Instruction Sets V2.5RC

Contents

Update Description

Version number	Update date	Prepared by	Reviewed by	Firmware version	Update description
V2.5RC	2023-03-21			V3.0.05.230321RC	Add the function of getting the name of the currently displayed window
V2.4RC	2023-03-17			V3.0.04.230317RC	Add functions such as external disk file reading, writing and uploading
V2.3RC	2023-01-09			V2.4.10.230209RC	Add the instruction of set_color to change the color of the widget in batch.
V2.2RC	2022-12-16			V2.4.09.221216RC	Add file renaming instruction
V2.1RC	2022-12-14			V2.4.08.221214RC	Add U disk identification and file write data status return instruction
V2.0RC	2022-10-27			V2.4.05.221027RC	Add functions related to reading and writing user files and downloading.
V1.9RC	2022-07-29			V2.2.21.220729RC	Add 3.4 and 3.5 general instructions get_xy and get_wh
V1.8RC	2022-07-27			V2.2.20.220727RC	Add set_frame instruction to the image animation widget, which can be set to display a specific frame in the pause/stop state
V1.7RC	2022-07-26			V2.2.19.220714RC	Add the instruction to set the value of edit/slider/pg_bar/image_value and other widgets in batch
V1.6RC	2022-07-06			V2.2.17.220706RC	Add the instruction to set the value and text of label in batch
V1.5RC	2022-06-15			V2.2.16.220615RC	Add chart_view axis related instructions
V1.3RC	2022-04-08			V2.2.13.220329RC	Add description of button related instruction set
V1.2RC	2022-04-05			V2.2.12.220324RC	Perfect instruction set example description
V1.1RC	2022-03-25			V2.2.12.220324RC	Instruction set description optimization
V1.0RC	2022-03-25			V2.2.11.220323RC	Summary of the instruction set

1. Instruction Description

1.1 MCU→HMI Module

The instructions sent by the main controlling device MCU to the HMI module are in the form of JSON plain text, with good self-description, hierarchical structure, high readability and scalability; In addition, the frame header and frame tail are added to improve the security and anti-collision of the instructions.

The format is as follows:

```
Frame header        +                  data                  +            frame tail
ST<      {"cmd_code":"set_text","type":"label","widget":"label","text":"Hello"}      >ET
```

Category	Content	Description	Remarks
Frame header	ST<	data frame header	Data start identifier
cmd_code	Instruction code	used to distinguish different instructions	The instruction code is functionally exclusive and is the unique identifier to distinguish different instructions
type	Type	widget type	Used to distinguish different widget types
widget	Widget name	widget name	Used to distinguish different widgets, the exclusive identifier of the widget
text	Text	function field	Data content part, different for different instructions
….	Same as above	other functional fields	Data content part, different widgets may be different
Frame tail	>ET	date frame tail	End of data identification

1. The data format adopts the format of frame header + data + frame tail, and the intermediate data part adopts JSON format and verification, and the maximum length of a single instruction should not exceed 20k bytes.
2. The intermediate JSON text part includes cmd_code, type, widget, etc. For details, see the above table; each JSON instruction is different, and you can choose the instruction according to your own needs.
3. The "instruction sending" described below refers to the instruction sent by the MCU to the HMI module;
4. Support setting the text/value of the widget in batch in the form of arrays, the naming rules of the widget are: "ASCII letter" + "start index" + "_" + "end index"; the amount of elements of the array must correspond to the name of the widget; for example, label1_11 means the name of the widget is label1 to label11, arrays ["30", "10", "12", " 800", "15", "12.8", "48.2", "52.6", "18.6", "13.5", "16.3"] have 11 text contents corresponding to them; for more details, please refer to the set_value/set_text instructions for label/edit widgets;

1.2 HMI Module → MCU

The data protocol sent by the HMI module adopts the form of hexadecimal format, which can effectively reduce the analysis difficulty and processing burden of the MCU of controlling module; Adding CRC16 verification is capable to improve the security of data significantly;

The format is as follows:

Frame header	+	CMD	+	LEN	+	DATA	+	Frame tail	+	Verification
ST<		0x1068		0x0004		0x01 0x02 0x03 0x04		>ET		CRC16

Category	Content	Length (bytes)	Remarks
Frame header	ST<	3	Data frame header
CMD	see below for details	2	The unique identifier of the MCU to distinguish the instruction
LEN	length	2	The length of the data part, excluding the frame header, frame tail, CMD, LEN and verification
DATA	see below for details	=LEN	The data part is generally composed of widget name + data
Frame tail	>ET	3	Date frame tail
Verification	CRC16	2	Adopt CRC16/MODBUS verification; high order in front, low order in back

1. The "instruction return" described below refers to the instruction issued by the HMI module to the MCU;

2. System Instruction

2.1 boot_cmd

Instruction return:

Return instruction	Description	Delivery type	Remarks
0x0000	system operating status	initiative	The startup program automatically send three times at an interval of 100ms

Data description:

Category	Data	Delivery type	Remarks
CMD	0x0000	initiative	The startup program automatically send three times at an interval of 100ms
LEN	0x0001		
DATA	0x01: system running 0x02: system standby (screen backlight off) 0xFF: system operation error		Last byte of data part

For example:

System running

Response: ST<0x00 0x00 0x00 0x01 0x01>ET

HEX:53 54 3C 00 00 00 01 01 3E 45 54 AB 25

2.2 sys_reboot

Instruction sending:

Instruction	Instruction description	Remarks
sys_reboot	system restart	Rquest MCU to restart HMI module

For example:

Send: ST<{"cmd_code":"sys_reboot","type":"system"}>ET

2.3 sys_hello

Instruction sending:

Instruction	Instruction description	Remarks
sys_hello	used for verifying whether the communication is in normal status	Device returns 0x0001 instruction

Instruction return:

Instruction	Instruction description	Delivery type	Remarks
0x0001	return instruction of system communication verification	passive	

Data description:

Category	Data	Delivery type	Remarks
CMD	0x0001	passive	System communication verification return instruction
LEN	0x0001		
DATA	0x01: system is in running mode		Last byte data part of instruction

For example:
Send: ST<{"cmd_code":"sys_hello","type":"system"}>ET
Response: ST<0x00 0x01 0x00 0x01 0x01>ET
HEX:53 54 3C 00 01 00 01 01 3E 45 54 6B 35

2.4 sys_version

Instruction sending:

Instruction	Instruction description	Remarks
sys_version	Acquiring version of GUI Software	Device returns 0x0002 instruction

Instruction return:

Return instruction	Return description	Delivery type	Remarks
0x0002	GUI software version number distribution (get_version)	passive	

Data description:

Category	Data	Delivery type	Remarks
CMD	0x0002	passive	GUI software version number sent from HMI to MCU
LEN	version number length		
DATA	software version		

For example:
a) Obtain the GUI software version, version number: v2 2.13.220329RC
Send: ST<{"cmd_code":"sys_version","type":"system"}>ET
Response: ST<0x00 0x02 0x00 0x10 V2.2.13.220329RC>ET
HEX:53 54 3C 00 02 00 10 56 32 2E 32 2E 31 33 2E 32 32 30 33 32 39 52 43 3E 45 54 1C 58

2.5 set_sleep

Instruction sending:

Instruction	Instruction description	Remarks
set_sleep	set the device to sleep mode	Turn off the backlight, the program keep running in the background

Send data description:

Category	Description	Type	Remarks
sleep	Activate or deactivate sleep mode	bool	Activate or deactivate sleep mode

For example:

a) Set the device to sleep:

Send: ST<{"cmd_code":"set_sleep","type":"system","sleep":true}>ET

b) Turn off device sleep:

Send: ST<{"cmd_code":"set_sleep","type":"system","sleep":false}>ET

2.6 set_buzzer

Instruction sending:

Instruction	Instruction description	Remarks
set_buzzer	set the buzzer of HMI	Since the message queue is used for instruction sending and receiving, if the interval of the instruction message to widget the buzzer is less than the duration of the buzzer sound, the sound will continue after the instruction stops when the message accumulates.

Send data description:

Category	Description	Type	Remarks
time	time	uint	Unit ms, sound duration

For example:

Send: ST<{"cmd_code":"set_buzzer","type":"system","time":100}>ET

2.7 set_brightness

Instruction sending:

Instruction	Instruction description	Remarks
set_brightness	set backlight brightness level	Set LCD backlight brightness level percentage

Send data description:

Category	Description	Type	Remarks
brightness	LCD backlight brightness	uint	1. The value range is 0-100. 2. The backlight adjustment level of the old version is 0-7;

For example:

Send: ST<{"cmd_code":"set_brightness","type":"system","brightness":100}>ET

2.8 set_touch_cal

Instruction sending:

Instruction	Instruction description	Remarks
set_touch_cal	set touchscreen calibration (for resistive screens)	The HMI will be restarted automatically after the completion of calibration

For example:
Send: ST<{"cmd_code":"set_brightness","type":"system","brightness":100}>ET

2.9 clear_touch_cal

Instruction sending:

Instruction	Instruction description	Remarks
clear_touch_cal	clear touchscreen calibration data	for resistive screens

For example:
Send: ST<{"cmd_code":"set_touch_cal","type":"system"}>ET

2.10 set_touch_test

Instruction sending:

Instruction	Instruction description	Remarks
set_touch_test	touchscreen test	A manual restart is required for running user desgined UI interface

For example:
Send: ST<{"cmd_code":"set_touch_test","type":"system"}>ET

2.11 set_vol

Instruction sending:

Instruction	Instruction description	Remarks
set_vol	volume adjustment	
set_vol_inc	volume up	
set_vol_dec	volume down .	
set_mute	set mute	

Send data description:

Category	Description	Type	Remarks
vol	volume	uint	Value: 0-100, volume percentage
step	step Value	uint	Volume percentage
mute	mute	bool	Mute or not

For example:
a) Set volume to 50%:
Send: ST<{"cmd_code":"set_vol","type":"system","vol":50}>ET

b) Volume up by 5%:
Send: ST<{"cmd_code":"set_vol_inc","type":"system","step":5}>ET

c) Volume down by 5%:
Send: ST<{"cmd_code":"set_vol_dec","type":"system","step":5}>ET

d) Set mute:
Send: ST<{"cmd_code":"set_mute","type":"system","mute":true}>ET

e) Unmute:
Send: ST<{"cmd_code":"set_mute","type":"system","mute":false}>ET

2.12 set_audio

Instruction sending:

Instruction	Instruction description	Remarks
set_audio_play	play audio start	
set_audio_pause	play audio pause	After the playback ends, no need to replay the audio through this instruction
set_audio_stop	play audio stop	

Send data description:

Category	Description	Type	Remarks
audio	audio name	text	Played audio name, supports wav and mp3
pause	audio pause	bool	Whether to pause audio playback

For example:
a) Play audio 01.wav:
Send: ST<{"cmd_code":"set_audio_play","type":"system","audio":"01.wav"}>ET

b) Pause:
Send: ST<{"cmd_code":"set_audio_pause","type":"system","pause":true}>ET

c) Continue playing:
Send: ST<{"cmd_code":"set_audio_pause","type":"system","pause":false}>ET

d) Stop playing:
Send: ST<{"cmd_code":"set_audio_stop","type":"system"}>ET

2.13 request_upgrade_firmware

Instruction sending:

Instruction	Instruction description	Remarks
request_upgrade_firmware	Firmware remote upgrade	Requesting for remote firmware upgrading

Send data description:

Category	Description	Type	Remarks
file_url	Address for downloading firmware	text	Support IP + file path and DNS + the format of file path does not support URL redirection
port	Port number for downloading firmware	uint	The default port number is 80, user may adjust it according the needs
file_size	Size of file	uint	Size of file, which means the number of bytes of the file
file_md5	MD5 value of file	text	MD5 value of the file for verifying the completeness of the file
software_version	Version number of file	text	Version number of the file (temporarily not mandatory to be mentioned)

For Example:

a) Requesting for upgrading firmware remotely (LAN):

Send:

```
ST<{"cmd_code":"request_upgrade_firmware","type":"system","file_url":"192.168.1.43/HTTP/STONE/bin/HMI.bin","port":80,"file_size":2266716,"file_md5":"3516BEDF5A4006D9CA4B501C11A176B7","software_version":"V3.0.02.230308RC"}>ET
```

b) Requesting for upfrading firmware remotely (WAN):

Send:

```
ST<{"cmd_code":"request_upgrade_firmware","type":"system","file_url":"www.hmi.com:6080/HTTP/STONE/bin/HMI.bin","port":6080,"file_size":2375412,"file_md5":"04E7594A00DCE9C907D11F596D6B5671","software_version":"V3.1.06.230817RC"}>ET
```

Special Note: Users have to be noted that every single part of the sending instruction must be precisely accurate according the actual circumstances, any incorrect content may lead to the failure for upgrading. Users may refer to the failure notice from the screen which mentioning the cause of failure.

2.14 request_upgrade_file

Instruction sending:

Instruction	Instruction description	Remarks
request_upgrade_file	Individual file remote upgrading	Rquesting for upgrading individual file for the project files

Send data description:

Category	Description	Type	Remarks
file_url	Address for downloading file	text	Support IP + file path and DNS + the format of file path does not support URL redirection
port	Port number for downloading firmware	uint	The default port number is 80, user may adjust it according the needs
file_type	Types of files	text	Types of files:audio,font,image,style,ui,video,data
file_size	Size of file	uint	Size of file, which means the number of bytes of the file
file_md5	MD5 value of file	text	MD5 value of the file for verifying the completeness of the file

For Example:

a) Requesting for remote upgrading of individual image file “1.jpg” of UI project (LAN):

Send:

ST<{"cmd_code":"request_upgrade_file","type":"system","file_url":"192.168.1.43/HTTP/STONE/project/default/raw/images/xx/1.jpg","port":80,"file_type":"image","file_size":25577,"file_md5":"E2A9EB41A07F04C0AC88F806CE91CEE2"}>ET

b) Requesting for remote upgrading of individual font file “default.ttf” of UI project (WAN):

Send:

ST<{"cmd_code":"request_upgrade_file","type":"system","file_url":""www.hmi.com:6080/HTTP/STONE/project/default/raw/fonts/default.ttf","port":6080,"file_type":"font","file_size":9460612,"file_md5":"F6448C216A6834D74AFB31271FE78EAE"}>ET

Special Note: Users have to be noted that every single part of the sending instruction must be precisely accurate according the actual circumstances, any incorrect content may lead to the failure for upgrading. Users may refer to the failure notice from the screen which mentioning the cause of failure.

2.15 request_upgrade_project

Instruction sending:

Instruction	Instruction description	Remarks
request_upgrade_project	Whole UI project files remote upgrading	Requesting for remote upgrading the whole UI project files

Send data description:

Category	Description	Type	Remarks
file_url	Address for downloading files	text	支持 IP+路径和域名+路径的格式,不支持 url 重定向
port	Port number for downloading firmware	uint	The default port number is 80, user may adjust it according the needs
file_type	Types of files	text	Type of file:file_list;
file_size	Size of file	uint	Size of file, which means the number of bytes of the file
file_md5	MD5 value of file	text	MD5 value of the file for verifying the completeness of the file

For Example:

a) Requesting for remote upgrading the whole UI project files (LAN):

Send:

ST<{"cmd_code":"request_upgrade_project","type":"system","file_url":"192.168.1.43/HTTP/STONE/project/default/raw/data/file_list.csv","port":80,"file_type":"file_list","file_size":935,"file_md5":"5D34CE395131A19139B7EAEE5D2FC5C9"}>ET

b) Requesting for remote upgrading the whole UI project files (WAN):

Send:

ST<{"cmd_code":"request_upgrade_project","type":"system","file_url":"http://www.hmi.com:6080/HTTP/STONE/project/default/raw/data/file_list.csv","port":6080,"file_type":"file_list","file_size":946,"file_md5":"1420D0909526BD7FE787BB9C7AC9EAEF"}>ET

c) Requesting for remote upgrading the whole UI project files (WAN):

Send:

ST<{"cmd_code":"request_upgrade_project","type":"system","file_url":"www.hmi.com/HTTP/STONE/project/default/raw/data/file_list.csv","port":6080,"file_type":"file_list","file_size":946,"file_md5":"1420D0909526BD7FE787BB9C7AC9EAEF"}>ET

Special Note:

1.Users have to be noted that every single part of the sending instruction must be precisely accurate according the actual circumstances, any incorrect content may lead to the failure for upgrading. Users may refer to the failure notice from the screen which mentioning the cause of failure.

