SW User Guide

1VV0301277, Rev. 06 – 2016-05-24

TELIT
TECHNICAL
DOCUMENTATION

stollmann

Stollmann is a Telit brand.

BlueMod+S

User Guide

Release r06

Table of contents

1 Introduction

This document describes the usage of the BlueMod+S Bluetooth module featuring firmware version V1.100 or later.

Firmware version V1.xxx is based on SoftDevice S110 V6.0.0 and Stollmann bootloader V1.000.

In the meantime there exists a new firmware generation V2.xxx. Firmware version V2.xxx is based on SoftDevice S110 V8.0.0 and Stollmann bootloader V2.003. Firmware version V2.xxx includes advanced functionality which is marked separately within this document.

Note:
Firmware version V2.xxx is not compatible with firmware version V1.xxx series. It is not possible to update devices using firmware version V1.xxx to firmware version V2.xxx and vice versa.

For a detailed description of the commands refer to the *BlueMod+S AT Command Reference [1]*.

All referenced documents can be downloaded from:

http://www.telit.com/support/technical-support/ and select “Downloadzone”.

2 Initial Configuration

In the default configuration (AT+LEADE=0) the BlueMod+S advertises the Terminal I/O service immediately after power up and therefore it is able to accept an incoming call request.

If the user wants to configure the BlueMod+S with specific settings without interruption by an incoming call we recommend to disable the automatic advertising by setting AT+LEADE=3. After the configuration is done the advertising can be enabled again (AT+LEADE=0 for advertising Terminal I/O service or AT+LEADE=1 for customized advertising).

3 Startup Timing

The startup time until the BlueMod+S is able to accept link requests or serial data depends on:

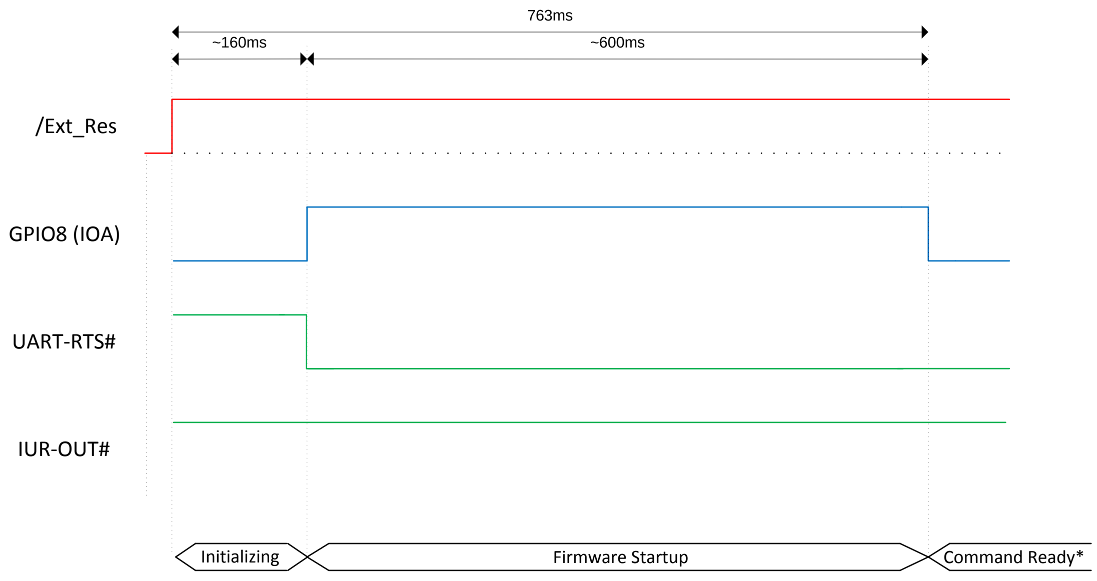
- the firmware version
- the source for the slow clock
- the usage of the UART Interface Control Protocol (UICP)

For more details about the UICP protocol please refer to the document
UICP+ UART Interface Control Protocol [3].

3.1 Firmware Version 1.106

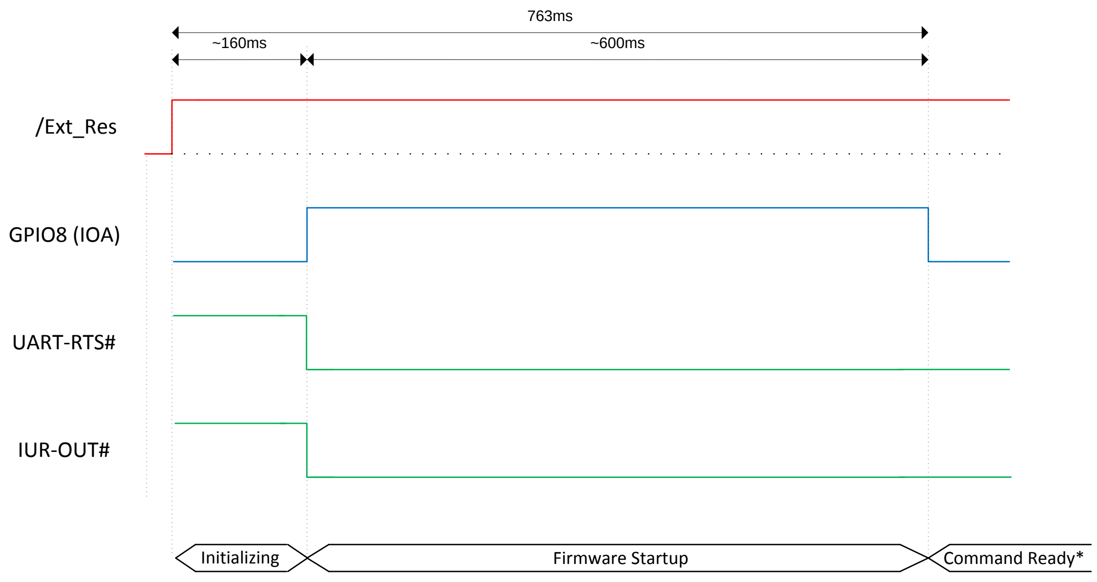
The following diagrams show the startup timing of the BlueMod+S based on firmware version 1.106 with external 32,768kHz crystal signal.

3.1.1 UICP disabled



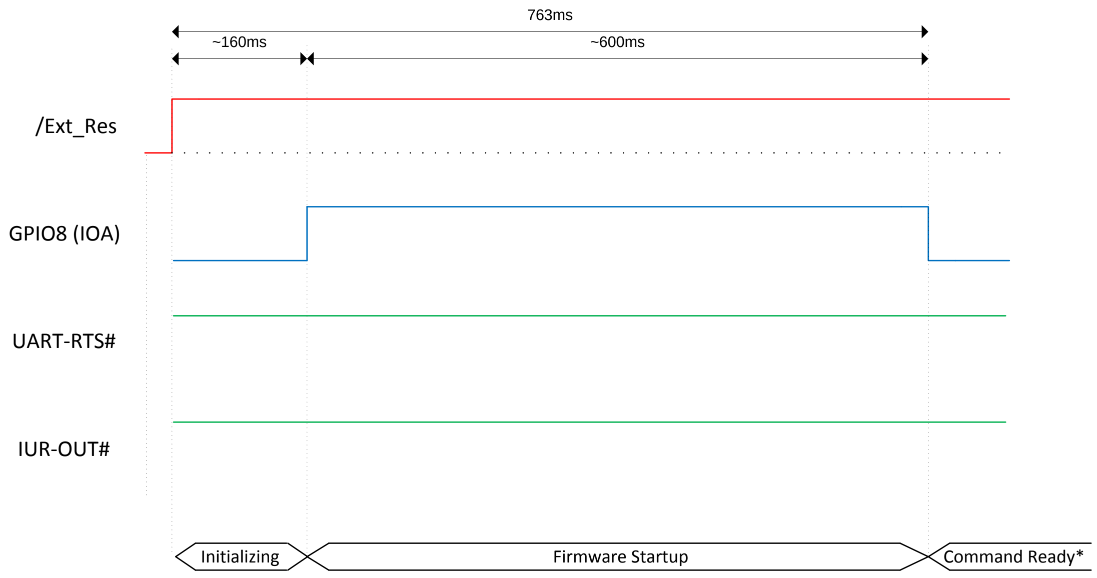
(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.1.2 UICP enabled, interface up



(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.1.3 UICP enabled, interface down

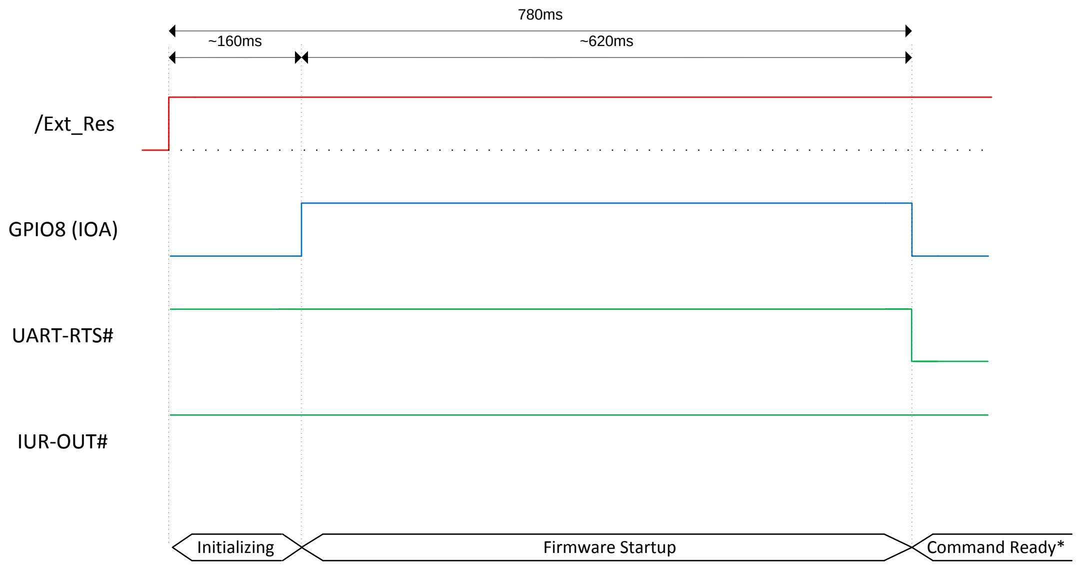


(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.2 Firmware Version 1.107

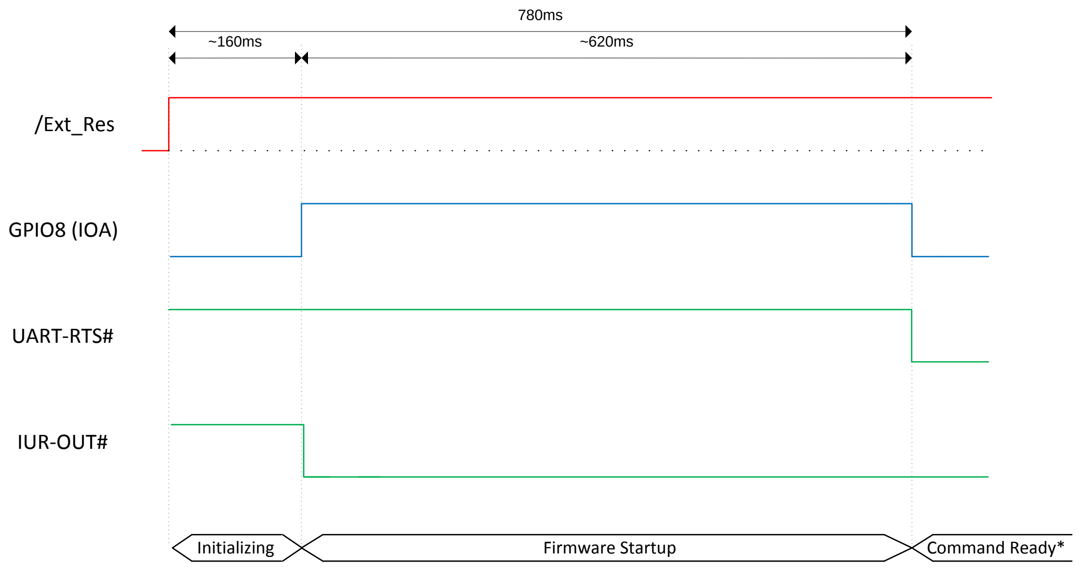
The following diagrams show the startup timing of the BlueMod+S based on firmware version 1.107 with external 32,768kHz crystal signal.

