SenseFuture

Communication Protocol

Striving for the Bright Future of Precision Optoelectronic Measurement.

SenseFuture Technologies Co., Ltd.

1. Factory Default Communication Parameters

Parameters	value
Baud rate (default)	38400(TTL Interface) / 9600(RS485 Interface)
Data Bits	8
Stop bit	1
Parity bits	no
Address (default)	1

The host computer software provided with the current temperature controller only supports serial communication at a baud rate of 38400. To establish communication via RS485, the RS485 baud rate must be set to 38400.

2. Communication Frame Format

Notice：Communication must be carried out in strict accordance with the instruction format, otherwise the device cannot respond.

2.1 ASCII Format

2.1.1 Temperature Controller General Parameter Command

Example：Query/Set the PWM Output Frequency

Read：	Send:FPWM=?@
	Reply:OKFPWM=2@\r\n
Write：	Send:FPWM=2@
	Reply:OKFPWM=2@\r\n

2.1.2 Temperature Controller Channel Control Command

Example：Query/Set Target Temperature Command (ASCII Code Communication (Channel 1).Channel 2 can change TC1 to TC2)

Read：	Send:TC1:TG=?@
	Reply:OKTC1:TG=2500000@\r\n
Write：	Send:TC1:TG=2500000@
	Reply:OKTC1:TG=2500000@\r\n

2.2 Modbus-RTU Format

1. This product supports 03H,10H function code format.

2. Channel Control Command：This protocol takes Channel 1 as an example, its address range is 0x1000H~0x1999H, Channel 2 on the basis of Channel 1 address offset 0x1000H, and so on.

Function Code (HEX)	Instructions
0x03H	Read single or multiple registers
0x10H	Write single or multiple registers

●Read：

Host Command Format **(HEX)**：

Station Number	Function Code	Register Address		Number of Registers		CRC	
		High Bytes	Low Bytes	High Bytes	Low Bytes	High Bytes	Low Bytes
01	03	10	00	00	02	C0	CB

Device Acknowledgment **(HEX)**：

Station Number	Function Code	Byte Count	Register Data				CRC	
			Data 1	Data 2	Data 3	Data 4	High Bytes	Low Bytes
01	03	04	00	26	25	A0	01	10

●Write：

Host Command Format **(HEX)**：

Station Number	Function Code	Register Address		Number of Registers		Byte Count	Register Data				CRC	
		High Bytes	Low Bytes	High Bytes	Low Bytes		Data 1	Data 2	Data 3	Data 4	High Bytes	Low Bytes
01	10	10	00	00	02	04	00	26	25	A0	C5	4C

Device Acknowledgment **(HEX)**：

Station Number	Function Code	Register Address		Number of Registers		CRC	
		High Bytes	Low Bytes	High Bytes	Low Bytes	High Bytes	Low Bytes
01	10	10	00	00	02	45	08

●Example：Query/Set Target Temperature Command

Write：	Send(HEX)：01 10 10 00 00 02 04 00 26 25 A0 C5 4C
	Reply(HEX)：01 10 10 00 00 02 45 08
Note：	Send:**01** represents the device address/station number, **10** is the write command, **10 00** is the address of the function register, **00 02**(the number of registers), **04** indicates 4 bytes, **00 26 25 A0** represents 2,500000 (from the high bit to the low bit), and **C5 4C** is the CRC check code at 25℃ (the lower eight bits are sent first).
	Reply:**01** is the device address/station number, **10** is the write command confirmation, **10 00** register write address confirmation, **00 02** register write quantity confirmation, **45 08** is the CRC check code of this line

	command (the lower eight bits are sent first).

3. Communication Command

3.1 Target Temperature Configuration

3.1.1 Query/Set Target Temperature Command

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	TG	0x1000H	int 32	2	Read / Write	-40000000~100000000
Description: 2500000 means 25.00000°C.						

3.2 Temperature Controller Parameter Configuration

Temperature Calculation Model and Polynomial Correction.For details,see Appendix 1.

3.2.1 Query/Set Actual Temperature Command

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	TCADJTEMP	0x1002H	int 32	2	Read / Write	-40000000~100000000
Description: 2500000 means 25.00000°C.						

3.2.2 Read the Resistance of the Sensor

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	RESISTOR	0x1004H	uint 64	4	Read Only	1~500000000000
Description: 10000000000 indicates that the resistance value of the current channel sensor is 10.000000kΩ.						

3.2.3 Query/Set the Calculation Model

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	POLYOMIAL	0x1300H	uint 16	1	Read / Write	0~3

Description:

0：**B-value** Model (Parameters：NTC R_0 and B-value, polynomial temperature correction function enabled);

1：**PT** Model (Parameters：PT R_0, A, B and C values, polynomial temperature correction function enabled);

2：Steinhart-Hart(**S-H**) Model (Parameters：A_0, A_1, A_2, A_3 and A_4 values, polynomial temperature correction function disabled);

3：MF501 Model (Parameters: MF501 A, B, C, polynomial temperature correction function disabled).

PS：Polynomial temperature correction function：A_0~A_7, $T_c = T_m + A_0 + A_1*T_m + A_2*T_m\text{^}2 + A_3*T_m\text{^}3 + A_4*T_m\text{^}4 + A_5*T_m\text{^}5 + A_6*T_m\text{^}6 + A_7*T_m\text{^}7$. (c= corrected; m = measurement).

3.2.4 Query/Set the NTC B

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	BX	0x1301H	uint 32	2	Read / Write	100000~5000000

Description： 395000 means the B value is 3950.00.The B value depends on the specific thermistor material and manufacturing process. The actual value should be based on the specifications provided by the sensor manufacturer.

3.2.5 Query/Set the NTC R_0(Ω)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	RP	0x1303H	uint 32	2	Read / Write	1~9000000
Description： 10000 means that the standard value of the NTC resistance at 25℃ is 10.000kΩ.						

3.2.6 Query/Set the NTC Internal Resistor(Ω)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	NTCRP	0x1305H	uint 64	4	Read / Write	1~11000000000
Description： 10000000000 means that the internal reference resistance of the NTC resistor is 10.000000000kΩ.						

3.2.7 Query/Set the PT(Platinum) R_0(Ω)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PT1000RP	0x1309H	uint 32	2	Read / Write	0~10000000
Description： 1000000 means that the standard value of the PT1000 resistance at 0℃ is 1.000kΩ.						

3.2.8 Query/Set the Callendar-van-Dusen Coefficient A of PT1000 Resistance (PT A)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PTA	0x130BH	int 32	2	Read / Write	-9000000~9000000
Description： 3908300 means that the A value in the PT model is 3.9083E-3.Coefficient A is the linear term coefficient in the Callendar-Van Dusen equation, which describes the fundamental linear change trend of the platinum resistor's resistance with temperature variation. It is the most dominant coefficient determining the resistance-temperature relationship. The default values of the ABC coefficients represent the core parameter set for standard PT sensors and are determined by the inherent properties of the platinum material.The platinum resistance specified in each standard is slightly different. This is the coefficient value in DIN EN 60751.						

3.2.9 Query/Set the PT B

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PTB	0x130DH	int 32	2	Read / Write	-9000000~9000000
Description： -577500 means that the B value in the PT model is -5.775E-7.Coefficient B is the quadratic correction coefficient, used to compensate for the nonlinear deviation of the platinum resistor within the positive temperature range from 0°C to 850°C.						

