SYN-CTL-FL10

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 Command Delimiters for Sent Commands 4
1. System Info and Setup 5
  1.1 Get Temperature 5
  1.2 Help 5
  1.3 Get Device Model 5
  1.4 Set Device Name 6
  1.5 Get Device Name 6
  1.6 Reset 6
  1.7 Reboot 7
  1.8 Get Device Serial Number 7
  1.9 Get Device Firmware Version 7
  1.10 Get Uptime 8
  1.11 Get Device IP Address 8
  1.12 Keepalive 8
  1.13 Set Keepalive Timeout 9
  1.14 Get Keepalive Timeout 9
2. Telnet and SSH Config 9
  2.1 Set TELNET_AUTH_FN 9
  2.2 Get TELNET_AUTH_FN 10
  2.3 Set Telnet Port 10
  2.4 Get Telnet Port 10
  2.5 Set SSH 11
  2.6 Get SSH 11
  2.7 Set Telnet and SSH API Password 11
3. UART and Serial Setup 12
  3.1 Set UART Mode 12
  3.2 Get UART Mode 12
  3.3 Set UART_CFG 12
  3.4 Get UART_CFG 13
  3.5 Set UART Flow 13
  3.6 Get UART Flow 13
  3.7 Set UART Proxy Function 14
  3.8 Get UART_PROXY_FN 14
  3.9 Set UART Proxy Port 14
  3.10 Get UART Proxy Port 15
  3.11 ACT UART_SEND 15
  3.12 Set Serial Mode 16
  3.13 Get Serial Mode 16
4. IR Configuration 17
  4.1 ACT IR_SEND 17
  4.2 ACT IR_SEND_X - Send Infrared Data (Extended) 18
  4.3 Set IR_PROXY_FN 18
  4.4 Get IR_PROXY_FN 19
  4.5 Set IR Proxy Port 19

4.6 Get IR Proxy Port....19
5. GPIO and Relay Commands....20
5.1 Set GPIO_CFG....20
5.2 Get GPIO_CFG....20
5.3 Set GPIO Output Control....20
5.4 Get GPIO_OUT_CTL....21
5.5 GPIO IN Status....21
5.6 Get GPIO IN Status....21
5.7 Set Relay....22
5.8 Get Relay....22
5.9 Set Relay Control Boot Mode....22
5.10 Get Relay Control Boot Mode....23

0. Device Communication

0.1 Overview

The following contains the connection and commands to control the SW-620-TX-W. 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 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. System Info and Setup

1.1 Get Temperature

Retrieve the temperature value from the temperature sensor for the selected target firmware version.

Command Structure **GET TEMP[CR/LF]**	Parameters: <prm> = temperature in Celsius
Response Structure **TEMP[<prm>℃,<prm>℃][CR/LF]**	
Command Example **GET TEMP[CR/LF]**	
Response Example **TEMP[46℃,63℃][CR/LF]**	
Additional Notes:	

1.2 Help

Retrieve a 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:	

1.3 Get Device Model

Get the device model

Command Structure **GET DEVICE_MODEL [CR/LF]**	Parameters: <model> = Model name of the device
Response Structure **DEVICE_MODEL <model>[CR/LF]**	
Command Example **GET DEVICE_MODEL[CR/LF]**	
Response Example **DEVICE_MODEL MyDeviceModelName[CR/LF]**	
Additional Notes:	

1.4 Set Device Name

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

Command Structure **SET DEVICE_NAME <name>[CR/LF]**	Parameters: <name> = custom device name, 1-31 ASCII characters, allowed characters: A–Z, a–z, 0–9, hyphen(-), underscore(_), space, must start and end with a letter or digit
Response Structure **DEVICE_NAME <name>[CR/LF]**	
Command Example **SET DEVICE_NAME SYN-CTL-HUB-001[CR/LF]**	
Response Example **DEVICE_NAME SYN-CTL-HUB-001[CR/LF]**	
Additional Notes:	

1.5 Get Device Name

Read the currently configured device name. If not set, returns the default name in the format: <model>-<last 6 characters of MAC (uppercase)>.