3.2.1 UICP disabled



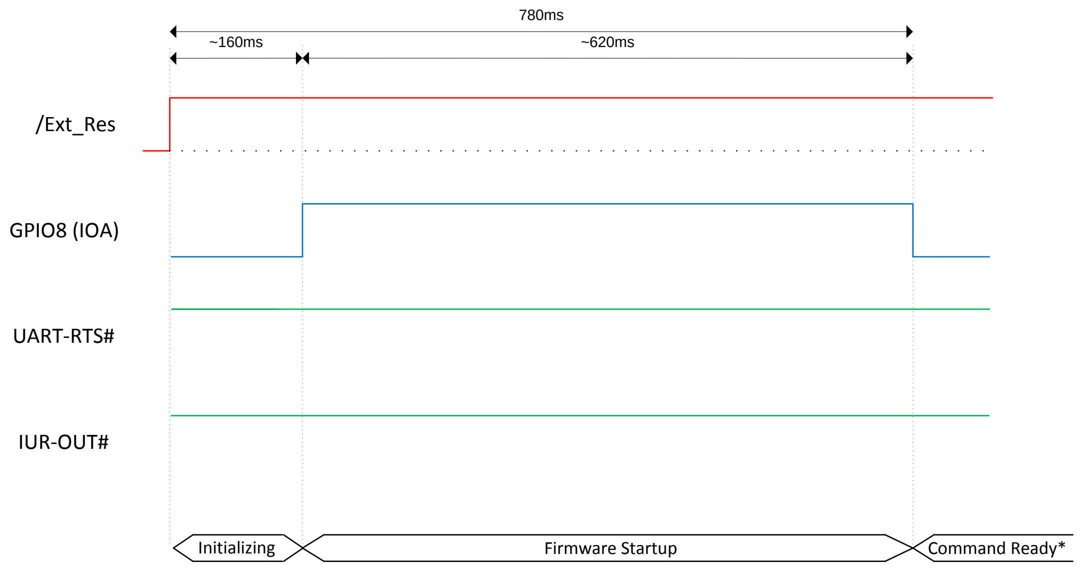
(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.2.2 UICP enabled, interface up



(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.2.3 UICP enabled, interface down

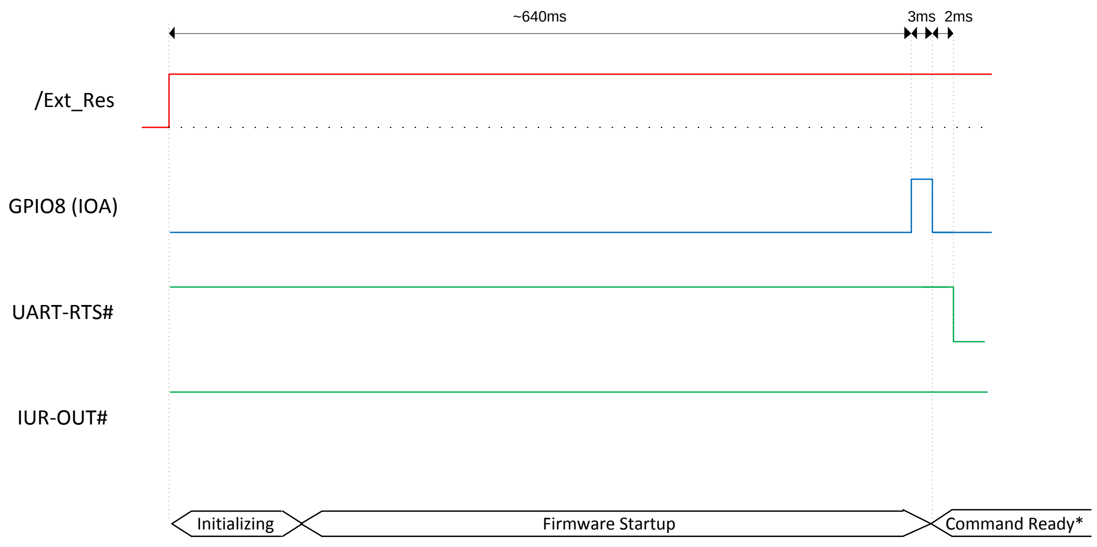


(*) The firmware is command ready ~800ms after the reset has been released and when GPIO8 (IOA) is low.

3.3 Firmware Version 2.003

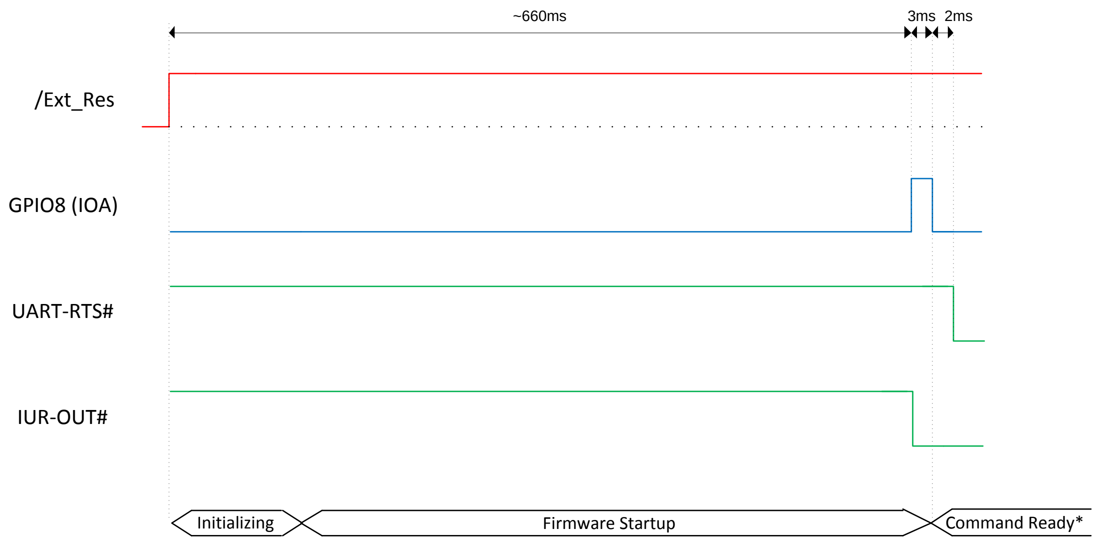
The following diagrams show the startup timing of the BlueMod+S based on firmware version 2.003 with external 32,768kHz crystal signal.

3.3.1 UICP disabled



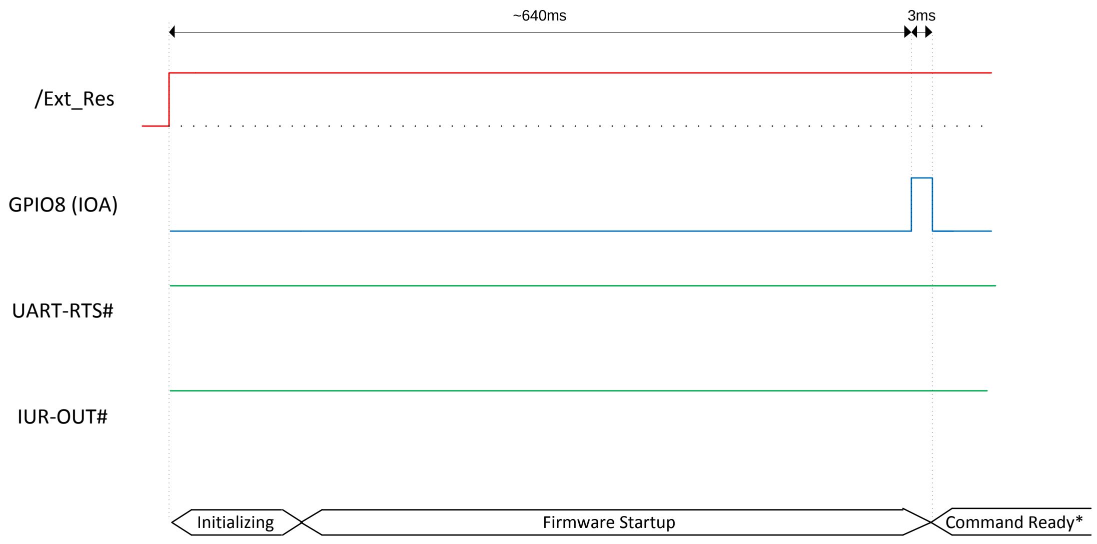
(*) The firmware is command ready ~650ms after the reset has been released and when GPIO8 (IOA) is low.

3.3.2 UICP enabled, interface up



(*) The firmware is command ready ~670ms after the reset has been released and when GPIO8 (IOA) is low.

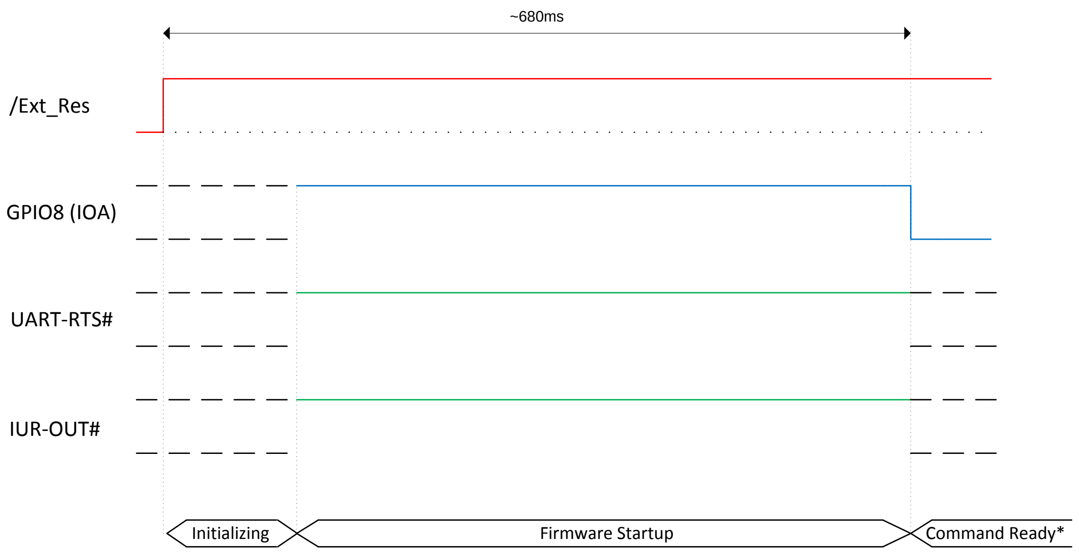
3.3.3 UICP enabled, interface down



(*) The firmware is command ready ~650ms after the reset has been released and when GPIO8 (IOA) is low.

3.4 Firmware Version 2.005 and later

The following diagram show the startup timing of the BlueMod+S based on firmware version 2.006 with external 32,768kHz crystal signal.



(*) The firmware is command ready ~680ms after the reset has been released and when GPIO8 (IOA) is low.

After GPIO8 gets low the state of the /RTS and /IUR-OUT lines depends on the UICP parameter. When UICP is disabled (AT+UICP=0) both output lines get low, otherwise the UICP function will be started.

For more details about the UICP protocol please refer to the document *UICP+ UART Interface Control Protocol [3]*.

4 Pairable and Bondable Mode

In general we distinguish between pairing and bond. Pairing is the active process to generate a set of encryption keys. The paring can be done with or without user interaction depending of the I/O capabilities. The pairing will result in a bond if the generated data is stored in the bonded device list (AT+BNDLIST).

AT+BPAIRMODE controls if a pairing is performed or not.

Value	Description
0	No pairing (pairing request will be refused)
1	Pairing

AT+BNDS controls the storing of the pairing information as bond.

Value	Description
0	No storing (no bond)
1	Storing (entry in the bonded device list)

The bonded device list is affected by the following commands:

- AT+BNDLIST shows the devices stored in the bonded device list
- AT+BNDSIZE determines the size of the bonded device list and deletes the whole list when modifying the size
- AT+BNDDEL deletes single entries or the whole list
- AT&F1 deletes the bonded device list

If the bonded device list is full and another device is bonded, the least recently used device will be overwritten by the new one. If bonds are not required please set AT+BNDS=0.

5 Security

This chapter describes the security mechanisms of the BlueMod+S to control the access to the local Bluetooth devices characteristics.

The pairing process is triggered automatically when an access to a characteristic is requested that requires security.

The behavior of LE Security is configurable using the parameters for I/O capabilities (AT+BIOCAP) and a man in the middle protection (AT+BMITM).

The securtity level of Terminal I/O is configurable using the parameter AT+LETIO.

Value	Description
0	Terminal I/O service disabled (no advertising, no characteristics)
1	Terminal I/O service enabled, security is required
2	Terminal I/O service enabled, no security required

AT+BIOCAP sets the input and output capabilities of the device used for LE Security.

Value	Description
0	Display only
1	Display Yes/No
2	Keyboard only
3	No input no output (default)
4	Display and keyboard

AT+BMITM controls the man in the middle (MITM) protection of the device during LE Security.

Value	**Description**
0	Man in the middle protection disabled (default)
1	Man in the middle protection enabled

LE Security defines the following association models based on the Input/Output (I/O) capabilities of the two devices:

- **Just Works:**

This method is used when at least one of the devices does not have display capability of six digits and also is not capable of entering six decimal digits using a keyboard or any other means (no I/O).
This method does not provide MITM protection (see 5.1 Connection Example Terminal I/O “Just Works”).

- **Passkey Entry:**

This method may be used between a device with a display and a device with numeric keypad entry (such as a keyboard), or two devices with numeric keypad entry (see 5.2 Connection Example Terminal I/O “Passkey Entry”).
In the first case, the display is used to show a six digit numeric code to the user, who then enters the code on the keypad.
In the second case, the user of each device enters the same six digit numeric code.
Both cases provide MITM protection.

Possible combinations of I/O capabilities and the possibility of MITM protection are listed in the table below. For each case of the “MITM protection” an example of the serial messages between the BlueMod+S and the DTE are listed.

In case the user choose a scenario where MITM protection is not allowed but one of the communication devices is configured to MITM protection, the pairing is refused.

Remote device / BM+S	Display only	Display Yes/No	Keyboard only	No input no output	Display and keyboard
Display only AT+BIOCAP=0	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>	Just Works (both automatic confirmation) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>
Display Yes/No AT+BIOCAP=1	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) No MITM protection	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>	Just Works (both automatic confirmation) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>
Keyboard only AT+BIOCAP=2	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>	Passkey entry (both input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>	Just Works (both automatic confirmation) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>
No input no output AT+BIOCAP=3	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) *No MITM protection*	Just Works (both automatic confirmation) *No MITM protection*
Display and keyboard AT+BIOCAP=4	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey>	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>	Just Works (both automatic confirmation) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* incoming connection: SSPPIN <BT addr> ? AT+BSSPPIN <BT addr>,<passkey> outgoing connection: SSPPIN <BT addr> <passkey>

Green color:	BM+S output message	SSPPIN <BT addr> ? (example)
Blue color:	BM+S input request	AT+BSSPPIN <BT addr> <passkey> (example)

The following flow charts will give an example for the different SSP authentication methods “just works” and “passkey entry” within an incoming call request from an iPhone (or other compatible iOS device) using Stollmann’s Terminal I/O Utility app to the BlueMod+S (see also the connection example in chapter 6.6 Connection Example with iPhone).

