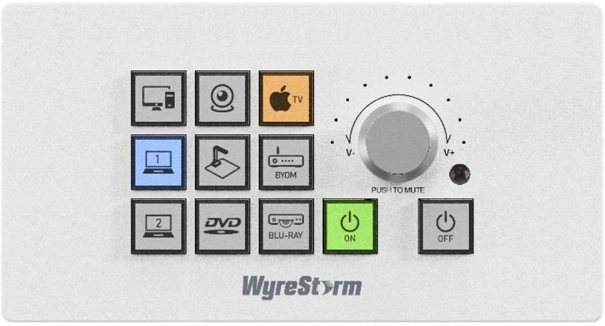
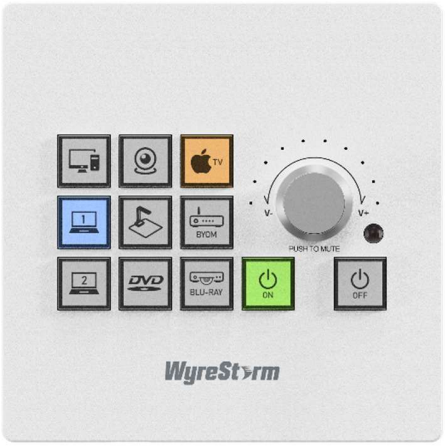
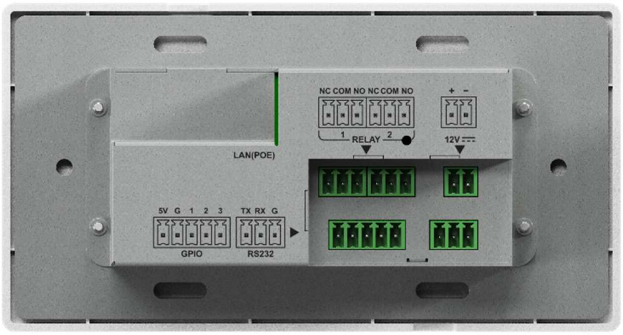
SYN-KEY12

Application Programming Interface

Document Revision	V1.0
Document Date	September 2026
Supported Firmware	Refer to supported product firmware/software for details









Contents

0. Device Communication 4
  0.1 Overview 4
  0.2 Network Connection 4
  0.3 Serial Connection 4
  0.4 Operation Modes (IMPORTANT) 4
  0.5 Command Delimiters for Sent Commands 4
1. Device and Mode Options 5
  1.1 Get Device Model 5
  1.2 Get Device Serial Number 5
  1.3 Get Device MAC Address 5
  1.4 Get Device Uptime 6
  1.5 Set Keypad Working Mode 6
  1.6 Get Keypad Working Mode 6
  1.7 Restore Factory Settings 7
  1.8 System Reboot 7
  1.9 Get the API Commands List 7
  1.10 Get Software Version of the Specified Device 8
2. GPIO & Relay 8
  2.1 Configure GPIO Output Type 8
  2.2 Get GPIO Output Type 8
  2.3 Configure GPIO Output Channel Pull-Up Resistor and Output State 9
  2.4 Get GPIO Output Channel Pull-Up Resistor and Output State 9
  2.5 Set Relay Status 9
  2.6 Get Relay Status 10
  2.7 Configure Relay Boot Mode 10
  2.8 Get Relay Boot Mode 10
3. Volume and Knob LED 11
  3.1 Increase Volume by One Step 11
  3.2 Decrease Volume by One Step 11
  3.3 Set Output Volume Directly 11
  3.4 Set Output Volume Step Size 12
  3.5 Get Output Volume Step Size 12
  3.6 Get Output Volume Size 12
  3.9 Set Knob LED Light 13
  3.8 Get Knob LED Light 13
4. Button Status 13
  4.1 Get Button Status 13
  4.2 Configure Button Color 14
  4.3 Get Button Color 14
  4.4 Set the Button LED On/Off 14
  4.5 Get the Button LED On/Off Status 15
5. Alias Name 15
  5.1 Set Port Alias 15
  5.2 Get Port Alias 15
  5.3 Set Device Name 16
  5.4 Get Device Name 16
6. LED Status 16
  6.1 Set LED Idle Timeout (Turn-Off Time) 16