3.2.10 Query/Set the PT C

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PTC	0x130FH	int 32	2	Read / Write	-90000~90000

Description： -41830 means that the C value in the PT model is -4.183E-12.Coefficient C is the cubic term correction coefficient, applied specifically in the negative temperature range to compensate for the more pronounced nonlinear deviation of the platinum resistor within this interval.

3.2.11 Query/Set the PT Internal Resistor(Ω)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PTRP	0x1311H	uint 64	4	Read / Write	1~2100000000
Description： 2000000000 means that the internal reference resistance of the PT1000 resistor is 2.000000kΩ.						

3.2.12 Query/Set the A Value of the MF501 Resistor(MF501 A)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	MF501A	0x1342H	int64	2	Read / Write	-100000000000000~ 100000000000000
Description： 1000000 represents that the A value is 1.000000, a constant term in the equation, which is the fundamental parameter of the fitting curve and affects the correspondence between resistance and temperature in the low-temperature section. It should be noted that the ABC values of MF501 thermistors produced by different manufacturers may vary due to different model specifications. For specific values, please refer to the data sheet of the product.						

3.2.13 Query/Set the MF501 B

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	MF501B	0x1346H	int64	2	Read / Write	-100000000000000~ 100000000000000
Description： 1000000 represents that the B value is 1.000000. It mainly affects the correspondence between resistance and temperature in the medium-temperature range and is also the basic source of the common "B value".						

3.2.14 Query/Set the MF501 C

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	MF501C	0x134AH	int64	2	Read / Write	-100000000000000~ 100000000000000
Description： 1000000 represents that the C value is 1.000000. The high-order correction term coefficient is mainly used to correct the fitting accuracy of the high-temperature section (higher temperature), making the resistance calculation at high temperatures more accurate (in low-precision scenarios, the C value may be ignored, and only A and B are used).						

3.2.15 Query/Set the A_0~A_7 Base

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	POLA0	0x1315H	int 64	4	Read / Write	-99999999999999~ 99999999999999
Description： The floating point part of the A_0 coefficient scientific notation：99999999999999 stands for 9.99999999999 (The same after A_1~A_7); ASCII Command of A_n：**POLAn** (n is 0~7); Modbus A_0~A_7 Address：0x1315H, 0x131AH, 0x131FH, 0x1324H, 0x1329H, 0x132EH, 0x1333H, 0x1338H.						

3.2.16 Query/Set the A_0~A_7 Exponent

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	POLEA0	0x1319H	int 16	1	Read / Write	-100~100
Description: A_0 coefficient the index part of scientific notation : 100 . At this point, A_0=9.9999999999999E+100 (The same after A_1~A_7); ASCII Command of A_n: **POLEAn** (n is 0~7); Modbus A_0~A_7 Address : 0x1319H, 0x131EH, 0x1323H, 0x1328H, 0x132DH, 0x1332H, 0x1337H, 0x133CH.						

3.2.17 Query/Set the Max Temperature Thresholds(℃)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	OVERTEMPUP	0x133DH	int 32	2	Read / Write	-300000000~500000000
Description: When the measured temperature is higher than the set maximum temperature, the output will stop/continue according to the setting of "3.3.12 Query/Set the Over_Temp Action". 500000000 represents 5000°C.						

3.2.18 Query/Set the Min Temperature Thresholds(℃)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	OVERTEMPLOWER	0x133FH	int 32	2	Read / Write	-300000000~500000000
Description: When the measured temperature is lower than the set minimum temperature, the output will stop/continue according to the setting of "3.3.12 Query/Set the Over_Temp Action". 300000000 represents 3000°C.						

3.2.19 Query/Set Open/Short Circuit Output Protection Function

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	ONSENSOR	0x110CH	int 16	1	Read / Write	0~1
Description: 0: No operation. (Temperature can be given by external communication instruction "3.2.1 Query/Set Actual Temperature Command"); 1: When the sensor is not connected or the sensor is short-circuited, the temperature controller does not output voltage.						

3.2.20 Query/Set Power Output Mode

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	POWERMODE	0x1110H	uint16	1	Read / Write	0~2
Description: 0: Startup follows the previous output status; 1: Power-on output; 2: Power on and turn off.						

3.2.21 Query the Output Current

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	CURRENT	0x1111H	uint16	1	Read Only	0~900000000

Description: 3256 indicates that the current output is 3.256A.

3.2.22 Set the Maximum Output Current

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	SETCURRENT	0x1112H	uint16	1	Read / Write	5~150
Description: 10 stands for 1.0A.						

3.3 Temperature Controller Output Configuration

3.3.1 Query/Set the Max Output(%)

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	LIMITED	0x110EH	int 16	1	Read / Write	0~90
Description: 50 means that the maximum output voltage of the current channel is 50% of the input voltage.						

3.3.2 Query/Set the Output Enable

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	ENABLE	0x1100H	uint 16	1	Read / Write	0~1
Description: 0：Output voltage is disabled; 1：Output voltage is enabled.						

3.3.3 Query/Set the Power-On Delay Output

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	STARTUPDELAY	0x110FH	uint 16	1	Read / Write	3~180
Description: x：Start after x seconds of startup delay (Only when it is output state before power off, the next startup will take effect delayed startup).						

3.3.4 Query/Set the Output Mode

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	MODE	0x1101H	uint 16	1	Read / Write	0~3
Description: 0：Cool&Heat; 1：Cool Only; 2：Heat Only; 3：Output voltage percentage set by communication.						

3.3.5 Query/Set the Output Polarity

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PIDPOL	0x1102H	uint 16	1	Read / Write	0~1
Description: 0：Positive polarity output; 1：Negative polarity output.						

3.3.6 Query/Set the Output Voltage Percentage

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	PWMDUTY	0x1103H	int 64	4	Read / Write	-2000000~2000000
Description: 200000 means that the current output voltage percentage is 10%.						

3.3.7 Query/Set the Temperature Ramp Rate

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	SPEED	0x1108H	uint 16	1	Read / Write	0~10000
Description: Temperature ramp rate, unit °C/s. 0 means no temperature change slope, 1000 means 1°C/s. Configuration Range: 0~10 °C/s. Adjustment Resolution: 0.001 °C/s.						

3.3.8 Query/Set the Forward Threshold Voltage

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	FDEADV	0x110AH	uint 16	1	Read / Write	0~400
Description: 200 represents 1% of the forward output initial voltage percentage.						

3.3.9 Query/Set the Reverse Threshold Voltage

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel Control Command	BDEADV	0x110BH	uint 16	1	Read / Write	0~400
Description: 200 represents -1% of the reverse output initial voltage percentage.						

3.3.10 Query/Set the PWM Output Frequency

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General Parameter Command	FPWM	0x000DH	uint 16	1	Read / Write	0~3
Description: 0：PWM output frequency is 0.5HZ; 1：PWM output frequency is 1HZ; 2：PWM output frequency is 10HZ; 3：PWM output frequency is 100HZ.						

3.3.11 Query/Set the Over_Temp Action

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General Parameter Command	OVERTTEMP	0x000BH	uint 16	1	Read / Write	0~1
Description: 0：When the sensor temperature is higher than the high threshold or lower than the low threshold, continue output. 1：When the sensor temperature is higher than the high threshold or lower than the low threshold, stop output.						

3.3.12 Query/Set Temperature Controller Mode Selection

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range

General Parameter Command	CONTMODE	0x0004H	int 16	1	Read / Write	0~3

Description:

0：Independent control for each channel (default).