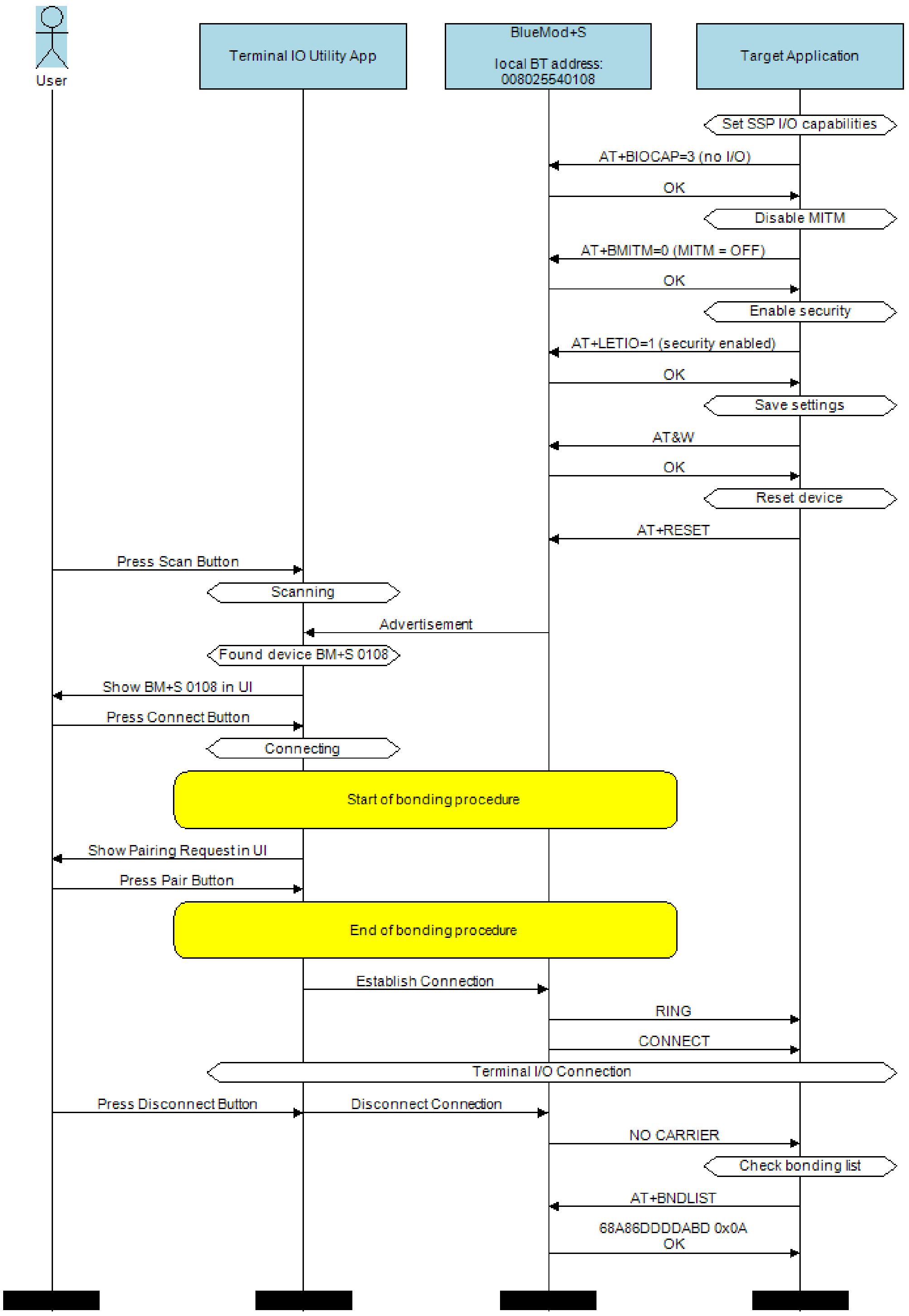
The “*Target Application*” part will simulate the device at the end (DTE) which communicates to the BlueMod+S with configuration commands.

The interesting part of the bonding procedure is placed between the yellow boxes “*Start of bonding procedure*” and “*End of bonding procedure*”.

All serial commands between the “*Target Application*” and the “*BlueMod+S*” out of the bonding procedure are used for further configuration of LE Security.

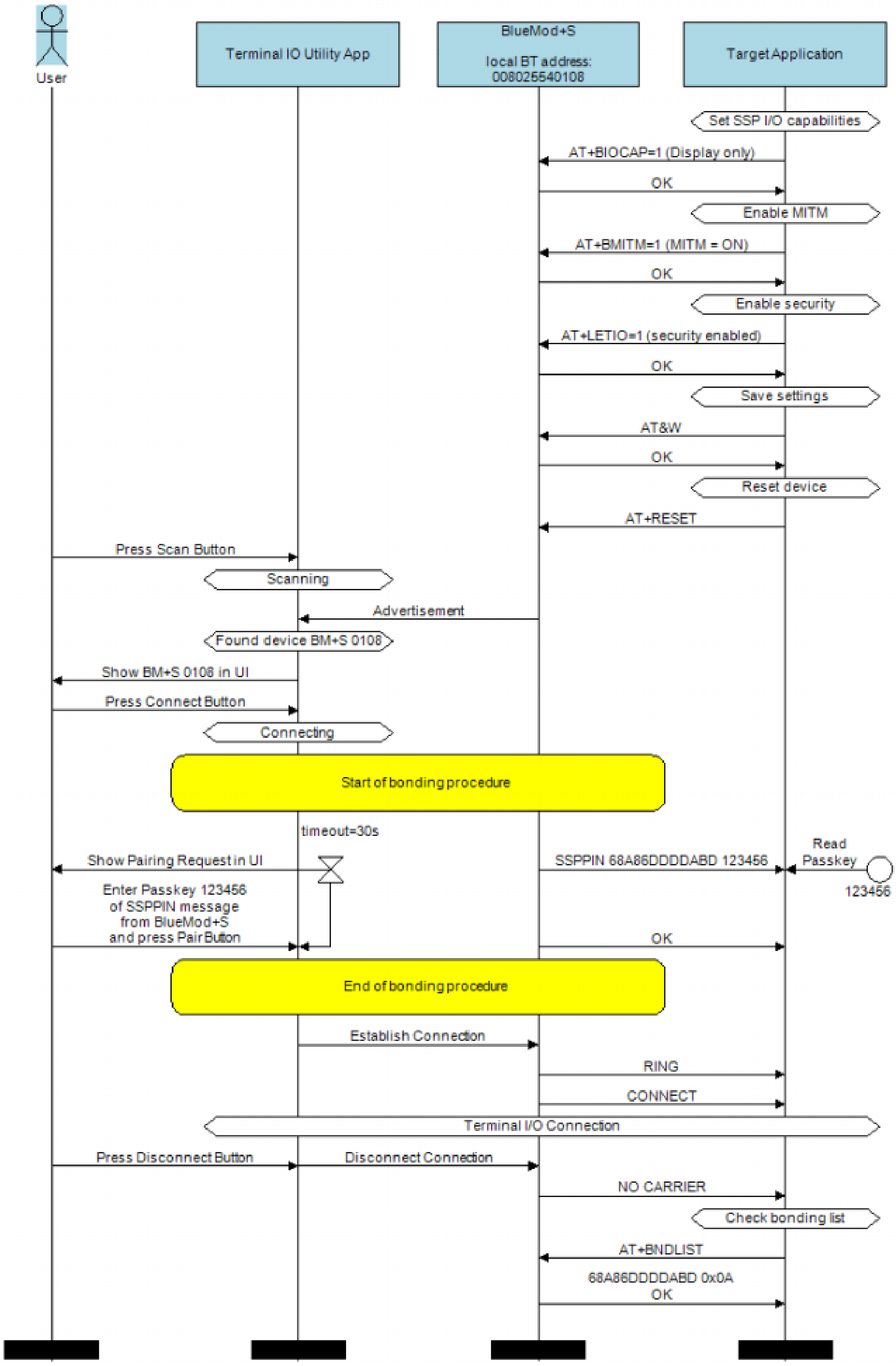
The configuration commands and responses within the flow charts are described in the *BlueMod+S AT Command Reference [1]*.

5.1 Connection Example Terminal I/O “Just Works”



5.2 Connection Example Terminal I/O “Passkey Entry”

with I/O capabilities “display only”



6 Terminal I/O over Bluetooth Low Energy

This chapter describes how to setup the BlueMod+S to establish a connection using Terminal I/O and Bluetooth Low Energy.

6.1 GAP Role

BlueMod+S supports the GAP role peripheral.

A peripheral is a device that advertises by using connectable advertising packets. Searching for Bluetooth Low Energy client devices is not possible.

6.2 Advertising Interval

A device advertises to be visible over the air for other scanning devices. The time between such advertising events is called advertising interval.

To set the advertising interval, use the parameters LEADINTMIN and LEADINTMAX described in the *BlueMod+S AT Command Reference [1]*.

6.3 Connection Interval

The connection interval is a delta time that determines the frequency with that the central will transmit and synchronize with a peripheral device during a connection. It can hardly affect power consumption and has to be set according to the required scenario.

To set the connection interval, use the parameters LECONINTMIN and LECONINTMAX described in the *BlueMod+S AT Command Reference [1]*.

6.4 Slave Latency

The slave latency is important for the power consumption of a peripheral device. It sets the number of master connection intervals that the slave can ignore. Setting this value to a high number increases the latency of the device.

To set the slave latency, use the parameter LESLAVELAT described in the *BlueMod+S AT Command Reference [1]*.

6.5 Local Device Name

To set the local device name, use the parameter BNAME described in the *BlueMod+S AT Command Reference [1]*.

With Bluetooth Low Energy the value of BNAME is used in the LE SCAN_RESP message (advertising) and in the GAP characteristic.

In the LE SCAN_RESP message the length of the name is truncated to 10 bytes, due to the maximum size of the data packet. Choose your name scheme that the devices can be differed within the first 10 bytes of the name.

In addition to the name delivered in the advertising the BlueMod+S provides the full name in the GAP – DEVICE NAME CHARACTERISTIC.

It depends on the scanning device when and which name it displays.

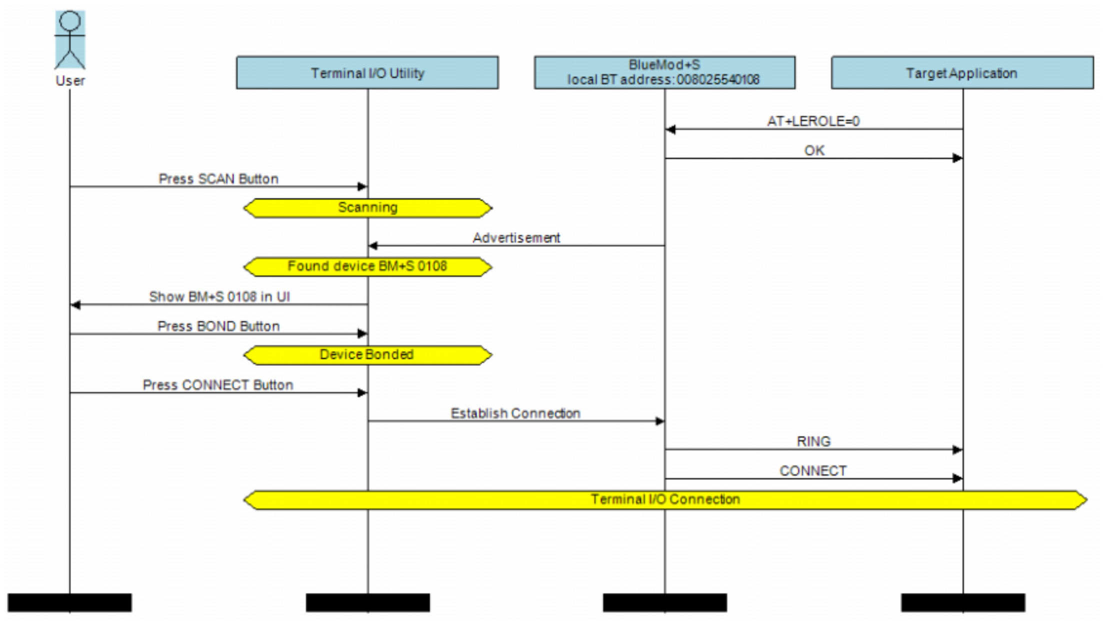
iOS for example takes the name from the advertising in the first step and updates it with the GAP name once it discovers the services from the device.

6.6 Connection Example with iPhone

To establish a Bluetooth Low Energy connection from the iPhone to the BlueMod+S the “Terminal IO Utility” App from Stollmann needs to be installed on the iPhone.

The following QR-Code provides the link to download the “Terminal IO Utility”.

The Terminal IO Utility App allows the user to connect to Terminal I/O peripheral devices and exchange data providing a simple terminal emulation.



As soon as the connection is established data can be sent from iPhone to BlueMod+S and vice versa.

7 UART Interface Control Protocol (UICP)

7.1 General Protocol Description

Stollmann UART Interface Control Protocol (UICP) defines a protocol to control the logical state of an UART based interface, thereby peers to switch off local UART devices for power saving (or other) reasons.

The UICP+ is a bi-directional, symmetrical protocol that allows to negotiate UART interface states with a communication partner connected via UART by the use of standard UART signal lines.

The UICP+ mechanisms defined here enable the involved peers to negotiate UART interface states by signaling the remote peer that it is allowed to enter or exit an UART interface up state.

The UICP+ does not enforce any power saving support of the involved peers but implements mechanisms to allow the save usage of MCU power saving features like UART peripheral switched off.

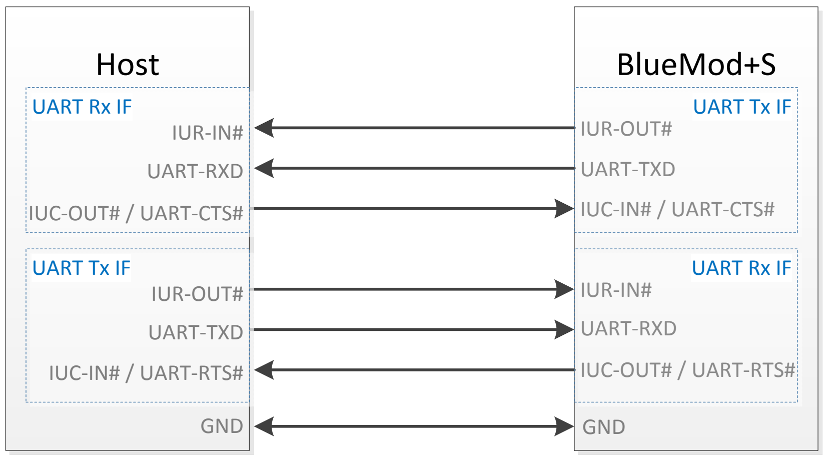
7.2 Requirements of Using UICP on BlueMod+S

To make use of UICP, the lines UART-TXD, UART-RXD, UART-RTS# (IUC-OUT#), UART-CTS# (IUC-IN#), IUR-OUT# and IUR-IN# should be connected between BlueMod+S and the host and additionally the UICP protocol should be implemented on host site.

