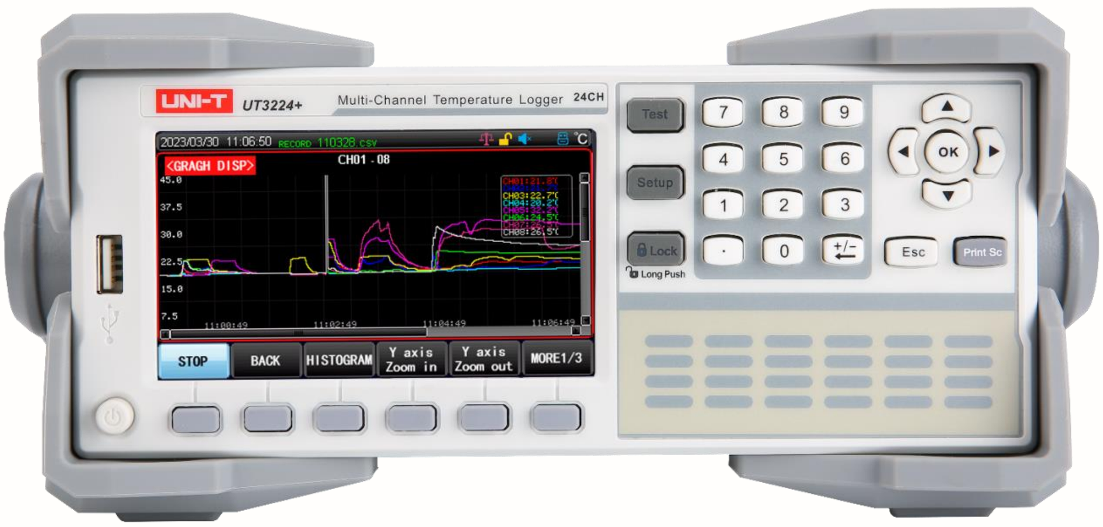
UNI-T®

Instruments.uni-trend.com



Programming Manual

UT3200+ Series Multi-channel Temperature Tester

Warranty and Statement

Copyright

Brand Information

UNI-T is the registered trademark of Uni-Trend Technology (China) Co., Ltd.

Statement

- UNI-T products are protected by patents (including obtained and pending) in China and other countries and regions.
- UNI-T reserves the right to change specifications and prices.
- The information provided in this manual supersedes all previous publications.
- The information provided in this manual is subject to change without notice.
- UNI-T shall not be liable for any errors that may be contained in this manual. For any incidental or consequential damages arising out of the use or the information and deductive functions provided in this manual.
- No part of this manual shall be photocopied, reproduced or adapted without the prior written permission of UNI-T.

Product Certification

UNI-T has certified that the product conforms to China national product standard and industry product standard as well as ISO9001:2008 standard and ISO14001:2004 standard. UNI-T will go further to certificate product to meet the standard of other member of the international standards organization.

SCPI

SCPI（Standard Commands for Programmable Instruments） is a standardized instrument programming language that builds on existing standards IEEE 488.1 and IEEE 488.2 and follows the floating point rules of IEEE 754 standard, ISO 646 message exchange 7-bit encoding notation (equivalent to ASCII programming) and many other standards.

This section introduces the format, symbols, parameters, and abbreviations of the SCPI command.

Command String Parse

The host computer can send a string of commands to the instrument and the command parser of the instrument starts to parsing after catching the terminator (\n) or an input 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.

Command Parse Rule

Command parser only parses and responds to ASCII data.

The command parser starts to command parsing when receive the end mark. The instrument only accept three contents as the following as the end mark.

CR

CR+LF

LF

The command parser will terminate the parsing immediately after parsing an error, and the current command will be invalidated.

The command parser is case-insensitive for parsing command strings.

he command parser supports abbreviated form of command and the detailed see the following section.

In RS485 mode, add ADDR☐Local address::☐ in front of SCPI, the local address can set to 1-32.

It's convenient to communicate with multiple devices via SCPI protocol.

For example: ADDR☐1::☐IDN? ☐ represents a blank

The end of data sent by the instrument defaults to 0x0A (LF).

