format

RS-232 COMMUNICATION INSTRUCTION MANUAL

Protocol Reference Guide

About This Manual

This manual is designed as a reference for communicating with the FTC100D Smart Temperature Controller for thermoelectric systems using the RS232 Serial Interface. Designed for system developers and advanced thermal management applications, the RS232 interface connection enables you to configure and command the FTC100D controller electronically.

This manual provides a basic overview of the communication protocol used by the FTC100. Once you have become familiar with commanding the FTC100D controller using the terminal interface, you can combine commands using scripting tools for advanced temperature control applications.

Before You Begin:

- You should have access to a computer with terminal software and an RS232 serial connection.

- You should set up and connect the FTC100D controller, according to the instructions supplied in the Quick Start Guide.

RS-232 COMMUNICATION INSTRUCTION MANUAL

Protocol Reference Guide

1. COMMUNICATION FUNCTIONS

1.1 General

For the computer (master unit) to communicate with the controller (slave unit), the format of the transmit/receive data must coincide.

For the FTC100D series, the format of the communication data is determined by the protocol of RTU mode (Remote Terminal Unit).

1.2 Quick Start

a. Communication protocol default: 38400, N, 8, 1
b. Read Chap# 6 & 7 for protocol format and address map
c. Read Chap# 8 for pseudo C program example

2. SPECIFICATIONS

2.1 Communication Specifications

Item	Specification	
Electrical Specification	Based on EIA RS-232	
Transmit system	3-wire, semi-duplicate	
Synchronizing System	Asynchronous mode	
Number of connected Units	1 unit	
Transmission Distance	500m max	
Transmission speed	2400 / 4800 / 9600 / 19200 / 38400 Selectable	
Data format	Type	Length
	Data length bit	8 bits
	Parity bit	None
	Stop bit	1 bits
Transmission code	HEX value	
Isolation	Functional isolation between transmission circuit and others (with stand voltage:500V AC)	

3. CONNECTIONS

Instructions for connecting the RS232 interface port are detailed in the Quick Start Guide. Please refer to those instructions for more information.

! WARNING
To avoid electric shock and potential damage to hardware, do not turn on the power supply until all wiring has been completed.

3.1 Wiring
Use shielded twisted pair cable.
The recommended cable type: UL2464, UL2448, etc.
The shielded wire of the cable should be grounded at the master unit.

4. SETTING UP THE CONTROLLER FOR COMMUNICATION

Before you begin communicating with the FTC100, you must configure the controller using the front panel keys. This section explains how to configure the controller.

Note: When connecting multiple FTC100 controllers in series on a line, they must be set to addresses (ADDR) that are different from each other.

4.1 Controller Set Items

The following table shows the parameters that must be set. Set them using the front panel keys.

Parameter	**Item**	**Value at delivery**	**Setting range**	**Remarks**
BAUD	Transmission speed	38400	2400 / 4800 / 9600 19200 / 38400	Set to the same communication protocol setup
-----	Data length	8 bits	Fixed (cannot be changed)	
-----	Stop bit	1 bit	Fixed (cannot be changed)	
-----	Parity setting	None	Fixed (cannot be changed)	
ADDR	Address	1	1 to 255	

4.2 Setting the Operation Method

The following example shows how to set up the controller for communication.

- ***Example:*** In this example, the controller is set to a transmission speed of 9600 bps and the controller's address is set at 12.

Key operation	**Indication**	**Description**
Power ON	25 100	Power on running state (PV/SV indication)
SET + ◀ (5seconds)	LEVL Pid	Press SET and SHIFT key simultaneously for approximately 5 seconds to get level parameter
▲ or ▼	LEVL OPTI	Press ▲ or ▼ key to select option level.
SET	TYPE K	Press SET key to go into option level.
SET	ADDR 0	Press SET key repeatedly until ADDR is display.
▲ or ▼	ADDR 1	Press ▲ or ▼key to setting "ADDR=1"
SET	BAUD 9.6K	Press SET key again to select next parameter "BAUD"
▲ or ▼	BAUD 9.6K	Press ▲ or ▼ key to setting "BAUD=9.6K"
SET	BAUD 9.6K	Press SET key to save "BAUD".
SET +▲	25 100	Press SET + ▲ one time to return the normal indication (PV/SV indication).