A detailed description of implementing UICP is described in the document *UICP+ UART Interface Control Protocol [3].*

To activate UICP on the BlueMod+S the configuration parameter AT+UICP=1 needs to be set (followed by AT&W and AT+RESET).

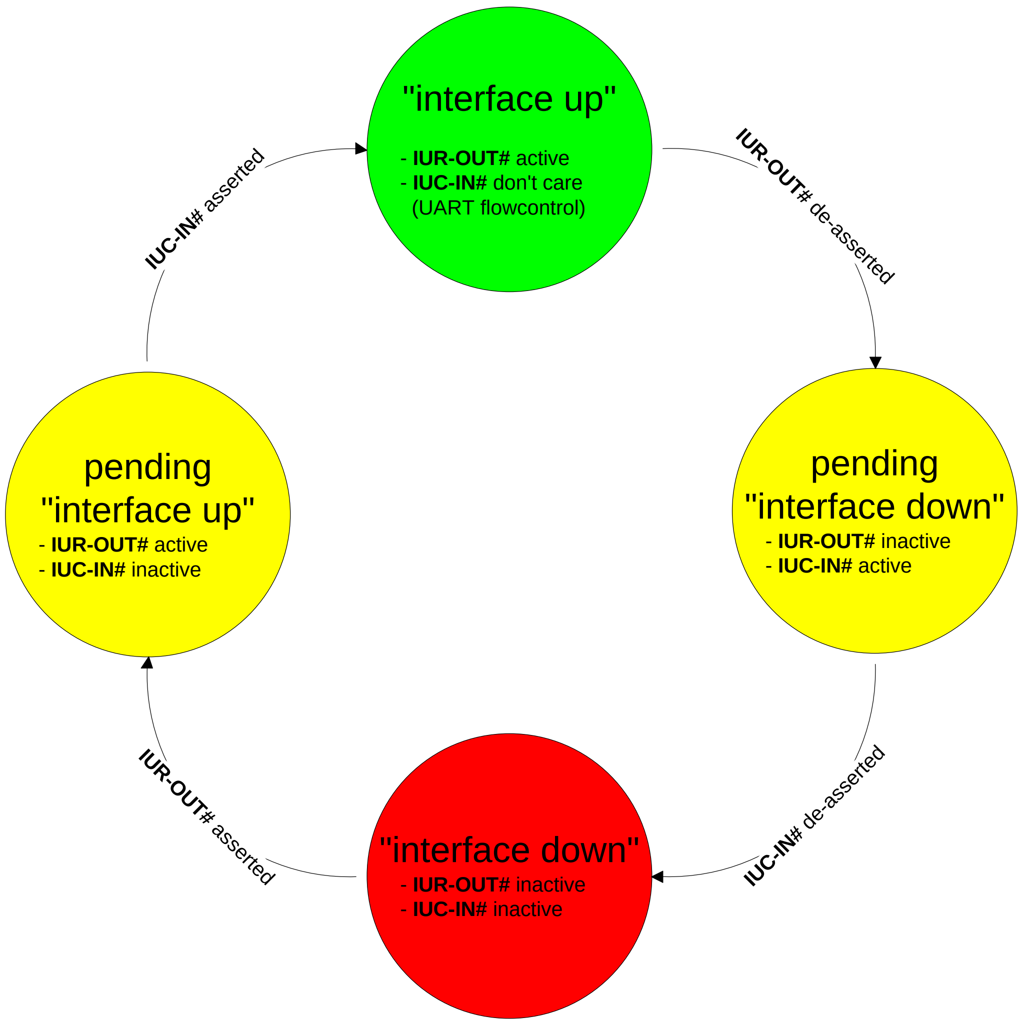
7.3 Connection Example between BlueMod+S and Host Controller



Further information about the BlueMod+S UART interface is described in the document *BlueMod+S Hardware Reference [4]*.

7.4 UICP Protocol States

The UICP protocol defines four states:



- **interface up**
  normal operation, RTS/CTS hardware flow control is active
- **pending interface down**
  IUR-OUT# is requested to go to “interface down” state
  IUC-IN# is not confirmed
- **interface down**
  IUR-OUT# and IUC-IN# are de-asserted in “interface down” state
  and can enable MCU power saving
- **pending interface up**
  IUR-OUT# is requested to go to “interface up” state,
  IUC-IN# is not confirmed

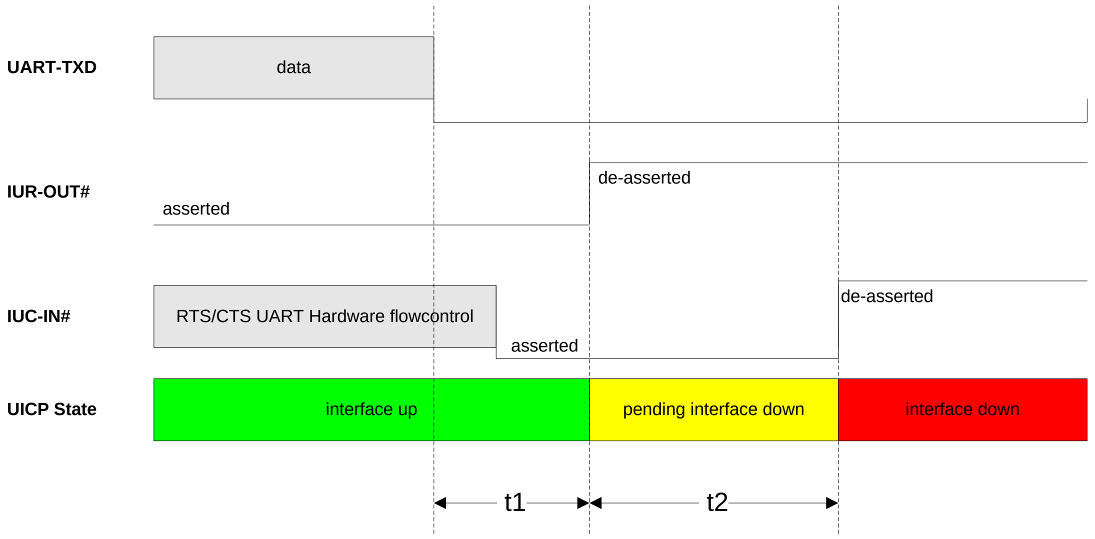
Note: All data received before the interface up state has been achieved shall be seen as invalid data and shall be discarded.

7.4.1 Drive from “interface up” to “interface down” State

Once a de-asserted IUR-OUT# signal of the initiator is detected by the acceptor, the acceptor shall confirm that signal by de-asserting its IUC-OUT# signal which is connected to the IUC-IN# signal of the initiator.

After the initiator detects a de-asserted IUC-IN# signal both devices go into “interface down” state and can enable MCU power saving mechanisms.

During MCU power saving, the MCU can switch off the UART but shall be able to detect an IUR# assert.



t1 >= 100ms (see this chapter)

t2 < 1s

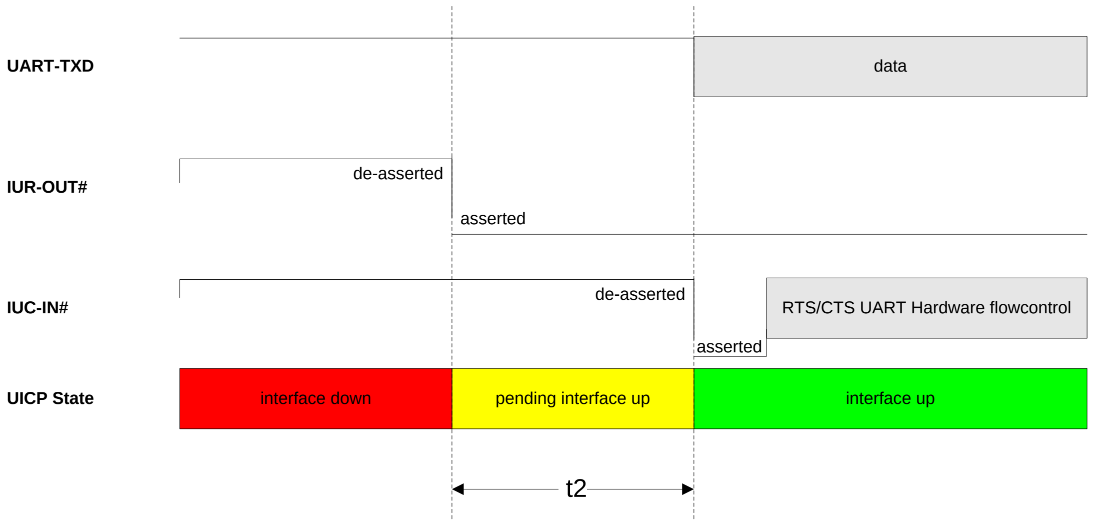
7.4.2 Drive from “interface down” to “interface up” State

To initiate the state change from “interface down” state to “interface up” state the initiator shall assert the IUR-OUT# signal.

The acceptor confirms the IUR-IN# signal with asserting its IUC-OUT# signal which is connected to the IUC-IN# signal of the initiator.

Once the acceptor detects the assert of the IUR-OUT# signal from the initiator, it can disable MCU power saving mechanisms but shall ensure the UART is ready to receive data before it confirms asserting its IUC-OUT# signal which is connected to the IUC-IN# signal of the initiator.

Once the initiator detects the assert of the IUC-IN# signal of the acceptor, the in initiator can send data to the acceptor.



7.5 Example of UICP Usage

The following examples shows the state change between the BlueMod+S and the host.

The scenario here might be that both devices use the “interface down” state to drive the MCU into some kind of power saving mode that allows to “wake up” the MCU with external GPIO signals.

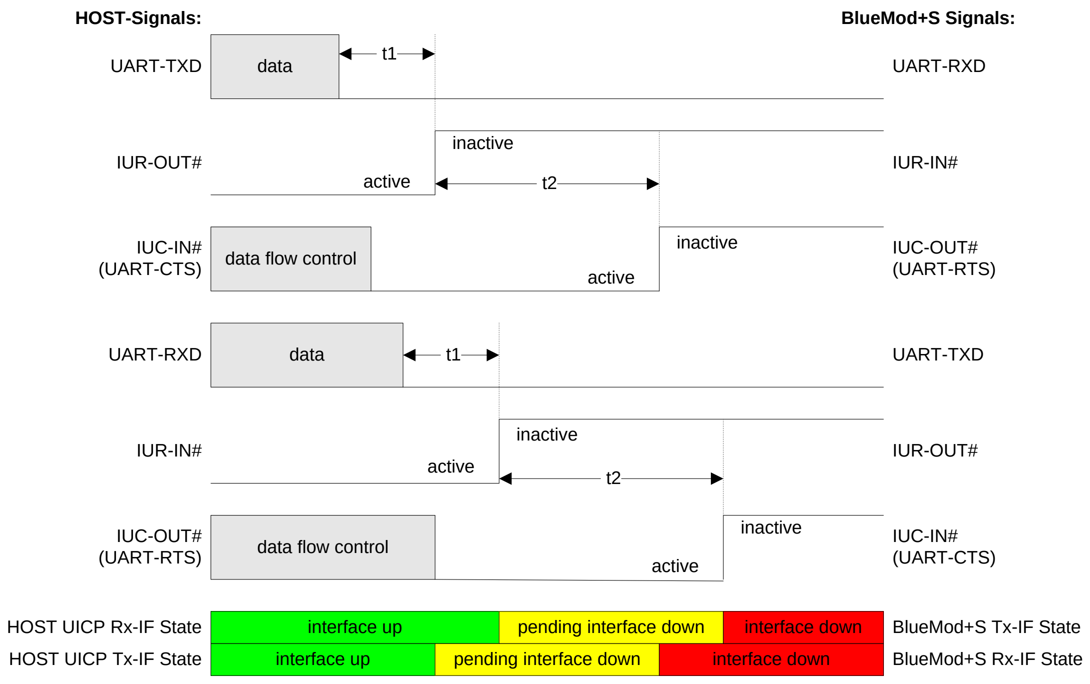
7.5.1 State Change from “interface up” to “interface down”

Host and BlueMod+S are in the state “interface up” and exchange bidirectional data. After the host has send all data and is idle for **t1** in its Tx direction it signals the BlueMod+S that it is allowed to go to “interface down” state by de-asserting IUR-OUT# signal.

Parallel to that UICP signaling from host to BlueMod+S the BlueMod+S has send all data as well and is idle for **t1** in its Tx direction, so it signals the host that it is allowed to go to “interface down” state by de-asserting IUR-OUT# signal.

The host and the BlueMod+S each wait for a maximum time **t2** to detect the de-asserted IUC-IN# signal. After receiving this input change via the IUC-IN# signal both devices may change from state “pending interface down” to state “interface down”.

Both UICP signaling sequences proceed in parallel until host and BlueMod+S interfaces are in “interface down” state.



