,<BINn count>,<OUT BIN count>,
<AUX BIN count><NL>

Where,

<BIN1-9 count> is the count result of BIN1-9, in NR1 data format.

<OUT BIN count> is the count result of the OUT OF BIN, in NR1 data format.

<AUX BIN count> is the count result of the auxiliary bin, in NR1 data format.

:BIN COUNT:CLEar

The COMParator:BIN:COUNt:CLEar command is used to clear all bin count results.

Command Syntax：COMParator:BIN:COUNt:CLEar

:BEEPer:PASS

The COMParator:BEEPer:PASS command is used set the beeper when comarator result is passed.

The COMParator:BEEPer:PASS? query returns the current status of the beeper when comarator result is passed.

Command Syntax：COMParator:BEEPer:PASS {OFF | LONG | SHORt | TSHort}

Query Command：COMParator:BEEPer:PASS?

Return Format：{OFF | LONG | SHOR | TSH}<NL>

:BEEPer:FAIL

The COMParator:BEEPer:FAIL command is used set the beeper when comarator result is failed.

The COMParator:BEEPer:FAIL? query returns the current status of the beeper when comarator result is failed.

Command Syntax：COMParator:BEEPer:FAIL { OFF | LONG | SHORt | TSHort }

Query Command：COMParator:BEEPer:FAIL?

Return Format：{OFF | LONG | SHOR | TSH}<NL>

2.14 LIST subsystem commands（Option）

The LIST subsystem commands are mainly used to set the list sweep function, sweep points, sweep mode, sweep limits.

LIST:SPOT[0-89]

Command Syntax： LIST:SPOT[0-89]
<func>,<freq>,<lev>,<bias>,<nom>,<low>,<high>

Query Command：LIST:SPOT[0-89]?

Set scan point parameters:：

<func>： indicates the measurement result parameter type, that is, one of the following: OFF= Turn OFF the scan point
{LS|LP|CS|CP|RS|RP|R|ESR|D|Q|Z|RAD|DEG|X|Y|G|B|OFF}

<freq>：set the scanning frequency, NR3 type, can follow the frequency unit

<lev>： set the scan level (without changing the level mode), NR3 type, can be in voltage or current units

<bias>：sets the bias value (does not change the bias nature), NR3 type, can be in voltage or current units

<nom>：Sets the list comparator nominal value, NR3 type, unit specified by <func>, and can be followed by multiplications

<low>： sets the lower limit of the list comparator, NR3 type, unit specified by <func>, and can be followed by multiplications

<high>： sets the upper limit of the list comparator, NR3 type, unit specified by <func>, and can be followed by multiplications

Description:

- allows you to set only the first N items and ignore the following items, but does not allow you to set the hop items

  For example, to set to < BIAS >, the previous <func>,<freq>,and < Lev > must be included

- If the data is set to 9.9E37 or -9.9e37, the corresponding data is not set (keep the original data).
- This command cannot change the level mode, bias mode (refer to the following list of commands to change the mode)

LIST:BIAS:CURRent

Command Syntax： LIST:BIAS:CURRent <lev0>,<lev1>,…<levn>

Query Command： LIST:BIAS:CURRent?

Set the bias current mode and value. The value of n ranges from 0 to 89

- If the value is 9.9E37 or -9.9e37, the original scan point data is not changed
- NR3 Data type, can follow current unit

LIST:BIAS:VOLTage

Command Syntax： LIST:BIAS:VOLTage <lev0>,<lev1>,…<levn>

Query Command： LIST:BIAS:VOLTage?

Set the bias voltage mode and the bias voltage. The value of n ranges from 0 to 89

- If the value is 9.9E37 or -9.9e37, the original scan point data is not changed
- NR3 data type, available in voltage units

LIST:BIAS:OFF

Command Syntax： LIST:BIAS:OFF Disable the bias scan function (no query)

LIST:CLEar:ALL

Command Syntax: LIST:CLEar:ALL

LIST:COMParator:FAIL

Command Syntax：LIST:COMParator:FAIL {CONTinue|RETRy|PAUSe|STOP}

Query Command：LIST:COMParator:FAIL?

Set the operation mode after the comparison fails

CONTinue ：Continues scanning

RETRy ：Test again and continue

PAUSe ：Wait for a valid trigger before continuing

STOP ：Stop scanning

Return Format：{CONTINUE|RETRY|PAUSE|STOP}<NL>

LIST:COMParator:MODE

Command Syntax：LIST:COMParator:MODE { ABSolute| PERCent}

Query Command：LIST:COMParator:MODE?

Sets the list comparison mode

ABSolute ：Indicates the ABSolute value of a list

PERCent ：Indicates the percentage mode of the list

Return Format：{ABS|PERC}<NL>

LIST:COMParator: NOMinal

Command Syntax：LIST:COMParator: NOMinal <nom1>,< nom 2>,…< nomn>

Query Command：LIST:COMParator: NOMinal?

Set the nominal value of the list comparator. N ranges from 0 to 89

- Impedance type data (NR3), without unit, can follow multiplier
- 9.9E37 or -9.9E37 indicates that the original limit data is not changed

LIST:COMParator:LOW

Command Syntax：LIST:COMParator:LOW <low1>,<low2>,…<lown>

Query Command：LIST:COMParator:LOW?

Set the lower limit of the list comparator. N ranges from 0 to 89

- Impedance type data (NR3), without unit, can follow multiplier
- 9.9E37 or -9.9E37 indicates that the original limit data is not changed

LIST:COMParator:HIGH

Command Syntax：LIST:COMParator:HIGH <low1>,<low2>,…<lown>

Query Command：LIST:COMParator:HIGH?

Set the upper limit of the list comparator. N ranges from 0 to 89

- Impedance type data (NR3), without unit, can follow multiplier
- 9.9E37 or -9.9E37 indicates that the original limit data is not changed

LIST:FREQuency

Command Syntax：LIST:FREQuency <freq0>,<freq1>,…<freqn>

Query Command：LIST:FREQuency?

