# TCP/IP protocol stack for SPWF04Sx Wi-Fi modules

## Introduction

The SPWF04Sx series of Wi-Fi modules integrate a complete TCP/IP protocol stack and a rich set of applications including, but not limited to, web server, web client RESTful API, TFTP, MQTT and SMTP.

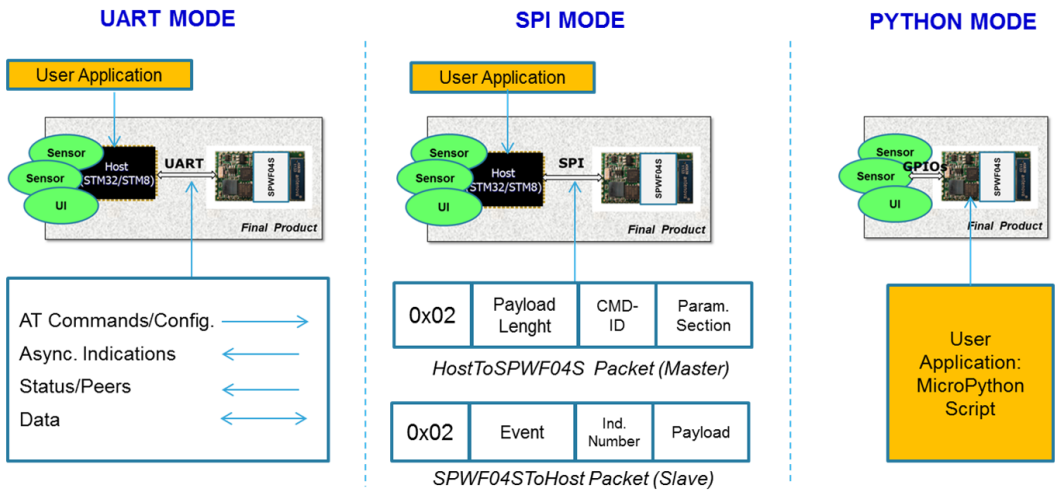
Security is added in the multiple layers of Wi-Fi and peer-to-peer protocols. The stack implements the WPS protocol, WPA2 in both the Personal and Enterprise options, and the TLS for end-to-end secure transactions.

Users access the features of the SPWF04Sx modules via UART using the simple AT command syntax, or via SPI using a custom packet format and protocol.

To enable a complete customization of the application on the module, the SPWF04Sx software integrates a MicroPython interpreter that provides the user with MicroPython standard libraries and a customized set of classes to export the specific SPWF04Sx features.

The diagram below summarizes the possible integrations of an SPWF04Sx module in a target application.

**Figure 1. SPWF04Sx user integration modes**



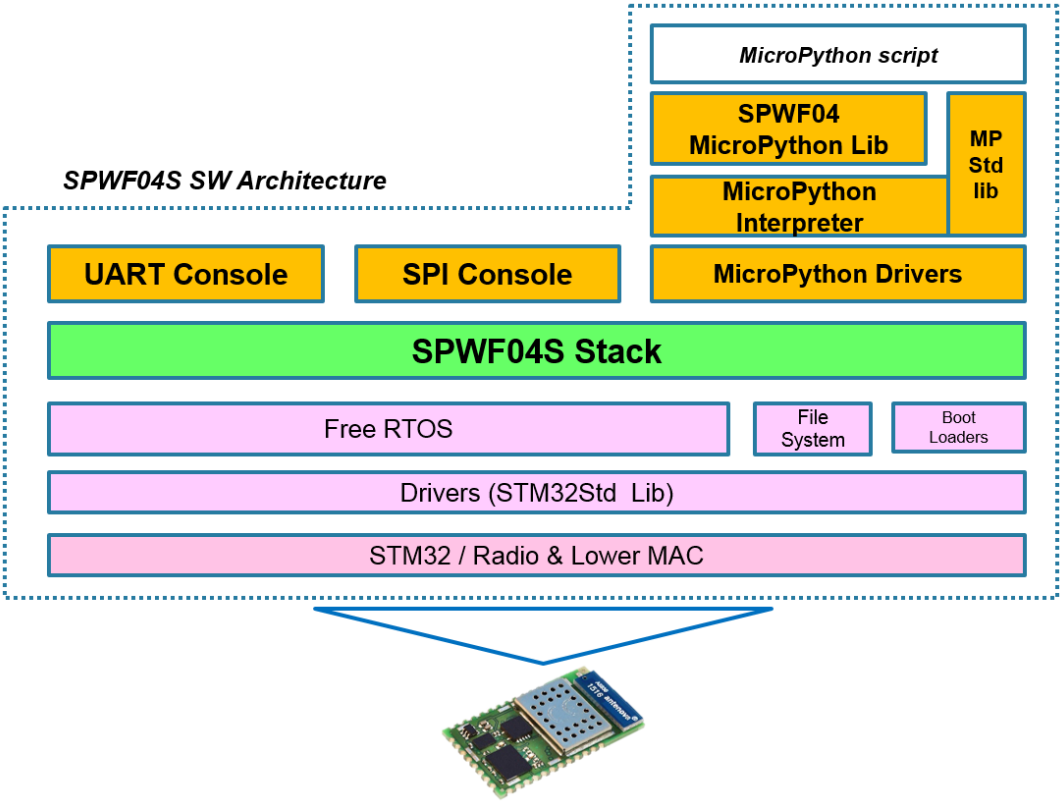

This user manual is intended as a guide to the set of commands available on the UART or the SPI console. A description and explanation of the configuration variables, status variables and asynchronous indication messages is available in the Appendix of the manual.

This manual is not intended as a technical guide of Wi-Fi and TCP/IP, or other technologies available in the module.

# 1 SPWF04Sx software architecture description

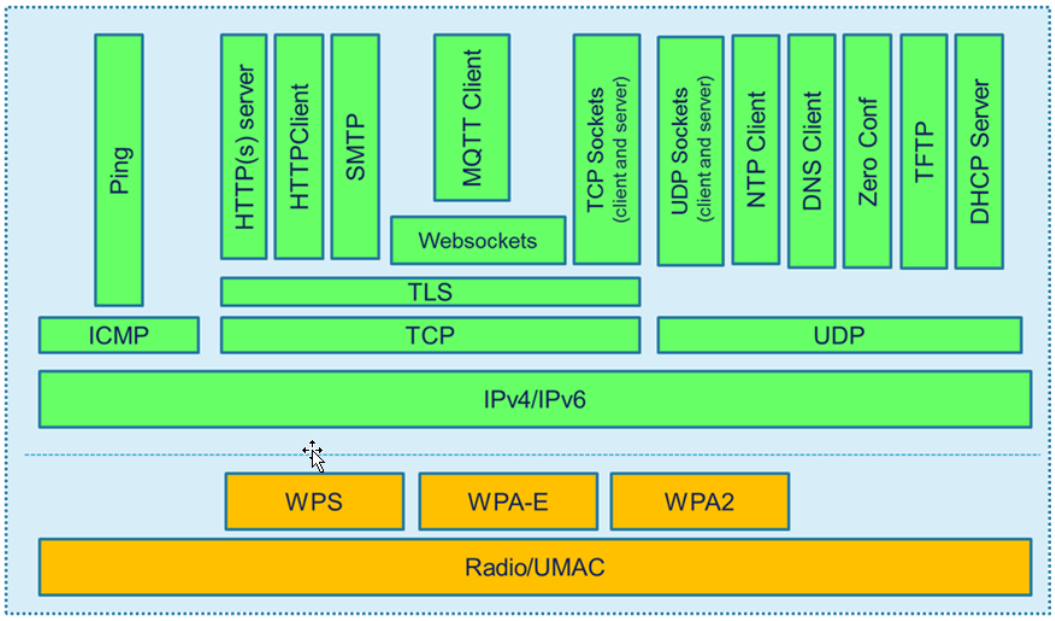
The complete SW architecture of SPWF04S is shown in the diagram below.

**Figure 2. SPWF04Sx software architecture**




A block diagram of the SPWF04Sx protocol stack is provided in Figure 3. SPWF04Sx protocol stack diagram.

Figure 3. **SPWF04Sx protocol stack diagram**



# 2 Interface and message types

As shown in Figure 1. SPWF04Sx user integration modes, the SPWF04Sx enables three options for integration of the device in the final user application. Refer to the SPWF04Sx datasheet for information useful for implementation, such as the pinout. The configuration variable, “console_enabled”, is used to define the module interface to use.

- **UART interface**

  *“console_enabled=1”*

  The UART console provides a user-friendly interface built on a set of AT commands that allow an external microcontroller connected to the SPWF04Sx UART to access the functions integrated in SPWF04Sx device. UART is set as the module's default interface.

- **SPI interface**

  *“console_enabled=0”*

  The device can be connected as a slave to the SPI interface of an external microcontroller. An ad hoc SPI protocol and corresponding packet format are defined.

- **MicroPython scripting**

  *"console_enabled=2”* - to use Python together with AT commands over UART.

  *"console_enabled=3”* - to use MicroPython only.

  The device implements a scripting methodology based on an integrated MicroPython interpreter. MicroPython scripts can map a target application, making it unnecessary to integrate the device with an external processor.

The following four message types represent the type of data exchanged over the interfaces.

- **Commands**

  Used to activate a feature defined within the stack. The list of commands supported is specified in Table 1. SPWF04Sx commands.

- **Command return messages**

  Synchronous messages that report the status of the execution of a command. The list of synchronous error codes is reported in Section 8 in the Appendix.

- **WINDs**

  Asynchronous messages reporting a network or radio status at the application level. The list of WIND messages is reported in Section 8 in the Appendix.

- **Data**

  Data sent to, or received (data payload) from, a remote device.

**Table 1. SPWF04Sx commands**

| Command ID | AT command | Description |
|---|---|---|
| **Utils: commands for debugging and retrieving module status** | | |
| 0x01 | AT | Null command |
| 0x02 | AT+S.HELP | Help command |
| 0x05 | AT+S.STS | Status configuration |
| 0x35 | AT+S.PEERS | Peers configuration |
| **Management: commands for module management and configuration** | | |
| 0x03 | AT+S.RESET | SW reset |
| 0x04 | AT+S.PMS | Set power mode |
| 0x08 | AT+S.PYTHON | Enter MicroPython execution |
| 0x09 | AT+S.GCFG | Read Configuration status |

| Command ID | AT command | Description |
|---|---|---|
| 0x0A | AT+S.SCFG | Set configuration variables |
| 0x0B | AT+S.WCFG | Save configuration to Flash |
| 0x0C | AT+S.FCFG | Restore factory configuration |
| 0x57 | AT+S.FSWRITE | FS update via serial UART/SPI |
| 0x58 | AT+S.FSUPDATE | FS download |
| 0x56 | AT+S.FWUPDATE | FW download |
| **STM32 peripherals: commands to manage the peripherals and related values** | | |
| 0x13 | AT+S.GPIOC | Configure GPIO |
| 0x14 | AT+S.GPIOR | Read GPIO |
| 0x15 | AT+S.GPIOW | Write GPIO |
| 0x16 | AT+S.DAC | Disable/Enable DAC |
| 0x17 | AT+S.ADC | Read ADC value |
| 0x18 | AT+S.PWM | Set PWM |
| 0x11 | AT+S.TIME | Get/Set time |
| 0x12 | AT+S.RANDOM | Provide random number |
| **File system management** | | |
| 0x21 | AT+S.FSM | Mount volume |
| 0x22 | AT+S.FSU | Umount/Erase volume |
| 0x23 | AT+S.FSC | Create file, append data |
| 0x25 | AT+S.FSD | Delete file |
| 0x26 | AT+S.FSR | Rename file |
| 0x27 | AT+S.FSL | List existing files |
| 0x28 | AT+S.FSP | Print file content |
| 0x29 | AT+S.HASH | Compute digest |
| **Security: commands to interact with the security features** | | |
| 0x2A | AT+S.WPAECERT | Manage WPA-Enterprise certificates |
| 0x2B | AT+S.TLSCERT | Manage TLS certificates |
| 0x36 | AT+S.WPS | Initiate a WPS Exchange |
| **Radio: commands to manage main Wi-Fi operations** | | |
| 0x32 | AT+S.WIFI | Set Wi-Fi radio |
| 0x33 | AT+S.SCAN | Network scan |
| 0x34 | AT+S.SSIDTXT | Get/Set ASCII SSID |
| 0x39 | AT+S.PING | Ping a specified host |
| **Sockets: commands to manage socket read and write** | | |
| 0x41 | AT+S.SOCKON | Open a socket client |
| 0x42 | AT+S.SOCKQ | Query a socket client for pending data |
| 0x43 | AT+S.SOCKC | Close a socket client |
| 0x44 | AT+S.SOCKW | Write data to a socket server |
| 0x45 | AT+S.SOCKR | Read data from a socket client |
| 0x46 | AT+S.SOCKL | List opened socket clients |

| Command ID | AT command | Description |
|---|---|---|
| 0x47 | AT+S.SOCKDON | Open a socket server |
| 0x48 | AT+S.SOCKDQ | Query socket server for pending data |
| 0x49 | AT+S.SOCKDC | Close a socket server |
| 0x4A | AT+S.SOCKDW | Write data to a socket server |
| 0x4B | AT+S.SOCKDR | Read data from a socket server |
| 0x4C | AT+S.SOCKDL | List bound socket clients |
| **Web sockets** | | |
| 0x61 | AT+S.WSOCKON | Open a web socket client |
| 0x62 | AT+S.WSOCKQ | Query a web socket client for pending data |
| 0x63 | AT+S.WSOCKC | Close web socket client |
| 0x64 | AT+S.WSOCKW | Write data to a web socket client |
| 0x65 | AT+S.WSOCKR | Read data from web socket client |
| 0x66 | AT+S.WSOCKL | List open web socket client |
| **Trivial FTP** | | |
| 0x51 | AT+S.TFTPGET | Get request to a TFTP server |
| 0x52 | AT+S.TFTPPUT | Put request to a TFTP server |
| **SMTP** | | |
| 0x53 | AT+S.SMTP | Send an email |
| **HTTP** | | |
| 0x54 | AT+S.HTTPGET | Get a request to an HTTP server |
| 0x55 | AT+S.HTTPPOST | Post request to an HTTP server |
| 0x59 | AT+S.INPUTSSI | Fill buffer for raw text input SSI |
| **MQTT** | | |
| 0x5A | AT+S.MQTTCONN | MQTT connect |
| 0x5B | AT+S.MQTTSUB | MQTT subscribe |
| 0x5C | AT+S.MQTTPUB | MQTT publish |
| 0x5D | AT+S.MQTTUNSUB | MQTT unsubscribe |
| 0x5E | AT+S.MQTTDISC | MQTT disconnect |

# 3 AT commands over the UART

The factory module configuration sets the UART console mode as the default interface for the SPWF04Sx. This corresponds to the configuration variable “console_enabled” being set to 1.