Command Structure **GET DEVICE_NAME[CR/LF]**	Parameters: <name> = Custom device name, 1-31 ASCII characters, allowed characters: A–Z, a–z, 0–9, hyphen(-), underscore(_), space; must start and end with a letter or digit
Response Structure **DEVICE_NAME <name>[CR/LF]**	
Command Example **GET DEVICE_NAME[CR/LF]**	
Response Example **DEVICE_NAME SYN-CTL-HUB-001[CR/LF]**	
Additional Notes:	

1.6 Reset

Restores the device to factory default settings and reboots it.

Command Structure **RESET[CR/LF]**	Parameters:
Response Structure **RESET[CR/LF]**	
Command Example **RESET[CR/LF]**	
Response Example **RESET[CR/LF]**	
Additional Notes:	

1.7 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:	

1.8 Get Device Serial Number

Retrieve 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:	

1.9 Get Device Firmware Version

Retrieve the firmware version number of the specified device.

Command Structure **GET VER[CR/LF]**	Parameters: <ver> = "VX.Y.Z", where X, Y, and Z are integers from 0 to 999
Response Structure **VER <ver>[CR/LF]**	
Command Example **GET VER[CR/LF]**	
Response Example **VER V1.2.0[CR/LF]**	
Additional Notes:	

1.10 Get Uptime

Get the system uptime since power-on or last reboot.

Command Structure **GET UPTIME[CR/LF]**	Parameters: <days> = Number of days the system has been running <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:	

1.11 Get Device IP Address

Get IP address

Command Structure **GET IPADDR[CR/LF]**	Parameters: <mode> = {STATIC, DHCP} <ipaddr> = IP address <netmask> = Subnet mask <gateway> = Gateway IP address (Optional)
Response Structure **IPADDR <mode> IP4ADDR <ipaddr> 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]**	
Additional Notes:	

1.12 Keepalive

Used to keep the API channel connection alive and carry a keepalive message. Can be used for link status detection, latency analysis, or custom identification.

Command Structure **KEEPALIVE <msg>[CR/LF]**	Parameters: msg = any string (e.g., timestamp, ID, "ping", etc.) used as keepalive data; case-sensitive; recommended length ≤ 64 bytes.
Response Structure **KEEPALIVE <msg>[CR/LF]**	
Command Example **KEEPALIVE msg1[CR/LF]**	
Response Example **KEEPALIVE msg1[CR/LF]**	
Additional Notes:	

1.13 Set Keepalive Timeout

Set the device's KEEPALIVE timeout threshold. If a keepalive is not received before the timeout, the connection is considered lost. 0 disables the keepalive feature.

Command Structure **SET KEEPALIVE_TIMEOUT <timeout>[CR/LF]**	Parameters: <timeout> = [10, 86400], timeout in seconds; 0 disables the keepalive timeout mechanism
Response Structure **KEEPALIVE_TIMEOUT <timeout>[CR/LF]**	
Command Example **SET KEEPALIVE_TIMEOUT 60[CR/LF]**	
Response Example **KEEPALIVE_TIMEOUT 60[CR/LF]**	
Additional Notes:	

1.14 Get Keepalive Timeout

Query the device KEEPALIVE mechanism timeout threshold. It indicates how long without messages will trigger the disconnection logic.

Command Structure **GET KEEPALIVE_TIMEOUT[CR/LF]**	Parameters: <timeout> = [10, 86400], timeout in seconds; 0 disables the keepalive timeout mechanism
Response Structure **KEEPALIVE_TIMEOUT <timeout>[CR/LF]**	
Command Example **GET KEEPALIVE_TIMEOUT[CR/LF]**	
Response Example **KEEPALIVE_TIMEOUT 60[CR/LF]**	
Additional Notes:	

2. Telnet and SSH Config

2.1 Set TELNET_AUTH_FN

Enables or disables authentication for Telnet API logins.