Set the scanning frequency. N ranges from 0 to 89

- 9.9E37 or -9.9E37 indicates that the original limit data is not changed
- NR3 data type, can be followed by frequency units, frequency range is limited by the allowable range of the instrument

LIST:FUNCtion

Command Syntax：LIST:FUNCtion <fun0>,<fun1>,….<funn>

Query Command：LIST:FUNCtion?

Sets the scan measurement function. N ranges from 0 to 89

- The parameter type is as follows：

  {LS|LP|CS|CP|RS|RP|R|ESR|D|Q|Z|RAD|DEG|X|Y|G|B|OFF}

OFF: Scanning is disabled

LIST:MODE

The LIST:MODE command is used to set the list sweep mode. The LIST:MODE? query returns the current list sweep mode.

Command Syntax：LIST:MODE {STEPped | SEQuence}

Where,

SEQuence means sequential mode.

STEPped means single step mode.

For example: WrtCmd (“LIST:MODE SEQ”)

Query Command：LIST:MODE?

Return Format：{STEP | SEQ}<NL>

LIST:VOLTage

Command Syntax：LIST:VOLTage <lev0>,<lev1>,…<levn>

Query Command：LIST:VOLTage?

Set the test level to the voltage mode and level size. N ranges from 0 to 89

- If the value is 9.9E37 or -9.9e37, the original scan point data is not changed
- NR3 data type, available in voltage units

2.15 Mass MEMory subsystem commands

The Mass MEMory subsystem commands are used for file storing and load.Figure 2-17-1 shows Mass MEMory subsystem commands tree.

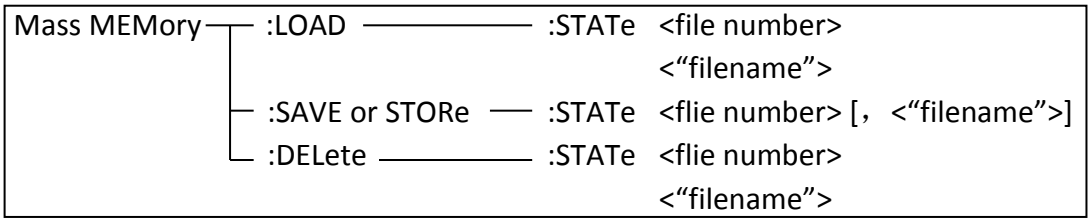


Figure 2-17-1

:LOAD:STATe

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

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

Where,

<file number> is the file number ranging from 1 to 100 (NR1).

For example: WrtCmd (“MMEM:LOAD:STAT 1”)

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

: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 100 , NR1 format without unit.

<“filename”> The file name consists of less than 17 ASCII characters. <Unnamed> will be the default name, if you don’t input a file name.

ⓘ NOTE: U2832 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 100, NR1 format without unit.

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

ⓘ **NOTE: U2832 will not give a warning message when a file is to be deleted.**

Command Syntax: MMEMory:DELete:STATe “filename”

2.16 KEY subsystem commands

KEY commands are used to control the keys and knobs on the operation panel of U2832.

KEY:LOCal	enable the front panel operation
KEY:MEASure	enter into the <MEAS DISP> page
KEY:SETup	enter into the <MEAS SETUP> page
KEY:BIAS	DC bias key
KEY:UPPer	upper the cursor
KEY:DOWN	down the cursor
KEY:LEFT	left the cursor
KEY:RIGHt	right the cursor
KEY:NUM<n>	numeric key， n range from 0 to 9
KEY:DOT	decimal point key
KEY:ENTer	enter key
KEY:BACKsapce	backspace key
KEY:TRIGger	trigger key
KEY:KEYLOCK	keylock key
KEY:CLEar	clear key
KEY:F<n>	soft key, n is from 1 to 6

2.17 SYSTem Subsystem Commands

The SYSTem subsystem commands are used for system setups. Figure 2-19-1 shows the SYSTem subsystem commands tree.

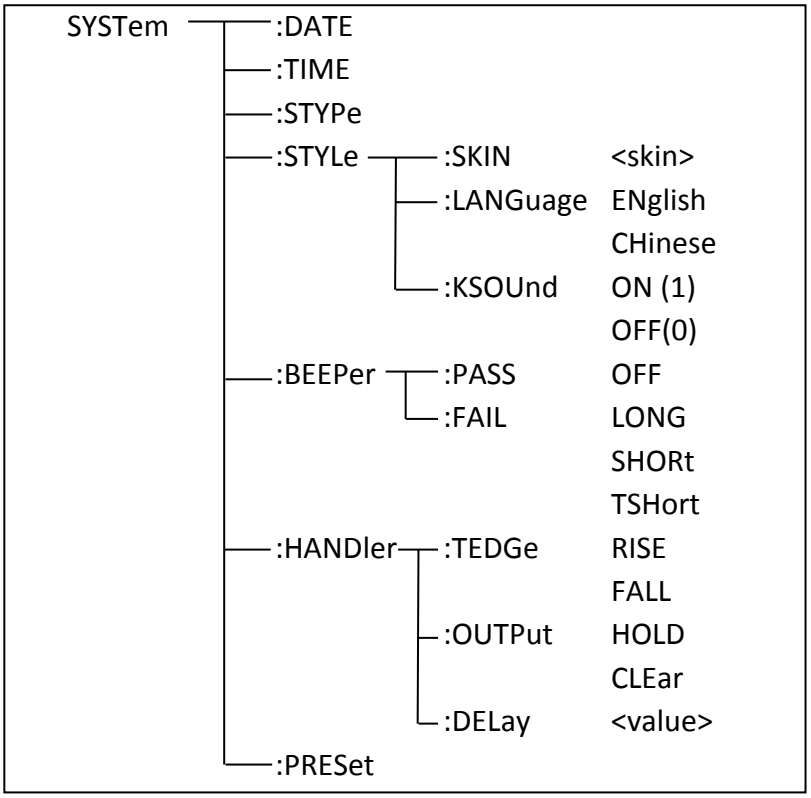


Figure 2-19-1

: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 Command: 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>

Where, hour, minute, second is NR1 format.

