Anybus® BY HMS NETWORKS

# Wireless Bolt™ / Wireless Bridge II™

## AT Commands

**REFERENCE GUIDE**

**SCM-1202-004 2.2 en-US ENGLISH**

HMS Connecting Devices™

# Important User Information

## Disclaimer

The information in this document is for informational purposes only. Please inform HMS Networks of any inaccuracies or omissions found in this document. HMS Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Networks and is subject to change without notice. HMS Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the product. The effects caused by any direct or indirect use of such aspects of the product are undefined and may include e.g. compatibility issues and stability issues.

# Table of Contents

Page

This page intentionally left blank

# 1 Preface

## 1.1 About This Document

This document describes the available AT commands for Anybus Wireless Bolt/Bridge II.

The reader of this document is expected to be familiar with the product and have a good knowledge of wireless communication and network technology.

For additional related documentation, file downloads and technical support, please visit the Anybus support website at www.anybus.com/support.

## 1.2 Document history

**Revision list**

| Version | Date | Description |
|---|---|---|
| 1.0 | 2016-06-27 | Beta release |
| 1.1 | 2016-10-01 | First public release |
| 1.2 | 2017-03-31 | Updated for Wireless Bridge II |
| 1.3 | 2017-09-21 | Update for SP2 |
| 1.4 | 2017-12-21 | Update for FW 1.3.9 |
| 1.5 | 2018-02-15 | Updated script examples |
| 1.6 | 2018-03-08 | Added unsolicited events |
| 1.7 | 2018-09-03 | Update for FW 1.6.3<br>Script examples are now available on support web |
| 1.8 | 2019-06-10 | Update for FW 2.0.2 |
| 1.9 | 2020-05-29 | Update for FW 2.02.1<br>Added ATS Serial Registers, AT*WKEYIA Write Encryption/Authentication Key (with Index and Authentication mode), AT*WSIP WLAN IP address and Serial Commands<br>Renamed AT*AMPSM PROFIsafe Mode to AT*AMPSM ConfigLock Mode |
| 2.0 | 2020-11-04 | Update for FW 2.03.02<br>Commands for Bluetooth Low Energy Peripheral and Neighborhood watch.<br>Feature to select interfaces for the DHCP server function. |
| 2.1 | 2021-03-12 | Update for FW 2.04.02<br>In Network Commands, added new command AT*ANDHCPIF.<br>The commands AT*STM, AT*SSP, AT*SCIP, AT*SMGM and AT*SMGP have been moved from Serial Commands to Serial Data Tunnel Commands.<br>Added CAN Commands, new section with commands for Anybus Wireless Bolt CAN. |

## 1.3 Trademarks

# 2 Introduction

AT commands allow more configuration options than the web interface and can be scripted for batch configuration of multiple units. A string of AT commands can for example be sent from a PLC for automatic configuration during initial setup or when replacing units.

Each command line can only contain a single command and must not exceed 300 characters. Some commands may have additional limitations. This document describes the structure and syntax of each command and also includes examples for most of them.

Some of the commands require that the unit is rebooted before they become effective. This is indicated in the description of the command.



*UPPER CASE is only used for clarity in this manual, AT commands are not case sensitive.*

## 2.1 Data Types

The description of each command also specifies the data types used for the parameter values. There are five different data types:

**String**

Strings can contain all the printable characters from the ISO 8859-1 (8-bit ASCII) character set except " (double quote) , (comma) and \ (backslash).

The string does not need surrounding quotes.

**Integer**

Integer values can be entered in decimal form or as a hexadecimal string beginning with `0x`; e.g. `15` can also be entered as `0x0000000F`.

**Boolean**

Boolean values can be either `0` (false) or `1` (true).

**NetworkAddress**

Used for IP addresses. Must be entered as four integer values in the range 0 to 255 separated by periods, e.g. `192.168.0.98`.

**MACAddress**

Used for Ethernet and Bluetooth MAC addresses. Addresses must be entered as six groups of two hexadecimal digits in one of the following formats:

```
00A0F7101C08
00:A0:F7:10:1C:08
00-A0-F7-10-1C-08
```

## 2.2 ATS Command Syntax

**Read S-register**

```
ats[reg]?
```

**Example 1:** Read register 3002

```
ATS3002?
```

**Set S-register**

```
ats[reg]=[value]
```

**Example 2:** Set register 3002 to value 15

```
ATS3002=15
```

# 3 AT Commands

## 3.1 Standard Commands

### 3.1.1 AT&F Restore to Factory Settings

**AT&F**

This command instructs the unit to set all parameters to their defaults as specified by the manufacturer.

**Syntax:**

```
AT&F
```

### 3.1.2 AT* List Available Commands

Returns a list of all available AT commands

**AT***

**Syntax:**

```
AT*
```

**Example:**

```
Input: AT*<br>Output:<br>AT&F<br>AT*<br>AT<br>AT*ANDHCP?
<br>AT*ANDHCP=<br>AT*ANIP?<br>AT*ANIP=<br>AT*ANHN?<br>AT*ANHN=
<br>AT*BCP=<br>...<br>
```

### 3.1.3 AT Attention

**AT**

Attention command determining the presence of a DCE

**Syntax:**

```
AT
```

# 3.2 Network Commands

## 3.2.1 AT*ANDHCP DHCP Mode

Set/get the DHCP mode. If activated, this will take precedence over settings made with AT*ANIP. For default value see AT*AMDEFAULT.

**AT*ANDHCP=**

Set the DHCP mode

**Syntax:**

`AT*ANDHCP=<dhcp_client>,<dhcp_server>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| dhcp_client | Integer | 0: Off, use static IP address<br>1: On, acquire an IP address using DHCP |
| dhcp_server | Integer | 0: Off, Disable DHCP server<br>1: ON, Enable DHCP server<br>2: DHCP Relay, Relay DHCP messages to an external DHCP server. |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*ANDHCP?**

Read the current DHCP setting

**Syntax:**

`AT*ANDHCP?`

**Example:**

`AT*ANDHCP?<br>*ANDHCP:<dhcp_client>,<dhcp_server>`

### 3.2.2 AT*ANDHCPIF DHCP Server Interfaces

Set/get the DHCP server interfaces. These are the interfaces that will be served by the internal DHCP server/relay.

**AT*ANDHCPIF=**

Set the DHCP server interfaces

**Syntax:**

`AT*ANDHCPIF=<interfaces>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| interfaces | Integer | 0: All<br>1: Only the wired Ethernet interface<br>2: All supported wireless interfaces (WLAN/Bluetooth) |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*ANDHCPIF?**

Read the current DHCP server interface setting

**Syntax:**

`AT*ANDHCPIF?`

**Example:**

`AT*ANDHCPIF?<br>*ANDHCPIF:<interfaces>`

### 3.2.3 AT*ANDHCPSTA DHCP Start Address Offset

Set/get the DHCP start address offset. Used when the internal DHCP server is enabled. For default value see AT*AMDEFAULT.

**AT*ANDHCPSTA=**

Set the DHCP start address offset.

**Syntax:**

`AT*ANDHCPSTA=<start_address_offset>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| start_address_offset | Integer | Start address of the DHCP IP range. The internal DHCP server IP range will then be calculated once the DHCP server is enabled by taking the static assigned IP-address (AT*ANIP) and modifying the last octet to the inputted start address offset (start_address_offset). It is impossible to assign the start address offset to values equal to 0 or above 247. If the device static IP-address is within the calculated DCHP range, that address will be skipped and the next address used instead.<br>Example:<br>If the start address offset is set to 201, it will start at 201 and hand out 7 addresses. If the devices static IP-address is 192.168.0.99, then the DHCP IP-addresses shall be calculated as follows.<br>*192.168.0.201<br>*192.168.0.202<br>*192.168.0.203<br>*192.168.0.204<br>*192.168.0.205<br>*192.168.0.206<br>*192.168.0.207 |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*ANDHCPSTA?**

Read the DHCP start address offset.

**Syntax:**

`AT*ANDHCPSTA?`

**Example:**

`INPUT: AT*ANDHCPSTA? <br>OUTPUT: *ANDHCPSTA:<start_address_offset>`

### 3.2.4 AT*ANDHCPTAB DHCP Table

Get the DHCP Table. If the DHCP server is enabled, this command will read out an array of assigned IP-addresses, the associated Client-ID and the associated lease times (time until the lease expires, in seconds) for the IP-address, where the first element is the first assigned address. If the DHCP server is disabled, this command will return ERROR.

**AT*ANDHCPTAB?**

Read the DHCP Table

**Syntax:**

```
AT*ANDHCPTAB?
```

**Example:**

```
Output format: *ANDHCPTAB: <ip>,<client_id>,<lease><br><br>Input:
AT*ANDHCPTAB?<br>Output:<br>*ANDHCPTAB:
192.168.0.201,03001D002B01,600<br>*ANDHCPTAB:
192.168.0.202,030146002D00,600<br>...
```

### 3.2.5 AT*ANIP IP Settings

Set/get IP settings for the device

**AT*ANIP=**

Write IP address and related information. The information set by this command will not be valid until after the module is restarted.

**Syntax:**

```
AT*ANIP=<ip_addr>,<netmask>,<gateway>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ip_addr | NetworkAddress | IP address for the device |
| netmask | NetworkAddress | Netmask for the device |
| gateway | NetworkAddress | The IP address of the gateway |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*ANIP?**

Get the IP settings

**Syntax:**

```
AT*ANIP?
```

**Example:**

```
AT*ANIP?<br>*ANIP:<ip_addr>,<netmask>,<gateway>
```

### 3.2.6 AT*ANHN Hostname

Set/get the hostname used with dynamic DNS

**AT*ANHN=**

Set hostname

**Syntax:**

```
AT*ANHN=<hostname>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| hostname | String | The hostname to set. Maximum of 128 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*ANHN?**

Get hostname

**Syntax:**

```
AT*ANHN?
```

**Example:**

```
AT*ANHN?<br>*ANHN:<hostname>
```

## 3.3 Bluetooth Classic Commands

### 3.3.1 AT*BCP Connect Peer

**AT*BCP=**

Bluetooth Connect to Peer. The connection will not be retried if unsuccessful.

**Syntax:**

`AT*BCP=<bd_addr>,<name>,<role>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | MACAddress | If specified the MAC address of the remote Bluetooth device must match this value. |
| name | String | If name is specified and S register 2017 is 1 the remote name must match this value exactly. If S register 2017 is 0 this is a case sensitive substring of the remote name to connect to, e.g. if specified to DUT it will try to connect to DUT, DUTx, xDUT and xDUTx, but not to dut. |
| role | Integer | The role of the remote device:<br>100: PAN User role, PAN Profile<br>101: Network Access Point role, PAN Profile,<br>103: PAN, This will first try to connect to PANU, and if it fails, connect to NAP<br>All others:Reserved |

**Example:**

`Input: AT*BCP=8C8B83EE2ACB,,101 will return the handle of the connection and OK if the connection succeeds, ERROR otherwise.`

### 3.3.2 AT*BCC Close Connection

**AT*BCC=**

Bluetooth Close Connection

**Syntax:**

`AT*BCC=<handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| handle | Integer | The handle of the connection to close. If set to 0 and there is no connection with handle 0 any ongoing connection attempts and retries will be aborted. |

**Example:**

`Input: AT*BCC=0 gives OK when the connection with handle 0 is closed.`

### 3.3.3 AT*BC Connect

**AT*BC**

Bluetooth Connect (according to the Connection List).

**Syntax:**

`AT*BC`

**Example:**

`Input: AT*BC will return the handle of the connection and OK if the connection succeeds, ERROR otherwise.`

### 3.3.4 AT*BND Name Discovery

**AT*BND=**

Bluetooth Name Discovery

**Syntax:**

`AT*BND=<bd_addr>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | MACAddress | MAC address of the Bluetooth device to get the name of. |

**Example:**

`Input: AT*BND=8C8B83EE2ACB gives the name of the device and OK if successful, ERROR otherwise.`

### 3.3.5 AT*BDD Device Discovery

**AT*BDD**

Perform a Bluetooth Device Discovery i.e. an Inquiry followed by a named lookup for any device that does not report a name in the inquiry response.

**Syntax:**

`AT*BDD`

**Example:**

`Input: AT*BDD returns *BDD:<bd_addr>,<cod>,<device_name_valid>,<bluetooth_name>,<rssi> for each found device followed by OK or ERROR.`

### 3.3.6 AT*BI Inquiry

**AT*BI**

Perform a Bluetooth inquiry.

**Syntax:**

```
AT*BI
```

**Example:**

```
Input: AT*BI returns *BI:<bd_addr>,<cod>,<device_name_
valid>,<bluetooth_name>,<rssi> for each found device followed by OK or
ERROR.
```

### 3.3.7 AT*BSP Server Profile

**AT*BSP=**

Sets the Bluetooth server profile. A reboot is needed for the setting to take effect. Please note that following values will be affected depending on what role is selected: NAP: AT*BMSP Master Slave policy will be set to 0, ATS2010 max number of connections will be set to 7, AT*BCM Connectability mode will be set to 2. PANU: AT*BMSP Master Slave policy will be set to 1, ATS2010 max number of connections will be set to 1, AT*BCM Connectability mode will be set to 1. IMPORTANT: As the device is connectable after NAP has been set an appropriate Security Mode should be configured.