Command Structure **SET TELNET_AUTH_FN <status>[CR/LF]**	Parameters: <status> = {ON, OFF}: ON means enable authentication, OFF means disable authentication
Response Structure **TELNET_AUTH_FN <status>[CR/LF]**	
Command Example **SET TELNET_AUTH_FN ON[CR/LF]**	
Response Example **TELNET_AUTH_FN ON[CR/LF]**	
Additional Notes:	

2.2 Get TELNET_AUTH_FN

Queries whether the Telnet API login authentication feature is enabled or disabled.

Command Structure **GET TELNET_AUTH_FN[CR/LF]**	Parameters: <status> = {ON, OFF}: ON indicates authentication is enabled; OFF indicates authentication is disabled
Response Structure **TELNET_AUTH_FN <status>[CR/LF]**	
Command Example **GET TELNET_AUTH_FN[CR/LF]**	
Response Example **TELNET_AUTH_FN ON[CR/LF]**	
Additional Notes:	

2.3 Set Telnet Port

Sets the listening port for the Telnet service.

Command Structure **SET TELNET_PORT <net_port>[CR/LF]**	Parameters: <port> = 1 to 65535
Response Structure **TELNET_PORT <net_port>[CR/LF]**	
Command Example **SET TELNET_PORT 2323[CR/LF]**	
Response Example **TELNET_PORT 2323[CR/LF]**	
Additional Notes:	

2.4 Get Telnet Port

Get the TCP port on which the Telnet service is currently listening.

Command Structure **GET TELNET_PORT[CR/LF]**	Parameters: <port> = [1, 65535], the TCP port number the Telnet service is currently listening on.
Response Structure **TELNET_PORT <net_port>[CR/LF]**	
Command Example **GET TELNET_PORT[CR/LF]**	
Response Example **TELNET_PORT 2323[CR/LF]**	
Additional Notes:	

2.5 Set SSH

Enables or disables the SSH API service on the device.

Command Structure **SET SSH <status>[CR/LF]**	Parameters: <status> = {ON, OFF}: ON means enable the SSH API service, OFF means disable the SSH API service
Response Structure **SSH <status>[CR/LF]**	
Command Example **SET SSH ON[CR/LF]**	
Response Example **SSH ON[CR/LF]**	
Additional Notes:	

2.6 Get SSH

Queries whether the device's SSH API service is currently enabled or disabled.

Command Structure **GET SSH[CR/LF]**	Parameters: <status> = {ON, OFF}: ON means the SSH API service is enabled, OFF means the SSH API service is disabled.
Response Structure **SSH <status>[CR/LF]**	
Command Example **GET SSH[CR/LF]**	
Response Example **SSH OFF[CR/LF]**	
Additional Notes:	

2.7 Set Telnet and SSH API Password

Set or update the authentication password for Telnet and SSH API access. The current password must be provided for verification.

Command Structure **SET API_PWD <oldpwd> <newpwd>[CR/LF]**	Parameters: <oldpwd>: The currently active API password used for verification <newpwd>: The new password, 8–24 characters; allowed characters: A–Z, a–z, 0–9, and these special symbols: !@#$%^&*()_+-=[]{};:'"|,.<>/? . Spaces, non-ASCII characters, or symbols not listed are not allowed <result>: {"Password wrong", "User name or password wrong", "Password changed successfully"}
Response Structure **<result>[CR/LF]**	
Command Example **SET API_PWD 12345678 NewP@ssw0rd![CR/LF]**	
Response Example **Password changed successfully[CR/LF]**	
Additional Notes:	

3. UART and Serial Setup

3.1 Set UART Mode

Set the UART mode for a serial port.

Command Structure **SET UART_MODE <port_id> <mode>[CR/LF]**	Parameters: <port_id> = Serial port number, starting from 1 <mode> = {RS232, RS422, RS485}
Response Structure **UART_MODE <port_id> <mode>[CR/LF]**	
Command Example **SET UART_MODE RS232[CR/LF]**	
Response Example **UART_CFG 1 RS232[CR/LF]**	
Additional Notes:	

3.2 Get UART Mode

Retrieve the UART mode for a given serial port.