AT commands over the UART have a max length of 512 bytes; they are case insensitive and are always in the form of:

***AT+S.*** *<cmd-parameters><cr>[data]*

**Note:** Any command requiring data after the <cr> is not reentrant. If bytes are lost during data transfer over the UART, the module remains in the waiting stage for incoming bytes.

A command is followed by a variable number of response lines that have the following format:

***AT-S.*** *<response-string><optional-parameters>*

The AT command line, up to the terminating <cr>, is sent from the host. Response lines are sent from the module to the host.

| | |
|---|---|
| `AT-S.OK:<free_heap>:<wifi_state>` | returned when a command is successfully executed.<br>Note that free_heap and wifi_state are shown depending on the "console_verbose" configuration variable value (0 to 2). |
| `AT-S.ERROR:<error-code>:<error-string>` | qualifies a synchronous error. The <error-code> field of each asynchronous indication type is unique. The descriptive string may be safely ignored.<br>Note that error_code and error_string are shown depending on the "console_errs" configuration variable value (0 to 2). |

**Command parameters**

A command can require parameters that follow an “=” character. The parameters are positional and separated by a configurable separator (by default, a comma).

Parameters can require mandatory or optional values. In the latter case, if the value is not specified a default value will be used. In the format of the command, an optional parameter is represented in squared brackets.

**Asynchronous indications**

Asynchronous indications may arrive at any time (except as noted below), and have the format:

`+WIND:<number>:<descriptive-string>[:<variables>]<cr><lf>`

The `<number>` field of each asynchronous indication type is unique. The descriptive string may be safely ignored.

Note that number and descriptive strings are shown depending on the "console_winds" configuration variable value (0 to 2).

Refer to Section 8 in the Appendix for a complete list of WIND messages.

**Note:** Immediately after reset, no commands should be sent and only asynchronous indications are present until the indication “*+WIND:0:Console active<cr><lf>*” is received. After this event, AT commands may be safely sent to the device.

# 4 SPI protocol

By setting to 0 the configuration variable "console_enabled", the module is enabled to use the SPI interface in place of the UART.

The data transferred over the MISO and MOSI signals are packed using a well-defined API packet format as represented below.

**Figure 4. SPI packet formats**

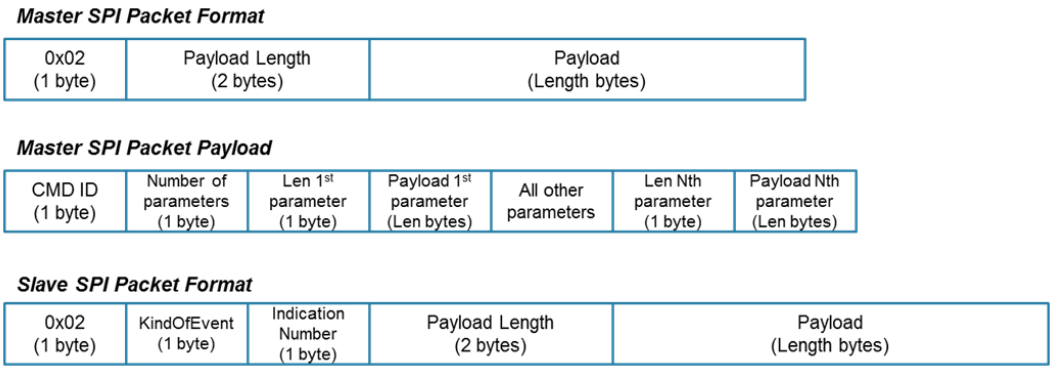



**Table 2. KindOfEvent Byte SubField**

| Bits | Event type |
|---|---|
| Bits 0:3 | Status variable wifi_state values range. Refer to Table 8. Status variables. |
| Bits 4:7 | Allowed values are:<br>• 0x01 for common indications like WIND or action confirmations<br>• 0x02 for critical error notifications<br>• 0x03 for incoming data sent over the SPI (in this case normally data are filled into payload field) |

**Table 3. Indication number field**

| Event type Bits 4:7 | Indication number |
|---|---|
| 0x01 | Refer to Table 14. WIND messages. |
| 0x02 | Refer to Table 10. AT-S.ERROR:=ERROR ID= =ERROR String=. |
| 0x03 | Data payload. |

Commands over the SPI have a max length of 512 bytes.

To map an AT command in the equivalent SPI command, the following procedure applies:

1. Use the corresponding CMD ID specified in Table 1. SPWF04Sx commands and fill the 4$^{th}$ bytes of the master packet.
2. If optional parameters are available, count the number of comma-separated items after the “=” character and with that number the 6$^{th}$ byte of the message. Starting from the 7$^{th}$ byte, start to write 1 byte containing the field lengths and then copy the field bytes. Then continue with the remaining parameters.

3. Once the full payload has been filled, calculate the full message payload lengths and accordingly fill bytes 1 and 2 in the SPI message packet request.

**First example**

AT command: AT+S.FSL

1. Command ID: 0x25
2. Number of parameters: 0x00
3. Full message payload length: 0x02

**Result:** SPI message 0x02 0x00 0x02 0x25 0x00

**Second example**

AT command: AT+S.SCAN=d,/scan.txt

1. Command ID: 0x33
2. Number of parameters: 0x02
   a. First Parameter: 0x01,d
   b. Second Parameter: 0x09,"/scan.txt"
3. Full message payload length: 0x0D

**Result:** SPI Message 0x02 0x00 0x0D 0x33 0x02 0x01 d 0x09 "/scan.txt"

**Third example**

AT command: AT+S.SOCKW=0,5<Cr>hello

1. Command ID: 0x44
2. Number of parameters: 0x02
   a. First Parameter: 0x01,0
   b. Second Parameter: 0x01,5
3. Payload: "hello"

**Result:** SPI Message 0x02 0x00 0x0B 0x44 0x02 0x01 0 0x01 5 "hello"

**Fourth example**

Asynchronous event: +WIND:1:Poweron:170726-b7ac1ba-SPWF04S

1. kind of event (4bits): 0x01
2. wifi_state (4 bits): 0x0
3. Indication number: 0x01
4. Full message payload length: 0x16

**Result**: SPI Message 0x02 0x10 0x01 0x16 0x00 (170726-b7ac1ba-SPWF04S)

# 5 MicroPython scripting

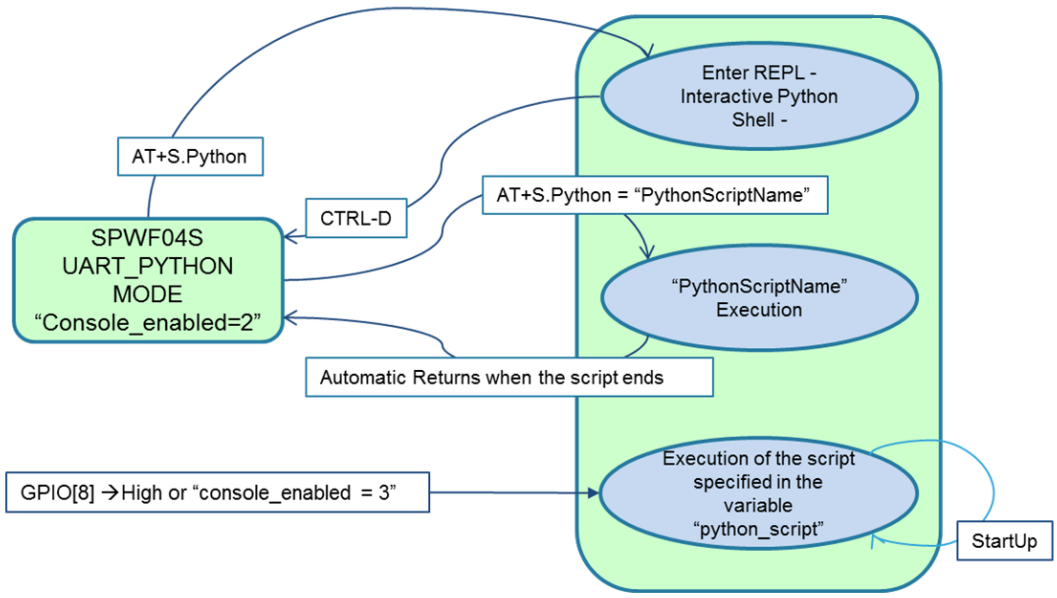
The SPWF04Sx supports the MicroPython modes represented in Figure 5. MicroPython modes. The configuration variable “console_enabled” identifies the use of the console together with MicroPython. In detail:

- "console_enabled" set to 2: this setting allows using both AT commands over the UART and MicroPython REPL shell.
- "console_enabled" set to 3: this setting allows only a MicroPython preloaded script to be executed. No UART/SPI communication is allowed to/from a host processor.

Consequently, the following Python modes are defined:

- **Python Interactive Console (REPL – Read Evaluation Print Loop)**. Mainly used for debugging purposes, it is activated by the AT+S.Python (Python command) launched without parameters. The REPL is exited by using a CTRL-D escape character.
- **Run Time Script execution**. Activated with the Python command, it allows to execute the script specified as parameter. After script execution, control returns to the AT console.
- **DefaultScript Hard Execution**. By using the GPIO(8) set to high at boot time, or when "console_enabled" is set to 3, the module automatically enters the execution of the script specified by the variable “python_script”. This mode allows the use of the module without a connected host. Note that "console_enabled" is not changed by GPIO(8) access: this means that on the subsequent reboot, accessing Python mode again will require GPIO(8) to be high.

**Figure 5. MicroPython modes**



# 6 Command reference guide

This section details each of the SPWF04Sx commands listed in Table 1. SPWF04Sx commands, including a brief description of its behavior and complete list of parameters. Examples and syntax are described in the “AT” format introduced in Section 3 AT commands over the UART. The mapping mechanism described in Section 4 SPI protocol is used to map the AT command in the packet format used by the SPI.

## 6.1 AT

AT, by itself, is a null command that always returns an OK result code. It is useful for testing the module interface for readiness.

Usage:

```
AT<cr>
```

Parameters:

None

## 6.2 AT+S.HELP

AT+S.HELP prints the list of all the AT commands supported with a brief help text for each of them. Refer to Table 1. SPWF04Sx commands for a complete list of the commands available.

Usage:

| | |
|---|---|
| 1) `AT+S.HELP<cr>` | to get the full list of commands |
| 2) `AT+S.HELP=<keyword><cr>` | to get the list of commands containing the specified keyword |

Parameters:

| | |
|---|---|
| `<keyword>` | Specifies the name of a keyword |

Result:

`AT-S.Command:<command> -- <help>`

| | |
|---|---|
| `<command>` | Specifies the command |
| `<help>` | Specifies the description of the command |

## 6.3 AT+S.STS

AT+S.STS displays the current values of all the status variables.

Usage:

| | |
|---|---|
| 1) AT+S.STS<cr> | to get the full list of configuration variables |
| 2) AT+S.STS=<variable><cr> | to get the value of the specified variable |

Parameters:

| | |
|---|---|
| <variable> | Specifies the name of a variable. See Table 8. Status variables for a list of the available status variables |

Result:

AT-S.Var:<var>=<value>

| | |
|---|---|
| <var> | Specifies the status variable name |
| <value> | Specifies the status variable value |

## 6.4 AT+S.PEERS

AT+S.PEERS displays the current values of the peer table. These values are useful to obtain additional information about the module connected to the AP or about the client connected to the module when it is configured in Mini AP mode.

Usage:

| | |
|---|---|
| 1) AT+S.PEERS<cr> | returns the list of peers variable |
| 2) AT+S.PEERS=<peer_number><cr> | returns the list of peers variable for the peer_number peer |
| 3) AT+S.PEERS=<peer_number>,<peer_var><cr> | returns the specific value of a peer_variable |

Parameters:

| | |
|---|---|
| <peer_number> | identifier of the peer |
| <peer_var> | displays the current value of the specified peer variable. Refer to Table 9. Peer variables in the Appendix for a complete list of the peer variables |

Result:

AT-S.Var:<id>.<var>=<value>

| | |
|---|---|
| <id> | Specifies the peer number |

| | |
|---|---|
| <var> | Specifies the peer variable name |
| <value> | Specifies the peer variable value |

## 6.5 AT+S.RESET

Command used to provide a SW reset to the module.

Usage:

```
AT+S.RESET<cr>
```

Parameters:

None

## 6.6 AT+S.PMS

The SPWF04Sx features different power states as a consequence of the different operating modes of the radio and the STM32 microcontroller. This command is used to manage the module power state.

The table that follows summarizes the power state conditions supported by the module. The values of the variables affecting the output of the command are also reported in the following table.