1：Channel 1($_{\text{Actual target T}}$)= Channel 2($_{\text{measured T}}$)+ Channel 1($_{\text{set target T}}$).

2：The Voltage /PWM output of Channel 2 follows the Voltage /PWM output of Channel 1.

3：Channel 1($_{\text{Actual target T}}$) = Channel 2 ($_{\text{measured T}}$)+ Channel 1($_{\text{set target T}}$) , Voltage /PWM output of Channel 2 follows Voltage /PWM output of Channel 1.

Notice：When using CONTMODE=2 or 3 modes, any resistance temperature sensor should be connected to the sensor interface of Channel 2.

3.4 PID Configuration

3.4.1 Query/Set the P in PID

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel control Command	KP	0x1200H	uint 32	2	Read / Write	0~9000000

Description: The coefficient of P in the PID formula.

3.4.2 Query/Set the I in PID

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel control Command	KI	0x1202H	uint 32	2	Read / Write	0~9000000

Description: The coefficient of I in the PID formula.

3.4.3 Query/Set the D in PID

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel control Command	KD	0x1204H	uint 32	2	Read / Write	0~9000000

Description: The coefficient of D in the PID formula.

3.4.4 Query/Set PID Self-tuning

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
Channel control Command	AUTOPID	0x1107H	uint 16	1	Read / Write	0~2

Description:

0：non-PID self-tuning and non-PID real-time automatic optimization;

1：PID self-tuning;

2：PID real-time automatic optimization (Reserved Instruction).

3.5 Temperature Controller Basic Configuration

3.5.1 Query the Current Temperature Controller Model

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range

General parameter Command	TEC	0x0001H	uint 16	1	Read Only	0~255
Description: 1：103; 2：207L; 3：207; 4：215L; 5：215; 6：215Pro; 7：107L; 8：107; 9：115L; 10：115L; 11：115Pro; 12：100L; 13：100; 14：100Pro; 15：403L; 16：403; 17：403Pro; 18：415L; 19：415; 20：603L; 21：603; 22：615L; 23：615; 24：615Pro; 25：803L; 26：803; 27：815L; 28：815; 29：815Pro; 30：203L; 31：203.						

3.5.2 Query the Firmware Version Number

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	FPV	0x000CH	uint 16	1	Read Only	100~9999
Description: 100 represents the version number 1.0.0.						

3.5.3 Query/Set the Temperature Controller 485 Address

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	ADDRESS	0x0002H	uint 16	1	Read / Write	0~255
Description: Temperature controller device address.						

3.5.4 Query/Set the UART TTL Baud Rate

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	BOUNDTABLEONE	0x0008H	uint 16	1	Read / Write	0~7
Description: 0：4800; 1：9600; 2：19200; 3：38400; 4：57600; 5：115200; 6：23400; 7：460800.						

3.5.5 Query/Set the RS485 Baud Rate

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	BOUNDTABLETWO	0x0009H	uint 16	1	Read / Write	0~7
Description: 0：4800; 1：9600; 2：19200; 3：38400; 4：57600; 5：115200; 6：23400; 7：460800. To communicate with the dedicated software the temperature controller, set the baud rate to 38400.						

3.5.6 Query the Temperature Controller's Internal Temperature

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	SINTERIORTEMP	0x0003H	int 16	1	Read Only	-20~120
Description: 20 indicates that the temperature of the temperature controller itself is 20°C.						

3.5.7 Query/Set the Temperature Controller Over-temperature Threshold

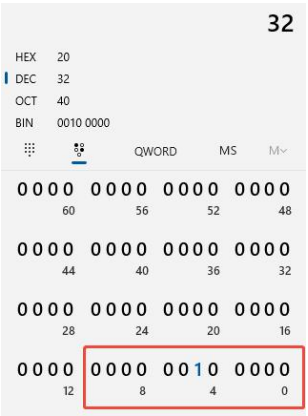
Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	OVERTVPT	0x000AH	uint 16	1	Read / Write	40~100
Description : 70 indicates that the temperature controller begins to gradually reduce the output power after its own temperature reaches 70.						

3.5.8 Query the Error Code

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	ERRORCODE	0x0007H	uint 16	1	Read Only	Defined by Bit

Description: The temperature controller reflects the operating status of the system through 16-bit binary status words (decimal values), and each bit is defined as follows:

Status	Bit Number	Trigger Condition
The temperature controller system is in an incorrect position	Bit0	High-temperature alarm (When the temperature exceeds the over-temperature threshold of the temperature controller itself, output is restricted)
	Bit1	Over-temperature alarm (Output stops when the temperature exceeds the maximum temperature set inside the thermostat)
	Bit2	Under-voltage alarm (triggered when voltage drops below 7V)
	Bit3	Over-voltage alarm (triggered when voltage exceeds 30V)
Channel 1 Error	Bit4	/
	Bit5	The temperature of the sensor in Channel 1 is greater than the high threshold or lower than the low threshold
	Bit6	The output current of Channel 1 has reached the set maximum current (under current limiting)
	Bit7	/
Channel 2 Error	Bit8	/
	Bit9	The temperature of the sensor in Channel 2 is greater than the high threshold or lower than the low threshold
	Bit10	The output current of Channel 2 has reached the set maximum current (under current limiting)
	Bit11	/



When the value is 32, Bit5 is 1, and the error is that the temperature of the sensor in Channel 1 is greater than the high threshold or lower than the low threshold.

3.5.9 Restore Factory Setting

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	RESET	0x0000H	uint 16	1	Write Only	1

Description: Restore factory Settings.

3.6 Other Data Query

3.6.1 Query: Comprehensive Data Retrieval

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	INQUIRE	/	/	/	/	1

Description:
Send: INQUIRE=1@
Return:
OKTC1:TG=2500000@TC2:TG=2500000@OKTC1:LIMITED=30@TC2:LIMITED=30@OKTC1:MODE=0@TC2:MODE=0@OKTC1:ENABLE=0@TC2:ENABLE=0@OKTC1:KP=3000@TC2:KP=3000@OKTC1:KI=150@TC2:KI=150@OKTC1:KD=0@TC2:KD=0@OKTC1:RP=10000@TC2:RP=10000@OKTC1:BX=395000@TC2:BX=395000@OKTEC=215OKTC1:PT1000RP=1000000@TC2:PT1000RP=1000000@OKTC1:CHRATIO=100@TC2:CHRATIO=100@OKTC1:SPEED=0@TC2:SPEED=0@OKTC1:STEADYIOB=0@TC2:STEADYIOB=0@OKTC1:OVERTEMPUP=500000000@TC2:OVERTEMPUP=500000000@OKTC1:OVERTEMPLOWER=-300000000@TC2:OVERTEMPLOWER=-300000000@OKTC1:FDEADV=0@TC2:FDEADV=0@OKTC1:BDEADV=0@TC2:BDEADV=0@OKTC1:NTCRP=10000000000@TC2:NTCRP=10000000000@OKTC1:PTRP=1000000000@TC2:PTRP=1000000000@OKTC1:PTA=3908300@TC2:PTA=3908300@OKTC1:PTB=-5775000@TC2:PTB=-5775000@OKTC1:PTC=-4183000@TC2:PTC=-4183000@OKTC1:PIDPOL=0@TC2:PIDPOL=0@