Command Structure **GET UART_MODE <port_id>[CR/LF]**	Parameters: <port_id> = Serial port number, starting from 1 <mode> = {RS232, RS422, RS485}
Response Structure **UART_MODE <port_id> <mode>[CR/LF]**	
Command Example **GET UART_MODE 1[CR/LF]**	
Response Example **UART_CFG 1 RS232[CR/LF]**	
Additional Notes:	

3.3 Set UART_CFG

Set UART configuration

Command Structure **SET UART_CFG <port_id> <baud_rate> <data_bits> <parity> <stop_bits>[CR/LF]**	Parameters: <port_id> = serial port number, starting from 1 <baud_rate> = {2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200} <data_bits> = {5, 6, 7, 8} <parity> = {NONE, ODD, EVEN} <stop_bits> = {1, 2}
Response Structure **UART_CFG <port_id> <baud_rate> <data_bits> <parity> <stop_bits>[CR/LF]**	
Command Example **SET UART_CFG 1 9600 8 NONE 1[CR/LF]**	
Response Example **UART_CFG 1 9600 8 NONE 1[CR/LF]**	
Additional Notes:	

3.4 Get UART_CFG

Get UART configuration

Command Structure **GET UART_CFG <port_id>[CR/LF]**	Parameters: <port_id> = Serial port number, starting from 1 <baud_rate> = {2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200} <data_bits> = {5, 6, 7, 8} <parity> = {NONE, ODD, EVEN} <stop_bits> = {1, 2}
Response Structure **UART_CFG <port_id> <baud_rate> <data_bits> <parity> <stop_bits>[CR/LF]**	
Command Example **GET UART_CFG 1[CR/LF]**	
Response Example **UART_CFG 1 9600 8 NONE 1[CR/LF]**	
Additional Notes:	

3.5 Set UART Flow

Set UART flow control

Command Structure **SET UART_FLOW <port_id> <flow_control>[CR/LF]**	Parameters: <port_id> = UART port number, starting from 1 <flow_control> = {RTS/CTS, NONE}
Response Structure **UART_FLOW <port_id> <flow_control>[CR/LF]**	
Command Example **SET UART_FLOW 1 HARDWARE[CR/LF]**	
Response Example **UART_FLOW 1 HARDWARE[CR/LF]**	
Additional Notes:	

3.6 Get UART Flow

Retrieve UART flow-control setting

Command Structure **GET UART_FLOW <port_id>[CR/LF]**	Parameters: <port_id> = UART port number, starting from 1 <flow_control> = {RTS/CTS, NONE}
Response Structure **UART_FLOW <port_id> <flow_control>[CR/LF]**	
Command Example **GET UART_FLOW 1[CR/LF]**	
Response Example **UART_FLOW 1 HARDWARE[CR/LF]**	
Additional Notes:	

3.7 Set UART Proxy Function

Enable or disable the proxy function on a specified UART port for transparent data passthrough.

Command Structure **SET UART_PROXY_FN <port_id> <state>[CR/LF]**	Parameters: <port_id> = UART port number, starting from 1 <state> = {ON, OFF}
Response Structure **UART_PROXY_FN <port_id> <state>[CR/LF]**	
Command Example **SET UART_PROXY_FN 1 ON[CR/LF]**	
Response Example **UART_PROXY_FN 1 ON[CR/LF]**	
Additional Notes:	

3.8 Get UART_PROXY_FN

Query whether the proxy function is enabled for the specified UART port.

Command Structure **GET UART_PROXY_FN <port_id>[CR/LF]**	Parameters: <port_id> = Serial port number, starting from 1 <state> = {ON, OFF}
Response Structure **UART_PROXY_FN <port_id> <state>[CR/LF]**	
Command Example **GET UART_PROXY_FN 1[CR/LF]**	
Response Example **UART_PROXY_FN 1 ON[CR/LF]**	
Additional Notes:	

3.9 Set UART Proxy Port

Configure the logical mapped port used by a specified UART proxy. This sets up a transparent passthrough binding to internal or external system ports.