**Table 4. Power states and configuration**

| Module state | <Mode> option value | STM32 state | WLAN state | AT variable default values |
|---|---|---|---|---|
| Active | 0 | Run | Rx Idle<br>Rx Active<br>Tx Active | AT+S.SCFG=sleep_enabled,0<br>AT+S.SCFG=wifi_powersave,0<br>AT+S.SCFG=standby_enabled,0 |
| PowerSave (1) | 1 | Run | PS or Fast PS | AT+S.SCFG=sleep_enabled,0<br>AT+S.SCFG=standby_enabled,0<br>AT+S.SCFG=wifi_powersave,1 |
| Sleep(1) | 2 | Stop | PS or Fast PS | AT+S.SCFG=sleep_enabled,1<br>AT+S.SCFG=standby_enabled,0<br>AT+S.SCFG=wifi_powersave,1 |
| StandBy(2) | 3 | Standby | Off | AT+S.SCFG=standby_enabled,1<br>AT+S.SCFG=sleep_enabled,0 |

1. *Variables wifi_beacon_wakeup, wifi_operational_mode, and wifi_listen_interval, need to be set to the desired value.*
2. *The variable standby_time to be set to the desired value.*

Usage:

```
AT+S.PMS=[<mode>]<cr>
```

Parameters:

| | |
|---|---|
| <mode> | Default value: 0<br>Possible values:<br>• 0 → active mode<br>• 1 → powersave mode<br>• 2 → stop mode<br>• 3 → standby mode |

## 6.7 AT+S.PYTHON

The command has a Python script executed when the file name is specified or allows to enter a Python shell when the option is not used.

Usage:

| | |
|---|---|
| 1) AT+S.PYTHON<cr> | to enter the MicroPython REPL shell. |
| 2) AT+S.PYTHON=<filename><cr> | to execute the specified script. |

Parameters:

| | |
|---|---|
| <filename> | specifies the Python script to be executed. After the execution of the script the command returns to the AT console. |

## 6.8 AT+S.GCFG

This command lists all the configuration variables together with their current value.

Usage:

| | |
|---|---|
| 1) AT+S.GCFG<cr> | to get the full list of configuration variables |
| 2) AT+S.GCFG=<cfg_var><cr> | to get the value of the specified variable |

Parameters:

| | |
|---|---|
| <cfg_var> | configuration variable. Refer to Table 7. Configuration variables for a complete list of the configurable variables. |

Result:

AT-S.Var:<var>=<value>

| | |
|---|---|
| <var> | Specifies the configuration variable name |
| <value> | Specifies the configuration variable value |

## 6.9 AT+S.SCFG

Command to set the value of the named configuration variable. The value is saved in the RAM until the command AT+S.WCFG is used.

Usage:

```
AT+S.SCFG=<key>,<value><cr>
```

Parameters:

| | |
|---|---|
| <key> | variable to configure |
| <value> | value to be set |

## 6.10 AT+S.WCFG

Command to save the configuration set to the Flash.

Usage:

```
AT+S.WCFG<cr>
```

Parameters:

None

## 6.11 AT+S.FCFG

Command to restore the factory configuration variables from the Flash. It is mandatory to run a reset (HW or SW) after a factory restore.

Usage:

```
AT+S.FCFG<cr>
```

*Note: HW factory restore of the variables is performed by pulling the pin GPIO0 high at power-up (until the "+WIND:1:Poweron" indication is printed). In order to use the HW factory reset (GPIO0 enabled) and FWUPDATE at the same time, see Section 6.13 AT+S.FSUPDATE.*

Parameters:

None

## 6.12 AT+S.FSWRITE

The command is used to update the external volume via serial interface.

*Note: The HW flow control MUST be enabled in order to use the command via UART.*

Usage:

AT+S.FSWRITE=<length><cr>{data}

Parameters:

<length> Data length to send (in bytes)

## 6.13 AT+S.FSUPDATE

To download an updated file system from the named host and path. The downloaded image overwrites the existing one. In this case, the user needs to perform a backup of the current file system.

Usage:

```
AT+S.FSUPDATE=<mem>,<hostname>,[<path&queryopts>],[<port>],[<TLS>],[<username>],
[<passwd>]<cr>
```

Parameters:

| | |
|---|---|
| <mem> | specifies the memory where the file system is saved.<br>• e → user Flash<br>• i → application Flash<br>• x → external memory volume |
| <hostname> | Target host. DNS resolvable name or IP address. |
| <path&queryopts> | Default:/fs.img. Document path and optional query arguments. |
| <port> | Default 80 (if TLS=0) or 443 (if TLS>0). |
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none. |
| <passwd> | Default: none. |

## 6.14 AT+S.FWUPDATE

This command downloads an updated firmware image located at the named host and path. The downloaded image is temporary stored in the internal Flash. The user should perform a procedure to save the content of the file system that will be overwritten during the process.

*Note: FWUPDATE can be invalidated by pulling GPIO0 high during the first reset after the execution of AT +S.FWUPDATE command. In this case, the external filesystem is erased and the FW update is not performed.*

Usage:

*Note: The HW factory reset pin (GPIO0) must be tight low during F/W update. The HW factory reset can be used after "+WIND:17:F/W update complete!"*

```
AT+S.FWUPDATE=e,<hostname>,[<path&queryopts>],[<port>],[<TLS>],[<username>],
[<passwd>]<cr>
```

Parameters:

| | |
|---|---|
| <hostname> | Target host. DNS resolvable name or IP address. |

| | |
|---|---|
| <path&queryopts> | Default: /fw.fota. Document path and optional query arguments. If a secure FOTA is required, the extension of the file needs to be ".sfota". |
| <port> | Default 80 (if TLS=0) or 443 (if TLS>0). |
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none. |
| <passwd> | Default: none. |

## 6.15 AT+S.GPIOC

Command used to configure the function of the various GPIOs on the module. GPIOs can be configured as inputs or outputs. When used as inputs, they generate an interrupt when the state changes that can be configured on the signal edge.

Usage:

| | |
|---|---|
| 1) AT+S.GPIOC=<number>,out<cr> | to configure a gpio as an output |
| 2) AT+S.GPIOC=<number>,in,[<interrupt>]<cr> | to configure a gpio as an input |

Parameters:

| | |
|---|---|
| <number> | GPIO Number (see Datasheet) |
| <interrupt> | Default: No interrupt (turn it off, if enabled). It can assume one of the following values:<br>• R → Rising edge<br>• F → Falling edge<br>• B → Both rising and falling edges |

## 6.16 AT+S.GPIOR

AT+S.GPIOR is used to read the value and the direction of a previously-configured GPIO.

Usage:

```
AT+S.GPIOR=<num><cr>
```

Parameters:

| | |
|---|---|
| <num> | specifies the GPIO to read |

Result:

```
AT-S.Value:<num>:<level>:<direction>
```

| | |
|---|---|
| <num> | Specifies the number of GPIO read |
| <level> | 0 (low) or 1 (high) |
| <direction> | 0 (output pullup) or 1 (input) |

## 6.17 AT+S.GPIOW

AT+S.GPIOW is used to set the value of a previously-configured output GPIO.

Usage:

```
AT+S.GPIOW=<num>,<level><cr>
```

Parameters:

| | |
|---|---|
| <num> | specifies the GPIO to be set |
| <level> | specifies one possible value(0\|1) for the specified GPIO. |

## 6.18 AT+S.DAC

The DAC command enables DAC on GPIO15.

Usage:

```
AT+S.DAC=<value><cr>
```

Parameters:

| | |
|---|---|
| <value> | must be set in mV (between 1 and 3300). The value=0 disables DAC on GPIO15 |

## 6.19 AT+S.ADC

AT+S.ADC returns ADC value on the selected GPIO. The value range is between 0 and 3300 mV with a measurement accuracy of 10 mV.

Usage:

```
AT+S.ADC=<num><cr>
```

Parameters:

| | |
|---|---|
| <num> | specifies the GPIO to be used for conversion. Available GPIOs are 0, 1 and 16 |

Result:

```
AT-S.Value:<value>
```

| | |
|---|---|
| <value> | Specifies the ADC value |

## 6.20 AT+S.PWM

The PWM command enables PWM on the selected GPIO, with a specified frequency and duty-cycle.
Usage:

```
AT+S.PWM=<num>,<frequency>[,<duty_cycle>]<cr>
```

Parameters:

| | |
|---|---|
| <num> | specifies the GPIO to be used for PWM. Available GPIOs are 2 and 4 |
| <frequency> | value between 1 and 10 kHz. The value=0 disables PWM |
| <duty_cycle> | default: 50%. The value is in the 0 - 100 range. |

## 6.21 AT+S.TIME

Command to get or set the time (date, time)
Usage:

| | |
|---|---|
| 1) AT+S.TIME<cr> | returns the current date in the format “AT-S.Date:yy.mm.dd:nn” and the current time in the format “AT-S.Time:hh:mm:ss”. The time refers to UTC format and must be expressed as the time in seconds since 1970-Jan-01. |
| 2) AT+S.TIME=<time> | sets the time as specified by the parameter |

Parameters:

| | |
|---|---|
| <time> | value to set the time. The time refers to UTC format and it must be expressed in seconds since 1970-Jan-01. |

Result:

```
AT-S.Date:<yy>.<mo>.<dd>:<day>
AT-S.Time:<hh>.<mi>.<ss>
```

| | |
|---|---|
| <yy> | Specifies the year |

| | |
|---|---|
| <mo> | Specifies the month |
| <dd> | Specifies the day |
| <day> | Specifies the year of the week |
| <hh> | Specifies the hours |
| <mi> | Specifies the minutes |
| <ss> | Specifies the seconds |

## 6.22 AT+S.RANDOM

Command that provides a random number generated by the peripheral integrated in the STM32.

Usage:

```
AT+S.RANDOM<cr>
```

Parameters:

None

Result:

```
AT-S.Number:<value>
```

| | |
|---|---|
| <value> | Specifies the 32-bit random number |

## 6.23 AT+S.FSC

This command has effect in the RAM memory volume and external SD card volume. The command creates a file in the selected volume or appends the data following the command in case the file already exists. The space available in the file system in RAM can be set using ramdisk_memsize configuration variable. Minimum size is 2 Kb, resulting in maximum 2 files.

Usage:

```
AT+S.FSC=<filename>,<datalen><cr>{data}
```

Parameters:

| | |
|---|---|
| <filename> | name of the file. Max size is 64 bytes |
| <datalen> | amount of space in bytes to allocate for the file |

## 6.24 AT+S.FSD

The command has effect in the RAM memory volume and external SD card volume. The command deletes an existing file.

Usage:

```
AT+S.FSD=<filename><cr>
```

Parameters:

| | |
|---|---|
| `<filename>` | name of the file to be deleted |

## 6.25 AT+S.FSR

The command has effect in the RAM memory volume and external SD card volume. The command renames an existing file.

Usage:

```
AT+S.FSR=<old_filename>,<new_filename><cr>
```

Parameters:

| | |
|---|---|
| `<old_filename>` | filename to be renamed |
| `<new_filename>` | new file name |

## 6.26 AT+S.FSL

This command lists the existing filenames together with their prefix that indicates where they are stored and the size.

| | |
|---|---|
| X → External Flash | (Volume ID:0) |
| E → User Flash (stored at STM32F439 address 0x08110000) | (Volume ID:1) |
| D → RAM | (Volume ID:2) |
| I → Application Flash (stored at STM32F439 address 0x08100000) | (Volume ID:3) |

Note that Volume ID 3, Application Flash, when protected (app_fs status variable set to 2) does not show content files.

Usage:

| | |
|---|---|
| 1) `AT+S.FSL<cr>` | print the list of the files on the std output |
| 2) `AT+S.FSL=[<output_filename>]` | print the list of the files in the specified name |

Parameters:

| | |
|---|---|
| <output_filename> | Default:2:/dir.txt. When this option is specified, the output of the command is saved in the named filename |

Result:

```
AT-S.File:<flag>\t<len>\t<name>
```

| | |
|---|---|
| <flag> | X (External Flash) or E (User Flash) or D (RAM) or I (Application Flash) |
| <len> | Specifies the file length |
| <name> | Specifies the file name |

## 6.27 AT+S.FSP

This command prints the content of the specified file starting from the offset and for the specified length.

Usage:

```
AT+S.FSP=<filename>,[<offset>],[<len>]<cr>
```

Note that Volume ID 1, Application Flash, when protected (app_fs status variable set to 2) does not allow print on selected file.

Parameters:

| | |
|---|---|
| <filename> | name of the file to be printed |
| <offset> | Default Value:0; indicates the byte from where the file is printed |
| <len> | Default Value: Filesize-Offset; indicates the number of bytes that are printed |

## 6.28 AT+S.FSM

The command is used to mount the selected volume.

Usage:

AT+S.FSM=[<volume>]<cr>

Parameters:

<volume> Default:0; Indicates the memory volume

## 6.29 AT+S.FSU

The command is used to umount/erase the user memory volumes.

Usage:

```
AT+S.FSU=[<volume>],[<erase>]<cr>
```

Parameters:

| | |
|---|---|
| <volume> | Default:0; Indicates the memory volume. |
| <erase> | Default:0; when”1”, the specified volume is fully erased |

## 6.30 AT+S.HASH

The command is used to compute a message digest of a given file.

Usage:

```
AT+S.HASH=[<function>],<filename><cr>
```

Parameters:

| | |
|---|---|
| <function> | Default:0; Indicates the selected hash function. Possible values are 0: SHA1; 1: SHA224; 2: SHA256; 3: MD5 |
| <filename> | filename to be used |

Result:

AT-S.SHA1:<digest> or

AT-S.SHA224:<digest> or

AT-S.SHA256:<digest> or

AT-S.MD5:<digest>

| | |
|---|---|
| <digest> | Specifies the digest of given file |

## 6.31 AT+S.WPAECERT

To manage WPA certificates in the module Flash memory.