**Syntax:**

```
AT*BSP=<server_profile>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| server_profile | Integer | The role of the device:<br>100: PAN User role, PAN Profile<br>101: Network Access Point role, PAN Profile. |

*Requires a reboot for the changes to take effect.*

**Example:**

```
Input: AT*BSP=101 sets the device to the Network Access Point role.
```

**AT*BSP?**

Gets the Bluetooth server profile.

**Syntax:**

```
AT*BSP?
```

**Example:**

```
Input: AT*BSP? returns the server profile. See AT*BSP= for values.
```

### 3.3.8 AT*BFP Fixed PIN

**AT*BFP=**

Set the fixed pin/passkey used for BT authentication

**Syntax:**

`AT*BFP=<pin>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| pin | String | The pin/passkey to set. A numerical value 0..999999. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BFP?**

Get the fixed pin/passkey used for BT authentication.

**Syntax:**

`AT*BFP?`

### 3.3.9 AT*BPM Pairing Mode

**AT*BPM=**

Set the pairing mode for BT

**Syntax:**

`AT*BPM=<pair_mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| pair_mode | Integer | The mode to set. Pairing off = 1, Pairing on = 2.<br>Note: This also applies to Bluetooth LE. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BPM?**

Get the pairing mode for BT. Pairing off = 1, Pairing on = 2.

**Syntax:**

`AT*BPM?`

**Example:**

`Input: AT*BPM?<br>*BDM: <pair_mode><br>OK`

## 3.3.10 AT*BSM Security Mode

**AT*BSM=**

Set the security mode to use for BT. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BSM=<security_mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| security_mode | Integer | The security mode to set.<br>1 = Security disabled, No encryption or authentication.<br>2 = Fixed pin, Encrypted connection with PIN code security. This mode only works between two units of this type and brand (Not with third-party devices, Use Just works in that case). PIN codes must consist of 4 to 6 digits.<br>3 = Just works, Encrypted connection without PIN code.<br>Note: For Bluetooth LE only 1 = Security Disabled and 3 = Just Works is supported. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BSM?**

Get the security mode used for BT. See AT*BSM= for values.

**Syntax:**

`AT*BSM?`

**Example:**

`Input: AT*BSM?<br>*BSM: <security_mode><br>OK`

### 3.3.11 AT*BBM BT Bridge Mode

**AT*BBM=**

Set the bridge mode for BT. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BBM=<mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The bridge mode to set.<br>1 = Bridge mode disabled.<br>2 = IP forward. (Needed if Bluetooth connection to an android device. You also need an active DCHP server to be able to connect to an android device.) |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BBM?**

Get the bridge mode used for BT. See AT*BBM= for values.

**Syntax:**

`AT*BBM?`

**Example:**

`Input: AT*BBM?<br>*BBM: <mode><br>OK`

### 3.3.12 AT*BBD Bonded Devices

**AT*BBD?**

Get the bonded devices. Note: This also applies to Bluetooth LE.

**Syntax:**

`AT*BBD?`

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | String | Bluetooth address of the bonded device. |
| is_le_device | Boolean | 0: BT Classic Device<br>1: BT LE Device |

**Example:**

`AT*BBD?<br>*BBD:<bd_addr1>,<is_le_device1><br>*BBD:<bd_addr2>,<is_le_device2><br>...<br>OK`

### 3.3.13 AT*BUB Unbond

**AT*BUB=**

Un-bonds a previously bonded device.

**Syntax:**

`AT*BUB=<bd_addr>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | MACAddress | MAC address of the Bluetooth device to un-bond. If address FFFFFFFFFFFF is selected, all bonded devices will be removed.<br>Note: Deleting seperate BLE devices is not supported, to delete BLE device bonds FFFFFFFFFFFF have to be used. |

**Example:**

`Input: AT*BUB=8C8B83EE2ACB`

### 3.3.14 AT*BLEM Low Emission Mode

Note: this command is not related to Bluetooth low energy.

**AT*BLEM=**

Set current Low Emission Mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BLEM=<mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The Low Emission mode to set:<br>0: Connection period: 5000 ms Paging timeout: 8 000 ms Inquiry timeout: 10 240 ms Link Supervision Timeout: 2000ms<br>1: Connection period: 5000ms Paging timeout: 8 000 ms Inquiry timeout: 10 240 ms Link Supervision Timeout: 500ms<br>2: Connection period: 5000ms Paging timeout: 8000 ms Inquiry timeout: 8000 ms Link Supervision Timeout: 200ms<br>3 - 63: Reserved<br>64: User specified times, see the ATS General Settings S Register Manipulation command |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BLEM?**

Get the current Low Emission Mode. See AT*BLEM= for values.

**Syntax:**

`AT*BLEM?`

**Example:**

`Input: AT*BLEM?<br>*BLEM: <mode><br>OK`

### 3.3.15 AT*BDM GAP Discoverability Mode

**AT*BDM=**

Set current GAP discoverability mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BDM=<gap_mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| gap_mode | Integer | The GAP discoverability mode to set:<br>1: GAP non-discoverable mode<br>2: GAP limited discoverable mode<br>3: GAP general discoverable mode |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BDM?**

Get the current GAP discoverability mode. See AT*BDM= for values.

**Syntax:**

`AT*BDM?`

**Example:**

`AT*BDM?<br>*BDM:<gap_mode>`

### 3.3.16 AT*BCM GAP Connectability Mode

**AT*BCM=**

Set current GAP connectability mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BCM=<gap_mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| gap_mode | Integer | The GAP connectability mode to set:<br>1: GAP non-connectable mode<br>2: GAP connectable mode |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BCM?**

Get the current GAP connectability mode. See AT*BCM= for values.

**Syntax:**

`AT*BCM?`

**Example:**

`AT*BCM?<br>*BCM:<gap_mode>`

### 3.3.17 AT*BCA Connection Accept

Used when external connection control is enabled (see ATS2012)

**AT*BCA=**

Accept or reject a connection attempt. This must be sent to answer the *BCI Connect Indication.

**Syntax:**

`AT*BCA=<handle>,<accept>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| handle | Integer | The handle of the connection, received in the *BCI Connect Indication. |
| accept | Boolean | Set to 1 to accept the connection, 0 to reject it. |

### 3.3.18 AT*BLN Local Name

**AT*BLN=**

Set the unit's Bluetooth name. A reboot is needed for the setting to take effect.

**Syntax:**

`AT*BLN=<name>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| name | String | The Bluetooth name to use. The maximum length is 31 characters. |

**AT*BLN?**

Get the unit's Bluetooth name.

**Syntax:**

`AT*BLN?`

**Example:**

`AT*BLN?<br>*BLN:<name>`

### 3.3.19 AT*BRSS Read RSSI

**AT*BRSS=**

Get the RSSI for a connection.

**Syntax:**

`AT*BRSS=<handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| handle | Integer | The handle of the connection to get the RSSI for. |

**Example:**

`AT*BRSS=<handle><br>*BRSS:<rssi>`

### 3.3.20 AT*BLQ Read Link Quality

**AT*BLQ=**

Get the link quality for a connection. Link Quality is a value between 0 and 255 and it only applies to Bluetooth connections.

**Syntax:**

AT*BLQ=<handle>

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| handle | Integer | The handle of the connection to get the link quality for. |

**Example:**

AT*BLQ=<handle><br>*BLQ:<link_quality>

### 3.3.21 AT*BLP Limited Pairing

**AT*BLP=**

Enables or disables limited pairing, only valid for current power cycle. If the device should be pairable after power cycle, see S register 2007. Note: This also applies to Bluetooth LE.

**Syntax:**

AT*BLP=<enable>,<time_limit>

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| enable | Boolean | 0: Disable pairing<br>1: Enable. Pairing will be limited. |
| time_limit | Integer | The time (in seconds) the unit will be pairable. Valid time is 0 to 300 seconds. Values less than 0 will be treated as 0. |

## 3.3.22 AT*BCHM Channel Map

**AT*BCHM=**

Write the Bluetooth channel map. Note that at least 20 channels must be enabled. For default value see AT*AMDEFAULT.

**Syntax:**

```
AT*BCHM=
<ch0to15>,<ch16to31>,<ch32to47>,<ch48to63>,<ch64to78>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ch0to15 | Integer | Bit mask used to enable or disable channels 0 to 15 (Bit 0 = Channel 0). |
| ch16to31 | Integer | Bit mask used to enable or disable channels 16 to 31. (Bit 0 = Channel 16) |
| ch32to47 | Integer | Bit mask used to enable or disable channels 32 to 47 (Bit 0 - Channel 32). |
| ch48to63 | Integer | Bit mask used to enable or disable channels 48 to 63 (Bit 0 = Channel 48). |
| ch64to78 | Integer | Bit mask used to enable or disable channels 64 to 78 (Bit 0 = Channel 64). |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BCHM?**

Read the Bluetooth channel map.

**Syntax:**

```
AT*BCHM?
```

**Example:**

```
AT*BCHM?<br>*BCHM:
<ch0to15>,<ch16to31>,<ch32to47>,<ch48to63>,<ch64to78>
```

### 3.3.23 AT*BPP Packet policy

**AT*BPP=**

Set the Bluetooth packet policy. This policy is used for subsequent connections. Any ongoing connections are not affected. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BPP=<policy>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| policy | Integer | 0: Long Range (only DM1 packets).<br>1: Short Latency, basic rates (all DM packets).<br>2: High Throughput, basic rates (DM + DH packets).<br>3: As 2 but with 2-EDR enabled.<br>4:As 3 but with 3-EDR enabled |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BPP?**

Get the Bluetooth packet policy.

**Syntax:**

`AT*BPP?`

**Example:**

`AT*BPP?<br>*BPP:<policy>`

## 3.3.24 AT*BMSP Master Slave policy

**AT*BMSP=**

Set the Bluetooth Master Slave Role Policy. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BMSP=<policy>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| policy | Integer | 0: Always attempt to become master on incoming connections. Should be used for a unit configured as NAP.<br>1: Always let the connecting device select master/slave role on incoming connections. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BMSP?**

Get the Bluetooth Master Slave Role Policy.

**Syntax:**

`AT*BMSP?`

**Example:**

`AT*BMSP?<br>*BMSP:<policy>`

## 3.3.25 AT*BLCOD Local class of device.

**AT*BLCOD=**

Set the Bluetooth Local Class Of Device code. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BLCOD=<cod>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| cod | Integer | Valid values for this parameter are specified in the Bluetooth Assigned Numbers Document, www.bluetooth.com. The parameter has been divided into three segments, a service class segment, a major device class segment and a minor device class segment (bits 2-7). |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BLCOD?**

Get the Bluetooth Local Class Of Device code.

**Syntax:**

`AT*BLCOD?`

**Example:**

`AT*BLCOD?<br>*BLCOD:<cod>`

## 3.3.26 AT*BRCD Read Connected Devices.

**AT*BRCD?**

Retrieves the MAC address and handle of every connected Bluetooth device.

**Syntax:**

`AT*BRCD?`

**Example:**

`AT*BRCD? returns *BRCD:<bd_addr>,<handle> for each connected device followed by OK or ERROR.`

### 3.3.27 AT*BCLC Clear the Connection list

**AT*BCLC=**

Clears all the entries in the Connection list.

**Syntax:**

`AT*BCLC=<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`AT*BCLC=1`

### 3.3.28 AT*BCLR Read the Connection list

**AT*BCLR=**

Reads an entry in the Connection list.

**Syntax:**

`AT*BCLR=<index>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | The index of the entry to read. |

**Example:**

`AT*BCLR=2<br>*BCLR:<2>,<bd_addr>,<name>,<role>`

**AT*BCLR?**

Reads the list of Connections that the unit can use.

**Syntax:**

`AT*BCLR?`

**Example:**

`AT*BCLR?<br>returns *BCLR:<index>,<bd_addr>,<name>,<role> for each entry in the list followed by OK.`

## 3.3.29 AT*BCLW Write an entry in the Connection list

**AT*BCLW=**

Writes an entry in the Connection list. NOTE: If store is set to 1 all entries in the connection list will be stored.

**Syntax:**

```
AT*BCLW=<index>,<bd_addr>,<name>,<role>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | The index of the entry to write. |
| bd_addr | MACAddress | If specified the MAC address of the remote Bluetooth device must match this value. |
| name | String | If name is specified and S register 2017 is 1 the remote name must match this value exactly. If S register 2017 is 0 this is a case sensitive substring of the remote name to connect to, e.g. if specified to DUT it will try to connect to DUT, DUTx, xDUT and xDUTx, but not to dut. |
| role | Integer | The role of the remote device:<br>100: PAN User role, PAN Profile<br>101: Network Access Point role, PAN Profile<br>103: PAN, This will first try to connect to PANU, and if it fails, connect to NAP, All others:Reserved |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*BCLW=0,00026F668FA8,dut1,101,1<br>OK
```

## 3.3.30 AT*BPPCM Set PANU PANU Connection Mode.

**AT*BPPCM=**

Set the PANU PANU Connection Mode used in easy config. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*BPPCM=<connection_mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| connection_mode | Integer | The connection mode of PANU-PANU:<br>1: MAC only<br>2: NAME only<br>3: Both MAC and NAME |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*BPPCM?**

Reads the PANU PANU Connection Mode.

**Syntax:**

`AT*BPPCM?`

**Example:**

`AT*BPPCM?<br>*BPCCM:<mode>`

# 3.4 WLAN Commands

## 3.4.1 AT*WMODE WLAN Mode

**AT*WMODE=**

Set WLAN mode, Station or AP.

