Command set reference guide for "AT full stack" for SPWF01Sx series of Wi-Fi modules

## Introduction

This document is a guide to the "AT full stack" command set, a firmware application embedded in the SPWF01Sx series of Wi-Fi modules.

The "AT full stack" command set is a user-friendly interface of a complete TCP/IP stack supporting both direct links with Wi-Fi enabled devices and infrastructure communication modes with an access point. Application utilities such as an http client and a web server are also featured in the software to allow easy integration with many Internet-based applications.

This document provides a detailed description of each command supported by the "AT full stack" interface. A description and explanation of configuration variables, status variables and asynchronous indication messages are also integrated into the document, respectively, in *Section 3*, *Section 4* and *Section 5*.

# Contents

# 1 Overview

The "AT full stack" command interface described in this user guide consists of a set of:

- AT-style commands
- configuration variables
- status variables and
- asynchronous indications (also known as unsolicited responses or WINDs)

The communication of commands, variables, and asynchronous indications is executed via the serial port and implies the integration of the SPWF01Sx modules with a host processor as indicated in *Figure 1*.

**Figure 1. SPWF01S integration with an host processor**
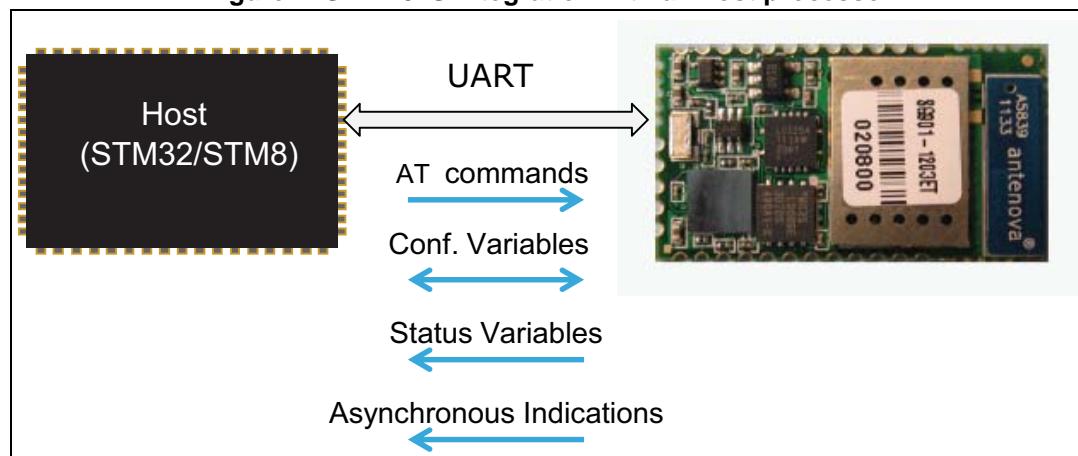


Asynchronous indications may arrive at any time (except as noted below), and have the format:

```
<cr><lf>+WIND:<number>:<descriptive string><cr><lf>
```

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

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

AT commands are always in the form of:

```
AT<cmd><cr>
<zero or more response lines>
<cr><lf><responsecode><cr><lf>
```

The AT command line, up to the terminating <cr>, is sent from the host. The (optional) response lines followed by <cr><lf><responsecode><cr><lf> are sent from the module.

The <response code> is one of the following:

```
OK
ERROR: <descriptive text>
```

The AT command handler allows AT commands to be entered in upper or lower case.

Command arguments are case sensitive.

The maximum command length is 512 characters.

Note that asynchronous indications are blocked from the time the initial "A" is sent to the module until the <responsecode> line is sent. Any indications generated from events during the processing of an AT command are queued for delivery after the command is complete.

# 2 Command reference

This chapter details each of the AT commands including brief descriptions of the behavior, syntax of the command, example of use and types of responses.

The commands described are those listed in *Table 1*.

Table key:

- S: command is supported in this release
- NS: command is not supported in this release
- New: command is introduced with this release
- Deprecated: command is deprecated in this release

**Table 1. AT command list summary**

| "AT full stack" commands | Rel. 1.0 | Rel. 2.0 | Rel. 3.3 | Rel. 3.4 | >=Rel. 3.5 |
|---|---|---|---|---|---|
| AT | S | S | S | S | S |
| AT+CFUN | S | S | S[1] | S | S |
| AT+S.HELP | S | S | S | S | S |
| AT+S.GCFG | S | S | S | S | S |
| AT+S.SCFG | S | S | S | S | S |
| AT+S.SSIDTXT | S | S | S | S | S |
| AT&V | S | S | S | S | S |
| AT&F | S | S | S | S | S |
| AT&W | S | S | S | S | S |
| AT+S.NVW | S | S | Deprecated | NS | NS |
| AT+S.STS | S | S | S | S | S |
| AT+S.PEERS | - | New | S | S | S |
| AT+S.PING | S | S | S | S | S |
| AT+S.SOCKON | S | S | S | S | S |
| AT+S.SOCKOS | S | S | Deprecated[2] | NS | NS |
| AT+S.SOCKW | S | S | S | S | S |
| AT+S.SOCKQ | S | S | S | S | S |
| AT+S.SOCKR | S | S | S | S | S |
| AT+S.SOCKC | S | S | S | S | S |
| AT+S.SOCKD | - | - | New | S | S |
| AT+S. | - | - | New | S | S |
| AT+S.HTTPGET | S | S | S | S | S |
| AT+S.HTTPPOST | - | New | S | S | S |
| AT+S.FSC | S | S | S | S | S |

**Table 1. AT command list summary (continued)**

| "AT full stack" commands | Rel. 1.0 | Rel. 2.0 | Rel. 3.3 | Rel. 3.4 | >=Rel. 3.5 |
|---|---|---|---|---|---|
| AT+S.FSA | S | S | S | S | S |
| AT+S.FSD | S | S | S | S | S |
| AT+S.FSL | S | S | S | S | S |
| AT+S.FSP | S | S | S | S | S |
| AT+S.MFGTEST | S | S | Deprecated | NS | NS |
| AT+S.PEMDATA | S | S | Deprecated | NS | NS |
| AT+S.WIFI | S | S | S | S | S |
| AT+S.ROAM | S | S | S | S | S |
| AT+S.GPIOC | S | S | S | S | S |
| AT+S.GPIOR | S | S | S | S | S |
| AT+S.GPIOW | S | S | S | S | S |
| AT+S.FWUPDATE [3] | S | S | S | S | S |
| AT+S.HTTPDFSUPDATE [3] | S | S | S | S | S |
| AT+S.HTTPDFSERASE[3] | - | - | New | S | S |
| AT+S.HTTPD | - | - | New | S | S |
| AT+S.SCAN | - | New | S | S | S |
| AT+S.ADC | - | - | - | New | S |
| AT+S.DAC | - | - | - | New | S |
| AT+S.PWM | - | - | - | New | S |
| AT+S.TLSCERT | - | - | - | New | S |
| AT+S.TLSCERT2 | - | - | - | New | S |
| AT+S.TLSDOMAIN | - | - | - | New | S |
| AT+S.SETTIME | - | - | - | New | S |
| AT+S.RMPEER | - | - | - | - | New |
| AT+S.HTTPREQ | - | - | - | - | New |
| AT+S.FSR | - | - | - | - | New |
| AT+S.HTTPDFSWRITE | - | - | - | - | New |

1. Extended to manage low power configuration modes

2. The use of UART 2 and UART 3 interfaces is disabled

3. Command only supported on the HW versions SPWF01Sx.11

The following subsections cover each AT command in detail.

## 2.1 AT: Attention

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

Arguments:

`none`

Example:

```
AT<cr>
<cr><lf>OK<cr><lf>
```

## 2.2 AT+CFUN: comm function

AT+CFUN sets a power mode with default values as it is indicated in the table 2 and includes a reset of the device.

Arguments:

<num> 0 = switch to the active state and reset the device

1 = keep the current state and reset the device

2 = switch to the powersave state and reset the device

3 = switch to the sleep state and reset the device

4 = switch to the standby mode and reset the device

**Table 2. Power states default configuration**

| Module states | Shortcut command | STM32 states | WLAN states | AT variables default values |
|---|---|---|---|---|
| Active | CFUN 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 |
| Power save | CFUN 2 | Run | PS or Fast PS | at+s.scfg=sleep_enabled,0<br>at+s.scfg=wifi_powersave,1<br>at+s.scfg=wifi_operational_mode,11<br>at+s.scfg=wifi_beacon_wakeup,1<br>at+s.scfg=wifi_listen_interval,0 |
| Sleep | CFUN 3 | Stop | PS or Fast PS | at+s.scfg=sleep_enabled,1<br>at+s.scfg=wifi_powersave,1<br>at+s.scfg=wifi_operational_mode,11<br>at+s.scfg=wifi_beacon_wakeup,1<br>at+s.scfg=wifi_listen_interval,0 |
| Standby | CFUN 4 | Standby | Off | at+s.scfg=standby_enabled,1<br>at+s.scfg=standby_time,10<br>at+s.scfg=sleep_enabled,0 |

Example:

```
AT+CFUN=1<cr>
```

```
<cr><lf>+WIND:2:Reset<cr><lf>
```

## 2.3 AT+S.HELP: display help text

AT+S.HELP prints a list of all commands supported with a brief help text for each command.