Command Structure **SET UART_PROXY_PORT <port_id> <net_port>[CR/LF]**	Parameters: <port_id> = UART port number, starting from 1 <net_port> = Network port
Response Structure **UART_PROXY_PORT <port_id> <net_port>[CR/LF]**	
Command Example **SET UART_PROXY_PORT 1 5001[CR/LF]**	
Response Example **UART_PROXY_PORT 1 5001[CR/LF]**	
Additional Notes:	

3.10 Get UART Proxy Port

Query the target port number (mapped port) currently configured for the proxy function of the specified serial port.

Command Structure **GET UART_PROXY_PORT <port_id>[CR/LF]**	Parameters: <port_id> = Serial port number, starting from 1 <net_port> = Network port
Response Structure **UART_PROXY_PORT <port_id> <net_port>[CR/LF]**	
Command Example **GET UART_PROXY_PORT 1[CR/LF]**	
Response Example **UART_PROXY_PORT 1 5001[CR/LF]**	
Additional Notes:	

3.11 ACT UART_SEND

Send serial data over the specified UART port in ACT format. Supports ASCII (string) or hexadecimal formats. The entire command must not exceed 4000 characters.

Command Structure **ACT UART_SEND <port_id> <format> <data>[CR/LF]**	Parameters: <port_id> = UART port number, starting from 1 <format> = {ASCII, HEX}, data format - ASCII mode: supports only printable characters from the ASCII table (0x20–0x7E). Does not support Chinese or other multibyte characters. - HEX mode: hexadecimal mode where each byte is represented by two hex digits, case-insensitive, bytes separated by spaces. In HEX mode, <data> may be enclosed in double quotes or not; both forms are equivalent. <data> = data to send - ASCII mode: if the content contains spaces, double quotes, carriage return or line feed, it must be enclosed in double quotes. Supports four escape sequences: \" (double quote), \\ (backslash), \r (carriage return), \n (line feed). No other escape sequences are supported. - HEX mode: <data> can be written as an unquoted sequence of hexadecimal bytes or enclosed in double quotes; both forms are equivalent.
Response Structure **UART_SEND <port_id> <format> <data>[CR/LF]**	
Command Example **ACT UART_SEND 1 ASCII "hello \"uart\" test"[CR/LF]**	
Response Example **UART_SEND 1 ASCII "hello \"uart\" test"[CR/LF]**	
Additional Notes:	

3.12 Set Serial Mode

Set the serial port to IR or one-way RS-232 mode.

Command Structure **SET SERIAL_MODE <port_id> <mode>[CR/LF]**	Parameters: <port_id> = Serial port number, starting at 1 <mode> = {IR, RS-232}
Response Structure **SERIAL_MODE <port_id> <mode>[CR/LF]**	
Command Example **SET SERIAL_MODE 1 RS-232[CR/LF]**	
Response Example **SERIAL_MODE 1 RS-232[CR/LF]**	
Additional Notes:	

3.13 Get Serial Mode

Get serial port working mode

Command Structure **GET SERIAL_MODE <port_id>[CR/LF]**	Parameters: <port_id> = serial port ID <mode> = {IR, RS-232}
Response Structure **SERIAL_MODE <port_id> <mode>[CR/LF]**	
Command Example **GET SERIAL_MODE 1[CR/LF]**	
Response Example **SERIAL_MODE 1 RS-232[CR/LF]**	
Additional Notes:	

4. IR Configuration

4.1 ACT IR_SEND

Send an infrared command through a specified IR port. Supports a custom pulse format (format 0) or Pronto Hex format (format 1).

Command Structure

ACT IR_SEND <port> <format> <ir_code>[CR/LF]

Response Structure

IR_SEND <port> <format> <ir_code>[CR/LF]

Parameters:
<port> = IR output port, e.g., 1, 2...
<format> = command format type: 0 = custom pulse format, 1 = Pronto Hex format
<ir_code> = IR code string; the format depends on <format>

Command Example

```
ACT IR_SEND 1 0
38000,2,3,900,450,600,450,900,450,600,450[CR/LF]
ACT IR_SEND 1 1 0000 0067 0000 0015 0060 0018 0018 0018
0030 0018 0030 0018 0030 0018 0018 0018 0018 0018 0018
0018 0018 0018 0030 0018 0018 0018
0030 0018 0030 0018 0030 0018 0030 0018 0030 0018 0030
0018 0018 0018 0030 0018 0030 0018 0018 01d9[CR/LF]
```

Response Example

```
IR_SEND 1 0
38000,2,3,900,450,600,450,900,450,600,450[CR/LF]
IR_SEND 1 1 0000 0067 0000 0015 0060 0018 0018 0018 0030
0018 0030 0018 0030 0018 0018 0018 0018 0018 0018 0018
0018 0018 0018 0030 0018 0018 0018
0030 0018 0030 0018 0030 0018 0030 0018 0030 0018 0030
0018 0018 0018 0030 0018 0030 0018 0018 01d9[CR/LF]
```

Additional Notes:

4.2 ACT IR_SEND_X - Send Infrared Data (Extended)

Sends infrared data through a specified IR output port. Supports either a custom pulse format (format 0) or Pronto Hex format (format 1). You can set the repeat count, repeat interval, and command gap to precisely control the timing of IR code transmission. Compared to ACT IR_SEND, this command adds timing control parameters (repeat_times, interval, gap).

Command Structure
ACT IR_SEND_X <port> <format> <repeat_times> <interval> <gap> <data>[CR/LF]

Response Structure
IR_SEND_X <port> <format> <repeat_times> <interval> <gap> <data>[CR/LF]

Command Example

```
ACT IR_SEND_X 1 0 3 100 50
38000,2,3,900,450,600,450[CR/LF]
ACT IR_SEND_X 1 1 3 100 50 0000 0067 0000 0015 0060 0018
0018 0018 0030 0018 0030 0018 0030 0018 0018 0018 0018
0018 0018 0018 0018 0018 0030 0018 0018 0018
0030 0018 0030 0018 0030 0018 0030 0018 0030 0018 0030
0018 0018 0018 0030 0018 0030 0018 0018 01d9[CR/LF]
```

Response Example

```
ACT IR_SEND_X 1 0 3 100 50
38000,2,3,900,450,600,450[CR/LF]
ACT IR_SEND_X 1 1 3 100 50 0000 0067 0000 0015 0060 0018
0018 0018 0030 0018 0030 0018 0030 0018 0018 0018 0018
0018 0018 0018 0018 0018 0030 0018 0018 0018
0030 0018 0030 0018 0030 0018 0030 0018 0030 0018 0030
0018 0018 0018 0030 0018 0030 0018 0018 01d9[CR/LF]
```

Parameters:
<port> = Infrared port number, starting from 1
<format> = {0, 1}, command format type, 0 = custom pulse format, 1 = Pronto Hex format;
<repeat_times> = [1, 255], times, number of repetitions for the same IR code;
<interval> = [0, 1000], ms, interval time between each repeated send;
<gap> = [0, 1000], ms, minimum wait time after this IR code sequence before the next IR send;
<data> = string, [1, 4000] characters, infrared data (format depends on <format>)

Additional Notes:

4.3 Set IR_PROXY_FN

Enable or disable the proxy function on a specified infrared (IR) port, used for serial data passthrough control.

Command Structure
SET IR_PROXY_FN <port_id> <state>[CR/LF]

Response Structure
IR_PROXY_FN <port_id> <state>[CR/LF]

Command Example

```
SET IR_PROXY_FN 1 ON[CR/LF]
```

Response Example

```
IR_PROXY_FN 1 ON[CR/LF]
```

Parameters:
<port_id> = IR port number, starting from 1
<state> = {ON, OFF}

Additional Notes:

4.4 Get IR_PROXY_FN

Query whether the proxy function is enabled for a specified infrared (IR) port.