Represents the current control temperature of Channel 1 is 25°C, the control temperature of Channel 2 is 25°C. The maximum output of Channel 1 is 30%, the maximum output of Channel 2 is 30%. The output mode of Channel 1 is two-way mode, the output mode of Channel 2 is two-way mode. The output of Channel 1 can be opened, the output of Channel 2 can be opened. The P of Channel 1 PID is 3000, the P of Channel 2 PID is 3000. The I of Channel 1 PID is 150, the I of Channel 2 PID is 150. The D of Channel 1 PID is 0, the D of Channel 2 PID is 0. The standard resistance of Channel 1 NTC resistance 10KΩ, the standard resistance of Channel 2 NTC resistance 10KΩ. The B value of Channel 1 NTC resistance is 3950, the B value of Channel 2 NTC resistance is 3950.The current version number of the temperature controller is 215. The standard resistance of Channel 1 PT1000 is 1kΩ, the standard resistance of Channel 2 PT1000 is 1kΩ. The cooling/heating coefficient of Channel 1 is 1.00, Channel 2 coefficient/heating coefficient is 1.00. Channel 1 of temperature change slope change is not limited, Channel 2 of temperature change slope change is not limited. Reserved function code: STEADYIOB (STATE Pin Trigger Condition),When Channel 1 stabilizes near its target temperature, the STATE pin outputs a high level, when Channel 2 stabilizes near its target temperature, the STATE pin outputs a high level. When the temperature of Channel 1 is greater than 5000°C, the controller will turn off the output; when the temperature of Channel 2 is greater than 5000°C, the controller will turn off the output; when the temperature of Channel 1 is less than -3000°C, the controller will turn off the output; when the temperature of Channel 2 is less than -3000°C, the controller will turn off the output. Channel 1 forward dead-zone voltage is 0,Channel 2 forward dead-zone voltage is 0; Channel 1 reverse dead-zone voltage is 0,Channel 2 reverse dead-zone voltage is 0. The internal reference resistance of Channel 1 NTC resistor is 10KΩ, the internal reference resistance of Channel 2 NTC resistor is 10kΩ,the internal reference resistance of Channel 1 PT1000 resistor is 2KΩ, The internal reference resistance of the Channel 2 PT1000 resistance is 2KΩ. The PT A of the Channel 1 is divided by 10E9, and the PT A of the Channel 2 is divided by 10E9.The PT B of the Channel 1 divided by 10E12,the PT B of the Channel 2 divided by 10E12. PT C of the Channel 1 divided by 10E16, and the PT C of the Channel 2 is divided by 10E16. The polarity output in the Channel 1 is positive, and the polarity output in the Channel 2 is positive.

Note: If it is a two-channel temperature controller, the data of TC2 will be sent directly after sending the data of TC1.

3.6.2 Query: Key Data Retrieval

Command Type	ASCII Command	Register Address	Data Type	Number of Registers	Accessibility	Data Range
General parameter Command	DATADEMAND	/	/	/	/	1~2

Description:
Send: DATADEMAND=1@
Return:
TC1:TCADJTEMP=2259187@TC1:RESISTOR=11139104486@TC1:PWM=0@TC2:TCADJTEMP=999999999@TC2:RESISTOR=0@TC2:PWM=0@SINTERIORTEMP=23@

This indicates that the current temperature of Channel 1 is 22.59187°C, the resistance of Channel 1 is 11139.104486Ω, and

the current output voltage percentage of Channel 1 is 0. Channel 2 shows 999999999 as it has no temperature sensor connected, its resistance is 0, and its current output voltage percentage is 0. The internal temperature of the controller itself is 23°C.

Send: DATADEMAND=2@
Return:
TC1:TCADJTEMP=2518788@TC1:RESISTOR=9916909257@TC1:OUTV=1000000000@TC2:TCADJTEMP=999999999@TC2:RESISTOR=0@TC2:OUTV=0@SINTERIORTEMP=34@

This indicates that the current temperature of Channel 1 is 25.18788°C, the resistance of Channel 1 is 9916.909257Ω, and the current output voltage of Channel 1 is 10V. Channel 2 shows 999999999 as it has no temperature sensor connected, its resistance is 0, and its current output voltage is 0V. The internal temperature of the controller itself is 34°C.

Note: For a dual-channel temperature controller, after sending the data for TC1, the data for TC2 follows immediately.

4. Command Preview Table

Function	Register Name	Communication Protocol	Register Address（0x）	Access ibility	Data Type	Data Range	Default value (Sensor not connected）
Target Temperature Configuration	Query/Set Target Temperature Command	TC1:TG	1000(TC1) 2000(TC2)	Read / Write	int32	-40000000~100000000	2500000
Temperature Controller Parameter Configuration	Query/Set Actual Temperature Command	TC1:TCADJTEMP	1002(TC1) 2002(TC2)	Read / Write	int32	-40000000~100000000	999999999
	Read the Resistance of the Sensor	TC1:RESISTOR	1004(TC1) 2004(TC2)	Only Read	uint64	1~500000000000	0
	Query/Set the Calculation Model	TC1:POLYOMIAL	1300(TC1) 2300(TC2)	Read / Write	uint16	0~3	0
	Query/Set the NTC B	TC1:BX	1301(TC1) 2301(TC2)	Read / Write	uint32	100000~5000000	395000
	Query/Set the NTC R0(Ω)	TC1:RP	1303(TC1) 2303(TC2)	Read / Write	uint32	1~9000000	10000
	Query/Set the NTC Internal Resistor(Ω)	TC1:NTCRP	1305(TC1) 2305(TC2)	Read / Write	uint64	1~110000000000	10000000000
	Query/Set the PT(Platinum) R0(Ω)	TC1:PT1000RP	1309(TC1) 2309(TC2)	Read / Write	uint32	0~10000000	1000000
	Query/Set the Callendar-van-Dusen Coefficient A of PT1000 Resistance (PT A)	TC1:PTA	130B(TC1) 230B(TC2)	Read / Write	int32	-9000000~9000000	3908300
	Query/Set the PT B	TC1:PTB	130D(TC1) 230D(TC2)	Read / Write	int32	-9000000~9000000	-577500
	Query/Set the PT C	TC1:PTC	130F(TC1) 230F(TC2)	Read / Write	int32	-90000~90000	-41830
	Query/Set the PT Internal Resistor(Ω)	TC1:PTRP	1311(TC1) 2311(TC2)	Read / Write	uint64	1~2100000000	1000000000
	Query/Set the A Value of the MF501 Resistor(MF501 A)	TC1:MF501A	1342(TC1) 2342(TC2)	Read / Write	int64	-100000000000000~1000000000000000	/
	Query/Set the MF501 B	TC1:MF501B	1346(TC1) 2346(TC2)	Read / Write	int64	-100000000000000~1000000000000000	/
	Query/Set the MF501 C	TC1:MF501C	134A(TC1) 234A(TC2)	Read / Write	int64	-100000000000000~1000000000000000	/
	Query/Set the A0 Base	TC1:POLA0	1315(TC1) 2315(TC2)	Read / Write	int64	-9999999999999999~9999999999999999	0
	Query/Set the A0 Exponent	TC1:POLEA0	1319(TC1) 2319(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A1 Base	TC1:POLA1	131A(TC1) 231A(TC2)	Read / Write	int64	-9999999999999999~9999999999999999	0
	Query/Set the A1 Exponent	TC1:POLEA1	131E(TC1) 131E(TC2)	Read / Write	int16	-100~100	0