Query Command: SYSTem:TIME?

Return Format: <NR1>,<NR1>,<NR1><NL>

:STYPe

The :STYPe command sets the file type when pressing SAVE key. The :STYPe? query returns the current file type.

Command Syntax: SYSTem:STYPe {CSV | GIF | BMP | PNG}

Query Command: SYSTem:STYPe?

Return Format: {CSV | GIF | BMP | PNG}<NL>

:STYLe:SKIN

The :STYLe:SKIN command sets the display color. The :STYLe:SKIN? Query returns the current display color.

Command Syntax: SYSTem:STYLe:SKIN {GRAY | BLACk | BLUE | CYAN}

Query Command: SYSTem:STYLe:SKIN?

Return Format: {GRAY | BLACK | BLUE | CYAN}<NL>

:STYLe:LANGuage

The :STYLe:LANGuage command sets the display language. The :STYLe:LANGuage? query returns the current display language.

Command Syntax: SYSTem:STYLe:LANGuage {ENglish | Chinese}

Query Command: SYSTem:STYLe:LANGuage?

Return Format: {EN | CH}<NL>

:STYLe:KSOUnd

The :STYLe:KSOUnd command sets the key sound. The :STYLe:KSOUnd? query returns the current state of key sound.

Command Syntax: SYSTem:STYLe:KSOUnd {{1 | ON} | {0 | OFF}}

Query Command: SYSTem:STYLe:KSOUnd?

Return Format: {1 | 0}<NL>

:BEEPer:PASS

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

Query Command: SYSTem:BEEPer:PASS?

Return Format: {OFF | LHIG | LLOW | SSH | TSH}<NL>

:BEEPer:FAIL

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

Query Command: SYSTem:BEEPer:FAIL?

Return Format： {OFF | LHIG | LLOW | SSH | TSH}<NL>

:HANDler:TEDGe

Command Syntax： SYSTem:HANDler:TEDGe {RISing | FALLing}

Query Command: SYSTem:HANDler:TEDGe?

Return Format： {RIS| FALL}<NL>

:HANDler:OUTPut

Command Syntax： SYSTem:HANDler:OUTPut {HOLD | CLEar}

Query Command: SYSTem:HANDler:OUTPut?

Return Format： {HOLD| CLE}<NL>

:HANDler:DELay

Command Syntax： SYSTem:HANDler:DELay <value>

Query Command: SYSTem:HANDler:DELay?

Return Format： <NR3><NL>

:PRESet

Command Syntax： SYSTem:PRESet

3 Error and warning message

The bus commands may have some spelling errors, syntax errors or wrong parameters. U2832 executes a command after the command is analyzed. If one of above errors occurs, U2832 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: TRIG:DEL 66s，the trig delay time is out of 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 time.
Data too long	Data is too long. For example: The number of characters for a file name can not exceed 17 characters and numeric parameter, 17 characters.
Syntax error	Error syntax, for example:DISP.PAGE MSET, where（.）should be（:）.
Missing parameter	
Character data not allowed	
Numeric data not allowed	
Command ignored	Some command may 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

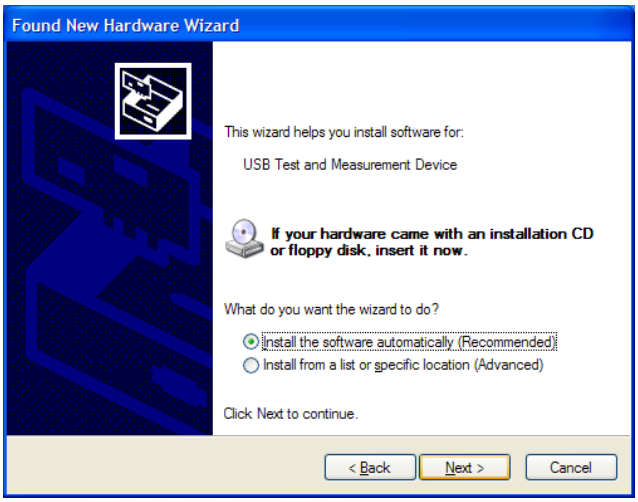
Preparation for Programming

Download NI-VISA software from http://www.ni.com to install it. The installing path is C:\Program Files\IVI Foundation\VISA.

Take U2832 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.



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.

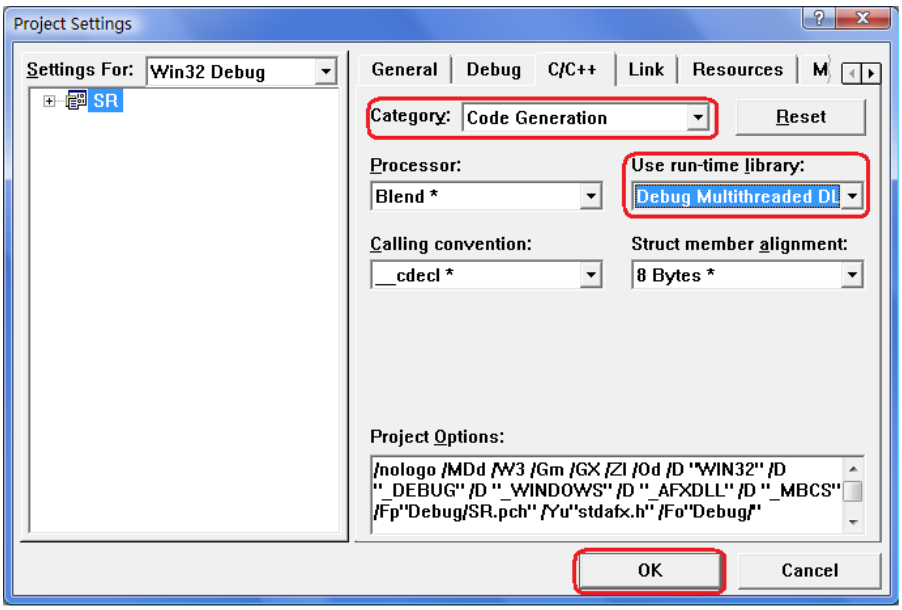


Figure 4-1-1

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.