6.2 Get LED Idle Timeout Turn-Off Time 17
6.3 Enable or Disable LED Idle Auto Turn-Off Function 17
6.4 Get LED Idle Auto Turn-Off Function Status 17
7. IR Learning 18
7.1 Control IR Learning Process 18
8. Network 18
8.1 Set Static IP 18
8.2 Set DHCP IP Address 19
8.3 Get IP Address 19
8.4 Set HTTPS Service Status 20
8.5 Get HTTPS Service Status 20
8.6 Set TELNETS Service Status 20
8.7 Get TELNET Service Status 21
8.8 Enable/Disable SSH API Service 21
8.9 Get SSH Service Status 21
8.10 Set API Access Password 22

0. Device Communication

0.1 Overview

The following contains the connection and commands to control the SYN-KEY12. By following the content contained in this document, the device can be controlled and configured via 3rd party control system.

⚠ IMPORTANT NOTE: Due to differences between WyreStorm device model versions within a series, some commands have different parameters based on the model and version. These differences are noted where applicable and should be followed, as sending an incorrect parameter may cause the unit to lock up and become inoperable.

0.2 Network Connection

By default, the IP address is assigned by the DHCP server. If the DHCP server is not available the unit will use a link local IP address based on its MAC address, if the unit was not already configured via the Web interface.

Default communication port number is 23.

0.3 Serial Connection

The SYN-KEY12 uses a 3-pin RS-232 with no hardware flow control. Most control systems and computers are DTE where pin 2 is RX, this can vary from device to device. Refer to the documentation for the connected device for pin functionally to ensure that the correct connections can be made.

RS-232 Settings
Baud Rate: 9600
Data Bits: 8bits
Parity: None
Stop Bits: 1bit
Flow Control: None



WyreStorm Connector			3rd Party device
Pin 1	TX (Transmit)	---> To --->	RX (Receive)
Pin 2	RX (Receive)	---> To --->	TX (Transmit)
Pin 3	G (Ground)	---> To --->	G (Ground)

0.4 Operation Modes (IMPORTANT)

SYN-KEY12 has 2 operation modes one is STANDALONE in which you completely configure it via Web interface and second one is CONTROLLER in which it works as a keypad that can be custom integrated via any 3rd party control system. In CONTROLLER mode you programmatically set LED button color and feedback, as well as Knob volume level LED status. These modes you can set up via API command explained in chapter 1.5.

0.5 Command Delimiters for Sent Commands

All commands should be delimited with carriage return and line feed.

Accepted delimiter characters are:

Character	Shorthand	Hex Notation	Escape Notation	Decimal Notation
Line Feed	LF	0A	\n	10
Carriage Return + Line Feed	CR LF	0D 0A	\r\n	13 10

Please note, most 3rd party control software will either append these characters automatically or an option to specify them will be present. It is important that the last delimited character is LF and not CR.

1. Device and Mode Options

1.1 Get Device Model

Use this command to retrieve the device's model information.

Command Structure **GET DEVICE_MODEL<CR><LF>**	Parameters: model = the device's model identifier
Response Structure **DEVICE_MODEL <model><CR><LF>**	
Command Example **GET DEVICE_MODEL<CR><LF>**	
Response Example **DEVICE_MODEL SYN-KEY12-US<CR><LF>**	
Additional Notes: Queries and returns the device's model information.	

1.2 Get Device Serial Number

Retrieves the device serial number.

Command Structure **GET SYS_SN<CR><LF>**	Parameters: <sn> = Device serial number
Response Structure **SYS_SN <sn><CR><LF>**	
Command Example **GET SYS_SN<CR><LF>**	
Response Example **SYS_SN 112233445566<CR><LF>**	
Additional Notes:	

1.3 Get Device MAC Address

Use this command to retrieve the device's MAC address.

Command Structure **GET MACADDR<CR><LF>**	Parameters: <mac> = Device MAC address
Response Structure **MACADDR <mac><CR><LF>**	
Command Example **GET MACADDR<CR><LF>**	
Response Example **MACADDR 00:90:6F:46:80:A0<CR><LF>**	
Additional Notes:	

1.4 Get Device Uptime

Use this command to retrieve the device's uptime.

Command Structure **GET UPTIME<CR><LF>**	Parameters: DAYS = system uptime in days HH = hours (00–23) MM = minutes (00–59) SS = seconds (00–59)
Response Structure **UPTIME <days>,<HH>:<MM>:<SS><CR><LF>**	
Command Example **GET UPTIME<CR><LF>**	
Response Example **UPTIME 12,5:33:21<CR><LF>**	
Additional Notes:	

1.5 Set Keypad Working Mode

Set the keypad's working mode.