Usage:

| | |
|---|---|
| AT+S.WPAECERT=content,<option><cr> | to list or to remove certificates within the Flash |
| AT+S.WPAECERT=<kind>,<size><cr>{data} | to load certificates/key within the module memory |

Parameters:

| | |
|---|---|
| <option> | When the value is equal to 1, the command lists the certificates and key stored in the Flash memory. When the value is equal to 2 the command removes certificates and keys stored in the Flash. |

| | |
|---|---|
| <kind> | Possible values are as follows:<br>• Ca → {data} correspond to the RADIUS CA<br>• Cert → {data} correspond to the SPWF04S certificate<br>• Key → {data} correspond to the SPWF04S key |
| <size> | Data size in bytes. Data can be transferred in PEM (textual) or in DER (binary) format. |

## 6.32 AT+S.TLSCERT

To Manage TLS certificates in the Module Memory. Note that if Flash is empty, the TLS layer will look for files in the filesystem with the default names of “tls.cert” and “tls.key” for client certificate and key, while CA certificates need filenames corresponding to their Subject Key Identifier (e.g.: "C07A98688D89FBAB05640C117DAA7D65B8CACC4E.ca"). Note also that depending on the type of certificate format loaded (DER or PEM), the subject key identifier is automatically found by the module itself. The “auth” parameter is only required when loading PEM format.

Usage:

```
AT+S.TLSCERT=content,<option><cr>
AT+S.TLSCERT=<kind>,<size><cr>{data}
```

Parameters:

| | |
|---|---|
| <option> | When the value is equal to 1, the command lists the certificates and key stored in the Flash memory. When the value is equal to 2, the command removes the certificates and keys stored in the Flash. |
| <kind> | Possible values are as follows:<br>• Ca → {data} correspond to the Peer CA<br>• Cert → {data} correspond to the SPWF04S certificate<br>• Key → {data} correspond to the SPWF04S key<br>• Auth → {data} peer’s CA Authority Key ID |
| <size> | Data can be transferred in PEM (textual) or in DER (binary) format. |

## 6.33 AT+S.WPS

Initiate a WPS exchange.

Usage:

```
AT+S.WPS=<mode><cr>
```

Parameters:

| | |
|---|---|
| <mode> | mode=0: emulates the push button as defined by the WPS standard<br>mode=1: executes WPS PIN mode using the factory PIN of the device (stored in the configuration variable wifi_wps_pin)<br>mode=2: executes WPS PIN mode using a random generated PIN |

Result:

```
AT-S.Generated user PIN:<pin> or
AT-S.Generated random PIN:<pin>
```

| | |
|---|---|
| <pin> | Specifies the 8-digit pin used for WPS exchange |

## 6.34 AT+S.WIFI

AT+S.WIFI allows the radio to be enabled or disabled at run-time. Please note that the configuration variable "wifi_mode" controls the state of the radio at power-up.

Usage:

```
AT+S.WIFI=<action><cr>
```

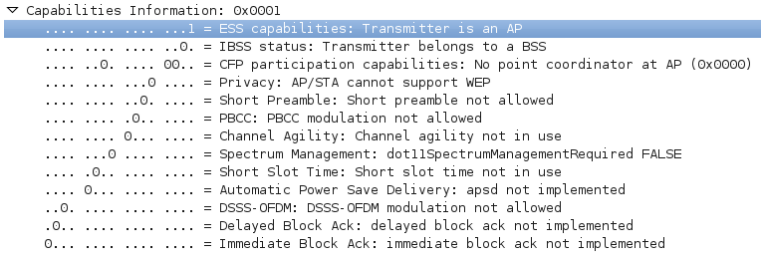
Parameters:

| | |
|---|---|
| <action> | can assume the values: 0: radio is off, 1: radio is on |

## 6.35 AT+S.SCAN

AT+S.SCAN performs an immediate scan for available networks. The output reports the BSSID, SSID, the network channel, signal strength (RSSI), AP capabilities and 802.11 options included security.

**Figure 6. AP capabilities list**




Usage:

```
AT+S.SCAN=<filter>,[<filename>]<cr>
```

Parameters:

| | |
|---|---|
| <filter> | Range of possible values is: d → no filter, s → filter on ssid, m → filter on the mac |
| <filename> | Default:std output. When specified, the output of the command is saved in the <filename> file. |

## 6.36 AT+S.SSIDTXT

AT+S.SSIDTXT allows to get/set the current value of the variables wifi_ssid and wifi_ssid_len as text strings

Usage:

| | |
|---|---|
| AT+S.SSIDTXT<cr> | get the value of the variable wifi_ssid. |
| AT+S.SSIDTXT=<ssidtxt><cr> | set the value of the variables wifi_ssid and wifi_ssid_len |

Parameters:

| | |
|---|---|
| <ssidtxt> | SSID in textual format |

Result:

AT-S.Network:<SSID>

| | |
|---|---|
| <SSID> | Specifies the SSID of current configured network |

## 6.37 AT+S.PING

Send a single ICMP ECHO request to a specified remote host. When IPv6 is enabled, every network request for name resolution will use IPv6 at the beginning. This means that WIND:24:IPv6 must be waited when ip_use_v6.

Usage:

AT+S.PING=[<counter>],[<size>],<hostname><cr>

Parameters:

| | |
|---|---|
| <counter> | Default value:1. Number of echo requests to send |
| <size> | Default value:56 bytes. Highest size of the echo packet. Every sent, or incoming, request will be truncated |
| <hostname> | Remote Host. DNS resolvable name or IP address |

## 6.38 AT+S.SOCKON

To open a socket client. Up to 8 socket clients can be set simultaneously on the device. The command returns the identifier to be used for operations on the socket.

Usage:

```
AT+S.SOCKON=<hostname>,<port>,<NULL>,<kind><cr>
```

Parameters:

| | |
|---|---|
| <hostname> | Remote Server. DNS resolvable name or IP address |
| <port> | TCP/UDP socket port |
| <kind> | This parameter can assume the values:<br>• t → tcp<br>• u → udp<br>• s → TLS socket using <Hostname> as domain name.<br>• TLS Server Domain Name → Common name of the server (URL or the CN field reported into server certificate) for TLS socket |

Result:

```
AT-S.On:<IP>:<id>
```

| | |
|---|---|
| <IP> | Specifies the server IP address |
| <id> | Specifies the socket client identifier |

## 6.39 AT+S.SOCKQ

To query a socket client for pending data. The command returns the number of bytes that are waiting on the socket.

Usage:

```
AT+S.SOCKQ=<id><cr>
```

Parameters:

| | |
|---|---|
| <id> | Socket Client ID |

Result:

AT-S.Query:<len>

| | |
|---|---|
| <len> | Specifies the number of bytes in socket client buffer |

## 6.40 AT+S.SOCKC

To close a socket client.

Usage:

AT+S.SOCKC=<id><cr>

Parameters:

| | |
|---|---|
| <id> | Socket Client ID |

**Note:** When a socket client receives an indication about socket server gone (only for TCP/TLS sockets, WIND:58), the socket resource is not automatically cleared. Moreover, flushing pending data (using the AT +S.SOCKR command) is mandatory before closing the socket connection (AT+S.SOCKC). If the buffer is not empty, the “ERROR:Pending data” is raised.

## 6.41 AT+S.SOCKW

To write data to a socket client. The host is expected to send <length> characters of data after the end of the command.

Usage:

AT+S.SOCKW=<id>,<length><cr>{data}

Parameters:

| | |
|---|---|
| <id> | Socket Client ID |
| <length> | Data length to send (in bytes) |

## 6.42 AT+S.SOCKR

To read data from a socket client.

Usage:

AT+S.SOCKR=<id>,[<length>]<cr>

Parameters:

| | |
|---|---|
| <id> | Socket Client ID |
| <length> | Defaut:0. Length (in bytes) of the buffer to read. The value 0 indicates to read the full buffer |

## 6.43 AT+S.SOCKL

To list opened socket clients.

Usage:

```
AT+S.SOCKL<cr>
```

Parameters:

None

Result:

```
AT-S.List::<id>:<connected>:<kind>:<len>:<IP>:<port>
```

| | |
|---|---|
| <id> | Specifies the socket client identifier |
| <connected> | Specifies whether socket client is connected to socket server |
| <kind> | Specifies the kind of socket client |
| <len> | Specifies the number of bytes in socket client buffer |
| <IP> | Specifies the server IP address |
| <port> | Specifies the client port |

## 6.44 AT+S.SOCKDON

To open a socket server. The device can manage up to 2 socket servers each supporting 8 socket clients. The command returns the server socket ID to be used in the related commands.

Usage:

```
AT+S.SOCKDON=<port>,<kind><cr>
```

Parameters:

| | |
|---|---|
| <port> | socket port |
| <kind> | specifies the kind of socket as one of the following:<br>• t → tcp<br>• u → udp<br>• s1 → TLS socket with one-way authentication<br>• s2 → TLS socket with mutual authentication |

Result:

```
AT-S.On:<id>
```

| | |
|---|---|
| <id> | Specifies the socket server identifier |

## 6.45 AT+S.SOCKDQ

To query a socket server for pending data.

Usage:

```
AT+S.SOCKDQ=<sid>,<cid><cr>
```

Parameters:

| | |
|---|---|
| <sid> | socket server identifier |
| <cid> | socket client identifier |

Result:

```
AT-S.Query:<len>
```

| | |
|---|---|
| <len> | Specifies the number of bytes in client buffer |

## 6.46 AT+S.SOCKDC

To close a socket server or to disconnect a client.

Usage:

| | |
|---|---|
| AT+S.SOCKDC=<sid> | to close the specified socket server (and disconnect all clients) |
| AT+S.SOCKDC=<sid>[,<cid>] | to disconnect the specified client, if client_id is given. If not, all clients are disconnected, and server is closed |

Parameters:

| | |
|---|---|
| <sid> | socket server identifier |
| <cid> | socket client identifier |

**Note:** Flushing pending data (using the AT+S.SOCKDR command) is mandatory before closing the socket connection (AT+S.SOCKDC). If the buffer is not empty, the “ERROR: Pending data” is raised.

## 6.47 AT+S.SOCKDW

To write data to a socket server.

Usage:

```
AT+S.SOCKDW=<sid>,<cid>,<len><cr>{data}
```

Parameters:

| | |
|---|---|
| <sid> | socket server identifier |
| <cid> | socket client identifier |
| <len> | length (in bytes) of the buffer to write that is sent after the command. |

## 6.48 AT+S.SOCKDR

To read data from a socket server.

Usage:

```
AT+S.SOCKDR=<sid>,<cid>,[<len>]<cr>
```

Parameters:

| | |
|---|---|
| <sid> | socket server identifier |
| <cid> | socket client identifier |
| <len> | Default:0. length (in bytes) of the buffer to read. The value=0 indicates to read the entire buffer. |

## 6.49 AT+S.SOCKDL

To list the bound socket client.

Usage:

```
AT+S.SOCKDL
AT+S.SOCKDL=<sid><cr>
```

to list bound clients on all servers

Parameters:

| | |
|---|---|
| <sid> | socket server identifier |

Result:

```
AT-S.List:<id>:<cid>:<connected>:<kind>:<len>:<IP>:<port>
```

| | |
|---|---|
| <id> | Specifies the socket server identifier |
| <cid> | Specifies the socket client identifier |
| <connected> | Specifies whether socket client is connected to socket server |
| <kind> | Specifies the kind of socket client |
| <len> | Specifies the number of bytes in client buffer |
| <IP> | Specifies the client IP address |
| <port> | Specifies the client port |

## 6.50 AT+S.TFTPGET

The stack implements the TFTP client protocol to transfer files on a UDP port. The command performs a request to a specified TFTP server.

Usage:

```
AT+S.TFTPGET=<hostname>,[<port>],<filename>,<local_filename><cr>
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable name or IP address of the TFTP remote server |
| <port> | Default:69. Socket UDP port |
| <filename> | filename to get from the remote host. It contains the complete path |
| <local_filename> | Default:2:<Filename>. Filename used locally |

## 6.51 AT+S.TFTPPUT

The stack implements the TFTP client protocol to transfer files on a UDP port. The command performs a request to a specified TFTP server.

Usage:

```
AT+S.TFTPPUT=<hostname>,[<port>],<local_filename><cr>
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable name or IP address of the TFTP remote server |
| <port> | Default: 69. Socket UDP port |
| <local_Filename> | filename to send to the remote host |

## 6.52 AT+S.SMTP

The stack implements the protocol SMTP to send an email.

Usage:

```
AT+S.SMTP=<hostname>,[<port>],[<TLS>],[<username>],[<passwd>],
[<ID>],<address>,<to>,<NULL>,<NULL>,<subject>,<NULL>,<len><cr>{data}
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable Name or IP address of the remote host |
| <port> | Default is 25 (if TLS=0) or 465 (if TLS>0). Server port |
| <TLS> | Default:unsecured TLS Security option. 0 → unsecured, 5 → SMTPS on port 465 if available, otherwise SMTP + STARTTLS if available, otherwise no security, 8 → SMTP + STARTTLS if available, otherwise the mail is not sent, 9 → SMTPS on port 465 if available, otherwise SMTP + STARTTLS if available, otherwise the mail is not sent. |
| <username> | User of the SMTP server |
| <passwd> | Password of the SMTP server |
| <ID> | Default: nv_model used during Helo |
| <address> | Email address on the SMTP server |
| <to> | Email recipients. Multiple emails are separated by a semicolon<br>A recipient can be just an email address, or be in the extended format consisting of a name followed by a '<', followed by the email address and terminated by a '>' (e.g. "Nick <nick@name.xy>") |
| <subject> | Email Subject. String message |
| <len> | Length of the Body message |

## 6.53 AT+S.HTTPGET

To perform a single http get request to the named host and path.

Usage:

```
AT+S.HTTPGET=<hostname>,[<path&queryopts]>,[<port>],[<TLS>],[<username>],
[<passwd>],[<in_filename>],[<out_filename>]<cr>
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable Name or IP address |
| <path&queryopts> | Default:/index.html. document path & optional query arguments |
| <port> | Default 80 (if TLS=0) or 443 (if TLS>0) |

| | |
|---|---|
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none |
| <passwd> | Default: none |
| <in_filename> | Default: none. Custom http requests |
| <out_filename> | Default: Console. When specified the return data are saved in a file |

## 6.54 AT+S.HTTPPOST

To perform a post of the specified file to a remote host.