**Syntax:**

`AT*WMODE=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The mode to set, Station (0) or AP (1). |

*Requires a reboot for the changes to take effect.*

**Example:**

`Input: AT*WMODE=1 sets WLAN mode to AP.`

**AT*WMODE?**

Get WLAN mode, Station (0) or AP (1)

**Syntax:**

`AT*WMODE?`

**Example:**

`AT*WMODE?<br>*WMODE:<mode>`

### 3.4.2 AT*WKEY Encryption/Authentication Key

**AT*WKEY=**

Write encryption/authentication key at index 1. This command is a shortcut for AT*WKEYI=1,

**Syntax:**

`AT*WKEY=<key>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| key | String | The key to use. Max 63 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*WKEY=Sesame,1`

**AT*WKEY?**

Read encryption/authentication key

**Syntax:**

`AT*WKEY?`

**Example:**

`Input: AT*WKEY? returns the encryption/authentication key at index 1.`

### 3.4.3 AT*WKEYI Write Encryption/Authentication Key (with Index)

**AT*WKEYI=**

Write encryption/authentication key at any index.

**Syntax:**

`AT*WKEYI=<index>,<pKey>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | 1...4 |
| pKey | String | The key to use. Max 63 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*WKEYI=2,Sesame,1.`

### 3.4.4 AT*WKEYIA Write Encryption/Authentication Key (with Index and Authentication mode)

**AT*WKEYIA=**

Write encryption/authentication key at any index. Also enter the encoding of the key WEP/WPA

**Syntax:**

`AT*WKEYIA=<index>,<authentication>,<pKey>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | 1...4 |
| authentication | Integer | Authentication Mode: 1 = WEP64/128 (shared secret), 2 = WPA/WPA2 PSK |
| pKey | String | The key to use. Max 63 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*WKEYIA=1,2,Sesame,1.`

### 3.4.5 AT*WACTKEY Active Encryption/Authentication Key

**AT*WACTKEY=**

Set the index of the active Encryption/Authentication Key

**Syntax:**

`AT*WACTKEY=<index>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | 1..4 |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*WACTKEY?**

Get the index of the active Encryption/Authentication Key, 1..4.

**Syntax:**

`AT*WACTKEY?`

**Example:**

`Input: AT*WACTKEY?<br>*WACTKEY:<index><br>OK`

## 3.4.6 AT*WMIMO WLAN MIMO

**AT*WMIMO=**

Configures which antennas should be enabled 1: Enable only primary antenna 2: Enable both antennas NOTE: This value is ignored on devices with only one antenna.

**Syntax:**

`AT*WMIMO=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | Configures which antennas should be enabled. |

*Requires a reboot for the changes to take effect.*

**Example:**

`Input: AT*WMIMO=2 enable WLAN MIMO.`

**AT*WMIMO?**

Get WLAN MIMO, MIMO off (1) or MIMO on (2)

**Syntax:**

`AT*WMIMO?`

**Example:**

`AT*WMIMO?<br>*WMIMO:<mode>`

### 3.4.7 AT*WWM WLAN World Mode

When WLAN World Mode is enabled only frequencies accepted all over the world are used. When disabled local frequencies can be used but then the device must search for location during startup which will increase the startup time. New channels must be added manually with AT*WSCHL. To scan and discover these channels in EU, at least three scan results for Access Points containing country information indicating ETSI are needed.

**AT*WWM=**

Configures WLAN World Mode 0: Disable WLAN World Mode 1: Enable WLAN World Mode NOTE: When WLAN World Mode is disabled the startup time will increase.

**Syntax:**

`AT*WWM=<wlan_world_mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| wlan_world_mode | Integer | Enable/disable WLAN World Mode. |

*Requires a reboot for the changes to take effect.*

**Example:**

`Input: AT*WWM=0 disable WLAN World Mode.`

**AT*WWM?**

Get WLAN World Mode, World Mode off (0) or World Mode on (1)

**Syntax:**

`AT*WWM?`

**Example:**

`AT*WWM?<br>*WWM:<mode>`

# 3.5 WLAN AP Commands

## 3.5.1 AT*WASSID Access Point SSID

**AT*WASSID=**

Sets the SSID for AP mode.

**Syntax:**

`AT*WASSID=<ssid>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ssid | String | The SSID to set. Max 32 characters. |

*Requires a reboot for the changes to take effect.*

**AT*WASSID?**

Gets the SSID for AP mode.

**Syntax:**

`AT*WASSID?`

**Example:**

`AT*WASSID?<br>*WASSID:<ssid>`

### 3.5.2 AT*WACH Access Point Channel

**AT*WACH=**

Sets the channel for AP mode.

**Syntax:**

`AT*WACH=<channel>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| channel | Integer | The channel to use. Valid channels are 1-11 for 2.4 GHz and 36, 40, 44, 48 for 5 GHz. |

*Requires a reboot for the changes to take effect.*

**AT*WACH?**

Gets the channel for AP mode.

**Syntax:**

`AT*WACH?`

**Example:**

`AT*WACH?<br>*WACH:<channel>`

### 3.5.3 AT*WAAM Authentication Mode for AP

**AT*WAAM=**

Set the AP Authentication Mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*WAAM=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | Authentication Mode: 0 = Open, 2 = WPA/WPA2 PSK |

*Requires a reboot for the changes to take effect.*

**AT*WAAM?**

Get the AP Authentication Mode.

**Syntax:**

`AT*WAAM?`

**Example:**

`AT*WAAM?<br>*WAAM:<mode><br>OK`

# 3.6 WLAN Client (STA) Commands

## 3.6.1 AT*WSMAC WLAN MAC address

Can be used to clone a MAC address from an Ethernet device. This will allow layer 2 data to be bridged by one device. Can be combined with IP forwarding.

**AT*WSMAC=**

Set the WLAN MAC address. If set to all 0 or all FF the unit's default address will be used. This command is not supported on Bolt Serial products.

**Syntax:**

`AT*WSMAC=<mac>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mac | MACAddress | The MAC address to set. |

*Requires a reboot for the changes to take effect.*

**AT*WSMAC?**

Get the MAC address.

**Syntax:**

`AT*WSMAC?`

**Example:**

`AT*WSMAC?<br>*WSMAC:<mac>`

### 3.6.2 AT*WSBM WLAN Bridge Mode

**AT*WSBM=**

Set the WLAN Bridge Mode. In layer 2 tunnel mode all layer 2 data will be bridged over WLAN. Please note that this option uses a custom protocol and can only be used when the AP and the remote device are of the same type Layer 2 cloned MAC only mode is used in combination with AT*WSMAC. In this mode only data from the cloned MAC will be bridged over WLAN. When using MAC clone two devices will use the same MAC - hence there is no way for a DHCP server to distinguish them from one another and both devices will receive the same IP address. This mode is not supported on Bolt Serial products. In layer 3 IP forward mode IP data from all data will be bridged over WLAN. Please note that this mode can be combined with AT*WSMAC to enable layer 2 data for one device. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*WSBM=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The Bridge Mode to set.<br>0: Layer 2 tunnel<br>1: Layer 2 cloned MAC only<br>2: Layer 3 IP forward |

*Requires a reboot for the changes to take effect.*

**AT*WSBM?**

Get the Bridge Mode.

**Syntax:**

`AT*WSBM?`

**Example:**

`AT*WSBM?<br>*WSBM:<mode>`

### 3.6.3 AT*WSC Connect

**AT*WSC**

Connect to Access Points as specified in the Connection List.

**Syntax:**

`AT*WSC`

**Example:**

`Input: AT*WSC returns OK if the connection succeeds, ERROR otherwise.`

### 3.6.4 AT*WSCC Close Connection

**AT*WSCC**

Close WLAN connection in Station mode. If there is no connection but a connect as specified by the Connection List is in progress this is terminated.

**Syntax:**

```
AT*WSCC
```

**Example:**

```
Input: AT*WSCC, returns OK when the connection is closed.
```

### 3.6.5 AT*WSSCAN Scan

**AT*WSSCAN=**

Scan the surroundings for access points with a specific Network Name (SSID) on a specified channel.

**Syntax:**

```
AT*WSSCAN=<pssid>,<channel>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| pssid | String | The SSID to scan for. Max 32 characters. |
| channel | Integer | The channel to scan for |

**Example:**

```
Input: AT*WSSCAN=dutAP,1 will return 0...48 access points in the
immediate surroundings, then return OK.
```

**AT*WSSCAN?**

Scan the surroundings for access points. Will return 0...48 access points in the immediate surroundings, then return OK.

**Syntax:**

```
AT*WSSCAN?
```

**Example:**

```
Input: AT*WSSCAN?<br>*WSSCAN:
[bssid],[ssid],[channel],[rssi],[authentication_suit],[unicast_
ciphers],[group_cipher]
```

### 3.6.6 AT*WSAM Authentication Mode for Station

**AT*WSAM=**

Set the Station Authentication Mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*WSAM=<mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | Authentication Mode: 0 = Open, 1 = WEP64/128 (shared secret), 2 = WPA/WPA2 PSK, 3 = LEAP, 4 = PEAP<br>For WEP64/128 mode the actual mode is determined by the length of the password. WEP64 shall have a 5 character password and WEP128 shall have a 13 character password. The password can be given in hexadecimal form starting with 0x. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*WSAM?**

Get the Station Authentication Mode.

**Syntax:**

`AT*WSAM?`

**Example:**

`Input: AT*WSAM?<br>*WSAM:<mode><br>OK`

### 3.6.7 AT*WSRSS Read RSSI

**AT*WSRSS?**

Read RSSI value of the connection.

**Syntax:**

`AT*WSRSS?`

**Example:**

```
If station is connected:<br>AT*WSRSS?<br>*WSRSS:<rssi_
value><br>OK<br><br>If station is not connected:<br>AT*WSRSS?
<br>*WSRSS:-32768<br>OK<br><br>If not configured as station:
<br>AT*WSRSS?<br>ERROR
```

### 3.6.8 AT*WSLNK Read Link Status

**AT*WSLNK?**

Read current WLAN link status.

**Syntax:**

```
AT*WSLNK?
```

**Example:**

```
AT*WSLNK?<br>*WSLNK:<link_status>,<bssid>
```

### 3.6.9 AT*WSUSER User name for WLAN LEAP/PEAP authentication.

**AT*WSUSER=**

Set the user name.

**Syntax:**

```
AT*WSUSER=<user_name>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| user_name | String | The user name to set (max 63 characters) |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
Input: AT*WSUSER=Joe,1
```

**AT*WSUSER?**

Get the user name.

**Syntax:**

```
AT*WSUSER?
```

**Example:**

```
AT*WSUSER?<br>*WSUSER:<user><br>OK
```

### 3.6.10 AT*WSDOMAIN Domain for WLAN LEAP/PEAP authentication.

**AT*WSDOMAIN=**

Set the domain.

**Syntax:**

`AT*WSDOMAIN=<domain>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| domain | String | The domain to set. Max 63 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*WSDOMAIN=Cool,1`

**AT*WSDOMAIN?**

Get the domain.

**Syntax:**

`AT*WSDOMAIN?`

**Example:**

`AT*WSDOMAIN?<br>*WSDOMAIN:<domain><br>OK`

### 3.6.11 AT*WSPASS Pass phrase for WLAN LEAP/PEAP authentication.

**AT*WSPASS=**

Set the pass phrase.

**Syntax:**

```
AT*WSPASS=<pass_phrase>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| pass_phrase | String | The pass phrase to set. Max 63 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
Input: AT*WSPASS=Secret,1
```

**AT*WSPASS?**

Get the pass phrase.

**Syntax:**

```
AT*WSPASS?
```

**Example:**

```
AT*WSPASS?<br>*WSPASS:<pass_phrase><br>OK
```

## 3.6.12 AT*WSCHL Channel list

**AT*WSCHL=**

Sets the Channel list for Station mode.

**Syntax:**

```
AT*WSCHL=<channel_list_str>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| channel_list_str | String | A comma separated string of channels to use.<br>Valid channels are 1-11 for 2.4 GHz and 36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 132, 136, 140 for 5 GHz.<br>If world mode is disabled and the Unit is configured as a WLAN client channels 12, 13, 120, 124, 128, 149, 153, 157, 161, 165 are possible, but only when added to channel list AT*WSCHL=<ch1,ch2,ch3,1>.<br>See AT*WWM Description for requirements of Access Points in EU. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*WSCHL=<channel0>,<channel1>,<channel2>...,1
```

**AT*WSCHL?**

Gets the Channel list for Station mode.

**Syntax:**

```
AT*WSCHL?
```

**Example:**

```
AT*WSCHL?<br>*WSCHL:<channel0>,<channel1>,<channel2>...
```

### 3.6.13 AT*WSDP Add a default peer

**AT*WSDP=**

Adds a default peer, the setting will be stored

**Syntax:**

`AT*WSDP=<ssid>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ssid | String | The SSID of the AP. Max 32 characters. |

**Example:**

`AT*WSDP=dutAP1`

**AT*WSDP?**

Reads the default peer

**Syntax:**

`AT*WSDP?`

**Example:**

`AT*WSDP?<br>*WSDP:<ssid>`

### 3.6.14 AT*WSIP WLAN IP address

Can be used to clone a IP address from an Ethernet device. This will allow layer 2 data to be bridged in enterprise network.

**AT*WSIP=**

Set the WLAN ip address. If set to all 0 or all 255 then this setting is ignored.

**Syntax:**

```
AT*WSIP=<ip>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ip | NetworkAddress | The ip address to set. |

*Requires a reboot for the changes to take effect.*

**AT*WSIP?**

Get the IP address.