5. COMMUNICATION PROTOCOL

5.1 General

The communication system follows a standard protocol. Communication is always started from the master unit (computer) and then a slave unit (controller) responds to the received message.

Transmission procedures are shown below:

(a) The master unit sends a command message to a slave unit.

(b) The slave unit checks if the address in the received message matches its address.

(c) If the address matches, the slave unit executes the command and sends back a response message.

(d) If the message doesn't match, the slave unit ignores the command message and waits for the next command message.

(e) For systems using multiple controllers (slave units) on the same line, the master unit can communicate with any slave unit individually by setting the address of the specific unit in the command message.

5.2 Composition of Message

Both command and response messages consist of three fields — Address, Function code, and Data — sent in this same order. For all fields, the allowable characters transmitted are hexadecimal 0~9 & A~F.

RTU mode framing

ADDRESS	FUNCTION	DATA
8 BITS	8 BITS	N × 8 BITS

The following provides a detailed explanation of the specific field:

(a) Address

Address is the number specifying a slave unit. Valid slave device addresses are in the range 0f 1-255 decimal. A master addresses a slave by placing the slave address in the address field of the message.

When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding.

Note: Each master can only talk to one slave unit, it is recommended to set the address to '1' as default.

(b) Function

This is a code to designate the function executed at a slave unit. When a message is sent from a master to a slave device, the function code field tells the slave what kind of action to perform.

When the slave responds to the master, it uses the function code field to indicate either a normal response or that some kind of error occurred. For normal response, the slave echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most-signification bit set to logic 1.

(c) Data

The Data field contains the values required for the function to execute. The format of the data field varies by the function codes. Refer to chapter 6 for details. A data address is assigned to each data in the temperature controller.

5.3 Response of the Slave Unit (controller)

When the slave unit receives a command, it responds to the command. This section explains how the controller responds to a command.

(a) Response to a normal command

When responding to a relevant message, the slave unit creates a response message that corresponds to the original command message and sends it back. The composition of the message is the same as detailed in section 5.2. The content of the data field depends on the function code. For details about function codes, refer to Chapter 6.

(b) Response to an abnormal command

Other than a transmission error, if the contents of a command message generate an abnormality (for example, non-actual function code is designated), the slave unit does not execute that command; instead, the slave unit creates a response message at error detection and sends it back.

The composition of the response message at error detection is shown below; the value used for the function field is the function code of the command message plus 80H.

ADDRESS	FUNCTION (Function code + 80H)	ERROR CODE
8 BITS	8 BITS	8 BITS

Error Codes

Error Code	Contents	Description
01	Illegal function	The function code received is not an allowable action for the slave.
02	Illegal data address	The data address received is not an allowable address for the slave.
03	Illegal data value	A value contained in the data field is not an allowable value for the slave.

5.4 Function Codes

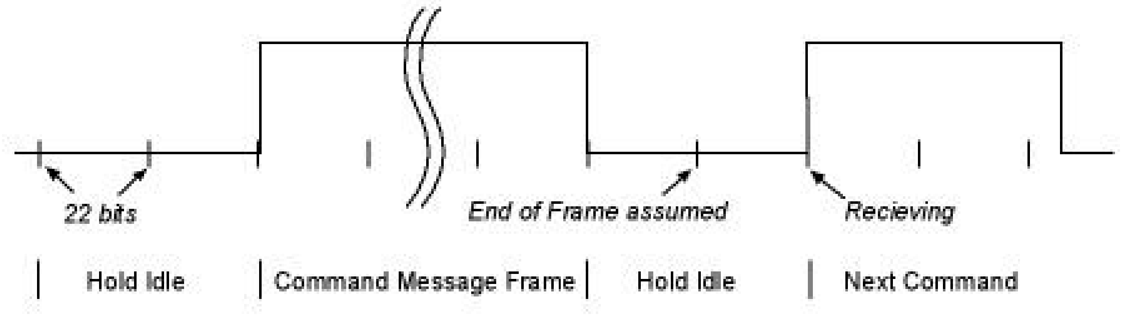
The listing below shows the function codes supported by the controller.