7.5.2 State Change from “interface down” to “interface up”

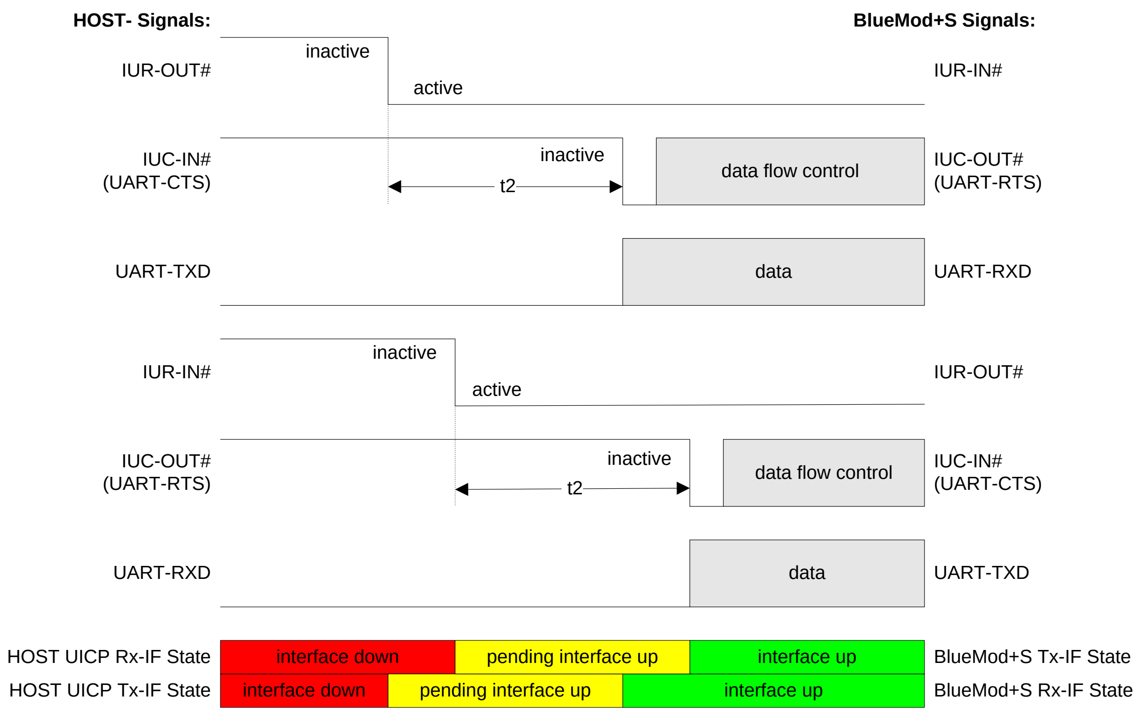
Host and BlueMod+S are in the state “Interface down” and may have the MCU into some kind of power saving state.

The host wants to send data to the BlueMod+S and asserts its IUR-OUT# signal.

Parallel to that UICP signaling from host to BlueMod+S the BlueMod+S wants to send data to the host and asserts its IUR-OUT# signal as well.

The host and the BlueMod+S each wait for a maximum time **t2** to detect the assertion via the IUC-IN# signal. After receiving this input change of IUC-IN# both devices may assume that the interface of the remote device changed from state “pending interface up” to state “interface up”.

Both UICP signaling sequences proceed in parallel until host and BlueMod+S interfaces are in “interface up” state and data can be exchanged bidirectional.

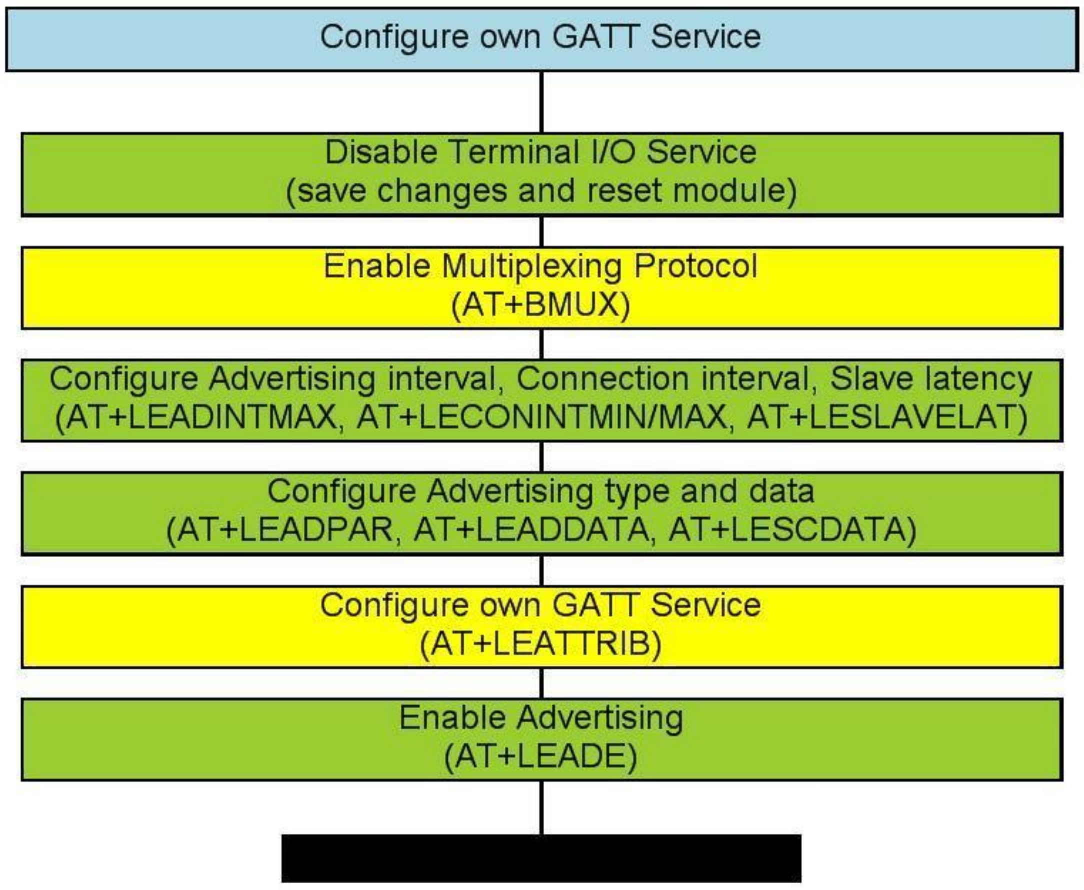


8 GATT Configuration

This chapter describes the usage of the GATT server commands. Using these commands, a user can setup own GATT services or profiles in the BlueMod+S and define its own advertising. The services can exist in addition to the Terminal I/O service or stand-alone.

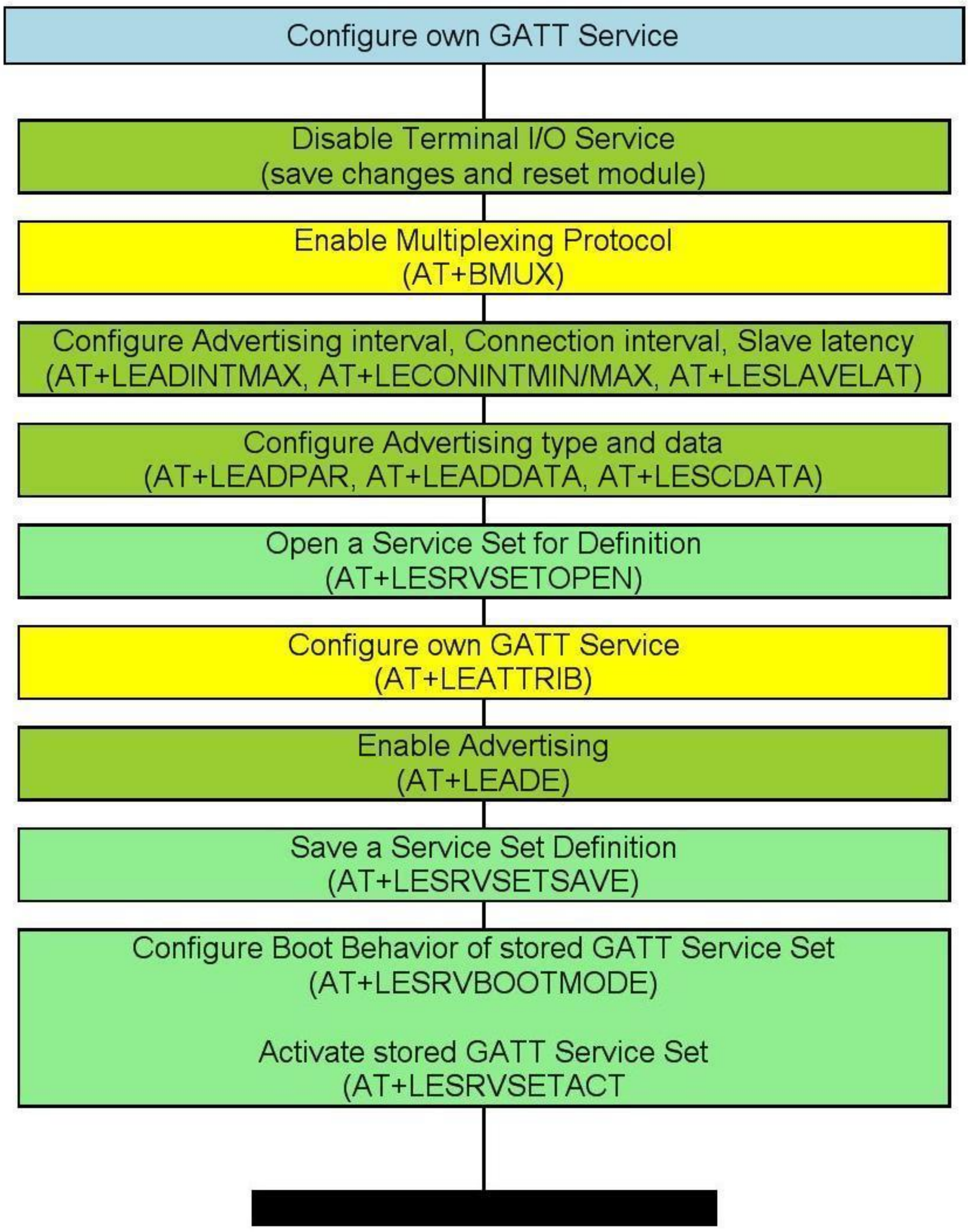
In order to handle different GATT characteristic values of an own GATT service it is required to use the UART based multiplexing mode (MUX).
For a detailed description of the multiplexing mode refer to the
BlueMod+S AT Command Reference [1].

The following flow chart lists the different steps to create an own GATT service.
For a detailed description of the GATT configuration commands refer to the
BlueMod+S AT Command Reference [1].
The detailed steps of commands are listed in a separate example in a following chapter.


Configure own GATT Service
Disable Terminal I/O Service (save changes and reset module)
Enable Multiplexing Protocol (AT+BMUX)
Configure Advertising interval, Connection interval, Slave latency (AT+LEADINTMAX, AT+LECONINTMIN/MAX, AT+LESLAVELAT)
Configure Advertising type and data (AT+LEADPAR, AT+LEADDATA, AT+LESCDATA)
Configure own GATT Service (AT+LEATTRIB)
Enable Advertising (AT+LEADE)

The steps to configure the advertising are optional (green).
A GATT service can be used without being stated in the advertising (yellow).
If the user wants a customized advertising all steps have to be done.

Since firmware version V2.xxx the BlueMod+S supports the possibility to store a custom specific GATT server configuration.
For a detailed description of these commands (light green) refer to the *BlueMod+S AT Command Reference [1]*.


Configure own GATT Service
Disable Terminal I/O Service
(save changes and reset module)
Enable Multiplexing Protocol
(AT+BMUX)
Configure Advertising interval, Connection interval, Slave latency
(AT+LEADINTMAX, AT+LECONINTMIN/MAX, AT+LESLAVELAT)
Configure Advertising type and data
(AT+LEADPAR, AT+LEADDATA, AT+LESCDATA)
Open a Service Set for Definition
(AT+LESRVSETOPEN)
Configure own GATT Service
(AT+LEATTRIB)
Enable Advertising
(AT+LEADE)
Save a Service Set Definition
(AT+LESRVSETSAVE)
Configure Boot Behavior of stored GATT Service Set
(AT+LESRVBOOTMODE)
Activate stored GATT Service Set
(AT+LESRVSETACT

8.1 Configuring the Advertising

When using the integrated Terminal I/O profile it is not possible to include an additional UUID within the existing Terminal I/O advertising.

If the user wants to set the UUID of a service other than Terminal I/O in the advertising, the Terminal I/O advertising has to be disabled by setting AT+LEADE=3.

The complete set of advertising data has to be set with the commands: AT+LEADPAR, AT+LEADDATA, AT+LESCDATA.

The advertising starts, as soon as it is enabled by AT+LEADE=1.

All referenced services shall be defined before enabling advertising, otherwise the device sends advertising without providing the appropriate services for a short time.

8.1.1 List of available UUIDs

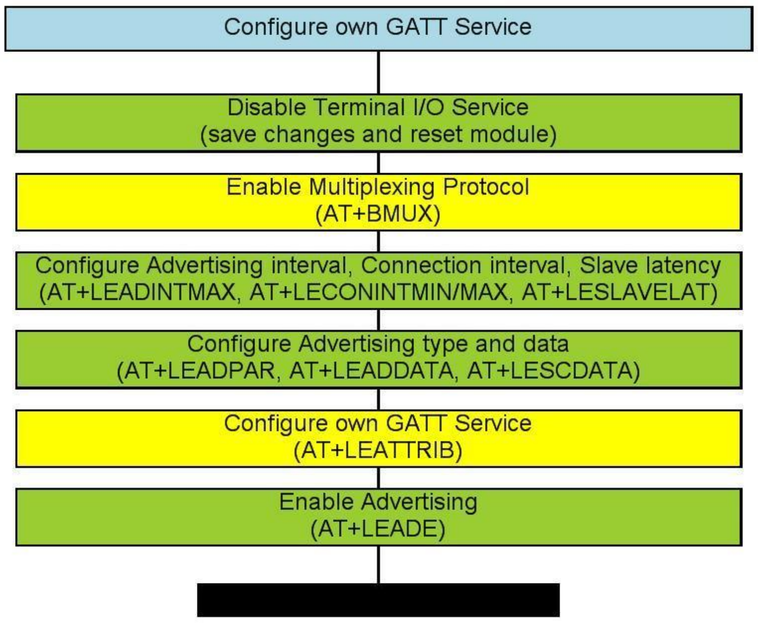
A list of reserved 16bit UUIDs is available onto the web site of the Bluetooth SIG https://www.bluetooth.org.

8.1.2 Beacon Mode

To configure the BlueMod+S as a Beacon or iBeacon please follow the instructions in the document *Using the BlueMod+S as a Beacon [7]*.

8.2 Configure own GATT Service

The following example shows how to setup an own standard based GATT service “Health Thermometer Monitor” (HTM) profile in the BlueMod+S using the global parts of the following flow chart.