**Syntax:**

```
AT*WSIP?
```

**Example:**

```
AT*WSIP?<br>*WSIP:<ip>
```

## 3.7 Informational Commands

### 3.7.1 AT*AILVI Local Version Info

**AT*AILVI?**

Reads the local version info for the product

**Syntax:**

```
AT*AILVI?
```

**Example:**

```
AT*AILVI?<br>*AILVI:<vendor>,<fw_version>
```

### 3.7.2 AT*AILVIE Local Version Info Extended

**AT*AILVIE?**

Reads the extended local version info for the product

**Syntax:**

```
AT*AILVIE?
```

**Example:**

```
AT*AILVIE?<br>*AILVIE: Network Type: 0x4544<br>*AILVIE: Module Type:
0x0056<br>*AILVIE: Pre-Boot Version: "2.0.10"<br>*AILVIE: Bootloader
Version: "2.0.10"<br>*AILVIE: Software Version: "1.2.2-
FS2.0.0"<br>*AILVIE: Software Description: "Name of product 1.2.2-RC-
FS2.0.0"
```

### 3.7.3 AT*AIMAC Read MAC

**AT*AIMAC=**

Reads the the MAC for the specified interface

**Syntax:**

```
AT*AIMAC=<interface>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| interface | Integer | The MAC to get.<br>0: Ethernet<br>1: WLAN<br>2: Bluetooth |

**Example:**

```
AT*AIMAC=<interface><br>*AIMAC:<mac>
```

# 3.8 Miscellaneous Commands

## 3.8.1 AT*AMLI Login

**AT*AMLI=**

Log in to the AT command interface

**Syntax:**

AT*AMLI=<password>

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| password | String | The password set using AT*AMPW |

**Example:**

AT*AMLI=<password>

**AT*AMLI?**

Returns 1 if logged in

**Syntax:**

AT*AMLI?

**Example:**

AT*AMLI?<br>*AMLI:<0/1><br>OK

## 3.8.2 AT*AMLO Logout

**AT*AMLO**

Log out from the AT command interface

**Syntax:**

AT*AMLO

**Example:**

AT*AMLO<br>OK

### 3.8.3 AT*AMPW Password

**AT*AMPW=**

Set password to the AT command interface

**Syntax:**

```
AT*AMPW=<password>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| password | String | Max length is 16 characters |
| store | Boolean | If store is 1 the new value is stored permanently. |

### 3.8.4 AT*AMSTAT System status

Get the system status.

**AT*AMSTAT=**

Get the system status.

**Syntax:**

```
AT*AMSTAT=<verbose>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| verbose | Integer | 0: Terse, 1: Verbose, 2: DHCP info. |

**Example:**

```
AT*AMSTAT=0<br>*AMSTAT: Uptime:25<br><br>*AMSTAT: WLAN Mode:Station,
MAC:02:01:2E:00:24:00, state:3<br>*AMSTAT: Connected to
AP:02:01:2E:00:28:00, channel:1, rssi:-35<br><br>*AMSTAT: Bluetooth:
MAC:8C:8B:83:EE:2A:E6, State:1<br>*AMSTAT: Local name:
dut<br><br>*AMSTAT: Ethernet: MAC:02:00:2E:00:24:00, State:1,
Type:1<br>*AMSTAT: IP_ADDR:192.168.0.99<br>OK
```

### 3.8.5 AT*AMESS Event and Status Subscriber

**AT*AMESS=**

Set event and status subscriber configuration

**Syntax:**

```
AT*AMESS=<mac_addr>,<eth_type>,<ip_addr>,<udp_
port>,<protocol>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mac_addr | String | MAC address of event subscriber. Only used when protocol bit 1 is set |
| eth_type | Integer | The 16 bit Ethernet type to use. Only used when protocol bit 1 is set |
| ip_addr | String | IP address of event subscriber. Only used when protocol bit 2 is set |
| udp_port | Integer | The UDP port to use. Only used when protocol bit 2 is set |
| protocol | Integer | The protocol to use for sending events.<br>Bit 0: Send events over TCP AT connections<br>Bit 1: Send events over Layer-2 (mac_address must be specified)<br>Bit 2: Send events over Syslog<br>Bit 3: Send events over Serial port (in AT Mode) |
| store | Boolean | If store is 1 the new value is stored permanently. |

**AT*AMESS?**

Read current event subscriber settings

**Syntax:**

```
AT*AMESS?
```

**Example:**

```
AT*AMESS?<br>*AMESS:<mac_addr>,<eth_type>,<ip_addr>,<udp_
port>,<protocol><br>OK
```

### 3.8.6 AT*AMEECM Execute Easy Configuration Mode

Executes the specified Easy Configuration Mode

**AT*AMEECM=**

Executes the supplied Easy Configuration Mode.

**Syntax:**

```
AT*AMEECM=<mode>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The mode number to execute. |

**Example:**

```
AT*AMEECM=3<br>OK
```

### 3.8.7 AT*AMECFL Read/Write Easy Configuration Modes Function List

Reads/Writes the list of supported Easy Configuration Modes

**AT*AMECFL=**

Sets the list of supported Easy Configuration Modes. For default value see AT*AMDEFAULT.

**Syntax:**

```
AT*AMECFL=<easy_config_modes>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| easy_config_modes | String | Comma-separated string of up to 15 modes and their order to be supported. Valid modes are 1 to 15. Using mode=0 (INVALID MODE) will terminate the list at the given position. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*AMECFL=1,2,3,4,5,6,1<br>OK
```

**AT*AMECFL?**

Get the list of supported Easy Configuration Modes.

**Syntax:**

```
AT*AMECFL?
```

**Example:**

```
AT*AMECFL?<br>*AMECFL:<easy_config_modes><br>OK
```

### 3.8.8 AT*AMTFTP TFTP Upgrade

**AT*AMTFTP=**

Trigger a firmware update via TFTP. Device will automatically be rebooted into bootloader mode.

**Syntax:**

```
AT*AMTFTP=<device_ip>,<server_ip>,<filename>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| device_ip | NetworkAddress | The IP that the device shall use during the upgrade procedure |
| server_ip | NetworkAddress | TFTP server IP address |
| filename | String | Firmware filename (.fwz) |

### 3.8.9 AT*AMPID Product ID

**AT*AMPID?**

Get product ID

**Syntax:**

```
AT*AMPID?
```

**Example:**

```
AT*AMPID?<br>*AMPID:<vendor_id>-<platform_id>-<variant_id>
```

### 3.8.10 AT*AMSI Supported Interfaces

**AT*AMSI?**

Get the supported interfaces

**Syntax:**

```
AT*AMSI?
```

**Example:**

```
AT*AMSI?<br>*AMSI:Ethernet<br>*AMSI:WLAN (2.4 GHz)<br>*AMSI:WLAN
(5.0 GHz)<br>*AMSI:WLAN (MIMO)
```

### 3.8.11 AT*AMSBC Supported Bluetooth Configuration

**AT*AMSBC?**

Get the supported Bluetooth configuration

**Syntax:**

```
AT*AMSBC?
```

**Example:**

```
AT*AMSBC?<br>*AMSBC:<nap>,<panu>,<max_connections_classic>
```

### 3.8.12 AT*AMGD General Data

General data storage for custom data

**AT*AMGD=**

Write general data.

**Syntax:**

`AT*AMGD=<general_data>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| general_data | String | A custom string to store. Max length is 32 characters. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*AMGD=<general_data>,1`

**AT*AMGD?**

Read previously written data.

**Syntax:**

`AT*AMGD?`

**Example:**

`AT*AMGD?<br>*AMGD:<general_data>`

### 3.8.13 AT*AMTL TCP Listener

Configures the AT over TCP server

**AT*AMTL=**

Set TCP listener settings

**Syntax:**

`AT*AMTL=<port>,<enable>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| port | Integer | TCP port to listen for incoming connections |
| enable | Boolean | 0: Disables TCP Listener<br>1: Enables TCP Listener |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*AMTL?**

Get TCP listener settings

**Syntax:**

`AT*AMTL?`

**Example:**

`AT*AMTL?<br>*AMTL:<port>,<enabled>`

### 3.8.14 AT*AMBD Bridging Disable

**AT*AMBD=**

Set bridging enable/disable

**Syntax:**

`AT*AMBD=<disable>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| disable | Boolean | 0: Bridging Enabled<br>1: Bridging Disabled |
| store | Boolean | If store is 1 the new value is stored permanently. |

### 3.8.15 AT*AMLCR Layer 2 Configuration Receiver

Configure AT over layer 2 (Ethernet)

**AT*AMLCR=**

Set AT over layer 2 configuration

**Syntax:**

`AT*AMLCR=<eth_type>,<enable>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| eth_type | Integer | 16 bit Ethernet type that should be used for AT commands |
| enable | Boolean | 0: Disable AT over Ethernet<br>1: Enable AT over Ethernet |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**AT*AMLCR?**

Get AT over layer 2 configuration

**Syntax:**

`AT*AMLCR?`

**Example:**

`AT*AMLCR?<br>*AMLCR:<eth_type>,<enabled>`

### 3.8.16 AT*AMREBOOT Reboot

**AT*AMREBOOT**

Reboot device

**Syntax:**

`AT*AMREBOOT`

### 3.8.17 AT*AMPSM ConfigLock Mode

Enable ConfigLock Mode, in ConfigLock mode it's only possible to read configuration. To write configuration again, it's necessary to reset to factory defaults with the physical button.

**AT*AMPSM=**

Enable/Disable ConfigLock mode.

**Syntax:**

`AT*AMPSM=<config_lock_mode>,<apply_now>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| config_lock_mode | Integer | Set state of ConfigLock mode.<br>0: Disable.<br>1: Enable. |
| apply_now | Boolean | 0: Apply changes after reboot.<br>1: Apply changes immediately. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`Input: AT*AMPSM=1,1,1<br>Output:<br>*AMPSM:OK`

**AT*AMPSM?**

Read state of ConfigLock mode.

**Syntax:**

`AT*AMPSM?`

**Example:**

`Input: AT*AMPSM?<br>Output:<br>*AMPSM:<config_lock_mode>`

### 3.8.18 AT*AMIC Interface configuration

**AT*AMIC?**

Command to read back interface configuration bit mask

**Syntax:**

`AT*AMIC?`

**Example:**

`AT*AMIC?<br>*AMIC:<iface_config>`

### 3.8.19 AT*AMSERIAL Serial Number

**AT*AMSERIAL?**

Command to read back serial number

**Syntax:**

```
AT*AMSERIAL?
```

**Example:**

```
AT*AMSERIAL?<br>*AMSERIAL:<serial>
```

### 3.8.20 AT*AMDEFAULT Read DEFAULT

**AT*AMDEFAULT?**

Get the DEFAULT VALUES.

**Syntax:**

```
AT*AMDEFAULT?
```

**Example:**

```
AT*AMDEFAULT?<br>*AMBNAME: <name><br>*AMECFL: <easy_config_
modes><br>...<br>OK
```

### 3.8.21 AT*AMLOG Read event log

**AT*AMLOG=**

Command to clear current event log.

**Syntax:**

```
AT*AMLOG=<clear>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| clear | Boolean | Set this to 1 to clear the event log. |

**AT*AMLOG?**

Read the event log.

**Syntax:**

```
AT*AMLOG?
```

**Example:**

```
Input: AT*AMLOG?<br>...<br>OK
```

# 3.9 Bluetooth Low Energy Commands

## 3.9.1 AT*BLEAD Get/Set the advertise data

**AT*BLEAD=**

Set the data to be advertised by this device.

**Syntax:**

```
AT*BLEAD=<data>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| data | String | The new data to be advertised as a HEX string with max 28 bytes. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*BLEAD=020A04051218002800,1<br>OK
```

**AT*BLEAD?**

Get the data currently being advertised by this device.

**Syntax:**

```
AT*BLEAD?
```

**Example:**

```
AT*BLEAD?<br>*BLEAD:<data>
```

### 3.9.2 AT*BLESRD Get/Set the scan response data

**AT*BLESRD=**

Set the data to be responded by this device.

**Syntax:**

AT*BLESRD=<data>,<store>

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| data | String | The new data to be responded as a HEX string with max 31 bytes. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

AT*BLESRD=0B096177625F316533623135,1<br>OK

**AT*BLESRD?**

Get the data currently being responded by this device.

**Syntax:**

AT*BLESRD?

**Example:**

AT*BLESRD?<br>*BLESRD:<data><br>OK

### 3.9.3 AT*BLEOM Get/Set the BLE operating mode

**AT*BLEOM=**

Set the BLE operating mode.

**Syntax:**

```
AT*BLEOM=<mode>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The BLE operating mode:<br>0: BLE off.<br>1: Central operating mode.<br>2: Peripheral operating mode, requires that the wlan interface is disabled. |
| store | Boolean | If store is 1 the new value is stored permanently. |

*Requires a reboot for the changes to take effect.*

**Example:**

```
AT*BLEOM=1,1<br>OK
```

**AT*BLEOM?**

Get the current BLE operating mode.

**Syntax:**

```
AT*BLEOM?
```

**Example:**

```
AT*BLEOM?<br>*BLEOM:<mode><br>OK
```

### 3.9.4 AT*BLEND BLE name discovery

**AT*BLEND=**

Discover the name of a Bluetooth device.