Command Structure **SET DEVICE_WORK_MODE <prm><CR><LF>**	Parameters: prm = {STANDALONE, CONTROLLER} // default: STANDALONE
Response Structure **DEVICE_WORK_MODE <prm><CR><LF>**	
Command Example **SET DEVICE_WORK_MODE STANDALONE<CR><LF>**	
Response Example **DEVICE_WORK_MODE STANDALONE<CR><LF>**	
Additional Notes: Configure the working mode of the keypad. STANDALONE mode allows the keypad to function independently. CONTROLLER mode makes it act as a keypad for third-party control system integration.	

1.6 Get Keypad Working Mode

Retrieve the keypad working mode.

Command Structure **GET DEVICE_WORK_MODE<CR><LF>**	Parameters: prm = {STANDALONE, CONTROLLER}
Response Structure **DEVICE_WORK_MODE<prm><CR><LF>**	
Command Example **GET DEVICE_WORK_MODE<CR><LF>**	
Response Example **DEVICE_WORK_MODE STANDALONE<CR><LF>**	
Additional Notes:	

1.7 Restore Factory Settings

Restores the device to its factory default settings.

Command Structure **RESET<CR><LF>**	Parameters:
Response Structure **RESET<CR><LF>**	
Command Example **RESET<CR><LF>**	
Response Example **RESET<CR><LF>**	
Additional Notes: This command resets all board configurations.	

1.8 System Reboot

Reboot the device.

Command Structure **REBOOT<CR><LF>**	Parameters:
Response Structure **REBOOT<CR><LF>**	
Command Example **REBOOT<CR><LF>**	
Response Example **REBOOT<CR><LF>**	
Additional Notes: Reboots the system.	

1.9 Get the API Commands List

Retrieve the list of APIs supported by the device.

Command Structure **help<CR><LF>**	Parameters:
Response Structure **xxxx<CR><LF>**	
Command Example **help<CR><LF>**	
Response Example **xxxx<CR><LF>**	
Additional Notes:	

1.10 Get Software Version of the Specified Device

Retrieve the software version of the specified device.

Command Structure **GET VER<CR><LF>**	Parameters: prm = {...} // Contents depend on the device's firmware version
Response Structure **VER <prm><CR><LF>**	
Command Example **GET VER<CR><LF>**	
Response Example **VER V1.0.0<CR><LF>**	
Additional Notes: Retrieves the firmware version of all modules	

2. GPIO & Relay

2.1 Configure GPIO Output Type

Sets the output type for a specified GPIO port.

Command Structure **SET GPIO_TYPE <port_id> <type><CR><LF>**	Parameters: PORT_ID = GPIO port number (1, 2, 3) TYPE = output type (DIGITAL-OUT, IR-OUT)
Response Structure **GPIO_TYPE <port_id> <type><CR><LF>**	
Command Example **SET GPIO_TYPE 1 DIGITAL-OUT<CR><LF>**	
Response Example **GPIO_TYPE 1 DIGITAL-OUT<CR><LF>**	
Additional Notes:	

2.2 Get GPIO Output Type

Returns the output type of a specified GPIO port.

Command Structure **GET GPIO_TYPE <port_id><CR><LF>**	Parameters: PORT_ID = GPIO port number (1, 2, 3) TYPE = output type (DIGITAL_OUT, IR_OUT)
Response Structure **GPIO_TYPE <port_id> <type><CR><LF>**	
Command Example **GET GPIO_TYPE 1<CR><LF>**	
Response Example **GPIO_TYPE 1 DIGITAL-OUT<CR><LF>**	
Additional Notes: This command reads the GPIO port's output type. The response includes the port ID and its output type.	

2.3 Configure GPIO Output Channel Pull-Up Resistor and Output State

Sets the pull-up resistor and output state for a GPIO output channel.

Command Structure **SET GPIO_OUT_CTL <ch> <pullup> <status><CR><LF>** Response Structure **GPIO_OUT_CTL <ch> <pullup> <status><CR><LF>** Command Example **SET GPIO_OUT_CTL 1 ON HIGH<CR><LF>** Response Example **GPIO_OUT_CTL 1 ON HIGH<CR><LF>**	Parameters: CH = {1, 2, 3}: GPIO output channel number PULLUP = {ON, OFF}: ON means the 5V pull-up resistor is connected; OFF means it is disconnected STATUS = {LOW, HIGH}: LOW means the pull-up is active; HIGH means the output is in high-impedance state
Additional Notes:	

2.4 Get GPIO Output Channel Pull-Up Resistor and Output State

Reads the pull-up resistor setting and the current output state of the specified GPIO output channel.