Multiple instruction can be send via semicolon " ; ".

Symbol Stipulation and Definition

This chapter uses some symbols that are not part of the command tree, but only for a better understanding of the command string.

Mark	Description
<……>	The text in angle brackets indicates the parameter of the command. For example: <float> represents floating point number <integer> represents integer parameter
[……]	The text in square brackets indicates the optional command.
{……}	When the curly brackets contain several parameter items, it means that only one item can be selected from them.
Capital letter	Abbreviated form of the command.
□	Blank mark, it represents a blank and only for reading.

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. SCPI uses a colon (:) to separate high level commands from low level commands.

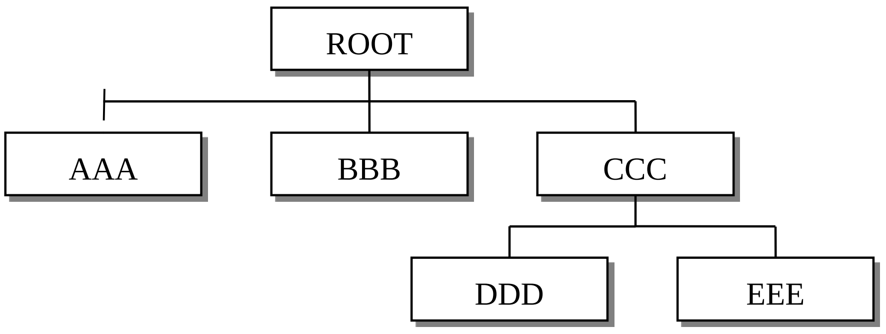


Figure 1-1 Command Tree Structure

For Example

ROOT:CCC:DDD ppp

ROOT	Subsystem command
CCC	Second level
DDD	Third level
ppp	Parameter

Command and Parameter

A command tree is consist of command and [parameter], use a blank to separate (ASCII: 20H).

For example AAA:BBB 1.234
Command [parameter]

Command

Command words can be in long command format or in abbreviated form. Long format facilitates engineers to better understand the meaning of the command string; abbreviated form is suitable for writing.

Parameter

Single character command, no parameter
For Example AAA:BBB

Parameter can be string format and its abbreviated form is also follow the last section " command abbreviated rule"

For example AAA:BBB□1.23

Parameter can be numerical value format.

<integer>	123, +123, -123
<float>	Floating point number of arbitrary form: <fixfloat>: fixed floating point number: 1.23, -1.23 <Sciloat>: floating point number represented by scientific notation: 1.23E+4, +1.23e-4 <mpfloat>: floating point number represented by multiplying power: 1.23k, 1.23M, 1.23G, 1.23u

Table 0-1 Abbreviation of Multiplying Power

Numerical 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

SCPI is case-insensitive, so the written is different from standard name.
For example :

"1M" represents 1 milli, not 1 mega.

"1MA" represents 1 mega.

Separator

The instrument command parser can only receive allowable separator. Other separator will cause error "Invalid separator".

;	**Semicolon** is for separating two commands. For Example AAA:BBB 100.0 ; CCC:DDD
:	**Colon** is for separating command tree or restart the command tree. For Example AAA : BBB : CCC 123.4; : DDD : EEE 567.8
?	**Question mark** is for querying. For Example AAA ?
□	**Blank** is for separating the parameter. For Example AAA:BBB□1.234

Command Reference

All commands are explained by the subsystem command order.

MEAS	Measurement subsystem
SYST	System subsystem
FETCH	Fetch data subsystem
ERROR	ERROR subsystem
IDN?	Query subsystem

MEAS Subsystem

MEAS subsystem is used to switch to different display page.

MEAS	:MODEL	{tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b}
	:RATE	{fast,slow}
	:START	{on,off}
	:CMODEL	<para>,<level>
	:CHANON	<para>,<on,off>
	:LOW	<level>
	:CLOW	<para>,<level>
	:HIGH	<level>
	:CHIGH	<para>,<level>
	:SENSOR	{tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b}

MEAS:MODEL

MEAS:MODEL is used to set sensor type.