**Syntax:**

`AT*BLEND=<bd_addr>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | String | Bluetooth address of the device, on the format "XX-XX-XX-XX-XX-XXy" where "y" is "p" or "r" depending on if the address is private or random. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| devicename | String | Devicename of found BLE device. The value is limited to a maximum of 19 bytes. |

**Example:**

`AT*BLEND=<bd_addr><br>*BLEND:<devicename><br>OK`

### 3.9.5 AT*BLERM Enter/Exit RAW BLE mode

**AT*BLERM=**

Enter/Exit RAW BLE mode. Note that this only affects the current AT session. When the AT session is in RAW BLE mode, it will receive unsolicited BLE events. The following AT sessions are available: * RAW Ethernet * JSON/Web interface * Each TCP connection is its own session. Note: The RAW BLE mode may not be entered from the JSON/Web interface.

**Syntax:**

`AT*BLERM=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Boolean | Enter/Exit RAW BLE mode.<br>0: Exit RAW BLE mode.<br>1: Enter RAW BLE mode. |

**Example:**

`AT*BLERM=<mode><br>OK`

**AT*BLERM?**

Get the current RAW BLE mode.

**Syntax:**

`AT*BLERM?`

**Example:**

`AT*BLERM?<br>*BLERM:<mode><br>OK`

### 3.9.6 AT*BLEDD BLE device discovery

Note: Only available when the device is in central operating mode.

**AT*BLEDD=**

Discover BLE devices. If a passive discovery is performed the devices advertisement data will be included. If an active scan is performed the devices advertisement and scan response data will be included.

**Syntax:**

```
AT*BLEDD=<type>,<length>,<scan>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| type | Integer | Type of discovery.<br>0:.Discover all devices, but only display each device once.<br>1: Discover devices in general or limited discoverability mode.<br>2: Discover devices in limited discoverability mode.<br>3: Discover all devices, each device may be displayed multiple times. |
| length | Integer | Length of discovery in milliseconds. Max 65535 ms. |
| scan | Integer | Type of scan.<br>0: Active scan.<br>1: Passive scan. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | String | Bluetooth address of the BLE device. |
| rssi | Integer | RSSI. |
| name | String | Complete local name of BLE device, if included in data. |
| data_type | Integer | 0: Scan response data<br>1: Advertise data |
| data | String | Advertise/Scan response data as a HEX string. |

**Example:**

```
AT*BLEDD=<type>,<length>,<scan><br>*BLEDD:<bd_
addr>,<rssi>,<name>,<data_type>,<data><br>OK
```

### 3.9.7 AT*BLEC Connect to BLE device

Note: Only available when the device is in central operating mode.

**AT*BLEC=**

Initiate an attempt to connect to a BLE device. The result will come in an unsolicited *BLEC event.

**Syntax:**

```
AT*BLEC=<bd_addr>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | String | Address of device to connect to, on the format "XX-XX-XX-XX-XX-XXy" where "y" is "p" or "r" depending on if the address is private or random. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. ffff if the connection fails. |
| bd_addr | String | Bluetooth address of the device connecting to. |

**Example:**

```
AT*BLEC=<bd_addr><br>OK<br>*BLEC:<con_handle>,<bd_addr>
```

### 3.9.8 AT*BLED Disconnect from BLE device

Note: Only available when the device is in central operating mode.

**AT*BLED=**

Initiate an attempt to disconnect from a BLE device. The result will come in an unsolicited *BLED event.

**Syntax:**

```
AT*BLED=<con_handle>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |

**Example:**

```
AT*BLED=<con_handle><br>OK<br>*BLED:<con_handle>
```

### 3.9.9 AT*BLERCD Bluetooth LE Read Connected Devices.

**AT*BLERCD?**

Retrieves the MAC address and handle of every connected Bluetooth LE device.

**Syntax:**

`AT*BLERCD?`

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| bd_addr | String | Bluetooth address of the device connected to. |
| con_handle | Integer | Hexadecimal formatted connection handle. |

**Example:**

`AT*BLERCD? returns *BLERCD:<bd_addr>,<con_handle> for each connected device followed by OK or ERROR.`

# 3.10 Bluetooth Low Energy GATT Client Commands

## 3.10.1 AT*BGCPSD Discover All Primary Services

Note: Only available when the device is in central operating mode.

**AT*BGCPSD=**

Discover all primary services of a GATT server.

**Syntax:**

`AT*BGCPSD=<con_handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Decimal formatted attribute handle. |
| start_group_handle | Integer | Decimal formatted start group handle. |
| end_group_handle | Integer | Decimal formatted end group handle. |
| uuid | Integer | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |

**Example:**

```
AT*BGCPSD=<con_handle><br>*BGCPSD:<attr_handle>,<start_group_
handle>,<end_group_handle>,<uuid><br>OK
```

### 3.10.2 AT*BGCPSDU Discover All Primary Services By Service UUID

Note: Only available when the device is in central operating mode.

**AT*BGCPSDU=**

Discover primary services by UUID. This will filter out the results based on UUID.

**Syntax:**

`AT*BGCPSDU=<con_handle>,<uuid>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| uuid | String | UUID to search for. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Decimal formatted attribute handle. |
| start_group_handle | Integer | Decimal formatted start group handle. |
| end_group_handle | Integer | Decimal formatted end group handle. |

**Example:**

`AT*BGCPSDU=<con_handle>,<uuid><br>*BGCPSDU:<attr_handle>,<start_group_handle>,<end_group_handle><br>OK`

### 3.10.3 AT*BGCFIS Find Included Services

Note: Only available when the device is in central operating mode.

**AT*BGCFIS=**

Find included services of a given service.

**Syntax:**

```
AT*BGCFIS=<con_handle>,<start_handle>,<end_handle>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| start_handle | Integer | Start handle of the service. |
| end_handle | Integer | End handle of the service. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Decimal formatted attribute handle. |
| start_group_handle | Integer | Decimal formatted start group handle. |
| end_group_handle | Integer | Decimal formatted end group handle. |
| uuid | Integer | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |

**Example:**

```
AT*BGCFIS=<con_handle>,<start_handle>,<end_handle><br>*BGCFIS:
<attr_handle>,<start_group_handle>,<end_group_handle>,<uuid><br>OK
```

### 3.10.4 AT*BGCDCS Discover All Characteristic of a Service

Note: Only available when the device is in central operating mode.

**AT*BGCDCS=**

Discover all characteristics of a service.

**Syntax:**

```
AT*BGCDCS=<con_handle>,<start_handle>,<end_handle>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| start_handle | Integer | Start handle of the service. |
| end_handle | Integer | End handle of the service. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Decimal formatted attribute handle. |
| properties | Integer | Hexadecimal formatted properties. The individual bits indicate a specific property:<br>* Bit 0: Broadcast<br>* Bit 1: Readable<br>* Bit 2: Writable with no response<br>* Bit 3: Writable<br>* Bit 4: Notify<br>* Bit 5: Indicate<br>* Bit 6: Authenticated signed write<br>* Bit 7: Extended property available |
| value_handle | Integer | Decimal formatted value handle. |
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 001122334455667788 99aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |

**Example:**

```
AT*BGCDCS=<con_handle>,<start_handle>,<end_handle><br>*BGCDCS:
<attr_handle>,<properties>,<value_handle>,<uuid><br>OK
```

### 3.10.5 AT*BGCDCD Discover All Characteristic Descriptors

Note: Only available when the device is in central operating mode.

**AT*BGCDCD=**

Discover all descriptors of a characteristic.

**Syntax:**

```
AT*BGCDCD=<con_handle>,<value_handle>,<service_end_handle>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| value_handle | Integer | Handle of the characteristic value. |
| service_end_handle | Integer | End handle of the service which the characteristic belongs to. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| char_attr_handle | Integer | Decimal formatted handle to the characteristic. |
| attr_handle | Integer | Decimal formatted attribute handle. |
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |

**Example:**

```
AT*BGCDCD=<con_handle>,<value_handle>,<end_handle><br>*BGCDCD:
<char_attr_handle>,<attr_handle>,<uuid><br>OK
```

### 3.10.6 AT*BGCRC Read Characteristic Value, Read Characteristic Descriptors

Note: Only available when the device is in central operating mode.

**AT*BGCRC=**

Read the value of a characteristic or descriptor.

**Syntax:**

`AT*BGCRC=<con_handle>,<attr_handle>,<offset>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle | Integer | Handle to the characteristic or descriptor value. |
| offset | Integer | Offset where to start read from. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| data | String | Data formatted as a hexadecimal string. |

**Example:**

```
AT*BGCRC=<con_handle>,<attr_handle>,<offset><br>*BGCRC:
<data><br>OK
```

### 3.10.7 AT*BGCRCU Read Using Characteristic UUID

Note: Only available when the device is in central operating mode.

**AT*BGCRCU=**

Read the value of a characteristic by UUID. ERROR will be returned if no characteristic with the specified UUID could be found in the handle range.

**Syntax:**

```
AT*BGCRCU=<con_handle>,<start_handle>,<end_handle>,<uuid>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| start_handle | Integer | Handle, where to start looking for the UUID. |
| end_handle | Integer | Handle, where to stop looking for the UUID. |
| uuid | String | UUID to look for. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the attribute read. |
| data | String | Data formatted as a hexadecimal string. |

**Example:**

```
AT*BGCRCU=<con_handle>,<start_handle>,<end_
handle>,<uuid><br>*BGCRCU:<attr_handle>,<data><br>OK
```

### 3.10.8 AT*BGCRCM Read Multiple Characteristic Values

Note: Only available when the device is in central operating mode.

**AT*BGCRCM=**

Read multiple characteristics in a single read. The application must know the length of each data element in the returned list. Therefore only the last data element may have a variable length.

**Syntax:**

`AT*BGCRCM=<con_handle>,<attr_handle_list>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle_list | String | List of characteristic handles for which values to read, as a string where each handle is four hexadecimal digits.<br>Ex: "00010003000A" would read characteristics with handles 0x0001, 0x0003 and 0x000A. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| data | String | Data formatted as a hexadecimal string. |

**Example:**

`AT*BGCRCM=<con_handle>,<attr_handle_list><br>*BGCRCM:<data><br>OK`

### 3.10.9 AT*BGCWC Write Characteristic Value, Write Characteristic Descriptors

Note: Only available when the device is in central operating mode.

**AT*BGCWC=**

Write a value to a characteristic or descriptor.

**Syntax:**

`AT*BGCWC=<con_handle>,<attr_handle>,<data>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle | Integer | Handle to the characteristic or descriptor value. |
| data | String | Data to write as a HEX string with max 20 bytes. |

**Example:**

`AT*BGCWC=<con_handle>,<attr_handle>,<data><br>OK`

### 3.10.10 AT*BGCWCN Write Without Response

Note: Only available when the device is in central operating mode.

**AT*BGCWCN=**

Write a value to a characteristic without any response from the remote side.

**Syntax:**

```
AT*BGCWCN=<con_handle>,<attr_handle>,<data>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle | Integer | Handle to the characteristic value. |
| data | String | Data to write as a HEX string with max 20 bytes. |

**Example:**

```
AT*BGCWCN=<con_handle>,<attr_handle>,<data><br>OK
```

### 3.10.11 AT*BGCWCL Write Long Characteristic Values, Write Long Characteristic Descriptors

Note: Only available when the device is in central operating mode.

**AT*BGCWCL=**

Write long characteristic or descriptor value.

**Syntax:**

```
AT*BGCWCL=<con_handle>,<attr_
handle>,<data>,<reliable>,<flag>,<offset>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle | Integer | Handle to the characteristic or descriptor value. |
| data | String | Data to write as a HEX string with max 18 bytes. |
| reliable | Boolean | Whether or not to verify the data written.<br>0: Do not verify.<br>1: Verify. |
| flag | Integer | All but the last writes to a long value should have the flag set to 1. The last write should have the flag set to 0. Set the flag to 2 to abort the write. |
| offset | Integer | Offset of the data to write. |

**Example:**

```
AT*BGCWCL=<con_handle>,<attr_
handle>,<data>,<reliable>,<flag>,<offset><br>OK
```

# 3.11 Bluetooth Low Energy GATT Server Commands

## 3.11.1 AT*BGSDS Define service

**AT*BGSDS=**

Define a primary or secondary service.

**Syntax:**

```
AT*BGSDS=<type>,<uuid>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| type | Integer | Type of service.<br>0: Primary.<br>1: Secondary. |
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the created service. |

**Example:**

```
AT*BGSDS=<type>,<uuid><br>*BGSDS:<attr_handle><br>OK
```

## 3.11.2 AT*BGSDI Include service

**AT*BGSDI=**

Include service to the last defined service, executed before adding any characteristic to the service.

**Syntax:**

```
AT*BGSDI=<service_attr_handle>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| service_attr_handle | Integer | Handle to the service to include. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the created include declaration. |

**Example:**

```
AT*BGSDI=<service_attr_handle><br>*BGSDI:<attr_handle><br>OK
```

### 3.11.3 AT*BGSDC Define characteristic

**AT*BGSDC=**

Define a characteristic to the last defined service.

**Syntax:**

```
AT*BGSDC=<uuid>,<properties>,<read_security>,<write_
security>,<value>,<max_length>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |
| properties | Integer | Characteristic properties. The individual bits indicate a specific property:<br>Bit 0: Broadcast<br>Bit 1: Readable<br>Bit 2: Writable with no response<br>Bit 3: Writable<br>Bit 4: Notify<br>Bit 5: Indicate<br>Bit 6: Authenticated signed write<br>Bit 7: Reliable write |
| read_security | Integer | Characteristic read security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |
| write_security | Integer | Characteristic write security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |
| value | DataString | Default value as a HEX string with max 512 bytes. |
| max_length | Integer | Maximum length for the characteristic value. Must be at least the length of `<value>`. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| value_handle | Integer | Handle to the created characteristic value. |

**Example:**

```
AT*BGSDC=<uuid>,<properties>,<read_security>,<write_
security>,<value>,<max_length><br>*BGSDC:<value_handle><br>OK
```

### 3.11.4 AT*BGSDD Define descriptor

**AT*BGSDD=**

Define a descriptor to the last defined characteristic.