2.Please be noted that “file_list.csv”is mandatory to be as the target of “file_url” part of the instruction. “file_list.csv” usually generated by

3. General Instruction

3.1 set_enable

Instruction sending:

Instruction	Instruction description	Remarks
set_enable	set enabled state of widget	

Send data description:

Category	Description	Type	Remarks
enable	whether to enable	bool	Set the enabled state of the widget, the value is true/false

For example:

a) Set the button1 widget available:

Send: ST<{"cmd_code":"set_enable","type":"widget","widget":"button1","enable":true}>ET

b) Set the button1 widget unavailable:

Send: ST<{"cmd_code":"set_enable","type":"widget","widget":"button1","enable":false}>ET

3.2 set_visible

Instruction sending:

Instruction	Instruction description	Remarks
set_visible	set the visible state of the widget	

Send data description:

Category	Description	Type	Remarks
visible	visible or not	bool	Set whether the widget is visible, the value is true/false

For example:

a) Set the button1 widget visible:

Send: ST<{"cmd_code":"set_visible","type":"widget","widget":"button1","visible":true}>ET

b) Set the button1 widget invisible:

Send: ST<{"cmd_code":"set_visible","type":"widget","widget":"button1","visible":false}>ET

3.3 set_xy

Instruction sending:

Instruction	Instruction description	Remarks
set_xy	set coordinates of widget	x y is of type int and can be negative.

Send data description:

Category	Description	Type	Remarks
x	x-axis coordinate	int	x-axis coordinate value
y	y-axis coordinate	int	y-axis coordinate value

For example:
a) Set slider1 xy coordinates to (0,0):
Send: ST<{"cmd_code":"set_xy","type":"widget","widget":"slider1","x":0,"y":0}>ET

b) Set slider1 xy coordinates to (-40,240):
Send: ST<{"cmd_code":"set_xy","type":"widget","widget":"slider1","x":-40,"y":240}>ET

c) Set slider1 xy coordinates to (400,240):
Send: ST<{"cmd_code":"set_xy","type":"widget","widget":"slider1","x":400,"y":240}>ET

3.4 get_xy

Instruction sending:

Instruction	Instruction description	Remarks
get_xy	get coordinates of widget	The obtained x and y types are int and can be negative.

Instruction return:

Instruction	Instruction description	Return type	Remarks
0x0400	get widget coordinates	passive	

Data description:

Category	Data	Delivery type	Remarks
CMD	0x0400	passive	Get widget coordinates command
LEN	widget name length +8		
DATA	widget name + x + y		The last 8 bytes of the data section are the x and y coordinate data; the high-order byte are in front and the low-order byte are behind

For examples:
a) Get the coordinates of button1 as (273,85):
Send:ST<{"cmd_code":"get_xy","type":"widget","widget":"button1"}>ET
Response:ST<0x04 0x00 0x00 0x0F button1 0x00 0x00 0x01 0x11 0x00 0x00 0x00 0x55>ET
HEX:53 54 3C 04 00 00 0F 62 75 74 74 6F 6E 31 00 00 01 11 00 00 00 55 3E 45 54 08 07

b) Get the coordinates of button5 as (524,221):
Send:ST<{"cmd_code":"get_xy","type":"widget","widget":"button5"}>ET
Response:ST<0x04 0x00 0x00 0x0F button5 0x00 0x00 0x02 0x0C 0x00 0x00 0x00 0xDD>ET
HEX:53 54 3C 04 00 00 0F 62 75 74 74 6F 6E 35 00 00 02 0C 00 00 00 DD 3E 45 54 02 09

c) Get the coordinates of svg1 as (188,10):
Send:ST<{"cmd_code":"get_xy","type":"widget","widget":"svg1"}>ET
Response:ST<0x04 0x00 0x00 0x0C svg1 0x00 0x00 0x00 0xBC 0x00 0x00 0x00 0x0A>ET
HEX:53 54 3C 04 00 00 0C 73 76 67 31 00 00 00 BC 00 00 00 0A 3E 45 54 57 EC

3.5 get_wh

Instruction sending:

Instruction	Instruction description	Remarks
get_wh	get size of widget	The obtained w and h types are uint (width and height of widget by pixel)

Instruction return:

Instruction	Instruction description	Return type	Remarks
0x0401	get size of widget	passive	width and height of widget by pixel

Data description:

Category	Data	Delivery type	Remarks
CMD	0x0401	passive	Get widget size command
LEN	widget name length +8		
DATA	widget name + w + h		The last 8 bytes of the data section are w(width) and h(height); the high order byte are in front and the low order bit are behind

For example.
a) Get the size of button1 as 246x114:
Send:ST<{"cmd_code":"get_wh","type":"widget","widget":"button1"}>ET
Response:ST<0x04 0x01 0x00 0x0F button1 0x00 0x00 0x00 0xF6 0x00 0x00 0x00 0x72>ET
HEX:53 54 3C 04 01 00 0F 62 75 74 74 6F 6E 31 00 00 00 F6 00 00 00 72 3E 45 54 53 5F

b) Get svg1 size 119x132:
Send:ST<{"cmd_code":"get_wh","type":"widget","widget":"svg1"}>ET
Response:ST<0x04 0x01 0x00 0x0C svg1 0x00 0x00 0x00 0x77 0x00 0x00 0x00 0x84>ET
HEX:53 54 3C 04 01 00 0C 73 76 67 31 00 00 00 77 00 00 00 84 3E 45 54 60 DB

3.6 set_state

Instruction sending:

Instruction	Instruction description	Remarks
set_state	set widget state	The values can be "normal", "pressed", "disable"; see the widget state property for details

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Widget state, see the state property of each widget for details

For example:
a) Set the button1 widget to the pressed state:
Send: ST<{"cmd_code":"set_state","type":"widget","widget":"button1","state":"pressed"}>ET

3.7 set_border_type

Instruction sending:

Instruction	Instruction description	Remarks
set_border_type	set border type of widget	

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the borders of “normal” state will be modified according the instruction content
value	border type	uint	The values are as follows: 0: No border 1: Left border 2: Right border 4: Top border 8: Bottom border 15: All borders

For example:
a) Set the border type in the normal state of the b widget to full border:
Send:
ST<{"cmd_code":"set_border_type","type":"widget","widget":"b","state":"normal","value":15}>ET

b) Set the border type in the normal state of the b widget to left and right borders:
Send: ST<{"cmd_code":"set_border_type","type":"widget","widget":"b","state":"normal","value":3}>ET

c) Set the border type in the pressed state of the b widget to the top and bottom borders:

Send:
ST<{"cmd_code":"set_border_type","type":"widget","widget":"b","state":"pressed","value":12}>ET

d) Set the border type in the pressed state of the b widget to no border:
Send:
ST<{"cmd_code":"set_border_type","type":"widget","widget":"b","state":"pressed","value":0}>ET

e) Set the border type in the normal state of the b widget to full border:
Send: ST<{"cmd_code":"set_border_type","type":"widget","widget":"b","value":15}>ET

3.8 set_border_width

Instruction sending:

Instruction	Instruction description	Remarks
set_border_width	set border line width of widget	

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the borders width of "normal" state will be modified according the instruction content
width	border width	uint	Border line width

For example:
a) Set the line width to be 1 in the normal state of the b widget:
Send: ST<{"cmd_code":"set_border_width","type":"widget","widget":"b","state":"normal","width":1}>ET

b) Set the line width to be 2 in the pressed state of the b widget
Send: ST<{"cmd_code":"set_border_width","type":"widget","widget":"b",
"state":"pressed","width":2}>ET

c) Set the line width to be 5 in the normal state of the b widget:
Send: ST<{"cmd_code":"set_border_width","type":"widget","widget":"b","width":5}>ET

3.9 set_fg_image

Instruction sending:

Instruction	Instruction description	Remarks
set_fg_image	set foreground image of widget	If no state has been specified, the foreground image of "normal" state will be modified

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the image of “normal” state will be modified according the instruction content
fg_image	front image	text	Front image name, no need to specify suffix name, support png/jpg/bmp format

For example:

a) Set the front image in the pressed state of the pg1 widget to n0 (image file):

Send:

ST<{"cmd_code":"set_fg_image","type":"widget","widget":"pg1","state":"pressed","fg_image":"n0"}>ET

b) Set the front image in the normal state of the pg1 widget to n1(image file):

Send: ST<{"cmd_code":"set_fg_image","type":"widget","widget":"pg1","fg_image":"n1"}>ET

3.10 set_bg_image

Instruction sending:

Instruction	Instruction description	Remarks
set_bg_image	set background image of widget	

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the background image of “normal” state will be modified according the instruction content
bg_image	front image	text	Background image name, no need to specify suffix name, support png/jpg/bmp

For example:

a) Set the background image in the pressed state of the i1 widget to n0(image file):

Send:

ST<{"cmd_code":"set_bg_image","type":"widget","widget":"i1","state":"pressed","bg_image":"n0"}>ET

b) Set the background image in the normal state of the i1 widget to n1(image file):

Send: ST<{"cmd_code":"set_bg_image","type":"widget","widget":"i1","bg_image":"n1"}>ET

3.11 set_font

Instruction sending:

Instruction	**Instruction description**	**Remarks**
set_font	set font name (replace font of widget)	If no state has been specified, the font of “normal” state will be modified according to the content of sent instruction

Send data description:

Category	**Description**	**Type**	**Remarks**
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the font of “normal” state will be modified according the instruction content
font	font name	text	Font name, no suffix required, only ttf vector fonts are supported

For example:

a) Set the font in the normal state of the b1 widget to msyh:

Send: ST<{"cmd_code":"set_font","type":"widget","widget":"b1","state":"normal","font":"msyh"}>ET

b) Set the font in the pressed state of the b1 widget to default:

Send: ST<{"cmd_code":"set_font","type":"widget","widget":"b1","state":"pressed","font":"default"}>ET

c) Set the font in the normal state of the b1 widget to default:

Send: ST<{"cmd_code":"set_font","type":"widget","widget":"b1","font":"default"}>ET

3.12 set_font_size

Instruction sending:

Instruction	**Instruction description**	**Remarks**
set_font_size	set font size	If no state is specified, modify the font size in the normal state

Send data description:

Category	**Description**	**Type**	**Remarks**
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the font size of “normal” state will be modified according the instruction content
size	font size	uint	Font size

For example:
a) Set the font size to 18 in the normal state of the b1 widget:
Send: ST<{"cmd_code":"set_font_size","type":"widget","widget":"b1","state":"normal","size":18}>ET

b) Set the font size to 24 in the pressed state of the b1 widget:
Send: ST<{"cmd_code":"set_font_size","type":"widget","widget":"b1","state":"pressed","size":24}>ET

c) Set the font size to 18 in the normal state of the b1 widget:
Send: ST<{"cmd_code":"set_font_size","type":"widget","widget":"b1","size":18}>ET

3.13 set_text_align_h

Instruction sending:

Instruction	Instruction description	Remarks
set_text_align_h	set the horizontal alignment of the font	If no state is specified, modify the horizontal alignment of the font in the normal state

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the text horizontal alignment of “normal” state will be modified according the instruction content
align_h	font horizontal alignment	uint	The values are as follows: 0: no alignment 1: center alignment 2: text-align: left 3: text-align: right

For example:
a) Set the font to center alignment in the normal state of the b1 widget:
Send:
ST<{"cmd_code":"set_text_align_h","type":"widget","widget":"b1","state":"normal","align_h":1}>ET

b) Set the font to the text-align: left in the normal state of the b1 widget:
Send:
ST<{"cmd_code":"set_text_align_h","type":"widget","widget":"b1","state":"normal","align_h":2}>ET

c) Set the font to the text-align: right in the normal state of the b1 widget:
Send:
ST<{"cmd_code":"set_text_align_h","type":"widget","widget":"b1","align_h":3}>ET

3.14 set_text_align_v

Instruction sending:

Instruction	Instruction description	Remarks
set_text_align_v	set the vertical alignment of the font	If no state is specified, modify the vertical alignment of the font in the normal state

Send data description:

Category	Description	Type	Remarks
state	widget state	text	Please be noted that different widget has different avaliable states for seting use. If sent instruction has not specified particular state, the text vertical alignment of “normal” state will be modified according the instruction content
align_v	font vertical alignment	uint	The values are as follows: 0: no alignment 1: center alignment 2: top alignment 3: bottom alignment

For example:

a) Set the font to center alignment in the normal state of the b1 widget:

Send:

ST<{"cmd_code":"set_text_align_v","type":"widget","widget":"b1","state":"normal","align_v":1}>ET

b) Set the font to top-align in the normal state of the b1 widget:

Send:

ST<{"cmd_code":"set_text_align_v","type":"widget","widget":"b1","state":"normal","align_v":2}>ET

c) Set the font to bottom alignment in the normal state of the b1 widget:

Send: ST<{"cmd_code":"set_text_align_v","type":"widget","widget":"b1","align_v":3}>ET

3.15 set_color

Instruction sending:

Instruction	Instruction description	Remarks
set_color	set the showcolor relative to the widget color.	The color value is in ARGB format from high-order to low-order, R=0x11 G=0x22 B=0x33 A=0xFF, 0xFF332211 after combination, 4281541137 in decimal system, and it supports translucent effect.

Send data description:

Category	**Description**	**Type**	**Remarks**
color_object	color target object	text	The value of the color-related attributes contained in the current widget avaliable options, such as text_color/fg_color/bg_color, etc.; you may set the widget state before seting the color attribute, and set the color in different states, such as normal:bg_color. If no state has been specified, the color value of “normal” state will be modified accordingly;
color	color	uint	The color value is in ARGB format from high-order to low-order, for example: A=0xFF R=0x11 G=0x22 B=0x33, 0xFF112233 after the combination, 4279312947 in decimal system, and the transparent effect is supported;

For example:

a) Set the normal state bg_color of the switch widget to black:

Send: ST<{"cmd_code":"set_color","type":"widget","widget":"switch","color_object":"bg_color", "color":4278190080}>ET

b) Set the normal state text_color of the edit widget to blue:

Send: ST<{"cmd_code":"set_color","type":"widget","widget":"edit","color_object":"text_color", "color":4278190335}>ET

c) Set the dialog widget normal state bg_color to red:

Send: ST<{"cmd_code":"set_color","type":"widget","widget":"dialog","color_object":"bg_color", "color":4294901760}>ET

d) Set the switch widget dialog state bg_color to yellow:

Send: ST<{"cmd_code":"set_color","type":"widget","widget":"dialog","color_object":"normal:bg_color", "color":4294967040}>ET

e) Set the dialog widget pressed state bg_color to green:

Send:

ST<{"cmd_code":"set_color","type":"widget","widget":"dialog","color_object":"pressed:bg_color", "color":4278255360}>ET

f) Set the text color of the normal state of edit1 to edit5 widgets in batch:

Send:

ST<{"cmd_code":"set_color","type":"widget","widget":"edit1_5","color_object":"normal:text_color", "color":[4278190335,4278190080,4294901760,4294967040,4278255360] }>ET

3.16 take_snapshot

Instruction sending:

Instruction	Instruction description	Remarks
take_snapshot	screenshots/snapshots	1. The screenshot function can only screenshoot all contents of the whole window. Making screenshots for particular region or sections of the window are not avaliable at the moment. 2. The screenshot images will be saved in the “snapshot” folder which inside of “resource” folder ;

For example:

a) Screenshot home_page page:

Send: ST<{"cmd_code":"take_snapshot","type":"widget","widget":"home_page"}>ET

b) Screenshot led_demo interface:

Send: ST<{"cmd_code":"take_snapshot","type":"widget","widget":"led_demo"}>ET

4. Widget Instruction

4.1 window

1. Instruction sending:

Instruction	Instruction description	Remarks
open_win	open any window	Windows which running at the background can also be opened with this instruction
close_win	close any window	The data of targeting window will not going to be cached after closing by this instruction. Using this instruction should be cautious
back_win	return to upper window	Close the targeting window without caching the data of targeting window
back_win_to	return to any upper-level window	Other opened windows will remain running at the background
back_home	return to the main window	Previously opened window will not be closed and other windows will remain running at the background
get_displayed_window	get the name of currently displayed window	Get the name of main window currently displayed in the foreground (except for popup/keyboard)

2. Instruction returns:

Return instruction	Description	Return type	Remarks
0x2001	Get the return instruction of the window	passive	Return instruction after sending get_displayed_window instruction
0x2007	Return instruction for open windows	active	Return instruction after sending opening window instruction or opening window event by pressed button happened
0x2008	Return instruction for closing the window	active	Return instruction after sending closing window instruction or closing window event by pressed button happened

Return data description:

Category	Data	Description	Remarks
CMD	0x2001	Window return instruction	MCU acquires the name of the window after sending get_displayed_window instruction
LEN	Length of "widget name"	data length	The length of the window name
DATA	"widget name"	data content	The window name

Category	Data	Description	Remarks
CMD	0x2007	Window return instruction	Active instruction when opening/returning windows
LEN	Length of "widget name"	data length	The length of the window name
DATA	"widget name"	data content	The window name

Category	Data	Description	Remarks
CMD	0x2008	Window return instruction	Active instruction when closing windows
LEN	Length of "widget name"	data length	The length of the window name
DATA	"widget name"	data content	The window name