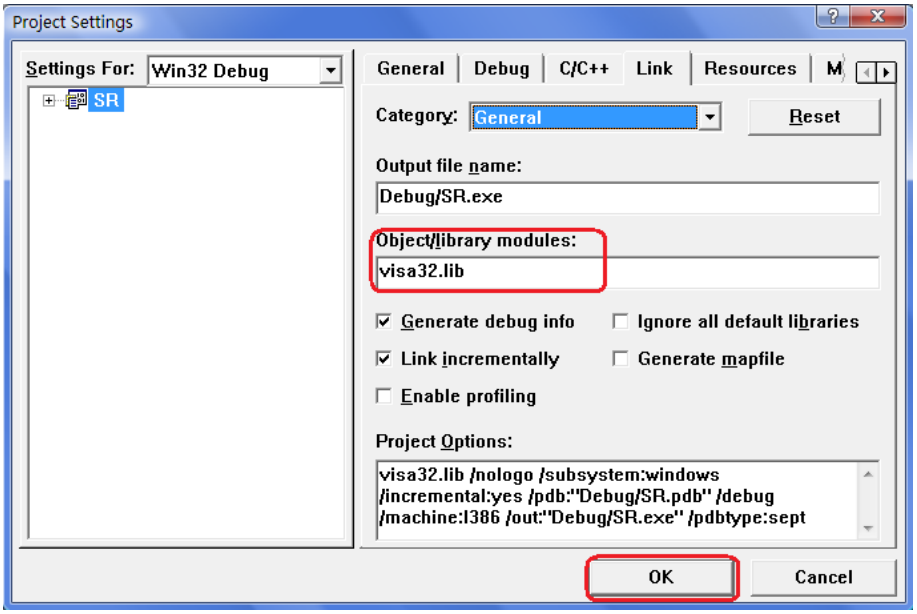


Figure 4-1-2

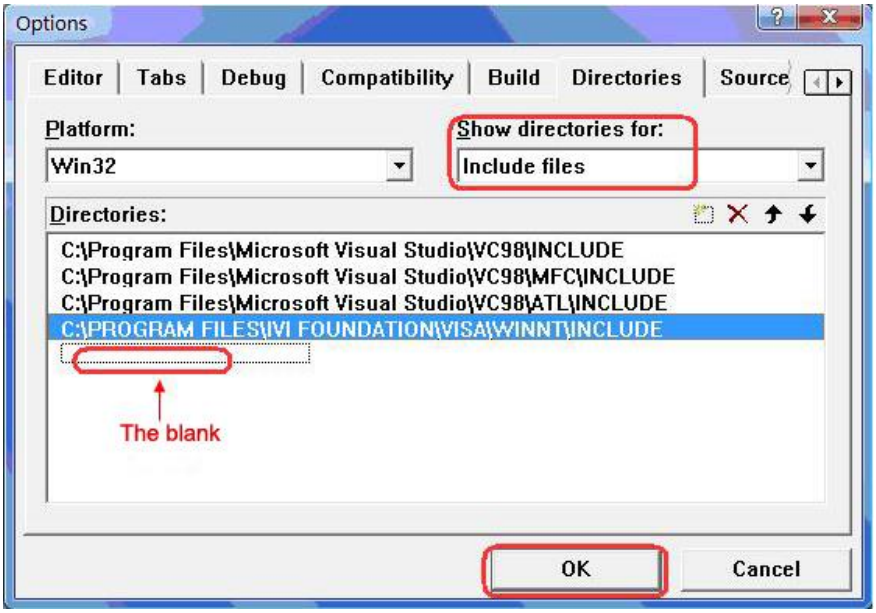


Figure 4-1-3

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

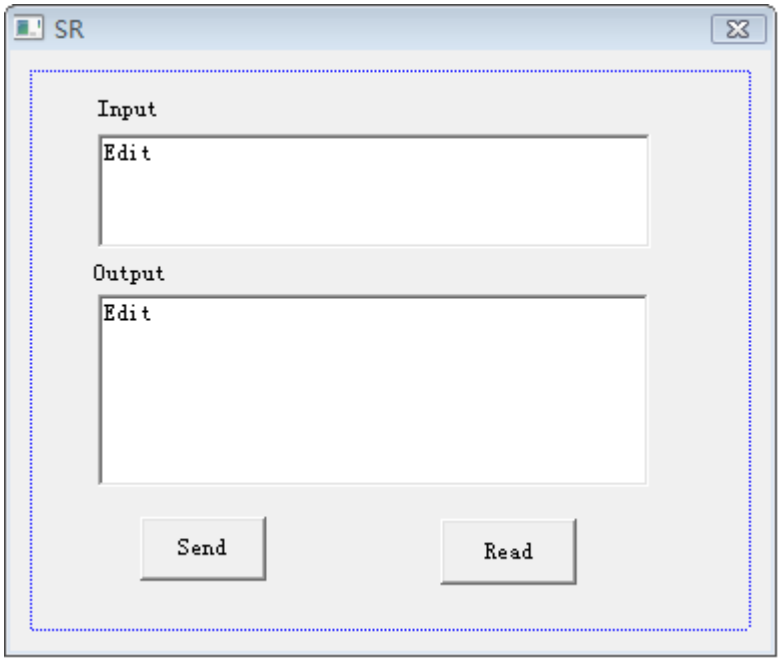


Figure 4-1-4

(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.