Command Structure **GET IR_PROXY_FN <port_id>[CR/LF]**	Parameters: <port_id> = IR port number, starting from 1 <state> = {ON, OFF}
Response Structure **IR_PROXY_FN <port_id> <state>[CR/LF]**	
Command Example **GET IR_PROXY_FN 1[CR/LF]**	
Response Example **IR_PROXY_FN 1 ON[CR/LF]**	
Additional Notes:	

4.5 Set IR Proxy Port

Configure the logical mapping port used by the proxy feature for a specified infrared (IR) port. This mapping establishes a pass-through binding with internal or external system ports.

Command Structure **SET IR_PROXY_PORT <port_id> <net_port>[CR/LF]**	Parameters: <port_id> = Infrared port number, starting from 1 <net_port> = Network port
Response Structure **IR_PROXY_PORT <port_id> <net_port>[CR/LF]**	
Command Example **SET IR_PROXY_PORT 1 5001[CR/LF]**	
Response Example **IR_PROXY_PORT 1 5001[CR/LF]**	
Additional Notes:	

4.6 Get IR Proxy Port

Query the target port number (mapped port) currently configured for the proxy function of the specified IR port.

Command Structure **GET IR_PROXY_PORT <port_id>[CR/LF]**	Parameters: <port_id> = IR port number, starting from 1 <net_port> = Network port
Response Structure **IR_PROXY_PORT <port_id> <net_port>[CR/LF]**	
Command Example **GET IR_PROXY_PORT 1[CR/LF]**	
Response Example **IR_PROXY_PORT 1 5001[CR/LF]**	
Additional Notes:	

5. GPIO and Relay Commands

5.1 Set GPIO_CFG

Set GPIO properties

Command Structure **SET GPIO_CFG <port> <direction> <pull_up>[CR/LF]**	Parameters: <port> = Channel number, starting from 1 <direction> = {IN, OUT} <pull_up> = {ON, OFF}, ON means a 5V pull-up resistor is connected; OFF means disconnected
Response Structure **GPIO_CFG <port> <direction> <pull_up>[CR/LF]**	
Command Example **SET GPIO_CFG 1 IN ON[CR/LF]**	
Response Example **GPIO_CFG 1 IN ON[CR/LF]**	
Additional Notes:	

5.2 Get GPIO_CFG

Query GPIO attributes

Command Structure **GET GPIO_CFG <port>[CR/LF]**	Parameters: <port> = Channel number, starting from 1 <directio> = {IN, OUT} <pull_up> = {ON, OFF}, ON means the 5V pull-up resistor is connected, OFF means it is disconnected
Response Structure **GPIO_CFG <port> <direction> <pull_up>[CR/LF]**	
Command Example **GET GPIO_CFG 1[CR/LF]**	
Response Example **GPIO_CFG 1 IN ON[CR/LF]**	
Additional Notes:	

5.3 Set GPIO Output Control

Set the logical state (HIGH or LOW) of a GPIO output channel. The "status" field specifies the output level.

Command Structure **SET GPIO_OUT_CTL <port> <status>[CR/LF]**	Parameters: <port> = channel number, starting from 1 <status> = {LOW, HIGH}: LOW means pull-down enabled (output driven low), HIGH means high-impedance (tri-stated)
Response Structure **GPIO_OUT_CTL <port> <status>[CR/LF]**	
Command Example **SET GPIO_OUT_CTL 1 HIGH[CR/LF]**	
Response Example **GPIO_OUT_CTL 1 HIGH[CR/LF]**	
Additional Notes:	

5.4 Get GPIO_OUT_CTL

Reads the current output logic state of a GPIO output channel.