3. For example:
a) Open the label_value window:
Send: ST<{"cmd_code":"open_win","type":"window","widget":"label_value"}>ET
Response: ST<0x20 0x07 0x00 0x0B label_value>ET
HEX:53 54 3C 20 07 00 0B 6C 61 62 65 6C 5F 76 61 6C 75 65 3E 45 54 B9 DF

b) Close the label_value window:
Send: ST<{"cmd_code":"close_win","type":"window","widget":"label_value"}>ET
Response: ST<0x20 0x08 0x00 0x0B label_value>ET
HEX:53 54 3C 20 08 00 0B 6C 61 62 65 6C 5F 76 61 6C 75 65 3E 45 54 4A EA

c) Return to the upper window:
Send: ST<{"cmd_code":"back_win","type":"window"}>ET
Response: ST<0x20 0x08 0x00 0x0B label_value>ET
HEX:53 54 3C 20 08 00 0B 6C 61 62 65 6C 5F 76 61 6C 75 65 3E 45 54 4A EA

d) Return to the upper window named label_value/home_page, close all windows above this window, generally applicable to multi-level windows:
Send: ST<{"cmd_code":"back_win_to","type":"window","widget":"label_value"}>ET
Send: ST<{"cmd_code":"back_win_to","type":"window","widget":"home_page"}>ET

e) Return to the main window:
Send: ST<{"cmd_code":"back_home","type":"window"}>ET

f) Get the currently displayed main window(home_page):
Send: ST<{"cmd_code":"get_displayed_window","type":"window"}>ET
Response: ST<0x20 0x01 0x00 0x09 home_page>ET
HEX:53 54 3C 20 01 00 09 68 6F 6D 65 5F 70 61 67 65 3E 45 54 60 1A

Special note: the main window home_page cannot be closed;

4.2 label

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the text showed by the label	
set_value	set the value showed by the label	
get_text	get the text showed by the label	
get_value	get the value showed by the label (float)	

Send data description:

Category	Description	Type	Remarks
text	text	text	Set the text to show
value	value	int/float	Set the value to show
format	number format	text	Value:%d,%02d,%03d,%04d,%05d,%06d,%f,%.1f,%.2f,%.3f,%.4f,%.5f,%.6f

2. Instruction returns:

Return instruction	Description	Return type	Remarks
0x1060	the label text is sent passively	passive	The MCU will only acquire this return data after sending "get_text" instruction, lable widget will not send this return data actively.
0x1062	label value delivery (float type)	active/passive	Once button widget has been binded with Lable widget and Label widget value changed by the action of button widget or "get_value"instruction has been sent

Return data description:

Category	Data	Description	Remarks
CMD	0x1060	label text delivery	The MCU will acquire this return data after sending "get_text" instruction
LEN	"widget name" + length of text	data length	
DATA	"widget name": text	data content	The data length should not exceed 1,024 bytes

Category	Data	Description	Remarks
CMD	0x1062	label value delivery	Once button widget has been binded with Lable widget and Label widget value changed by the action of button widget or "get_value"instruction has been sent
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	data content	The value is the last four bytes of the data part, float conforms to the IEEE 754 specification

3. For example:
Set text:

Hello Stone

1234567890

ST<{"cmd_code":"set_text","type":"label","widget":"label","text":"Hello Stone"}>ET
ST<{"cmd_code":"set_text","type":"label","widget":"label","text":"1234567890"}>ET

Set the text of the widget label2 to label11:
ST<{"cmd_code": "set_text", "type": "label", "widget": "label1_11", "text":["2022-07-11\n10:30", "10", "12", "800", "15", "12.8", "48.2", "52.6", "18.6", "13.5", "16.3"]}>ET

Special note: When using array to set label text in batch, the naming rule of widget is: "ASCII letter" + "start index"+ "_" + "end index"; the amount of elements in the text array must correspond to the widget name; for example, label1_11 means that the widget name is label1 to label11, total 11 label widgets;
text array ["2022-07-11\n10:30", "10", "12", "800", "15", "12.8", "48.2", "52.6", "18.6", "13.5", "16.3"] corresponds to the text content;

Set value:

1.23

ST<{"cmd_code":"set_value","type":"label","widget":"label2","value":5}>ET
ST<{"cmd_code":"set_value","type":"label","widget":"label2","value":5,"format":"%02d"}>ET
ST<{"cmd_code":"set_value","type":"label","widget":"label2","value":1.23}>ET
ST<{"cmd_code":"set_value","type":"label","widget":"label2","value":1.23,"format":"%.3f"}>ET

Set the value of the widget label2 to label11:
ST<{"cmd_code": "set_value", "type": "label", "widget": "label2_11", "value":[10,12,800,15,12.8,48.2,52.6,18.6,13.5,16.3]}>ET
ST<{"cmd_code": "set_value", "type": "label", "widget": "label2_11", "value":[11,12,800,15,12.8,48.2,52.6,18.6,13.5,16.3],"format":"%.1f"}>ET

Special note: When using array to set label value in batch, the naming rule of widget is: "ASCII letter" + "start index"+ "_" + "end index"; the amount of elements in the value array must correspond to the widget name; for example, label2_11 means that the widget name is label2 to label11, total 10 label widgets; value array[11,12,800,15,12.8,48.2,52.6,18.6,13.5,16.3]corresponds to the value content;

Get text:
a) Get the text content of the label widget as Stone:
Send: ST<{"cmd_code":"get_text","type":"label","widget":"label"}>ET
Response: ST<0x10 0x60 0x00 0x0D "label":Stone>ET
HEX: 53 54 3C 10 60 00 0D 22 6C 61 62 65 6C 22 3A 53 74 6F 6E 65 3E 45 54 00 CE

b) Get the text content of the label widget as 12345:
Send: ST<{"cmd_code":"get_text","type":"label","widget":"label"}>ET
Response: ST<0x10 0x60 0x00 0x0D "label":12345>ET
HEX:53 54 3C 10 60 00 0D 22 6C 61 62 65 6C 22 3A 31 32 33 34 35 3E 45 54 A4 2B

Get value:
a) Get the value of the lable widget as 1.26:
Send: ST<{"cmd_code":"get_value","type":"label","widget":"label"}>ET
Response: ST<0x10 0x62 0x00 0x09 label 0x3F 0xA1 0x47 0xAE>ET
HEX:53 54 3C 10 62 00 09 6C 61 62 65 6C 3F A1 47 AE 3E 45 54 6C 8B

b) Get the value of the lable widget as 8:
Send: ST<{"cmd_code":"get_value","type":"label","widget":"label"}>ET
Response: ST<0x10 0x62 0x00 0x0A label 0x41 0x00 0x00 0x00>ET
HEX:53 54 3C 10 62 00 0A 6C 61 62 65 6C 32 41 00 00 00 3E 45 54 C2 99

4.3 edit

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the text content showed by edit	
set_value	set the value showed by edit	
get_text	get the text content showed by edit	
get_value	get the value showed by edit (int/float)	

Send data description:

Category	Description	Type	Remarks
text	text	text	Set the text for displaying
value	value	int/float	Set the value for displaying
format	number format	text	Value:%d,%02d,%03d,%04d,%05d,%06d,%f,%.1f,%.2f,%.3f,%.4f,%.5f,%.6f

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1070	edit text delivery	active/passive	Actively or passively sent from HMI to MCU , it can be actively sent after the edit widget data has changed, or it can be passively sent after conducting “get_text” instruction
0x1071	edit value delivery	passive	Int type
0x1072	edit value delivery	passive	floating point type

Return data description:

Category	Data	Description	Remarks
CMD	0x1070	edit text delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the edit widget data has changed, or it can be passively sent by HMI after conducting “get_text” instruction by MCU
LEN	"widget name" + length of text	data length	
DATA	"widget name": text	data content	The data length should not exceed 1,024 bytes

Category	Data	Description	Remarks
CMD	0x1071	edit value delivery	Once button widget has been binded with Edit widget and Edit widget value changed by the action of button widget or “get_value”instruction has been sent by MCU
LEN	"widget name" + the length of the	data length	

	value		
DATA	widget name + value	data content	The value will be the last four bytes of the data, int type data

Category	Data	Description	Remarks
CMD	0x1072	edit value delivery	Once button widget has been binded with Edit widget and Edit widget value changed by the action of button widget or “get_value”instruction has been sent by MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	data content	The value will be the last four bytes of the data string, float type, IEEE 754 specification

3. For example:
Set text:
```
ST<{"cmd_code":"set_text","type":"edit","widget":"edit","text":"Hello Stone"}>ET
ST<{"cmd_code":"set_text","type":"edit","widget":"edit","text":"1234567890"}>ET
```

Set the text of edit1 to edit19 in batch:
```
ST<{"cmd_code":"set_text","type":"edit","widget":"edit1_19","text":["10","12","800","15","12.8","48.2",
"52.6","18.6","13.5","16.3","10","12","800","15","12.8","48.2","52.6","18.6","13.5"]}>ET
```

Special Note: Please refer to the label widget for the description of setting widget values/text in batch;

Set value:
a) The edit data type is int and it is showed as 3:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit","value":3}>ET
```

b) The edit data type is int and it is showed as 03:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit","value":3,"format":"%02d"}>ET
```

c) The edit data type is float and it is showed as 2.500000:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit1","value":2.5}>ET
```

d) The edit data type is float, which shows 2.50:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit1","value":2.5,"format":"%.2f"}>ET
```