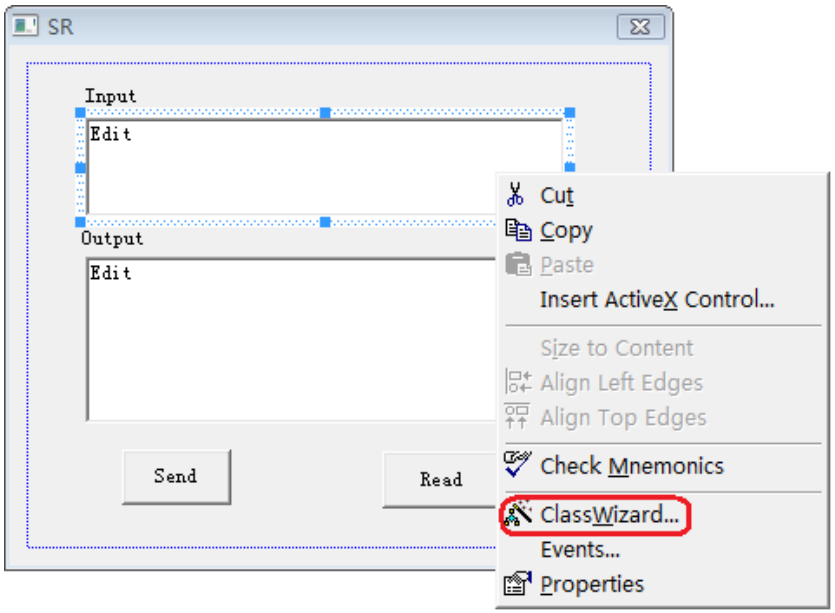


Figure 4-1-5

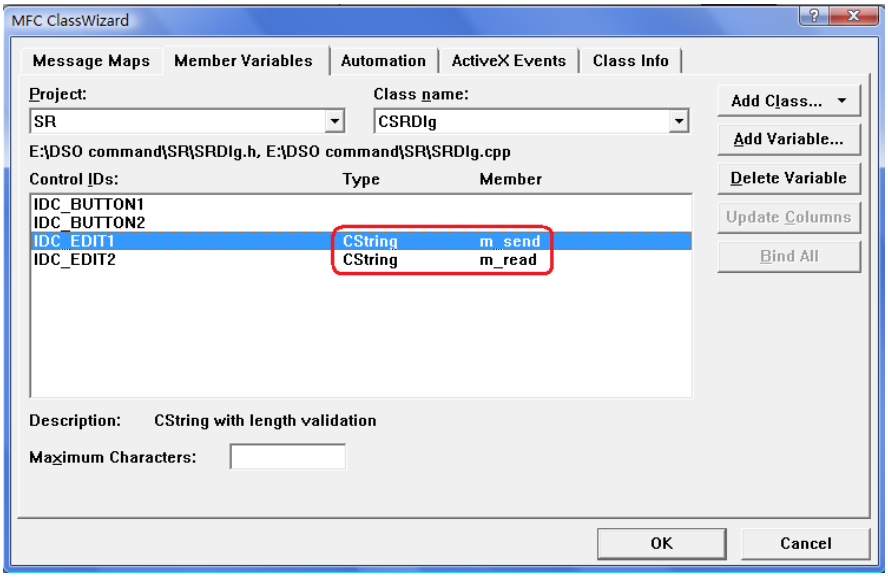


Figure 4-1-6

(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.

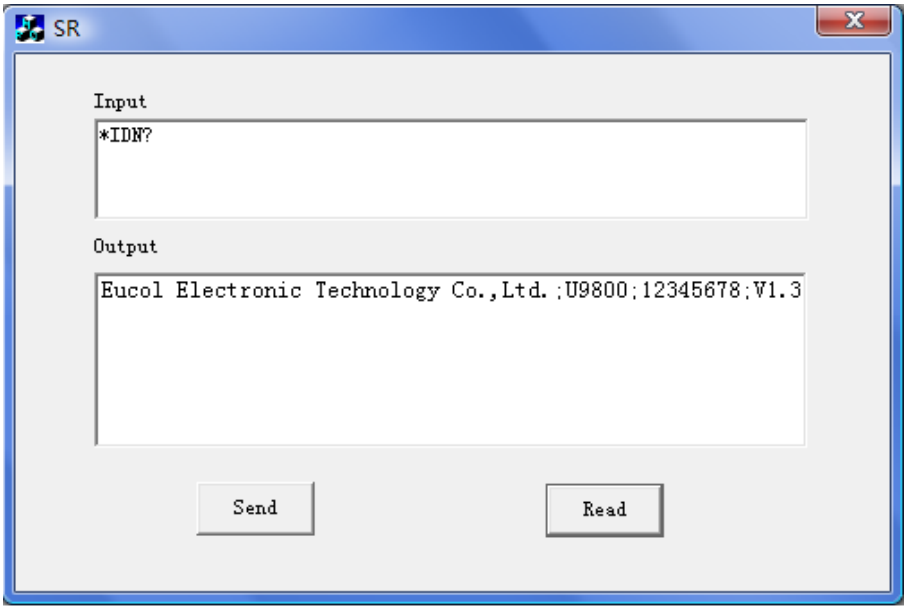


Figure 4-1-7

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.