Arguments:

```
none
```

Example:

```
AT+S.HELP<cr>
# Recognized commands

#  AT -- Null cmd, always returns OK
#  AT+CFUN =<0|1|2|3|4> -- Enable common functionalities
#  AT+S. -- Switch to data mode
#  AT+S.HELP -- This text
#  AT&F -- Restore factory default settings
#  AT&V -- Dump all settings
#  AT&W -- Save current settings to flash
#  AT+S.GCFG =<key> -- Get config key
#  AT+S.SCFG =<key>,<value> -- Set config key
#  AT+S.STS [=<sts_var>] -- Report current status/statistics
#  AT+S.SETTIME =<time_in_sec> -- Set current time
#  AT+S.FSC =<fname>,<max_len>[,<http_header>] -- Create a file for httpd
use
#  AT+S.FSA =<fname>,<datalen><CR><data> -- Append to an existing file
#  AT+S.FSD =<fname> -- Delete an existing file
#  AT+S.FSR =<old_fname>,<new_fname> -- Rename an existing file
#  AT+S.FSL -- List existing filename(s)
#  AT+S.FSP =<fname>[,<offset>,<length>] -- Print the contents of an
existing file
#  AT+S.GPIOC =<num>,<in|out>[,<0|R|F|B>] -- Configure specified GPIO
[Optional IRQ]
#  AT+S.GPIOR =<num> -- Read specified GPIO
#  AT+S.GPIOW =<num>,<val> -- Write specified GPIO
#  AT+S.DAC =<0|value> -- Disable/Enable DAC on GPIO15
#  AT+S.ADC [=raw] -- Read [raw] ADC value on GPIO8
#  AT+S.PWM =frequency[,duty_cycle] -- Set PWM on GPIO1
#  AT+S.WIFI =<0|1> -- Disable/Enable WiFi
#  AT+S.ROAM -- Trigger a WiFi Roam
#  AT+S.RMPEER =peer_number -- Disconnect peer from miniAP
#  AT+S.SCAN [=<a|p>[,<r|s|m>[,<fname>]]] -- Perform a [active/passive]
network scan, [filter off/SSID/MAC], [print to file]
#  AT+S.SSIDTXT [=<ssidtxt>] -- Set a textual SSID (not hex), otherwise
prints current SSID
#  AT+S.PEERS [=peer_number[,peer_var]] -- Dump contents of the peer table
```

```
#  AT+S.TLSCERT =<f_ca|f_cert|f_key>,<length> -- Configure SSL/TLS
certificates
#  AT+S.TLSCERT2 =clean,<f_ca|f_cert|f_key|f_domain|all> -- Cleanup SSL/TLS
certificates resources
#  AT+S.TLSDOMAIN =<f_domain>,<Server domain name> -- Set Server domain
name. It must match the secured site name EXACTLY.
#  AT+S.SOCKD =<0|port>[,<t|u>] -- Disable/Enable socket server. Default is
TCP
#  AT+S.SOCKON =<hostname>,<port>,<t|u|s>[,ind] -- Open a network socket
#  AT+S.SOCKQ =<id> -- Query socket for pending data
#  AT+S.SOCKC =<id> -- Close socket
#  AT+S.SOCKW =<id>,<len> -- Write data to socket
#  AT+S.SOCKR =<id>,<len> -- Read data from socket
#  AT+S.HTTPD =<0|1> -- Disable/Enable web server
#  AT+S.HTTPGET =<hostname>,<path&queryopts>[,port] -- Http GET of the
given path to the specified host/port
#  AT+S.HTTPPOST =<hostname>,<path&queryopts>,<formcontent>[,port] -- Http
POST of the given path to the specified host/port
#  AT+S.HTTPREQ =<hostname>,[,port],<length><CR><data> -- Custom Http
Request to the specified host/port
#  AT+S.HTTPDFSERASE -- Erase the external httpd filesystem
#  AT+S.HTTPDFSWRITE =<datalen><CR><data> -- Write the external httpd
filesystem
#  AT+S.HTTPDFSUPDATE =<hostname>,<path&queryopts>[,port] -- Download a new
httpd filesystem from the specified host/port
#  AT+S.FWUPDATE =<hostname>,<path&queryopts>[,port] -- Upgrade the onboard
firmware from the specified host/port
#  AT+S.PING =<hostname> -- Send a ping to a specified host


OK
```

## 2.4 AT+S.GCFG: get configuration value

AT+S.GCFG prints the value of one named configuration variable. See *Section 3: Configuration variable reference* for a list of available variables. AT&V provides the list of all variables and values on a running module.

Arguments:

```
<key>                  Name of the configuration variable
```

Example:
```
AT+S.GCFG=ip_ipaddr<cr>
# ip_ipaddr = 192.168.0.50<cr><lf>
<cr><lf>OK<cr><lf
```

## 2.5 AT+S.SCFG: set configuration value

AT+S.SCFG sets the value of one named configuration variable.See *Section 3: Configuration variable reference* for a list of available variables. AT&V provides the list of all variables and values on a running module.

Arguments:

| | |
|---|---|
| `<key>` | Name of the configuration variable |
| `<value>` | Value to store in the variable |

Example:

```
AT+S.SCFG=ip_ipaddr,192.168.10.10<cr>
# ip_ipaddr = 192.168.10.10<cr><lf>
<cr><lf>OK<cr><lf>
```

## 2.6 AT+S.SSIDTXT: get/set a textual SSID

AT+S.SSIDTXT is a convenience command that, when called with no arguments, returns the current value of wifi_ssid and wifi_ssid_len configuration variables as a text string. All non-printable characters are displayed with the '.' character. When called with an argument, the wifi_ssid and wifi_ssid_len configuration variables are set to the specified text string.

IEEE 802.11 and some access point implementations allow SSID values consisting of non-

printable octet values, therefore the wifi_ssid variable is set and stored as a sequence of octets (specified in hexadecimal) and wifi_ssid_len cannot be set based on any given flag value or delimiter. This command is only present as a convenience for those circumstances where the SSID is known to consist of printable characters.

Arguments:

| | |
|---|---|
| `<ssid>` | Textual SSID [1-32 characters] |

Example:

```
AT+S.SSIDTXT=AnSSID<cr>
<cr><lf>OK<cr><lf>
AT+S.GCFG=wifi_ssid<cr>
#  wifi_ssid = 41:6E:53:53:49:44:00:00:00:00:00:00:00:00:00:00:00:00:00:
00:00:00:00:00:00:00:00:00:00:00:00:00<cr><lf>
<cr><lf>OK<cr><lf>
AT+S.GCFG=wifi_ssid_len<cr>
#  wifi_ssid_len = 6<cr><lf>
<cr><lf>OK<cr><lf>
```

Example:

```
AT+S.SSIDTXT<cr>
SSID = 'AnSSID'<cr>
<cr><lf>OK<cr><lf>
```

## 2.7 AT&V: display all configuration values

AT&V prints the name and value of all configuration variables in the module.

Arguments:

```
none
```

Example:

```
AT&V<cr>
# Dumping All Configuration Keys:
#  nv_manuf = ST
#  nv_model = SPWF01SA1
#  nv_serial = 3513D42090
#  nv_wifi_macaddr = 00:80:E1:FF:C6:B3
#  etf_mode = 0
#  blink_led = 0
#  wind_off_low = 0x00000000
#  wind_off_medium = 0x00000000
#  wind_off_high = 0x00000000
#  user_desc = anonymous
#  escape_seq = at+s.
#  localecho1 = 0
#  console1_speed = 115200
#  console1_hwfc = 0
#  console1_enabled = 1
#  console1_delimiter = 0x0000002C
#  console1_errs = 1
#  sleep_enabled = 0
#  standby_enabled = 0
#  standby_time = 10
#  wifi_tx_msdu_lifetime = 0
#  wifi_rx_msdu_lifetime = 0
#  wifi_operational_mode = 0x00000011
#  wifi_beacon_wakeup = 1
#  wifi_beacon_interval = 100
#  wifi_listen_interval = 0
#  wifi_rts_threshold = 3000
#  wifi_ssid =
43:65:72:69:6F: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
#  wifi_ssid_len = 5
#  wifi_txfail_thresh = 5
#  wifi_ht_mode = 1
#  wifi_channelnum = 1
#  wifi_opr_rate_mask = 0x00003FCF
#  wifi_bas_rate_mask = 0x0000000F
#  wifi_mode = 1
```

```
#  wifi_region = 1
#  wifi_auth_type = 0
#  wifi_atim_window = 0
#  wifi_powersave = 1
#  wifi_tx_power = 18
#  wifi_rssi_thresh = 0
#  wifi_rssi_hyst = 0
#  wifi_ap_idle_timeout = 120
#  wifi_beacon_loss_thresh = 10
#  wifi_priv_mode = 2
#  wifi_wep_keys[0] = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
#  wifi_wep_keys[1] = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
#  wifi_wep_keys[2] = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
#  wifi_wep_keys[3] = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
#  wifi_wep_key_lens = 00:00:00:00
#  wifi_wep_default_key = 0
#  wifi_wpa_psk_raw =
FE:58:08:CA:71:63:99:1C:09:84:F1:ED:D1:79:25:6C:DC:6C:FC:C2:B8:48:DB:18:43
:A9:B7:73:FB:01:51:CE
#  wifi_wpa_psk_text = IoT1Time2Go
#  ip_use_dhcp = 1
#  ip_use_httpd = 0
#  ip_mtu = 1500
#  ip_hostname = iwm-FF-C6-B3
#  ip_apdomainname = captiveportal.net
#  ip_apredirect = firstset.html
#  ip_ipaddr = 192.168.0.50
#  ip_netmask = 255.255.255.0
#  ip_gw = 192.168.0.1
#  ip_dns = 192.168.0.1
#  ip_http_get_recv_timeout = 3000
#  ip_wait_timeout = 12000
#  ip_dhcp_timeout = 20
#  ip_sockd_timeout = 250
#  ip_dhcp_lease_time = 120
#  ip_dns_mode = 0
#  ip_use_cgis = 0x0000000F
#  ip_use_ssis = 0x0000000F
#  ip_use_decoder = 0x00000000


OK
```

## 2.8 AT&F: restore factory default settings

AT&F restores the factory default values of the configuration variables and writes them to non-volatile storage. Running this command is mandatory after every FW update.

*Note:* *To perform the HW factory reset of the variables, the pin GPIO0 must be high at the power-up (before the "+WIND:1:Poweron" indication). In order to use the HW factory reset (GPIO0 enabled) and FWUPDATE at the same time, see AT+S.FWUPDATE section.*

Arguments:

```
none
```

Example:

```
AT&F<cr>
<cr><lf>OK<cr><lf>
```

## 2.9 AT&W: save current settings

AT&W stores the current RAM-based settings to non-volatile storage.

Arguments:

```
none
```

Example:

```
AT&W<cr>
<cr><lf>OK<cr><lf>
```

## 2.10 AT+S.STS: report current status/statistics

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


Arguments (optional):

```
<sts_var>: displays the current value of the specified variable
```

Example:

```
AT+S.STS<cr>
# Status & Statistics:
#  version = ******-*******-SPWF01S
#  reset_reason = 2
#  conf_flag = 5
#  system_uptime = 1067
#  system_sleeptime = 0
#  gpio_enable = 0
#  captiveportal = 0
#  wifi_state = 10
#  wifi_bssid = 8C:4D:EA:04:93:47
#  wifi_aid = 3
#  wifi_channelnum = 1
#  wifi_sup_rate_mask = 0x003FFFCF
#  wifi_bas_rate_mask = 0x0000000F
#  wifi_chan_activity2 = 0x00003FFF
#  wifi_max_tx_power = 18
#  wifi_gf_mode = 0
#  wifi_reg_country = JP
#  wifi_dtim_period = 0
#  wifi_sleeping = 0
#  wifi_num_assoc = 0
#  ip_ipaddr = 192.168.1.14
#  ip_netmask = 255.255.255.0
#  ip_gw = 192.168.1.1
#  ip_dns = 192.168.1.1
#  ip_sock_open = 0
#  ip_sockd_port = 0
#  free_heap = 20592
#  min_heap = 20080
#  current_time = 1134

OK


AT+S.STS=system_uptime<cr>
#  system_uptime = 7001

OK
```

## 2.11 AT+S.PEERS: dump contents of the peer table

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

Arguments (optional):

```
<peer_number>: identifier of the peer
<peer_var>: displays the current value of the specified peer variable
```

Example:

```
AT+S.PEERS<cr>
- device connected to the MiniAP

# Size of peer table: 5
#  1.link_id = 1
#  1.state = 4
#  1.addr = 90:18:7C:96:0D:0B
#  1.last_rx = 32
#  1.last_tx = 15
#  1.rx_drops = 8
#  1.tx_drops = 0
#  1.rx_pkts = 206
#  1.tx_pkts = 7
#  1.tx_errs = 0
#  1.rate_mask = 0x00003FCF
#  1.cur_rate_idx = 6
#  1.cur_rate_ok = 1
#  1.cur_rate_fail = 0
#  1.tx_consec_fail = 0
#  1.rx_seqnum = 0x00002920
#  1.rx_seqnum_mc = 0x00000000
#  1.rx_rssi = -27
#  1.rx_rateidx = 0
#  1.setprot = 0
#  1.listen_interval = 10
#  1.capinfo = 0x00000000
#  2.link_id = 0
#  2.state = 0
#  2.addr = 00:00:00:00:00:00
#  3.link_id = 0
#  3.state = 0
#  3.addr = 00:00:00:00:00:00
#  4.link_id = 0
```

```
#  4.state = 0
#  4.addr = 00:00:00:00:00:00
#  5.link_id = 0
#  5.state = 0
#  5.addr = 00:00:00:00:00:00

OK

AT+S.PEERS=1,rx_rssi<cr>

#  0.rx_rssi = -33

OK
```

## 2.12 AT+S.RMPEER: disassociate a peer

AT+S.RMPEERS allows to disassociate a peer when the module works in MiniAP mode.
The peer number can be retrieved with the AT+S.PEERS command and must be specified.