	Query/Set the A2 Base	TC1:POLA2	131F(TC1) 231F(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A2 Exponent	TC1:POLEA2	1323(TC1) 2323(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A3 Base	TC1:POLA3	1324(TC1) 2324(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A3 Exponent	TC1:POLEA3	1328(TC1) 2328(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A4 Base	TC1:POLA4	1329(TC1) 2329(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A4 Exponent	TC1:POLEA4	132D(TC1) 232D(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A5 Base	TC1:POLA5	132E(TC1) 232E(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A5 Exponent	TC1:POLEA5	1332(TC1) 2332(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A6 Base	TC1:POLA6	1333(TC1) 2333(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A6 Exponent	TC1:POLEA6	1337(TC1) 2337(TC2)	Read / Write	int16	-100~100	0
	Query/Set the A7 Base	TC1:POLA7	1338(TC1) 2338(TC2)	Read / Write	int64	-99999999999999~99999999999999	0
	Query/Set the A7 Exponent	TC1:POLEA7	133C(TC1) 233C(TC2)	Read / Write	int16	-100~100	0
	Query/Set the Max Temperature Thresholds(℃)	TC1:OVERTEMPUP	133D(TC1) 233D(TC2)	Read / Write	int32	-300000000~500000000	500000000
	Query/Set the Min Temperature Thresholds(℃)	TC1:OVERTEMPLOWER	133F(TC1) 233F(TC2)	Read / Write	int32	-300000000~500000000	-300000000
	Query/Set Open/Short Circuit Output Protection Function	TC1:ONSENSOR	110C(TC1) 210C(TC2)	Read / Write	int16	0~1	1
	Query/Set Power Output Mode	TC1:POWERMODE	1110(TC1) 2110(TC2)	Read / Write	uint16	0~2	0
	Query the Output Current	TC1:CURRENT	1111(TC1) 2111(TC2)	Read Only	uint16	0~900000000	0
	Set the Maximum Output Current	TC1:SETCURRENT	1112(TC1) 2112(TC2)	Read / Write	uint16	5~150	/
Temperature Controller Output Configuration	Query/set the Max Output(%)	TC1:LIMITED	110E(TC1) 210E(TC2)	Read / Write	int16	0~90	30
	Query/Set the Output Enable	TC1:ENABLE	1100(TC1) 2100(TC2)	Read / Write	uint16	0~1	0
	Query/Set the Power-On Delay Output	TC1:STARTUPDELAY	110F(TC1) 210F(TC2)	Read / Write	uint16	3~180	3

	Query/Set the Output Mode	TC1:MODE	1101(TC1) 2101(TC2)	Read / Write	uint16	0~3	0
	Query/Set the Output Polarity	TC1:PIDPOL	1102(TC1) 2102(TC2)	Read / Write	uint16	0~1	0
	Query/Set the Output Voltage Percentage	TC1:PWMDUTY	1103(TC1) 2103(TC2)	Read / Write	int64	-2000000~2000000	0
	Query/Set the Temperature Ramp Rate	TC1:SPEED	1108(TC1) 2108(TC2)	Read / Write	uint16	0~10000	0
	Query/Set the Forward Starting Voltage	TC1:FDEADV	110A(TC1) 210A(TC2)	Read / Write	uint16	0~400	0
	Query/Set the Reverse Starting Voltage	TC1:BDEADV	110B(TC1) 210B(TC2)	Read / Write	uint16	0~400	0
	Query/Set the PWM Output Frequency	FPWM	000D	Read / Write	uint16	0~3	2
	Query/Set the Over_Temp Action	OVERTTEMP	000B	Read / Write	uint16	0~1	1
	Query/Set Temperature Controller Mode Selection	CONTMODE	0004	Read / Write	int16	0~3	0
PID Configuration	Query/Set the P in PID	TC1:KP	1200(TC1) 2200(TC2)	Read / Write	uint32	0~9000000	3000
	Query/Set the I in PID	TC1:KI	1202(TC1) 2202(TC2)	Read / Write	uint32	0~9000000	150
	Query/Set the D in PID	TC1:KD	1204(TC1) 2204(TC2)	Read / Write	uint32	0~9000000	0
	Query/Set PID Self-tuning	TC1:AUTOPID	1107(TC1) 2107(TC2)	Read / Write	uint16	0~2	0
Temperature Controller Basic Configuration	Query the Current Temperature Controller Model	TEC	0001	Only Read	uint16	0~255	依型号而定
	Query the Firmware Version Number	FPV	000C	Only Read	uint16	100~9999	依版本而定
	Query/Set the Temperature Controller 485 Address	ADDRESS	0002	Read / Write	uint16	0~255	1
	Query/Set the UART TTL Baud Rate	BOUNDTABLEONE	0008	Read / Write	uint16	0~7	3
	Query/Set the RS485 Baud Rate	BOUNDTABLETWO	0009	Read / Write	uint16	0~7	1
	Query the Temperature Controller's Internal Temperature	SINTERIORTEMP	0003	Only Read	int16	-20~120	/
	Query/Set the Temperature Controller Over-temperature Threshold	OVERTVPT	000A	Read / Write	uint16	40~100	70
	Query the Error Code	ERRORCODE	0007	Only Read	uint16	Defined by Bit	0
	Restore Factory Setting	RESET	0000	Only Write	uint16	1	0

Other Data Query	Query: Comprehensive Data Retrieval	INQUIRE	/	/	/	1	/
	Query: Key Data Retrieval	DATADEMAND	/	/	/	1~2	/

Notice：Due to product updates, previous version maybe not support the latest communication protocol, please understand!

Appendix 1：Temperature Calculation Model and Polynomial Correction

01 NTC temperature sensor

Basic equation calculation method (B-Value Model)	
$R= R_0 \times \exp[B \times (1/(T+273.15)-1/298.15)]$	
T	Temperature, in degrees Celsius (°C)
R	The actual resistance value of the sensor, in ohms (Ω)
R_0	Resistance of NTC at 25°C R(25°C), in ohms (Ω)
B	Sensor β value parameters
C language temperature calculation formula： $T= 1/(1/(298.15)+1/B*\ln(R/R_0))-273.15$	

Steinhart-Hart equation calculation method (S-H Model)	
$1/(T+273.15)= A_0+ A_1 \times \ln(R) + A_2 \times [\ln(R)]^2+A_3 \times [\ln(R)]^3+A_4 \times [\ln(R)]^4$	
T	Temperature, in degrees Celsius (°C)
R	The actual resistance value of the sensor, in ohms (Ω)
$A_0,...,A_4$	Sensor coefficient (shared with polynomial temperature correction)

02 PT (platinum) resistance temperature sensor

-200~0°C temperature range (PT Model)	
$R= R_0 \times [1+A \times T+B \times T^2+C \times (T-100)T^3]$	
T	Temperature, in degrees Celsius (°C)
R	The actual resistance value of the sensor, in ohms (Ω)
R_0	Resistance value of PT at 0°C, in ohms (Ω)
A, B, C	Sensor coefficient

0~800°C temperature range (PT Model)	
$R= R_0 \times [1+A \times T+B \times T^2]$	
T	Temperature, in degrees Celsius (°C)