Command Structure **GET GPIO_OUT_CTL <ch> <CR><LF>** Response Structure **GPIO_OUT_CTL <ch> <pullup> <status> <CR><LF>** Command Example **GET GPIO_OUT_CTL 1<CR><LF>** Response Example **GPIO_OUT_CTL 1 ON HIGH<CR><LF>**	Parameters: CH = {1, 2, 3}: GPIO output channel number PULLUP = {ON, OFF}: ON = connect 5V pull-up resistor; OFF = disconnect STATUS = {LOW, HIGH}: LOW indicates the pull-up is active; HIGH indicates the pin is high-impedance
Additional Notes: Use this command to monitor the GPIO output status.	

2.5 Set Relay Status

Use this command to set the relay state for a specified port.

Command Structure **SET RELAY_STATUS <port_id> <state><CR><LF>** Response Structure **RELAY_STATUS <port_id> <state><CR><LF>** Command Example **SET RELAY_STATUS 1 NC<CR><LF>** Response Example **RELAY_STATUS 1 NC<CR><LF>**	Parameters: <port_id> = {1, 2} - Port number <state> = {NC, NO} - NC (Normally Closed): NC pin is connected to the COM pin. NO (Normally Open): NO pin is connected to the COM pin
Additional Notes:	

2.6 Get Relay Status

Use this command to read the RELAY status for a specified port.

Command Structure **GET RELAY_STATUS <port_id><CR><LF>** Response Structure **RELAY_STATUS <port_id> <state><CR><LF>** Command Example **GET RELAY_STATUS 1<CR><LF>** Response Example **RELAY_STATUS 1 NC<CR><LF>**	Parameters: <port_id> = {1, 2} - Port number <status> = {NC, NO} - NC: Normally Closed — NC pin is connected to COM pin; NO: Normally Open — NO pin is connected to COM pin
Additional Notes:	

2.7 Configure Relay Boot Mode

Set the relay boot mode.

Command Structure **SET RELAY_CTL_BOOT_MODE <num> <prm><CR><LF>** Response Structure **RELAY_CTL_BOOT_MODE <num> <prm><CR><LF>** Command Example **SET RELAY_CTL_BOOT_MODE 1 SAVED<CR><LF>** Response Example **RELAY_CTL_BOOT_MODE 1 SAVED<CR><LF>**	Parameters: num = {1–2} (relay number) prm = {DEFAULT, SAVED} (boot mode)
Additional Notes:	

2.8 Get Relay Boot Mode

Get the relay boot mode.

Command Structure **GET RELAY_CTL_BOOT_MODE<num> <CR><LF>** Response Structure **RELAY_CTL_BOOT_MODE <num> <prm><CR><LF>** Command Example **GET RELAY_CTL_BOOT_MODE 1<CR><LF>** Response Example **RELAY_CTL_BOOT_MODE 1 SAVED<CR><LF>**	Parameters: num = {1~2} (relay number) prm = {DEFAULT, SAVED} (boot mode)
Additional Notes:	

3. Volume and Knob LED

3.1 Increase Volume by One Step

Increase the output volume by one step and return the new value.

Command Structure **SET VOL_LEVEL_INC<CR><LF>**	Parameters: prm = {0-100}
Response Structure **VOL_LEVEL_INC <prm><CR><LF>**	
Command Example **SET VOL_LEVEL_INC<CR><LF>**	
Response Example **VOL_LEVEL_INC 60<CR><LF>**	
Additional Notes: Increases the volume by one step and returns the actual value.	

3.2 Decrease Volume by One Step

Decreases the output volume by one step and returns the new volume value.

Command Structure **SET VOL_LEVEL_DEC<CR><LF>**	Parameters: prm = integer in range 0–100
Response Structure **VOL_LEVEL_DEC <prm><CR><LF>**	
Command Example **SET VOL_LEVEL_DEC<CR><LF>**	
Response Example **VOL_LEVEL_DEC 40<CR><LF>**	
Additional Notes: Decreases the volume by one step and returns the current volume value.	