Arguments:

<peer_number>          identifier of the peer

## 2.13 AT+S.PING: send a ping to a specified host

AT+S.PING issues a single ICMP ECHO request to the given host.

Arguments:

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

Example:
```
AT+S.PING=192.168.1.254<cr>
#PING: sendto 192.168.1.254<cr><lf>
<cr><lf>OK<cr><lf>

AT+S.PING=example.com<cr>
#PING: sendto 192.0.43.10<cr><lf>
<cr><lf>OK<cr><lf>

AT+S.PING=nonexistent.example.com<cr>
<cr><lf>ERROR: DNS lookup failure<cr><lf>

AT+S.PING=192.168.1.1<cr>
```

```
#PING: sendto 192.168.1.1<cr><lf>
<cr><lf>ERROR: Timed out<cr><lf>
```

## 2.14 AT+S.SOCKON: open a network socket

AT+S. SOCKON opens a TCP/UDP/Secure socket to "myserver" on port "xxxx"

Arguments:

```
<hostname>: target host. DNS resolvable name or IP address
<port>: TCP/UDP socket port
<protocol>: t for TCP socket, u for UDP socket, s for secure socket,
ServerName for secure socket using given string as server name indication
(SNI)
ind: indicate when data has arrived (optional); this option requires to read
the socket when a pending indication message is received.
```

*Note:*     *- Up to 8 TCP or UDP sockets can be opened contemporary. In this case it is strictly suggested to immediately empty the buffer (using sockr command) when a pending data is received.*

*- Up to 4 consecutive 730 Bytes "Pending data" messages (w/o SOCKR) are guaranteed. To prevent data loss, it is suggested to empty the buffer by using the AT+S.SOCKR command and to avoid exceeding 4 indications.*

*- TCP: The MSS (maximum segment size) is equal to 730 bytes; UDP: the datagram len is further limited as the TCP's MSS*

*- When a socket client receives an indication about socket server gone (only for TCP/Secure sockets, WIND:58), the socket connection is not automatically closed. Moreover, both for TCP and UDP sockets, 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.*

Example:

```
AT+S.SOCKON=myserver,1234,t<cr>
ID: 00<cr><lf>
<cr><lf> OK<cr><lf>

AT+S.SOCKON=myserver,456,u,ind<cr>
ID: 01<cr><lf>
<cr><lf> OK<cr><lf>
```

## 2.15 AT+S.SOCKW: write len bytes of data to socket

AT+S. SOCKW allows data to socket to be written. This command accepts data after the <cr> at the end of the command line. The host is expected to supply <len> characters of data after the end of the command line.

*Note:* *When the ok token has not been received, the error stage needs to be handled properly. In fact, the "AT+S.SOCKW" command is not reentrant. If bytes are lost during data transfer over the UART, the module remains in waiting stage for incoming bytes.*

*"AT+S.SOCKW" command splits outgoing bytes into smaller packets, TCP MSS (730 bytes) as maximum size. This also implies that ip_mtu configuration variable only has effect for smaller sizes than 730.*

*"AT+S.SOCKW" command over UDP sockets is not delayed till IP layer becomes ready for transmission. It is suggested to run an "AT+S.PING" command before very first write command, to avoid fragments loss.*

Arguments:
```
<ID>: socket identifier
<len>: data length to send (in bytes), up to 4096 bytes
```

Example:
```
AT+S.SOCKW=00,11<cr>
Test_socket
<cr><lf> OK<cr><lf>
```

## 2.16 AT+S.SOCKQ: query pending data

AT+S.SOCKQ returns the number of bytes of data waiting on socket.

Arguments:
```
<ID>: socket identifier
```

Example:
```
AT+S.SOCKQ=01<cr>
<cr><lf>
DATALEN: 12
<cr><lf> OK<cr><lf>
```

## 2.17 AT+S.SOCKR: return len bytes of data from socket

AT+S. SOCKR allows to read data from socket.

Arguments:
```
<ID>: socket identifier
<len>: data length to read
```

Example:
```
AT+S.SOCKR=01,12<cr>
Test_socket1
<cr><lf> OK<cr><lf>
```

## 2.18 AT+S.SOCKC: close socket

The SOCKC command allows to close socket.

*Note:* *Both for TCP/Secure and UDP sockets, 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.*

Arguments:

```
<ID>: socket identifier
```

Example:

```
AT+S.SOCKC=00<cr>
<cr><lf> OK<cr><lf>
```

## 2.19 AT+S.TLSCERT: configure SSL/TLS certificates

AT+S.TLSCERT allows to store the certificates in the Flash memory of the module.

*Note:* *Refer to the SSL/TLS application note for details.*

Arguments:

```
<f_ca|f_cert|f_key>: store the CA certificate, the client certificate or
the key file (PEM format)
<length>: size of the certificate
Examples:

- Store the CA's certificate

AT+S.TLSCERT=f_ca,<ca_size_in_bytes><CR>
-----BEGIN CERTIFICATE-----<CR><LF>
[...]
-----END CERTIFICATE-----<CR><LF>

OK<CR><LF>

- Store the client's certificate

AT+S.TLSCERT=f_cert,<cert_size_in_bytes><CR>
-----BEGIN CERTIFICATE-----<CR><LF>
[...]
-----END CERTIFICATE-----<CR><LF>

OK<CR><LF>

- Store the client's private key

AT+S.TLSCERT=f_key,<key_size_in_bytes><CR>
-----BEGIN EC PRIVATE KEY-----<CR><LF>
```

```
[...]
-----END EC PRIVATE KEY-----<CR><LF>

OK<CR><LF>
```

## 2.20 AT+S.TLSCERT2: Cleanup SSL/TLS certificate resources

AT+S.TLSCERT2 allows to clean the certificates in the flash memory of the module.

*Note:* *Refer to the SSL/TLS application note for details.*

Arguments:

```
clean,<f_ca|f_cert|f_key|f_domain|all>: clean the CA certificate, the
client certificate, the key file, the server domain or all
Examples:

- Clean the CA's certificate

AT+S.TLSCERT2=clean,f_ca<CR>

OK<CR><LF>

- Clean the client's certificate

AT+S.TLSCERT2=clean,f_cert<CR>

OK<CR><LF>

- Clean the client's private key

AT+S.TLSCERT2=clean,f_key<CR>

OK<CR><LF>

- Clean the server's domain

AT+S.TLSCERT2=clean,f_domain<CR>

OK<CR><LF>

- Clean all the certificates

AT+S.TLSCERT2=clean,all<CR>

OK<CR><LF>
```

## 2.21 AT+S.TLSDOMAIN: set Server domain name. It must match the secured site name

The TLSDOMAIN command allows the certification authority domain name to be stored in the Flash memory of the module.

*Note:* *Refer to the SSL/TLS application note for details.*

Arguments:

```
<f_domain>: store the Server domain in the flash memory
<Server domain name>: domain name of the Server
Example:


AT+S.TLSDOMAIN=f_domain,testserver.org<CR>


OK<CR><LF>
```

## 2.22 AT+S.SETTIME: initiate module reference time

The SETTIME command allows the reference time used for secure socket connections to be set. The module reference time must be initialized after each module reset. The time refers to UTC format and must be expressed as the time in seconds since 1970-Jan-01.

*Note:* *Refer to the SSL/TLS application note for details.*

Arguments:

```
<time_in_sec>: set the reference time in seconds
no args: returns the current time in UTC format


Example:
- Set the current time to 2017-01-01 08:00:00 (i.e. 148431340)


AT+S.SETTIME=1483257600<CR>


OK<CR><LF>


- Get the current time in UTC format


AT+S.SETTIME<CR>


OK<CR><LF>


# Date and GMT Time:<CR><LF>
#  2017-1-1<CR><LF>
#  8:0:0<CR><LF>
```

## 2.23 AT+S.SOCKD: enable/disable the socket server

The SOCKD command enables the socket server listening on incoming connection on the given port. When the port argument is equal to zero, the command is used to turn off the socket server.

Under specific cases, "data mode" (if in place) automatically switches to "command mode". These cases include: network is lost (+WIND:33 is raised, followed by +WIND:41) while in STA mode, or module is disassociated (+WIND:41 is raised) or deuthenticated (+WIND:40 is raised) while in STA mode, or module is in miniAP mode and accepted client disappears, or is deauthenticating or disassociating (match is done on STA's MAC address).

Arguments:

```
<port>: server listening port (from 1 to 65634, 0 to disable the socket
server)
```

```
<protocol> t for TCP, u for UDP protocol. Default is TCP.
```

*Note:* *Client is accepted under 2 conditions: no other client was already in, and module is not executing an AT command. Be sure that console is idle, otherwise client is rejected. Do not send spurious bytes outside "data mode". If this is the case, clear console state through "AT" null command, expecting "OK" as a reply.*

*Once accepted a client, module automatically switches from "command mode" (the one which accepts AT commands) to "data mode" (the one which send to remote every byte received by UART, and prints on UART every byte received by remote). Switch back to "command mode" can be addressed by an escape sequence (set by escape_seq configuration variable) or automatically when the client closes the socket; switch again into "data mode" can be addressed by AT+S. command. Ref. to section 2.24.*

*About sending stage, internal buffer used to collect bytes received by UART is 1024 bytes in length. This also implies that ip_mtu configuration variable only has effect for smaller sizes than 1024 bytes. Since escape sequence must also be managed, buffer is considered full when it contains 1024-escape sequence length. By default, escape_seq is "at+s.", so, buffer is considered full when it contains 1019 bytes.*

*Send is automatically triggered by socket server both when buffer is full, and when timeout is detected (set by ip_sockd_timeout configuration variable).*

Example:

```
- Listening on port 32000 using TCP.


AT+S.SOCKD=32000<CR>


<CR><LF>
OK<CR><LF>
- Listening on port 32000 using UDP
AT+S.SOCKD=32000,u<CR>
<CR><LF>
OK<CR><LF>
- Turn off the socket server


AT+S.SOCKD=0<CR>


<CR><LF>
```

```
OK<CR><LF>
```

## 2.24 AT+S.: command mode to data mode

The AT+S. command allows switching from command mode to data mode.