Command Structure **GET GPIO_OUT_CTL <port>[CR/LF]**	Parameters: <port> = channel number, starting from 1 <status> = {LOW, HIGH}: LOW indicates pull-down (output driven low), HIGH indicates high-impedance
Response Structure **GPIO_OUT_CTL <port> <status>[CR/LF]**	
Command Example **GET GPIO_OUT_CTL 1[CR/LF]**	
Response Example **GPIO_OUT_CTL 1 HIGH[CR/LF]**	
Additional Notes:	

5.5 GPIO IN Status

GPIO IN status report

Command Structure	Parameters: <port> = channel number, starting at 1 <status> = {LOW, HIGH}
Response Structure **GPIO_IN_STATUS <port> <status>[CR/LF]**	
Command Example	
Response Example **GPIO_IN_STATUS 1 HIGH[CR/LF]**	
Additional Notes:	

5.6 Get GPIO IN Status

Query GPIO IN status

Command Structure **GET GPIO_IN_STATUS <port>[CR/LF]**	Parameters: <port> = Channel number, starting from 1 <status> = {LOW, HIGH}
Response Structure **GPIO_IN_STATUS <port> <status>[CR/LF]**	
Command Example **GET GPIO_IN_STATUS 1[CR/LF]**	
Response Example **GPIO_IN_STATUS 1 HIGH[CR/LF]**	
Additional Notes:	

5.7 Set Relay

Controls the relay state for the specified port; can be set to OPEN or CLOSE.

Command Structure **SET RELAY_CTL <port> <mode>[CR/LF]**	Parameters: <port>: Relay port number. If there is only one relay, set this to 1 <mode>: Relay control mode: - OPEN: NO (Normally Open) is disconnected from COM (Common), NC (Normally Closed) is connected to COM - CLOSE: NO (Normally Open) is connected to COM, NC (Normally Closed) is disconnected from COM
Response Structure **RELAY_CTL <port> <mode>[CR/LF]**	
Command Example **SET RELAY_CTL 1 OPEN[CR/LF]**	
Response Example **RELAY_CTL 1 OPEN[CR/LF]**	
Additional Notes:	

5.8 Get Relay

Retrieve the current operating status (OPEN or CLOSE) of the relay at the specified port.

Command Structure **GET RELAY_CTL <port>[CR/LF]**	Parameters: <port>: Relay port number; if there is only one relay, still set to 1 <mode>: Relay control mode: - OPEN: disconnects NO (Normally Open) from COM, connects NC (Normally Closed) to COM - CLOSE: connects NO (Normally Open) to COM, disconnects NC (Normally Closed) from COM
Response Structure **RELAY_CTL <port> <mode>[CR/LF]**	
Command Example **GET RELAY_CTL 1[CR/LF]**	
Response Example **RELAY_CTL 1 OPEN[CR/LF]**	
Additional Notes:	

5.9 Set Relay Control Boot Mode

Set the control behavior mode for the specified relay port after the device restarts. DEFAULT: the relay restores to the factory default state; SAVED: the relay restores to the last saved state.

Command Structure **SET RELAY_CTL_BOOT_MODE <port> <mode>[CR/LF]**	Parameters: <port> = relay port number; if there is only one relay, set to 1 <mode> = {DEFAULT, SAVED}, DEFAULT means use the factory default configuration, SAVED means use the state configured by RELAY_CTL
Response Structure **RELAY_CTL_BOOT_MODE <port> <mode>[CR/LF]**	
Command Example **SET RELAY_CTL_BOOT_MODE 1 SAVED[CR/LF]**	
Response Example **RELAY_CTL_BOOT_MODE 1 SAVED[CR/LF]**	
Additional Notes:	

5.10 Get Relay Control Boot Mode

Query the boot (startup) mode setting for the relay on the specified port.

Command Structure **GET RELAY_CTL_BOOT_MODE <port>[CR/LF]**	Parameters: <port> = Relay port number; if there is only one relay, set this to 1 <mode> = {DEFAULT, SAVED}, DEFAULT means use the factory default configuration, SAVED means use the state configured by RELAY_CTL
Response Structure **RELAY_CTL_BOOT_MODE <port> <mode>[CR/LF]**	
Command Example **GET RELAY_CTL_BOOT_MODE 1[CR/LF]**	
Response Example **RELAY_CTL_BOOT_MODE 1 SAVED[CR/LF]**	
Additional Notes:	