Configure own GATT Service
Disable Terminal I/O Service
(save changes and reset module)
Enable Multiplexing Protocol
(AT+BMUX)
Configure Advertising interval, Connection interval, Slave latency
(AT+LEADINTMAX, AT+LECONINTMIN/MAX, AT+LESLAVELAT)
Configure Advertising type and data
(AT+LEADPAR, AT+LEADDATA, AT+LESCDATA)
Configure own GATT Service
(AT+LEATTRIB)
Enable Advertising
(AT+LEADE)

The following MSC flow chart list all required AT commands between the host controller and the BlueMod+S device to configure a HTM profile.

Note: After changing from standard AT command mode to the MUX mode all AT commands needs to be sent in MUX mode which includes the MUX header.

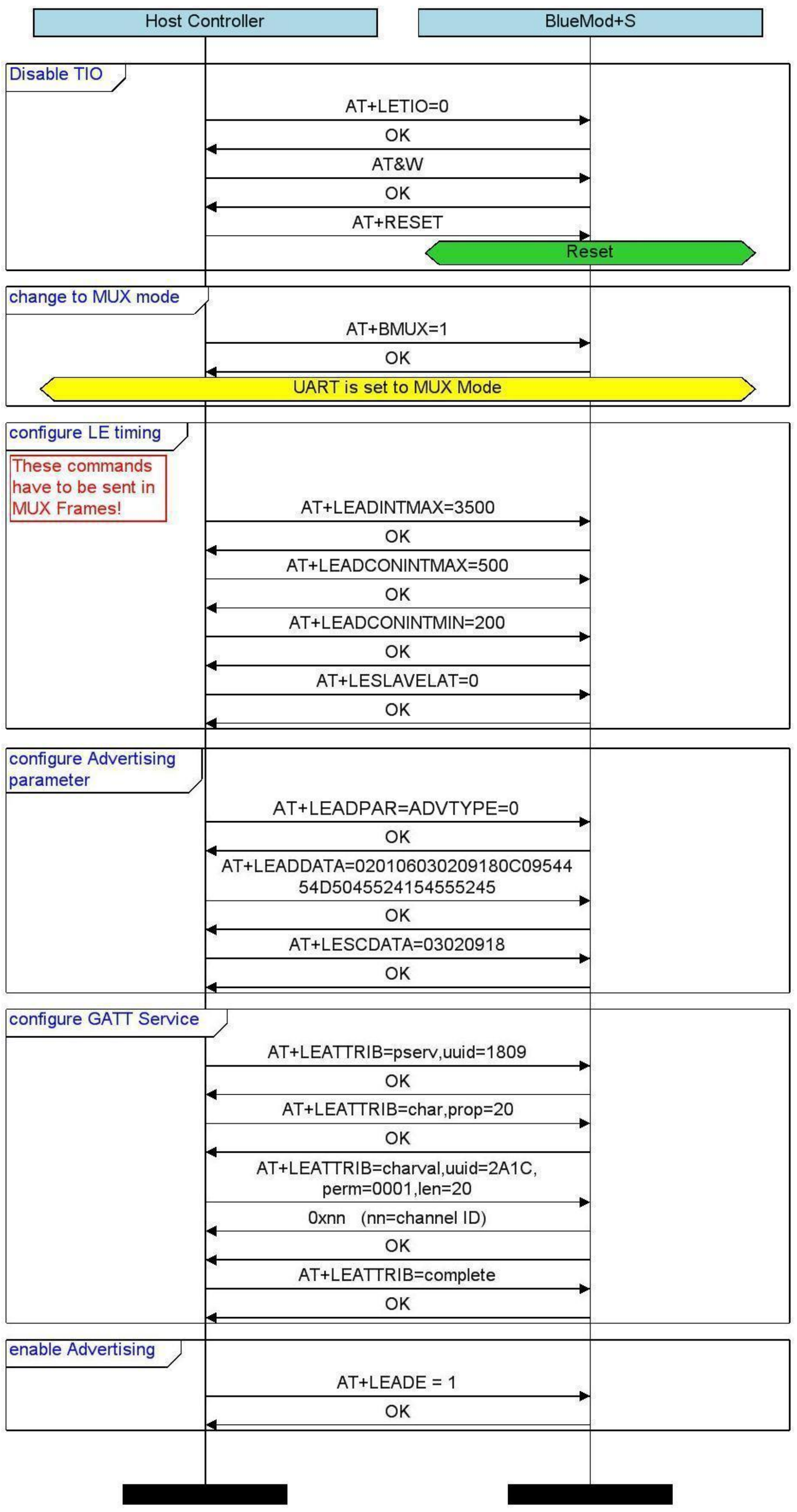
Name	Description	Length	Value
Start	Start of frame	8 bit	CC
Channel ID	Channel identifier	8 bit	00 – FF
Length	Length of data	8 bit	-

Channel ID	Description
02 … FE	Customer defined GATT services
FF	AT command interface

Example:

CC FF 06 41 54 49 39 39 0D	Send AT command “ATI99<CR>” via MUX protocol
CC FF 06 0D 0A 4F 4B 0D 0A	Receive response “<CR><LF>OK<CR><LF>” from AT command interface via MUX protocol

For an easier kind of read the required MUX header is not added in the following flow chart.



```
AT+LETIO=0
OK
AT&W
OK
AT+RESET

AT+BMUX=1
OK

AT+LEADINTMAX=3500
OK
AT+LEADCONINTMAX=500
OK
AT+LEADCONINTMIN=200
OK
AT+LESLAVELAT=0
OK

AT+LEADPAR=ADVTYPE=0
OK
AT+LEADDATA=020106030209180C09544
54D5045524154555245
OK
AT+LESCDATA=03020918
OK

AT+LEATTRIB=pserv,uuid=1809
OK
AT+LEATTRIB=char,prop=20
OK
AT+LEATTRIB=charval,uuid=2A1C,
perm=0001,len=20
0xnn   (nn=channel ID)
OK
AT+LEATTRIB=complete
OK

AT+LEADE = 1
OK
```

8.3 Test own configured GATT Service

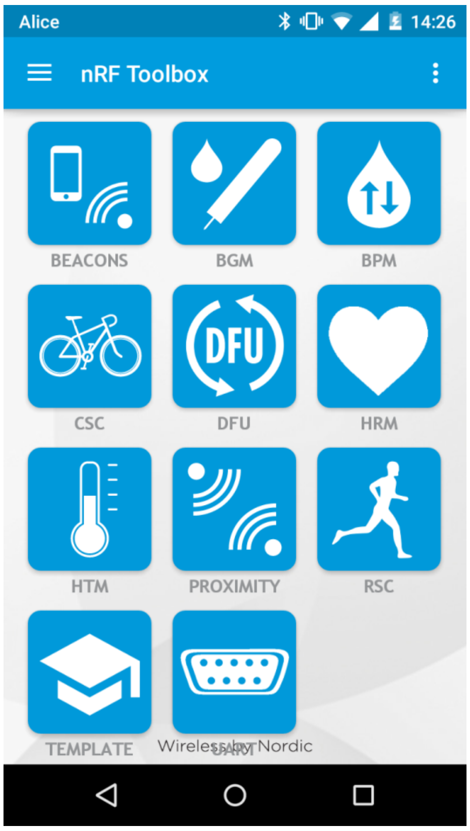
In order to test the own created GATT service there are a lot of applications available.

We will demonstrate the functionality of the own created Health Thermometer Monitor (HTM) GATT service with two appliactions:

- Nordic nRF Toolbox using HTM device
  This application will only connect to the BlueMod+S device and display the received measured value.
- Nordic nRF Master Control Panel
  This application allows detailed information of the configured BlueMod+S device (advertising data, services and characteristic information)

8.3.1 Connect from Smartphone with Nordic nRF Toolbox

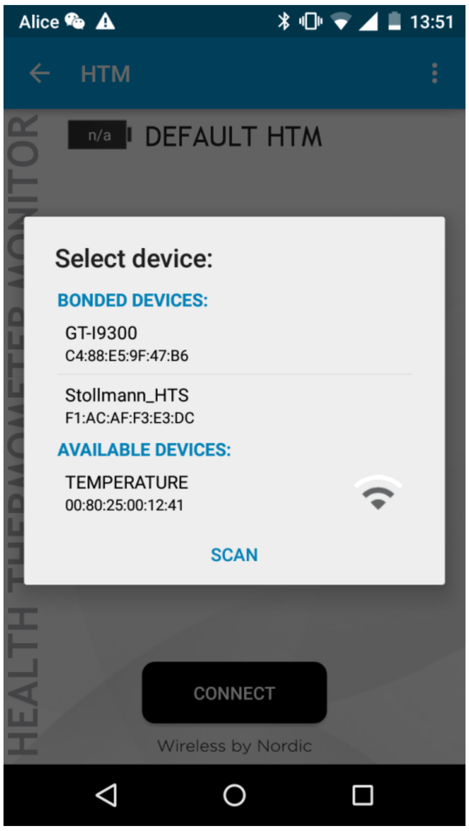
- Start the nRF Toolbox application and select the “HTM” icon to search for Health Thermometer Monitor devices.



- Start scanning

- The BlueMod+S is displayed as "TEMPERATURE" because the local name of the advertising data is set to this identifier

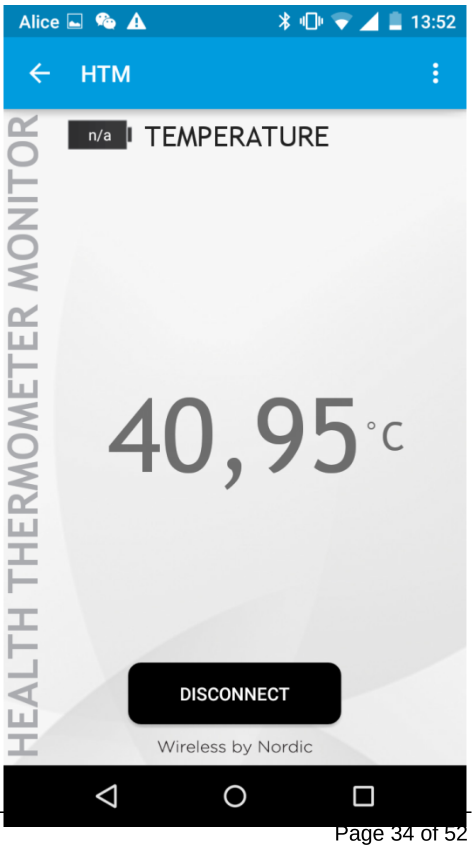
```
AT+LEADDATA=02010603020918
0C0954454D5045524154555245
```


Alice 13:51
HTM
DEFAULT HTM
Select device:
BONDED DEVICES:
GT-I9300
C4:88:E5:9F:47:B6
Stollmann_HTS
F1:AC:AF:F3:E3:DC
AVAILABLE DEVICES:
TEMPERATURE
00:80:25:00:12:41

- Press the "CONNECT" button to establish the connection to the BlueMod+S device.

- To read a value on the smartphone side it is required to send a data packet from the host controller to the BlueMod+S device.

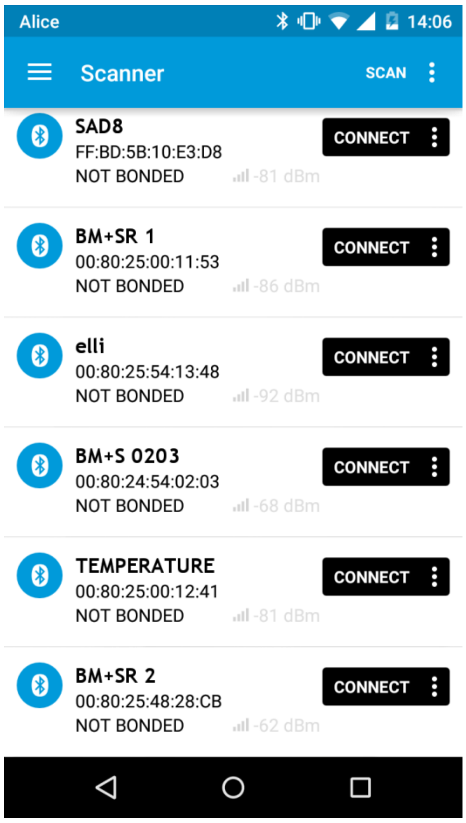
Example:
```
0xCC 02 0D 06 FF 0F 00 FE DC 07
0C 05 0B 34 2a 04 00 00 00 00
```


Alice 13:52
HTM
TEMPERATURE
40,95 °C

8.3.2 Connect from Smartphone with Nordic nRF Master Control Panel

- Start Nordic Master Control Panel application.
- This application will list all connectable devices.
- The BlueMod+S is displayed as "TEMPERATURE" because the local name of the advertising data is set to this identifier

```
AT+LEADDATA=020106030209180C0954454D5045524154555245
```


Alice 14:06
Scanner SCAN

SAD8
FF:BD:5B:10:E3:D8
NOT BONDED -81 dBm
CONNECT

BM+SR 1
00:80:25:00:11:53
NOT BONDED -86 dBm
CONNECT

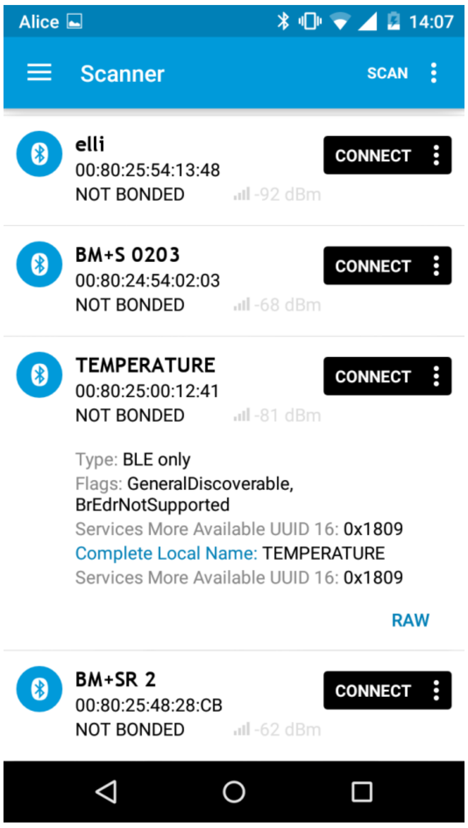
elli
00:80:25:54:13:48
NOT BONDED -92 dBm
CONNECT

BM+S 0203
00:80:24:54:02:03
NOT BONDED -68 dBm
CONNECT

TEMPERATURE
00:80:25:00:12:41
NOT BONDED -81 dBm
CONNECT

BM+SR 2
00:80:25:48:28:CB
NOT BONDED -62 dBm
CONNECT

- Reading the details of the selected device "TEMPERATURE" will display the configured advertising data.