Arguments:

```
<none>
```

*Note:* *The switch from data mode to command mode switch can be done by using the "at+s." escape sequence. This sequence can be customized by using the escape_seq configuration variable.* **The sequence is case-sensitive and it must be sent in a single complete packet with no CR or LF in the sequence.** *escape_seq must also be followed by a timeout (customized by using ip_sockd_timeout configuration variable). Otherwise, whole escape sequence and next bytes are sent to remote client.*

*Moreover, since internal buffer is 1024 bytes in size, escape sequence must not be sent to socket server around the end of the buffer. Otherwise, whole previous bytes and escape sequence are sent to remote client. Example: if buffer is containing 1023 bytes, and escape sequence is also sent to socket server, this will result into 2 packet sent to remote client: the first one will be 1024 in size (1023 data bytes, and first byte of escape sequence), and the second one with the rest of escape sequence. To avoid this case, a timeout can also be added before escape sequence sending to socket server.*

Example:

```
+WIND:59:Back to Command Mode<CR><LF>


AT+S.<CR>


<CR><LF>
+WIND:60:Now in Data Mode<CR><LF>
```

## 2.25 AT+S.HTTPGET: issue an HTTP GET

AT+S.HTTPGET performs a single HTTP GET request to the named host and path. The GET request and server response are printed on the module console. Any url-encoding required for special characters in the <path&queryopts> argument must be performed by the host prior to command submission.

*Note:* *NOTE: the <cr><lf> pairs in the example responses below are part of the data sent from the server and not inserted by the module.*

Arguments:

| | |
|---|---|
| `<hostname>` | Target host. DNS resolvable name or IP address. |
| `<path&queryopts>` | document path and optional query arguments |
| `<port>` | Target host port. Optional. |

Example:

```
AT+S.HTTPGET=host.example.com,/index.html<cr>
GET /index.html HTTP/1.0
User-Agent: SPWF01S
Host: 192.168.0.103
Connection: close


HTTP/1.0 200 OK
Server: lwIP/1.3.1 (http://savannah.nongnu.org/projects/lwip)
Content-type: text/html

<html>
<head><title>SPWF01Sx.11</title></head>
<body bgcolor="white" text="black">
<h1>ST SPWF01Sx.11 WiFi Module</h1>
<p>
Welcome to the ST SPWF01Sx.11 WiFi Module.
</p>
<p>
This page was delivered from the SPWF01Sx.11 internal HTTP server.
</p>
<p>
<a href=/config.shtml>SPWF01Sx.11 Configuration Settings Page</a>
</p>
<p>
<a href=/status.shtml>SPWF01Sx.11 Status Page</a>
</p>

</body>
</html>

OK


AT+S.HTTPGET=nonexistent.example.com,/<cr>
<cr><lf>ERROR: host not found<cr><lf>
```

## 2.26 AT+S.HTTPPOST: issue an HTTP POST

The HTTP POST performs a post of the given path to the specified host. The module can be only used as an HTTP POST client.

*Note:* *This command sends a POST request using hard coded "Content-Type: application/x-www-form-urlencoded". For custom POST requests, refer to HTTPREQ command (see* *Section 2.27: AT+S.HTTPREQ: custom HTTP request).*

Arguments:

```
<hostname>: target host. DNS resolvable name or IP address
<path&queryopts>: document path
<formcontent>: form to be submitted
<port>:target host port. Optional
```

Example:

```
at+s.httppost=posttestserver.com,/post.php,name=demo&email=mymail&subject=
subj&body=message<CR>
HTTP/1.1 200 OK<CR><LF>
Date: Wed, 22 Jan 2014 15:36:18 GMT<CR><LF>
Server: Apache<CR><LF>
Access-Control-Allow-Origin: *<CR><LF>
Vary: Accept-Encoding<CR><LF>
Content-Length: 141<CR><LF>
Connection: close<CR><LF>
Content-Type: text/html<CR><LF>
<CR><LF>
Successfully dumped 4 post variables.<LF>
View it at
http://www.posttestserver.com/data/2014/01/22/07.36.181370961643<LF>
Post body was 0 chars long.<CR><LF>
<CR><LF>
OK<CR><LF>
```

## 2.27 AT+S.HTTPREQ: custom HTTP request

AT+S.HTTPREQ performs a custom HTTP request to a specified host/port.

For example, the command can replace the AT+S.HTTPGET or AT+S.HTTPPOST when user needs to add a custom header of the request.

This command accepts data after the <cr> at the end of the command line. The host is expected to supply <datalen> characters of data after the end of the command line.

Arguments:

| | |
|---|---|
| `<hostname>` | target host. DNS resolvable name or IP address |
| `<port>` | target host port. Optional |
| `<len>` | Amount of bytes to be sent |

Example:

```
at+s.httpreq=www.wikipedia.org,80,73<CR>
GET / HTTP/1.0<CR><LF>
User-Agent: SPWF01S<CR><LF>
```

```
Host: SPWF01S<CR><LF>
Connection: close<CR><LF>
<CR><LF>
HTTP/1.1 200 OK<CR><LF>
Server: mw1252.eqiad.wmnet<CR><LF>
X-Powered-By: HHVM/3.3.0-static<CR><LF>
Last-Modified: Fri, 01 Jul 2016 18:26:44 GMT<CR><LF>
ETag: "3b2-5369720eefb07"<CR><LF>
Backend-Timing: D=232 t=1475575785211059<CR><LF>
Content-Type: text/html<CR><LF>
X-Varnish: 3816852566, 1057256502, 301625741 301428007<CR><LF>
Via: 1.1 varnish, 1.1 varnish, 1.1 varnish<CR><LF>
Date: Tue, 04 Oct 2016 10:10:20 GMT<CR><LF>
Age: 35<CR><LF>
Connection: close<CR><LF>
X-Cache: cp1053 miss, cp3033 miss, cp3043 hit/1<CR><LF>
X-Cache-Status: hit<CR><LF>
Set-Cookie: WMF-Last-Access=04-Oct-2016;Path=/;HttpOnly;secure;Expires=Sat,
05 Nov 2016 00:00:00 GMT<CR><LF>
X-Analytics: nocookies=1<CR><LF>
X-Client-IP: 62.110.21.125<CR><LF>
Set-Cookie: GeoIP=IT:25:Milan:45.47:9.20:v4; Path=/; secure;
Domain=.spwf01s<CR><LF>
<CR><LF>
<!DOCTYPE html><LF>
<html lang=en><LF>
<meta charset="utf-8"><LF>
<title>Unconfigured domain</title><LF>
<link rel="shortcut icon" href="//wikimediafoundation.org/favicon.ico"><LF>
<style><LF>
* { margin: 0; padding: 0; }<LF>
body { background: #fff; margin: 7% auto 0; padding: 2em 1em 1em; font:
14px/21px sans-serif; color: #333; max-width: 560px; }<LF>
img { float: left; margin: 0 2em 2em 0; }<LF>
a img { border: 0; }<LF>
h1 { margin-top: 1em; font-size: 1.2em; }<LF>
p { margin: 0.7em 0 1em 0; }<LF>
a { color: #0645AD; text-decoration: none; }<LF>
a:hover { text-decoration: underline; }<LF>
em { font-style: normal; color: #777; }<LF>
</style><LF>
<a href="//www.wikimedia.org"><img
src="//www.wikimedia.org/static/images/wmf.png"
srcset="//www.wikimedia.org/static/images/wmf-2x.png 2x" alt=Wikimedia
width=135 height=135></a><LF>
```

```
<h1>Domain not configured</h1><LF>
<p>This domain points to a <a href="//www.wikimedia.org">WikimediaÂ
Foundation</a> server, but is not configured on thisÂ server.</p><LF>
</html><LF>
<CR><LF>
<SUB><SUB><SUB><CR><LF>
<CR><LF>
OK<CR><LF>
```

## 2.28 AT+S.FSC: create a file

AT+S.FSC creates a "file" inside the RAM of the module for delivery by the http server. The <name> argument establishes the URL path that must be used in the HTTP GET from a remote client to access this file. The file must include a complete HTTP response header plus the document content that is delivered to the client. The software includes some statically-defined pages that cannot be removed but they can be overridden by creating a file of the same name. A dynamic file of the same name overwrites the old one.

To add content to a file see AT+S.FSA below.

---

**Warning:** **Space for files is allocated from available RAM in the module and extremely limited. Minimize the requirement for these as much as possible.**

---

Arguments:

| | |
|---|---|
| <fname> | Filename |
| <max_len> | Amount of space to allocate for file, max = 4096 bytes |
| <http_header> | 0=HTML header automatically added |
| | 1=HTML header not added (as by default) |

Example:
```
AT+S.FSC=/new.html,1024<cr>
<cr><lf>OK<cr><lf>
```

## 2.29 AT+S.FSA: Append to an existing file

AT+S.FSA appends blocks of data to an existing file. This command accepts data after the
<cr> at the end of the command line. The host is expected to supply <datalen> characters of
data after the end of the command line.

Arguments:

| | |
|---|---|
| `<fname>` | Filename |
| `<datalen>` | Amount of bytes to be appended to an existing file. The limit of bytes that can be appended depends on the space allocated during the file creation. |

Example:

```
AT+S.FSA=/data.json,165<cr>
HTTP/1.0 200 OK<cr><lf>
Server: MyProduct<cr><lf>
Connection: close<cr><lf>
Content-Type: application/json<cr><lf>
<cr><lf>
<cr><lf>
{<cr><lf>
        "device" {<cr><lf>
                "name" : "SPWF01SX.11",<cr><lf>
                "serial" : "802.11n"<cr><lf>
        }<cr><lf>
}<cr><lf>
<cr><lf>OK<cr><lf>
```

## 2.30 AT+S.FSR: rename an existing dynamic file

AT+S.FSR renames a dynamic file stored in the web server. The old name of the file and the
new name must be specified.

Arguments:

| | |
|---|---|
| `<old_name>` | name of the existing file |
| `<new_name>` | new name of the file |

Example:

```
AT+S.FSR=/old.html,/new.html<CR>


<CR><LF>
OK<CR><LF>
```

## 2.31 AT+S.FSD: delete an existing file

AT+S.FSD deletes an existing file by name. Static files may not be deleted, only overridden.

Arguments:

```
<fname>              Filename
```

Example:
```
AT+S.FSD=/data.json<cr>
<cr><lf>OK<cr><lf>
```

## 2.32 AT+S.FSL: list existing filename(s)

AT+S.FSL lists the types (I=Internal Flash Memory, D=RAM Memory, E=External Flash Memory), sizes, and names of all the existing files.

- Internal Flash pages: HTML header automatically added
- RAM memory pages: HTML header added/not added depending on the <http_header> parameter
- External Flash pages: HTML header added by default (it can be disabled acting on httpd_gen.c included in the FW package, commenting the "#define APPEND_HEADER")

Arguments:
```
none
```

Example:
```
D 1965 /wifidemo.html
 I 461 /input_demo.shtml
 I 180 /message.shtml
 I 384 /output_demo.html
 I 614 /index.html
 I 157 /peers.shtml
 I 193 /config.shtml
 I 174 /status.shtml
 I 212 /404.html
 I 2022 /firstset.html
 I 2898 /remote.html

OK
```

## 2.33 AT+S.FSP: Print the contents of an existing file

AT+S.FSP prints the contents of an existing file.