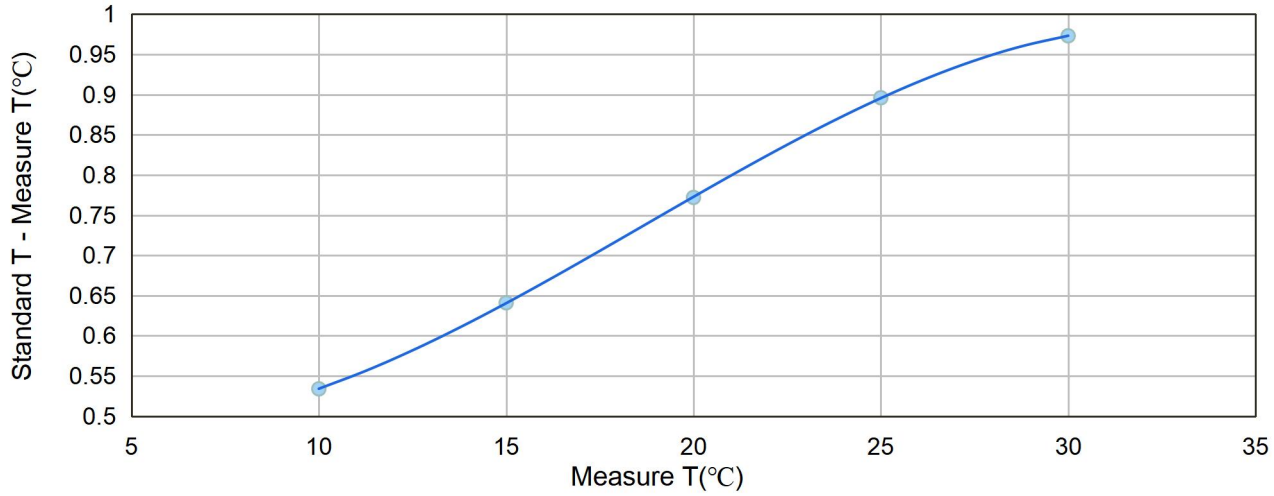
R	The actual resistance value of the sensor, in ohms (Ω)
R_0	Resistance value of PT at 0°C, in ohms (Ω)
A, B	Sensor coefficient

03 Polynomial temperature correction

$T_C=T_B+A_0+A_1\times T_B+A_2\times T_B^2+A_3\times T_B^3+A_4\times T_B^4+A_5\times T_B^5+A_6\times T_B^6+A_7\times T_B^7$	
T_C	Polynomial temperature corrected temperature in degrees Celsius (°C)
T_B	Polynomial temperature the temperature before correction, in degrees Celsius (°C), which can be the temperature calculated by the B-Value or PT model, but not by the S-H model. That is to say, when the B-Value and PT models are selected, the polynomial temperature correction function is always on. But when the S-H model is selected, the polynomial temperature correction function is disabled. (Please note that this function is only available in hardware versions v4.2.2 and above.)
$A_0,\ldots,A_7$	Temperature correction factor

Case description: NTC temperature sensor with temperature controller polynomial temperature correction

An instrument uses an NTC temperature sensor for temperature measurement, setting R0 and B values of 10,000Ω and 3950, respectively. The polynomial correction coefficient A0~A7 is set to 0, and the whole system is calibrated with the standard temperature sensor. Then measure the temperature and standard temperature gauge as shown in the table below：



Measure T(°C) T_B	Standard T(C) T_S	Standard T - measured T(°C) ΔT
10.000	10.534	0.534
15.000	15.641	0.641
20.000	20.772	0.772
25.000	25.896	0.896

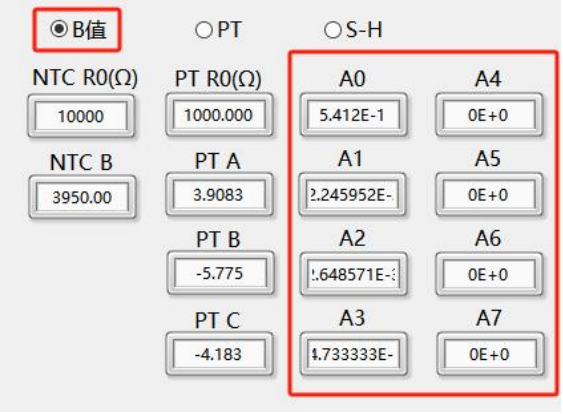
30.000	30.973	0.973

Using the polynomial of Excel to fit the temperature difference (standard T - measured T) vs measured temperature in 3rd degree polynomial (select several polynomials according to the situation), the polynomial correction formula is as follows：

$$TC=TB+(5.412000e-1)+(-2.245952e-2)\times TB+(2.648571e-3)\times TB2+(-4.733333e-5)\times TB3$$

To wit：

Temperature correction factor	value
A_0	5.412000e-1
A_1	-2.245952e-2
A_2	2.648571e-3
A_3	-4.733333e-5
A_4... A_7	0



Write the value of A0-A7 through the computer software, as shown in the figure above, that is, the polynomial temperature correction is complete.

Notice: The temperature sensor of our company is not calibrated by default. If you need calibration, you can contact our official customer service.(Email：sales@sensefuture.com)

Appendix 2：Programming Case

01 C++ Programming Case

```
#include <iostream>
#include <windows.h>
// This case uses C language to communicate with the OptoFuture temperature control module, implementing the setting of target temperature data and acquisition of current temperature data through TTL commands and ModBus commands respectively. For other commands, please refer to the programming manual for independent development.
// If communication fails, please carefully check the serial port number, baud rate, and whether the temperature controller is powered on.
// MODBUS CRC check function
uint16_t crc_update(int data_length, uint8_t* data) {
    uint16_t CRC_Value = 0xFFFF; // CRC initial value
    uint8_t i, j;
    // Calculate the CRC check value
    for (i = 0; i < data_length; i++)
    {
        CRC_Value ^= static_cast<uint8_t>(data[i]);
        for (j = 0; j < 8; j++)
        {
            if (CRC_Value & 0x0001)
```

```
                    CRC_Value = (CRC_Value >> 1) ^ 0xA001;
                else
                    CRC_Value = (CRC_Value >> 1);
            }
        }
        return CRC_Value;
}
int ExampleTest() {
        // Open the serial port handle. Assume the serial port name is COM9
        HANDLE hSerial = CreateFile(L"COM9",
            GENERIC_READ | GENERIC_WRITE,
            0,
            NULL,
            OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL,
            NULL);
        if (hSerial == INVALID_HANDLE_VALUE) {
            std::cout << "Failed to open serial port" << std::endl;
            return 1;
        }
        // Configure serial port parameters
        DCB dcb = { 0 };
        dcb.DCBlength = sizeof(dcb);
        GetCommState(hSerial, &dcb);
        dcb.BaudRate = CBR_38400; // Baud rate 38400
        dcb.ByteSize = 8;             // Data bit 8
        dcb.Parity = NOPARITY;      // No parity
        dcb.StopBits = ONESTOPBIT;// Stop bit 1
        SetCommState(hSerial, &dcb);

        // Set timeout
        COMMTIMEOUTS timeouts = { 0 };
        timeouts.ReadIntervalTimeout = 50;
        timeouts.ReadTotalTimeoutConstant = 50;
        timeouts.ReadTotalTimeoutMultiplier = 10;
        SetCommTimeouts(hSerial, &timeouts);

        // TTL command sets channel 1 target temperature to 25°C

        // Send data
        const char* sendData = "TC1:TG=2500000@\n";
        DWORD bytesWritten;
        if (!WriteFile(hSerial, sendData, strlen(sendData), &bytesWritten, NULL)) {
            std::cerr << "Failed to send data" << std::endl;
            CloseHandle(hSerial);
```

```
        return 1;
    }

    // Receive data
    char receivedData[32] = { 0 };
    DWORD bytesRead;
    if (!ReadFile(hSerial, receivedData, sizeof(receivedData) - 1, &bytesRead, NULL)) {
        std::cerr << "Failed to receive data" << std::endl;
        CloseHandle(hSerial);
        return 1;
    }

    // Check response
    if (strcmp(receivedData, "OKTC1:TG=2500000@\r\n") != 0) {
        std::cout << "Temperature setting failed" << std::endl;
    }
    else {
        std::cout << "Target temperature set to 25°C" << std::endl;
    }

    // TTL command queries current temperature of channel 1
    float myCurrentTemperature = 0.0f;

    // Send data
    Sleep(5); // Wait 5 milliseconds to ensure data is not sent consecutively
    const char* sendData1 = "TC1:TCADJTEMP=?@\n";
    DWORD bytesWritten1;
    if (!WriteFile(hSerial, sendData1, strlen(sendData1), &bytesWritten1, NULL)) {
        std::cerr << "Failed to send data" << std::endl;
        CloseHandle(hSerial);
        return 1;
    }

    // Receive data
    char receivedData1[32] = { 0 };
    DWORD bytesRead1;
    if (!ReadFile(hSerial, receivedData1, sizeof(receivedData1) - 1, &bytesRead1, NULL)) {
        std::cerr << "Failed to receive data" << std::endl;
        CloseHandle(hSerial);
        return 1;
    }

    // Check response: whether the first part is "OKTC1:TCADJTEMP=" and the last part is "@\r\n"
    if (strncmp(receivedData1, "OKTC1:TCADJTEMP=", 16) != 0 || strcmp(receivedData1 + bytesRead1 - 4, "@\r\n") != 0)
{
```

```
        std::cout << "Response data error" << std::endl;
    }
    else {
        // Parse data
        char tempBuffer[16] = { 0 };
        int tempLength = bytesRead1 - 19; // Remove the last three characters
        if (tempLength > 0 && tempLength < sizeof(tempBuffer)) {
            strncpy_s(tempBuffer, sizeof(tempBuffer), receivedData1 + 16, tempLength);
            myCurrentTemperature = (float)(atoi(tempBuffer) / 100000.0f);
            std::cout << "Current temperature: " << myCurrentTemperature << "°C" << std::endl;
        }
        else {
            std::cerr << "Failed to parse temperature data" << std::endl;
        }
    }
    // ModBus command sets channel 1 target temperature to 25°C

    // Set send command
    int8_t sendData2[13];
    int index = 0;
    sendData2[index++] = 0x01; // Assume temperature controller ModBus address is 0x01
    sendData2[index++] = 0x10; // Register write command function code
    sendData2[index++] = 0x10; // Concatenate with next byte
    sendData2[index++] = 0x00; // Concatenate with previous byte: 0x1000 means setting target temperature
    sendData2[index++] = 0x00; // Concatenate with next byte
    sendData2[index++] = 0x02; // Concatenate with previous byte: 0x0002 means sending two registers
    sendData2[index++] = 0x04; // 0x04 means data needs to be sent in four bytes
    sendData2[index++] = 2500000 >> 24; // High byte of 2500000
    sendData2[index++] = 2500000 >> 16; // Second high byte of 2500000
    sendData2[index++] = 2500000 >> 8;    // Second low byte of 2500000
    sendData2[index++] = 2500000;            // Low byte of 2500000

    // CRC checksum
    uint16_t crc = crc_update(index, reinterpret_cast<uint8_t*>(sendData2));
    sendData2[index++] = crc & 0xFF;
    sendData2[index++] = crc >> 8;

    // Send data
    Sleep(5); // Wait 5 milliseconds to ensure data is not sent consecutively
    DWORD bytesWritten2;
    WriteFile(hSerial, sendData2, index, &bytesWritten2, NULL);

    // Receive data
    int8_t receivedData2[32] = { 0 };
    DWORD bytesRead2;
```

```
ReadFile(hSerial, receivedData2, sizeof(receivedData2) - 1, &bytesRead2, NULL);

// Check response
// CRC check
crc = crc_update(bytesRead2 - 2, reinterpret_cast<uint8_t*>(receivedData2));
uint16_t crc_recv = (receivedData2[bytesRead2 - 1] << 8) + receivedData2[bytesRead2 - 2];
if (crc != crc_recv) {
    std::cout << "CRC check failed" << std::endl;
}
else {
    // CRC check succeeded, parse data
    if (receivedData2[0] != 0x01)
    {
        std::cout << "Device error" << std::endl;
    }
    else if (receivedData2[1] != 0x10)
    {
        std::cout << "Not a register write command" << std::endl;
    }
    else if (receivedData2[2] != 0x10 || receivedData2[3] != 0x00)
    {
        std::cout << "Register address error" << std::endl;
    }
    else if ((receivedData2[4] << 8 | receivedData2[5]) != 0x0002)
    {
        std::cout << "Register quantity error" << std::endl;
    }
    else
    {
        std::cout << "Target temperature set to 25°C" << std::endl;
    }

}
// ModBus command reads current temperature of channel 1

// Set send command
int8_t sendData3[8];
index = 0;
sendData3[index++] = 0x01; // Assume temperature controller ModBus address is 0x01
sendData3[index++] = 0x03; // Register read command function code
sendData3[index++] = 0x10; // Concatenate with next byte
sendData3[index++] = 0x02; // Concatenate with previous byte: 0x1002 means current temperature of channel 1
sendData3[index++] = 0x00; // Concatenate with next byte
sendData3[index++] = 0x02; // Concatenate with previous byte: 0x0002 means sending two registers
```

```
// CRC checksum
crc = crc_update(index, reinterpret_cast<uint8_t*>(sendData3));
sendData3[index++] = crc & 0xFF;
sendData3[index++] = crc >> 8;
// Send data
Sleep(5); // Wait 5 milliseconds to ensure data is not sent consecutively
DWORD bytesWritten3;
WriteFile(hSerial, sendData3, index, &bytesWritten3, NULL);

// Receive data
int8_t receivedData3[32] = { 0 };
DWORD bytesRead3;
ReadFile(hSerial, receivedData3, sizeof(receivedData3) - 1, &bytesRead3, NULL);

// Check response
// CRC check
crc = crc_update(bytesRead3 - 2, reinterpret_cast<uint8_t*>(receivedData3));
crc_recv = (static_cast<uint8_t>(receivedData3[bytesRead3 - 2]) |
            (static_cast<uint8_t>(receivedData3[bytesRead3 - 1]) << 8));
if (crc != crc_recv) {
    std::cout << "CRC check failed" << std::endl;
}
else {
    // CRC check succeeded, parse data
    if (receivedData3[0] != 0x01)
    {
        std::cout << "Device error" << std::endl;
    }
    else if (receivedData3[1] != 0x03)
    {
        std::cout << "Not a register read command" << std::endl;
    }

    else if (receivedData3[2] != 0x04)
    {
        std::cout << "Data length error" << std::endl;
    }
    else
    {
        // Parse data
        myCurrentTemperature = ((static_cast<uint8_t>(receivedData3[3]) << 24)
            | (static_cast<uint8_t>(receivedData3[4]) << 16)
            | (static_cast<uint8_t>(receivedData3[5]) << 8)
            | static_cast<uint8_t>(receivedData3[6])) / 100000.0f;
        std::cout << "Current temperature: " << myCurrentTemperature << "°C" << std::endl;
```

```
            }
        }
        if (hSerial != INVALID_HANDLE_VALUE) {
            CloseHandle(hSerial);
        }
        return 0;
}
int main()
{
        ExampleTest();
}
```

02 Python Programming Case

```
import serial
import time
from typing import List

def crc_update(data: bytes) -> int:
        """
        MODBUS CRC-16 check calculation
        :param data: Byte data for which CRC needs to be calculated
        :return: Calculated CRC check value
        """
        crc = 0xFFFF    # CRC initial value
        for byte in data:
                crc ^= byte    # XOR operation
                for _ in range(8):    # Process each bit
                        if crc & 0x0001:    # If the least significant bit is 1
                                crc = (crc >> 1) ^ 0xA001    # Right shift and XOR with polynomial
                        else:
                                crc = crc >> 1    # Shift right directly
        return crc

def example_test():
        """
        Temperature controller communication example test function
        Includes both TTL command and ModBus protocol communication methods
        """
        try:
                # Open serial port (COM9, baud rate 38400)
                # Parameter description:
                # port: Serial port number
                # baudrate: Baud rate
```

```
# bytesize: Data bits (8 bits)
# parity: Parity (N for no parity)
# stopbits: Stop bits (1 bit)
# timeout: Read timeout in seconds
with serial.Serial('COM9', baudrate=38400, bytesize=8,
                   parity='N', stopbits=1, timeout=0.05) as ser:

    # ========== TTL command to set target temperature ==========
    # Send temperature setting command (channel 1 target temperature 25°C)
    # Command format: TC1:TG=2500000@\n
    # 2500000 corresponds to 25.00000°C
    send_data = b"TC1:TG=2500000@\n"
    ser.write(send_data)   # Send command
    received_data = ser.read(32).decode('ascii')   # Read return data

    # Check if return data is correct
    if received_data == "OKTC1:TG=2500000@\r\n":
        print("Target temperature set to 25°C")
    else:
        print("Temperature setting failed")
        return

    # ========== TTL command to query current temperature ==========
    time.sleep(0.005)   # Wait 5ms to avoid consecutive data transmission
    # Send temperature query command (channel 1 current temperature)
    # Command format: TC1:TCADJTEMP=?@\n
    send_data = b"TC1:TCADJTEMP=?@\n"
    ser.write(send_data)   # Send command
    received_data = ser.read(32).decode('ascii')   # Read return data

    # Check if return data format is correct
    # Correct format: "OKTC1:TCADJTEMP=xxxxx@\r\n"
    if not (received_data.startswith("OKTC1:TCADJTEMP=") and
            received_data.endswith("@\r\n")):
        print("Response data error")
    else:
        # Extract temperature value part (remove fixed characters from front and back)
        temp_str = received_data[16:-3]
        try:
            # Convert string to integer and calculate actual temperature value
            # Temperature value needs to be divided by 100000 to get actual temperature (°C)
            current_temp = int(temp_str) / 100000.0
            print(f"Current temperature: {current_temp}°C")
        except ValueError:
            print("Failed to parse temperature data")
```

```
# ========== ModBus protocol to set target temperature ==========
time.sleep(0.005)   # Wait 5ms

# Construct ModBus command frame
# Function code 0x10: Write multiple registers
# Register address 0x1000: Target temperature setting
# Data: 2500000 (corresponds to 25.00000°C)
data = bytearray([
    0x01,           # Device address (assumed to be 1)
    0x10,           # Function code (write multiple registers)
    0x10, 0x00, # Register address (0x1000)
    0x00, 0x02, # Number of registers (2)
    0x04,           # Number of data bytes (4 bytes)
    (2500000 >> 24) & 0xFF,   # Temperature value high byte
    (2500000 >> 16) & 0xFF,
    (2500000 >> 8) & 0xFF,
    2500000 & 0xFF              # Temperature value low byte
])

# Calculate CRC check and add to end of command
crc = crc_update(data)
data.append(crc & 0xFF)      # CRC low byte
data.append(crc >> 8)         # CRC high byte

# Send ModBus command
ser.write(data)

# Read response data
received_data = ser.read(32)

if len(received_data) < 8:   # Check minimum response length
    print("ModBus response data too short")
else:
    # Verify CRC check
    crc_calc = crc_update(received_data[:-2])
    crc_recv = received_data[-2] | (received_data[-1] << 8)

    if crc_calc != crc_recv:
        print("ModBus CRC check failed")
    else:
        # Parse response data
        if received_data[0] != 0x01:
            print("Device address error")
        elif received_data[1] != 0x10:
```

```
                print("Function code error")
            elif received_data[2] != 0x10 or received_data[3] != 0x00:
                print("Register address error")
            elif (received_data[4] << 8 | received_data[5]) != 0x0002:
                print("Register quantity error")
            else:
                print("ModBus target temperature set to 25°C")

# ========== ModBus protocol to query current temperature ==========
time.sleep(0.005)  # Wait 5ms

# Construct ModBus query command
# Function code 0x03: Read holding registers
# Register address 0x1002: Current temperature reading
data = bytearray([
    0x01,       # Device address
    0x03,       # Function code (read holding registers)
    0x10, 0x02, # Register address (0x1002)
    0x00, 0x02  # Number of registers to read (2)
])

# Calculate CRC check
crc = crc_update(data)
data.append(crc & 0xFF)
data.append(crc >> 8)

# Send query command
ser.write(data)

# Read response data
received_data = ser.read(32)

if len(received_data) < 9:  # Check minimum response length
    print("ModBus response data too short")
else:
    # Verify CRC check
    crc_calc = crc_update(received_data[:-2])
    crc_recv = received_data[-2] | (received_data[-1] << 8)

    if crc_calc != crc_recv:
        print("ModBus CRC check failed")
    else:
        # Parse response data
        if received_data[0] != 0x01:
            print("Device address error")
```

```
                elif received_data[1] != 0x03:
                    print("Function code error")
                elif received_data[2] != 0x04:
                    print("Data length error")
                else:
                    # Extract temperature value (4 bytes)
                    temp_value = ((received_data[3] << 24) |
                                  (received_data[4] << 16) |
                                  (received_data[5] << 8) |
                                  received_data[6])
                    current_temp = temp_value / 100000.0
                    print(f"ModBus current temperature: {current_temp}°C")

        except serial.SerialException as e:
            print(f"Serial communication error: {e}")
        except Exception as e:
            print(f"Program error: {e}")

if __name__ == "__main__":
    example_test()
```

Version Change Log

Version No.	Modification Details	Change Date	Reviewer
1.0	Latest Initial Version	2025/4/10	WST、WYR
1.0-1.1	New additions: 4. Command preview table New thermostat models	2025/4/21	WST、WYR
1.1-1.2	Content optimization	2025/8/8	WST、WYR
1.2-1.3.0	Content optimization	2026/1/7	WST、YJH、ZZA

Web: www.sensefuture.com

Tel: +86 187 1868 8108

Mail: sales@sensefuture.com

Add: Room 1308, Zone A1, Sanhuan Technology Building,
Fenghuang Subdistrict, Guangming District Shenzhen,
Guangdong Province, China

SenseFuture

Original Aspiration Determines the Future, Innovation Creates Value, Sharing Unites Hearts.

Anticipating a win-win collaboration!

Support
Downloads

SenseFuture Technologies Co., Ltd.