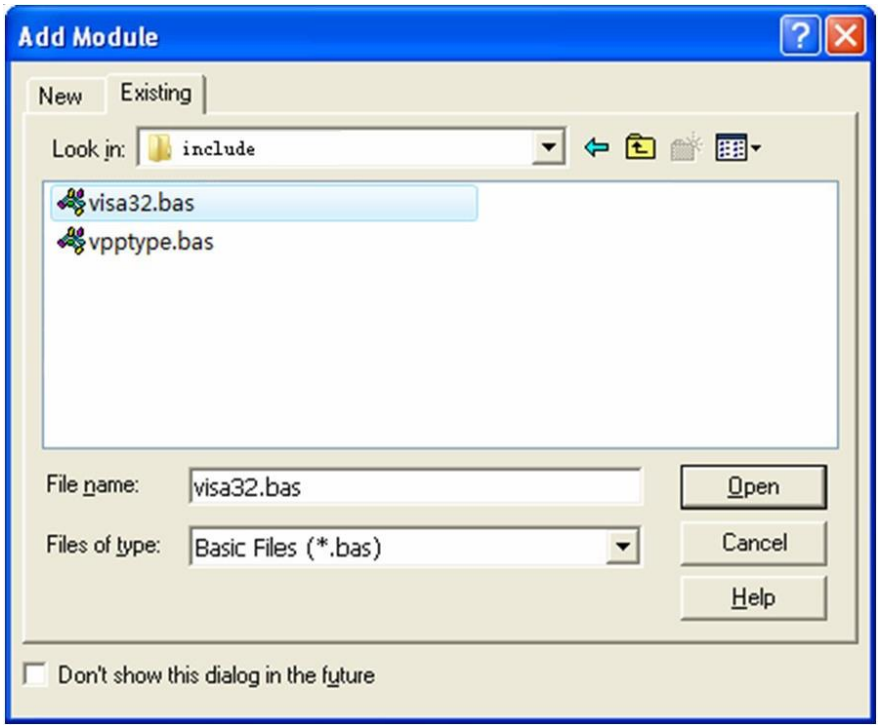


Figure 4-2-1

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.

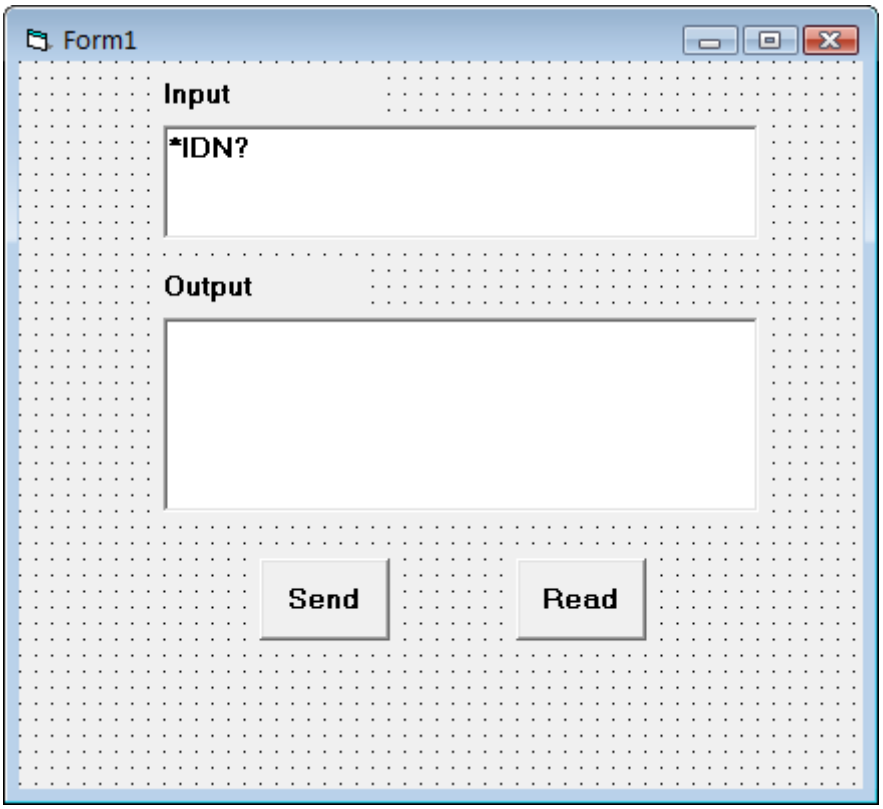


Figure 4-2-2

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.

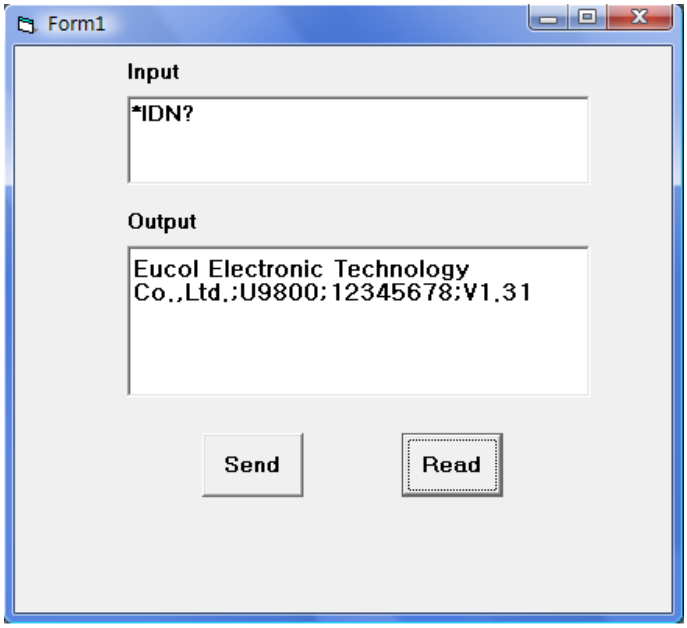

Form1
Input
*IDN?
Output
Eucol Electronic Technology Co.,Ltd.;U9800;12345678;V1.31
Send
Read

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.

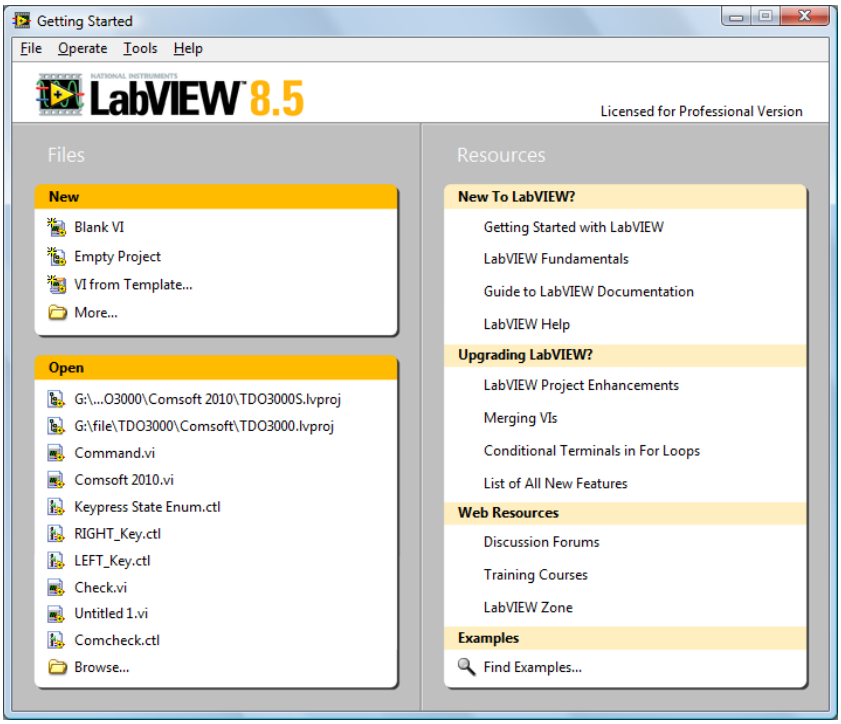


Figure 4-3-1

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.

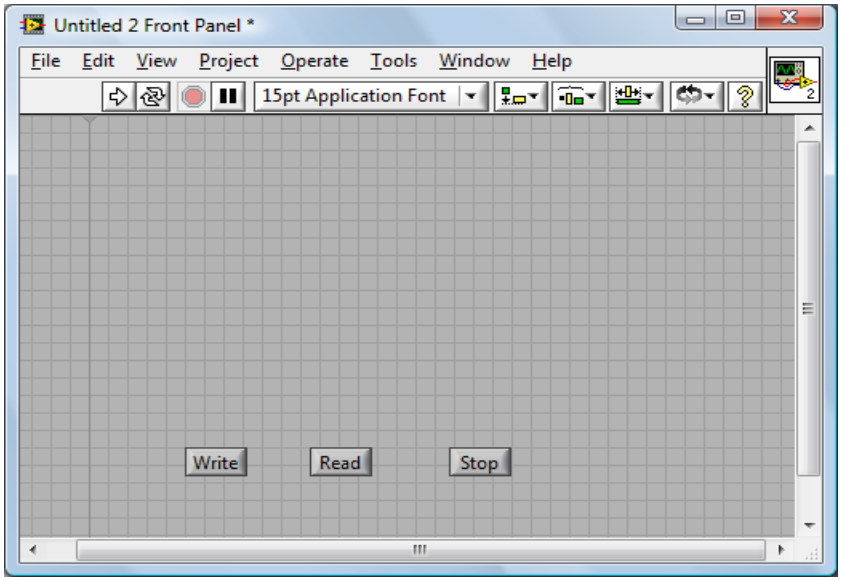


Figure 4-3-2

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.

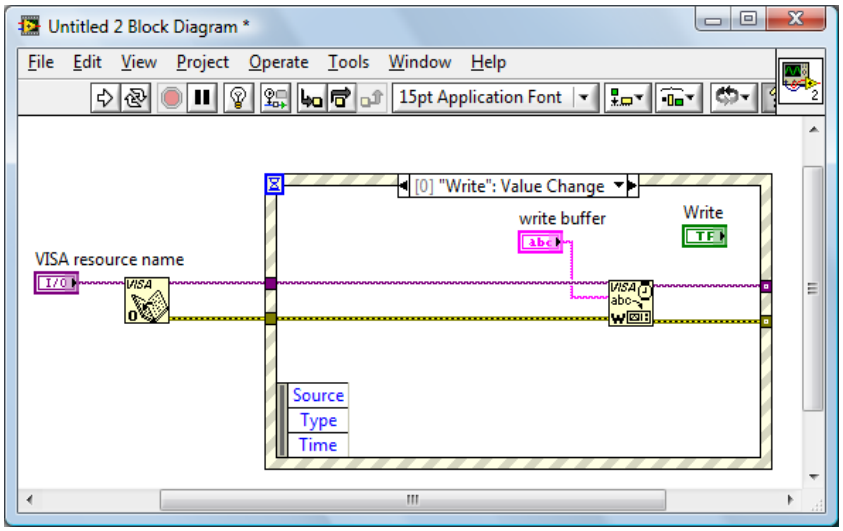


Figure 4-3-3

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.

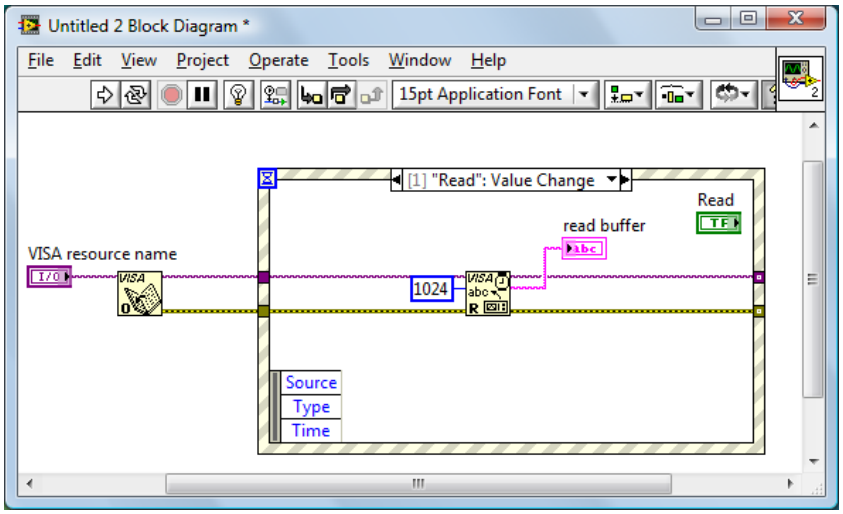


Figure 4-3-4

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.

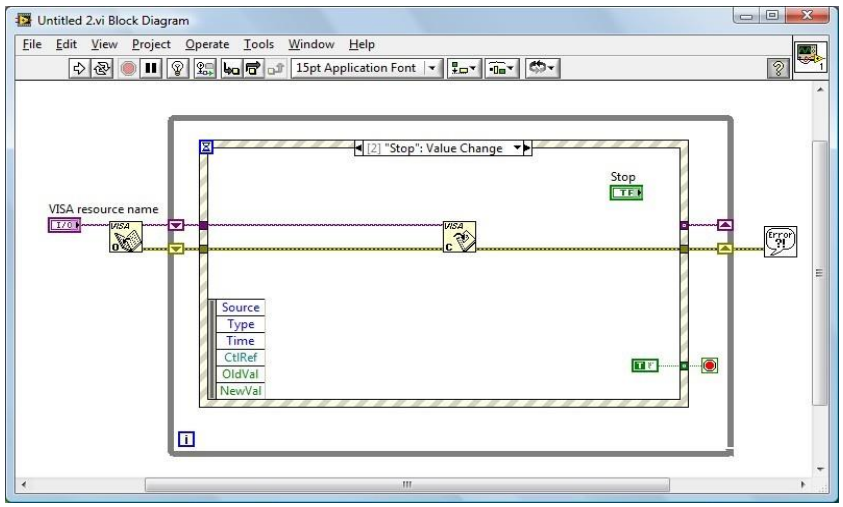


Figure 4-3-5

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.

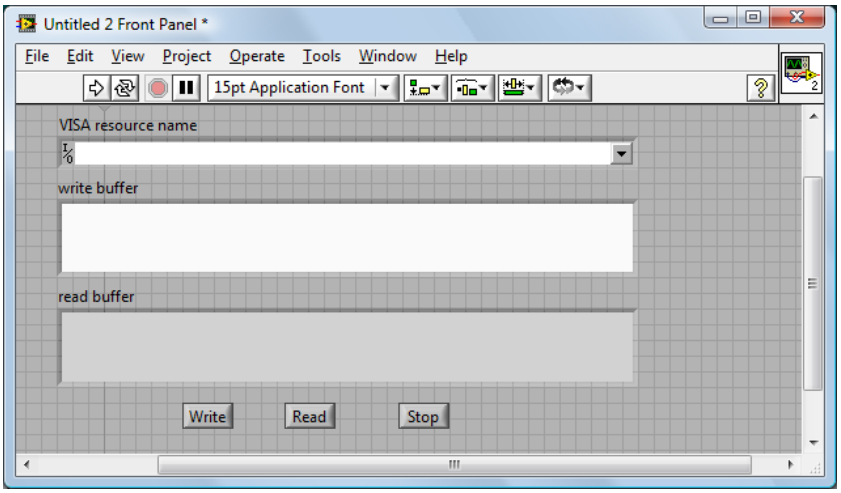


Figure 4-3-6

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.

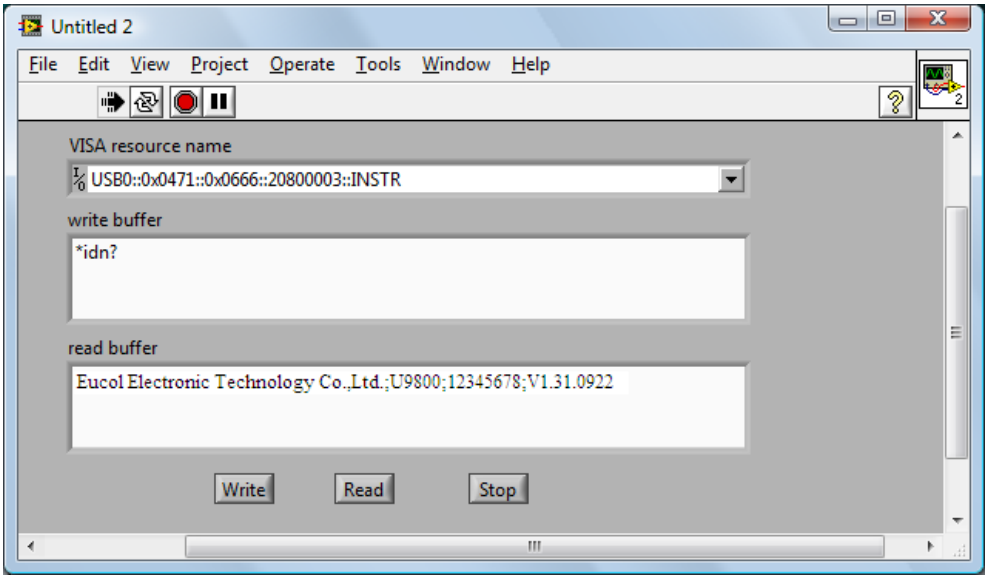


Figure 4-3-7

23. Click the Stop control to exit this program.