Arguments:

| | |
|---|---|
| `<fname>` | Filename |
| `<offset>` | Offset from where the file is printed.Optional. |
| `<len>` | Lenght in bytes. Mandatory if Offset is specified. |

Example:

```
AT+S.FSP=/t2.json<cr>
HTTP/1.0 200 OK<cr><lf>
Server: MyProduct<cr><lf>
Connection: close<cr><lf>
Content-Type: application/json<cr><lf>
<cr><lf>
<cr><lf>
{<cr><lf>
        "device" {<cr><lf>
                "name" : "SPWF01SX.11",<cr><lf>
                "serial" : "802.11n"<cr><lf>
        }<cr><lf>
}<cr><lf>
<cr><lf>OK<cr><lf>
```

## 2.34 AT+S.WIFI: enable/disable Wi-Fi device

AT+S.WIFI allows the radio to be enabled or disabled at runtime. Please note that the configuration variable wifi_mode controls the state of the radio at powerup.

Arguments:

0 or 1, for disabled or enabled, respectively.

Example:

```
AT+S.WIFI=0<cr>
<CR><LF>
OK<CR><LF>
<CR><LF>
+WIND:49:WPA:Terminated: 0<CR><LF>
<CR><LF>
+WIND:38:WiFi:Powered Down<CR><LF>
```

## 2.35 AT+S.ROAM: trigger Wi-Fi reassociation sequence

AT+S.ROAM tells the module to disassociate from its current access point and to re-acquire the network. This is particularly useful if the network settings have been changed and a reboot is not desired. The function is not active when the module is configured in mini AP mode.

Arguments:

Example:

```
AT+S.ROAM<cr>
<cr><lf>OK<cr><lf>
```

## 2.36 AT+S.GPIOC: configure general purpose inputs/outputs

AT+S.GPIOC is used to configure the function of the various GPIOs on the module. GPIOs can be configured as inputs or outputs. Additionally, inputs can be configured to generate an indication when their state changes.

Hint.

Use an external pull-up/pull-down connected to a given GPIO to prevent unwanted commutations.

Arguments:

| | |
|---|---|
| `<num>` | GPIO Number (0-15 on the SPWF01SX.11) |
| `<direction>` | "in" or "out" |
| `<interrupt>` | Optional parameter: |
| 0 | Off |
| R | Rising edge |
| F | Falling edge |
| B | Both rising and falling edges |

Example:

```
AT+S.GPIOC=7,out<cr>
<cr><lf>OK<cr><lf>

AT+S.GPIOC=11,in,B<cr>
<cr><lf>OK<cr><lf>
```

## 2.37 AT+S.GPIOR: query general purpose input

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

Arguments:

```
<num>  GPIO Number (0-15 on SPWF01SX)
```

Example:

```
at+s.gpior=4<CR>

GPIO 4 = 0,in<CR><LF>
<CR><LF>
OK<CR><LF>
```

## 2.38 AT+S.GPIOW: set general purpose output

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

Arguments:

| | |
|---|---|
| `<num>` | GPIO Number (0-15 on the SPWF01SX) |
| `<value>` | 0 or 1 for off and on, respectively |

Example:
```
AT+S.GPIOW=7,1<cr>
<cr><lf>OK<cr><lf>
```

## 2.39 AT+S.FWUPDATE: perform a firmware update

AT+S.FWUPDATE downloads an updated firmware image via a single HTTP GET request to the named host and path, much like the AT+S.HTTPGET command. The SPWF01SX.11 validates the firmware image it downloads, loads it into a staging area, then prompts the user to issue a reset command in order to complete the update. A restoring of factory default settings (AT&F) is mandatory after every FW update.

*Note:*     *- Command only enabled on the module versions SPWF01Sx.1y*

*- The HW factory reset (GPIO0 enabled) can be used after "+WIND:17:F/W update complete!".*

Arguments:

| | |
|---|---|
| `<hostname>` | Target host. DNS resolvable name or IP address |
| `<path&queryopts>` | document path and optional query arguments |
| `<port>` | Target host port. Optional |

Example:
```
AT+S.FWUPDATE=host.example.com,/1203-120918_01.ota<cr>
Staging F/W update for 'SPWF01SX.11' version '1203-120918_01'
F/W length 276824 @ 0x00002800 (offset 0x00000000, block len 4096)
Write len 4096 -> 0x0
Write len 4096 -> 0x1000
Write len 4096 -> 0x2000
Write len 4096 -> 0x3000
```
*(note - deleted extra output for clarity)*
```
Write len 4096 -> 0x41000
```

```
Write len 4096 -> 0x42000
Write len 2476 -> 0x43000 (final)
Wrote 276904 bytes
Complete! Please reboot. (at+cfun=1)
AT+CFUN=1<cr>
+WIND:2:RESET
```

*(note: if the GPIO0 is high at this stage, the external flash is erased and the FWUPDATE is not performed)*

```
+WIND:17:Validating F/W update


+WIND:17:Performing F/W update
```

*(note - at this point the LEDS blink rapidly until update is complete)*

```
+WIND:17:F/W update complete!
```

*(note - The HW factory reset (GPIO0 enabled) can be used after the above +WIND)*

```
+WIND:1:Poweron 140128-caf4b79-SPWF01S
```

## 2.40 AT+S.HTTPDFSUPDATE: update static HTTPD file system via HTTP GET

AT+S.HTTPDFSUPDATE downloads an updated file system via a single HTTP GET request to the named host and path, much like the AT+S.HTTPGET command. The SPWF01SX.11 validates the image it downloads, flashes the contents and then prompts the user to issue a reset command in order to complete the file system update.

Arguments:

| | |
|---|---|
| `<hostname>` | Target host. DNS resolvable name or IP address |
| `<path&queryopts>` | document path and optional query arguments |
| `<port>` | Target host port |

Example:

```
AT+S.HTTPDFSUPDATE=host.example.com,/custom_httpdfs.img<cr>
Image length 777 (offset 0x00080000, block len 4096)
Write len 784 -> 0x80000 (final)
Wrote 780 bytes
Complete! Please reboot. (at+cfun=1)
OK
```

## 2.41 AT+S.HTTPDFSWRITE: update static HTTPD file system via UART

AT+S.HTTPDFSWRITE downloads an updated file system via UART interface. The

SPWF01SX.11validates the image, flashes the contents and then prompts the user to issue a reset command in order to complete the file system update. This command accepts data after the <cr> at the end of the command line. The host is expected to supply <datalen> characters of data after the end of the command line.

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

Arguments:

```
<datalen>                Amount of bytes to be sent
```

Example:

```
AT+S.HTTPDFSWRITE=777<cr>
(binary data must be sent now)


Image length 777 (offset 0x00080000, block len 4096)
.....
......
......


Complete! Please reboot. (at+cfun=1)


OK
```

## 2.42 AT+S.HTTPDFSERASE: erase the external Flash memory

The HTTPDFSERASE allows the content of the external flash t be erased.

Arguments:

```
<none>
```

Example:

```
AT+S.HTTPDFSERASE<cr>


Complete! Please reboot. (at+cfun=1)


OK
```

## 2.43 AT+S.HTTPD: disable/enable web server

The HTTPD command enables or disables the module web server.

Alternatively, the <ip_use_httpd> variable can be modified in order to permanently enable/disable the web server.

Arguments:

```
<on/off>: 0 to disable, 1 to enable
```

Example:

```
AT+S.HTTPD=0<CR>


<CR><LF>
OK<CR><LF>
```

## 2.44 AT+S.SCAN: perform site survey (scan)

AT+S.SCAN performs an immediate scan for available networks. Infrastructure (AP) and IBSS (Ad-Hoc) networks are both reported. Network type, Channel, BSSID, SSID, Signal strength (RSSI), and 802.11 capabilities are all reported. The module supports the active/passive scan and the filtered/unfiltered scan. Default is active and filtered on SSID.

**Figure 2. 802.11 Capabilities information example**

```
▽ Capabilities Information: 0x0001
      .... .... .... ...1 = ESS capabilities: Transmitter is an AP
      .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
      .... ..0. .... 00.. = CFP participation capabilities: No point coordinator at AP (0x0000)
      .... .... ...0 .... = Privacy: AP/STA cannot support WEP
      .... .... ..0. .... = Short Preamble: Short preamble not allowed
      .... .... .0.. .... = PBCC: PBCC modulation not allowed
      .... .... 0... .... = Channel Agility: Channel agility not in use
      .... ...0 .... .... = Spectrum Management: dot11SpectrumManagementRequired FALSE
      .... .0.. .... .... = Short Slot Time: Short slot time not in use
      .... 0... .... .... = Automatic Power Save Delivery: apsd not implemented
      ..0. .... .... .... = DSSS-OFDM: DSSS-OFDM modulation not allowed
      .0.. .... .... .... = Delayed Block Ack: delayed block ack not implemented
      0... .... .... .... = Immediate Block Ack: immediate block ack not implemented
```

*Note:* *The automatic scan, performed by the module to connect to the access point, is passive by default. This is done to avoid violating spectral emission. The switch to active scan only happens when the module finds an AP advertising the country IE, or when wifi_region configuration variable is set to 2 to 7.*

Arguments:

`<a|p>`: Perform an active scan (a) or a passive scan (p)

`<r|s|m>`: duplicated network are displayed (r), default-filtered networks with the same SSID (s), filtered networks with the same MAC (m)