Command Syntax	MEAS:MODEL<tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b>
Example	SEND>MEAS:MODEL tc-k <NL> // Set the sensor type to Type K thermocouple.
Query Syntax	MEAS:MODEL?
Query Return	<tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b>
Example	SEND> MEAS:MODEL?<NL> RET> tc-t <NL>

MEAS:RATE

MEAS:RATE is used to set sampling rate.

Command Syntax	MEAS:RATE<fast,slow>
Example	SEND>MEAS:RATE fast<NL> // Set sampling rate to fast.
Query Syntax	MEAS:RATE?
Query Return	<fast,slow>
Example	SEND> MEAS:RATE?<NL> RET> fast <NL>

MEAS:START

MEAS:START is used to enable the sampling.

Command Syntax	MEAS:START <on,off>
Example	SEND>MEAS:START off<NL> // Stop sampling.
Query Syntax	MEAS:START?
Query Return	<on,off>
Example	SEND> MEAS:START?<NL> RET> on <NL>

MEAS:CMODEL

MEAS:CMODEL is used to set the sensor type of each channel.

Command Syntax	MEAS:MODEL <para>,<tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b>
For Example	SEND>MEAS:CMODEL 1,TC-T<NL> // Set the sensor of CH001 to Type T.
Query Syntax	MEAS:CMODEL? // Acquire the sensor type of all channels. MEAS:CMODEL?<int> // Acquire the sensor type of single channel, the minimum of channel number is 1.
Query Return	< tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b >
For Example	SEND> MEAS:CMODEL?<NL> RET> < tc-t,tc-k,tc-j,tc-n,tc-e,tc-s,tc-r,tc-b ><NL> SEND> MEAS:CMODEL? 1<NL> // Acquire the sensor type of CH001. RET> < tc-t ><NL>

MEAS:LOW

MEAS:LOW is used to set the lower limit of all channels.

Command Syntax	MEAS:LOW <float>
For Example	SEND>MEAS:LOW -200.0<NL> // Set the lower limit of all channels to -200.0.
Query Syntax	MEAS:LOW?
Query Return	<float,float> <NL>
For Example	SEND> MEAS:LOW? <NL> RET> <-2.00000e+02,-2.00000e+02> <NL>

MEAS:CLOW

MEAS:CLOW is used to set the lower limit of each channel.

Command Syntax	MEAS:CLOW <para>,<float>
For Example	SEND>MEAS:CLOW 1,-200.0<NL> // Set the lower limit of CH001 to -200.0.

MEAS:HIGH

MEAS:HIGH is used to set the upper limit of all channels.

Command Syntax	MEAS:HIGH <float>
For Example	SEND>MEAS:HIGH 1800.0<NL> // Set the upper limit of all channels to 1800.0.
Query Syntax	MEAS:HIGH?
Query Return	<float,float> <NL>
For Example	SEND> MEAS:HIGH? <NL> RET> <1.80000e+03, 1.80000e+03> <NL>

MEAS:CHIGH

MEAS:CHIGH is used to set the upper limit of each channel.

Command Syntax	MEAS:CHIGH <para>,<float>
For Example	SEND>MEAS:CHIGH 1,1800.0<NL> // Set the upper limit of CH001 to 1800.0.
Query Syntax	MEAS:CHIGH? 1
Query Response	<float> <NL>
Example	SEND> MEAS:CHIGH? 1<NL> RET> <1.80000e+03> <NL>

MEAS:SENSOR

MEAS:SENSOR is used to acquire sensor type of each channel.

Command Syntax	MEAS:SENSOR
Query Response	<TC-T,TC-K,TC-J,TC-N,TC-E,TC-S,TC-R,TC-B> <NL>
Example	SEND> MEAS:SENSOR <NL> RET> <TC-T,TC-K,TC-J,TC-N,TC-E,TC-S,TC-R,TC-B> <NL>

SYST Subsystem

SYST subsystem is used to set SETUP page.

SYST	:COMP	{on,off}
	:BEEP	{on,off}
	:KEYTONE	{on,off}
	:UNIT	{cel,kel,fah}

SYST:COMP

SYST:COMP is used to set the comparator state.