3.3 Set Output Volume Directly

Set the output audio volume level.

Command Structure **SET VOL_LEVEL_DATA <prm><CR><LF>**	Parameters: prm = {0~100}; // 0 mutes the output
Response Structure **VOL_LEVEL_DATA <prm><CR><LF>**	
Command Example **SET VOL_LEVEL_DATA 50<CR><LF>**	
Response Example **VOL_LEVEL_DATA 50<CR><LF>**	
Additional Notes:	

3.4 Set Output Volume Step Size

Set the step size for output volume adjustment.

Command Structure **SET VOL_LEVEL_STEP <prm><CR><LF>**	Parameters: prm = integer from 1 to 20
Response Structure **VOL_LEVEL_STEP <prm><CR><LF>**	
Command Example **SET VOL_LEVEL_STEP 10<CR><LF>**	
Response Example **VOL_LEVEL_STEP 10<CR><LF>**	
Additional Notes:	

3.5 Get Output Volume Step Size

Retrieve the current volume adjustment step size.

Command Structure **GET VOL_LEVEL_STEP <CR><LF>**	Parameters: prm = {1~20} (step size value: 1 to 20)
Response Structure **VOL_LEVEL_STEP <prm><CR><LF>**	
Command Example **GET VOL_LEVEL_STEP<CR><LF>**	
Response Example **VOL_LEVEL_STEP 10<CR><LF>**	
Additional Notes:	

3.6 Get Output Volume Size

Retrieve the current output volume level.

Command Structure **GET VOL_LEVEL_DATA <CR><LF>**	Parameters: prm = {0~100}; // 0 = output muted
Response Structure **VOL_LEVEL_DATA <prm><CR><LF>**	
Command Example **GET VOL_LEVEL_DATA<CR><LF>**	
Response Example **VOL_LEVEL_DATA 50<CR><LF>**	
Additional Notes:	

3.9 Set Knob LED Light

Configure the knob LED light.

Command Structure **SET KNOB_MUTE_LED <prm><CR><LF>**	Parameters: prm = {OFF, ON} // ON: all red LEDs; OFF: normal state
Response Structure **KNOB_MUTE_LED <prm><CR><LF>**	
Command Example **SET KNOB_MUTE_LED ON<CR><LF>**	
Response Example **KNOB_MUTE_LED ON<CR><LF>**	
Additional Notes:	

3.8 Get Knob LED Light

Get the knob LED light state.

Command Structure **GET KNOB_MUTE_LED<CR><LF>**	Parameters: prm = {OFF,ON} // ON: All red LEDs; OFF: Normal state
Response Structure **KNOB_MUTE_LED <prm><CR><LF>**	
Command Example **GET KNOB_MUTE_LED<CR><LF>**	
Response Example **KNOB_MUTE_LED ON<CR><LF>**	
Additional Notes:	

4. Button Status

4.1 Get Button Status

Get button status

Command Structure **GET BTN_STATUS <num><CR><LF>**	Parameters: num = {1–12} status = {PUSH, RELEASE}
Response Structure **BTN_STATUS <num> <status><CR><LF>**	
Command Example **GET BTN_STATUS 1<CR><LF>**	
Response Example **BTN_STATUS 1 PUSH<CR><LF>**	
Additional Notes:	

4.2 Configure Button Color

Set button color

Command Structure **SET BTN_COLOR <num> <color><CR><LF>**	Parameters: num = {1–11} (button number) color = {0xRRGGBB} (hex color in RRGGBB format, e.g. 0x00FF00)
Response Structure **BTN_COLOR <num> <color><CR><LF>**	
Command Example **SET BTN_COLOR 1 0X00FF00<CR><LF>**	
Response Example **BTN_COLOR 1 0X00FF00<CR><LF>**	
Additional Notes:	

4.3 Get Button Color

Retrieve the color of a button.

Command Structure **GET BTN_COLOR <num> <CR><LF>**	Parameters: num = {1–11} color = {0xRRGGBB} (hexadecimal RGB format)
Response Structure **BTN_COLOR <num> <color><CR><LF>**	
Command Example **GET BTN_COLOR 1<CR><LF>**	
Response Example **BTN_COLOR 1 0X00FF00<CR><LF>**	
Additional Notes:	

4.4 Set the Button LED On/Off

Set the button LED state

Command Structure **SET BTN_LED <num> <prm><CR><LF>**	Parameters: num = {1–11}; <prm> = {OFF, ON}
Response Structure **BTN_LED <num> <prm><CR><LF>**	
Command Example **SET BTN_LED 1 ON<CR><LF>**	
Response Example **BTN_LED 1 ON<CR><LF>**	
Additional Notes:	

4.5 Get the Button LED On/Off Status

Retrieve the on/off status of a button LED.