`<fname>`: print the results to file

Example:

```
AT+S.SCAN<cr>
1:<HT> BSS 00:18:74:D3:53:C0 CHAN: 01 RSSI: -85 SSID: 'Ambu2' CAPS: 0431 WPA2 <CR><LF>
2:<HT> BSS 00:18:74:D3:53:C3 CHAN: 01 RSSI: -85 SSID: 'PAWAM' CAPS: 0431 WPA <CR><LF>
3:<HT> BSS 00:18:74:D3:53:C1 CHAN: 01 RSSI: -85 SSID: 'AmbuM' CAPS: 0431 WPA2 <CR><LF>
4:<HT> BSS 00:18:74:D3:53:C2 CHAN: 01 RSSI: -84 SSID: 'Guest' CAPS: 0421 <CR><LF>
5:<HT> BSS C8:D3:A3:15:98:14 CHAN: 05 RSSI: -79 SSID: 'AmbuITguest' CAPS: 0431 WPA WPA2 WPS <CR><LF>
6:<HT> BSS 02:62:1F:51:8F:0B CHAN: 06 RSSI: -41 SSID: 'ciscosb2' CAPS: 0411 WPA WPA2 <CR><LF>
7:<HT> BSS 00:18:0A:31:EA:78 CHAN: 11 RSSI: -89 SSID: 'ZyckoItalyWiFi' CAPS: 0531 WPA WPA2 <CR><LF>
8:<HT> BSS 00:1F:33:FE:66:17 CHAN: 11 RSSI: -91 SSID: 'TRI_AGRATE' CAPS: 0411 WPA2 <CR><LF>
<CR><LF>
```

```
OK<CR><LF>


AT+S.SCAN=a,r<cr>
1:<HT> BSS 00:18:74:D3:53:C0 CHAN: 01 RSSI: -81 SSID: 'Ambu2' CAPS: 0431
WPA2 <CR><LF>
2:<HT> BSS 00:18:74:D3:53:C1 CHAN: 01 RSSI: -80 SSID: 'AmbuM' CAPS: 0431
WPA2 <CR><LF>
3:<HT> BSS 00:18:74:D3:53:C2 CHAN: 01 RSSI: -81 SSID: 'Guest' CAPS: 0421
<CR><LF>
4:<HT> BSS 00:18:74:D3:53:C3 CHAN: 01 RSSI: -81 SSID: 'PAWAM' CAPS: 0431 WPA
<CR><LF>
5:<HT> BSS 00:18:74:D3:53:C1 CHAN: 01 RSSI: -81 SSID: 'AmbuM' CAPS: 0431
WPA2 <CR><LF>
6:<HT> BSS 00:18:74:D3:53:C0 CHAN: 01 RSSI: -81 SSID: 'Ambu2' CAPS: 0431
WPA2 <CR><LF>
7:<HT> BSS 00:18:74:D3:53:C3 CHAN: 01 RSSI: -80 SSID: 'PAWAM' CAPS: 0431 WPA
<CR><LF>
8:<HT> BSS 00:18:74:D3:53:C1 CHAN: 01 RSSI: -81 SSID: 'AmbuM' CAPS: 0431
WPA2 <CR><LF>
9:<HT> BSS C8:D3:A3:15:98:14 CHAN: 05 RSSI: -79 SSID: 'AmbuITguest' CAPS:
0431 WPA WPA2 WPS <CR><LF>
10:<HT> BSS 02:62:1F:51:8F:0B CHAN: 06 RSSI: -39 SSID: 'ciscosb2' CAPS:
0411 WPA WPA2 <CR><LF>
11:<HT> BSS 02:62:1F:51:8F:0B CHAN: 06 RSSI: -45 SSID: 'ciscosb2' CAPS:
0411 WPA WPA2 <CR><LF>
12:<HT> BSS 00:1F:33:FE:66:17 CHAN: 11 RSSI: -91 SSID: 'TRI_AGRATE' CAPS:
0411 WPA2 <CR><LF>
<CR><LF>
OK<CR><LF>
Example: active scan, filtered on MAC address, results saved in a file


AT+s.scan=a,m,/scan.html<CR>


<CR><LF>
OK<CR><LF>


at+s.fsl<CR>


 D 452 /scan.html<CR><LF>
 I 461 /input_demo.shtml<CR><LF>
 I 384 /output_demo.html<CR><LF>
 I 614 /index.html<CR><LF>
 I 157 /peers.shtml<CR><LF>
 I 193 /config.shtml<CR><LF>
 I 180 /message.shtml<CR><LF>
 I 174 /status.shtml<CR><LF>
 I 5496 /remote.html<CR><LF>
```

```
 I 3447 /firstset.html<CR><LF>
 I 212 /404.html<CR><LF>
<CR><LF>
OK<CR><LF>


at+s.fsp=/scan.html<CR>


1:<HT>  BSS 9C:97:26:B1:03:B5 CHAN: 06 RSSI: -68 SSID: 'InfostradaWiFi-
B103B5' CAPS: 0411 WPA WPA2 WPS <CR><LF>
2:<HT>  BSS 9C:D3:6D:01:10:23 CHAN: 06 RSSI: -85 SSID: 'NETGEAR36' CAPS:
0411 WPA2 WPS <CR><LF>
3:<HT>  BSS 80:B6:86:6F:CF:72 CHAN: 11 RSSI: -68 SSID: 'E5830-cf721' CAPS:
0431 WPA2 WPS <CR><LF>
4:<HT>  BSS 14:CC:20:F8:9F:FC CHAN: 11 RSSI: -39 SSID: 'E5830-cf72' CAPS:
0431 WPA2 <CR><LF>


<CR><LF>
OK<CR><LF>
```

## 2.45 AT+S.ADC: read ADC value on GPIO8

AT+S.ADC returns ADC value on GPIO8, between 0 and 2500 mV.

*Note:* *Measurement accuracy is around 10 mV*

Arguments:

```
<raw>: returns raw ADC value on GPIO8 unprocessed (between 0 and 4096). This
argument is optional and when it is not specified the value is returned
processed (between 0 and 2500 mV)
```

## 2.46 AT+S.DAC: enable/disable DAC on GPIO15

The DAC command enables DAC on GPIO15.

Arguments:

```
<Value>: must be set in mV (between 1 and 2500), 0 disables DAC on GPIO15
```

## 2.47 AT+S.PWM: set PWM on GPIO1

The PWM command enables PWM on GPIO1 with a specified frequency and duty-cycle.

*Note:* *The max. frequency value (10 kHz) allows user to set any duty-cycle between 0 and 100*

Arguments:

```
<frequency>: value between 1 and 10 KHz, 0 disables PWM on GPI01
<Duty-Cycle>: value between 0 and 100 (default=50%)
```

# 3        Configuration variable reference

The configuration variable space is split into two areas: production data (PDATA) and configuration data. The production data space contains factory-set variables that can be modified in RAM (AT+S.SCFG) but cannot be saved to non-volatile storage. The configuration data space contains variables that can be written in RAM (AT+S.SCFG) and written to non-volatile storage (AT&W). Additionally, the non-volatile values can be restored to their factory state using the AT&F command.

Variables have the following types:

**Table 3. Variable types**

| Type code | Description |
|---|---|
| TEXT[<len>] | Printable text up to <len> characters |
| HEX[<len>] | Octets, specified in hexadecimal, up to <len> octets |
| INT | Integer |
| IP | IP address or netmask, specified as a dotted-quad |

The following table lists the production data variables:

**Table 4. Production data variables**

| Variable | Sample Value | Type | Description |
|---|---|---|---|
| nv_manuf | ST | TEXT[32] | Manufacturer ID string |
| nv_model | SPWF01Sxyz | TEXT[32] | Manufacturer model string |
| nv_serial | 1214003 | TEXT[32] | Manufacturer serial number |
| nv_wifi_macaddr | 02:4D:53:4D:00:01 | HEX[6] | Manufacturer assigned 802.11 MAC Address |

The following table lists the configuration data variables:

**Table 5. Configuration data variables**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| eft_mode | 0 | INT | Enable/disable the Engineering Test Functions. It is used for controlling the radio for the certification tests (disabled by default=0, enabled=1) |
| blink_led | 0 | INT | Enable/disable the blinking LED (default=0). In MiniAP, the blinking indicates the number of clients associated to the module. |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| wind_off_low | 0x00000000 | INT | Wind 0:31 mask 0xFFFFFFFF are disabled all the 32 Wind indicator |
| wind_off_medium | 0x00000000 | INT | Wind 32:63 mask |
| wind_off_high | 0x00000000 | INT | Wind 64:95 mask |
| user_desc | anonymous | TEXT[64] | Free form textual field for host use (used as basic authentication during Mini AP configuration) |
| escape_seq | at+s. | TEXT[7] | Escape sequence from data mode to command mode (max. 7 chars) |
| localecho1 | 1 | INT | Echo command input: 0=off, 1=on (save config and reboot to take effect) |
| console1_speed | 115200 | INT | Serial port speed: from 9600 to 921600, default: 115200 |
| console1_hwfc | 0 | INT | Hardware flow control: 0=off, 1=on |
| console1_enabled | 1 | INT | Enable console on UART1 |
| console1_delimiter | 0x0000002C | HEX | Set the console delimiter. The comma (hex=2C) is set by default. The new delimiter must be specified in hex |
| console1_errs | 1 | INT | Disable/enable the error numbering (0 = displays only ERROR message , 1 = displays ERROR and REASON (default), 2 = displays ERROR, NUMBER and REASON) |
| sleep_enabled | 0 | INT | Enable/disable the sleep mode |
| standby_enabled | 0 | INT | Enable/disable the standby mode |
| standby_time | 10 | INT | Standby mode time, in seconds. Up to $2^{32}$-1 sec |
| wifi_tx_msdu_lifetime | 0 | INT | MSDU lifetime. From 0 to 2^32-1 TUs (1 TUs= 1024µs). Zero is default (automatic) |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|----------|--------------|------|-------------|
| wifi_rx_msdu_lifetime | 0 | INT | MSDU lifetime. From 0 to 2^32-1 TUs (1 TUs= 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 | 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 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: from 0 to 3000 |
| wifi_ssid | 50:72:6F:64:75: 63:74:69:6F:6E: 31:00:00:00:00: 00:00:00:00:00: 00:00:00:00:00: 00:00:00:00:00: 00:00 | HEX[32] | Desired SSID specified in hex. All 32 octets should be written. Note that wifi_ssid_len must also be set |
| wifi_ssid_len | 11 | INT | Length of the actual SSID 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 |
| wifi_add_tim_ie | 0 | INT | Enable broadcasting of TIM information element into miniAP beacons |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| wifi_ht_mode | 0 | 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=3FFFCF to enable all the data rate supported) |
| wifi_channelnum | 6 | 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 | 0x00003FCF | 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] |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| wifi_mode | 1 | INT | Radio Mode. 0=IDLE 1=STA (Supported Security Modes: OPEN, WEP OpenSystem, WEP SharedKey, WPA/WPA2 - wifi_auth_type must be set to 0) 2=IBSS (Supported Security Modes: OPEN, WEP OpenSystem, WEP SharedKey) 3=MiniAP (Supported Security Modes: OPEN, WEP OpenSystem - Supported Classes: b,g) * |
| wifi_region | 1 | INT | 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] |
| 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 | 1 | 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 |
| wifi_rssi_thresh | -50 | INT | Low signal strength threshold |
| wifi_rssi_hyst | 10 | 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 |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| 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 | 2 | 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 |
| 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: 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: 00:00 | HEX[32] | Pre-calculated PSK key |
| wifi_wpa_psk_text | a_psk_pass | TEXT[64] | WPA(2) PSK passphrase, if set the actual PSK is generated from this. |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| ip_use_dhcp | 1 | INT | DHCP server on/off. Used in STA, IBSS and MiniAP. 0=off (in STA mode: the variables ip_ipaddr, ip_netmask and ip_gw must be properly set to connect to the AP), 1=on (in STA mode: the ipaddr, netmask and gw will be provided by the AP), 2=on&customize (in MiniAP mode: user can customize the ip_ipaddr of the MiniAP, the ip_address of the client is automatically assigned by the MiniAP) |
| ip_use_httpd | 1 | INT | HTTP server on/off. 0=off, 1=on |
| ip_mtu | 1500 | INT | IP maximum transmission unit size. Limit: from 634 to 2412 (1500 for maximum compatibility with Ethernet networks) |
| ip_hostname | iwm-02-09-97 | 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_dns | 192.168.0.1 | IP | IP Primary DNS server for static usage (DHCP off) |

**Table 5. Configuration data variables (continued)**

| Variable | Sample value | Type | Description |
|---|---|---|---|
| ip_http_get_recv_timeout | 1000 | INT | HTTP connection timeout in milliseconds |
| ip_wait_timeout | 12000 | INT | Amount of time (in milliseconds) in time_wait state |
| ip_dhcp_timeout | 20 | INT | DHCP client timeout, in seconds |
| ip_sockd_timeout | 250 | INT | Socket server - buffer timeout management (from 1 ms to 250 ms) |
| ip_dhcp_lease_time | 120 | INT | IP address renew of the peers in MiniAP mode |
| ip_dns_mode | 0 | INT | Select the method of Domain Name resolution (0 = 3 requests to primary DNS and then tries 3 times with secondary DNS (default), 1 = alternates requests to primary and secondary DNS |
| ip_use_cgis | 0x0000000F | INT | Enable/disable the CGIs integrated into webserver, all CGIs enables by default (OUTPUT_CGI_BIT=0, INPUT_CGI_BIT=1, REMOTE_CGI_BIT=2, FIRSTSET_CGI_BIT=3) |
| ip_use_ssis | 0x0000000F | INT | Enable/disable the SSIs integrated into webserver, all SSIs enables by default (STATUS_SSI_BIT=0, CONFIG_SSI_BIT=1, PEERS_SSI_BIT=2, INPUT_SSI_BIT=3) |
| ip_use_decoder | 0x00000000 | INT | Enable/disable the decoding functions into weberser in order to manage non-ASCII chars (0 = no decoding [default], 1 = RAW decoding (the fields must be completed in HEX, 2 = UTF-8 decoding, 6 = HTML entities decoding) |
| ip_block_pings | 0 | INT | Allow/block echoing to unicast pings |