Usage:

```
AT+S.HTTPPOST=<hostname>,[<path&queryopts>],[<port>],[<TLS>],[<username>],
[<passwd>],[<in_filename>],[<out_filename>]<cr>
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable Name or IP address |
| <path&queryopts> | Default:/index.html. document path & optional query arguments |
| <port> | Default 80 (if TLS=0) or 443 (if TLS>0) |
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none |
| <passwd> | Default: none |
| <in_filename> | Default: none. Console. When specified the return data are saved in a file |
| <out_filename> | Default: none. Filename to transfer to the server. Please refer to following table the right association between file extension and HTTP Content-Type. |

**Table 5. Association between file extension and HTTP content-type**

| File extension | Content-type |
|---|---|
| html | text/html |
| fhtml | text/html |
| css | text/css |
| txt | text/plain |
| xml | text/xml |
| xls | text/xls |
| gif | image/gif |
| jpg | image/jpg |
| bmp | image/bmp |

| File extension | Content-type |
|---|---|
| png | image/png |
| ico | image/x-icon |
| class | application/x-java-applet |
| swf | application/x-shockwave-flash |
| json | application/json |
| woff | application/font-woff |
| form | application/x-www-form-urlencoded |
| unknown extension | application/x-raw-stuff |

## 6.55 AT+S.INPUTSSI

Fill buffer for raw text input SSI <!--|06|Input|index|-->. See AN4965 for details on the use of the SSI.

Usage:

```
AT+S.INPUTSSI=<length><cr>{data}
```

Parameters:

| | |
|---|---|
| <length> | Length of the data to save in the buffer. The length=0 cleans the buffer. |

**Note**: {data} used to fill the buffer must be properly set. The first byte is used as a token splitter; so use this special character to fill the buffer with multiple tokens. Every token can be directly referred remotely by SSI indexes contained inside the HTML page. For example: “|hello|world|” will create 2 tokens: "hello” accessed by <!--|06|Input|0|-->, and “world” accessed by <!--|06|Input|1|-->.

## 6.56 AT+S.MQTTCONN

To open a connection with an MQTT Broker. The command returns a local ID =0, used in the correspondent commands. The device manages one MQTT connection at a time.

Usage:

```
AT+S.MQTTCONN=<hostname>,[<port>],[<path>],[<TLS>],[<username>],[<passwd>],
[<userID>],[<keep_alive>],[<retry>],[<lastWill_QoS>],[<lastWill_topic>],
[<lastWill_message>]<cr>
```

Parameters:

| | |
|---|---|
| <hostname> | DNS resolvable name or IP address of the MQTT broker |
| <port> | Default:1883 |
| <path> | Default:/ |

| | |
|---|---|
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none. User Name |
| <passwd> | Default: none. Passwd |
| <userID> | Default: nv_model used during MQTT communications |
| <keep_alive> | Default: 60 seconds |
| <retry> | Default: 15 seconds |
| <lastWill_QoS> | Default: 0. Last action to be executed by the broker when the node disappears without a disconnect procedure |
| <lastWill_topic> | Default: None. Last Will Topic |
| <lastWill_message> | Default: None. Published on the Last Will Topic |

Result:

```
AT-S.On:<id>:<session>
```

| | |
|---|---|
| <id> | Specifies the MQTT client identifier |
| <session> | 0 (new session) or 1 (session restored) |

## 6.57 AT+S.MQTTPUB

To publish a message to an MQTT broker

Usage:

```
AT+S.MQTTPUB=0,<topic>,[<QoS>],[<retained_flag>],<len><cr>{data}
```

Parameters:

| | |
|---|---|
| <topic> | Topic where the message is published |
| <QoS> | Default: 0. Values Range: 0 → at most once delivery; 1 → at least one delivery; 2 → exactly one delivery |
| <retained_flag> | Default: 0. Possible values: 0 → do not retain, 1 → retain |
| <len> | MQTT message length |

## 6.58 AT+S.MQTTDISC

To disconnect from an MQTT broker.

Usage:

```
AT+S.MQTTDISC=0<cr>
```

Parameters:

None

## 6.59 AT+S.MQTTSUB

To subscribe topic to an MQTT broker.

Usage:

```
AT+S.MQTTSUB=0,<topic>,[<QoS>]<cr>
```

Parameters:

| | |
|---|---|
| `<topic>` | Topic where the node subscribes |
| `<QoS>` | Default:0. Values Range: 0 → at most once delivery; 1 → at least one delivery; 2 → exactly one delivery |

## 6.60 AT+S.MQTTUNSUB

To unsubscribe topic from an MQTT broker.

Usage:

```
AT+S.MQTTUNSUB=0,<topic><cr>
```

Parameters:

| | |
|---|---|
| `<topic>` | Topic where the node unsubscribes |

## 6.61 AT+S.WSOCKON

The module supports the usage of two web socket clients. The command returns the ID that is used in the corresponding commands.

The MQTT layer is built on top of web sockets. This means that (when an MQTT broker has been connected through AT+S.MQTTCONN) the total amount of available web sockets decreases to just 1. Note that it is not possible to run another AT+S.WSOCKON to the same web socket server.

Usage:

```
AT+S.WSOCKON=<hostname>,[<port>],[<path>],[<TLS>],[<username>],[<passwd>],
[<origin>],[<protocols>],[<extensions>]<cr>
```

Parameters:

| | |
|---|---|
| `<hostname>` | DNS resolvable name or IP address of the web socket server |
| `<port>` | Default 80 (if TLS=0) or 443 (if TLS>0) |

| | |
|---|---|
| <path> | Default:/ |
| <TLS> | Default: 0. Values range: 0 → unsecured; 1 → autodetect; 2 → TLS |
| <username> | Default: none. Username on the remote server |
| <passwd> | Default: none. Password on the remote server |
| <origin> | Default:none. Header Field Origin |
| <protocols> | Default:none. Header Field Protocols |
| <extensions> | Default:none. Header Field Extensions |

Result:

```
AT-S.On:<id>
```

| | |
|---|---|
| <id> | Specifies the Web Socket client identifier |

## 6.62 AT+S.WSOCKQ

To query web socket client for pending data.

Usage:

```
AT+S.WSOCKQ=<id><cr>
```

Parameters:

| | |
|---|---|
| <id> | Web socket client identifier |

Result:

```
AT-S.Query:<len>
```

| | |
|---|---|
| <len> | Specifies the number of bytes in Web Socket client buffer |

## 6.63 AT+S.WSOCKC

To close a web socket client using a specific status code.

Usage:

```
AT+S.WSOCKC=<id>,[<status>]<cr>
```

Parameters:

| | |
|---|---|
| <id> | Web socket client identifier |
| <status> | Default:0; 0 → Normal Closure; 1 → Going Away; For a complete list of the status values defined for the web socket, refer to the related standard |

**Note:** When a web socket client receives an indication about web socket server gone (WIND:89), the web socket resource is not automatically cleared. Moreover, flushing pending data (using the AT+S.WSOCKR command) is mandatory before closing the socket connection (AT+S.WSOCKC). If the buffer is not empty, the "ERROR:Pending data" is raised.

## 6.64 AT+S.WSOCKW

To write data to a web socket.

Usage:

```
AT+S.WSOCKW=<id>,[<last_frame>],[<last_frag>],[<binary>],<len><cr>{data}
```

Parameters:

| | |
|---|---|
| <id> | Web socket client identifier |
| <last_frame> | Default:0. 1 → Last frame flag |
| <last_frag> | Default:0. 1 → Last frag flag |
| <binary> | Default:0, textual. 1 → Binary flag |

## 6.65 AT+S.WSOCKR

To read data from a web socket client.

Usage:

```
AT+S.WSOCKR=<id>,[<len>]<cr>
```

Parameters:

| | |
|---|---|
| <id> | Web socket client identifier |
| <len> | Default 0. 0 value indicates read the full buffer |

## 6.66 AT+S.WSOCKL

To list opened web socket clients.

Usage:

```
AT+S.WSOCKL<cr>
```

Parameters:

None

Result:

```
AT-S.List:<id>:<connected>:<len>>:<port>
```

| | |
|---|---|
| <id> | Specifies the Web Socket client identifier |
| <connected> | Specifies wheter Web Socket client is connected to Web socket server |
| <len> | Specifies the number of bytes in Web Socket client buffer |
| <port> | Specifies the local port |

# 7 Technology partners

Table 6. **Technology partners**

| Partner | Device description | Part number |
|---|---|---|
| MACRONIX | 8-Mbit CMOS Serial Flash memory | MX25L8006E |
| MACRONIX | 64-Mbit CMOS Serial Flash memory | MX25L6433F |

# 8 Variables and values

The following table lists the configuration variables together with their default values. The list of configuration variables is generated by the command AT+S.GCFG.

**Table 7. Configuration variables**

| Variable name | Default value | Type | Description |
|---|---|---|---|
| nv_manuf | STMicroelectronics Inc. | TEXT[32] | Manufacturer ID string |
| nv_model | SPWF04SA | TEXT[32] | Manufacturer Model String |
| nv_serial | 0517<11222 | TEXT[32] | Manufacturer Serial Number |
| nv_wifi_macaddr | 00:80:E1:FA:12:34 | HEX[6] | Manufacturer assigned 802.11 MAC Address |
| standby_time | 10 | INT | Standby mode time, in seconds. Up to 2exp(32)-1 sec |
| standby_enabled | 0 | INT | Enable/disable the standby mode |
| sleep_enabled | 0 | INT | Enable/Disable the sleep mode |
| etf_mode | 0 | INT | Enable/Disable the Engineering test functions. It is used to control the radio for the certification tests |
| blink_led | 1 | INT | Manage led drive GPIO(10). 0-off 1-0n |
| ext_volume | 3 | INT | Used to manage ext. storage. 0-disabled, 1-SDCARD, 2-SPI, 3-Autodetection |
| ramdisk_memsize | 16 | INT | Heap size dedicated to RAM disk in unit of 1024 byte |
| aes128_key | 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[16] | AES-128 key used to encrypt FOTA image |
| user_desc | anonymous | TEXT[16] | Password used during remote configuration |
| python_script | 3:/uPython_test.py | TEXT[64] | Specifies the python script to run when GPIO[8] is high at the boot time |
| python_memsize | 32 | INT | Heap size dedicated to MicroPython execution in unit of 1024 byte |
| console_enabled | 1 | INT | 0→SPI; 1→ UART; 2→UART+Python; 3→Python |
| console_speed | 115200 | INT | Serial Port Speed: from 9600 to 921600. |
| console_hwfc | 0 | INT | Hardware Flow Control. 0=off; 1=on |
| console_echo | 1 | INT | Echo command input. 0=off, 1=on |
| console_errs | 2 | INT | ERROR:#:string Format.<br>Format Selection. 0→ simple notification 1:→ numbered notification 2→ verbose notification |
| console_winds | 2 | INT | WIND:#:string:p<br>Format Selection. 0→ simple notification 1:→ numbered notification 2→ verbose notification |
| console_verbose | 1 | INT | Verbose output. 0=off; 1=on, with short OK message; 2=full on |
| console_repeater | 0x21 | HEX | Set the console repeater. |
| console_delimiter | 0x2C | HEX | Set the console delimiter. |

| Variable name | Default value | Type | Description |
|---|---|---|---|
| console_wind_off_low | 0x00000000 | HEX | Wind 0:31 mask<br>0xFFFFFFFF are disabled<br>all the 32 Wind indicator |
| console_wind_off_medium | 0x00000000 | HEX | Wind 32:63 mask |
| console_wind_off_high | 0x00000000 | HEX | Wind 64:95 mask |
| wifi_tx_msdu_lifetime | 0 | INT | MSDU lifetime. From 0 to 2^32-1 TUs (1 TUs=<br>1024µs). Zero is default (automatic) |
| wifi_rx_msdu_lifetime | 0 | INT | MSDU lifetime. From 0 to 2^32-1 TUs (1 TUs=<br>1024µs). Zero is default (automatic) |
| wifi_operational_mode | 0x00000011 | INT | Allows choosing Doze (11) or quiescent (12) power device modes |
| wifi_beacon_wakeup | 1 | INT(sec) | Set the wakeup interval of the WLAN device, from 1 to 255 if wifi_listen_interval = 0; from 1 to 65535 if wifi_listen_interval = 1 |
| wifi_beacon_interval | 100 | INT | Beaconing interval in MiniAP mode, from 0 to<br>2^16-1 |
| wifi_listen_interval | 0 | INT | Define the wakeup mode (0 = sleep up to the beacon_wakeup specified, 1 = sleep at least to the beacon_wakeup specified) |
| wifi_rts_threshold | 3000 | INT | Frame size over which RTS/CTS is used. Limit:<br>from 0 to 3000 |
| wifi_ssid | 53:54:54:65:73:74:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[32] | Can be get/set with the command AT +S.SSIDTXT. Desired SSID specified in hex. All 32 octets should be written. Note that wifi_ssid_len must also be set. |
| wifi_ssid_len | 6 | INT | Can be set with the command AT +S.SSIDTXT.<br>Length of the actual SSID<br>in the 32 byte buffer |
| wifi_txfail_thresh | 5 | INT | Maximum number of lost packets before disassociation |
| wifi_dtim_period | 1 | INT | Amount of frames stored for associated powersaving STAs. 0=do not store frames, 1=store 1 frame |
| wifi_add_tim_ie | 0 | INT | Whether or not to add the TIM information element in miniAP beacons. 0=do not add, 1=add |
| wifi_region | 1 | INT | Setting the scan process used during the command AT+S.SCAN. set the channels allowed for active scan (0 = always passive scan, 1 = active scan only on channels enabled by wifi_chan_activity2 status variable [default], 2 = USA X10 [1-11], 3 = Canada X20 [1-11], 4 = Europe ETSI X30 [1-13], 5 = France X32 [10,11], 6 = Japan X40 [1-13], 7 = Japan X41 [10,11,14] |

| Variable name | Default value | Type | Description |
|---|---|---|---|
| wifi_ht_mode | 1 | INT | Enable the 802.11n mode. The 11n data rates must be enabled with the wifi_opr_rate_mask variable (i.e. wifi_opr_rate_mask=3FFF CF to enable all the data rate supported) |
| wifi_channelnum | 1 | INT | Channel number to use for MiniAP operation. The user must properly set the channel number to not violate IEEE 802.11 Wi- Fi/ WLAN standards. |
| wifi_opr_rate_mask | 0x003FFFCF | INT | BIT0: 1 Mbps<br>BIT1: 2 Mbps<br>BIT2: 5.5 Mbps<br>BIT3: 11 Mbps<br>BIT6: 6 Mbps<br>BIT7: 9 Mbps<br>BIT8: 12 Mbps<br>BIT9: 18 Mbps<br>BIT10: 24 Mbps<br>BIT11: 36 Mbps<br>BIT12: 48 Mbps<br>BIT13: 54 Mbps<br>BIT14: MCS0 (6.5Mbps)<br>BIT15: MCS1 (13Mbps)<br>BIT16: MCS2 (19.5Mbps)<br>BIT17: MCS3 (26Mbps)<br>BIT18: MCS4 (39Mbps)<br>BIT19: MCS5 (52Mbps)<br>BIT20: MCS6 (58.5Mbps)<br>BIT21: MCS7 (65Mbps) |
| wifi_bas_rate_mask | 0x0000000F | INT | Basic data rate mask, 0x0000000f is [1,2,5.5,11] |
| wifi_mode | 1 | INT | Radio mode.<br>0=IDLE<br>1=STA (Supported Security Modes: OPEN, WEP OpenSystem, WEP SharedKey, WPA/ WPA2 - wifi_auth_type must be set to 0, WPA-Enterprise)<br>2=IBSS (Supported Security Modes: OPEN, WEP OpenSystem, WEP SharedKey)<br>3=MiniAP (Supported Security Modes: OPEN, WEP OpenSystem, WPA/WPA2 - wifi_auth_type must be set to 0, Supported Classes: b,g) * |
| wifi_auth_type | 0 | INT | Authentication type used in STA, IBSS and MiniAP mode: 0=OpenSystem, 1=SharedKey |
| wifi_atim_window | 0 | INT | Reserved |
| wifi_powersave | 0 | INT | Allows choosing between Active (0), PS (1) or Fast-PS (2) |
| wifi_tx_power | 18 | INT | Transmit power [from 0 to 18], in dBm |

| Variable name | Default value | Type | Description |
|---|---|---|---|
| wifi_rssi_thresh | 0 | INT | Low signal strength threshold |
| wifi_rssi_hyst | 0 | INT | Amount of change in RSSI to trigger signal state change |
| wifi_ap_idle_timeout | 120 | INT | Seconds of inactivity to trigger disassociate of the client |
| wifi_beacon_loss_thresh | 10 | INT | Number of consecutive loss beacon to detect the AP disassociation (0=network lost not notified, from 1 to 200) |
| wifi_priv_mode | 0 | INT | Privacy Mode: 0=none, 1=WEP, 2=WPA-Personal (TKIP/AES) or WPA2- Personal (TKIP/AES) - wifi_auth_type must be set to 0,3=WPA2-Enterprise |
| wifi_wep_keys[0] | 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[16] | WEP key buffer |
| wifi_wep_keys[1] | 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[16] | WEP key buffer |
| wifi_wep_keys[2] | 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[16] | WEP key buffer |
| wifi_wep_keys[3] | 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 | HEX[16] | WEP key buffer |
| wifi_wep_key_lens | 00:00:00:00 | HEX[4] | Four octets specifying the length of the actual key data in each WEP key buffer |
| wifi_wep_default_key | 0 | INT | WEP key index |
| wifi_wpa_psk_raw | 00:00:00:00:00:<br>00:00:00:00:00:<br>00:00:00:00:00:<br>00:00:00:00:00:<br>00:00:00:00:00:<br>00:00:00:00:00:<br>00:00 | HEX[32] | Pre-calculated PSK key |
| wifi_wpa_psk_text | | TEXT[64] | WPA(2) PSK passphrase, if set the actual PSK will be generated from this. Used in STA, IBSS and MiniAP. |
| wifi_eap_identity | identity | TEXT[32] | specifies the username by which the SPWF04S authenticates itself to the network |
| wifi_eap_anon_identity | anonymous@identity.org | TEXT[32] | specifies the anonymous identity string used in the first phase of most of the EAP authentication methods |
| wifi_eap_passwd | password | TEXT[32] | specifies the password by which the SPWF04S authenticates itself to the network. |
| wifi_eap_type | 0 | INT | EAP authentication method. 0=EAP-TLS; 1=TTLS-MD5, 2=TTLS-MSCHAPv2, 3=PEAP-MD5,4=PEAP-MSCHAPv2 |
| wifi_eap_skip_datechecks | 0 | INT | RADIUS certificate date validity check.<br>0→enabled; 1→disabled |
| wifi_wps_walk_time | 120 | INT | Button push time window length |

| Variable name | Default value | Type | Description |
|---|---|---|---|
| wifi_wps_pin | 1234567 | INT | Wps pin |
| ip_sock_memsize | 1 | INT | Heap size dedicated to every socket connection in unit of 1500 byte |
| ip_sock_threshold | 0 | INT | Percentage of ip_sock_memsize used to allow +WIND:55 and +WIND:88 indications |
| ip_dhcp_lease_time | 120 | INT | IP address renew of the peers in MiniAP mode (default: 120 sec) |
| ip_macfilter | 00:00:00:00:00:00 | HEX[6] | Mask for filtering of mac address during the association process. The IP is not released to not selected mac address |
| ip_num_clients | 5 | INT | Number of allowed clients. IP not released beyond this number |
| ip_allow_port_scans | 1 | INT | Scan from remote port enabler. 0=do not allow; 1=allow |
| ip_use_v6 | 1 | INT | IPv6 Enable. |
| ip_use_dhcpd | 1 | INT | DHCP Server Enable. To be Used in MiniAP mode. 0=off;1=on |
| ip_use_httpd | 1 | INT | HTTP Enable. 0=off;1=on |
| ip_use_tftpd | 1 | INT | TFTP Enable. 0=off;1=on |
| ip_use_dhcpc | 2 | INT | Dhcp Client. 0→off; 1→on; 2→ autoip |
| ip_hostname | iwm-FA-12-34 | TEXT[32] | IP local hostname |
| ip_apdomainname | captiveportal.net | TEXT[32] | Domain name in Mini AP mode. If the AP domain name is not quickly opened, it's suggested to turn off an eventual proxy server (check the connection settings of the device or the browser preferences). Please be sure to provide a standard extension (.net, .com, etc.) |
| ip_apredirect | firstset.html | TEXT[16] | Default homepage opening the ip_apdomainname in miniAP |
| ip_ipaddr | 192.168.0.50 | IP | IP address for static usage (DHCP off) |
| ip_netmask | 255.255.255.0 | IP | IP netmask for static usage (DHCP off) |
| ip_gw | 192.168.0.1 | IP | IP default gateway for static usage (DHCP off) |
| ip_dns1 | 208.67.222.222 | IP | IP Primary DNS server for static usage (DHCP off) |
| ip_dns2 | 208.67.220.220 | IP | IP Secondary DNS server for static usage (DHCP off) |
| ip_local | 0:0:0:0:0:0:0:0 | IPv6 | IPv6 unique local address |
| ip_dns1v6 | 0:0:0:0:0:0:0:0 | IPv6 | IPv6 Primary DNS server for static usage (DHCP off) |
| ip_dns2v6 | 0:0:0:0:0:0:0:0 | IPv6 | IPv6 Secondary DNS server for static usage (DHCP off) |
| ip_dhcp_timeout | 20 | INT | DHCP client timeout, in seconds |
| ip_ntp_server1 | ptbtime1.ptb.de | TEXT[32] | Primary NTP server (IPv4) |
| ip_ntp_server2 | ntp0.ipv6.fau.de | TEXT[32] | Secondary NTP server (IPv4 when ip_use_v6=0, IPv6 when ip_use_v6=1) |

| Variable name | Default value | Type | Description |
|---|---|---|---|
| ip_ntp_refresh | 3600 | INT | Refresh time interval |
| ip_ntp_startup | 1 | INT | NTP enabler at the startup. 0→off; 1→on |
| ip_mdns_domain_name | SPWF04S-Default | TEXT[32] | MDNS domain name |
| ip_mdns_device_name_ttl | 120 | INT | MDN Time To Live |
| ip_mdns_services_name | SPWF04S-WebSrv SPWF04S-TFTPSrv | TEXT[64] | Services list separated by space (max two services) |
| ip_mdns_services_prot | _http._tcp _tftp._udp | TEXT[64] | Services protocol list separated by space |
| ip_mdns_services_keys | dev1 dev2 | TEXT[32] | Services keys list separated by space |
| ip_mdns_services_vals | number1 number2 | TEXT[32] | Services values list separated by space |
| ip_mdns_services_port | 80 69 | INT | Services ports list separated by space |
| ip_mdns_services_ttl | 120 60 | INT | Services ttl list separated by space |
| ip_mdns_startup | 01:01 | HEX[2] | Services enabler at the startup separated by space |