Command Structure **GET BTN_LED <num> <CR><LF>**	Parameters: num = {1–11} — button number; <prm> = {OFF, ON}
Response Structure **BTN_LED <num> <prm><CR><LF>**	
Command Example **GET BTN_LED 1<CR><LF>**	
Response Example **BTN_LED 1 ON<CR><LF>**	
Additional Notes:	

5. Alias Name

5.1 Set Port Alias

Use this command to set the alias for a specified port.

Command Structure **SET PORT_ALIAS <port_type> <port_alias><CR><LF>**	Parameters: port_type = {GPIO1, GPIO2, GPIO3, RELAY1, RELAY2} port_alias = {xxxx}: alias string of up to 16 bytes
Response Structure **PORT_ALIAS <port_type> <port_alias><CR><LF>**	
Command Example **SET PORT_ALIAS GPIO1 ABCDEF<CR><LF>**	
Response Example **PORT_ALIAS GPIO1 ABCDEF<CR><LF>**	
Additional Notes:	

5.2 Get Port Alias

Gets the alias for a specified port.

Command Structure **GET PORT_ALIAS <port_type><CR><LF>**	Parameters: <port_type> = {GPIO1, GPIO2, GPIO3, RELAY1, RELAY2} <port_alias> = {xxxx}: alias of up to 16 bytes
Response Structure **PORT_ALIAS <port_type> <port_alias><CR><LF>**	
Command Example **GET PORT_ALIAS GPIO1<CR><LF>**	
Response Example **PORT_ALIAS GPIO1 ABCDEF<CR><LF>**	
Additional Notes: The example shows how to use the command to get the alias of the GPIO1 port; the response returns the port type and its alias.	

5.3 Set Device Name

Set the user-defined device name. If not set, the device uses the default name format: <model>-<last 6 digits of MAC (uppercase)>, e.g., ModelX-431BAC.

Command Structure **`SET DEVICE_NAME <name>`**	Parameters: NAME = User-defined device name Length: 1-31 ASCII characters Allowed characters: A-Z, a-z, 0-9, hyphen (-), underscore (_), space Must start and end with a letter or number
Response Structure **`DEVICE_NAME <name>`**	
Command Example **`SET DEVICE_NAME MS340-000-431BAC`**	
Response Example **`DEVICE_NAME MS340-000-431BAC`**	
Additional Notes:	

5.4 Get Device Name

Read the current device name. If not set, the default name is returned in the format: <model>-<last 6 digits of MAC (uppercase)>.

Command Structure **`GET DEVICE_NAME`**	Parameters: NAME = user-defined device name 1-31 ASCII characters Allowed characters: A-Z, a-z, 0-9, hyphen (-), underscore (_), space Must begin and end with a letter or number
Response Structure **`DEVICE_NAME <name>`**	
Command Example **`GET DEVICE_NAME`**	
Response Example **`DEVICE_NAME MS340-000-431BAC`**	
Additional Notes:	

6. LED Status

6.1 Set LED Idle Timeout (Turn-Off Time)

Sets the timeout (in seconds) after which the LED indicator automatically turns off if the system remains in the IDLE state with no operations.

Command Structure **`SET KNOB_LED_TIMEOUT <seconds><CR><LF>`**	Parameters: SECONDS = Idle timeout duration in seconds Range: 1–60
Response Structure **`KNOB_LED_TIMEOUT <seconds><CR><LF>`**	
Command Example **`SET KNOB_LED_TIMEOUT 30<CR><LF>`**	
Response Example **`KNOB_LED_TIMEOUT 30<CR><LF>`**	
Additional Notes: IDLE means the system has no operation events (e.g., key presses, touch actions, or control commands) during the specified period	

6.2 Get LED Idle Timeout Turn-Off Time

Retrieve the current timeout value used to automatically turn off the LED indicator after the system remains in an IDLE state with no activity.