e) The edit data type is float, which sets the values of edit1 to edit6 widgets in batch:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit1_6","value":[10,12,800,15,12.8]}>ET
```

f) The edit data type is float, set the values of the edit1 to edit6 widgets in batch, and the display format is %.2f:
```
ST<{"cmd_code":"set_value","type":"edit","widget":"edit1_6","value":[10,12,800,15,12.8],"format":"%.2
f"}>ET
```

Special Note: Please refer to the label widget for the description of setting widget values/text in batch;

Get text:
a) Get edit text data: abcdefg:
Send: ST<{"cmd_code":"get_text","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x70 0x00 0x0E "edit":abcdefg>ET
HEX:53 54 3C 10 70 00 0E 22 65 64 69 74 22 3A 61 62 63 64 65 66 67 3E 45 54 CA EB

b) Get edit text data: StoneDesigner:
Send: ST<{"cmd_code":"get_text","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x70 0x00 0x15 "edit":StoneDesigner>ET
HEX: 53 54 3C 10 70 00 15 22 65 64 69 74 22 3A 53 74 6F 6E 65 44 65 73 69 67 6E 65 72 3E 45 54 04 32

Get value:
a) edit int type data delivery, data: 123:
Send: ST<{"cmd_code":"get_value","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x71 0x00 0x08 edit 0x00 0x00 0x00 0x7B>ET
HEX: 53 54 3C 10 71 00 08 65 64 69 74 00 00 00 7B 3E 45 54 B6 5C
b) Edit int type data delivery, data: -123:
Send: ST<{"cmd_code":"get_value","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x71 0x00 0x08 edit 0xFF 0xFF 0xFF 0x85>ET
HEX:53 54 3C 10 71 00 08 65 64 69 74 FF FF FF 85 3E 45 54 4A 62

c) edit float type data delivery, data: 123.456:
Send: ST<{"cmd_code":"get_value","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x72 0x00 0x08 edit 0x42 0xF6 0xE9 0x79>ET
HEX:53 54 3C 10 72 00 08 65 64 69 74 42 F6 E9 79 3E 45 54 48 75

d) Edit float type data delivery, data: -123.456:
Send: ST<{"cmd_code":"get_value","type":"edit","widget":"edit"}>ET
Response: ST<0x10 0x72 0x00 0x08 edit 0xC2 0xF6 0xE9 0x79>ET
HEX:53 54 3C 10 72 00 08 65 64 69 74 C2 F6 E9 79 3E 45 54 80 F4

4.4 spin_box

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the showed text content	
set_value	set the showed value	
get_text	get the showed text content	
get_value	get the showed value (int/float)	

Send data description:

Category	Description	Type	Remarks
text	text	text	Set the text for displaying
value	value	int/float	Set the value for displaying
format	number format	text	Value:%d,%02d,%03d,%04d,%05d,%06d,%f,%.1f,%.2f,%.3f,%.4f,%.5f,%.6f

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10A0	spin_box text delivery	active/passive	It can be actively issued after the spin_box data is changed, or it can be actively obtained using get_text (generally not used)
0x10A1	spin_box value delivery	passive	Int type
0x10A2	spin_box value delivery	passive	Float type

Return data description:

Category	Data	Description	Remarks
CMD	0x10A0	spin_box text delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the spin_box widget data has changed, or it can be passively sent from HMI after conducting “get_text” instruction by MCU (not commonly used)
LEN	"widget name" + length of text	text length	
DATA	"widget name": text	text content	The data length should not exceed 1,024 bytes

Category	Data	Description	Remarks
CMD	0x10A1	spin_box value delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the spin_box widget data has changed, or it can be passively sent from HMI after conducting “get_text” instruction by MCU (not commonly used)
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	data content	The value is the last four bytes of the data part, int type

Category	Data	Description	Remarks
CMD	0x10A2	spin_box value delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the spin_box widget data has changed, or it can be passively sent from HMI after conducting “get_text” instruction by MCU (not commonly used)
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	data content	The value is the last four bytes of the data part, float type data, according to IEEE 754 standard

3. For example:
Set text:
ST<{"cmd_code":"set_text","type":"spin_box","widget":"spin_box1","text":"Stone"}>ET

Set value:
a) The data type is int and it shows 08:
ST<{"cmd_code":"set_value","type":"spin_box","widget":"spin_box1","value":8,"format":"%02d"}>ET
b) The data type is floa and it shows 7.30:
ST<{"cmd_code":"set_value","type":"spin_box","widget":"spin_box1","value":7.3,"format":"%.2f"}>ET

c) The data type is int, which is showed as 6:
ST<{"cmd_code":"set_value","type":"spin_box","widget":"spin_box1","value":6}>ET

Get text:
a) Text data delivery, text content: Stone:
Send: ST<{"cmd_code":"get_text","type":"spin_box","widget":"spin_box"}>ET
Response: ST<0x10 0xA0 0x00 0x10 "spin_box":Stone>ET
HEX:53 54 3C 10 A0 00 10 22 73 70 69 6E 5F 62 6F 78 22 3A 53 74 6F 6E 65 3E 45 54 19 3C

Get value:
a) Int type data delivery, and the data is 3:
Send: ST<{"cmd_code":"get_value","type":"spin_box","widget":"spin_box"}>ET
Response: ST<0x10 0xA1 0x00 0x0C spin_box 0x00 0x00 0x00 0x03>ET
HEX:53 54 3C 10 A1 00 0C 73 70 69 6E 5F 62 6F 78 00 00 00 03 3E 45 54 8A 1A

b) Int type data delivery, and the data is 9:
Send: ST<{"cmd_code":"get_value","type":"spin_box","widget":"spin_box"}>ET
Response: ST<0x10 0xA1 0x00 0x0C spin_box 0x00 0x00 0x00 0x09>ET
HEX:53 54 3C 10 A1 00 0C 73 70 69 6E 5F 62 6F 78 00 00 00 09 3E 45 54 52 19

c) Float type data delivery, and the data is 1.6:
Send: ST<{"cmd_code":"get_value","type":"spin_box","widget":"spin_box"}>ET
Response: ST<0x10 0xA2 0x00 0x0C spin_box 0x3F 0xCC 0xCC 0xCD>ET
HEX:53 54 3C 10 A2 00 0C 73 70 69 6E 5F 62 6F 78 3F CC CC CD 3E 45 54 F9 1A

d) Float type data delivery, and the data is 1.23:
Send: ST<{"cmd_code":"get_value","type":"spin_box","widget":"spin_box"}>ET
Response: ST<0x10 0xA2 0x00 0x0C spin_box 0x3F 0x9D 0x70 0xA4>ET
HEX:53 54 3C 10 A2 00 0C 73 70 69 6E 5F 62 6F 78 3F 9D 70 A4 3E 45 54 3F 5B

4.5 combo_box_ex

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the showed text content	
set_value	set the showed value	
set_selected	set current option	
get_text	get the showed text content	
get_value	get the showed value (int/float data)	
get_selected	get current option	

Send data description:

Category	Description	Type	Remarks
text	text	text	set/get the text for displaying
value	value	int/float	set/get the value for displaying
selected	selective	uint	set/get current option
format	number format	text	value:%d,%02d,%03d,%04d,%05d,%06d,%f,%.1f,%.2f,%.3f,%.4f,%.5f,%.6f

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10B0	combo_box_ex text sent from HMI to MCU	active/passive	Actively or passively sent from HMI to MCU , it can be actively sent after the combo_box_ex widget data has changed, or it can be passively sent from HMI after sending “get_text” instruction by MCU
0x10B1	combo_box_ex value sent from HMI to MCU	passive	Int type data
0x10B2	combo_box_ex value sent from HMI to MCU	passive	Float type data
0x10B8	combo_box_ex serial number sent from HMI to MCU	passive	Int type data, MCU may send “get_selected” instruction to get the serial number, starting from 0

Return data description:

Category	Data	Description	Remarks
CMD	0x10B0	combo_box_ex text delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the combo_box_ex widget data has changed, or it can be passively sent from HMI after sending “get_text” instruction by MCU
LEN	widget name" + length of text	text length	
DATA	widget name: Text	text content	

Category	Data	Description	Remarks
CMD	0x10B1	combo_box_ex value delivery	
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	data content	The value is the last four bytes of the data part, int type data

Category	Data	Description	Remarks
CMD	0x10B2	combo_box_ex value delivery	
LEN	"Widget name" + the length of the value	data length	
DATA	Widget name + value	data content	The value is the last four bytes of the data part, float type, according to IEEE 754 standard

Category	Data	Description	Remarks
CMD	0x10B8	combo_box_ex serial number delivery	
LEN	"widget name" + the length of the value	data length	
DATA	widget name + serial number value	data content	Int type data, MCU may send “get_selected” instruction to get the serial number, starting from 0

3. For example:

Set text:

ST<{"cmd_code":"set_text","type":"combo_box_ex","widget":"cbx1","text":"Stone"}>ET

Set value:

a) The data type is int and it shows 08:

ST<{"cmd_code":"set_value","type":"combo_box_ex","widget":"cbx1","value":8,"format":"%02d"}>ET

b) The data type is float and it shows 7.30:

ST<{"cmd_code":"set_value","type":"combo_box_ex","widget":"cbx1","value":7.3,"format":"%.2f"}>ET

c) The data type is int, which is showed as 6:

ST<{"cmd_code":"set_value","type":"combo_box_ex","widget":"cbx1","value":6}>ET

Set the current option:

ST<{"cmd_code":"set_selected","type":"combo_box_ex","widget":"cbx1","selected_index":2}>ET

Get text:
a) The text data is red:
Send: ST<{"cmd_code":"get_text","type":"combo_box_ex","widget":"combo_box_ex"}>ET
Response: ST<0x10 0xB0 0x00 0x02 "combo_box_ex":red>ET
HEX:53 54 3C 10 B0 00 12 22 63 6F 6D 62 6F 5F 62 6F 78 5F 65 78 22 3A 72 65 64 3E 45 54 D2 96

Get value:
a) The int type data is 123:
Send: ST<{"cmd_code":"get_value","type":"combo_box_ex","widget":"combo_box_ex"}>ET
Response: ST<0x10 0xB1 0x00 0x10 combo_box_ex 0x00 0x00 0x00 0x7B>ET
HEX:53 54 3C 10 B1 00 10 63 6F 6D 62 6F 5F 62 6F 78 5F 65 78 00 00 00 7B 3E 45 54 2C B2

b) The float type data is 1.23:
Send: ST<{"cmd_code":"get_value","type":"combo_box_ex","widget":"combo_box_ex"}>ET
Response: ST<0x10 0xB2 0x00 0x10 combo_box_ex 0x3F 0x9D 0x70 0xA4>ET
HEX:53 54 3C 10 B2 00 10 63 6F 6D 62 6F 5F 62 6F 78 5F 65 78 3F 9D 70 A4 3E 45 54 68 68
Get the current option:
a) The current option number of combo_box_ex is 4, which is the fifth selected item:
Send: ST<{"cmd_code":"get_selected","type":"combo_box_ex","widget":"combo_box_ex1"}>ET
Response: ST<0x10 0xB8 0x00 0x12 combo_box_ex 0x00 0x00 0x00 0x04>ET
HEX:53 54 3C 10 B8 00 10 63 6F 6D 62 6F 5F 62 6F 78 5F 65 78 00 00 00 04 3E 45 54 92 F2

4.6 mledit

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the showed text content	
get_text	get the showed text content	

Send data description:

Category	Description	Type	Remarks
text	text	text	set/get the text for displaying

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10C0	mledit text delivery	active/passive	Actively or passively sent from HMI to MCU , it can be actively sent after the mledit widget data has changed, or it can be passively sent from HMI after sending “get_text” instruction by MCU

Return data description:

Category	Data	Description	Remarks
CMD	0x10C0	mledit text delivery	Actively or passively sent from HMI to MCU , it can be actively sent after the mledit widget data has changed, or it can be passively sent from HMI after sending “get_text” instruction by MCU
LEN	"widget name" + text	data length	
DATA	widget name: text	text content	The data length should not exceed 1,024 bytes (the text content means the contect after “widget name:”)

3. For example:

Set text:

ST<{"cmd_code":"set_text","type":"mledit","widget":"mledit","text":"Stone"}>ET

Get text:

a) Get text data: Stone

Send: ST<{"cmd_code":"get_text","type":"mledit","widget":"mledit"}>ET

Response: ST<0x10 0xC0 0x00 0x02 "mledit":Stone>ET

HEX:53 54 3C 10 C0 00 0E 22 6D 6C 65 64 69 74 22 3A 53 74 6F 6E 65 3E 45 54 6F 92

4.7 progress_bar

1. Instruction sending:

Instruction	Instruction description	Remarks
set_max	set the progress bar maximum value	
show_text	set the progress bar for showing text content or not	
set_value	set the progress bar value	
get_value	get the progress bar value	
get_percent	get percentage of progress bar	

Send data description:

Category	Description	Type	Remarks
text	text	text	set the progress bar for showing text content or not
max	maximum value	uint	set the progress bar maximum value
value	value	uint	set the progress bar value / get the progress bar value
percent	percentage	uint	get percentage of progress bar

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1050	progress bar value sent from HMI to MCU	active/passive	MCU is able to send the “get_value” instruction for acquiring the value
0x1051	progress bar percentage	passive	MCU is able to send the “get_percentage” instruction for acquiring the percentage

Return data description:

Category	Data	Description	Remarks
CMD	0x1050	progress_bar value delivery	Active/passive delivery, MCU is able to send “get_value” instruction for acquring the data
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	text content	Float type data, according to IEEE 754 standard

Category	Data	Description	Remarks
CMD	0x1051	progress bar percentage	Passively sent from HMI to MCU once MCU sent “get_percent” instruction for acquiring relevant data
LEN	"widget name" + length in percentage	data length	
DATA	widget name + percentage	text content	Int type data

3. For example:
Set the progress bar maximum value:
ST<{"cmd_code":"set_max","type":"progress_bar","widget":"progress_bar","max":100}>ET

Set whether the progress bar shows text:
ST<{"cmd_code":"set_show_text","type":"progress_bar","widget":"progress_bar","show_text":true}>E
T
ST<{"cmd_code":"set_show_text","type":"progress_bar","widget":"progress_bar","show_text":false}>E
T

Set the progress bar value:
ST<{"cmd_code":"set_value","type":"progress_bar","widget":"progress_bar","value":40}>ET

Set the value of progress bar in batch:
ST<{"cmd_code":"set_value","type":"progress_bar","widget":"progress_bar1_35","value":[10,12,80,15,
12,10,12,10,10,12,8,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,8,15,12,10,12,10,10,12,80]}>ET

Special Note: Please refer to the label widget for the description of setting widget values/text in batch;

Get the progress bar value:
a) Progress bar data changed, data 54.978615:
Send: ST<{"cmd_code":"get_value","type":"progress_bar","widget":"progress_bar"}>ET
Response: ST<0x10 0x50 0x00 0x10 progress_bar 0x42 0x5B 0xEA 0x1A>ET
HEX:53 54 3C 10 50 00 10 70 72 6F 67 72 65 73 73 5F 62 61 72 42 5B EA 1A 3E 45 54 BF 09
q
b) Progress bar data changed, data: 54.999928:
Send: ST<{"cmd_code":"get_value","type":"progress_bar","widget":"progress_bar"}>ET
Response: ST<0x10 0x50 0x00 0x10 progress_bar 0x42 0x5B 0xFF 0xED>ET
HEX:53 54 3C 10 50 00 10 70 72 6F 67 72 65 73 73 5F 62 61 72 42 5B FF ED 3E 45 54 08 36

c) Progress bar data changed, data: 55.000000:
Send: ST<{"cmd_code":"get_value","type":"progress_bar","widget":"progress_bar"}>ET
Response: ST<0x10 0x50 0x00 0x10 progress_bar 0x42 0x5C 0x00 0x00>ET
HEX:53 54 3C 10 50 00 10 70 72 6F 67 72 65 73 73 5F 62 61 72 42 5C 00 00 3E 45 54 C7 16

Get the progress bar percentage:
a) Progress bar percentage: 40%:
Send: ST<{"cmd_code":"get_percent","type":"progress_bar","widget":"progress_bar"}>ET
Response: ST<0x10 0x51 0x00 0x10 progress_bar 0x00 0x00 0x00 0x28>ET
HEX:53 54 3C 10 51 00 10 70 72 6F 67 72 65 73 73 5F 62 61 72 00 00 00 28 3E 45 54 33 A1

4.8 progress_circle

1. Instruction sending:

Instruction	Instruction description	Remarks
set_max	set the progress circle maximum value	
show_text	set whether the progress circle shows text	
set_value	set the progress circle value	
get_value	get the progress circle value	
get_percent	get progress circle percentage	

Send data description:

Category	Description	Type	Remarks
text	text	text	set the progress circle for showing text or not
max	maximum value	uint	set the progress circle maximum value
value	value	uint	set the progress circle value/get the progress bar value
percent	percentage	uint	get progress circle percentage

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10E0	progress circle value	passive	Key value (the last four bytes of data part): value: 0x42400000 The current progress circle value is 48.000000 (float type data, according to IEEE 754 standard)
0x10E1	progress circle percentage	passive	Key value (the last four bytes of data part): percent: 0x00000028, the current progress circle percentage is 40% (int type data)

Return data description:

Category	Data	Description	Remarks
CMD	0x10E0	progress circle value	Passively sending data from HMI to MCU (responding to sent data from MCU to HMI)
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Float type data, according to IEEE 754 standard

Category	Data	Description	Remarks
CMD	0x10E1	progress_circle percentage	Passively sending data from HMI to MCU (responding to sent data from MCU to HMI)
LEN	"widget name" + length in percentage	"widget name" + percentage	
DATA	widget name + percentage	percentage value	Int type data

3. For example

Set the progress bar maximum value:

ST<{"cmd_code":"set_max","type":"progress_circle","widget":"pg_circle1","max":100}>ET

Set whether the progress bar shows text:

ST<{"cmd_code":"set_show_text","type":"progress_circle","widget":"pg_circle1","show_text":true}>ET

ST<{"cmd_code":"set_show_text","type":"progress_circle","widget":"pg_circle1","show_text":false}>ET

Set the progress bar value to 40%:

ST<{"cmd_code":"set_value","type":"progress_circle","widget":"progress_circle1","value":40}>ET

Acquiring the progress bar value:

a) Acquiring the value of progress_circle1 once the value of widget is 56.0:

Send:ST<{"cmd_code":"get_value","type":"progress_circle","widget":"progress_circle1"}>ET

Response:ST<0x10 0xE0 0x00 0x14 progress_circle1 0x42 0x60 0x00 0x00>ET

HEX:53 54 3C 10 E0 00 14 70 72 6F 67 72 65 73 73 5F 63 69 72 63 6C 65 31 42 60 00 00 3E 45 54 01 EE

b) Acquiring the value of progress_circle1 once the value of widget is 54.999928:

Send:ST<{"cmd_code":"get_value","type":"progress_circle","widget":"progress_circle1"}>ET

Response:ST<0x10 0xE0 0x00 0x14 progress_circle1 0x42 0x5B 0xFF 0xED>ET

HEX:53 54 3C 10 E0 00 14 70 72 6F 67 72 65 73 73 5F 63 69 72 63 6C 65 31 42 5B FF ED 3E 45 54 F2 CB

c) Acquiring the value of progress_circle1 once the value of widget is 55:

Send:ST<{"cmd_code":"get_value","type":"progress_circle","widget":"progress_circle1"}>ET

Response:ST<0x10 0xE0 0x00 0x14 progress_circle1 0x42 0x5C 0x00 0x00>ET

HEX:53 54 3C 10 E0 00 14 70 72 6F 67 72 65 73 73 5F 63 69 72 63 6C 65 31 42 5C 00 00 3E 45 54 3D EB

Acquiring the percentage of progress bar:

a) Programmed for acquiring the value of progress_circle1 once the value just reaching 40% :

Send:ST<{"cmd_code":"get_percent","type":"progress_circle","widget":"progress_circle1"}>ET

Respons:ST<0x10 0xE1 0x00 0x14 progress_circle1 0x00 0x00 0x00 0x28>ET

HEX:53 54 3C 10 E1 00 14 70 72 6F 67 72 65 73 73 5F 63 69 72 63 6C 65 31 00 00 00 28 3E 45 54 FA 74

4.9 hscroll_label

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the text showed by hscroll_label	
set_loop	set enable/disable the hscroll_label to loop playback	
set_yoyo	set enable/disable the hscroll_label to yoyo	
set_direction	set the direction of hscroll_label scrolling	
set_lull	set hscroll_label lull	Interval for showing one time label text content
set_duration	set the duration for hscroll_label to scroll once	
get_text	get the text showed by hscroll_label	

Send data description:

Category	Description	Type	Remarks
text	text	text	Set the text showed by hscroll_label
loop	loop	bool	Set enable/disable the hscroll_label to loop playback
yoyo	yoyo	bool	Set enable/disable the hscroll_label to yoyo
direction	direction	bool	Set the direction of hscroll_label scrolling
lull	lull	uint	Set hscroll_label lull
duration	duration	uint	Set the duration for hscroll_label to scroll once

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1100	text returns	passive	

Return data description:

Category	Data	Description	Remarks
CMD	0x1100	text returns	
LEN	"widget name" + length of text	data length	
DATA	widget name + text	text content	Data format: text: "widget name": text content

3. For example:

Set text:

```
ST<{"cmd_code":"set_text","type":"hscroll_label","widget":"hscroll_label1","text":"Hello Stone"}>ET
ST<{"cmd_code":"set_text","type":"hscroll_label","widget":"hscroll_label1","text":"1234567890"}>ET
ST<{"cmd_code":"set_text","type":"hscroll_label","widget":"hscroll_label",
"text":"http://www.stoneitech.com http://www.stoneitech.com"}>ET
```

Set enable/disable to loop playback:

```
ST<{"cmd_code":"set_loop","type":"hscroll_label","widget":"hscroll_label1","loop":true}>ET
ST<{"cmd_code":"set_loop","type":"hscroll_label","widget":"hscroll_label1","loop":false}>ET
```

Set enable/disable to yoyo:

```
ST<{"cmd_code":"set_yoyo","type":"hscroll_label","widget":"hscroll_label1","yoyo":true}>ET
ST<{"cmd_code":"set_yoyo","type":"hscroll_label","widget":"hscroll_label1","yoyo":false}>ET
```

Set the direction:

a) Set hscroll_lable1 to scroll from left to right:

```
ST<{"cmd_code":"set_direction","type":"hscroll_label","widget":"hscroll_label1","direction":true}>ET
```

b) Set hscroll_lable1 to scroll from right to left:

```
ST<{"cmd_code":"set_direction","type":"hscroll_label","widget":"hscroll_label1","direction":false}>ET
```

Set the scroll lull:

```
ST<{"cmd_code":"set_lull","type":"hscroll_label","widget":"hscroll_label1","lull":2000}>ET
ST<{"cmd_code":"set_lull","type":"hscroll_label","widget":"hscroll_label1","lull":5000}>ET
```

Set the duration required to scroll once:

```
ST<{"cmd_code":"set_duration","type":"hscroll_label","widget":"hscroll_label1","duration":2000}>ET
ST<{"cmd_code":"set_duration","type":"hscroll_label","widget":"hscroll_label1","duration":5000}>ET
```

Get text:

a) Get the text of hscroll_lable1: http://www.stoneitech.com http://www.stoneitech.com:

```
Send: ST<{"cmd_code":"get_text","type":"hscroll_label","widget":"hscroll_label"}>ET
Response: ST<0x11 0x00 0x00 0x43 "hscroll_label":http://www.stoneitech.com
http://www.stoneitech.com>ET
HEX:53 54 3C 11 00 00 43 22 68 73 63 72 6F 6C 6C 5F 6C 61 62 65 6C 22 3A 68 74 74 70 3A 2F 2F
77 77 77 2E 73 74 6F 6E 65 69 74 65 63 68 2E 63 6F 6D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 74
6F 6E 65 69 74 65 63 68 2E 63 6F 6D 3E 45 54 B7 71
```

4.10 text_selector

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set the text of the text selector	The function of “set_text” instruction literally is request the widget switching to one of the targeting text option which been defined or input upon the widget already
set_value	set the value of the text selector	The function of “set_value” instruction literally is request the widget switching to one of the targeting value option which been defined or input upon the widget already
set_selected	set the current option of the text selector	switch to the targeting serial number option ;
get_text	get the text of the text selector	Get the text of the current option;
get_value	get the value of the text selector	Get the value of the current option;
get_selected	get the current option of the text selector	Get the serial number of the current option;

Send data description:

Category	Description	Type	Remarks
text	text	text	Set/get showed text
value	value	uint	Set/get the value of the text selector
selected_index	options	uint	Set/get the current option of the text selector

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1080	text selector text delivery	passive	
0x1081	text selector value delivery	active/passive	Int type data
0x1082	text selector serial number delivery	passive	Int type data

Return data description:

Category	Data	Description	Remarks
CMD	0x1080	text delivery	Passively sending data from HMI to MCU (responding to sent data from MCU to HMI)
LEN	"widget name" + length of text	data length	
DATA	widget name + text	text content	The data length should not exceed 1,024 bytes (the text after the widget name: number)

Category	Data	Description	Remarks
CMD	0x1081	value delivery	Passively sending data from HMI to MCU (responding to sent data from MCU to HMI) And Actively sending data from HMI to MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Int type data, the last four bytes of the data part

Category	Data	Description	Remarks
CMD	0x1082	serial number delivery	Passively sending data from HMI to MCU (responding to sent data from MCU to HMI)
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Int type data, the last four bytes of the data part

3. For example

Set jumping to option location containing this text:

```
ST<{"cmd_code":"set_text","type":"text_selector","widget":"text_selector1","text":"stone"}>ET
ST<{"cmd_code":"set_text","type":"text_selector","widget":"text_selector1","text":"designer"}>ET
```

Set jumping to option location containing this value:
ST<{"cmd_code":"set_value","type":"text_selector","widget":"text_selector1","value":2021}>ET

Set the option position to jump to this sequence number:
ST<{"cmd_code":"set_selected","type":"text_selector","widget":"text_selector1",
"selected_index":5}>ET

Acquiring the text of current selected option:

a) Acquiring the text of "text_selector1" once the content is "2020" as an example:

Send:ST<{"cmd_code":"get_text","type":"text_selector","widget":"text_selector1"}>ET
Response:ST<0x10 0x80 0x00 0x15 "text_selector1":2020>ET
HEX:53 54 3C 10 80 00 15 22 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 31 22 3A 32 30 32 30 3E 45 54 63 40

b) Acquiring the text of "text_selector2" once the content is "yellow" as an example:

Send:ST<{"cmd_code":"get_text","type":"text_selector","widget":"text_selector2"}>ET
Response:ST<0x10 0x80 0x00 0x17 "text_selector2":yellow>ET
HEX:53 54 3C 10 80 00 17 22 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 32 22 3A 79 65 6C 6C 6F 77 3E 45 54 06 5E

Acquiring the value of current selected option:

a) Acquiring the value of "text_selector1" once its value is 2021 as an example:

Send:ST<{"cmd_code":"get_value","type":"text_selector","widget":"text_selector1"}>ET
Response:ST<0x10 0x81 0x00 0x12 text_selector1 0x00 0x00 0x07 0xE5>ET
HEX:53 54 3C 10 81 00 12 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 31 00 00 07 E5 3E 45 54 FE 5A

b) Acquiring the value of "text_selector2" once its value is 4 as an example:

Send:ST<{"cmd_code":"get_value","type":"text_selector","widget":"text_selector2"}>ET
Response:ST<0x10 0x81 0x00 0x12 text_selector2 0x00 0x00 0x00 0x04>ET
HEX:53 54 3C 10 81 00 12 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 32 00 00 00 04 3E 45 54 17 99

Special note for "b)" : If the content of selected option is only text but not value, the response data will represent the user defined serial number of selected option but not the system default unchangeable serial number of selected option.
For example once the options for one text selector are as:
Option 1:red; option 2:blue;option 3:green;option 4:yellow(selected option);option 5:grey;
And "get_value" instruction has been sent from MCU to HMI, the user will acquire "4" which is the user defined serial number of "Yellow" option

Acquiring the system default unchangeable serial number of selected option

a) text_selector1 - for example the selected option is the 50th one of "text_selector1" widget which representing 51 as the system default unchangeable serial number

Send:ST<{"cmd_code":"get_selected","type":"text_selector","widget":"text_selector1"}>ET
Response:ST<0x10 0x82 0x00 0x12 text_selector1 0x00 0x00 0x00 0x32>ET
HEX:53 54 3C 10 82 00 12 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 31 00 00 00 32 3E 45 54 75 32

b) text_selector2 - for example the selected option is the 5th one of “text_selector2” widget which representing 6 as the system default unchangeable serial number

Send:ST<{"cmd_code":"get_selected","type":"text_selector","widget":"text_selector2"}>ET
Response:ST<0x10 0x82 0x00 0x12 text_selector2 0x00 0x00 0x00 0x05>ET
HEX:53 54 3C 10 82 00 12 74 65 78 74 5F 73 65 6C 65 63 74 6F 72 32 00 00 00 05 3E 45 54 14 7C

Special note for “b)” : If the content of selected option is only text but not value, the response data will represent the user defined serial number of selected option but not the system default unchangeable serial number of selected option.
For example once the options for one text selector are as:
1:red;2:blue;3:green;4:yellow(selected option);5:grey;
And “get_value” instruction has been sent from MCU to HMI, the user will acquire “4” which is the user defined serial number of “Yellow” option. However, “get_selected” instruction is for acquiring the system default unchangebale serial number of selected option.

4.11 slider

1. Instruction sending:

Instruction	Instruction description	Remarks
set_max	set the slider maximum value	
set_min	set the slider minimum value	
set_step	set the slider step value	
set_value	set the slider value	
get_value	get slider value	

Send data description:

Category	Description	Type	Remarks
max	text	text	Set the slider maximum value
min	value	uint	Set the slider minimum value
step	step value	uint	Set the slider step value
value	value	uint	Set/get slider value

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1040	slider value is changing	initiative	
0x1041	after the slider value is changed	initiative	

Return data description:

Category	Data	Description	Remarks
CMD	0x1040	slider value change	Actively send from HMI to MCU when the value of slider widget changes
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	The last four bytes of the data part, type float data, according to IEEE 754 standard

Category	Data	Description	Remarks
CMD	0x1041	value sent from HMI to MCU	The value of slider widget after changed
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	The last four bytes of the data part, type float, according to IEEE 754 standard

3. For example

Set slider parameters:

a) Set the slider1 widget maximum value to 200:

ST<{"cmd_code":"set_max","type":"slider","widget":"slider1","max":200}>ET

b) Set the slider1 widget minimum value to 0:

ST<{"cmd_code":"set_min","type":"slider","widget":"slider1","min":0}>ET

c) Set the current value of the slider1 widget to 10:

ST<{"cmd_code":"set_value","type":"slider","widget":"slider1","value":10}>ET

d) Set the values of slider1 to slider37 widgets in batch:

ST<{"cmd_code":"set_value","type":"slider","widget":"slider1_37","value":[10,12,80,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,80,15,12]}>ET

Special Note: Please refer to the label widget for the description of setting widget values/text in batch;

e) Set slider1 widget step value 1:

ST<{"cmd_code":"set_step","type":"slider","widget":"slider1","step":1}>ET

Get slider parameters:

a) The slider1 data is changing, which is 48.000000:

Response: ST<0x10 0x40 0x00 0x0B slider1 0x42 0x40 0x00 0x00>ET

HEX:53 54 3C 10 40 00 0B 73 6C 69 64 65 72 31 42 40 00 00 3E 45 54 27 6D

b) The slider1 data is changing, which is 49.000000:

Response: ST<0x10 0x40 0x00 0x0B slider1 0x42 0x44 0x00 0x00>ET
HEX:53 54 3C 10 40 00 0B 73 6C 69 64 65 72 31 42 44 00 00 3E 45 54 A3 6C

c) Get slider1 data (change completed), the data is 49.000000:
Send: ST<{"cmd_code":"get_value","type":"slider","widget":"slider1"}>ET
Response: ST<0x10 0x41 0x00 0x0B slider1 0x42 0x44 0x00 0x00>ET
HEX:53 54 3C 10 41 00 0B 73 6C 69 64 65 72 31 42 44 00 00 3E 45 54 33 3D

4.12 image

1. Instruction sending:

Instruction	Instruction description	Remarks
set_image	set the image name	
set_draw_type	set the drawing type of image	
set_scale	set the scale ratio of the image	
set_rotation	set the rotation of the image	

Send data description:

Category	Description	Type	Remarks
image	image name	text	
draw_type	drawing type	uint	The value of draw_type is as follows: 0: Default mode. Showing the image as its original size in the top left corner of the targeting rectangle area. 1: center mode. Showing the image as its original size in the center of the targeting rectangle area. 2: icon mode. Showing the image in the center position, but readjust its size based on screen density. 3: scale mode. Scale the image to the size as same as the size of targeting rectangle area (ratio of width and height is not fixed). 4: Automatic scale mode. Scale the image to the width or height of the targeting rectangle area (the smallest ratio will be selected), and showing the image in the center position. 5: If the image is larger than the targeting rectangle area, the size of the image will be diminished automatically, otherwise it will be showed in the center position. 6: Width scale mode. Scale the image to the width of the targeting rectangle area, and the height will be scaled according to the ratio, exceeded part of the image will not be displayed. 7: Height scale mode. Scale the image to the height of the targeting rectangle area, and the width will be scaled according to the ratio, exceeded part of the image will not be displayed. 8: Tile mode. 9: Tile mode in the horizontal direction and scale the image in vertical direction.

			10: Tile mode vertically and scale the image in horizontal direction. 11: Tile mode vertically and scale the image in horizontal direction (from bottom to top).
scale_x	x-axis scaling	float	image scaling
scale_y	y-axis scaling	float	image scaling
rotation	rotation angle	uint	Rotation angle degree

2. Instruction return:

Return instruction	Return description	Return type	Remarks
0x1090	image system key value sent from HMI to MCU	initiative	System key value (last two bytes of data part): 0x0001: press down / pressing down the key 0x0002: press click/ pressing down the key and released (trigger click event or state) 0x0004: press up/ after releasing the key (the key back to unpressed state)
0x1091	image user-defined key valuse sent from HMI to MCU	initiative	User key value (last two bytes of data part): User customized defined key

Return data description:

Category	Data	Description	Remarks
CMD	0x1090	key value sent from HMI to MCU	Image “button key” value sent from HMI to MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Last two bytes of the data part

Category	Data	Description	Remarks
CMD	0x1091	key value sent from HMI to MCU	User customized defined Image “button key” value sent from HMI to MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Last two bytes of the data part

3. For example

Set image parameters:

a) Set the image widget name to guage_bg/vgus01:

```
ST<{"cmd_code":"set_image","type":"image","widget":"image","image":"guage_bg"}>ET
ST<{"cmd_code":"set_image","type":"image","widget":"image","image":"vgus01"}>ET
```

b) Set the image widget drawing type to 2 (center show):

```
ST<{"cmd_code":"set_draw_type","type":"image","widget":"image","draw_type":2}>ET
```

c) Set the scaling of the image widget:

```
ST<{"cmd_code":"set_scale","type":"image","widget":"image","scale_x":0.5,"scale_y":0.5}>ET
```

ST<{"cmd_code":"set_scale","type":"image","widget":"image","scale_x":1,"scale_y":1}>ET

d) Set the rotation angle of the image widget to 90/180:
ST<{"cmd_code":"set_rotation","type":"image","widget":"image","rotation":90}>ET
ST<{"cmd_code":"set_rotation","type":"image","widget":"image","rotation":180}>ET

Image system key delivery:
a) Image1 widget pressed:
Response: ST<0x10 0x90 0x00 0x08 image1 0x00 0x01>ET
HEX:53 54 3C 10 90 00 08 69 6D 61 67 65 31 00 01 3E 45 54 CE 5C

b) The image1 widget is released (click event):
Response: ST<0x10 0x90 0x00 0x08 image1 0x00 0x02>ET
HEX:53 54 3C 10 90 00 08 69 6D 61 67 65 31 00 02 3E 45 54 8A 5C

c) The image1 widget is released:
Response: ST<0x10 0x90 0x00 0x08 image1 0x00 0x04>ET
HEX:53 54 3C 10 90 00 08 69 6D 61 67 65 31 00 04 3E 45 54 02 5C

Image user-defined key delivery:
a) Image1 widget is customized pressed:
Response: ST<0x10 0x91 0x00 0x08 image1 0x04 0xD2>ET
HEX:53 54 3C 10 91 00 08 69 6D 61 67 65 31 04 D2 3E 45 54 4B 95

b) Image1 widget is customized released (click event):
Response: ST<0x10 0x91 0x00 0x08 image1 0x16 0x2E>ET
HEX:53 54 3C 10 91 00 08 69 6D 61 67 65 31 16 2E 3E 45 54 18 1D

c) Image1 widget is customized released:
Response: ST<0x10 0x91 0x00 0x08 image1 0x22 0xCE>ET
HEX:53 54 3C 10 91 00 08 69 6D 61 67 65 31 22 CE 3E 45 54 1C 9B

Special note: The button function of image can only be used after checking the clickable attribute and setting the corresponding key-value attribute.
By default, no key will be delivered;

4.13 image_value

1. Instruction sending:

Instruction	Instruction description	Remarks
set_image	set the name prefix of the image value	Displaying numbers of this widget are composed of a series of images, this instruction is used to set the image name prefix
set_format	set format of image value	
set_max	set the maximum value of the image value	
set_min	set the minimum value of the image value	
set_value	set the value of the image value	
get_value	get the value of the image value	

Send data description:

Category	Description	Type	Remarks
image	image name	text	Set the name prefix of the images value widget, such as num0 to num9 prefixed with “num”
format	number format	text	Set the format of the image value widget, Value: %d,%02d,%03d,%04d,%05d,%06d,%f,%.1f,%.2f,%.3f,%.4f,%.5f,%.6f
max	maximum value	uint	Set the maximum value of the image value widget
min	minimum value	uint	Set the minimum value of the image value widget
value	image value	float	Set/get the value of the image value widget

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1092	image_value value delivery	initiative	Float type data

Return data description:

Category	Data	Description	Remarks
CMD	0x1092	value sent from HMI to MCU	Image_value widget value sent from HMI to MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Float type data, the last four bytes of the data part, according to IEEE 754 standard

3. For example

Set the image value parameters:

a) Set the image_value widget image name prefix:

ST<{"cmd_code":"set_image","type":"image_value","widget":"image_value","image":"num"}>ET

b) Set the image_value widget format:
ST<{"cmd_code":"set_format","type":"image_value","widget":"image_value","format":"%02.2f"}>ET

c) Set the maximum value of the image_value widget:
ST<{"cmd_code":"set_max","type":"image_value","widget":"image_value","max":200}>ET

d) Set the minimum value of the image_value widget:
ST<{"cmd_code":"set_min","type":"image_value","widget":"image_value","min":0}>ET

e) Set the current value of the image_value widget:
ST<{"cmd_code":"set_value","type":"image_value","widget":"image_value","value":6.66}>ET

f) Set the value of widgets image_value1 to image_value35 in batch:
ST<{"cmd_code":"set_value","type":"image_value","widget":"image_value1_35","value":[10,12,80,15,
12,10,12,10,10,12,8,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,80,15,12,10,12,10,10,12,80]}>ET

Special Note: Please refer to the label widget for the description of setting widget values/text in batch;

Get image value parameters:
a) Get the value of image_value as 4.23:
Send: ST<{"cmd_code":"get_value","type":"image_value","widget":"image_value"}>ET
Response: ST<0x10 0x92 0x00 0x0F image_value 0x40 0x87 0x5C 0x29>ET
HEX:53 54 3C 10 92 00 0F 69 6D 61 67 65 5F 76 61 6C 75 65 40 87 5C 29 3E 45 54 F6 DB

4.14 image_animation

1. Instruction sending:

Instruction	Instruction description	Remarks
set_play	set image animation to play	
set_pause	set image animation to pause	
set_stop	set image animation to stop	
set_format	set the format of image animation widget	
set_image	set the image name prefix of the image animation	
set_interval	set image animation interval	
set_loop	set on/off of the image animation to loop playback	
set_range	set image animation range	
set_frame	set the animation image to display a specific frame of image	The specific frame must be within the range between start_index and end_index

Send data description:

Category	Description	Type	Remarks
format	the composition format of image animation	text	Set the image composition format of the image animation, such as num0-num9 composition, then the format is %s%d
image	image name prefix	text	Set the image name prefix of the image animation
interval	interval	uint	Image animation playback interval, unit: ms
loop	on/off to loop image animation	bool	Set on/off to loop playback
start_index	image starting ordinal	uint	Image starting serial number
end_index	image ending ordinal	uint	Image ending serial number
frame	image specific frame index	uint	Specific Image frame index (value range: start_index - end_index)

2. For example:

Set image animation parameters:

a) Set image_animation to start playback:

```
ST<{"cmd_code":"set_play","type":"image_animation","widget":"image_ani1"}>ET
```

b) Set image_animation to pause playback:

```
ST<{"cmd_code":"set_pause","type":"image_animation","widget":"image_ani1"}>ET
```

c) Set image_animation to stop playback:

```
ST<{"cmd_code":"set_stop","type":"image_animation","widget":"image_ani1"}>ET
```

d) Set the image composition format of image_animation:

```
ST<{"cmd_code":"set_format","type":"image_animation","widget":"image_ani1","format":"%s%d"}>ET
```

e) Set the image animation name prefix of image_animation :

```
ST<{"cmd_code":"set_image","type":"image_animation","widget":"image_ani1","image":"num"}>ET
```

f) Set the image_animation playback interval:

```
ST<{"cmd_code":"set_interval","type":"image_animation","widget":"image_ani1","interval":200}>ET
```

g) Set image_animation whether to loop playback:

```
ST<{"cmd_code":"set_loop","type":"image_animation","widget":"image_ani1","loop":true}>ET
```

h) Set the start and end index of image_animation:

```
ST<{"cmd_code":"set_range","type":"image_animation","widget":"image_ani1","start_index":1,
"end_index":9}>ET
```

i) Set image_animation to display a specific frame (frame 1 image):

```
ST<{"cmd_code":"set_frame","type":"image_animation","widget":"image_ani1","frame":1}>ET
```

4.15 gif

1. Instruction delivery

Instruction	Instruction description	Remarks
set_image	set the image name to display	
set_play	set gif image to play	
set_pause	set gif image to pause	
set_stop	set gif image to stop	
set_loop	set the number of frames to loop playback	Stop after how many frames been played
set_frame	set which frame of the gif image to display	Only valid once gif image is in pause/stop state
set_scale	set the scale ratio of the image	
set_rotation	set the rotation angle of the image	

Send data description:

Category	Description	Type	Remarks
image	image name	text	Set the image name to display
loop	number of frames played	uint	Set the number of frames for loop playback (stop after how many frames been played)
frame	image frame	uint	Only valid once gif image is in pause/stop state
scale_x	x-axis scaling	float	Set the scale ratio of the image based on x-axis
scale_y	y-axis scaling	float	Set the scale ratio of the image based on y-axis
rotation	start and end images	float	Set the rotation angle and unit angle of the image

2. For example:
Set gif image parameters:
a) Set the showed gif image:

```
ST<{"cmd_code":"set_image","type":"gif","widget":"gif0","image":"bear"}>ET
ST<{"cmd_code":"set_image","type":"gif","widget":"gif0","image":"monkey"}>ET
```

b) Set gif image play:

```
ST<{"cmd_code":"set_play","type":"gif","widget":"gif0"}>ET
```

c) Set gif image pause:

```
ST<{"cmd_code":"set_pause","type":"gif","widget":"gif0"}>ET
```

d) Set gif image stop:

```
ST<{"cmd_code":"set_stop","type":"gif","widget":"gif0"}>ET
```

e) Set the number of frames to loop playback:

```
ST<{"cmd_code":"set_loop","type":"gif","widget":"gif0","loop":123}>ET
```

f) Set which frame of the gif to show:

```
ST<{"cmd_code":"set_frame","type":"gif","widget":"gif0","frame":0}>ET
ST<{"cmd_code":"set_frame","type":"gif","widget":"gif0","frame":1}>ET
ST<{"cmd_code":"set_frame","type":"gif","widget":"gif0","frame":10}>ET
```

g) Set the scaling of the image:

```
ST<{"cmd_code":"set_scale","type":"gif","widget":"gif0","scale_x":0.5,"scale_y":0.5}>ET
```

h) Set the rotation angle of the image:

```
ST<{"cmd_code":"set_rotation","type":"gif","widget":"gif0","rotation":90}>ET
```

4.16 svg

1. Instruction delivery

Instruction	Instruction description	Remarks
set_image	set the image name to display	
set_scale	set the scale ratio of the image	
set_rotation	set the rotation angle of the image	

Send data description:

Category	Description	Type	Remarks
image	image name	text	Set the image name to display
scale_x	x-axis scaling	float	Set the scale ratio of the image based on x-axis
scale_y	y-axis scaling	float	Set the scale ratio of the image based on y-axis
rotation	rotation	float	Set the rotation angle of the image

2. For example:

Set svg image parameters:

a) Set the image showed by svg0:

```
ST<{"cmd_code":"set_image","type":"svg","widget":"svg0","image":"1"}>ET
ST<{"cmd_code":"set_image","type":"svg","widget":"svg0","image":"login"}>ET
```

b) Set the scaling of the svg0 image:

```
ST<{"cmd_code":"set_scale","type":"svg","widget":"svg0","scale_x":0.5,"scale_y":0.5}>ET
ST<{"cmd_code":"set_scale","type":"svg","widget":"svg0","scale_x":1,"scale_y":1}>ET
```

c) Set the rotation angle of the svg0 image:

```
ST<{"cmd_code":"set_rotation","type":"svg","widget":"svg0","rotation":90}>ET
ST<{"cmd_code":"set_rotation","type":"svg","widget":"svg0","rotation":180}>ET
```

4.17 BTN button

1. Instruction returns:

Return instruction	Description	Data return type	Remarks
0x1001	system key value sent from HMI to MCU	initiative	System key value (last byte of data part): 0x01: press down / pressing down the button 0x02: press click/ pressing down the button and released (trigger click event or state) 0x03: long pressed/ long pressing the button (if repeat state is not value 0, the click event will be triggered continuously) 0x04: press up/ after releasing the button (the key back to unpressed state)
0x1002	user-defined key delivery	initiative	User key value (last two bytes of data part): Two bytes of data, the meaning of the key value can be defined by users

Return data description:

Category	Data	Description	Remarks
CMD	0x1001	System default key value	Click the button and HMI will automatically send the data to MCU , once the user does not set a customized key value to change the system default key value
LEN	"widget name" + the length of the key value	data length	
DATA	"widget name" + key value	data content	Last byte of data part

Category	Data	Description	Remarks
CMD	0x1002	user key	Click the button and HMI will automatically send data to MCU, if the user does not set a customized key value, the system key value will be delivered by default, if the user key value is set, the user-defined key value will be delivered
LEN	"widget name" + length of user customized key value	data length	
DATA	widget name + user key value	data content	The last two bytes of the data part; high-order data first, low-order data last

2. For example:

System key value:

a) Button press instruction:

Response: ST<0x10 0x01 0x00 0x08 button9 0x01>ET

HEX:53 54 3C 10 01 00 08 62 75 74 74 6F 6E 39 01 3E 45 54 E7 E0

b) Click button and release (complete button click action) instruction:

Response: ST<0x10 0x01 0x00 0x08 button9 0x02>ET
HEX:53 54 3C 10 01 00 08 62 75 74 74 6F 6E 39 02 3E 45 54 A3 E0

c) Button release instruction:

Response: ST<0x10 0x01 0x00 0x08 button1 0x04>ET
HEX:53 54 3C 10 01 00 08 62 75 74 74 6F 6E 31 04 3E 45 54 EA 01

d) Button long press instruction:

Response: ST<0x10 0x01 0x00 0x08 button9 0x03>ET
HEX:53 54 3C 10 01 00 08 62 75 74 74 6F 6E 39 03 3E 45 54 5F E1

User-defined key:

a) Button press customized instruction 0x04D2:

Response: ST<0x10 0x02 0x00 0x09 button1 0x04 0xD2>ET
HEX:53 54 3C 10 02 00 09 62 75 74 74 6F 6E 31 04 D2 3E 45 54 66 23

b) Button release (complete button click action) customized instruction 0x162E:

Response: ST<0x10 0x02 0x00 0x09 button1 0x16 0x2E>ET
HEX:53 54 3C 10 02 00 09 62 75 74 74 6F 6E 31 16 2E 3E 45 54 35 AB

c) Button release customized instruction 0x0315:

Response: ST<0x10 0x02 0x00 0x09 button1 0x03 0x15>ET
HEX:53 54 3C 10 02 00 09 62 75 74 74 6F 6E 31 03 15 3E 45 54 D2 AB

d) Button long press customized instruction 0x0064:

Response: ST<0x10 0x02 0x00 0x09 button1 0x00 0x64>ET
HEX:53 54 3C 10 02 00 09 62 75 74 74 6F 6E 31 00 64 3E 45 54 EE F4

4.18 ☑ check button

1. Instruction sending:

Instruction	Instruction description	Remarks
set_value	set on/off for the checkbox widget	

Send data description:

Category	Description	Type	Remarks
value	selected status values	bool	Set the selected state value, value options: true/false

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1020	check button value	initiative	Key: 0x00: unchecked state; 0x01: checked state

Return data description:

Category	Data	Description	Remarks
CMD	0x1020	check button value	
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Last byte of data part

3. For example:
Set parameters:
ST<{"cmd_code":"set_value","type":"check_button","widget":"check_button","value":true}>ET
ST<{"cmd_code":"set_value","type":"check_button","widget":"check_button","value":false}>ET

Get parameters:
a) The value of the check button is changed and the instruction is delivered actively - unchecked:
Response: ST<0x10 0x20 0x00 0x0D check_button 0x00>ET
HEX:53 54 3C 10 20 00 0D 63 68 65 63 6B 5F 62 75 74 74 6F 6E 00 3E 45 54 AF 1E

b) The value of the check button is changed and the instruction is delivered actively - selected
Response: ST<0x10 0x20 0x00 0x0D check_button 0x01>ET
HEX:53 54 3C 10 20 00 0D 63 68 65 63 6B 5F 62 75 74 74 6F 6E 01 3E 45 54 53 1F

4.19 ◉ radio_button

1. Instruction sending:

Instruction	Instruction description	Remarks
set_value	set on/off for checkbox state	Set the select state, value options: true/false
get_checked	get the currently checked radio button state	Key value: 0x00: unchecked state; 0x01: checked state

Send data description:

Category	Description	Type	Remarks
value	selected status values	bool	Set the select state, value options: true/false

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1030	radio_button value change	initiative	
0x1031	radio_button value change	passive	The MCU uses the get_checked instruction to obtain the value

Return data description:

Category	Data	Description	Remarks
CMD	0x1030	check button value	Active sent by HMI to MCU
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Last byte of data part

Category	Data	Description	Remarks
CMD	0x1031	check button value	Passively sent from HMI to MCU, use the get_checked instruction to obtain the value
LEN	"widget name" + the length of the value	data length	
DATA	widget name + value	value content	Last byte of data part

3. For example:
Set parameters:
ST<{"cmd_code":"set_value","type":"radio_button","widget":"radio_button","value":true}>ET
ST<{"cmd_code":"set_value","type":"radio_button","widget":"radio_button","value":false}>ET

Actively deliver instruction:
a) Manually select radio_button1, radio_button is automatically closed and selected:
Response: ST<0x10 0x30 0x00 0x0E radio_button1 0x01>ET
HEX:53 54 3C 10 30 00 0E 72 61 64 69 6F 5F 62 75 74 74 6F 6E 31 01 3E 45 54 34 4E
Response: ST<0x10 0x30 0x00 0x0D radio_button 0x00>ET
HEX:53 54 3C 10 30 00 0D 72 61 64 69 6F 5F 62 75 74 74 6F 6E 00 3E 45 54 32 36

b) Manually select radio_button2, radio_button1 is automatically closed and selected:
Response: ST<0x10 0x30 0x00 0x0E radio_button2 0x01>ET
HEX:53 54 3C 10 30 00 0E 72 61 64 69 6F 5F 62 75 74 74 6F 6E 32 01 3E 45 54 34 0A
Response: ST<0x10 0x30 0x00 0x0E radio_button1 0x00>ET
HEX:53 54 3C 10 30 00 0E 72 61 64 69 6F 5F 62 75 74 74 6F 6E 31 00 3E 45 54 C8 4F

MCU actively obtains the current option:
a) Actively get the current option: radio_button1
Send: ST<{"cmd_code":"get_checked","type":"radio_button","widget":"radio_button1"}>ET
Response: ST<0x10 0x31 0x00 0x0E radio_button1 0x01>ET
HEX:53 54 3C 10 31 00 0E 72 61 64 69 6F 5F 62 75 74 74 6F 6E 31 01 3E 45 54 E5 73

b) Actively get the current option: radio_button2
Send: ST<{"cmd_code":"get_checked","type":"radio_button","widget":"radio_button2"}>E
Response: ST<0x10 0x31 0x00 0x0E radio_button2 0x01>ET
HEX:53 54 3C 10 31 00 0E 72 61 64 69 6F 5F 62 75 74 74 6F 6E 32 01 3E 45 54 E5 37

4.20 switch

1. Instruction sending:

Instruction	Instruction description	Remarks
set_value	set switch value	

Send data description:

Category	Description	Type	Remarks
value	check value	bool	Set check value, options: true/false

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1010	after the switch value is changed	initiative	Key (last byte of data part): 0x00: switch off 0x01: switch on

Return data description:

Category	Data	Description	Remarks
CMD	0x1010	switch value	The value will be sent automatically from HMI to MCU once the widget been clicked
LEN	"widget name" + value	"widget name" + value content length	
DATA	Widget name + value	value content	Key value (last byte of data part)

3. For example:

Set switch parameters:

ST<{"cmd_code":"set_value","type":"switch","widget":"switch","value":true}>ET

ST<{"cmd_code":"set_value","type":"switch","widget":"switch","value":false}>ET

Instruction actively returns:

a) The switch value is changed and the instruction is delivered actively - the switch switch is turned off:

Response: ST<0x10 0x10 0x00 0x07 switch 0x00>ET

HEX:53 54 3C 10 10 00 07 73 77 69 74 63 68 00 3E 45 54 21 F2

b) The switch value is changed and the instruction is delivered actively - the switch switch is turned on:

Response: ST<0x10 0x10 0x00 0x07 switch 0x01>ET

HEX:53 54 3C 10 10 00 07 73 77 69 74 63 68 01 3E 45 54 DD F3

4.21 digit_clock/ time_clock

1. Instruction sending:

Instruction	Instruction description	Remarks
set_date	set RTC time	
set_format	set the format of the clock	Only applicable for digit clock
get_date	get RTC time	

Send data description:

Category	Description	Type	Remarks
date	time	text	Set/get RTC time
format	time format	text	See the table below for values

Format value	Description
Y	represent year (showed as full details)
M	represent month (1-12)
D	represent day (1-31)
h	represent hour (0-23)
m	represent minute (0-59)
s	represent second (0-59)
w	represent week (0-6)
W	abbreviation for the week
YY	represents the year (only the last two digits are showed)
MM	represent month (01-12)
DD	represent day (01-31)
hh	represent hour (00-23)
mm	represent minute (00-59)
ss	represent second (00-59)
MMM	abbreviation for month

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10F0	date+time return	passive	

Return data description:

Category	Data	Description	Remarks
CMD	0x10F0	date+time return	
LEN	"widget name": the length of the datetime	data length	
DATA	"widget name" + datetime	data content	

3. For example:

Set digital clock parameters:

a) Set the clock time:

ST<{"cmd_code":"set_date","type":"digit_clock","widget":"digit_clock","date":"12:23"}>ET
ST<{"cmd_code":"set_date","type":"digit_clock","widget":"digit_clock","date":"12:23:46"}>ET
ST<{"cmd_code":"set_date","type":"digit_clock","widget":"digit_clock","date":"2021-02-26 12:23"}>ET
ST<{"cmd_code":"set_date","type":"digit_clock","widget":"digit_clock",
"date":"2021-02-26 12:23:46"}>ET
ST<{"cmd_code":"set_date","type":"time_clock","widget":"time_clock1",
"date":"2021-02-26 12:23:46"}>ET

b) Set the clock displaying format:
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock","format":"hh:mm"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock","format":"hh:mm:ss"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-MM-DD hh:mm"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-MM-DD hh:mm:ss"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-MM-DD hh:mm:ss w"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-MM-DD hh:mm:ss W"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-MM-DD hh:mm:ss MMM"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY-M-D h:m:s"}>ET
ST<{"cmd_code":"set_format","type":"digit_clock","widget":"digit_clock",
"format":"YYYY/MM/DD hh:mm:ss"}>ET

Get the date and time data delivery (sent from HMI to MCU):
a) Get the date and time of digit_clock1: 2021-02-26 12:31:35
Send: ST<{"cmd_code":"get_date","type":"digit_clock","widget":"digit_clock1"}>ET
Response: ST<0x10 0xF0 0x00 0x22 "digit_clock1":2021-02-26 12:31:35>ET
HEX:53 54 3C 10 F0 00 22 22 64 69 67 69 74 5F 63 6C 6F 63 6B 31 22 3A 32 30 32 31 2D 30 32 2D 32
36 20 31 32 3A 33 31 3A 33 35 3E 45 54 30 BB

b) Get the date and time of time_clock1: 2021-02-26 12:34:57
Send: ST<{"cmd_code":"get_date","type":"time_clock","widget":"time_clock1"}>ET
Response: ST<0x10 0xF0 0x00 0x21 "time_clock1":2021-02-26 12:34:57>ET
HEX:53 54 3C 10 F0 00 21 22 74 69 6D 65 5F 63 6C 6F 63 6B 31 22 3A 32 30 32 31 2D 30 32 2D 32 36
20 31 32 3A 33 34 3A 35 37 3E 45 54 D7 35

4.22 gauge

1. Instruction sending:

Instruction	Instruction description	Remarks
set_image	set the image name for displaying	
set_draw_type	set the drawing type of the image (same as image widget)	

Send data description:

Category	Description	Type	Remarks
image	image	text	Set the image name for displaying
draw_type	drawing type	uint	Set the drawing type of the image, the value is the same as the image widget

2. For example:
Set image parameters:
a) Set gauge1 background image:

```
ST<{"cmd_code":"set_image","type":"gauge","widget":"gauge1","image":"gauge_bg"}>ET
ST<{"cmd_code":"set_image","type":"gauge","widget":"gauge1","image":"gauge_bg1"}>ET
```

b) Set gauge1 image drawing type:

```
ST<{"cmd_code":"set_draw_type","type":"gauge","widget":"gauge1","draw_type":2}>ET
```

4.23 gauge_pointer

1. Instruction sending:

Instruction	Instruction description	Remarks
set_image	set the image name for displaying	
set_angle	sets the rotation angle of the pointer	

Send data description:

Category	Description	Type	Remarks
image	image	text	Set the name of the image to be showed (same as image widget)
angle	angle	float	Set the rotation angle of the pointer

2. For example:
Set image parameters:
a) Set gp1 gauge pointer image:

```
ST<{"cmd_code":"set_image","type":"gauge_pointer","widget":"gp1","image":"guage_pointer"}>ET
ST<{"cmd_code":"set_image","type":"gauge_pointer","widget":"gp1","image":"guage_pointer1"}>ET
ST<{"cmd_code":"set_image","type":"gauge_pointer","widget":"gp1","image":"guage_pointer2"}>ET
```

b) Set gp1 gauge pointer rotation angle:

```
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":0}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":30}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":60}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":90}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":-90}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":180}>ET
ST<{"cmd_code":"set_angle","type":"gauge_pointer","widget":"gp1","angle":270}>ET
```

4.24 chart_view

4.24.1 x_axis / y_axis

1. Instruction sending：

Instruction	Instruction description	Remarks
set_min	set the minimum value of the curve axis	
set_max	set the maximum value of the curve axis	
set_range	set the value range of the curve axis	Set the minimum and maximum values, the same function as set_min and set_max instruction
set_data	set the date of the curve sequence point	

Sending data instructions:

Category	Description	Type	Remarks
min	minimum	float	Set the minimum value of the curve axis
max	maximum	float	Set the maximum value of the curve axis
data	the scale value of the coordinate axis	text	Set the scale value of the curve axis, Use the symbols "[ ]" to contain the data, Use the symbol "," split;

2. Instruction return：

Return instruction	Return description	Data return type	Remarks
0x1160	minimum value sent from HMI to MCU	passive	Minimum value format: widget name +float value
0x1161	maximum value sent from HMI to MCU	passive	Maximum value format: widget name +float value

Return data description:

Category	Data	Description	Remarks
CMD	0x1160	value delivery	
LEN	widget name +float value length	data length	
DATA	widget name +float value	data content	float type data

Category	Data	Description	Remarks
CMD	0x1161	capacity sent from HMI to MCU	
LEN	widget name +float value length	Data length	
DATA	widget name +float value	Data content	float type data

3. For example:：

Set the minimum value of the coordinate axis:

a) Set the minimum value of x_axis to 0:

ST<{"cmd_code":"set_min","type":"x_axis","widget":"x_axis1","min":0}>ET

b) Set the minimum value of y_axis to 0:

ST<{"cmd_code":"set_min","type":"y_axis","widget":"y_axis1","min":0}>ET

c) Set the maximum value of the x_axis to 19:

ST<{"cmd_code":"set_max","type":"x_axis","widget":"x_axis1","max":19}>ET

d) Set the minimum value of y_axis to 210:

ST<{"cmd_code":"set_max","type":"y_axis","widget":"y_axis1","max":210}>ET

Set the maximum and minimum values of the x_axis and y_axis:

ST<{"cmd_code":"set_range","type":"x_axis","widget":"x_axis1","min":0,"max":19}>ET
ST<{"cmd_code":"set_range","type":"y_axis","widget":"y_axis1","min":0,"max":210}>ET

Set the scale display value of the x_axis:

ST<{"cmd_code":"set_data","type":"x_axis","widget":"x_axis1","data":"[1,2,3,4,5,6,7,8,9,10]"}>ET
ST<{"cmd_code":"set_data","type":"x_axis","widget":"x_axis1",
"data":"[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]"}>ET

Set the scale display value of the y_axis:

ST<{"cmd_code":"set_data","type":"y_axis","widget":"y_axis1",
"data":"[0,20,40,60,80,100,120,140]"}>ET

ST<{"cmd_code":"set_data","type":"y_axis","widget":"y_axis1",
"data":"[0,30,60,90,120,150,180,210]"}>ET

Get the maximum and minimum values of the coordinate axes:

a) Get the minimum value of the x_axis is 0:

Send: ST<{"cmd_code":"get_min","type":"x_axis","widget":"x_axis1"}>ET
Response: ST<0x11 0x60 0x00 0x0B x_axis1 0x00 0x00 0x00 0x00>ET
HEX:53 54 3C 11 60 00 0B 78 5F 61 78 69 73 31 00 00 00 00 3E 45 54 CD 40

b) Get the maximum value of the x_axis is 9:

Send: ST<{"cmd_code":"get_max","type":"x_axis","widget":"x_axis1"}>ET
Response: ST<0x11 0x61 0x00 0x0B x_axis1 0x41 0x10 0x00 0x00>ET
HEX:53 54 3C 11 61 00 0B 78 5F 61 78 69 73 31 41 10 00 00 3E 45 54 C9 42

c) The minimum value of the y_axis is -10:
Send: ST<{"cmd_code":"get_min","type":"y_axis","widget":"y_axis1"}>ET
Response: ST<0x11 0x60 0x00 0x0B y_axis1 0xC1 0x20 0x00 0x00>ET
HEX:53 54 3C 11 60 00 0B 79 5F 61 78 69 73 31 C1 20 00 00 3E 45 54 A0 97

d) The maximum value of the y_axis is 210:
Send: ST<{"cmd_code":"get_max","type":"y_axis","widget":"y_axis1"}>ET
Response: ST<0x11 0x61 0x00 0x0B y_axis1 0x43 0x52 0x00 0x00>ET
HEX:53 54 3C 11 61 00 0B 79 5F 61 78 69 73 31 43 52 00 00 3E 45 54 EA 6E

4.24.2 line_series / bar_series

1. Instruction sending:

Line_series/bar_series related:

Instruction	Instruction description	Remarks
set_line	set the boundary line of the curve sequence for showing or not, whether it is smooth or not	Only for line_series
set_area	set whether the curve sequence area are showing or not	Only for line_series
set_symbol	set whether curve index markers are showing or not	Only for line_series
set_value	set curve index data	
set_capacity	set the curve index FIFO capacity	The curve sequence data will be reset after setting the volume
get_value	get curve index data	
get_capacity	get cancel index FIFO capacity	

Send data description:

Category	Description	Type	Remarks
show	whether to show or not	bool	Used to set whether the curve/image is showing or not
smooth	whether it is smooth o not	bool	Used to set whether the curve is showing smoothly
symbol	symbol	bool	Set whether curve series point markers are showing or not (only for line_series)
index	index	uint	index number, starting from 0
mode	mode	text	index set value mode, value: push, set the value in append mode
value	value	float	index value, which can be a single float data or an array of float data
capacity	capacity	uint	Curve/histogram FIFO Capacity

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x10D1	value sent from HMI to MCU	passive	index value format: widget name + index value + float value
0x10D2	capacity sent from HMI to MCU	passive	index capacity format: widget name + capacity value (uint type)

Return data description:

Category	Data	Description	Remarks
CMD	0x10D1	value sent from HMI to MCU	
LEN	index value format: widget name + index value + float value	data length	
DATA	widget name + index value + float value	data content	Index type: int, value type: float type

Category	Data	Description	Remarks
CMD	0x10D2	capacity sent from HMI to MCU	
LEN	index capacity format: widget name + capacity value	data length	
DATA	widget name + capacity value	data content	uint type data

3. For example:

Set whether curve boundary lines are showed:

a) Set line_series1 boundary line smooth show

ST<{"cmd_code":"set_line","type":"line_series","widget":"line_series1","show":true,"smooth":true}>ET

b) Set line_series1 boundary line polyline show

ST<{"cmd_code":"set_line","type":"line_series","widget":"line_series1",
"show":true,"smooth":false}>ET

c) Set line_series1 boundary line smooth not to show

ST<{"cmd_code":"set_line","type":"line_series","widget":"line_series1",
"show":false,"smooth":true}>ET

d) Set line_series1 boundary line polyline not to show

ST<{"cmd_code":"set_line","type":"line_series","widget":"line_series1",
"show":false,"smooth":false}>ET

Set whether the curve area is showed:

e) Set line_series1 to set the curve area show

ST<{"cmd_code":"set_area","type":"line_series","widget":"line_series1","show":true}>ET

f) Set line_series1 to set the curve area to not show
ST<{"cmd_code":"set_area","type":"line_series","widget":"line_series1","show":false}>ET

Set whether curve point markers are showed:
g) Set line_series1 to set the curve point marker show
ST<{"cmd_code":"set_symbol","type":"line_series","widget":"line_series1","show":true}>ET

h) Set line_series1 to set curve point markers not to show
ST<{"cmd_code":"set_symbol","type":"line_series","widget":"line_series1","show":false}>ET

Set the curve/histogram data:
a) Set the value of line_series1 index 4 to 10:
ST<{"cmd_code":"set_value","type":"line_series","widget":"line_series1","index":4,"value":10}>ET

b) Set the value after line_series1 index 4 to:
ST<{"cmd_code":"set_value","type":"line_series","widget":"line_series1","index":4,
"value":[10,29,69,45,67,34]}>ET
c) Set the line_series1 index value in push mode, that is, append data at the end, and move the previous data forward:
ST<{"cmd_code":"set_value","type":"line_series","widget":"line_series1",
"mode":"push","value":23}>ET

d) Set multiple values of line_series1 indexes in push mode, that is, append data at the end, and move the previous data forward:
ST<{"cmd_code":"set_value","type":"line_series","widget":"line_series1","mode":"push",
"value":[10,29,69,45,67,34]}>ET

e) Set the value of bar_series1 index 4 to 10:
ST<{"cmd_code":"set_value","type":"bar_series","widget":"bar_series1","index":4,"value":10}>ET

f) Set the value after bar_series1 index 4 to:
ST<{"cmd_code":"set_value","type":"bar_series","widget":"bar_series1","index":4,
"value":[10,29,69,45,67,34]}>ET

g) Set the bar_series1 index value in push mode, that is, append data at the end, and move the previous data forward:
ST<{"cmd_code":"set_value","type":"bar_series","widget":"bar_series1",
"mode":"push","value":23}>ET

h) Set multiple values of bar_series1 indexes in push mode, that is, append data at the end, and move the previous data forward:
ST<{"cmd_code":"set_value","type":"bar_series","widget":"bar_series1","mode":"push",
"value":[10,29,69,45,67,34]}>ET

Set index FIFO capacity:
ST<{"cmd_code":"set_capacity","type":"line_series","widget":"line_series1","capacity":15}>ET
ST<{"cmd_code":"set_capacity","type":"bar_series","widget":"bar_series1","capacity":15}>ET

Get index values:
ST<{"cmd_code":"get_value","type":"line_series","widget":"line_series1","index":4}>ET
ST<{"cmd_code":"get_value","type":"bar_series","widget":"bar_series1","index":4}>ET

Get index FIFO capacity:
ST<{"cmd_code":"get_capacity","type":"line_series","widget":"line_series1"}>ET
ST<{"cmd_code":"get_capacity","type":"bar_series","widget":"bar_series1"}>ET

Get index values:
a) Get the value of line_series1 index 4 as 140, of which 0x0004 is the index 4, and 0x430C0000 is the floating point number 140:
Send: ST<{"cmd_code":"get_value","type":"line_series","widget":"line_series1","index":4}>ET
Response: ST<0x10 0xD1 0x00 0x12 line_series1 0x00 0x04 0x43 0x0C 0x00 0x00 >ET
HEX:53 54 3C 10 D1 00 12 6C 69 6E 65 5F 73 65 72 69 65 73 31 00 04 43 0C 00 00 3E 45 54 8D 07

b) Get line_series1 index 9 with a value of 90:
Send: ST<{"cmd_code":"get_value","type":"line_series","widget":"line_series1","index":9}>ET
Response: ST<0x10 0xD1 0x00 0x12 line_series1 0x00 0x09 0x42 0xB4 0x00 0x00 >ET
HEX:53 54 3C 10 D1 00 12 6C 69 6E 65 5F 73 65 72 69 65 73 31 00 09 42 B4 00 00 3E 45 54 AC CD

Get the index capacity value:
a) Get the line_series1 index capacity value of 10:
Send: ST<{"cmd_code":"get_capacity","type":"line_series","widget":"line_series1"}>ET
Response: ST<0x10 0xD2 0x00 0x10 line_series1 0x00 0x00 0x00 0x0A >ET
HEX:53 54 3C 10 D2 00 10 6C 69 6E 65 5F 73 65 72 69 65 73 31 00 00 00 0A 3E 45 54 3F D1

b) Get the line_series1 index capacity value of 19:
Send: ST<{"cmd_code":"get_capacity","type":"line_series","widget":"line_series1"}>ET
Response: ST<0x10 0xD2 0x00 0x10 line_series1 0x00 0x00 0x00 0x13 >ET
HEX:53 54 3C 10 D2 00 10 6C 69 6E 65 5F 73 65 72 69 65 73 31 00 00 00 13 3E 45 54 63 D6

4.25 qr_code

1. Instruction sending:

Instruction	Instruction description	Remarks
set_text	set QR code text content	

Send data description:

Category	Description	Type	Remarks
text	text	text	set QR code text content

2. For example:
Set QR code text content:
ST<{"cmd_code":"set_text","type":"qr","widget":"qr1","text":"http://www.stoneitech.com"}>ET

4.26 pie_slice

1. Instruction sending:

Instruction	Instruction description	Remarks
set_value	set the current value of the pie chart	
set_max	set the maximum value of the pie chart	
set_start_angle	set the starting angle of the pie chart	
set_radius	set the ring thickness radius of the pie chart	
set_show_text	set for showing the pie chart text or not	

Send data description:

Category	Description	Type	Remarks
value	value	uint	Set the current value of the pie chart
max	maximum value	uint	Set the maximum value of the pie chart
start_angle	starting angle	int	Set the starting angle of the pie chart
radius	thickness radius	uint	Set the ring thickness radius of the pie chart
show_text	text	bool	Set for showing the pie chart text or not

2. For example:
Set the widget pie_slice3 value to 60:
ST<{"cmd_code":"set_value","type":"pie_slice","widget":"pie_slice3","value":60}>ET

Set the widget pie_slice3 maximum value to 60:
ST<{"cmd_code":"set_max","type":"pie_slice","widget":"pie_slice3","max":60}>ET

Set the starting angle of the widget pie_slice3 to 60:
ST<{"cmd_code":"set_start_angle","type":"pie_slice","widget":"pie_slice3","angle":60}>ET

Set the widget pie_slice3 loop thickness radius to 60:
ST<{"cmd_code":"set_radius","type":"pie_slice","widget":"pie_slice3","radius":60}>ET

Set whether the widget pie_slice3 shows text:
ST<{"cmd_code":"set_show_text","type":"pie_slice","widget":"pie_slice3","show_text":true}>ET
ST<{"cmd_code":"set_show_text","type":"pie_slice","widget":"pie_slice3","show_text":false}>ET

4.27 slide_indicator/ slide_indicator_arc

1. Instruction sending:

Instruction	Instruction description	Remarks
set_max	set indicator maximum value	
set_size	set indicator size	
set_value	set indicator options	Using with slide_view widget for switching the interface
set_spacing	set indicator spacing	
get_value	get the current value of the indicator (option)	

Send data description:

Category	Description	Type	Remarks
max	maximum value	uint	Set indicator maximum value
size	indicator size	uint	Set indicator size
value	options	uint	Get the current value of the indicator (option), value: 0-(max-1)
spacing	spacing	float	Set the indicator spacing, the value must be greater than 0

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1110	value sent from HMI to MCU	passive	Use get_value to get the current value (option)

Return data description:

Category	Data	Description	Remarks
CMD	0x1110	value sent from HMI to MCU	
LEN	format: widget name + value	data length	
DATA	widget name + value	data content	Int type, the last four bytes of the data part

3. For example:
Set the maximum value of the indicator slide_indicator1
ST<{"cmd_code":"set_max","type":"slide_indicator","widget":"slide_indicator1","max":5}>ET
ST<{"cmd_code":"set_max","type":"slide_indicator","widget":"slide_indicator1","max":7}>ET

Set the size of the indicator slide_indicator1
ST<{"cmd_code":"set_size","type":"slide_indicator","widget":"slide_indicator1","size":5}>ET
ST<{"cmd_code":"set_size","type":"slide_indicator","widget":"slide_indicator1","size":7}>ET

Set the options of the indicator slide_indicator1 (with slide_view to switch the interface)
ST<{"cmd_code":"set_value","type":"slide_indicator","widget":"slide_indicator1","value":0}>ET
ST<{"cmd_code":"set_value","type":"slide_indicator","widget":"slide_indicator1","value":1}>ET

Set the spacing of the indicator slide_indicator1
ST<{"cmd_code":"set_spacing","type":"slide_indicator","widget":"slide_indicator1","spacing":15}>ET
ST<{"cmd_code":"set_spacing","type":"slide_indicator_arc","widget":"slide_ind_arc1","spacing":5}>ET
ST<{"cmd_code":"set_spacing","type":"slide_indicator_arc","widget":"slide_ind_arc1",
"spacing":10}>ET

Get the current value of the indicator (option)
a) The current option of the indicator slide_indicator1 is 0, which is the first:
Send: ST<{"cmd_code":"get_value","type":"slide_indicator","widget":"slide_indicator1"}>ET
Response: ST<0x11 0x10 0x00 0x14 slide_indicator1 0x00 0x00 0x00 0x00>ET
HEX:53 54 3C 11 10 00 14 73 6C 69 64 65 5F 69 6E 64 69 63 61 74 6F 72 31 00 00 00 00 3E 45 54
EB 75

b) The current option of the indicator slide_indicator1 is 5, which is the sixth:
Send: ST<{"cmd_code":"get_value","type":"slide_indicator","widget":"slide_indicator1"}>ET
Response: ST<0x11 0x10 0x00 0x14 slide_indicator1 0x00 0x00 0x00 0x05>ET
HEX:53 54 3C 11 10 00 14 73 6C 69 64 65 5F 69 6E 64 69 63 61 74 6F 72 31 00 00 00 05 3E 45 54
27 75

4.28 slide_view

1. Instruction sending:

Instruction	Instruction description	Remarks
set_view	set the serial number of the current viewing interface page (switch to a specific interface page)	
set_auto_play	set the current viewing interface page to autoplay (automatically switch the interface pages)	
get_view	get the current viewing interface page serial number	

Send data description:

Category	Description	Type	Remarks
index	serial number	uint	set the serial number of the current viewing interface page (switch to a specific interface page)
auto_play	autoplay	uint	Sliding view pages autoplay interval length, 0 cancels autoplay; unit: ms

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1120	current view serial number	passive	The MCU uses the get_view instruction to obtain this value

Return data description:

Category	Data	Description	Remarks
CMD	0x1120	current view serial number	
LEN	format: widget name + length of serial number value	data length	
DATA	widget name + serial number value	data content	Int type, the last four bytes of the data part

3. For example:

Set the current view interface serial number:

```
ST<{"cmd_code":"set_view","type":"slide_view","widget":"slide_view0","index":0}>ET
ST<{"cmd_code":"set_view","type":"slide_view","widget":"slide_view0","index":2}>ET
```

Set the automatic switching interface interval:

```
ST<{"cmd_code":"set_auto_play","type":"slide_view","widget":"slide_view0","auto_play":0}>ET
ST<{"cmd_code":"set_auto_play","type":"slide_view","widget":"slide_view0","auto_play":1000}>ET
ST<{"cmd_code":"set_auto_play","type":"slide_view","widget":"slide_view0","auto_play":3000}>ET
```

Get the current view number:

a) The current page of slide_view0 is 0, which is the first:

```
Send: ST<{"cmd_code":"get_view","type":"slide_view","widget":"slide_view0"}>ET
Response: ST<0x11 0x20 0x00 0x0F slide_view0 0x00 0x00 0x00 0x00>ET
HEX:53 54 3C 11 20 00 0F 73 6C 69 64 65 5F 76 69 65 77 30 00 00 00 00 3E 45 54 65 11
```

b) The current page of slide_view0 is 1, which is the second:

```
Send: ST<{"cmd_code":"get_view","type":"slide_view","widget":"slide_view0"}>ET
Response: ST<0x11 0x20 0x00 0x0F slide_view0 0x00 0x00 0x00 0x01>ET
HEX:53 54 3C 11 20 00 0F 73 6C 69 64 65 5F 76 69 65 77 30 00 00 00 01 3E 45 54 99 10
```

c) The current page of slide_view0 is 6, which is the 7th:

```
Send: ST<{"cmd_code":"get_view","type":"slide_view","widget":"slide_view0"}>ET
Response: ST<0x11 0x20 0x00 0x0F slide_view0 0x00 0x00 0x00 0x06>ET
HEX:53 54 3C 11 20 00 0F 73 6C 69 64 65 5F 76 69 65 77 30 00 00 00 06 3E 45 54 ED 11
```

4.29 slide_menu

1. Instruction sending:

Instruction	Instruction description	Remarks
set_value	set the current menu option	Switch to a specific menu
set_scale	set the current menu scale ratio	Value range: 0.5-1.0
set_align_v	set the current menu alignment state	
get_value	get the current menu option	

Send data description:

Category	Description	Type	Remarks
value	value	uint	Current menu option (specific menu)
scale	scale	float	Set the current menu scale ratio (value ragng: 0.5-1.0)
align_v	alignment	uint	Set the current menu alignment state, value range: 0-3 0: no alignment 1: center alignment 2: top alignment 3: bottom alignment

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1130	deliver the current menu option	passive	Use get_value to get the current menu option

Return data description:

Category	Data	Description	Remarks
CMD	0x1130	send the current menu option from HIM to MCU	
LEN	format: widget name + length of serial number value	data length	
DATA	widget name + serial number value	data content	Int type, the last four bytes of the data part

3. For example:

Set the current menu option for the widget slide_menu0:

```
ST<{"cmd_code":"set_value","type":"slide_menu","widget":"slide_menu0","value":0}>ET
ST<{"cmd_code":"set_value","type":"slide_menu","widget":"slide_menu0","value":2}>ET
```

Set the current menu scale ratio of the widget slide_menu0:

```
ST<{"cmd_code":"set_scale","type":"slide_menu","widget":"slide_menu0","scale":0.5}>ET
ST<{"cmd_code":"set_scale","type":"slide_menu","widget":"slide_menu0","scale":0.8}>ET
ST<{"cmd_code":"set_scale","type":"slide_menu","widget":"slide_menu0","scale":1.0}>ET
```

Set the current menu alignment of the widget slide_menu0:

```
ST<{"cmd_code":"set_align_v","type":"slide_menu","widget":"slide_menu0","align_v":1}>ET
ST<{"cmd_code":"set_align_v","type":"slide_menu","widget":"slide_menu0","align_v":2}>ET
ST<{"cmd_code":"set_align_v","type":"slide_menu","widget":"slide_menu0","align_v":3}>ET
```

Get the current option (serial number) of the sliding menu:
a) The current menu option of the widget slide_menu0 is 0, which is the first menu option:
Send: ST<{"cmd_code":"get_value","type":"slide_menu","widget":"slide_menu0"}>ET
Response: ST<0x11 0x30 0x00 0x0F slide_menu0 0x00 0x00 0x00 0x00>ET
HEX:53 54 3C 11 30 00 0F 73 6C 69 64 65 5F 6D 65 6E 75 30 00 00 00 00 3E 45 54 05 70

b) When the front menu option of the widget slide_menu0 is 8, that is, the ninth menu option:
Send: ST<{"cmd_code":"get_value","type":"slide_menu","widget":"slide_menu1"}>ET
Response: ST<0x11 0x30 0x00 0x0F slide_menu1 0x00 0x00 0x00 0x08>ET
HEX:53 54 3C 11 30 00 0F 73 6C 69 64 65 5F 6D 65 6E 75 31 00 00 00 08 3E 45 54 A9 B3

4.30 TAB tab_button

1. Instruction delivery

Instruction	Instruction description	Remarks
set_value	set current label button value	Switch to a specific tab view
get_value	get current menu button value	Get current tab view options

Send data description:

Category	Description	Type	Remarks
value	whether been selected or not	bool	The value is true: select, false: deselect

2. Data returns:

Return instruction	Return description	Data return type	Remarks
0x1140	get the current view label serial number	Passive	

Return data description:

Category	Data	Description	Remarks
CMD	0x1140	get the current label view number	
LEN	format: widget name + length of value	data length	
DATA	widget name + serial number value	data content	Last byte of data part

3. For example:
Set current label button value
ST<{"cmd_code":"set_value","type":"tab_button","widget":"tab_button4","value":true}>ET
ST<{"cmd_code":"set_value","type":"tab_button","widget":"tab_button4","value":false}>ET

Get current menu button value
a) The current tab button value of the widget tab_button1 is 0, that is, it is not selected:
Send: ST<{"cmd_code":"get_value","type":"tab_button","widget":"tab_button1"}>ET

Response: ST<0x11 0x40 0x00 0x0C tab_button1 0x00>ET
HEX: 53 54 3C 11 40 00 0C 74 61 62 5F 62 75 74 74 6F 6E 31 00 3E 45 54 29 90

b) The current tab button value of the widget tab_button1 is 1, that is, it is selected:
Send: ST<{"cmd_code":"get_value","type":"tab_button","widget":"tab_button1"}>ET
Response: ST<0x11 0x40 0x00 0x0C tab_button1 0x01>ET
HEX: 53 54 3C 11 40 00 0C 74 61 62 5F 62 75 74 74 6F 6E 31 01 3E 45 54 D5 91

4.31 tab_view

1. Instruction delivery

Instruction	Instruction description	Remarks
get_view	get the current label view serial number	

Send data description:

Category	Description	Type	Remarks
view	serial number	uint	Get the current label view serial number

2. Instruction return:

Return instruction	Return description	Data return type	Remarks
0x1150	get the current label view number	passive	sending get_view instruction to acquire the value

Return data description:

Category	Data	Description	Remarks
CMD	0x1150	get the current label view number	
LEN	format: widget name + length of value	data length	
DATA	widget name + serial number value	data content	The last four bytes of the data part

3. For example:
Get the current label view number:
a) The current option number of the tab view tab_view0 is 0, which is the first view
Send: ST<{"cmd_code":"get_view","type":"tab_view","widget":"tab_view0"}>ET
Response: ST<0x11 0x50 0x00 0x0D tab_view0 0x00 0x00 0x00 0x00>ET
HEX:53 54 3C 11 50 00 0D 74 61 62 5F 76 69 65 77 30 00 00 00 00 3E 45 54 FA 1D

b) The current option number 2 of the tab view tab_view0, which is the third view
Send: ST<{"cmd_code":"get_view","type":"tab_view","widget":"tab_view1"}>ET
Response: ST<0x11 0x50 0x00 0x0D tab_view1 0x00 0x00 0x00 0x02>ET
HEX: 53 54 3C 11 50 00 0D 74 61 62 5F 76 69 65 77 31 00 00 00 02 3E 45 54 8E DD

4.32 scroll_view

1. Instruction sending:

Instruction	Instruction description	Remarks
set_xslidable	set whether to allow x- axis direction sliding or not	
set_yslidable	set whether to allow y- axis direction sliding or not	
set_snap_to_page	set whether to align by page when scrolling	
set_move_to_page	set whether or not to turn one page once scrolling	
set_scroll_to	set scroll to the specified offset	
set_scroll_delta_to	set to scroll to specified offset	

Send data description:

Category	Description	Type	Remarks
value	value	bool	Whether to enable or not
xoffset	x-axis scroll offset	int	Negative values representing scrolling to the opposite direction
yoffset	y-axis scroll offset	int	Negative values representing scrolling to the opposite direction

2. For example:

Set whether to allow sliding in the x direction:

```
ST<{"cmd_code":"set_xslidable","type":"scroll_view","widget":"scroll_view2","value":true}>ET
ST<{"cmd_code":"set_xslidable","type":"scroll_view","widget":"scroll_view2","value":false}>ET
```

Set whether to allow sliding in the y direction:

```
ST<{"cmd_code":"set_yslidable","type":"scroll_view","widget":"scroll_view2","value":true}>ET
ST<{"cmd_code":"set_yslidable","type":"scroll_view","widget":"scroll_view2","value":false}>ET
```

Set whether to align by page when scrolling:

```
ST<{"cmd_code":"set_snap_to_page","type":"scroll_view","widget":"scroll_view1","value":true}>ET
ST<{"cmd_code":"set_snap_to_page","type":"scroll_view","widget":"scroll_view1","value":false}>ET
```

Set whether to turn one page at a time when scrolling:

```
ST<{"cmd_code":"set_move_to_page","type":"scroll_view","widget":"scroll_view1","value":true}>ET
ST<{"cmd_code":"set_move_to_page","type":"scroll_view","widget":"scroll_view1","value":false}>ET
```

Set scroll to the specified offset:

a) Set the x-axis of the widget scroll_view2 to scroll to a coordinate of 50 pixels:

```
ST<{"cmd_code":"set_scroll_to","type":"scroll_view","widget":"scroll_view2","xoffset":50}>ET
```

b) Set the y-axis of the widget scroll_view2 to scroll to a coordinate of 50 pixels:

```
ST<{"cmd_code":"set_scroll_to","type":"scroll_view","widget":"scroll_view2","yoffset":50}>ET
```

c) Set the x-axis and y-axis of the widget scroll_view2 to scroll to the coordinates (50,50) pixels:
ST<{"cmd_code":"set_scroll_to","type":"scroll_view","widget":"scroll_view2", "xoffset":50,"yoffset":50}>ET

Set the specified offset for scrolling (send instructions to scroll continuously):
a) Set the x-axis scroll offset of widget scroll_view2 to 50 pixels:
ST<{"cmd_code":"set_scroll_delta_to","type":"scroll_view","widget":"scroll_view2","xoffset":50}>ET

b) Set the y-axis scroll offset of widget scroll_view2 to 50 pixels:
ST<{"cmd_code":"set_scroll_delta_to","type":"scroll_view","widget":"scroll_view2","yoffset":50}>ET

c) Set the x-axis and y-axis of the widget scroll_view2 to scroll with an offset of 50 pixels each:
ST<{"cmd_code":"set_scroll_delta_to","type":"scroll_view","widget":"scroll_view2","xoffset":50, "yoffset":50}>ET

d) Set the x-axis and y-axis of the widget scroll_view2 to scroll -50 pixels offset (reverse scrolling):
ST<{"cmd_code":"set_scroll_delta_to","type":"scroll_view","widget":"scroll_view2","xoffset":-50, "yoffset":-50}>ET

4.33 list_view

1. Instruction sending:

Instruction	Instruction description	Remarks
set_height	set the height of the list item	

Send data description:

Category	Description	Type	Remarks
height	height	uint	Set the height of the list item

2. For example:
Set the height of the list item:
ST<{"cmd_code":"set_height","type":"list_view","widget":"list_view0","height":40}>ET
ST<{"cmd_code":"set_height","type":"list_view","widget":"list_view0","height":60}>ET

4.34 list_view_h

1. Instruction sending:

Instruction	Instruction description	Remarks
set_width	set the width of the list item	
set_spacing	set the spacing of list item	

Send data description:

Category	Description	Type	Remarks
width	height of the list item	uint	Set the height of the list item
spacing	spacing of list item	uint	Set the spacing of list item

2. For example:

Set the width of the list item:

```
ST<{"cmd_code":"set_width","type":"list_view_h","widget":"list_view_h3","width":60}>ET
ST<{"cmd_code":"set_width","type":"list_view_h","widget":"list_view_h3","width":120}>ET
```

Set the spacing of list items:

```
ST<{"cmd_code":"set_spacing","type":"list_view_h","widget":"list_view_h3","spacing":5}>ET
ST<{"cmd_code":"set_spacing","type":"list_view_h","widget":"list_view_h3","spacing":15}>ET
```