Alice 14:07
Scanner SCAN

elli
00:80:25:54:13:48
NOT BONDED -92 dBm
CONNECT

BM+S 0203
00:80:24:54:02:03
NOT BONDED -68 dBm
CONNECT

TEMPERATURE
00:80:25:00:12:41
NOT BONDED -81 dBm
CONNECT

Type: BLE only
Flags: GeneralDiscoverable, BrEdrNotSupported
Services More Available UUID 16: 0x1809
Complete Local Name: TEMPERATURE
Services More Available UUID 16: 0x1809
RAW

BM+SR 2
00:80:25:48:28:CB
NOT BONDED -62 dBm
CONNECT

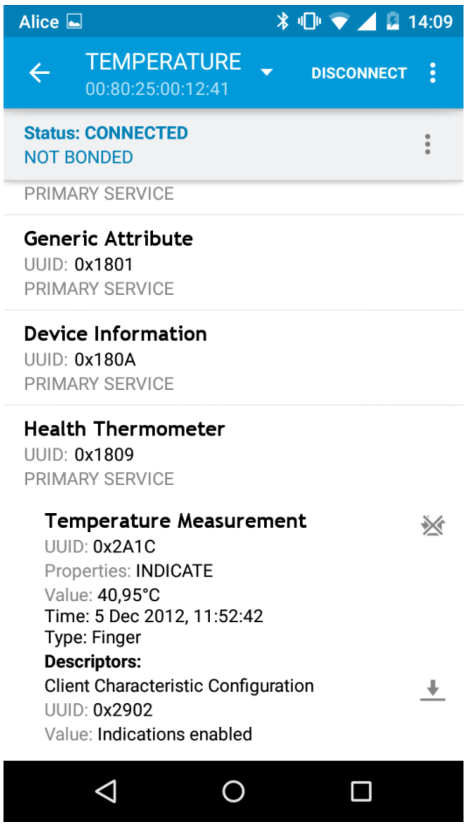
- After connecting to the BlueMod+S device the primary services will be listed in the application.

- To read a value on the smartphone side it is required to send a data packet from the host controller to the BlueMod+S device.

  Example:
  `0xCC 02 0D 06 FF 0F 00 FE DC 07 0C 05 0B 34 2a 04 00 00 00 00`

- This data content is listed as value in the appropriate service.

(For detailed information about the data content please contact Bluetooth SIG, https://www.bluetooth.org.)



Alice 14:09
TEMPERATURE
00:80:25:00:12:41
DISCONNECT
Status: CONNECTED
NOT BONDED
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Device Information
UUID: 0x180A
PRIMARY SERVICE

Health Thermometer
UUID: 0x1809
PRIMARY SERVICE

Temperature Measurement
UUID: 0x2A1C
Properties: INDICATE
Value: 40,95°C
Time: 5 Dec 2012, 11:52:42
Type: Finger
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Indications enabled

9 Firmware Upgrade

The firmware upgrade will be done locally by either

- a Stollmann provided firmware update tool. This is a Windows™ program that contains the firmware and uses a PC with a serial port for the update
- implementing the firmware update protocol on the host system.

Or over the air (feature available since firmware version V2.xxx).

9.1 Serial Firmware Upgrade

9.1.1 Prerequisites for Serial Firmware Upgrade

You need to have access to the UART interface of the BlueMod+S.

Serial firmware update requires at least the serial lines UART-RXD, UART-TXD, UART-CTS#, UART-RTS# and GND.

Serial firmware update requires a UART speed of 38400 bps.

Pin BOOT0 (E-1) shall be pulled high to access the bootloader at start-up.

9.1.2 Stollmann BlueMod+S Updater

The firmware upgrade will be done by a Stollmann provided firmware update tool. This is a Windows™ program that contains the firmware and uses a PC with a serial port for the update.

For example a firmware version 1.101 will result in the executable file "BM+S_1_101_Fwupdate.exe".

The software used for the upgrade is able to run on the following Win32/Win64 platforms:

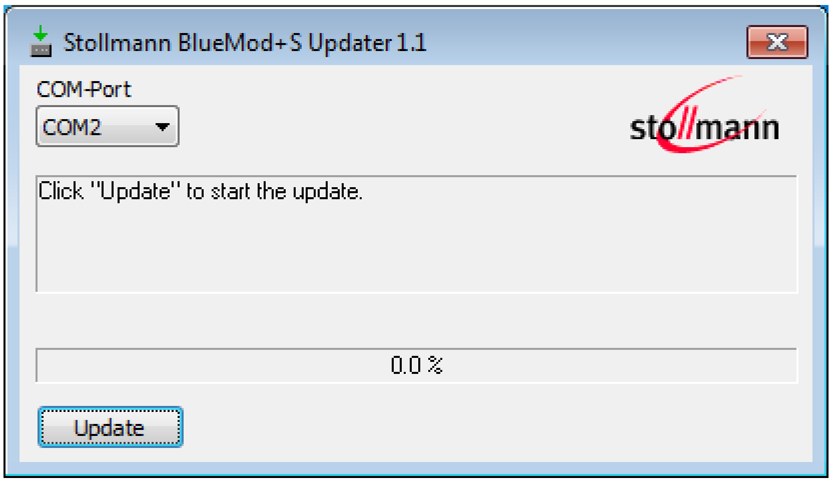
- Windows XP
- Windows Vista
- Windows 7
- Windows 8

Note: Testing was carried out on Windows 8 Pro, Windows 7 Ultimate and XP Professional platforms; however experience suggests that the described software runs on all XP platforms and all Windows 8 / 7 / Vista 32 and 64-bit platforms.

The program requires a PC with at least one free COM port.

The upload is processed via the serial port the device is attached to.

Before starting the update by pressing the "Update" button the device shall be reset.



Click "Update" to start the update.

0.0 %

- COM-Port
  The COM-Port the device is attached to
- Update
  Starts the update procedure

After the successful update close the software, remove the high level on pin BOOT0 and reset the BlueMod+S.

Note:
Do not disconnect the device while the update is in progress, otherwise the update will fail and has to be repeated. In case it is not possible to update the BlueMod+S please contact the Telit support (e-mail: ts-srd@telit.com).

9.1.3 Firmware Update Protocol on the Host System

This chapter describes the protocol layer used for firmware update over serial on the BlueMod+S.

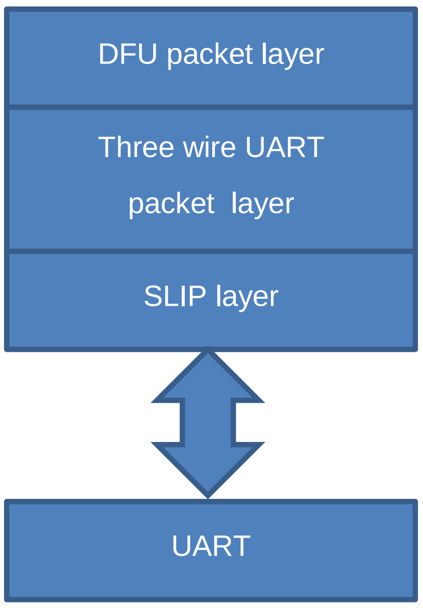
The table below contains the maximum possible binary firmware sizes:

Firmware variant	**Maximum possible binary firmware size**
V1.xxx	162 kBytes
V2.xxx	138 kBytes

Note: The actual size of each firmware binary file can be found in the firmware release notes.

9.1.3.1 Layer Structure

The device firmware update uses the HCI Three-Wire UART Transport Layer specified in [5]. Instead of HCI frames, three different DFU packets are sent to the BlueMod+S over the UART serial interface using this transport.



9.1.3.2 DFU Packet Layer Bootloader V1.000

There are three different packet types in this layer. The packet type is an unsigned 32 bit integer in LSB first order.

1.	Start packet	0x00000002
2.	Application data packet	0x00000003
3.	Stop packet	0x00000004

Start Packet

With the “Start packet” the length of the binary application image is transferred. The length of the image is an unsigned 32 bit integer in LSB first order. With a binary application image size of 17,336 bytes (0x000043B8), the packet is coded as:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x02	0x00	0x00	0x00	0xB8	0x43	0x00	0x00

Application Data Packet

With the “Application data packet” the binary application image is transferred to the BlueMod+S. The maximum packet size is 512 bytes per packet. Each packet is coded as:

Byte1	Byte2	Byte3	Byte4	Byte 5 … up to Byte 516
0x03	0x00	0x00	0x00	Max 512 bytes of binary application data

Stop Packet

When the application image has been transferred to the BlueMod+S boot loader the image must be activated. The “Stop packet” will inform the boot loader that transferring of the image has completed and the application can be started. The packet is coded as:

Byte1	Byte2	Byte3	Byte4
0x04	0x00	0x00	0x00

9.1.3.3 DFU Packet Layer Bootloader V2.003

There are four different packet types in this layer. The packet type is an unsigned 32 bit integer in LSB first order.

1. Start packet	0x00000003
2. Init packet	0x00000001
3. Application data packet	0x00000004
4. Stop packet	0x00000005

Start Packet

With the “Start packet” the DFU bootloader is informed about the kind of update and the length of the binary image. The length of the image is an unsigned 32 bit integer in LSB first order. For a binary application (type of the image to be transferred is 0x04 for an application image) with image size of 17,336 bytes (0x000043B8), the packet is coded as:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x03	0x00	0x00	0x00	0x04	0x00	0x00	0x00

Byte9	Byte10	Byte11	Byte12	Byte13	Byte14	Byte15	Byte16
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Byte17	Byte18	Byte19	Byte20
0xB8	0x43	0x00	0x00

Init Packet

The “Init packet” contains information about the application that is transferred. The DFU bootloader checks this information to determine if the image is valid for the device. Please use only the init packet provided by Stollmann. It can be found in the Stollmann delivery package (zip file). It is the contents of the file with the extension “.dat”. The binary application to load with the data packets is the file with the “.bin” extension.

For a binary application with CRC 0x0E6F the init packet is coded as:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x00	0x00	0x00	0xFF	0xFF	0xFF	0xFF

Byte9	Byte10	Byte11	Byte12	Byte13	Byte14	Byte15	Byte16
0xFF	0xFF	0xFF	0xFF	0x01	0x00	0x64	0x00

Byte17	Byte18
0x6F	0x0E

Required Waiting

After accepting the init packet the bootloader prepares (erases) the internal flash memory to accommodate the new image. The bootloader accepts no data packets during this time. Please wait 10 seconds before sending the first data packet.

Application Data Packet

With the “Application data packet” the binary application image is transferred to the BlueMod+S. The maximum packet size is 512 bytes of data + header per packet. Each packet is coded as:

Byte1	Byte2	Byte3	Byte4	Byte 5 … up to Byte 516
0x04	0x00	0x00	0x00	Max 512 bytes of binary application data

Stop packet

When the application image has been transferred to the BlueMod+S bootloader the image must be activated. The stop packet will inform the bootloader that transferring of the image has completed and the application can be started. The packet is coded as:

Byte1	Byte2	Byte3	Byte4
0x05	0x00	0x00	0x00

9.1.3.4 Three-Wire UART Packet Layer

Every packet that is sent over the Three-Wire UART Transport Layer has a packet header. It also has 16 bit CCITT-CRC at the end of the payload.

Each transport packet will contain one higher layer packet. A transport packet consists of a packet header of 4 octets, a payload of 4 to 516 octets, and a 16 bit CCITT-CRC.

The packet header consists of a sequence number of 3 bits, an acknowledge number of 3 bits, a data integrity check present bit, a reliable packet bit, a packet type of 4 bits, a payload length of 12 bits and a 8 bit header checksum.

The used packet type is vendor specific (0xe).

LSB MSB

4 Octets	1.. 156 Octets	2 Octets
Packet Header	DFU packet layer	16 bit CCITT-CRC

The detailed format description of the used packet header is found in [6].

For a detailed description of the procedural requirements of this protocol have a look at [5], chapter “Three-Wire UART Transport Layer”.

9.1.3.5 SLIP Layer

The SLIP layer performs octet stuffing on the octets entering the layer so that specific octet codes which may occur in the original data do not occur in the resultant stream.

The SLIP layer places octet 0xC0 at the start and end of every packet it transmits.

Any occurrence of 0xC0 in the original packet is changed to the sequence 0xDB 0xDC before being transmitted. Any occurrence of 0xDB in the original packet is changed to the sequence 0xDB 0xDD before being transmitted. These sequences, 0xDB 0xDC and 0xDB 0xDD are SLIP escape sequences.

For a detailed description of this protocol have a look at [5], chapter “Three-Wire UART Transport Layer”.

9.2 Firmware Update Over The Air (OTA)

Since firmware version V2.xxx the BlueMod+S supports firmware over the air update. The firmware update over the air can be performed by using the Nordic nRF ToolBox app available for iOS and Android or by using the Nordic Master Control Panel and the corresponding Nordic Bluetooth hardware.

The firmware over the air update in the BlueMod+S will be enabled with the commands below:

- AT+DFUMODE=2
- AT+DFUSTART

After sending the AT+DFUSTART command the BlueMod+S is visible in the air as “BM+S_DFU” (name configured with command AT+DFUNAME) for a time period of 2 minutes. If no firmware update is performed during this time the BlueMod+S will continue with normal operation.

Alternatively to AT+DFUSTART the following procedure can be used to enable the firmware over the air update in the BlueMod+S:

- AT+DFUMODE=2
- AT&W
- Set pin BOOT0 to high level
- AT+RESET

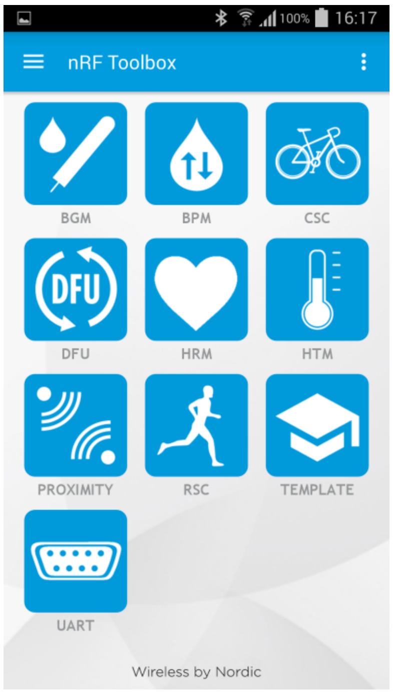
With the above settings the BlueMod+S stays in firmware over the air update mode as long as pin BOOT0 has high level (no 2 minutes timeout).