Command Structure **GET KNOB_LED_TIMEOUT<CR><LF>**	Parameters: SECONDS = Idle timeout duration (unit: seconds). Range: 1–60
Response Structure **KNOB_LED_TIMEOUT <seconds><CR><LF>**	
Command Example **GET KNOB_LED_TIMEOUT<CR><LF>**	
Response Example **KNOB_LED_TIMEOUT 30<CR><LF>**	
Additional Notes: IDLE means no user activity (for example: key presses, touch input, or control commands) occurs during the specified period.	

6.3 Enable or Disable LED Idle Auto Turn-Off Function

Set whether the LED automatically turns off when the system is idle. If set to OFF, the LED remains on during idle.

Command Structure **SET KNOB_LED_AUTO_OFF <prm><CR><LF>**	Parameters: PRM = Whether the LED automatic turn-off is enabled. ON means enabled, OFF means disabled
Response Structure **KNOB_LED_AUTO_OFF <prm><CR><LF>**	
Command Example **SET KNOB_LED_AUTO_OFF ON<CR><LF>**	
Response Example **KNOB_LED_AUTO_OFF ON<CR><LF>**	
Additional Notes:	

6.4 Get LED Idle Auto Turn-Off Function Status

Query whether the automatic LED turn-off function during system idle is enabled.

Command Structure **GET KNOB_LED_AUTO_OFF<CR><LF>**	Parameters: PRM = current auto turn-off status. ON = enabled, OFF = disabled
Response Structure **KNOB_LED_AUTO_OFF <\prm><CR><LF>**	
Command Example **GET KNOB_LED_AUTO_OFF<CR><LF>**	
Response Example **KNOB_LED_AUTO_OFF OFF<CR><LF>**	
Additional Notes:	

7. IR Learning

7.1 Control IR Learning Process

Controls the device's infrared (IR) code learning: START begins learning (an optional timeout can be set), STOP cancels learning.

Command Structure **ACT IR_LEARN <action> [<timeout>]<CR><LF>**	Parameters: action = {START, STOP}; START = begin IR learning, STOP = cancel learning TIMEOUT = 1-60 (seconds). Only valid with START. Default is 10 seconds.
Response Structure **IR_LEARN <action><CR><LF>**	
Command Example **ACT IR_LEARN START 5<CR><LF>**	
Response Example **IR_LEARN START 5<CR><LF>**	
Additional Notes: Use START to begin IR learning with an optional timeout (in seconds). Use STOP to end the learning process at any time.	

8. Network

8.1 Set Static IP

Set a static IPv4 address.

Command Structure **SET IPADDR STATIC IP4ADDR <xx.xx.xx.xx> NETMASK <xx.xx.xx.xx> [GATEWAY <xx.xx.xx.xx>]<CR><LF>**	Parameters:
Response Structure **IPADDR STATIC IP4ADDR <xx.xx.xx.xx> NETMASK <xx.xx.xx.xx> [GATEWAY <xx.xx.xx.xx>]<CR><LF>**	
Command Example **SET IPADDR STATIC IP4ADDR 192.168.11.243 NETMASK 255.255.255.0 GATEWAY 192.168.11.1<CR><LF>**	
Response Example **IPADDR STATIC IP4ADDR 192.168.11.243 NETMASK 255.255.255.0 GATEWAY 192.168.11.1<CR><LF>**	
Additional Notes: Sets the static IP address, subnet mask, and gateway. If the GATEWAY parameter is not provided, the default gateway 0.0.0.0 is used. If the GATEWAY parameter is provided, verify that IP4ADDR, NETMASK, and GATEWAY are consistent.	

8.2 Set DHCP IP Address

Set the DHCP IP address.

Command Structure **SET IPADDR DHCP<CR><LF>**	Parameters:
Response Structure **IPADDR DHCP<CR><LF>**	
Command Example **SET IPADDR DHCP<CR><LF>**	
Response Example **IPADDR DHCP<CR><LF>**	
Additional Notes:	

8.3 Get IP Address

Retrieve the device's IP address.

Command Structure **GET IPADDR<CR><LF>**	Parameters:
Response Structure **IPADDR DHCP IP4ADDR <ipv4> NETMASK <netmask> GATEWAY <gateway><CR><LF>** **"OR"** **IPADDR STATIC IP4ADDR <ipv4> NETMASK <netmask> GATEWAY <gateway><CR><LF>**	
Command Example **GET IPADDR<CR><LF>**	
Response Example **IPADDR DHCP IP4ADDR 192.168.11.243 NETMASK 255.255.255.0 GATEWAY 192.168.11.1<CR><LF>** **"OR"** **IPADDR STATIC IP4ADDR 192.168.11.243 NETMASK 255.255.255.0 GATEWAY 192.168.2.1<CR><LF>**	
Additional Notes:	

8.4 Set HTTPS Service Status

Enable or disable the HTTPS service.