The following table lists the status variables together with their associated values. The list of status variables is generated by the command AT+S.STS.

**Table 8. Status variables**

| Status variable | Value | Note |
|---|---|---|
| build | 170216-fd39c59-SPWF04S | SW Build Version |
| fw_version | 1.0.0 | SW Major Version |
| boot_version | 1.0 | Boot Version |
| var_version | 1 | Variable List version |
| free_heap | 55200 | Current free heap space |
| min_heap | 52576 | Minimum free heap space thus far |
| system_time | 1467386876 | Current Time in UTC format in sec |
| system_uptime | 1780 | System Running Time in sec |
| system_sleeptime | 0 | System Sleeping time in seconds |
| reset_reason | 2 | H/W reported reason for last reset<br>0 = POWER_ON<br>1 = WATCHDOG<br>2 = SOFT RESET<br>3 = LOW POWER<br>4 = HW RESET |
| startup | 0 | Failure reasons at the boot time (see wind:7 description) |
| random_number | 3725839442 | Random Number generated by the random generator with the AT+S.RANDOM command |
| gpio_enable | 0x0000 | Interrupt-enabled GPIO bitmask, expressed in HEX |
| app_fs | 1 | App Flash. 0 → FS not mounted; 1→ mounted; 2 → mounted and protected. Application Filesystem can be protected through “__LOCKED__” file insert. |
| user_fs | 0 | User Flash. 0→ FS not mounted; 1→ mounted |
| extvol_fs | 0 | Ext. Storage. 0 → FS not mounted; 1 → mounted |
| nv_power_cycles | 5 | Number of executed power-on |

| Status variable | Value | Note |
|---|---|---|
| nv_wdog_resets | 0 | Number of reset due to watchdog |
| nv_reset_cycles | 43 | Number of reset cycles |
| wifi_state | 10 | 0= Hardware power up<br>1=Hardware failure<br>2=Radio task terminated by user<br>3=Radio idle<br>4=Scan in progress<br>5=Scan complete<br>6=Join in progress<br>7=Joined<br>8=Access point started<br>9=802.11 handshake complete<br>10=Ready to transmit data (i.e. “Link Up”) |
| wifi_own_macaddr | 00:80:E1:FA:12:34 | Mac address |
| wifi_bssid | 00:18:F8:3C:D9:18 | BSSID of current association |
| wifi_aid | 0 | Association ID of current association |
| wifi_channelnum | 11 | Current radio channel number |
| wifi_sup_rate_mask | 0x003FFFCF | Radio: supported data rate mask |
| wifi_bas_rate_mask | 0x0000000F | AP reported: basic data rate mask |
| wifi_chan_activity | 0x00003FFF | Channels where we are allowed to transmit.<br>Channel mask. i.e. 0x00003FFF → from channel 0 to channel 13 |
| wifi_max_tx_power | 18 | max allowed transmit power for the defined regdomain |
| wifi_gf_mode | 0 | Greenfield (high throughput) mode |
| wifi_reg_country | IT | Current regulatory domain |
| wifi_dtim_period | 1 | AP reported DTIM period (used in STA mode) |
| wifi_num_assoc | 1 | Number of the client associated to the module |
| ip_from_AutoIP | 0 | Indicates if the IP derives from AutoIP |
| ip_ipaddr | 192.168.121.184 | Current IP address |
| ip_netmask | 255.255.252.0 | Current IP netmask |
| ip_gw | 192.168.123.20 | Current IP default gateway |
| ip_dns1 | 192.168.123.20 | Current IP Primary DNS server |
| ip_dns2 | 192.168.123.20 | Current IP Secondary DNS server |
| ip_linklocal | fe80:0:0:0:280:e1ff:fefa:1234 | Current IPv6 linklocal |
| ip_local | fec0:0:0:0:280:e1ff:fefa:1234 | Current IPv6 local |
| ip_dns1v6 | 0:0:0:0:0:0:0:0 | Current IPv6 Primary DNS Server |
| ip_dns2v6 | 0:0:0:0:0:0:0:0 | Current IPv6 Secondary DNS server |

The following table lists the peer variables together with their associated values. The list of peer variables is generated by the command AT+S.PEERS.

**Table 9. Peer variables**

| Peers variables | Value | Note |
|---|---|---|
| ***AT-S.Var:n.<var>*** *where n:0=mode connected to STA; n between 1 to 5=id of the connected client* | | |
| link_id | 0 | Identifier of the client |
| state | 4 | 0 = Hardware Power Up<br>1 = HW link initialization<br>2 = Client Link identifier allocated<br>3 = Authenticated<br>4 = Associated<br>5 = Peer lost beacons<br>6 = Peer in power save state |
| addr | 02:62:1F:51:8F:0B | MAC address of the client |
| last_rx | 14 | Timestamp of last received packet |
| last_tx | 14 | Timestamp of last transmitted packet |
| rx_drops | 0 | Count of frames dropped during reception |
| tx_drops | 0 | Count of frames dropped during transmission |
| rx_pkts | 22 | Count of received frames |
| tx_pkts | 24 | Count of transmitted frames |
| tx_errs | 0 | Count of errors detected during frame transmit |
| rate_mask | 0x00003FCF | AP reported Operational data rate mask |
| cur_rate_idx | 3 | Most significant byte of the rate_mask |
| cur_rate_ok | 3 | Counter to perform rate step up |
| cur_rate_fail | 0 | Counter to perform rate step down |
| tx_consec_fail | 0 | Counter to perform disassociation |
| rx_seqnum | 0x000031F0 | Sequence number of last RX directed frame |
| rx_seqnum_mc | 0x00002780 | Sequence number of last RX multicast frame |
| rx_rssi | -51 | Signal strength of last received packet |
| rx_rateidx | 13 | Rate index of last received packet |
| setprot | 3 | Bitmask to indicate protection for TX (bit 1) and/or RX (bit 0) IEEE 802.11 frames |
| listen_interval | 0 | AP reported listen interval |
| capinfo | 0x00000411 | Information about the AP capabilities |

# 9 Synchronous errors and WIND messages

### Synchronous errors

The following tables list the synchronous errors, each specified with its ID. The tables also show the errors that are specific to an HTTP client connection. They are characterized by the prefix AT-S.

**Table 10. AT-S.ERROR:=ERROR ID= =ERROR String=**

| ERROR ID | ERROR string |
|---|---|
| 1 | "Command not found" |
| 2 | "Missing argument(s)" |
| 3 | Reserved |
| 4 | "Variable not found" |
| 5 | Reserved |
| 6 | "Too many argument(s)" |
| 7 | "Invalid argument(s)" |
| 8 | "Invalid value" |
| 9 | Reserved |
| 10 | Reserved |
| 11 | Reserved |
| 12 | Reserved |
| 13 | Reserved |
| 14 | Reserved |
| 15 | Reserved |
| 16 | "Cannot switch to Python shell" |
| 17 | "Radio enabled" |
| 18 | "Radio not running" |
| 19 | "Direction must be 'in' or 'out'" |
| 20 | "Invalid GPIO Num (0-15)" |
| 21 | "Cannot use GPIO6 when sleep_enabled" |
| 22 | "Cannot use interrupt. Reserved for WPS feature" |
| 23 | "Cannot use GPIO10 when blink_led" |
| 24 | "Cannot use when SPI" |
| 25 | "Interrupt line already set" |
| 26 | "Output voltage not allowed" |
| 27 | "Frequency not supported" |
| 28 | "Duty cycle not supported" |
| 29 | "PWM not running" |
| 30 | Reserved |
| 31 | Reserved |
| 32 | "Argument exceeds allowed size" |
| 33 | "Argument evaluated to zero" |

| ERROR ID | ERROR string |
|---|---|
| 34 | Reserved |
| 35 | Reserved |
| 36 | Reserved |
| 37 | Reserved |
| 38 | Reserved |
| 39 | "PIN needs to be 7 digits" |
| 40 | "Scan in Progress" |
| 41 | "Scan Failed" |
| 42 | "Wait for Hardware Busy" |
| 43 | "Wait for Hardware Starting" |
| 44 | "Wait for Connection Up" |
| 45 | "Unable to complete PWM setting" |
| 46 | "ADC conversion failed" |
| 47 | Reserved |
| 48 | Reserved |
| 49 | Reserved |
| 50 | "Unable to access filesystem" |
| 51 | "Unable to open file" |
| 52 | "Unable to seek file" |
| 53 | "Unable to read file" |
| 54 | "Unable to close file" |
| 55 | "Unable to rename file" |
| 56 | "Unable to delete file" |
| 57 | "Unable to open directory" |
| 58 | "Unable to read directory" |
| 59 | "Unable to (un)mount volume" |
| 60 | "IP not ready to send" |
| 61 | "Cannot renew IP address" |
| 62 | "Failed to renew IP address" |
| 63 | "DNS busy" |
| 64 | "DNS start failed" |
| 65 | "DNS address failure" |
| 66 | "Cannot update Date Time" |
| 67 | "WPS only allowed on STA mode" |
| 68 | Reserved |
| 69 | Reserved |
| 70 | Reserved |
| 71 | Closing socket |
| 72 | "Closed socket" |
| 73 | "Port already opened" |

| ERROR ID | ERROR string |
|---|---|
| 74 | "Failed to open socket" |
| 75 | "Too many sockets" |
| 76 | "Illegal Socket ID" |
| 77 | "Pending data" |
| 78 | "Socket not connected" |
| 79 | "Write failed" |
| 80 | "No valid HTTP Client Instance ID" |
| 81 | "HTTP Client busy" |
| 82 | "Failed to release HTTP Client Instance" |
| 83 | Reserved |
| 84 | Reserved |
| 85 | Reserved |
| 86 | Reserved |
| 87 | Reserved |
| 88 | Reserved |
| 89 | Reserved |
| 90 | Reserved |
| 91 | "Low Memory" |
| 92 | Reserved |
| 93 | Reserved |
| 94 | Reserved |
| 95 | Reserved |
| 96 | Reserved |
| 97 | Reserved |
| 98 | Reserved |
| 99 | "Scan Aborted" |
| 100 | "Failed to read flash" |
| 101 | "Failed to write flash" |
| 102 | Reserved |
| 103 | Reserved |
| 104 | Reserved |
| 105 | Reserved |
| 106 | Reserved |
| 107 | Reserved |
| 108 | Reserved |
| 109 | Reserved |
| 110 | Reserved |
| 111 | "Request failed" |
| 112 | Reserved |
| 113 | Reserved |

| ERROR ID | ERROR string |
|---|---|
| 114 | Reserved |
| 115 | Reserved |
| 116 | Reserved |
| 117 | Reserved |
| 118 | Reserved |
| 119 | Reserved |

**Table 11. AT-S.WebSocket Client Error:=Error ID=**

| ERROR ID | Description |
|---|---|
| 0 | Connection Refused |
| 1 | TCP error (TimeoutConn \| TimeoutData \| Disconnected \| Send \| NoTcpConnection) |
| 2 | TLS Connection Error |
| 3 | Server Resolve Error |
| 4 | HTTP Error (Parser \| Generator \| ProtocolSwitch) |
| 5 | HTTP Timeout |

**Table 12. AT-S.Http Client Error:=Error ID=**

| ERROR ID | Description |
|---|---|
| 0 | Error while resolving Hostname |
| 1 | Timeout while TCP-Connect (TCP-Syn) |
| 2 | Error or timeout while TCP-Connect |
| 3 | HTTP Client refused connecting to server |
| 4 | Timeout while receiving TCP data |
| 5 | Received unexpected TCP-Disconnect while sending |
| 6 | Error while sending TCP data |
| 7 | Error during TLS connect |
| 8 | Error while generating HTTP data (HTTP-Generator) |
| 9 | Error while parsing HTTP data (HTTP-Parser) |
| 10 | Timeout for whole HTTP-Request +HTTP-Response |
| 11 | Error while Protocol switch handshake |
| 12 | Error while File access Open |
| 13 | Error while File access Close |
| 14 | Error while File access Read |
| 15 | Error while File access Write |

**Table 13. AT-S.Certificate Error:=Error ID=**

| ERROR ID | Description |
|---|---|
| 0 | Reserved |

| ERROR ID | Description |
|---|---|
| 1 | Reserved |
| 2 | Subject certificate is null or length is zero |
| 3 | Parsing the certificate failed |
| 4 | Common name does not match (wildcard certificate) |
| 5 | Common name does not match |
| 6 | Desired key usage is not permitted |
| 7 | Parsing the AuthorityKeyId extension failed |
| 8 | Initializing decryption of signature failed |
| 9 | Decryption of signature failed |
| 10 | Invalid signature |
| 11 | Parsing the signature failed |
| 12 | Parsing the authority certificate failed |
| 13 | Missing subject key id extension |
| 14 | Parsing the SubjectKeyId extension failed |
| 15 | SubjectKeyId does not match AuthorityKeyId |
| 16 | Desired key usage is not permitted |
| 17 | Basic constraints are not good |
| 18 | Parsing the AuthorityKeyId extension failed |
| 19 | The certificate is not supported |
| 20 | The certificate has been revoked |
| 21 | The certificate has expired |
| 22 | An unspecified error has occurred |
| 23 | The CA could not be found |
| 24 | Reserved |
| 25 | Reserved |

### WIND messages

The following table lists the asynchronous messages. The message format is: +WIND:<WIND ID>:<WIND Message>[:<Variables>]

**Table 14. WIND messages**

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 0 | “Console active” | | | After a reset, wait for this wind message before sending commands |
| 1 | “Poweron” | %s-%s-%s | DATE,BUILD,MODEL | |
| 2 | “Reset” | | | |
| 3 | “Watchdog Running” | %u | Configured watchdog timeout | |
| 4 | “Low Memory” | | | |

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 5 | “WiFi Hardware Failure” | %u | DEAD Reason. Range of values:<br>1. Shown when a WiFi exception “WIND:31” happens<br>2. Shown when a WPA Init “WIND:50” happens<br>3. Shown when startup “WIND:32” cannot be reached<br>4. Shown when driver is not able to write to cw1100<br>5. Shown when cw1100 does not send an ack within given timeout | |
| 6 | Reserved | | | |
| 7 | “Configuration Failure” | %u | Failure reason. Range of values:<br>1. Reserved<br>2. Reserved<br>3. Reserved<br>4. Shown when RTC cannot be initialized<br>5. Reserved<br>6. Shown when System Tick cannot be initialized<br>7. Shown when Flash cannot be written<br>8. Shown when uPython detects a fatal error<br>9. Shown when UART/SPI queues cannot be created<br>10. (Does not Halt) Shown when HW pull-up resistor has no match with enabled interface<br>11. (Does not Halt) Shown when Ext SPI Flash cannot be formatted<br>12. (Does not Halt) Shown when Ext SPI Flash cannot be mounted<br>13. Shown when APP disk cannot be mounted<br>14. Shown when RAM disk cannot be created | |
| 8 | “Hard Fault” | %s:(%08x)*8 | Task, name, STM32 registers content | |