**Syntax:**

```
AT*BGSDD=<uuid>,<properties>,<read_security>,<write_
security>,<value>,<max_length>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |
| properties | Integer | Descriptor properties. The individual bits indicate a specific property:<br>Bit 0: Broadcast<br>Bit 1: Readable<br>Bit 2: Writable with no response<br>Bit 3: Writable<br>Bit 4: Notify<br>Bit 5: Indicate<br>Bit 6: Authenticated signed write<br>Bit 7: Reliable write |
| read_security | Integer | Descriptor read security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |
| write_security | Integer | Descriptor write security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |
| value | DataString | Default value as a HEX string with max 512 bytes. |
| max_length | Integer | Maximum length for the descriptor. Must be at least the length of `<value>`. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the created descriptor. |

**Example:**

```
AT*BGSDD=<uuid>,<properties>,<read_security>,<write_
security>,<value>,<max_length><br>*BGSDD:<attr_handle><br>OK
```

### 3.11.5 AT*BGSLD List defined attributes

**AT*BGSLD?**

List all defined services, characteristics and descriptors.

**Syntax:**

```
AT*BGSLD?
```

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the attribute. |
| type | Integer | Type of attribute.<br>0: Service.<br>1: Characteristic<br>2: Descriptor. |
| uuid | String | 128-bit UUID on the format 00112233-4455-6677-8899-aabbccddeeff, 00112233445566778899aabbccddeeff or 16-bit UUID on hexadecimal format 0011. |
| properties | Integer | Characteristic properties. The individual bits indicate a specific property:<br>Bit 0: Broadcast<br>Bit 1: Readable<br>Bit 2: Writable with no response<br>Bit 3: Writable<br>Bit 4: Notify<br>Bit 5: Indicate<br>Bit 6: Authenticated signed write<br>Bit 7: Reliable write |
| read_security | Integer | Characteristic read security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |
| write_security | Integer | Characteristic write security mode.<br>0: No encryption required.<br>1: Unauthenticated encryption required.<br>2: Authenticated encryption required. |

**Example:**

```
AT*BGSLD?<br>*BGSLD:<attr_
handle>,<type>,<uuid>,<properties>,<read_security>,<write_
security><br>...<br>OK
```

### 3.11.6 AT*BGSCL Clear attribute table

**AT*BGSCL**

Delete all defined services, characteristics and descriptors. Requires reboot for the changes to take effect.

**Syntax:**

```
AT*BGSCL
```

**Example:**

```
AT*BGSCL<br>OK
```

### 3.11.7 AT*BGSWC Write attribute value

**AT*BGSWC=**

Set the value of a user defined characteristic or descriptor. Will not trigger notifications or indications.

**Syntax:**

```
AT*BGSWC=<attr_handle>,<value>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the attribute. |
| value | DataString | New value as a HEX string. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*BGSWC=<attr_handle>,<value>,<store><br>OK
```

### 3.11.8 AT*BGSRC Read attribute value

**AT*BGSRC=**

Get the value of a user defined characteristic or descriptor.

**Syntax:**

`AT*BGSRC=<attr_handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| attr_handle | Integer | Handle to the attribute. |

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| data | String | Current value as a HEX string. |

**Example:**

`AT*BGSRC=<attr_handle><br>*BGSRC:<data><br>OK`

### 3.11.9 AT*BGSWR Reply to write request

Note: Only available when the device is in peripheral operating mode.

**AT*BGSWR=**

Reply to an unsolicited write request event. If no reply is sent within 20 seconds of receiving the write request event "Unlikely Error" will be sent.

**Syntax:**

`AT*BGSWR=<con_handle>,<attr_handle>,<status>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| attr_handle | Integer | Handle to the attribute. |
| status | Integer | One of the following status values:<br>0: OK<br>1: INVALID_HANDLE<br>2: READ_NOT_PERMITTED<br>3: WRITE_NOT_PERMITTED<br>4: INVALID_PDU<br>5: INSUFFICIENT_AUTHENTICATION<br>6: REQUEST_NOT_SUPPORTED<br>7: INVALID_OFFSET<br>8: INSUFFICIENT_AUTHORIZATION<br>9: PREPARE_FULL_QUEUE<br>10: ATTRIBUTE_NOT_FOUND<br>11: ATTRIBUTE_NOT_LONG<br>12: INSUFFICIENT_ENCRYPT_KEY_SIZE<br>13: INVALID_ATTRIBUTE_VALUE_LENGTH<br>14: UNLIKELY_ERROR<br>15: INSUFFICIENT_ENCRYPTION<br>16: UNSUPPORTED_GROUP_TYPE<br>17: INSUFFICIENT_RESOURCES |

**Example:**

`AT*BGSWR=<con_handle>,<attr_handle>,<status><br>OK`

### 3.11.10 AT*BGSNOT Send notification

Note: Only available when the device is in peripheral operating mode.

**AT*BGSNOT=**

Send a notification to a client. The value in the notification is the 20 first bytes in the characteristic with handle <char_handle>.

**Syntax:**

`AT*BGSNOT=<con_handle>,<char_handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| char_handle | Integer | Handle to the characteristic. |

**Example:**

`AT*BGSNOT=<con_handle>,<char_handle><br>OK`

### 3.11.11 AT*BGSIND Send indication

Note: Only available when the device is in peripheral operating mode.

**AT*BGSIND=**

Send an indication to a client. The value in the indication is the 20 first bytes in the characteristic with handle <char_handle>.

**Syntax:**

`AT*BGSIND=<con_handle>,<char_handle>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Connection handle. |
| char_handle | Integer | Handle to the characteristic. |

**Example:**

`AT*BGSIND=<con_handle>,<char_handle><br>OK`

## 3.12 Unsolicited events

In order to receive unsolicited events the AT session needs to be put into a RAW BLE mode. This is done with the AT*BLERM command.

### 3.12.1 AT*BLEC Device connected

Event Device connected, for Central or Peripheral Bluetooth LE mode.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. ffff if the connection fails. |
| bd_addr | String | Bluetooth address of the device connecting to. |

**Example:**

```
Event: *BLEC:<con_handle>,<bd_addr>
```

### 3.12.2 AT*BLED Device disconnected

Event Device disconnected, for Central or Peripheral Bluetooth LE mode.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |

**Example:**

```
Event: *BLED:<con_handle>
```

### 3.12.3 AT*BLENR Notification received

Event Notification received, for Central Bluetooth LE mode.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |
| attr_handle | Integer | Handle to the attribute. |
| value | String | Value as a HEX string. |

**Example:**

```
Event: *BLENR:<con_handle>,<attr_handle>,<value>
```

### 3.12.4 AT*BLEIR Indication received

Event Indication received, for Central Bluetooth LE mode.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |
| attr_handle | Integer | Handle to the attribute. |
| value | String | Value as a HEX string. |

**Example:**

```
Event: *BLEIR:<con_handle>,<attr_handle>,<value>
```

### 3.12.5 AT*BGSWR Write request received

Event Write request received, if no reply is sent (using the AT*BGSWR command) within 20 seconds error code Unlikely Error will be sent instead. For Peripheral Bluetooth LE mode. Note: If ERROR is returned, for example due to bad parameters, it is possible to send AT*BGSWR again within the 20 second response window.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |
| attr_handle | Integer | Handle to the attribute. |
| offset | String | Offset of the data to write. |
| value | String | Value as a HEX string. |

**Example:**

```
Event: *BGSWR:<con_handle>,<attr_handle>,<offset>,<value>
```

### 3.12.6 AT*BGSWOR Write without response received

A write without response has been received. For Peripheral Bluetooth LE mode. NOTE: A write without response will always update the characteristic value, no validation is possible. If it is necessary to validate the value before it is updated, do not set the write_wo_rsp property of the characteristic.

**Output Parameters:**

| Name | Type | Description |
|---|---|---|
| con_handle | Integer | Hexadecimal formatted connection handle. |
| attr_handle | Integer | Handle to the attribute. |
| offset | String | Offset of the data to write. |
| value | String | Value as a HEX string. |

**Example:**

```
Event: *BGSWOR:<con_handle>,<attr_handle>,<offset>,<value>
```

## 3.13 Serial Commands

These commands are specific to Bolt Serial products. To configure where the serial data is sent/ received from, see section Serial Data Tunnel.

### 3.13.1 AT*SOM Operating mode

**AT*SOM=**

Set the Serial operating mode. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*SOM=<mode>,<store>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The Serial operating mode:<br>0: Disabled.<br>1: RS232.<br>2: RS485. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

`AT*SOM=1,1<br>OK`

**AT*SOM?**

Get the current Serial operating mode.

**Syntax:**

`AT*SOM?`

**Example:**

`AT*SOM?<br>*SOM:<mode><br>OK`

### 3.13.2 AT*SPS Port settings

**AT*SPS=**

Set the Serial port settings. For default value see AT*AMDEFAULT.

**Syntax:**

```
AT*SPS=<baud_rate>,<data_bits>,<stop_bits>,<parity>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| baud_rate | Integer | The baud rate expressed in bits per second (bps). A value over 2400 will be accepted if the adapter is able to generate the baudrate with less than 2% error. Otherwise it be rejected and an ERROR will be returned. |
| data_bits | Integer | The data length in bits. Valid values are 7 and 8. Note that the data length of 7 bits cannot be used without parity. |
| stop_bits | Integer | 1: 1 Stop bit<br>2: 2 Stop bits |
| parity | Integer | 0: No parity<br>1: Odd parity<br>2: Even parity |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*SPS=57600,8,1,0,1<br>OK
```

**AT*SPS?**

Get the current Serial port settings.

**Syntax:**

```
AT*SPS?
```

**Example:**

```
AT*SPS?<br>*SPS:<baud_rate>,<data_bits>,<stop_bits>,<parity><br>OK
```

### 3.13.3 AT*SDM Serial Data Mode

**AT*SDM=**

Switch the Serial communication between AT Mode and Data Mode. In AT Mode, the data received from the serial port are interpreted as AT commands whereas in Data Mode, the data is directly sent over the air. After a successful response, the Serial driver will leave the AT Mode and enter the Data Mode or vice versa. Note: In order to switch from the Data Mode to AT Mode from the serial port, an escape sequence consisting of three forward slashes (configurable using S register 6) preceeded and followed by 1 second of no data activity (configurable using S register 7000) can be sent within a maximum 200ms timeframe.

**Syntax:**

```
AT*SDM=<mode>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | 0: Disabled (AT Mode)<br>1: Enabled (Data Mode) |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*SDM=1,1<br>OK
```

**AT*SDM?**

Get the current Serial data mode.

**Syntax:**

```
AT*SDM?
```

**Example:**

```
AT*SDM?<br>*SDM:<mode><br>OK
```

## 3.14 Serial Data Tunnel Commands

These commands are specific to Bolt Serial/CAN products. Use these commands to configure the Ethernet endpoint to bridge the RS232/485 or CAN interface with.

### 3.14.1 AT*STM Serial TCP Mode

Configures the serial communication TCP mode (for RS232/485 or CAN data).

**AT*STM=**

Set serial TCP mode.

**Syntax:**

`AT*STM=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | 1: TCP Client<br>2: TCP Server |

*Requires a reboot for the changes to take effect.*

**Example:**

`AT*STM=2<br>OK`

**AT*STM?**

Get serial TCP mode.

**Syntax:**

`AT*STM?`

**Example:**

`AT*STM?<br>*STM:<mode>`

### 3.14.2 AT*SSP Serial Server Port

Configures the TCP port to use for serial communication in server mode.

**AT*SSP=**

Set serial server TCP port

**Syntax:**

`AT*SSP=<port>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| port | Integer | TCP port number to use for serial connection. |

*Requires a reboot for the changes to take effect.*

**Example:**

`AT*SSP=5100<br>OK`

**AT*SSP?**

Get serial server TCP port

**Syntax:**

`AT*SSP?`

**Example:**

`AT*SSP?<br>*SSP:<port>`

### 3.14.3 AT*SCIP Serial Connection IP and Port

Configures the IP and TCP port to use for serial communication in client mode.

**AT*SCIP=**

Set the serial connection IP and port.

**Syntax:**

`AT*SCIP=<ip>,<port>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| ip | NetworkAddress | The IP address to connect to. |
| port | Integer | TCP port number to use for serial connection. |

*Requires a reboot for the changes to take effect.*

**AT*SCIP?**

Get the serial connection IP and port.

**Syntax:**

`AT*SCIP?`

**Example:**

`AT*SCIP?<br>*SCIP:<ip>,<port>`

### 3.14.4 AT*SMGM Serial Modbus Gateway Mode

Configures the serial Modbus gateway mode. The Modbus gateway feature is specific to Bolt Serial products.

**AT*SMGM=**

Set the serial Modbus gateway mode.

**Syntax:**

`AT*SMGM=<mode>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | 0: Disabled<br>1: RTU Gateway |

*Requires a reboot for the changes to take effect.*

**AT*SMGM?**

Get the serial Modbus gateway mode.

**Syntax:**

`AT*SMGM?`

**Example:**

`AT*SMGM?<br>*SMGM:<mode>`

### 3.14.5 AT*SMGP Serial Modbus Gateway Port

Configures the TCP port to use for serial Modbus gateway communication.

**AT*SMGP=**

Set serial Modbus gateway TCP port.

**Syntax:**

`AT*SMGP=<port>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| port | Integer | TCP port number to use for serial connection. |

*Requires a reboot for the changes to take effect.*

**Example:**

`AT*SMGP=502<br>OK`

**AT*SMGP?**

Get serial Modbus gateway TCP port.

**Syntax:**

`AT*SMGP?`

**Example:**

`AT*SMGP?<br>*SMGP:<port>`

## 3.15 CAN Commands

These commands are specific to Bolt CAN products. To configure where CAN data is sent/ received from, see section Serial Data Tunnel.

### 3.15.1 AT*COM CAN Operating Mode

**AT*COM=**

Set the CAN operating mode. For default value see AT*AMDEFAULT.

**Syntax:**

```
AT*COM=<mode>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| mode | Integer | The CAN operating mode:<br>0: Disabled.<br>1: Enabled. |

*Requires a reboot for the changes to take effect.*

**Example:**

```
AT*COM=1<br>OK
```

**AT*COM?**

Get the current CAN operating mode.

**Syntax:**

```
AT*COM?
```

**Example:**

```
AT*COM?<br>*COM:<mode><br>OK
```

### 3.15.2 AT*CBR CAN Bitrate

**AT*CBR=**

Set the CAN bitrate. For default value see AT*AMDEFAULT.

**Syntax:**

`AT*CBR=<bitrate>`

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| bitrate | Integer | The CAN bitrate in bits per second. |

*Requires a reboot for the changes to take effect.*

**Example:**

`AT*CBR=250000<br>OK`

**AT*CBR?**

Get the current CAN bitrate.

**Syntax:**

`AT*CBR?`

**Example:**

`AT*CBR?<br>*CBR:<bitrate><br>OK`

### 3.15.3 AT*CBRP CAN Bitrate Parameters

**AT*CBRP=**

Set the CAN bitrate parameters.

**Syntax:**

```
AT*CBRP=<prescaler>,<seg1>,<seg2>,<sjw>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| prescaler | Integer | The internal 42MHz peripheral clock will be divided by this number to generate the CAN clock. |
| seg1 | Integer | Number of CAN clock cycles in time segment 1 (propagation segment + phase segment 1). |
| seg2 | Integer | Number of CAN clock cycles in time segment 2 (phase segment 2). |
| sjw | Integer | The CAN sync jump width, i.e. number of CAN clock cycles seg1 and seg2 are allowed to increase or decrease during bit synchronization. |

*Requires a reboot for the changes to take effect.*

**Example:**

```
AT*CBRP=12,11,2,1<br>OK
```

**AT*CBRP?**

Get the current CAN bitrate parameters.

**Syntax:**

```
AT*CBRP?
```

**Example:**

```
AT*CBRP?<br>*CBRP:<prescaler>,<seg1>,<seg2>,<sjw><br>OK
```

### 3.15.4 AT*CFILT CAN RX Filter configuration

**AT*CFILT=**

Reconfigure a CAN filter. For default values see AT*AMDEFAULT.

**Syntax:**

```
AT*CFILT=<index>,<type>,<id>,<mask>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| index | Integer | The filter index to write (0-27). |
| type | Integer | The type of CAN frames this filter will match:<br>0: Disabled.<br>1: Standard frames.<br>2: Extended frames. |
| id | Integer | The CAN frame ID this filter will match. |
| mask | Integer | The mask used when matching CAN frame ID to this filter. |

*Requires a reboot for the changes to take effect.*

**Example:**

```
AT*CFILT=0,1,0x123,0x7FF<br>OK
```

**AT*CFILT?**

List the currently active CAN filters.

**Syntax:**

```
AT*CFILT?
```

**Example:**

```
AT*CFILT?<br>*CFILT:<index>,<type>,<id>,<mask><br>*CFILT:
<index>,<type>,<id>,<mask><br>OK
```

### 3.15.5 AT*CFILTCL Clear CAN RX Filter configuration

**AT*CFILTCL**

Delete all CAN filters. No CAN frames will be possible to receive until new filters are configured.

**Syntax:**

```
AT*CFILTCL
```

**Example:**

```
AT*CFILTCL<br>OK
```

### 3.15.6 AT*CEP CAN Ethernet Protocol

**AT*CEP=**

Set the CAN Ethernet protocol. For default value see AT*AMDEFAULT.

**Syntax:**

AT*CEP=<protocol>,<store>

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| protocol | Integer | The CAN Ethernet protocol:<br>0: Optimized.<br>1: SLCan.<br>2: Simple. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

AT*CEP=0,1<br>OK

**AT*CEP?**

Get the current CAN Ethernet protocol.

**Syntax:**

AT*CEP?

**Example:**

AT*CEP?<br>*CEP:<protocol><br>OK

### 3.15.7 AT*CEPSI CAN Ethernet Protocol Simple ID

**AT*CEPSI=**

Set the CAN frame ID used by the Simple protocol. For default values see AT*AMDEFAULT.

**Syntax:**

```
AT*CEPSI=<extended>,<id>,<store>
```

**Input Parameters:**

| Name | Type | Description |
|---|---|---|
| extended | Boolean | 0: Standard CAN frame.<br>1: Extended CAN frame. |
| id | Integer | The CAN frame ID. |
| store | Boolean | If store is 1 the new value is stored permanently. |

**Example:**

```
AT*CEPSI=0,0x123,1<br>OK
```

**AT*CEPSI?**

Get the current CAN frame ID used by the Simple protocol.

**Syntax:**

```
AT*CEPSI?
```

**Example:**

```
AT*CEPSI?<br>*CEPSI:<extended>,<id><br>OK
```

### 3.15.8 AT*CSTAT CAN Statistics

**AT*CSTAT?**

Reads the current statistics of the CAN driver.

**Syntax:**

```
AT*CSTAT?
```

**Example:**

```
AT*CSTAT?<br>*CSTAT: TCP frames (TX/RX): 0/0<br>*CSTAT: TCP bytes (TX/
RX): 0/0<br>*CSTAT: Delivered frames (RX/TX): 0/0<br>*CSTAT: Dropped
frames (RX/TX): 0/0<br>*CSTAT: RX buffer max usage/trigger/capacity:
0/320/400<br>*CSTAT: TX buffer max usage/capacity: 0/400<br>*CSTAT:
Error count: 0<br>*CSTAT: Last error code: 0x0<br>OK
```

# 4 S Registers

## 4.1 ATS S Registers

Changes made to S registers requires reboot

**Standard Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 3 | Command Line Termination Character | 1..127 | 13 | This setting changes the decimal value of the character recognized by the DCE from the DTE to terminate an incoming command line. It is also generated by the DCE as part of the header, trailer, and terminator for result codes and information text along with the S4 parameter. The previous value of S3 is used to determine the command line termination character for entry of the command line containing the S3 setting command. However, the result code issued shall use the value of S3 as set during the processing of the command line. For example, if S3 was previously set to 13 and the command line "ATS3=30" is issued, the command line shall be terminated with a CR, character (13), but the result code issued will use the character with the ordinal value 30 in place of the CR. |
| 4 | Response Formatting Character | 1..127 | 10 | This setting changes the decimal value of the character generated by the DCE as part of the header, trailer, and terminator for result codes and information text, along with the S3 parameter. If the value of S4 is changed in a command line, the result code issued in response to that command line will use the new value of S4. |
| 5 | Backspace Character | 0..255 | 8 | This setting changes the decimal value of the character recognized by the DCE as a request to delete from the command line the immediately preceding character. |
| 6 | Escape Character | 0..255 | 47 | This setting changes the decimal value of the character recognized by the DCE as part of a sequence to escape from the Data Mode to the AT Mode. The sequence consists of 3 escape characters preceded and followed by a period of no data activity which by default is 1 second (configurable using S register 7000). The sequence must be received within a 200ms window. Note that the escape sequence is only relevant when using the serial port. |

**Miscellaneous Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 1000 | Reserved | | | |
| 1001 | Reserved | | | |
| 1002 | Ignore Broadcast Layer 2 AT | 0..1 | 0 | Setting this register to 1 will ignore Layer2 AT broadcast packets |
| 1003 | Event Subscriber Protocol | 0..255 | 0 | Bit mask deciding how events should be sent.<br>Bit 0: Send events over TCP AT connections<br>Bit 1: Send events over Layer-2 (mac_address must be specified, using AT*AMESS)<br>Bit 2: Send events over Syslog<br>Bit 3: Send events over Serial port (in AT Mode) |
| 1004 | Event Subscriber Ethernet Type | 0..65535 | 0 | The 16 bit Ethernet type to use when sending events over layer 2 |
| 1005 | Event Subscriber Syslog Port | 0..65535 | 0 | The 16 bit UDP port to use when sending events using Syslog |
| 1006 | Reserved | | | |

**Miscellaneous Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 1007 | Easy Config LED Mode | 0..3 | 3 | Bit mask representing Easy Config LED Mode when smart mode is finished<br>Value 0: Link quality LEDs are off<br>Value 1: WLAN RSSI<br>Value 2: BT Link Quality<br>Value 3: Auto show WLAN if active otherwise show BT if active |
| 1008 | Reserved | | | |
| 1009 | Reserved | | | |
| 1010 | Reserved | | | |
| 1011 | Reserved | | | |
| 1012 | Reserved | | | |
| 1013 | Diagnose Mode | 0..4294967295 | 0x00010001 | Set diagnose mode bitmask. The following events will be sent when the bit is set:<br>Bit 0:<br>*WSCO - Connection to AP up<br>*WSCC - Connection to AP down<br>*WASA - Station has connected<br>*WASR - Station has disconnected<br>Bit 1:<br>*WSRSS - RSSI periodically sent while connected, with interval set by S3007<br>*WSCH - Used WLAN channel, sent upon connection setup<br>Bit 3:<br>*WSFRG - Roaming reassociating (only applicable if S4004 is 1)<br>*WSFRD - Roaming reassociated (only applicable if S4004 is 1)<br>Bit 16:<br>*BCI - Connection indication (incoming connection)<br>*BCO - Connection opened<br>*BCC - Connection closed<br>*BDST - Digital signal transition (only applicable if S2702 is 1)<br>*BLQC - Link Quality value falls below or rises above the value in S2700<br>Bit 17:<br>*BLQ - Link Quality periodically sent while connected, with interval 1000ms<br>Bit 18:<br>*BI - Background scan result, regardless of the current roaming operation.<br>Bit 19:<br>*BI - When background scan is active results for the currently connected device and possible handover candidates are sent.<br>*BRSS - When background scan is active an approximated RSSI value for the current connection is sent periodically.<br>*BRHC - Roaming handover candidate<br>*BRH - Roaming handover<br>Bit 20:<br>*BC - Connect request<br>*BCRC - Connect retry counter<br>*BCE - ACL Connection error<br>*BCF - Incorrect state<br>*BS - State changed<br>*BIE - Bluetooth inquiry event<br>*BIS - Bluetooth inquiry schedule<br>*BIC - Bluetooth inquiry complete<br>Bit 24:<br>*App start |

**Miscellaneous Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 1014 | Easy Config Mode Timeout | 0..4294967295 | 5000 | Maximum time to wait for first push on SMART button in milliseconds |
| 1015 | Radio Mode | 0..3 | 2 | Configures which radios should be enabled<br>0: All radio off<br>1: Enable only Bluetooth<br>2: Enable only WLAN<br>3: Enable Bluetooth and WLAN |
| 1016 | Log level | 0..3 | 1 | Configures what kind of events that should be presenred in the event log<br>0: Off, No events should be logged.<br>1: Error, Error events should be logged.<br>2: Warning, Warning and Error events should be logged.<br>3: Verbose, All events should be logged. |
| 1017 | Reserved | | | |
| 1018 | Reserved | | | |

**Bluetooth Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 2000 | Reserved | | | |
| 2001 | Reserved | | | |
| 2002 | Reserved | | | |
| 2003 | Reserved | | | |
| 2004 | Reserved | | | |
| 2005 | Reserved | | | |
| 2006 | Inquiry Class of Device Filter | 0..4294967295 | 0x00020300 | A found device must match this bitmask to be reported to the higher layers, e.g. to find all networking devices, bit 17 should be set (0x00020300 or 131840). Bit 0-1 and 24-31 are reserved and will be ignored |
| 2007 | Reserved | | | |
| 2008 | Reserved | | | |
| 2009 | Reserved | | | |
| 2010 | Max Number of Connections | 0..7 | 1 | The default value is set at production time and might differ for different product models. It is readable using the AT*AMSBC? command. |
| 2011 | Max Radio Output Power | -128..127 | 20 | The parameter value is not linear to the output power.<br>This is how the parameter value is linked to the output power:<br>16 or higher : approx 10 dBm<br>10 : approx 0 dBm<br>5 : approx -5 dBm<br>0 : approx -10 dBm<br>-5 : approx -15 dBm<br>-10 : approx -20 dBm<br>-20 or lower : approx -25 dBm |
| 2012 | External Connection Control | 0..1 | 0 | 0: Disable.<br>1: Enable. |
| 2013 | Reserved | | | |
| 2014 | Reserved | | | |
| 2015 | Reserved | | | |
| 2016 | Connect to Name Scheme | 0..2 | 1 | 0: Connect to first name (first found). This will make a limited inquiry searching for 1 unit, get the name of the found unit and, if the name matches, connect to it.<br>1: Connect to best name (highest RSSI value). This will make an inquiry, sort the devices regarding to the RSSI value. |