The following chapter describes the firmware over the air update by using the Nordic nRF Toolbox app on Android.

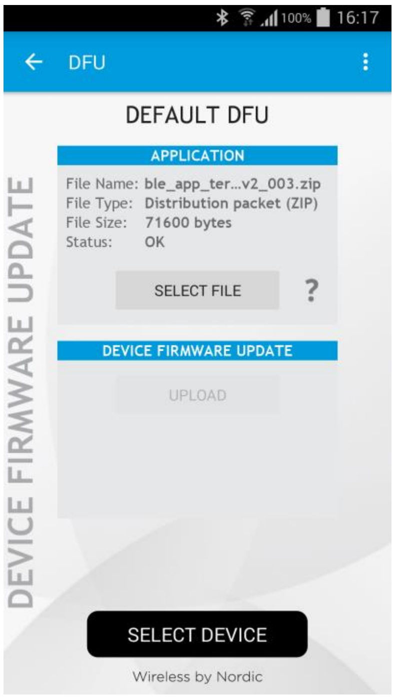
9.2.1 Firmware Update Over The Air using Nordic nRF Toolbox on Android

Make sure the BlueMod+S has already activated the firmware over the air update.

Open the nRF ToolBox app on the smartphone and choose “DFU“.

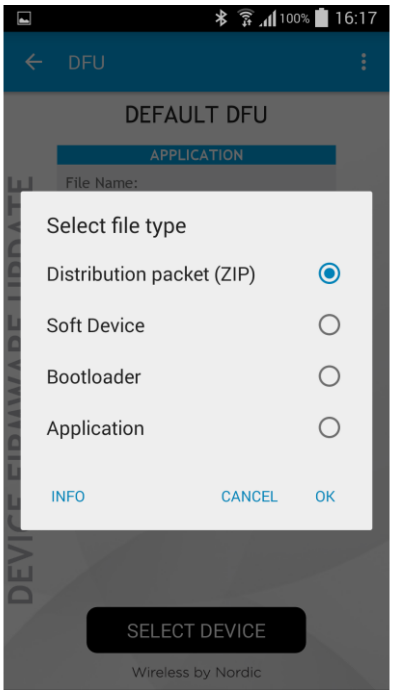


Press the button “SELECT FILE”.



File Name: ble_app_ter...v2_003.zip
File Type: Distribution packet (ZIP)
File Size: 71600 bytes
Status: OK

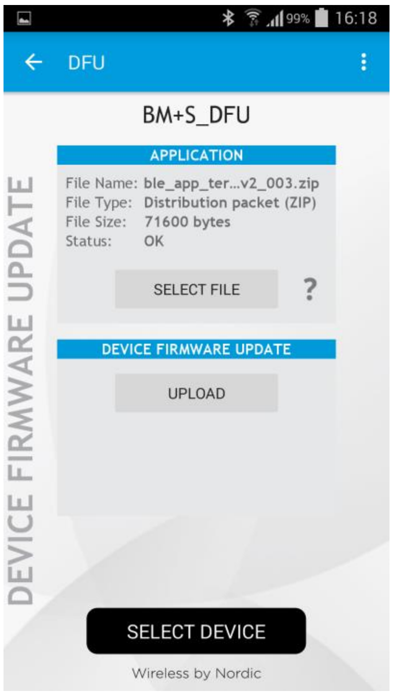
Select file type “Distribution packet (ZIP)”.



Select file type

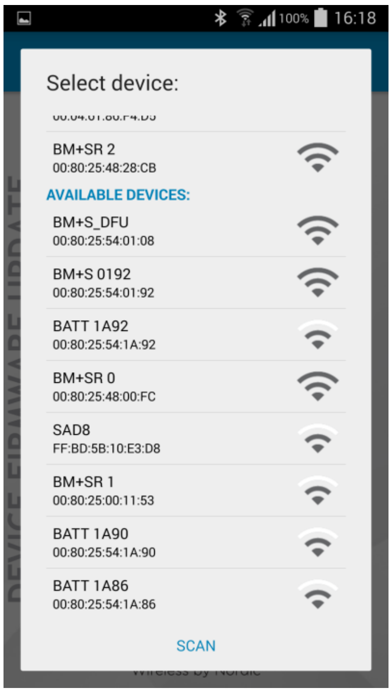
Distribution packet (ZIP) ●
Soft Device ○
Bootloader ○
Application ○

Search via file manager for the firmware package which was previously copied to the smartphone (e.g. ble_app_terminal_io_v2_003.zip in the example below).



File Name: ble_app_ter...v2_003.zip
File Type: Distribution packet (ZIP)
File Size: 71600 bytes
Status: OK

Press the button “SELECT DEVICE” and select the “BM+S_DFU” from the list of available devices.


Select device:

BM+SR 2
00:80:25:48:28:CB

AVAILABLE DEVICES:

BM+S_DFU
00:80:25:54:01:08

BM+S 0192
00:80:25:54:01:92

BATT 1A92
00:80:25:54:1A:92

BM+SR 0
00:80:25:48:00:FC

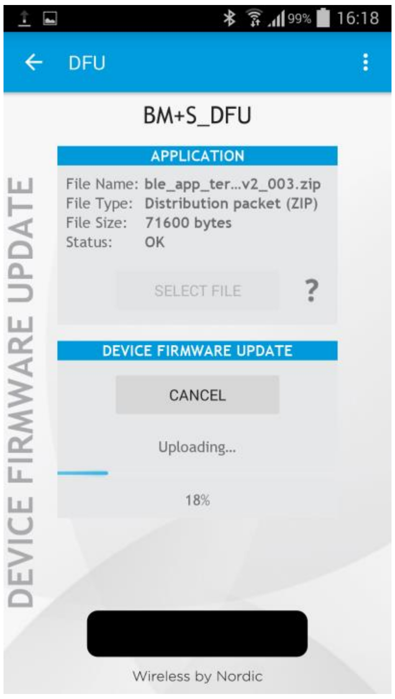
SAD8
FF:BD:5B:10:E3:D8

BM+SR 1
00:80:25:00:11:53

BATT 1A90
00:80:25:54:1A:90

BATT 1A86
00:80:25:54:1A:86

Press the “UPLOAD” button to upload the firmware package over the air to the BlueMod+S.



BM+S_DFU

APPLICATION

File Name: ble_app_ter...v2_003.zip
File Type: Distribution packet (ZIP)
File Size: 71600 bytes
Status: OK

Uploading...
18%

After the file was uploaded successfully the BlueMod+S will start with the new firmware.

10 System OFF Mode

Since firmware version V2.xxx the BlueMod+S supports the possibility to set the module into low power mode during the time the module is not used with the AT+SYSTEMOFF command.

The BlueMod+S will restart on activity at the GPIO input lines UART-RTS#, IUR-IN# or GPIO[4].

The host controller can use the IOA pin (GPIO[8]) to monitor the system status. Please also verify the configuration of the AT+IOACFG parameter.

It is also possible to monitor the UART flow control line UART-RTS#.

10.1 Using System OFF Mode for Terminal I/O

The following example will list the communication between the host controller and the BlueMod+S using the integrated Terminal I/O profile.

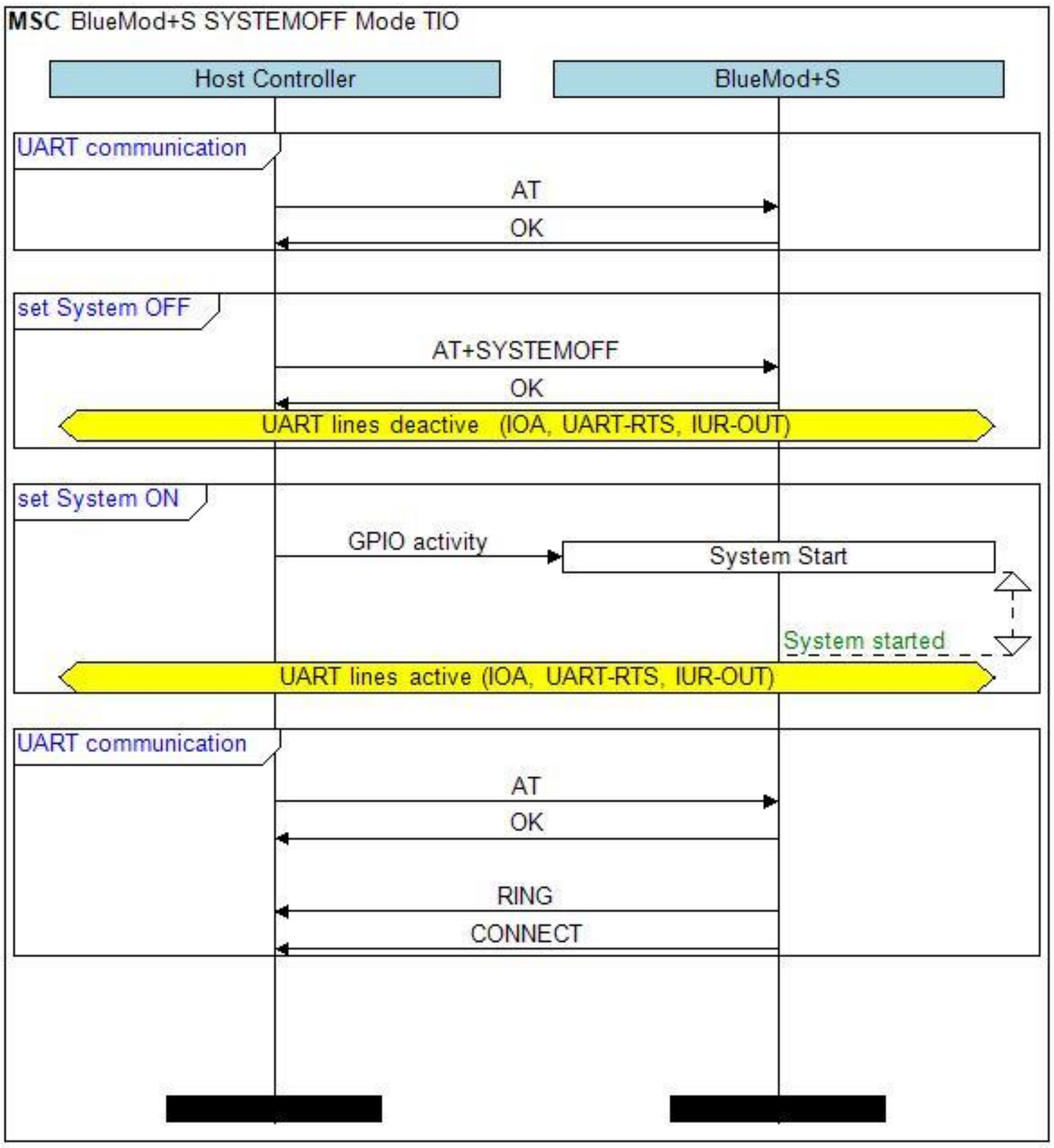
To set the BlueMod+S into the low power mode the host controller needs to send the AT+SYSTEMOFF command.
The BlueMod+S will respond “OK” before changing into low power mode.

To activate the BlueMod+S from low power mode the host controller needs to activate one of the following GPIO lines: UART-RTS#, IUR-IN#, GPIO[4]

The module detects the GPIO change and starts the firmware.

After the firmware is started the host can continue the UART communication.

An incoming call is reported with RING and CONNECT.



10.2 Using System OFF Mode for an own configured GATT Service

It is also possible to use the low power mode in an own custom specific GATT server configuration.

In this case the custom specific GATT server service set needs to be stored with the AT+LESRVSETOPEN and AT+LESRVSETSAVE commands.

The stored GATT server service can be activated by using the AT+LESRVBOOTSET or AT+LESRVSETACT command.

Please note that the values within the configured characteristics are not stored over the low power mode.

11 Related Documents

[1] BlueMod+S AT Command Reference

[2] BlueMod+S Startup Timing

[3] UICP+ UART Interface Control Protocol

[4] BlueMod+S Hardware Reference

[5] Bluetooth 4.0 Core Specification

[6] nRf51 SDK Documentation
https://devzone.nordicsemi.com/documentation/nrf51/5.2.0/html/a00028.html

[7] Using the BlueMod+S as a Beacon

12 History

Version	Release Date	By	Change description
r01	18.11.2014	ta	First release
r02	30.01.2015	ta	Added description about firmware update protocol on the host system and maximum firmware size
r03	09.07.2015	bs, ta	Added new chapter “GATT Configuration” Changes in chapter UICP: Schematics of state changes, added connection example Added new chapter “System OFF Mode” Added new chapter “Firmware Update Over The Air” Added firmware dependencies of V1.xxx and V2.xxx in chapter “Introduction”
r04	26.10.2015	ta	Added note in UICP chapter that all data received before interface up state has been achieved shall be discarded Added changes between bootloader V1.000 and V2.003 for Firmware Update Protocol on the Host System Added startup timing in the document Added maximum possible binary firmware size of firmware version 2.xxx Added required UART speed for serial firmware update
r05	29.04.2016	bs ta	Add startup timing of firmware versions 1.007 and 2.006 Added new chapter “Initial Configuration”
r06	24.05.2016	bg	Telit cover page added.

Telit Wireless Solutions GmbH
Mendelssohnstraße 15 D
22761 Hamburg
Germany

Phone: +49 (0)40 890 88-0
Fax: +49 (0)40 890 88-444
E-mail: ts-srd@telit.com
www.telit.de

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil

Telit reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. The information contained herein is provided "as is". No warranty of any kind, either express or implied, is made in relation to the accuracy, reliability, fitness for a particular purpose or content of this document. This document may be revised by Telit at any time. For most recent documents, please visit www.telit.com

Copyright © 2016, Telit

[04.2016]