*Note:*         *Please check the settings of the wireless adapter of the peer in case of browser connection issue. The wireless adapter in power save mode can limit the stability of the final application*

# 4 Status variable reference

**Table 6. Status variable**

| Variable | Sample value | Description |
|---|---|---|
| version | 140128-caf4b79-SPWF01S | SPWF01S software version |
| 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 |
| conf_flag | 5 | Module HW revision |
| system_uptime | 22006 | System running time in seconds |
| system_sleeptime | 500 | System sleeping time in seconds |
| gpio_enable | 0 | Interrupt-enabled GPIO bitmask, expressed in base 10 |
| captiveportal | 1 | Mini AP DHCP and DNS Server enabled/disabled |
| wifi_state | 10 | 0= Hardware power up<br>1=Hardware failure (user must reboot the module)<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_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_activity2 | 0x00003FFF | Channels where we are allowed to transmit. Channel mask. i.e. 0x00003FFF => from channel 0 to channel 13 |
| wifi_max_tx_power | 18 | max allowed transmit power for the defined reg domain |
| wifi_reg_country | IT | Current regulatory domain |

**Table 6. Status variable (continued)**

| Variable | Sample value | Description |
|---|---|---|
| wifi_dtim_period | 1 | AP reported DTIM period (used in STA mode) |
| wifi_add_tim_ie | 1 | AP send TIM (traffic indication map) information element (used in AP mode) |
| wifi_sleeping | 0 | Radio sleeping state (0 = active, 1 = sleep) |
| wifi_num_assoc | 1 | Number of the client associated to the module |
| 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_dns | 192.168.123.20 | Current IP Primary DNS server |
| ip_dns2 | 0.0.0.0 | Current IP secondary DNS server (if given by AP) |
| ip_sock_open | 0 | Bitmask of Socket Client ID currently opened, expressed in base 10 (ip_sock_open=13 (00001101 in binary), means that socket#0 socket#2 and socket#3 are currently opened) |
| ip_sockd_port | 0 | Socket server port opened |
| ip_client_maccaddr | 00:00:00:00:00:00 | MAC address of accepted socket client |
| free_heap | 30472 | Current free heap space |
| min_heap | 26552 | Minimum free heap space thus far |
| current_time | 90643 | Current time in seconds |

**Table 7. Peers table**

| Variable | Sample value | Description |
|---|---|---|
| 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 | 90:18:7C:96:0D:0B | MAC address of the client |
| last_rx | 21244 | Timestamp of last received packet |
| last_tx | 21244 | Timestamp of last transmitted packet |
| rx_drops | 0 | Count of frames dropped during reception |
| tx_drops | 1 | Count of frames dropped during transmission |
| rx_pkts | 50 | Count of received frames |

**Table 7. Peers table (continued)**

| Variable | Sample value | Description |
|---|---|---|
| tx_pkts | 44 | 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 | 5 | 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 | 0x0000AF40 | Sequence number of last RX directed frame |
| rx_seqnum_mc | 0x00000000 | Sequence number of last RX multicast frame |
| rx_rssi | -37 | Signal strength of last received packet |
| rx_rateidx | 0 | Rate index of last received packet |
| setprot | 0 | Bitmask to indicate protection for TX (bit 1) and/or RX (bit 0) IEEE 802.11 frames |
| listen_interval | 10 | AP reported listen interval |
| capinfo | 0x00000000 | Information about the AP capabilities |

# 5 Asynchronous indication reference

The SPWF01SX modules can output asynchronous indications at any time except while an AT command is in progress. The format for all asynchronous indications is:

`<cr><lf>+WIND:<num>:<description><cr><lf>`

**Table 8. Asynchronous indication messages**

| Indication | Notes |
|---|---|
| +WIND:0:console active | Console task is running and can accept AT commands |
| +WIND:1:Poweron (%s) | Initial power-up indication, with f/w version |
| +WIND:2:RESET | System reset is being asserted/triggered |
| +WIND:3:watchdog running | Watchdog task initialized and running |
| +WIND:4:heap too small | Selected heap allocation is too small for normal operation |
| +WIND:5:WiFi hardware failure: (%d) | WIFI radio failure; user must reboot the module |
| +WIND:6:Watchdog Terminating, reset pending | Watchdog reset asserted |
| +WIND:7:SysTickConfigure | Failure to configure system tick clock |
| +WIND:8:Hard Fault | OS hard fault detected |
| +WIND:9:StackOverflow | OS stack overflow detected |
| +WIND:10:MallocFailed (%d/%d) | OS heap allocation failed (RequiredSize/FreeSpace) |
| +WIND:11:<error> | Radio Initialization failure |
| +WIND:12:WiFi PS Mode Failure: %s: %d | Radio failed to enter power saving state (%s=step,%d=state) |
| +WIND:13:<copyright information> | Copyright information of SPWF01SX |
| +WIND:14:WiFi BSS Regained | Radio regained association after loss |
| +WIND:15:WiFi Signal LOW (%d) | Radio low signal threshold triggered |
| +WIND:16:WiFi Signal OK (%d) | Radio signal level recovered |
| +WIND:17:F/W update <state> | Firmware update in progress [1] |
| +WIND:18:Keytype%d Not implemented | Encryption key type not recognized |
| +WIND:19:WiFi Join:%m | BSS join successful,%m=BSSID |
| +WIND:20:JOINFAILED:%04x | BSS join failed,%x = status code |
| +WIND:21:WiFi Scanning | Radio is scanning for a BSS that matches the currently configured SSID. (Note: WIND hidden when fast reconnect [1] is performed) |
| +WIND:22:SCANBLEWUP | Radio failed to accept scan command |
| +WIND:23:SCANFAILED:%04x | Radio failed to execute scan command |
| +WIND:24:WiFi Up:%i | Radio has successfully connected to a BSS and initialized the IP stack.%i=IP Address |

**Table 8. Asynchronous indication messages (continued)**

| Indication | Notes |
|---|---|
| +WIND:25:WiFi Association with '%s' successful | Radio successfully associated to the "%s" BSS |
| +WIND:26:WiFi Started AP with network "%d" | Radio successfully started the Mini AP, where %d=network SSID |
| +WIND:27:STARTFAILED: %04x | Radio failed to start the Mini AP,%x=status code |
| +WIND:28:Station %m Associated: %d | Client associated to the module in Mini AP, %m=BSSID, %d=peers assoc status (0=default, 1=client re-association) |
| +WIND:29:DHCP reply for %i/%m | DHCP reply sent for the client ,%i = client IP address, %m = client MAC Address |
| +WIND:30:WiFi BSS Lost | Beacon missed from the BSS |
| +WIND:31:WiFi EXCEPTION: <data> | Radio reported an internal exception. Radio is non-functional from this point; User must reboot the module. |
| +WIND:32:WiFi Hardware Started | Radio reports successful internal initialization |
| +WIND:33:WiFi Network Lost | Connection to BSS lost due to excessive beacon misses |
| +WIND:34:WiFi Unhandled Event: %d | Unhandled internal event occurred,%d=identifier of the event occurred |
| +WIND:35:Scan Complete:0x%x | Scan complete indication,%x=result code (0: scan ok; 1: scan error). Note: WIND hidden when fast reconnect [2] is performed). |
| +WIND:36:WiFi UNHANDLED IND (%02x) : <hexdata> | Unparsed radio indication occurred |
| +WIND:37:WiFi UNHANDLED (%d) : <hexdata> | Unhandled radio response message received |
| +WIND:38:WiFi: Powered Down | Radio and radio thread shut down |
| +WIND:39:HW in miniAP mode (GPIO7 Low) | Module started in miniAP mode (SSID = iwm-XX-YY-ZZ, where XXYYZZ are the last 6 digits of MAC Address) |
| +WIND:40:WiFi Deauthentication: %d | Radio: Access point sent deauthentication, :%d=reason code (802.11 Deauthentication Reason Code) |
| +WIND:41:WiFi Disassociation: %d | Radio: Access point sent disassociation, :%d=reason code (802.11 Disassociation Reason Code) |
| +WIND:42:RX_MGMT: %04x | Unhandled management frame subtype received |
| +WIND:43:RX_DATA: %04x | Unhandled data frame subtype received |
| +WIND:44:RX_UNK: %04x | Unhandled frame type received |
| +WIND:45:DOT11 AUTHILLEGAL | Illegal authentication type detected |
| +WIND:46:WPA: Crunching PSK... | Creating PSK from PSK passphrase |
| +WIND:47:WPA:%s | Factory Debug |
| +WIND:48:WPAC:%s | Factory Debug |
| +WIND:49:WPA:Terminated: %d | WPA supplicant thread terminated |

**Table 8. Asynchronous indication messages (continued)**

| Indication | Notes |
|---|---|
| +WIND:50:WPA Supplicant failed to initialize. | WPA supplicant thread initialization failed |
| +WIND:51:WPA Handshake Complete | WPA 4-way handshake successful |
| +WIND:52:GPIO%d %d | GPIO line changed state (%d=GPIO changed, %d=GPIO logic state |
| +WIND:53:Wakeup (GPIO6 High) | Device woken up from sleep from external signal |
| +WIND:54:ETF %04d | Factory Debug |
| +WIND:55:Pending Data:%d:%d | Pending data from the socket, %d =socket identifier:%d=pending byte available for reading |
| +WIND:56:Insert message to client:%d | Input_demo indicator, displayed when the "input_demo.shtml" page is requested by a client, %d is the Nth input SSI into html page |
| +WIND:57:<data> | First set indicator, displayed during the remote configuration of the module |
| +WIND:58:Socket Closed:%d" | Socket closed, %d = identifier of the socket |
| +WIND:59:Back to Command Mode | Command mode is active (after the escape sequence) |
| +WIND:60:Now in Data Mode | Data mode is active |
| +WIND:61:Incoming Socket Client:%i | Socket client is connected to the module, %i = client IP address |
| +WIND:62:Socket Client Gone:%i | Socket client disconnected, %i = client IP address |
| +WIND:63:Sockd Dropping Data:%d:%d | Data dropped due to low memory, %d=bytes dropped, %d=free heap |
| +WIND:64:Sockd Pending Data:%c:%d:%e | Data pending while module is in command mode, %c = number of message received, %d = bytes received in the last message, %e = tot bytes received |
| +WIND:65:HW Factory Reset (GPIO0 High) | Factory variables are restored via GPIO0 |
| +WIND:66:Low Power mode enabled:%d | Power Save Mode enabled, %d = 1 for PS or 2 for Fast-PS |
| +WIND:67:Going into Standby:%d | Standby mode enabled, %d is time in sec |
| +WIND:68:Resuming from Standby | Standby mode disabled |
| +WIND:69:Going into DeepSleep | Sleep mode enabled |
| +WIND:70:Resuming from DeepSleep | Sleep mode disabled |
| +WIND:71:DNS reply for %d | DNS reply from MiniAP to the client,%d = client IP address |
| +WIND:72:Station %m Disassociated:%d" | Client dissociated to the module in Mini AP, %m=BSSID,%d=reason code (802.11 Deauthentication Reason Code) |
| +WIND:73:System Configuration Updated (Run AT&W to Save it) | The configuration variables have been updated, it needs an AT&W to save it (this WIND is usually shown when an old FW version is updated) |

