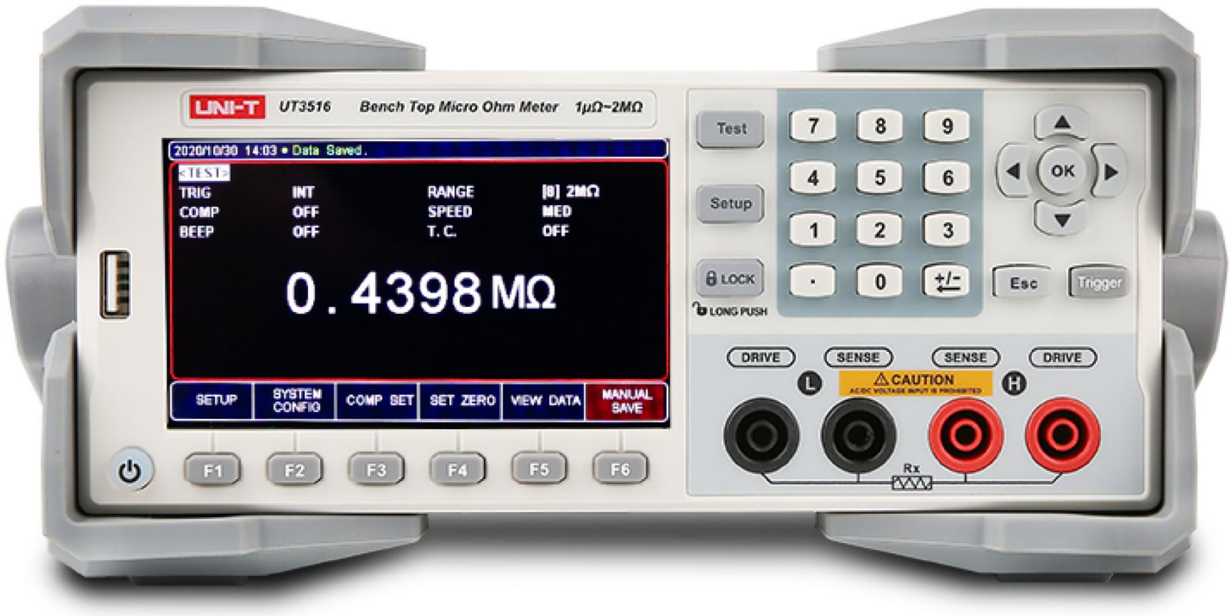
Instruments.uni-trend.com



Programming Manual

UT3510 Series

1. SCPI

This chapter contains the following contents.

- Command Parser — Learn about a certain rule of command parser
- Command Syntax — Write rule of command line
- Query Syntax — Query write rule of query command
- Query Response —Query the format of query respond
- Command Reference

This chapter provides all SCPI commands used by the instrument, so user can totally control all functions of the instrument through these command.

1.1 Parse Command String

The host computer can send a command string to the instrument, and the instrument parser will start to analysis the command when capture an end mark (\n) or buffer overflow.

For Example Valid command string:

```
AAA:BBB CCC;DDD EEE;:FFF
```

The instrument command parser is responsible for all command parsing and execution, and you must understand its parsing rules before writing a program.

1.1.1 Command Parse Rule

1. Command parser only parses and responds to ASCII data.
2. The end mark of SCPI command string must be NL (‘ \n’ ASCII 0x0A). The command parser does not start executing a command string until it receives an end mark or a buffer overflow.
3. If handshake command is open, the command parser sends each character back to the host as soon as it is received, and the host can continue to send the next character only after it receives this returned character.
4. The command parser will terminate the parsing immediately after parsing an error, and the current command will be invalidated.
5. The command parser will immediately terminate the current command string analysis when analyzing the query command and the followed character string will be ignored.
6. The command parser is not case-insensitive.
7. The command parser supports command abbreviation, abbreviation format see the following

section.

1.1.2 Symbol Stipulation and Definition

This section has some symbols that they are not the part of command tree, but for better understanding of command string.

< > Word in angle brackets represents the parameter of command

[ ] Word in square brackets represents the optional command.
{ } If the braces contains several parameter items, it indicates only one of them can be selected.

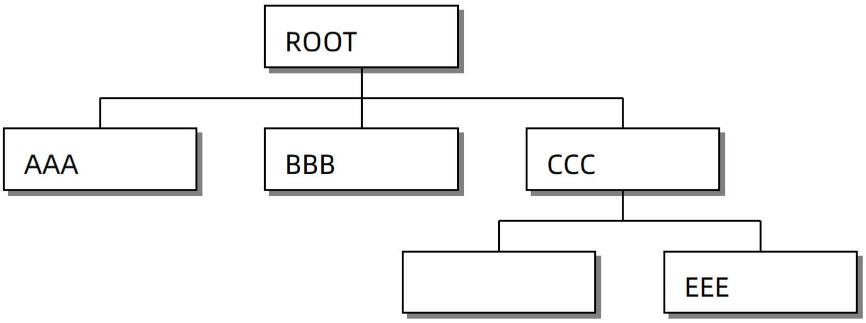
( ) The abbreviated form of the parameter is enclosed in parentheses.

Capital letter Abbreviation format of command.

1.1.3 Command Tree Structure

SCPI commands have a tree-like structure with three level (note: the command parser of this instrument can parse any level), where the highest level is called the subsystem command. Its subordinate can only be valid when subsystem command is selected. SCPI uses a colon (:) to separate high level commands from low level commands.

Figure 9- 1 Command Tree Structure



For Example

ROOT:CCC:DDD ppp
ROOT Sub-system command
CCC Second level
DDD Third level
ppp Parmeter

1.2 Command and Parameter

A command tree consists of **Command and [Parameter]**, use one blank (ASCII: 20H) to separate.

For Example

AAA:BBB 1.234
Command [Parameter]

1.2.1 Command

The command string can be long string form or abbreviation form, long string form is for engineer to understand string meaning; abbreviation form is for write.

1.2.2 Parameter

1. Single command word command, no parameter.

   For Example, AAA:BBB
2. Parameter can be character string form, and its abbreviation form should obey "command abbreviation rule" at last section.

   For Example, AAA:BBB 1.23
3. Parameter can be numeric value.

   <integer>: 123, +123, -123

   <float>: Floating point number

   <fixfloat>: Fixed floating point number , such as 1.23, -1.23

   <Sciloat>: Floating point number with scientific notation, such as 1.23E+4, +1.23e-4

   <mpfloat>: Floating point number with multiplying power, such as 1.23k, 1.23M, 1.23G, 1.23u

Table 9- 1 Abbreviation of Multiplying Power

Numeric Value	Multiplying power
1E18 (EXA)	EX
1E15 (PETA)	PE
1E12 (TERA)	T
1E9 (GIGA)	G
1E6 (MEGA)	MA
1E3 (KILO)	K
1E-3 (MILLI)	M
1E-6 (MICRO)	U
1E-9 (NANO)	N
1E-12 (PICO)	P
1E-15 (PEMTO)	F
1E-18 (ATTO)	A

i The multiplying power is not case-insensitive, so the written is different from standard name.