Function code		
Code	**Function**	**Object**
03	Read back	Read back holding register
04	Read back	Read back value at PV register
06	Write-in	Write to RAM & EEPROM Register

5.5 Transmission Control Procedure

5.5.1 Transmission procedure of master unit

Communication from the master unit must conform to the following rules.



(a) Before sending a command message, hold idle for 44 bits time or more.

(b) The time between bytes of command message should be less than 22 bits time.

(c) Within 22 bits time after sending a command message, the receiving status is posted.

(d) Between the end of a response message received and the beginning of the next command message sent, provide 44 bits time or more to held idle (same as in a.1).

(e) To secure the communication, it is suggested to read back the response message and to provide 3 or more retries in case of no response or error occurrence, etc.

Note: The above definitions are for most unfavorable states. It is recommended to program the master unit to work with the timing for a safety factor of 2 to 3.More securely, to arrange the program for 9600 bps to idle 10ms or more (a), and within 1ms for byte interval (b) then changeover from sending to receiving (c).

5.5.2 Description

(a) Detecting the message frame

Since the communication system uses the 2-write RS-232 interface, there may be 2 conditions on a line, outlined below.

(i) Idle status (no data on line)

(ii) Communication status (data is existing)

Units connected on the line are initially in a receiving status, monitoring the line. When 22 bits of hold idle time (or more) is detected on the line, the end of the preceding frame is assumed; and, within following 22 bits of time, a receive status is posted. When data appears on the line, units receive it until 22 bits of hold idle time (or more) is detected again and the end of that frame is assumed. Data that appears on the line within the first 44ms of hold idle time and the next 44 bits of hold idle time is fetched as one frame.

5.5.1.a There should be at least 44 bits time or more prior to sending command message.

5.5.1.b The time between bytes of command message should be less than 22 bits time.

(b) Response from the FTC100 Receiving Terminal Unit (RTU)

Beginning 22 bits of time or more after a frame is detected, the RTU starts processing the frame as a command message. If the command message is addressed to the unit, a response message is returned. The unit's processing time is from 1 to 10ms (depending on the contents of command message).

After sending a command message, the master unit must observe the following.

5.5.1.c After sending a command message, a receiving status must be posted within 22 bits of time.

5.6 Fix Processing (Cautions at write parameters data to EEPROM)

The controller (slave unit) uses non-volatile memory (EEPROM) to hold the setting parameters. When data is written in the non-volatile memory, it won't be lost even if the power is turned off. FIX processing is required to store the parameters written in via communication in the non-volatile memory. FIX execution writes the parameters from the internal memory into the non-volatile memory. See 6.4 for further detail.

Caution:

FIX processing lasts approximately 0.1 seconds.
The EEPROM has a limited number of write-in times. The guaranteed number of write-in times is 10000 minimums. Avoid FIX processing except when absolutely necessary (for example, after rewriting the setting parameters). Failure to follow this practice may result in a dramatic reduction in the life of the controller.

6. MESSAGE DETAILS

6.1 Read back of Word Data [Function Code: 03]

This function reads back the contents of holding registers (hex:0000~ 007D) in the slave. A broadcast of this function code is not possible. This function reads in all the parameters of the controller except the PV value.

(a) Message composition

Command message composition

Address	Function	Starting Address	Word Number*
01 ~ FF	03	00xx	0001 ~ 0018
1 byte	1 byte	2 byte	2 bytes

* Maximum word number = 7E

Response message composition

Address	Function	Byte Number *	Word Data
01 ~ FF	03	02 ~ 30	xxxx
1 byte	1 byte	1 bytes	N bytes

* Byte number = Word number × 2

(b) ***Example:*** Message transmission

The following shows an example of reading the Setpoint Value (hex:0000) from address #1 controller.

Command message composition

Address	Function	Starting Address	Word Number
01	03	0000	0001

Response message composition

Address	Function	Byte Number	Word Data
01	03	02	03E8

6.2 Read back of Read-Only Word Data [Function Code: 04]

This function reads the contents of the input registers (hex:1000) in the slave.
A broadcast of this function code is not possible. This function only does one thing, which is reading in PV value (your sensor temperature).

(a) Message composition

Command message composition

Address	Function	Starting Address	Word Number
01 ~ FF	04	1000	0001
1 byte	1 byte	2 bytes	2 bytes

Response message composition

Address	Function	Byte Number	Word Data
01 ~ FF	04	02	xxxx
1 byte	1 byte	1 bytes	2 bytes

(b) ***Example:*** Message transmission

The following shows an example of reading the Process Value (27) from address #1 controller.

Command message composition

Address	Function	Starting Address	Word Number
01	04	1000	0001

Response message composition

Address	Function	Byte Number	Word Data
01	04	02	001B

6.3 Write Parameter Data (1 word) to RAM address [Function Code: 05]

Use this function for non-fixed processing (to store the holding register data into RAM) . When broadcast, the function code affects all attached slaves.

(a) Message composition

Command message composition

Address	Function	Starting Address	Word Data
1 ~ FF	05	00xx	xxxx
1 byte	1 byte	2 bytes	2 bytes

Response message composition

Address	Function	Starting Address	Word Data
1 ~ FF	05	0xxx	xxxx
1 byte	1 byte	2 bytes	2 bytes

(b) ***Example:*** Message transmission

The following shows an example of storing the Setpoint Value of address #1 controller into RAM.

Command message composition

Address	Function	Starting Address	Word Data
01	05	0000	01F4

Response message composition

Address	Function	Starting Address	Word Data
01	05	0000	01F4

6.4 Write Parameters Data (1 word) to EEPROM [Function Code: 06]

Use this function for Fix processing (to store the holding register data into EEPROM).

(a) Message composition

Command message composition

Address	Function	Starting Address	Word Data
1 ~ FF	06	00xx	xxxx
1 byte	1 byte	2 bytes	2 bytes

Response message composition

Address	Function	Starting Address	Word Data
1 ~ FF	06	0xxx	xxxx
1 byte	1 byte	2 bytes	2 bytes

(c) ***Example:*** Message transmission

The following shows an example of storing the Setpoint Value of address #1 controller into RAM & EEPROM.

Command message composition

Address	Function	Starting Address	Word Data
01	06	0000	01F4

Response message composition

Address	Function	Starting Address	Word Data
01	06	0000	01F4

7. ADDRESS MAP AND DATA FORMAT

7.1 Data Format

(a) Transmission data format

The communication protocol used by the FTC100 is RTU mode.

Transmitted data is "numeric value" and not "ASCII code".

(b) Internal calculation value and engineering units

There are 3 different kinds of set value used by the FTC100 series controllers.

(b.1) Normal value:

The data value is transfer into Hexadecimal regardless of decimal. For example:

1000 is translated to 03E8(hex)
And for output percentage and Pb (proportional band), 100.0 % is also translated to 03E8(hex).

(b.2) Time value:

For time values such as St1(soak time) and rt1(ramp time), the value will be expressed as follows:

55.30 (55 minute 30 second) will be express as 0D02(hex)
which is calculated by (55 × 60) ＋ 30 = 3330 (sec.)

(b.3) English code:

Some parameter values are set by index code. For example, to change the unit to ℃ via communication; the data value would be 0014(hex).

Code	English	Code	English	Code	English	Code	English	Code	English
00	OFF	01	ON	02		03	REV	04	DIR
05	J	06	K	07	T	08	E	09	B
0A	R	0B	S	0C	N	0D	C	0E	D-PT
0F	J-PT	10	TR-1	11		12		13	
14	℃	15	℉	16		17	0000.	18	000.0
20	OFF	21	EnOn	22	PROG	23	HOLD	24	SEC
25	MIN	26	END	27	HOLD	28	LOOP	29	NEXT

7.2 Data Address Map

Word data map (read-out/write-in): Function code [03 , 06]

Data Address	Parameter	Range	Unit
0000	SV	HiLt ~ LoLt	℃ / ℉
0001	A1SP	HiLt ~ LoLt	℃ / ℉
0002	A2SP	HiLt ~ LoLt	℃ / ℉
0003	AT	Off / On (English code)	
0004	HAND	Off / On (English code)	
0005	OUTL	-100.0% ~ 100.0%	%
0006	ENAB	Off / EnOn / PROG / HOLD(English code)	
0007	PB	0.0% ~ 300.0%	%
0008	TI	3600	Sec
0009	TD	900	Sec
000A	DB	NO USE	N/A
000B	MR	0.0% ~ 51.0%	℃ / ℉
000C	AR	0.0% ~ 100.0%	℃ / ℉
000D	SPOF	-1000 ~ 1000	℃ / ℉
000E	PVOF	-1000 ~ 1000	℃ / ℉
000F	ACT	REV / DIR (English code)	
0010	TYPE	J / K / T / E / B / R / S / N / C DPT / JPT / TR1 (English code)	
0011	UNIT	℃ / ℉ (English code)	
0012	DP	0000. / 000.0 (English code)	
0013	LOLT	-1999 ~ 9999	℃ / ℉
0014	HILT	-1999 ~ 9999	℃ / ℉
0015	FILT	0.0 ~ 9.9	
0016	ADDR	0 ~ 255	
0017	BAUD	2400 / 4800 / 9600 / 19200 / 38400 (English code)	

The following address map is for script programming:

0018	PANO	1 ~ 16	
0019	PTME	SEC / MIN (English code)	
001A	STAR	Off / On (English code)	
001B	BAND	200 ~ 0	℃ / ℉
001C	RT1	0 ~65535	Min / Sec
001D	SP1	HiLt ~ LoLt	℃ / ℉
001E	ST1	0 ~65535	Min / Sec
001F	SF1	END / HOLD / LOOP / NEXT	

		(English code)	
0020	IN1	8 ~ 1	
0021	LN1	100 ~ 1	
0022	RT2	0 ~65535	Min / Sec
0023	SP2	HiLt ~ LoLt	℃ / ℉
0024	ST2	0 ~65535	Min / Sec
0025	SF2	END / HOLD / LOOP / NEXT (English code)	
0026	IN2	8 ~ 1	
0027	LN2	100 ~ 1	
0028	RT3	0 ~65535	Min / Sec
0029	SP3	HiLt ~ LoLt	℃ / ℉
002A	ST3	0 ~65535	Min / Sec
002B	SF3	END / HOLD / LOOP / NEXT (English code)	
002C	IN3	8 ~ 1	
002D	LN3	100 ~ 1	
002E	RT4	0 ~65535	Min / Sec
002F	SP4	HiLt ~ LoLt	℃ / ℉
0030	ST4	0 ~65535	Min / Sec
0031	SF4	END / HOLD / LOOP / NEXT (English code)	
0032	IN4	8 ~ 1	
0033	LN4	100 ~ 1	
0034	RT5	0 ~65535	Min / Sec
0035	SP5	HiLt ~ LoLt	℃ / ℉
0036	ST5	0 ~65535	Min / Sec
0037	SF5	END / HOLD / LOOP / NEXT (English code)	
0038	IN5	8 ~ 1	
0039	LN5	100 ~ 1	
003A	RT6	0 ~65535	Min / Sec
003B	SP6	HiLt ~ LoLt	℃ / ℉
003C	ST6	0 ~65535	Min / Sec
003D	SF6	END / HOLD / LOOP / NEXT (English code)	
003E	IN6	8 ~ 1	
003F	LN6	100 ~ 1	
0040	RT7	0 ~65535	Min / Sec
0041	SP7	HiLt ~ LoLt	℃ / ℉
0042	ST7	0 ~65535	Min / Sec
0043	SF7	END / HOLD / LOOP / NEXT (English code)	
0044	IN7	8 ~ 1	

0045	LN7	100 ~ 1	
0046	RT8	0 ~65535	Min / Sec
0047	SP8	HiLt ~ LoLt	℃ / ℉
0048	ST8	0 ~65535	Min / Sec
0049	SF8	END / HOLD / LOOP (English code)	
004A	IN8	8 ~ 1	
004B	LN8	100 ~ 1	

Word data map (read-out only) : Function code [04]

Data Address	**Parameter**	**Contents**	**Unit**
1000	PV	Process value	℃ / ℉

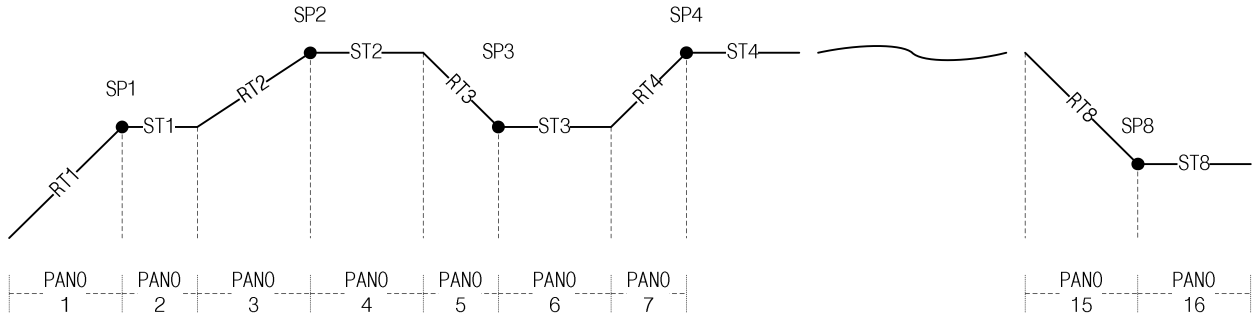
Description of Script Program Coding:

ENAB:

“OFF”: Amplifier Enable is OFF
“EnOn”: Enable Amplifier for single SV control
“PROG”: Running Script Program
“HOLD”: Hold the temperature at the end of the Step without turn off the amp

PANO:

Return the current script step (number from 1 – 16)



PTME:

Unit of time used in the script. Choose either SEC(second) or MIN(minute).

BAND:

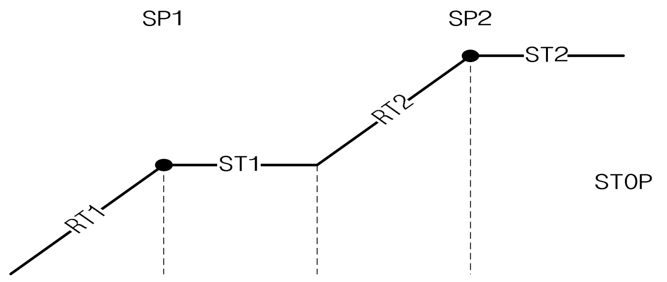
Tolerance band; When SV-BAND ≦ PV ≦ SV+BAND is satisfied, the program will go to next step when set time is reached.