**Table 8. Asynchronous indication messages (continued)**

| Indication | Notes |
|---|---|
| +WIND:74:Rejected found Network | A new scan needs to be scheduled due to a mismatch between SPWF configuration variables and Access Point configuration |
| +WIND:75:Rejected Association:yyy | Indicates an association failure (yyy=low memory, reject status code) |
| +WIND:76:Authentication Timed Out | Indicates that the authentication process is timed out |
| +WIND:77:Association Timed Out | Indicates that the association process is timed out |
| +WIND:78:MIC Failure | Michael MIC error is detected by the local driver |
| +WIND:79:Bad CRC Factory Reset. (Run AT&W to Save it) | The configuration variables have been restored to factory set, it needs an AT&W to save it (this WIND is usually shown when a CRC check failure happened on saved configuration) |
| +WIND:80:UDP Broadcast Received:%i:%d | Broadcast datagram from the socket client, %i = socket client IP address, %d = incoming datagram length. Datagram content will follow this message |

1. Boot messages are always sent out using 115200 as hardcoded baudrate.

2. Fast reconnect feature: allows fast reconnect to the last associated AP.

**Table 9. Errors**

| Number | String | Meaning |
|---|---|---|
| **Bad usage - generic** | | |
| 0 | Command not found | |
| 1 | Missing argument | |
| 2 | Missing argument(s) | |
| 3 | Bad argument(s) | |
| 4 | Variable not found | Argument refers to a not existent configuration variable |
| 5 | Unrecognized key | Argument refers to a not existent status/peer_status variable |
| 6 | Too many arguments | |
| 7 | Invalid argument | |
| 8 | Invalid option | |
| 9 | Unused | |
| **Bad usage - specific** | | |
| 10 | UART not supported | |
| 11 | Region not allowed | |
| 12 | Timeout ms value not allowed | $1<=ip\_sockd\_timeout<=250$ |
| 13 | Power setting not allowed | $0<=wifi\_tx\_power<=18$ |

**Table 9. Errors**

| Number | String | Meaning |
|---|---|---|
| 14 | Channel number not allowed | 1<=wifi_channelnum<=14 |
| 15 | Disable sleep, before enabling standby | |
| 16 | Disable standby, before enabling sleep | |
| 17 | SSID too long (32 chars max.) | |
| 18 | Must specify '0' or '1' | AT+S.WIFI and AT+S.HTTPD |
| 19 | Direction must be 'in' or 'out' | AT+S.GPIOC |
| 20 | Invalid GPIO number (0-15)" | |
| 21 | Cannot use GPIO6 when sleep_enabled | |
| 22 | Reserved | |
| 23 | Cannot use GPIO10 when blink_led | |
| 24 | Cannot use GPIO11 when wifi enabled | |
| 25 | Output voltage not allowed | AT+S.DAC manages 0V to 2V5 |
| 26 | Frequency not supported | AT+S.PWM manages 0 Hz to 10 kHz |
| 27 | Duty cycle not supported | AT+S.PWM manages 0 to 100% dc |
| 28 | PWM not running | Cannot AT+S.PWM=0 when PWM is already halted |
| 29 | Cannot roam in AP mode | |
| 30 | Can disconnect peers only in AP mode | |
| 31 | Cannot disconnect this peer | Cannot disconnect if not (at least) authenticated |
| 32 | Max_len exceeds 4096 | AT+S.FSC |
| 33 | Max_len evaluated to zero | AT+S.FSC size<=0 |
| 34 | datalen must be >0 | AT+S.FSA size<=0 |
| 35 | Data mode not available | |
| 36 | Cannot send more than 4Kb in a try | AT+S.SOCKW |
| 37 | Unhandled action | AT+S.TLSCERT argument different from "f_cert", "f_ca" and "f_key" |
| 38 | Invalid certificate type | |
| 39 | PIN needs to be 8 digits | |
| **Bad result** | | |
| 40 | Scan in progress | |
| 41 | Scan failed | |
| 42 | Wait for hardware busy | Join in progress |
| 43 | Wait for hardware starting | HW not ready, or dead (check cw1200 state) |
| 44 | Unable to complete PWM setting | Bad value evaluated on "auto reload register" |

**Table 9. Errors**

| Number | String | Meaning |
|--------|--------|---------|
| 45 | ADC calibration reset | Failed to reset calibration |
| 46 | ADC calibration status | Failed to get calibration status |
| 47 | ADC calibration flag | Failed to get "end of conversion" status |
| 48 | File not found | AT+S.FSA, AT+S.FSP, AT+S.FSD and AT+S.FSR |
| 49 | File is a static entry | AT+S.FSA, AT+S.FSD and AT+S.FSR |
| 50 | File length overrun | AT+S.FSA more than expected bytes |
| 51 | Data too large | AT+S.TLSCERT exceeds 4092 bytes |
| 52 | DNS lookup failure | Address not resolved |
| 53 | DNS address failure | Null address resolved |
| 54 | Timed out | Retry, or increase http_get_recv_timeout |
| 55 | Bad peer number | Peer==0 in STA/IBSS, and 0<peer<=5 in miniAP mode |
| 56 | Bad port number | HTTP port>65535 |
| 57 | host not found | Address not resolved |
| 58 | conn_new() failed, aborting | Netconn not created |
| 59 | connect() failed, aborting | |
| 60 | write() failed, aborting | HTTP request failed |
| 61 | callback failure, aborting | FW/HTTPDFS error on flash or binary image |
| 62 | close() failed, aborting | |
| 63 | delete() failed, aborting | |
| 64 | failed | Other generic failure on HTTP request |
| 65 | nothing received, increase ip_http_get_recv_timeout variable | |
| 66 | Shutdown not performed (timewaited connections) | |
| 67 | Socket Server already up and running | |
| 68 | Turn off Web Server before opening Socket Server on port 80 | |
| 69 | Socket Server not running | Cannot AT+S.SOCKD=0 when Socket Server is already halted |
| 70 | Init not performed (socket not created) | |
| 71 | Init not performed (bind failed) | |
| 72 | Init not performed (listen failed) | |
| 73 | Init not performed (time-waited connections) | |

**Table 9. Errors**

| Number | String | Meaning |
|---|---|---|
| 74 | Too many sockets | |
| 75 | Illegal Socket ID | |
| 76 | Pending data | Empty bytes before AT+S.SOCKC |
| 77 | TLS socket already in use | |
| 78 | Unable to load CA certificate | Invalid certificate |
| 79 | Unable to load Private Key | Invalid certificate |
| 80 | Unable to load client certificate | Invalid certificate |
| 81 | Failed to resolve name | Address not resolved |
| 82 | Failed to connect | |
| 83 | SSL/TLS Error: Bad Domain Name (%d) | Cfr. CyaSSL site |
| 84 | SSL/TLS Error: Server Domain Name is empty | |
| 85 | SSL/TLS Error: Unable to connect (%d) | Cfr. CyaSSL site |
| 86 | Data lost during upload phase | |
| 87 | Server closed during data sending | |
| 88 | Not enough data in buffer | AT+S.SOCKR more than stored bytes |
| 89 | Socket error | Failed to AT+S.SOCKR |
| **Low RAM and HW dead** | | |
| 90 | There is not enough free space | Low RAM |
| 91 | Low Memory Error | Low RAM |
| 92 | low memory, aborting | Low RAM |
| 93 | Low Memory Error, aborting | Low RAM |
| 94 | Failed to allocate netconn | Low RAM |
| 95 | Failed to allocate socket | Low RAM |
| 96 | Unable to create TLS context | Low RAM |
| 97 | HTTP GET request too long, aborting | |
| 98 | HTTP POST request too long, aborting | |
| 99 | Scan Aborted | WIFI radio failure; user must reboot the module |
| **Bad part** | | |
| 100 | Failed to restore to factory defaults | STM32 Flash broken |
| 101 | Failed to save settings to flash | STM32 Flash broken |
| 102 | Failed to save new NVDATA to flash | STM32 Flash broken |
| 103 | Reserved | |
| 104 | Failed to store TLS Certs into Flash | STM32 Flash broken |

**Table 9. Errors**

| Number | String | Meaning |
|---|---|---|
| 105 | Failed to clean CA Certs from Flash | STM32 Flash broken |
| 106 | Failed to clean Client Certs from Flash | STM32 Flash broken |
| 107 | Failed to clean Client Key from Flash | STM32 Flash broken |
| 108 | Failed to clean Domain Name from Flash | STM32 Flash broken |
| 109 | Invalid Flash Parameters | External Flash broken |
| **Not used** | | |
| 110 | GPIO test FAIL | |
| 111 | Request failed | |
| 112 | Open gone bad | |
| 113 | Write gone bad | |
| 114 | Close gone bad | |
| 115 | Cannot stop beaconing | |
| 116 | Cannot start beaconing | |
| 117 | Must specify '1', '2', or '3' for UART | |
| 118 | Console active | |
| 119 | Invalid port | |
| 120 | Unable to initialize socket | |
| 121 | Enable UART2 through console2_enabled variable, before using it | |
| 122 | Enable UART3 through console3_enabled variable, before using it | |

# 6 Revision history

**Table 10. Document revision history**

| Date | Revision | Changes |
|------|----------|---------|
| 05-Dec-2013 | 1 | Initial release. |
| 23-Jun-2014 | 2 | Major review for alignment with commands and variables introduced in the release 3.1 of "AT full stack" |
| 08-Oct-2014 | 3 | Minor changes. |
| 28-Nov-2014 | 4 | – Deleted the Appendix A.<br>– Minor changes related to fix introduced in the release 3.3. |
| 21-May-2015 | 5 | Changes throughout the document related to features introduced in release 3.4 of the "AT full stack". (see Table 1). |
| 19-Feb-2016 | 6 | Added: *Table 9 on page 49* |
| 25-Nov-2016 | 7 | Added Example in Section 2.27 on page 23.<br>Added Figure 2 on page 33, updated text in Section 2.44 on page 33.<br>Updated Table 9 on page 48 (several modifications).<br>Minor modifications throughout document |
| 10-Jul-2018 | 8 | Updated *Table 2*, *Table 5*, *Table 6*, *Table 8* and *Table 9*.<br>Updated *Section 2.3*, *Section 2.14*, *Section 2.15*, *Section 2.18*, *Section 2.19*, *Section 2.20*, *Section 2.21*, *Section 2.22*, *Section 2.23*, *Section 2.24*, *Section 2.26*, *Section 2.32*. |

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**