**Bluetooth Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| | | | | Then it will start from the device with best RSSI value, get the name and connect if it matches the desired name.<br>If it fails, it will go to the next device and so on.<br>2: Connect to name. This will make an inquiry and then get the name of the found devices.<br>When a matching name is found, a connection attempt will be done. If the connection fails, it will try with the next found and so on. |
| 2017 | Connect to Name Exact Match | 0..1 | 1 | 0: Connect to BT devices that contain a sub part of the connection name,<br>ex. 'DUT' will try to connect to 'xDUTx'. Where x is a sign for wild card.<br>1: Connect to an exact match of the BT local name in the connection list. |
| 2018 | Background Scanning Interval | 0..4294967295 | 15000 | Time between background scans in milliseconds |
| 2019 | Reserved | | | |
| 2020 | Active poll mode | 40..100 | 60 | Set active poll mode to introduce periodic BT classic link polling.<br>Active poll period in milliseconds. To Disable set 'Role to use Active poll mode' to None |
| 2021 | Role to use Active poll mode | 0..103 | 103 | Allow a Role to Set active poll mode.<br>0: None<br>100: PANU<br>101: NAP<br>103: PAN (PANU and NAP)<br>All others: Reserved |
| 2022 | RSSI poll interval | 0..65535 | 2000 | RSSI poll interval in milliseconds |

**Roaming BT Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 2700 | Reserved | | | |
| 2701 | Roaming list trigger RSSI | 0..1 | 0 | 0: Do not trig on RSSI<br>1: Trig on RSSI, Uses: Minimum acceptable RSSI |
| 2702 | Roaming list trigger input | 0..1 | 0 | 0: Do not trig on digital input<br>1: Trig on digital input |
| 2703 | Minimum acceptable RSSI | -85..-65 | -75 | Defines at what RSSI level a connection is considered acceptable. |
| 2704 | Roaming RSSI diff threshold | 0..127 | 10 | When the current connection's RSSI is below MIN_ ACCEPTABLE_RSSI and the difference between the connected device RSSI and the RSSI from the best device from background scan exceeds this threshold a roaming handover is done. |

**WLAN Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 3000 | Reserved | | | |
| 3001 | Reserved | | | |
| 3002 | Max Radio Power | -1..20 | -1 | Transmit power level in dBm. Valid values are 0-20 and -1. Adaptive transmit power level control is enabled with -1. |
| 3003 | Power Save Mode | 0..2 | 0 | 0: Off<br>1: Sleep<br>2: Deep Sleep |
| 3004 | Reserved | | | |
| 3005 | Reserved | | | |
| 3006 | Reserved | | | |

**WLAN Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 3007 | RSSI poll interval | 0..65535 | 1000 | RSSI poll interval in milliseconds |
| 3008 | Reserved | | | |
| 3009 | Reserved | | | |
| 3010 | Reserved | | | |
| 3011 | Hide SSID | 0..1 | 0 | Hide SSID from being broadcasted |
| 3012 | Quality Of Service | 0..1 | 0 | Enable Quality Of Service |
| 3013 | Quality Of Service NoAck | 0..1 | 0 | Quality Of Service NoAck acknowledgment policy. Used to avoid retransmission of highly time-critical data.<br>0: Normal ack<br>1: No-ack<br>NOTE: This is only available in Client mode. |
| 3014 | RSSI_MIN | -100..-70 | -85 | Never connect to a WLAN AP where RSSI is below this value. |
| 3015 | Reserved | | | |
| 3016 | Reserved | | | |
| 3017 | Reserved | | | |

**Roaming WLAN Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 4000 | Roaming list trigger RSSI | 0..1 | 1 | 0: Do not trig on RSSI, WLAN roaming disabled.<br>1: Trig on RSSI, WLAN roaming enabled. |
| 4001 | Reserved | | | |
| 4002 | Roaming RSSI diff threshold | 0..20 | 10 | When the difference between the connected AP RSSI and the RSSI from the best AP from background scan exceeds this threshold a roaming handover is done. |
| 4003 | Trigger Scan RSSI | -95..-55 | -70 | Defines at what RSSI level a background scan should be initiated to find a better connection. |
| 4004 | Fast Roaming | 0..1 | 1 | 0: Disable fast roaming.<br>1: Enable fast roaming according to IEEE 802.11r, this causes the device to roam quicker between APs with the same SSID, but the BSSID and Channel parameters in the connection list is ignored.<br>Note that this is in Client mode only and the the AP's also need to support IEEE 802.11r. |
| 4005 | Roaming neighborhood watch | 0..1 | 0 | 0: Disable neighborhood watch.<br>1: Enable neighborhood watch.<br>Improve roaming by scanning the environment before exceeding the roaming threshold. |

**Network Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 5000 | TCP Keep-Alive Enable | 0..1 | 0 | Turn on/off TCP keep-alive packets. It is important to understand that sending frequent keep-alive packets usually isn't a good solution to detect dropped connections. Detecting dead links should be done on a higher level, i.e. in the user application protocol. There is a lot of information available on the subject on the web.<br>0: TCP keep-alive packets turned off<br>1 = TCP keealive packets turned on |
| 5001 | TCP Keep-Alive Idle Time | 0..4294967295 | 7200000 | Time in milliseconds for a TCP connection to be idle before a keep-alive packet is sent. |
| 5002 | TCP Keep-Alive Interval | 0..4294967295 | 75000 | Time in milliseconds between keep-alive packets. |
| 5003 | TCP Keep-Alive Lost Count | 0..255 | 9 | Number of lost keep-alive packets to wait before a TCP connection is reset. |

**Network Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 5004 | LLDP Send Interval. | 0..65535 | 60 | The module will per default send information in LLDP frames with its current setup. This can also be used to stay alive on access points that do not properly wake the module before a disassociation. Value in seconds.<br>0 = Off, do not send or process incoming LLDP frames |
| 5005 | Reserved | | | |
| 5006 | LLDP Hold Multiplier | 2..10 | 4 | This value multiplied with LLDP interval makes the total time an LLDP update is valid. |
| 5007 | Reserved | | | |

**PROFINET Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 5100 | Reserved | | | |
| 5101 | Profinet Prioritization | 0..1 | 0 | Set prioritization for PROFINET.<br>0: Disable<br>1: Enable |
| 5102 | Prioritized Ethernet Type | 0..65535 | 0x8892 | The 16-bit Ethernet type to prioritize |
| 5103 | Reserved | | | |
| 5104 | PTCP Filter | 0..1 | 1 | Set ProfiNET PTCP filter. When enabled, all incoming PTCP packets will be dropped.<br>0: Disable<br>1: Enable |

**Bluetooth Low Energy Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 6000 | Advertising Interval Minimum | 32..16384 | 1600 | Advertising interval minimum (must be <= Advertising Interval Maximum). Unit in multiples of 0.625 ms. |
| 6001 | Advertising Interval Maximum | 32..16384 | 2000 | Advertising interval maximum (must be >= Advertising Interval Minimum). Unit in multiples of 0.625 ms. |
| 6002 | Advertising Channel Map | 0..7 | 7 | Bit mask for advertising channel map.<br>Bit 0: Channel 37<br>Bit 1: Channel 38<br>Bit 2: Channel 39 |
| 6003 | Connect Connection Interval Minimum | 6..3200 | 24 | Connect connection interval minimum (must be <= Connect Connection Interval Maximum). Unit in multiples of 1.25 ms. |
| 6004 | Connect Connection Interval Maximum | 6..3200 | 40 | Connect connection interval maximum (must be >= Connect Connection Interval Minimum). Unit in multiples of 1.25 ms. |
| 6005 | Connect Latency | 0..500 | 0 | Connect latency for number of connection events. Unit in multiples of 1.0 ms. |
| 6006 | Connect Link Loss Timeout | 100..32000 | 2000 | Connect link loss timeout. Unit in multiples of 1.0 ms. |
| 6007 | Connect Create Connection Timeout | 0..65535 | 5000 | Connect create connection timeout. Unit in multiples of 1.0 ms. |
| 6008 | Connect Connection Scan Interval | 16..16384 | 48 | Connect connection scan interval (must be >= Connect Scan Window). Unit in multiples of 0.625 ms. |
| 6009 | Connect Scan Window | 16..16384 | 48 | Connect scan window (must be <= Connection Scan Interval). Unit in multiples of 0.625 ms. |
| 6010 | Bond Connection | 6..3200 | 24 | Bond connection interval minimum (must be <= Bond Connection Interval Maximum). Unit in multiples of 1.25 ms. |

**Bluetooth Low Energy Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| | Interval Minimum | | | |
| 6011 | Bond Connection Interval Maximum | 6..3200 | 40 | Bond connection interval maximum (must be >= Bond Connection Interval Minimum). Unit in multiples of 1.25 ms. |
| 6012 | Bond Connection Latency | 0..500 | 0 | Bond latency for number of connection events. Unit in multiples of 1.0 ms. |
| 6013 | Bond Link Loss Timeout | 100..32000 | 2000 | Bond link loss timeout. Unit in multiples of 1.0 ms. |
| 6014 | Bond Create Connection Timeout | 0..65535 | 5000 | Bond create connection timeout. Unit in multiples of 1.0 ms. |
| 6015 | Bond Scan Interval | 16..16384 | 48 | Bond scan interval (must be >= Bond Scan Window). Unit in multiples of 0.625 ms. |
| 6016 | Bond Scan Window | 16..16384 | 48 | Bond scan window (must be <= Bond Scan Interval). Unit in multiples of 0.625 ms. |
| 6017 | Remote Name Connection Interval Minimum | 6..3200 | 24 | Remote name request connection interval minimum (must be <= Remote Name Request Connection Interval Maximum). Unit in multiples of 1.25 ms. |
| 6018 | Remote Name Connection Interval Maximum | 6..3200 | 40 | Remote name request connection interval maximum (must be >= Remote Name Request Connection Interval Minimum). Unit in multiples of 1.25 ms. |
| 6019 | Remote Name Connection Latency | 0..500 | 0 | Remote name request latency for number of connection events. Unit in multiples of 1.0 ms. |
| 6020 | Remote Name Link Loss Timeout | 100..32000 | 2000 | Remote name request link loss timeout. Unit in multiples of 1.0 ms. |
| 6021 | Remote Name Create Connection Timeout | 0..65535 | 5000 | Remote name request create connection timeout. Unit in multiples of 1.0 ms. |
| 6022 | Remote Name Request Scan Interval | 16..16384 | 48 | Remote name request connection scan interval (must be >= Remote Name Request Scan Window). Unit in multiples of 0.625 ms. |
| 6023 | Remote Name Request Scan Window | 16..16384 | 48 | Remote name request scan window (must be <= Remote Name Request Scan Interval). Unit in multiples of 0.625 ms. |

**Serial Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 7000 | Escape Sequence Timing | 50..5000 | 1000 | For an escape sequence to be valid, a period of no data activity is required before and after the escape sequence. This register specifies the minimum time of no data activity required before and after the escape sequence in milliseconds. |
| 7001 | Modbus RTU Mode | 0..1 | 0 | A MODBUS message is placed by transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message, and to know when the message is completed. Partial messages will be rejected by the receiving device. When enabled, this mode will result in waiting a maximum of 1.5 characters before transmitting the frame and thus eliminating the chances of transmitting a partial frame.<br>Note: This mode should only be enabled when communicating with MODBUS RTU devices! |
| 7002 | TCP Keep-Alive Enable | 0..1 | 1 | Turn on/off TCP keep-alive packets. It is important to understand that sending frequent keep-alive packets usually isn't a good solution to detect |

**Serial Registers (continued)**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| | | | | dropped connections. Detecting dead links should be done on a higher level, i.e. in the user application protocol. There is a lot of information available on the subject on the web.<br>0: TCP keep-alive packets turned off<br>1 = TCP keealive packets turned on |
| 7003 | TCP Keep-Alive Idle Time | 0..4294967295 | 5000 | Time in milliseconds for a TCP connection to be idle before a keep-alive packet is sent. |
| 7004 | TCP Keep-Alive Interval | 0..4294967295 | 2000 | Time in milliseconds between keep-alive packets. |
| 7005 | TCP Keep-Alive Lost Count | 0..255 | 3 | Number of lost keep-alive packets to wait before a TCP connection is reset. |
| 7006 | Modbus Gateway response timeout | 200..10000 | 1000 | Time in milliseconds for Modbus Gateway response timeout. |
| 7007 | Reserved | | | |

**CAN Registers**

| Register | Name | Value Range | Default Value | Description |
|---|---|---|---|---|
| 8000 | CAN automatic bus-off restart | 0..1 | 0 | Controls whether the CAN interface should automatically try to recover from a bus-off error:<br>0: Off (a reboot is necessary to recover from bus-off)<br>1: On |
| 8001 | CAN RX buffer timeout | 0..1000 | 4 | Time in ms to buffer CAN data before forwarding it over TCP |
| 8002 | Reserved | | | |

Box 4126
300 04 Halmstad, Sweden

info@hms.se