1.2.3 Separator

The command parser only receive allowed separator and other separators will make error of "Invalid separator".

These separators are semicolon mark, colon mark, question mark and space mark.

“;”: Semicolon mark is used to separate two commands.

For Example, AAA:BBB 100.0;CCC:DDD

“:”: Colon mark is used to separate command tree or restart command tree.

For Example, AAA:BBB:CCC 123.4;:DDD:EEE 567.8

“?”: Question mark is used to query.

For Example, AAA?

“□”: Space mark is used to separate parameter.

For Example, AAA:BBB□1.234

1.2.4 Error Code

Error Code	Description
*E00	No error
*E01	Bad command
*E02	Parameter error
*E03	Missing parameter
*E04	Buffer overrun
*E05	Syntax error
*E06	Invalid separator
*E07	Invalid multiplier
*E08	Numeric data error
*E09	Value too long
*E10	Invalid command
*E11	Unknown error

2. Command Reference

All commands is explained by the subsystem command order.

● DISPlay	Display subsystem
● FUNCtion	Function subsystem
● CORRection	Correction subsystem
● COMParator	Comparator subsystem
● SYSTem	System subsystem
● TRIGger	Trigger subsystem
● FETCh?	Fetch result subsystem
● ERRor	Error subsystem

Common Command

● IDN?	Query subsystem of instrument information
● TRG	Trigger and acquire data

2.1 DISPlay Subsystem

DISPlay subsystem is used to switch different display page or display a string of text in page hint tab.

Figure 9- 2 DISPlay Subsystem Tree

DISPlay	:PAGE	{TEST,SETUP(MSET),COMParator,CORRECTION(CSET),FILE,SYSTem,SYSTEMINFO(SINF)}
	:LINE	<string>

2.1.1 DISPlay:PAGE

DISP:PAGE is used to switch to the specified page.

Command Syntax	DISPlay:PAGE <Page name>
Parameter	< Page name > includes: **TEST** Measurement display page SETUP (MSET) Setup page COMParator Comparator page CORRection Correction page **FILE** File management page **SYSTem** System configuration page SYSTEMINFO (SINF) System information page
For Example	Send > disp:page setup <NL> // Switch to the setup page
Query Syntax	DISP:PAGE?
Query Response	< Page name > abbreviation test mset comp cset file syst sinf
For Example	Send > disp:page? <NL> Return > test<NL>

2.1.2 DISP:LINE

DISP:LINE is used to display a string of text in the tab on the page bottom. The text can be displayed up to 30 characters.

Command Syntax	DISPlay:LINE <*string*>
Parameter	<string> up to 30 characters
For Example	Send >DISP:LINE “This is a Comment.” <NL>

2.2 FUNCtion Subsystem

Figure 9- 3 FUNCtion Subsystem Tree

FUNCtion	:RANGe	{Range number, max, min}	
		:MODE	{AUTO,HOLD,NOMinal}
	:RATE	{SLOW,MED,FAST }	
	:TC	: COEFficient	<float>
		:REFEr	<float>

The parameter set by FUNCtion subsystem will not be saved in the system. It need to reset the setting when next boot up.

2.2.1 FUNCtion:RANGe

FUNC:RANG is used to set the range mode and range number.

Command Syntax	FUNCtion:RANGe {<Range number>,min,max}
Parameter	<Range number> 0~9 (AT517) 0~5 (AT517L) **min** indicates the minimum range **max** indicates the maximum range
For Example	Send >**FUNC:RANG 5**<NL> // Switch to range 5 (1 kΩ)
Query Syntax	FUNC:RANG?
Query Response	Range number0~9 (AT517)
For Example	Send >FUNC:RANGE? <NL> Return > 5<NL>

2.2.2 FUNCtion:RANGe:MODE

FUNC:RANG:MODE is used to switch the range mode.

Command Syntax	FUNCtion:RANGe:MODE {AUTO,HOLD(MANual),NOMinal}
For Example	Send >FUNC:RANG:MODE NOM<NL> // Switch to the nominal mode
Query Syntax	FUNC:RANG:MODE?
Query Response	{AUTO,HOLD,NOM}

2.2.3 FUNCtion:RATE

FUNC:RATE or FUNC:SPEED is used to set the test speed.

Command Syntax	FUNCtion:RATE {SLOW,MED,FAST}
For Example	Send >FUNC:RATE MED<NL> // Set to middle speed
Query Syntax	FUNC:RATE?
Query Response	{SLOW,MED,FAST}

2.2.4 FUNCtion:TC

FUNC:TC is used to turn on/off temperature compensation.

Command Syntax	FUNCtion:TC {ON,OFF,1, 0}
For Example	Send >FUNC:TC ON <NL> // Turn on temperature compensation
Query Syntax	FUNC:TC?
Query Response	{ON,OFF}

2.2.5 FUNCtion:TC: COEFficient

FUNC:TC:COEFficient is used to set the temperature coefficient.

Command Syntax	FUNCtion:TC: COEFficient <float> FUNCtion:TC: A <float> Note: The unit of temperature coefficients A is ppm, e.g. the temperature coefficient of silver-copper at 20°C is 3930 ppm.
For Example	Send >FUNC:TC:COEF 3930<NL> // Set the temperature coefficient to 3930 ppm Send >FUNC:TC:COEF 3930<NL> // Set the temperature coefficient to 3930 ppm
Query Syntax	FUNC:TC:COEF? FUNC:TC:A?
Query Response	<fixfloat>
For Example	Send >FUNC:Tc:A? <NL> Response > +3930.0<NL>

2.2.6 FUNCtion:TC:REFErence

FUNC:TC:REFE is used to set the reference temperature.

Command Syntax	FUNCtion:TC:REFErence <float> FUNCtion:TC:T0 <float>

	Note: The temperature unit is ℃.
For Example	Send >FUNC:TC:T0 25<NL> // Set the reference temperature to 25 ℃.
Query Syntax	FUNC:TC:REFE? FUNC:TC:T0?
Query Response	<fixfloat>
For Example	Send >FUNC:TC:REFE? <NL> Response > +20.00<NL>

2.2.7 FUNCtion:DT

FUNC:DC is used to turn on/off the temperature conversion function.

Command Syntax	FUNCtion:DT {ON,OFF,1, 0}
For Example	Send >FUNC:DC ON <NL> // Turn on the temperature conversion function.
Query Syntax	FUNC:DT?
Query Response	{ON,OFF}
Note	The temperature conversion function is only available when the temperature sensor is connected to the instrument.

2.2.8 FUNCtion:DT:T1

FUNC:DT:T1 is used to set the initial temperature.

Command Syntax	FUNCtion:DT:T1 <float>
For Example	Send >FUNC:DT:T1 20<NL> // Set the reference temperature to 25 ℃.
Query Syntax	FUNC:DT:T1?
Query Response	<fixfloat>
For Example	Send >FUNC:DT:T1? <NL> Response > +20.00<NL>
Note	1. The initial temperature T1 corresponds to the temperature at the initial resistance R1. 2. The temperature unit is ℃.

2.2.9 FUNCtion:DT:R1

FUNC:DT:R1 is used to set the resistance at the initial temperature.

Command Syntax	FUNCtion:DT:R1 <float>
For Example	Send >FUNC:DT:R1 100<NL> // Set the initial resistance to 100 Ω.
Query Syntax	FUNC:DT:R1?
Query Response	<Scifloat>
For Example	Send >FUNC:DT:R1? <NL> Response > 1.00000e+02<NL>
Note	1. The initial temperature T1 corresponds to the initial resistance R1. 2. The unit is Ω.

2.2.10 FUNCtion:DT:K

FUNC:DT:K is used to set the reciprocal of the temperature coefficient of the measured part at 0°C (1/α).

Command Syntax	FUNCtion:DT:K <float> FUNCtion:DT:K <float> Note: The temperature unit is ℃.
For Example	Send >FUNC:DT:K 234.5<NL>
Query Syntax	FUNC:DT:K?
Query Response	<fixfloat>
For Example	Send >FUNC:DT:K? <NL> Response > +234.5<NL>
Note	K is the reciprocal of the temperature coefficient (the standard is 0℃).

COMParator Subsystem

COMP subsystem is used to set the parameter of comparator.

Figure 9- 4 COMParator Subsystem Tree

COMParator	[:STATe]	{OFF,#-BIN} (AT517) {OFF,ON} (AT517L)
	:BEEP	{OFF,PASS(OK),FAIL(NG)}
	:MODE	{ABS,PER,SEQ}
	:NOMinal	<float>
	:BIN	<Scale number 1~10>, <float lower limit>, <float upper limit>

2.2.11 COMParator[:STATe]

COMP[:STATe] is used to turn off the comparator or set the scale number.

Command Syntax	COMParator[:STATe] {OFF,#-BIN} (AT517) COMParator[:STATe] {OFF,ON} (AT517L)
Parameter	<#-BIN> includes: **1-BIN ~ 6-BIN**
For Example	Send >COMP:STAT 6-BIN<NL> // Turn on the comparator and set to 6-BIN. Send >COMP:STAT OFF<NL> // Turn on the comparator.
Query Syntax	COMP[:STAT]?
Query Response	{OFF,#-BIN}

2.2.12 COMParator:BEEP

COMP:BEEP is used to turn on the beeper.

Command Syntax	COMParator:BEEP {OFF,OK,NG}
For Example	Send >COMP:BEEP OK<NL> // Qualified beeper.
Query Syntax	COMP:BEEP?
Query Response	{OFF,OK,NG}

2.2.13 COMParator:MODE

COMP:MODE is used to set the comparator mode.

Command Syntax	COMParator:MODE {ABS,PER,SEQ}
For Example	Send >**COMP:MODE SEQ** // Switch to sequence compare mode.
Query Syntax	COMP:MODE?
Query Response	{ABS,PER,SEQ}

2.2.14 COMParator:NOMinal

COMP:NOM is used to set the nominal value.

Command Syntax	COMParator:NOMinal <float>
For Example	Send >COMP:NOM 1.0000k // Set the nominal value to 1 k. Send >**COMP:NOM 1E3** // Set the nominal value to 1 k. Send >**COMP:NOM 1000** // Set the nominal value to 1 k.
Query Syntax	COMP:NOM?
Query Response	<scifloat>
For Example	Send >COMP:NOM? <NL> Return > 1.0000E+03<NL> // Set the nominal value to 1 k.

2.2.15 COMParator:BIN

COMP:BIN is used to set the nominal value.

Command Syntax	COMParator:BIN <Scale number1~6>,<float lower limit>,<float upper limit> (*AT517) COMParator:BIN <float lower limit>,<float upper limit> (*AT517L)

For Example	Send >**COMP:BIN 1,-10,+10** // If in percentage sorting mode: the lower limit of BIN1 is -10%, the upper limit is 10%.
Query Syntax	COMP:BIN? <1~6>
Query Response	<scifloat>,<scifloat>
For Example	Send >COMP:BIN? 1*<NL>* Return > -10.000E+00,+10.000E+00*<NL>*//-10,+10

2.3 TRIGger Subsystem

Figure 9- 5 TRIGger Subsystem Tree

TRIGger	[:IMMediate]	
	:SOURce	{INT,EXT}
	:DELAy	<float>
TRG		

TRIGger is used to set the trigger source and to generate one trigger.

2.3.1 TRIGger[:IMMediate]

TRIG[:IMM] generates one trigger when the trigger source is set to EXT, but does not return the data of trigger test. The TRG instruction is required to return data.

Command Syntax	TRIGger[IMMediate]
For Example	Send >**TRIG*<NL>*** // The instrument will stop after one test.

2.3.2 TRIGger:SOURce

TRIG:SOUR is used to set the trigger source.

Command Syntax	TRIGger:SOURce {INT,EXT}
For Example	Send >TRIG:SOUR BUS*<NL>* // Set to bus trigger mode.
Query Syntax	TRIG:SOUR?
Query Response	<INT,EXT>

2.3.3 TRIGger:DELAy

TRIG:DELAy is used to set the trigger delay.

Command Syntax	TRIGger:DELAy {0,<float>} Parameter 0: turn off the trigger delay. Parameter <float>: 0.001~9.0
For Example	Send >TRIG:DELA 0.1*<NL>* // Set the trigger delay to 0.1s. Send >TRIG:DELA 10m*<NL>* // Set the trigger delay to 10ms.
Query Syntax	TRIG:DELA?
Query Response	0.1

2.3.4 TRG

When TRG (trigger source) sets to EXT, it generates one trigger and return the data of trigger test.

Command Syntax	TRG
For Example	Send >**TRG*<NL>*** // The instrument performs one time and return test data. Return > +9.9651e+01,BIN00 *<NL>*

2.4 FETCh? Subsystem

FETCh? is used to acquire test data. Before using this command, [Result Sending] field under the <System Configuration> screen should set to [FETCH].

FETCh? command will return test data.

Figure 9- 6 FETCh? Subsystem Tree

FETCh?	<NONE>
	:RT?
	:T2?

2.4.1 FETCh?

Fetch the measured result.

Query Syntax	FETCh?
Query Response	<scifloat>, {BIN0~BIN6 } BIN0 indicates unqualified.
For Example	Send >FETC? <NL> Return >+9.9651e+01,BIN0<NL>

2.4.2 FETCh:RT?

Fetch the current room temperature.

Query Syntax	FETCh:RT?
Query Response	<Fixfloat>
For Example	Send >FETC:RT? <NL> Return >+27.94<NL>
Note	1. If sensor is not inserted, or temperature compensation and temperature conversion are not turned on, +999.99 will be returned. 2. The temperature is ℃.

2.4.3 FETCh:T2?

Fetch the current room temperature.

Query Syntax	FETCh:T2?
Query Response	<Fixfloat>
For Example	Send >FETC:T2? <NL> Return >+19.91<NL>
Note	1. If sensor is not inserted, or temperature compensation and temperature conversion are not turned on, +999.99 will be returned. 2. The temperature is ℃.

2.5 SYSTem Subsystem

SYSTem subsystem is used to set the parameter of system.

The data set by SYSTem subsystem will not be saved internal of the instrument.

Figure 9- 7 SYSTem Subsystem Tree

SYSTem	:LANGuage	{ENGLISH,CHINESE,EN,CN}
	:TIME	<YEAR>,<MONTH>,<DAY>,<HOUR>,<MINUTE>,<SECOND>
	:KEYLock(KLOC)	{ON(1),OFF(0)}
	:BEEP	{ON(1),OFF(0)}
	:SHAKEHAND(SHAK)	{ON(1),OFF(0)}
	:UPLOAD(UPLD)	{FETCh,AUTO}

2.5.1 SYSTem:LANGuage

Set the instrument’s system.

Command Syntax	SYSTem:LANGuage {ENGLISH,CHINESE,EN,CN}
For Example	Send >SYST:LANG EN // Set to English.
Query Syntax	SYST:LANG?
Query Response	{ENGLISH,CHINESE}

2.5.2 SYSTem:TIME

Set the system's time.

Command Syntax	SYSTem:TIME <YEAR>,<MONTH>,<DAY>,<HOUR>,<MINUTE>,<SECOND>
For Example	Send >SYST:TIME 2016,12,30,11,18,31 // 2016-12-30 11:18:31
Query Syntax	SYSTem:TIME?
Query Response	<YEAR>-<MONTH>-<DAY> <HOUR>:<MINUTE>:<SECOND>
For Example	Send >SYST:TIME? Receive > 2016-12-30 11:18:31

2.5.3 SYSTem:KEYLock or SYSTem:KLOCk

Key lock setup.

Command Syntax	SYSTem:KEYLock {ON,OFF,0,1}
	SYSTem:KLOCk {ON,OFF,0,1}

For Example	Send >SYST:KEYL OFF // Unlock the key.
Query Syntax	SYSTem:KEYLock? SYSTem:KLOCk?
Query Response	{on,off}

2.5.4 SYSTem:BEEPer

Key sound, this command is not affect the comparator's beeper.

Command Syntax	SYSTem:BEEPer {OFF,ON,0,1}
Parameter	{OFF,ON,0,1} OFF/0: Turn off the beeper ON/1: Turn on the beeper
For Example	Send >SYST:BEEP OFF
Query Syntax	SYSTem:BEEPer?
Query Response	{OFF,ON}

2.5.5 SYSTem:SHAKhand (Return data header)

When the communication handshake is opened, the instrument will original return the received commands to the host computer, and then returns the data.

Command Syntax	SYSTem:SHAKhand {ON,OFF,0,1}
	SYSTem:HEADer {ON,OFF,0,1}

For Example	Send >SYST:SHAK ON Send >SYST:HEAD ON
Query Syntax	SYSTem:SHAKhand? SYSTem:HEADer?
Query Response	{on,off}

2.5.6 SYSTem:UPLOAD (UPLD) (Send test result)

SYSTem:UPLOAD (UPLD) can set the send mode for data, automatic or FETCH.

Command Syntax	SYSTem:UPLOAD {FETCH,AUTO}
Parameter	{FETCH,AUTO} FETCH: The data needs to be returned to the host by the command fetch?, the instrument is passively sent. AUTO: The data is automatically sent to the host computer after each test is completed, and the instrument proactively send the results.
For Example	Send >SYST:UPLD AUTO // Set to automatic send
Query Syntax	SYST:UPLD?

Query Response	{FETCH,AUTO}

2.6 CORRect Subsystem

CORR subsystem is used to complete one short-circuit correction.

Figure 9- 8 CORRect Subsystem Tree

CORRect	:STATe	{ON,OFF,0,1}
	:SHORt	

2.6.1 CORRect:STATe

Command Syntax	SYSTem:STATe {OFF,ON,0,1}
Parameter	{OFF,ON,0,1} OFF/0: Turn off the short-circuit zero clearing ON/1: Short-circuit zero clearing is valid.
For Example	Send >SYST:STAT OFF
Query Syntax	SYSTem:STAT?
Query Response	{OFF,ON}

2.6.2 CORRect:SHORt

CORR:SHOR is used to complete one short-circuit correction. Before sending this command, the test terminal should be short-circuit.

Command Syntax	CORRect:SHORt
For Example	Send >CORRect:SHORt<NL> Return > Short Clear Zero Start. <NL> // Hint: Zero clearing is start. Return > **PASS**<NL> // Hint: Zero clearing is pass (FAIL)

2.7 FILE (MMEM) Subsystem

FILE (MMEM) subsystem is used to manage file, the user can save the parameter to internal flash memory or read the flash file to the system.

Figure 9- 9 FILE (MMEM) Subsystem Tree

FILE MMEM	:SAVE	<No parameter> or <File number 0-9>
	:LOAD	<No parameter> or <File number 0-9>
	:DELete	<File number 0-9>

2.7.1 **FILE:SAVE** (Save file)

FILE:SAVE can save the current settings to the current file or the specified file.

Command Syntax	FILE:SAVE FILE:SAVE <File No. 0-9>
For Example	Send >**FILE:SAVE** // Save to the current file Send >**FILE:SAVE 1** // Save to the File 1

2.7.2 **FILE:LOAD** (Read file)

FILE:LOAD can read file data to the system.

Command Syntax	FILE:LOAD FILE:LOAD <File No. 0-9>
For Example	Send >**FILE:LOAD** // Read the current file data to the system. Send >**FILE:LOAD 1** // Read the data of File 1 to the system.

2.7.3 **FILE:DELete** (Delete the specified file)

FILE:DELete can delete the data of the specified file.

Command Syntax	FILE:DELete <File No. 0-9>
For Example	Send >**FILE:DEL 1** // Delete the specified file.
Note	Delete the current file will not affect the system parameter.

2.7.4 SAV

SAV can save the current settings to the current file.

Command Syntax	SAV = FILE:SAVE
For Example	Send >**SAV** // Save the current settings to the current file.

2.7.5 RCL

RCL can read the current file data to the system.

Command Syntax	RCL = FILE:LOAD
For Example	Send >**FILE:LOAD** // Read the current file data to the system.

2.8 IDN? Subsystem

Figure 9- 10 IDN? Subsystem Tree

IDN?	IDN? subsystem is used to return the version number of instrument.

Query Syntax	IDN?
Query Response	<MODEL>,<Revision>,<SN>,< Manufacturer>
For Example	Send >IDN? <NL> Return > UT3513,REV A1.0,0000000,UNI-T<NL>

2.9 ERRor Subsystem

Error subsystem is used to acquire the previous error information.

Query Syntax	ERRor?
Query Response	Error string
For Example	Send >ERR?<NL> Return > no error.<NL>

Error Code

Error Code	Description
*E00	No error
*E01	Bad command
*E02	Parameter error
*E03	Missing parameter
*E04	buffer overrun
*E05	Syntax error
*E06	Invalid separator
*E07	Invalid multiplier
*E08	Numeric data error
*E09	Value too long
*E10	Invalid command
*E11	Unknown error

3. Modbus (RTU) Communication Protocol

&

This chapter contains the following contents.

- Data Format — Learn about the communication format of Modbus
- Function
- Variable Region
- Function Code

3.1 Data Format

Following Modbus (RTU) communication protocol, the instrument responds to the instruction of upper computer and returns the standard response frame.

3.1.1 Command Frame

Figure 10- 1 Modbus Command Frame

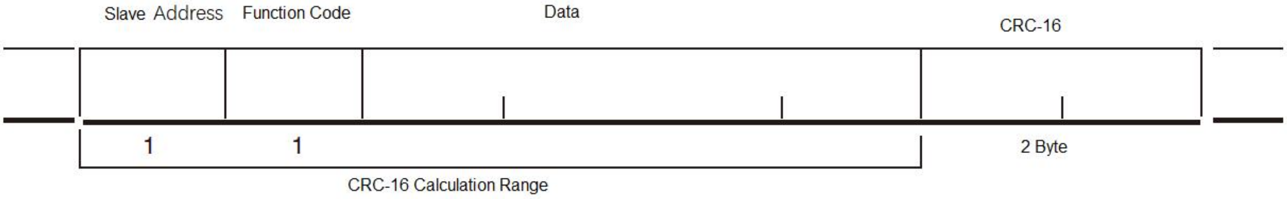


Table 10- 1 Description of Command Frame

	It needs mute interval time of 3.5 character at least.
Slave-station Address	1 byte Modbus supports 00~0x63 slave station It is specified as 00 for uniform broadcasting If the instrument does not have optional RS485, the default slave station address is 0x01
Function Code	1 byte 0x03: read multiple registers 0x04: =03H, not use 0x06: write a single register, which can replace by 10H 0x08: echo test (only for debugging) 0x10: write multiple registers
Data	The specified register address, quantity and content
CRC-16	2 bytes, LSB (least significant bit) Cyclic Redundancy Check Calculating all the data from slave station address to the last data, get CRC-16 check code

3.1.2 CRC-16 Calculation Method

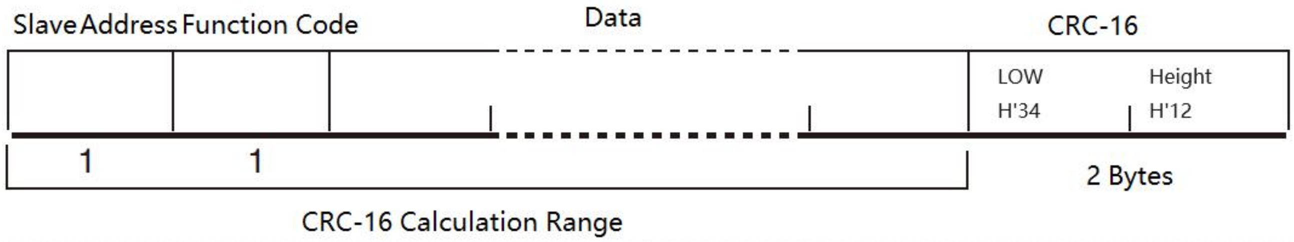
1. Set the initial value of CRC-16 register to 0xFFFF.
2. Performs an XOR operation on the CRC-16 register and the first byte of the message, and returns the result to the CRC register.
3. Fill the MSB with zero and shift the CRC register to right by 1 bit.
4. If the bit shifted from LSB is “0”, repeat step 3 (process the next shift bit). If the bit shifted from LSB is “1”, XOR operation is performed on CRC register and 0xA001, and the result is returned to CRC register.
5. Repeat execute the step 3 and step 4 until move 8 bits.
6. If the information processing is not finished yet, then perform an XOR operation on the CRC-register and the next Byte of the message and return to the CRC register. It repeat from the step 3.
7. The result of the calculation (the value of the CRC Register) is appended to the information from the lower Byte.

The following is a CRC calculation function of VB language.

```
Function CRC16(data() As Byte) As Byte()
 im CRC16Lo As Byte, CRC16Hi As Byte    'CRC register
 im CL As Byte, CH As Byte              'polynomial code &HA001
 im SaveHi As Byte, SaveLo As Byte
 im i As Integer
 im flag As Integer
 RC16Lo = &HFF
 RC16Hi = &HFF
 L = &H1
 H = &HA0
 or i = 0 To UBound(data)
        CRC16Lo = CRC16Lo Xor data(i) 'XOR each data and CRC register
        For flag = 0 To 7
            SaveHi = CRC16Hi
            SaveLo = CRC16Lo
            CRC16Hi = CRC16Hi \ 2       'move high bit to right by one bit
            CRC16Lo = CRC16Lo \ 2        ' move low bit to right by one bit
            If ((SaveHi And &H1) = &H1) Then 'if the last bit of high bit is 1
                CRC16Lo = CRC16Lo Or &H80    'then low bit move to right and fill 1
on the front
            End If                  'otherwise, it automatically fill 0
            If ((SaveLo And &H1) = &H1) Then 'IF LSB IS 1, oxr polynomial code
                CRC16Hi = CRC16Hi Xor CH
                CRC16Lo = CRC16Lo Xor CL
            End If
        Next flag
 ext i
 im ReturnData(1) As Byte
 eturnData(0) = CRC16Hi         'CRC high bit
 eturnData(1) = CRC16Lo         'CRC low bit
 RC16 = ReturnData
End Function
```

Caculated CRC-16 data should append to the end of command frame.
For example, 1234H:

Figure 10- 2 Modbus Additional CRC-16 Value



3.1.3 Response Frame

Except the instruction of 00H slave address boardcast, other slave station address will returns response frame.

Figure 10- 3 Normal Response Frame

Figure 10- 4 Exceptional Response Frame

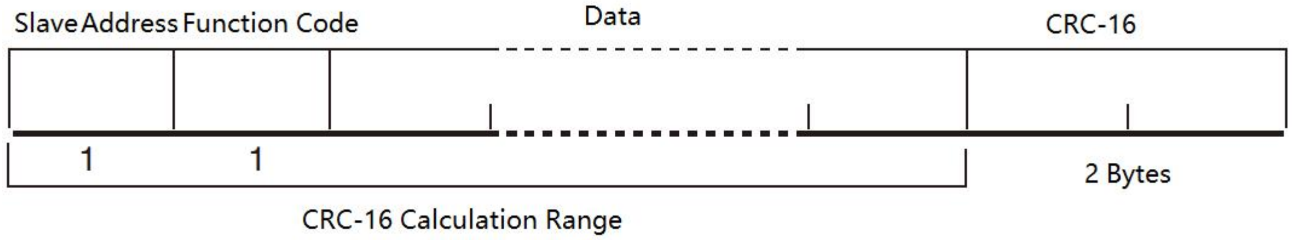


Table 10- 2 Description Exceptional Response Frame

Slave-station Address	1 byte Original return slave-station address
Function Code	1 byte 0x03: read multiple registers 0x04: =03H, not use 0x06: write a single register, which can replace by 10H 0x08: echo test (only for debugging) 0x10: write multiple registers
Error Code	Exceptional code 0x01 Function code error (function code does not support) 0x02 Register error (Register does not exist) 0x03 Data error 0x04 Execution error
CRC-16	2 bytes, LSB (least significant bit) Cyclic Redundancy Check Calculating all the data from slave station address to the last data, get CRC-16 check code

3.1.4 No Response

The instrument does not handle and response any case as follows, it may occurs communication time-out.

1. Slave station address error
2. Transmission error
3. CRC-16 error
4. Bit error, for example, total bit of function code 0x03 must be 8 and received bit should less than or greater than 8 bytes.
5. It represents broadcast address when the slave station is 0x00. The instrument has no response.

3.1.5 Error Code

Table 10- 3 Description of Error Code

Error Code	Name	Description	Priority
0x01	Function code error	Function code does not support	1
0x02	Register error	Register does not exist	2
0x03	Data error	Quantity of register or byte error	3
0x04	Execution error	Invalid data, write data is not in the allowed range	4

3.2 Function Code

The instrument can only support several function code. The other function code will responses error frame.

Table 10- 4 Function Code

Function Code	Name	Description
0x03	Read multiple registers	Read data of multiple consecutive register
0x04	Same with 0x03	Replace by 0x03
0x08	Echo test	Original return received data
0x10	Write multiple registers	Write multiple consecutive register

3.3 Register

The register quantity of the instrument is 2-byte mode, it requires that it must write 2 bytes for each time, for example, speed register is 0x3002, data is 2 bytes, and the numerical value must be written to 0x0001.

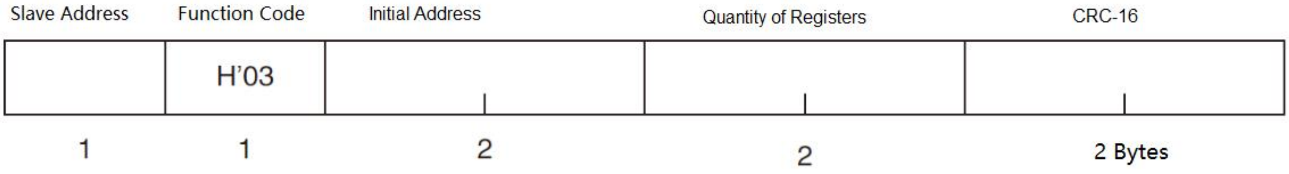
Data:

The instrument supports the following numerical value.

1. 1 register, double byte (16 bits) integer, for example, 0x64 → 00 64
2. 2 registers, four bytes (32 bits) integer, for example, 0x12345678 → 12 34 56 78
3. 2 registers, four bytes (32 bits) single float-point number, 3.14 → 40 48 F5 C3

3.4 Read Multiple Registers

Figure 10- 5 Read Multiple Registers (0x03)

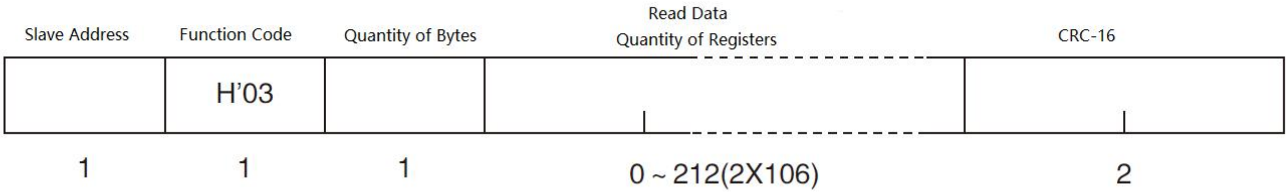


Read out the function code of multiple register is 0x03.

Table 10- 5 Read Multiple Registers

Name	Name	Description
	Slave station address	If the RS485 address is not specified, the default is 01.
0x03	Function code	
	Initial address	The initial address of register refer to Modbus instruction set.
	Quantity of read multiple registers 0001~006A (106)	Continuously read quantity of register refer to Modbus instruction set. To make sure all register address are exit, otherwise it returns error frame.
CRC-16	Check code	

Figure 10- 6 Read Multiple Register (0x03) Response Frame



Name	Name	Description
	Slave station address	Original return
0x03 or 0x83	Function code	No exceptional: 0x03 Error code: 0x83
	Byte number	= quantity of register x 2 For Example, 1 register returns 02
	Data	Read data
CRC-16	Check code	

3.5 Write Multiple Registers

Figure 10- 7 Write Multiple Registers (0x10)

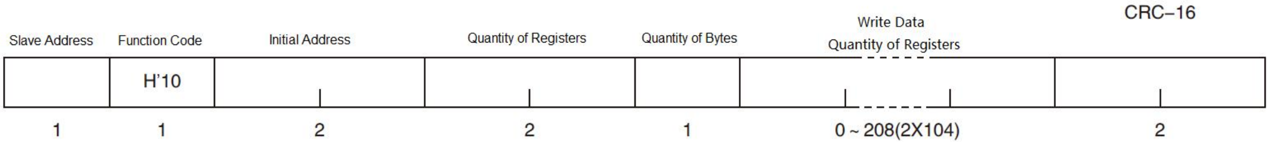
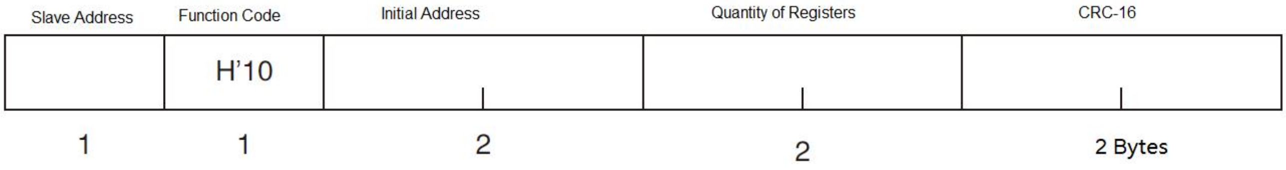


Table 10- 6 Write Multiple Registers

Name	Name	Description
	Slave station address	If the RS485 address is not specified, the default is 01.
0x10	Function code	
	Initial address	The initial address of register refer to Modbus instruction set.
	Quantity of write multiple registers 0001~0068 (104)	Continuously read quantity of register refer to Modbus instruction set. To make sure all register address are exit, otherwise it returns error frame.
	Byte number	= quantity of register x2
CRC-16	Check code	

Figure 10- 8 Write Multiple Registers (0x03) Response Frame



Name	Name	Description
	Slave station address	Original return
0x10 or 0x90	Function code	No exceptional: 0x10 Error code: 0x90
	Initial address	
	Quantity of register	
CRC-16	Check code	

3.6 Echo Test

The function code of echo test is 0x08, it used to debug Modbus.

Figure 10- 9 Echo Test (0x08)

Command

Slave Address	Function Code	Fixed Value	Test Data	CRC-16
	H’08	H’00 H’00		
1	1	2	2	2 Bytes

Response

Slave Address	Function Code	Fixed Value	Test Data	CRC-16
	H’08	H’00 H’00		
1	1	2	2	2 Bytes

Name	Name	Description
	Slave station address	Original return
0x08	Function code	
	Fixed value	00 00
	Test data	Arbitrary numerical value, such as 12 34
CRC-16	Check code	

For example, assume that the test data is 0x1234

Command:	01	08	00 00	12 34	ED 7C(CRC–16)
Response:	01	08	00 00	12 34	ED 7C(CRC–16)

4. Modbus (RTU) Instruction Set

This chapter contains the following content.

- Register Address

! Unless otherwise specified, numerical value of the instruction and response frame are all hexadecimal data.

4.1 Register Overview

The following lists all register addresses used by the instrument, any address not in the Table 11-1 will return error code 0x02.

Table 11- 1 Register Overview

Register Address	Name	Numerical Value	Description
2000	Read the measured resistance value	4 bytes float-point number	Read-only, data occupies 2 registers, 4 bytes. Byte sequence is ABCD, LSB
2100	Read the comparator result of channel	4 bytes integer	Read-only, data occupies 2 registers.
2200	Read the measured result	4 bytes float-point number	Read-only, data occupies 2 registers, 4 bytes. Byte sequence is CDAB.
2300	Trigger one time and read the measured result AABB CCDD	4 bytes float-point number with single precision Bit sequence: LSB AABB CCDD	Read-only, data occupies 2 registers, 4 bytes. It will automatically go to the measurement page when receive the command, and the trigger mode will be switched to remote trigger.
2400	Trigger one time and read the measured result CCDD AABB	4 bytes float-point number Bit sequence: LSB CCDD AABB	Read-only, data occupies 2 registers, 4 bytes. It will automatically go to the measurement page when receive the command, and the trigger mode will be switched to remote trigger.
0000	Read the version number of the instrument	4 bytes integer	Read-only, data occupies 2 registers, 4 bytes.
3000	Range number	0000~0009	Read and write register, 2 bytes integer.
3001	Auto range	0000: auto 0001: manual 0002: nominal	Read and write register, 2 bytes integer.
3002	Test speed	0000: slow 0001: middle 0002: fast	Read and write register, 2 bytes integer.

3003	Boot load file	0000: file 0 0001: the current file	Read and write register, 2 bytes integer.
3004	Automatically save	0000: forbidden 0001: allow	Read and write register, 2 bytes integer.
3005	System language	0000: English 0001: simplified Chinese	Read and write register, 2 bytes integer.
3006	Beeper	0000: OFF 0001: qualified beeper 0002: unqualified beeper	Read and write register, 2 bytes integer.
3008	Trigger	0000: internal trigger 0003: external trigger	Read and write register, 2 bytes integer.
3009	Trigger delay	0: turn off trigger delay Range of 4 bytes float-point number: 0.1~9.0s	Read and write register, 2 bytes integer.
3100	Scale number of comparator	0000: turn off the comparator 0001: 1-BIN 0002: 2-BIN 0003: 3-BIN 0004	Read and write register, 2 bytes integer.
3101	Comparator mode	0000: ABS 0001: PER 0002: SEQ	Read and write register, 2 bytes integer.
3102	Nominal	4 bytes float-point number	Read and write register, data occupies 2 registers.
3110	Lower limit of BIN1	4 bytes float-point number	Read and write register, data occupies 2 registers.
3112	Upper limit of BIN1	4 bytes float-point number	Read and write register, data occupies 2 registers.
3114	Lower limit of BIN2	4 bytes float-point number	Read and write register, data occupies 2 registers.
3116	Upper limit of BIN2	4 bytes float-point number	Read and write register, data occupies 2 registers.
3118	Lower limit of BIN3	4 bytes float-point number	Read and write register, data occupies 2 registers.
311A	Upper limit of BIN3	4 bytes float-point number	Read and write register, data occupies 2 registers.
311C	Lower limit of BIN4	4 bytes float-point number	Read and write register, data occupies 2 registers.
311E	Upper limit of BIN4	4 bytes float-point number	Read and write register, data occupies 2 registers.
3120	Lower limit of BIN5	4 bytes float-point number	Read and write register, data occupies 2 registers.
3122	Upper limit of BIN5	4 bytes float-point number	Read and write register, data occupies 2 registers.
3124	Lower limit of BIN6	4 bytes float-point	Read and write register, data

		number	occupies 2 registers.
3126	Upper limit of BIN6	4 bytes float-point number	Read and write register, data occupies 2 registers.
4000	Save the settings to the current file	Fixed value: 0001	Write-only register, data is 2 bytes.
4001	Read the current file data	Fixed value: 0001	Write-only register, data is 2 bytes.
4002	Save the settings to the specified file	0000~0009	Write-only register, data is 2 bytes.
4003	Read the specified file data	0000~0009	Write-only register, data is 2 bytes.
5000	Execute zero clearing register Read the state of zero clearing register	Read: 0001: zero clearing 0000: zero clearing is success	Read-only register, data occupies 1 register.
5001	Key lock	0000: unlock 0001: lock	Write-only register, 2 bytes.
5002	Trigger one time = Handler Trig pin	Fixed value: 0001	Write-only register, 2 bytes.

4.2 Fetch Measured Data

4.2.1 Fetch Measured Data

Register 2000~2003 is used to fetch measured data of the instrument.
For example, fetch measured data

Command

1	2	3	4	5	6	7	8
01	03	2000		0002		CRC-16	
Slave station	Read	Register		Quantity of register		Check code	

Response

1	2	3	4	5	6	7	8	9
01	03	Byte	Float-point number with single precision				CRC-16	

- Fetch Measured Data

Send

1	2	3	4	5	6	7	8
01	03	20	00	00	02	CF	CB

Response

1	2	3	4	5	6	7	8	9
01	03	04	60	AD	78	EC	56	5F

B4~B6 is measured data: 60AD78EC indicates float-point number with single precision, LSB.

Byte sequence: A BB CC DD, convert to decimal numeral 1E20.

4.2.2 Fetch Comparator Result [2100]

Returned 4 bytes integer indicates the comparator result.

00: unqualified

01: qualified 1

02: qualified 2

03: qualified 3

04: qualified 4

05: qualified 5

06: qualified 6

Send

1	2	3	4	5	6	7	8
01	03	21	00	00	02	CE	37

Response

1	2	3	4	5	6	7	8	9
01	03	04	00	00	00	00		

4.2.3 Fetch Measured Result (CCDD AABB) [2200]

Register 2200~2203 is used to fetch measured data of instrument.

For example, fetch measured data.

Command

1	2	3	4	5	6	7	8
01	03	2200		0002		CRC-16	
Slave station	Read	Register		Quantity of register		Check code	

Response

1	2	3	4	5	6	7	8	9
01	03	Byte	Float-point number with single precision				CRC-16	

- Fetch Measured Data

Send

1	2	3	4	5	6	7	8
01	03	22	00	00	02	CE	73

Response

1	2	3	4	5	6	7	8	9
01	03	04	43	8D	3F	80	6F	CC

B4~B6 is measured data: 43 8D 3F 80 indicates float-point number with single precision, MSB.

Byte sequence: CC DD AA BB

Exchange byte sequence AABBCCDD: 3F 80 8D 43 convert to decimal numeral

1.002061*4862442017*

4.2.4 Trigger One Time and Return Measured Result (AABB CCDD) [2300]

Register 2300~2303 is used to fetch measured data of instrument.

For example, fetch measured data.

Command

1	2	3	4	5	6	7	8
01	03	2300		0002		CRC-16	
Slave station	Read	Register		Quantity of register		Check code	

Response

1	2	3	4	5	6	7	8	9
01	03	Byte	Float-point number with single precision				CRC-16	

- Fetch Measured Data

Send

1	2	3	4	5	6	7	8
01	03	23	00	00	02	CF	8F

Response

1	2	3	4	5	6	7	8	9
01	03	04	3F	80	44	98	C5	65

B4~B6 is measured data: 3F80 4498 indicates float-point number with single precision, LSB.

Byte sequence: AA BB CC DD convert to decimal numeral 1.002093*3151245117*.

i It will automatically go to the measurement page when receive this command, and the trigger mode will be switched to remote trigger.

4.2.5 Trigger One Time and Return Measured Result (CCDD AABB) [2400]

Register 2400~2403 is used to fetch measured data of instrument.

For example, fetch measured data.

Command

1	2	3	4	5	6	7	8
01	03	2400		0002		CRC-16	
Slave station	Read	Register		Quantity of register		Check code	

Response

1	2	3	4	5	6	7	8	9
01	03	Byte	Float-point number with single precision				CRC-16	

- Fetch Measured Data

Send

1	2	3	4	5	6	7	8
01	03	24	00	00	02	CF	CB

Response

1	2	3	4	5	6	7	8	9
01	03	04	44	CE	3F	80	9F	6C

B4~B6 is measured data: 44CE 3F80 indicates float-point number with single precision, MSB.

Adjusting byte sequence CCDD AABB to AABBCCDD, that is 3F8044CE converts to decimal numeral 1.0020997524261475.

4.3 Parameter Setup

4.3.1 Speed [3002]

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	30	02	00	01	02	00	01	56	71
	Write	Register		Quantity of register		Byte	Data		CRC	

Response

1	2	3	4	5	6	7	8
01	10	30	02	00	01	AF	09
		Register		Quantity of register		CRC	

Read

1	2	3	4	5	6	7	8
01	03	30	02	00	01	2A	CA
	Read	Register		Quantity of register		CRC	

Response

1	2	3	4	5	6	7
01	03	02	00	00	B8	44
		Byte	Data		CRC	

0000: slow speed

0001: middle speed

0002: fast speed

0003: high speed

4.4 Comparator Setup

The register address of comparator parameter is from 3100.

4.4.1 Nominal Value [3102-3103]

The nominal value uses 2 registers, 3102 and 3103.

Note: 3103 cannot read alone.

Write

100E-3 (Float-point number with single precision: 0x3DCCCCCD)

1	2	3	4	5	6	7	8	9	10	11	12	13
01	10	31	02	00	02	04	3D	CC	CC	CD	72	E1
	Write	Register		Quantity of register		Byte	Data				CRC	

Response

1	2	3	4	5	6	7	8
01	10	31	02	00	02	EE	F4
		Register		Quantity of register		CRC	

Read

1	2	3	4	5	6	7	8
01	03	31	02	00	02	6B	37
	Read	Register		Quantity of register		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	3D	CC	CC	CD	A3	35
		Byte	Data 100E-3				CRC	

4.4.2 Limit Value [3110-3126]

The limit value of 6-scale comparator is start from 3110 and end with 3126. Each comparator uses 2 registers for the lower limit and 2 registers for the upper limit, a total of 4 registers.

The lower limit and the upper limit can separately set or set at the same time.

- Write

Lower limit: 1E-3, Upper limit: 2E3

Send: 01 10 3110 0004 08 3A83126F 3B03126F 6384

Response: 01 10 3110 0004 CEF3

- Read

Send: 01 03 3110 0004 4B30

Response: 01 03 08 3A83126F 3B03126F C2A7

4.5 File Operation

Since the instrument settings are stored in a file, if the **[Auto Save]** in the <File> page is not turned on, the data cannot be stored in real time in the internal FlashRom after all Modbus commands have been set. It will cause the register data being restored to the original file values before the next power-up.

The user can use file operation register to save all settings to the current file or the specified file. At the same time, the specified file data can also load to the register.

> Turn on the [Auto save] in the <File> page, the parameter will automatically save after set each time. The file command can be disregarded.

4.5.1 Save to the Current File [4000]

Send 0001 to the register 4000, the instrument will execute file write, all settings will save to the current file.

This register cannot read.

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	40	00	00	01	02	00	01	26	54
	Write	Register		Quantity of register		Byte	Data		CRC	

Response

1	2	3	4	5	6	7	8
01	10	40	00	00	01	14	09
		Register		Quantity of register		CRC	

Data

Data	Function	Description
0001	Allow to operate	Fixed value

4.5.2 Reload the Current File [4001]

Send the fixed value 0001to the register 4001, the instrument will load the current file data to the system.

This register cannot read.

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	40	01	00	01	02	00	01	27	85
	Write	Register		Quantity of register		Byte	Data		CRC	

Data

Data	Function	Description
0001	Fixed value	

4.5.3 Save to the Specified File [4002]

Send the file number to the register 4002, the instrument will execute file write operation, all setting will save to the specified file. At the same time, the specified file will used as the current system file.

This register cannot read.

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	40	02	00	01	02	00	01	27	85
	Write	Register		Quantity of register		Byte	Data		CRC	

Data

Data	Function	Description
0000~0009	File 0~9	

4.5.4 Load the Specified File [4003]

Send the file number to the register 4003, the instrument will load the specified file to the system.

At the same time, the specified file will used as the current system file.

This register cannot read.

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	40	03	00	01	02	00	01	26	67
	Write	Register		Quantity of register		Byte	Data		CRC	

Data

Data	Function	Description
0000~0009	File 0~9	

4.6 System Function

4.6.1 Zero Clearing [5000]

The instrument will start to execute short-circuit zero clearing when read the register 5000.

Before zero clearing, the test wire should be short-circuit, otherwise, zero clearing will be failed.

The process of zero clearing takes few seconds.

During the execution of zero clearing or after zero clearing is completed, the state of zero clearing will be returned.

0000: Zero clearing is success.

FFFF: Zero clearing is failed.

- Read

During the execution of zero clearing, determine whether zero clearing is completed by read register data.

Send: 01 03 5000 0001 950A

Response: 01 03 02 0000 ####

4.6.2 Key lock [5001]

Write key lock command 5001, Data 0000:

01 10 50 01 00 01 02 00 00 F7 84

4.6.3 Trigger [5002]

Write trigger command 5002, data 0001:

01 10 50 02 00 01 02 00 01 36 77

This command returns an error code only under internal trigger.