PTx: Ramping time from last temperature to new SPx; 0 ~ 65535

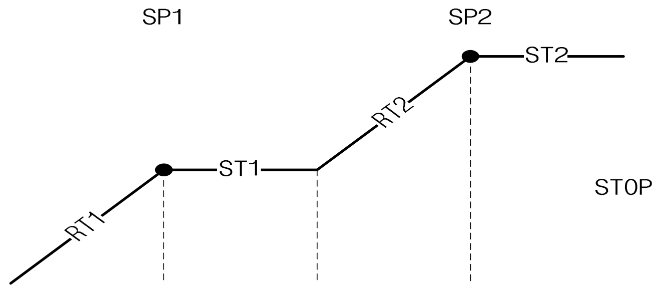
SPx: Set Temperature; HiIt ~ LoIt (within Hi/Lo limit)

STx: Holding Time at SPx temperature

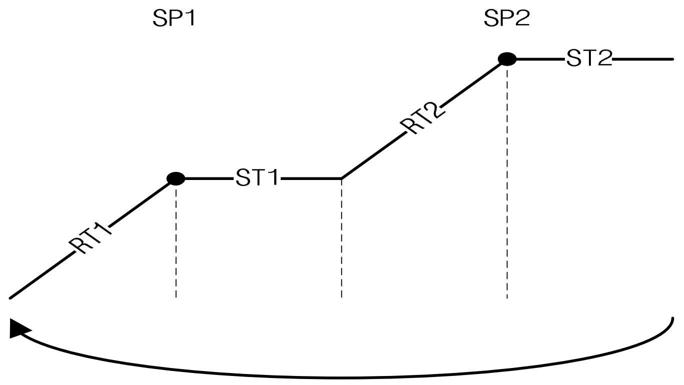
SFx: Script Functional setting for END/HOLD/LOOP/NEXT
Caution: only one loop can be used, because there is only one loop register, using more than two loops in the script program can result in infinitive loops.



if SF3 = "END" then outl = 0.0%



if SF3 = "HOLD" then outl hold



if SF3 = "LOOP" and IN3 = 1

INx: Indicate the initial loop step position

Lnx: Indicate the number of the loop to execute

8. Programming Code Example

If you would like to write your own software, we listed 3 examples here. Basically, when a user send a command to the controller, either is Read or Write command. The controller always returns with response char array. These response char array needs to be read and clear from the buffer.

8.1 Code Example in pseudo C

(a) Read SV value (SV: Set Value – the expecting temperature)

```
unsigned char Combuf[6]={0x01,0x03,0x00,0x00,0x00,0x01};
unsigned char Inbyte[5];
short int RdVal;
float fSV;1
    ComWrt(ComPort,Combuf,6);
    Delay (0.05); // delay 50msec
    ComRd (ComPort, Inbyte, 5);
    If( Inbyte[1]<0x80 ){
        RdVal=(short int)Inbyte[3]*256+(short int)Inbyte[4];
        fSV= (float)RdVal / 10.; // adjust to one decimal points
    } else { //process your error code here }
```

(b) Write in SV with 75.5°C (Hex number for 755 is x02F3; 75.5x10.=755)

```
unsigned char Combuf[6]={0x01,0x06,0x00,0x00,0x02,0xF3};
unsigned char Inbyte[6];
    ComWrt(ComPort,Combuf,6);
    Delay (0.05); // delay 50msec
    ComRd (ComPort, Inbyte, 6);
    If( Inbyte[1]<0x80 ) { // the 5 hex values in Inbyte[] should be the same as the Combuf[] }
    else { // process your error code here }
```

(c) Read PV value (PV: Process Value - the current temperature reading from the sensor)

```
unsigned char Combuf[6]={0x01,0x04,0x10,0x00,0x00,0x01};
unsigned char Inbyte[5];
short int RdVal;
float fSV;
    ComWrt(ComPort,Combuf,6);
    Delay (0.05); // delay 50msec
    ComRd (ComPort, Inbyte, 5);
    If( Inbyte[1]<0x80 ){
        RdVal=(short int)Inbyte[3]*256+(short int)Inbyte[4];
        fSV= (float)RdVal / 10.; // adjust to one decimal points
    } else { //process your error code here }
```