Command Structure **SET HTTPS <prm><CR><LF>**	Parameters: prm = {on, off} — 'on' to enable the HTTPS service, 'off' to disable it.
Response Structure **HTTPS <prm><CR><LF>**	
Command Example **SET HTTPS on<CR><LF>**	
Response Example **HTTPS on<CR><LF>**	
Additional Notes:	

8.5 Get HTTPS Service Status

Retrieve the current status of the HTTPS service.

Command Structure **GET HTTPS<CR><LF>**	Parameters: prm = {on, off} — 'on' means the HTTPS service is enabled/running; 'off' means the HTTPS service is disabled/stopped.
Response Structure **HTTPS<CR><LF>**	
Command Example **GET HTTPS<CR><LF>**	
Response Example **HTTPS on<CR><LF>**	
Additional Notes:	

8.6 Set TELNETS Service Status

TELNETS stands for Telnet-TLS (Telnet over TLS). 'on' enables TELNETS (TLS-protected Telnet). 'off' enables plain Telnet.

Command Structure **SET TELNETS <prm><CR><LF>**	Parameters: prm = {on, off} // 'on' enables TELNETS (Telnet-TLS); 'off' enables plain Telnet.
Response Structure **TELNETS <prm><CR><LF>**	
Command Example **SET TELNETS on<CR><LF>**	
Response Example **TELNETS on<CR><LF>**	
Additional Notes:	

8.7 Get TELNET Service Status

Use this command to enable or disable Telnet over TLS (TELNETS).

Command Structure **GET TELNETS<CR><LF>**	Parameters: prm = {on, off}; 'on' enables the Telnet-TLS service; 'off' disables it.
Response Structure **TELNETS <prm><CR><LF>**	
Command Example **GET TELNETS<CR><LF>**	
Response Example **TELNETS on<CR><LF>**	
Additional Notes:	

8.8 Enable/Disable SSH API Service

Use this command to enable or disable the SSH API service.

Command Structure **SET SSH <prm><CR><LF>**	Parameters: prm = {ON, OFF}: ON enables the SSH API service; OFF disables the SSH API service.
Response Structure **SSH <prm><CR><LF>**	
Command Example **SET SSH ON<CR><LF>**	
Response Example **SSH ON<CR><LF>**	
Additional Notes: When disabled, the device stops listening on the SSH API port (usually 22). Disabling the SSH API may prevent secure remote maintenance.	

8.9 Get SSH Service Status

Retrieves whether the SSH service is ON or OFF.

Command Structure **GET SSH<CR><LF>**	Parameters: prm = {ON, OFF}: ON means the SSH service is enabled; OFF means the SSH service is disabled.
Response Structure **SSH <prm><CR><LF>**	
Command Example **GET SSH<CR><LF>**	
Response Example **SSH OFF<CR><LF>**	
Additional Notes: Can be used together with SET SSH to confirm the result of a status change.	

8.10 Set API Access Password

Set or update the API authentication password used for Telnet and SSH. You must provide the current (old) password for verification.

Command Structure

```
SET API_PWD <oldpwd>
  <newpwd><CR><LF>
```

Response Structure

```
{Password wrong|User name or password wrong|Password
changed successfully}<CR><LF>
```

Command Example

```
SET API_PWD 12345678 NewP@ssw0rd!<CR><LF>
```

Response Example

```
Password changed successfully<CR><LF>
```

Parameters:
<oldpwd>: The currently active API password, used to verify permission.

<newpwd>: The new password (8–24 characters).
Allowed characters: A–Z, a–z, 0–9, and the following special symbols:
!@#$%^&*()_+-=[]{};:'"|,.<>/?
Spaces, non-ASCII characters, or any other unspecified symbols are not allowed.

Additional Notes: The response contains only the result text; it does not echo the plaintext password or the command.
If the old password is incorrect, the device returns "Password wrong", or in some implementations "User name or password wrong".
The password change takes effect immediately for new sessions. Existing sessions may remain active depending on the device implementation.