Command Syntax	SYST:COMP <on,off>
For Example	SEND>SYST:COMP on<NL> // Turn on the comparator.
Query Syntax	SYST:COMP?
Query Return	<on,off> <NL>
For Example	SEND> SYST:COMP? <NL> RET> <on> <NL>

SYST:BEEP

SYST:BEEP is used to set the comparator beep state.

Command Syntax	SYST:BEEP <on,off>
For Example	SEND>SYST:BEEP on<NL> // Turn on comparator beep.
Query Syntax	SYST:BEEP?
Query Return	<on,off> <NL>
For Example	SEND>SYST:BEEP? <NL> RET> <on> <NL>

SYST:KEYTONE

SYST:BEEP is used to set the state of key beep.

Command Syntax	SYST:KEYTONE <on,off>
For Example	SEND>SYST:KEYTONE on<NL> // Turn on key beep.
Query Syntax	SYST:KEYTONE?
Query Return	<on,off> <NL>
For Example	SEND>SYST:KEYTONE? <NL> RET> <on> <NL>

SYST:SYSINIT

Command Syntax	SYST:SYSINIT
Example	SEND> SYST:SYSINIT // Return to factory set.

SYST:UNIT

SYST:UNIT is used to set the temperature unit.

Command Syntax	SYST:UNIT <cel,kel,fah>
Parameter	<cel,kel,fah> cel: degree Celsius kel: Kelvin degree fah: Fahrenheit degree
For Example	SEND>SYST:UNIT cel<NL> // Set the temperature unit to degree Celsius.
Query Syntax	SYST:UNIT?
Query Return	<cel,kel,fah> <NL>
For Example	SEND> SYST:UNIT? <NL> RET> <cel> <NL>

FETCH Subsystem

FETCH subsystem is used to acquire the temperature data.

FETCH?	

FETCH?

FETCH? is used to fetch temperature data.

Query Syntax	FETCH?
Query Return	<float, float , float> <NL>
For Example	SEND>FETCH? <NL>
	RET> <+1.00000e-05, +1.00000e-05, +1.00000e-05> <NL>

ERROR Subsystem

ERROR subsystem is used to return error message.

Query Syntax	ERROR?
Query Return	Error string
For Example	SEND> ERR? <NL>
	RET>no error <NL>

*IDN? Subsystem

IDN? is used to query instrument ID.

Query Syntax	IDN?OR *IDN?
Query Return	<MODEL>,<Revision>,<SN>,<Manufacturer>

Modbus

Register Overview

All register addresses used by the instrument are listed below.

Notes:

1. Unless otherwise specified, the numeric value of instruction and response frame are hexadecimal.
2. The register only contains the instruction of acquiring the test result and starting/stopping the test. If user want to customize other instructions, please contact UNI-T sake department.
3. Floating point number online conversion can refer to website http://www.binaryconvert.com/convert_float.html

Register Address	Name	Numeric value	Description
0200	Start/Stop test	1 byte integer	Wirte-only register, data takes 1 register
0202~0261	Temperature value of channel 1~48	4 bytes floating point number	Read-only register, data of each channel takes 2 registers.

Start/Stop Test

Write

1	2	3	4	5	6	7	8	9	10	11
01	10	02	00	00	01	02	00	01	44	50
Station number	Write	Register		Register quantity		Byte	Data		CRC16	

0000：Stop

0001：Start

Written return

1	2	3	4	5	6	7	8
01	10	02	00	00	01	00	71
Slave station	Write	Register		Register quantity		CRC16	

Acquire Test Result

Register 0202~0261 is used to acquire the test result of all channels.

For example: acquire the test result of CH1

Send

1	2	3	4	5	6	7	8
01	03	02	02	00	02	64	73
Slave station	Read	Register		Register quantity		CRC-16	

Response

1	2	3	4	5	6	7	8	9
01	03	04	41	DC	44	5A	9C	CE
01	03	Byte	Float-point number with single precision				CRC-16	

B4~B7 is float-point number with single precision, byte order AA BB CC DD

Test data: 41 DC 44 5A converts to float-point number: 0x41DC445A = 27.5334; (If the channel is open circuit, then the test result is 100000.)