| 9 | “StackOverflow” | %s | Task name | |
| 10 | “Malloc Failed” | %s:%d:%d | Task name, Required Size, Available Size | |
| 11 | “Radio Startup Failure” | %u:%08x | Failure Reason, Found register content. Range of Values<br>1. Shown when cw1100 Prefetch bit cannot be set<br>2. Shown when cw1100 Firmware cannot be uploaded<br>3. Shown when cw1100 is not evaluated as sane<br>4. Shown when cw1100 DPLL register cannot be set<br>5. Shown when cw1100 cannot be woken up<br>6. Shown when cw1100 ID low register is evaluated as bad<br>7. Shown when cw1100 ID medium register is evaluated as bad<br>8. Shown when cw1100 ID high register is evaluated as bad<br>9. Shown when cw1100 cannot be set in Direct Mode<br>10. Shown when cw1100 cannot be set in ETF Mode | |
| 12 | “WiFi PS Mode Failure | %u:%d | Failure Reason, Found status. Range of Values<br>1. Shown when cw1100 is not allowing PM set<br>2. Shown when AP does not provide an answer within given timeout<br>3. Shown when cw1100 raises this event | |
| 13 | “Copyright (c) 2012-2017 STMicroelectronics, Inc. All rights Reserved” | %s | nv_model configuration variable | |

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 14 | “WiFi BSS Regained” | %d | Threshold | |
| 15 | “WiFi Signal Low” | %d | RSSI | |
| 16 | “WiFi Signal Ok” | %d | RSSI | |
| 17 | Boot Messages | Boot %u.%u | Welcome message from FW update | UART baudrate is hardcoded to 115200.<br>Message string (%s) cannot be removed through console_winds configuration variable. |
| | | Aborting F/W update:%u | Shown when FW cannot be updated (GPIO0 / CRC). Cleanup. | |
| | | Performing F/W update | Shown when FW was validated (before write) | |
| | | Completed F/W update | Shown when FW was updated. Cleanup | |
| | | Cleanup | Shown when mass storage is going to be erased | |
| 18 | “Keytype not implemented” | %u | Required Key Type | |
| 19 | “WiFi Join” | %m | AP MAC Address | |
| 20 | “WiFi Join Failed” | %04x | Failure Reason. Range of Values<br>0001 – Shown when cw1100 cannot be configured<br>0002 – Shown when cw1100 does not send an ack within a given timeout<br>0003 – shown when cw1100 cannot be configured after join operation<br>Shown when cw1100 reply status is evaluated as bad | |
| 21 | “WiFi Scanning” | | | |
| 22 | “Scan Blew Up” | | | |
| 23 | “Scan Failed” | %04x | Cw1100 reply status | |
| 24 | “WiFi Up” | %u:%i | IPFlag (0 → IPv4; 1 → temporary IPv4 coming from AutoIP), IPv4/6 address | It may be printed up twice with both the IP addresses. The WIND concludes the association process with the remote AP |
| 25 | ”WiFi Association successful” | %s | SSID | |
| 26 | “Started AP” | %s | SSID | |
| 27 | “AP Start Failed” | %04x | Cw1100 reply status | |
| 28 | “Station Associated” | %m:%u | Peer MAC Address, Re-association flag | |
| 29 | “DHCP Reply” | %i:%m | Peer IPv4 address, Peer MAC Address | |
| 30 | “WiFi BSS Lost” | | | |
| 31 | “WiFi Exception” | %u:%s:%08x:%08x | Reason, File Name, cw1100 registers content | |
| 32 | “WiFi Hardware Started” | | | Wait for this WIND before switching on the radio |
| 33 | “WiFi Network Lost” | | | |
| 34 | “WiFi Unhandled Event” | %02x | Detected Event | |
| 35 | “WiFi Scan Complete” | %02x | Aborting Reason | |
| 36 | “WiFi Unhandled Indication” | %02x | Detected Indication | |
| 37 | Reserved | | | |

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 38 | “WiFi Powered Down” | | | |
| 39 | “HW in MiniAP Mode” | | | |
| 40 | “WiFi Deauthentication” | %u | Refer to standard | |
| 41 | “WiFi Disassociation” | %u | Refer to standard | |
| 42 | “WiFi Unhandled Management” | %02x | Received Frame | |
| 43 | “WiFi Unhandled Data” | %02x | Received Frame | |
| 44 | “WiFi Unknown Frame” | %04x | Received Frame | |
| 45 | “Dot11 AuthIllegal” | %u:%u | Authentication Algorithm, authentication sequence | |
| 46 | “WPA Crunching PSK” | %s:%u | PSK,length | |
| 47 | Reserved | | | |
| 48 | Reserved | | | |
| 49 | "WPA Terminated" | %d | Exit Status | |
| 50 | "WPA Start Failed" | %u | Failure Reason. Range Values.<br>1. Shown when WPA Supplicant was not able to start<br>2. Shown when WPA Supplicant was not able to create peer table | |
| 51 | "WPA Handshake Complete" | | | |
| 52 | "GPIO Interrupt" | %u:%u | GPIO number, level | |
| 53 | "Wakeup" | | | |
| 54 | Reserved | | | |
| 55 | "Pending Data" | %u:%u:%u:%u | Server ID, Client ID, Received bytes, cumulative bytes | |
| 56 | "Input to remote" | %u | Input ID | Only printed when buffer for input request is empty. Buffer is filled and cleared by AT +S.INPUTSSI command |
| 57 | "Output from remote" | %u:%s | Received message length, message | Maximum allowed length is 128. Every message longer than 128 will be truncated. Use multipart/form-data webserver capability to send more than 128 bytes. Refer to AN4965 "WebServer on SPWF04S module |
| 58 | "Socket Closed" | %u:%u | Client ID<br>Closure Reason. Values range<br>**0.**Shown when remote TCP sent a FIN<br>**1.**Shown when Remote TCP sent a RST, or did not rely to FINACK<br>**2.**Shown when too many retransmissions have occurred<br>**3.**Shown when network layer has broken<br>**4.**Shown when a timeout occurred<br>**5.**Shown when ICMP Destination Unreachable was received | AT+S.SOCKC/AT +S.SOCKDC needs to be called to flush the buffer and clear the socket state |

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 59 | Reserved | | | |
| 60 | Reserved | | | |
| 61 | "Incoming Socket Client" | %i:%u:%u:%u | Client IP, client Port, server ID, client ID | |
| 62 | "Socket Client Gone" | %i:%u:%u:%u:%u | Client IP, client Port, server ID, client ID, closure reason | |
| 63 | "Socket Dropping Data" | %u:%u:%u | Server ID, client ID, dropped bytes | |
| 64 | "Remote Configuration" | %s:%s | Configuration item, configuration value | Item called “Key” is only shown when different from user_desc password |
| 65 | "Factory Reset" | %u | Restore Reason. Values Range<br>**0.**GPIO0 high at start-up, or reset line not stable<br>**1.**CRC check failure on saved configuration | |
| 66 | "Low Power Mode" | %u | Powersave Value. Values Range<br>1. Standard Powersave<br>2. Fast Powersave | |
| 67 | "Going into Standby" | %u | Configured Standby Time | |
| 68 | "Resuming from Standby" | | | |
| 69 | "Going into DeepSleep" | | | |
| 70 | "Resuming from DeepSleep" | | | |
| 71 | Reserved | | | |
| 72 | "Station Disassociated" | %m:%u | Peer MAC Address, reason code | |
| 73 | "System Configuration Updated" | | | |
| 74 | "Rejected found Network" | | | |
| 75 | "Rejected Association" | %04x | Rejection Reason. Values Range<br>Shown when cw1100 cannot be configured<br>0xffff – shown when WPA Supplicant was not able allocate memory | |
| 76 | "WiFi Authentication Timed Out" | | | |
| 77 | "WiFi Association Timed Out" | | | |
| 78 | "MIC Failure" | | | |
| 79 | Reserved | | | |
| 80 | "UDP Broadcast Received" | %u:%i:%u:%u:%s | Server ID, Client IP, client port, datagram size, datagram content | |
| 81 | "WPS generating DH keyset" | | | |
| 82 | "WPS enrollment attempt timed out" | | | |
| 83 | "Sockd Dropping Client" | %u | Dropping reason<br>1. Not enough RAM to allow access to socket client<br>2. Too many clients are connected (only when UDP) | |
| 84 | "NTP Server Delivery" | %u.%u.%u:%u:%u.%u.%u | Year, month,day,day of week, hour, min,sec | |

| WIND ID | WIND message | Variables | Description | Application notes |
|---|---|---|---|---|
| 85 | "DHCP failed to get a lease" | %u | Failure reason. Range Values.<br>**0.** Shown when DHCP server does not respond<br>**1.** Shown when DHCP server answer has a wrong message format<br>**2.** Shown when given timeout value is not acceptable<br>**3.** Shown when received a NACK<br>**4.** Shown when DHCP server does not provide an IP address<br>**5.** Shown when DHCP server provide a lease time smaller than 50s<br>**6.** Shown when DHCP Lease Time expired | |
| 86 | "MQTT Published" | %u:%s:%u:%u:%u:%u:%u:%s | Client ID, topic,QoS,retain,dupl,msg siz,tot siz,msg | |
| 87 | "MQTT Closed" | %u:%u | Closure reason (0: connection level; 1: MQTT protocol level), Client ID | |
| 88 | "WebSocket Data" | %u:%u:%u:%u:%u | Client ID, lastframe flag, lastfrag flag rcv bytes, cumulated bytes | |
| 89 | "WebSocket Closed" | %u | Client ID | AT+S.WSOCKC needs to be called to flush the buffer and clear the websocket state |
| 90 | "File Received" | %i:%s | Client IP (only provided if file comes from TFTP client), Filename | |

# Revision history

Table 15. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 03-Mar-2017 | 1 | Initial release. |
| 14-Nov-2017 | 2 | Added un-maskable results for AT-Commands.<br>Added FW1.1.0 new features.<br>Added part number in Table 6. Technology partners. |
| 27-Nov-2018 | 3 | Update Section 1 SPWF04Sx software architecture description. Added Section 6.12 AT+S.FSWRITE and Section 6.28 AT+S.FSM. Minor text changes. |

# Contents

1 SPWF04Sx software architecture description . . . . . 2
2 Interface and message types . . . . . 4
3 AT commands over the UART . . . . . 7
4 SPI protocol . . . . . 8
5 MicroPython Scripting . . . . . 10
6 Command reference guide . . . . . 11
6.1 AT . . . . . 11
6.2 AT+S.HELP . . . . . 11
6.3 AT+S.STS . . . . . 11
6.4 AT+S.PEERS . . . . . 12
6.5 AT+S.RESET . . . . . 13
6.6 AT+S.PMS . . . . . 13
6.7 AT+S.PYTHON . . . . . 14
6.8 AT+S.GCFG . . . . . 14
6.9 AT+S.SCFG . . . . . 14
6.10 AT+S.WCFG . . . . . 15
6.11 AT+S.FCFG . . . . . 15
6.12 AT+S.FSWRITE . . . . . 15
6.13 AT+S.FSUPDATE . . . . . 16
6.14 AT+S.FWUPDATE . . . . . 16
6.15 AT+S.GPIOC . . . . . 17
6.16 AT+S.GPIOR . . . . . 17
6.17 AT+S.GPIOW . . . . . 18
6.18 AT+S.DAC . . . . . 18
6.19 AT+S.ADC . . . . . 18
6.20 AT+S.PWM . . . . . 19
6.21 AT+S.TIME . . . . . 19
6.22 AT+S.RANDOM . . . . . 20
6.23 AT+S.FSC . . . . . 20

6.24 AT+S.FSD . . . . . 20
6.25 AT+S.FSR . . . . . 21
6.26 AT+S.FSL . . . . . 21
6.27 AT+S.FSP . . . . . 22
6.28 AT+S.FSM . . . . . 22
6.29 AT+S.FSU . . . . . 22
6.30 AT+S.HASH . . . . . 23
6.31 AT+S.WPAECERT . . . . . 23
6.32 AT+S.TLSCERT . . . . . 24
6.33 AT+S.WPS . . . . . 24
6.34 AT+S.WIFI . . . . . 25
6.35 AT+S.SCAN . . . . . 25
6.36 AT+S.SSIDTXT . . . . . 26
6.37 AT+S.PING . . . . . 26
6.38 AT+S.SOCKON . . . . . 26
6.39 AT+S.SOCKQ . . . . . 27
6.40 AT+S.SOCKC . . . . . 28
6.41 AT+S.SOCKW . . . . . 28
6.42 AT+S.SOCKR . . . . . 28
6.43 AT+S.SOCKL . . . . . 29
6.44 AT+S.SOCKDON . . . . . 29
6.45 AT+S.SOCKDQ . . . . . 30
6.46 AT+S.SOCKDC . . . . . 30
6.47 AT+S.SOCKDW . . . . . 31
6.48 AT+S.SOCKDR . . . . . 31
6.49 AT+S.SOCKDL . . . . . 31
6.50 AT+S.TFTPGET . . . . . 32
6.51 AT+S.TFTPPUT . . . . . 32
6.52 AT+S.SMTP . . . . . 33
6.53 AT+S.HTTPGET . . . . . 33

6.54 AT+S.HTTPPOST . . . . . 34
6.55 AT+S.INPUTSSI . . . . . 35
6.56 AT+S.MQTTCONN . . . . . 35
6.57 AT+S.MQTTPUB . . . . . 36
6.58 AT+S.MQTTDISC . . . . . 36
6.59 AT+S.MQTTSUB . . . . . 37
6.60 AT+S.MQTTUNSUB . . . . . 37
6.61 AT+S.WSOCKON . . . . . 37
6.62 AT+S.WSOCKQ . . . . . 38
6.63 AT+S.WSOCKC . . . . . 38
6.64 AT+S.WSOCKW . . . . . 39
6.65 AT+S.WSOCKR . . . . . 39
6.66 AT+S.WSOCKL . . . . . 39
**7 Technology partners . . . . . 41**
**8 Variables and values . . . . . 42**
**9 Synchronous errors and WIND messages . . . . . 50**
**Variables and values . . . . . 0**
**Synchronous errors and WIND messages . . . . . 0**
**Revision history . . . . . 60**

# List of tables

Table 1. SPWF04Sx commands . . . 4
Table 2. KindOfEvent Byte SubField . . . 8
Table 3. Indication number field . . . 8
Table 4. Power states and configuration . . . 13
Table 5. Association between file extension and HTTP content-type . . . 34
Table 6. Technology partners . . . 41
Table 7. Configuration variables . . . 42
Table 8. Status variables . . . 47
Table 9. Peer variables . . . 49
Table 10. AT-S.ERROR:=ERROR ID= =ERROR String= . . . 50
Table 11. AT-S.WebSocket Client Error:=Error ID= . . . 53
Table 12. AT-S.Http Client Error:=Error ID= . . . 53
Table 13. AT-S.Certificate Error:=Error ID= . . . 53
Table 14. WIND messages . . . 54
Table 7. Configuration variables . . . 0
Table 8. Status variables . . . 0
Table 9. Peer variables . . . 0
Table 10. AT-S.ERROR:=ERROR ID= =ERROR String= . . . 0
Table 11. AT-S.WebSocket Client Error:=Error ID= . . . 0
Table 12. AT-S.Http Client Error:=Error ID= . . . 0
Table 13. AT-S.Certificate Error:=Error ID= . . . 0
Table 14. WIND messages . . . 0
Table 15. Document revision history . . . 60

# List of figures

Figure 1. SPWF04Sx user integration modes . . . 1
Figure 2. SPWF04Sx software architecture . . . 2
Figure 3. SPWF04Sx protocol stack diagram . . . 3
Figure 4. SPI packet formats . . . 8
Figure 5. MicroPython modes . . . 10
Figure 6. AP capabilities list . . . 25

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved