SW User Guide

1VV0301278, Rev. 09 – 2016-05-26

TELIT
TECHNICAL
DOCUMENTATION

Stollmann is a Telit brand.

BlueMod+SR

User Guide

Release r09

Table of contents

1 Introduction

This document describes the usage of the BlueMod+SR Bluetooth module featuring software version V1.400 or later.

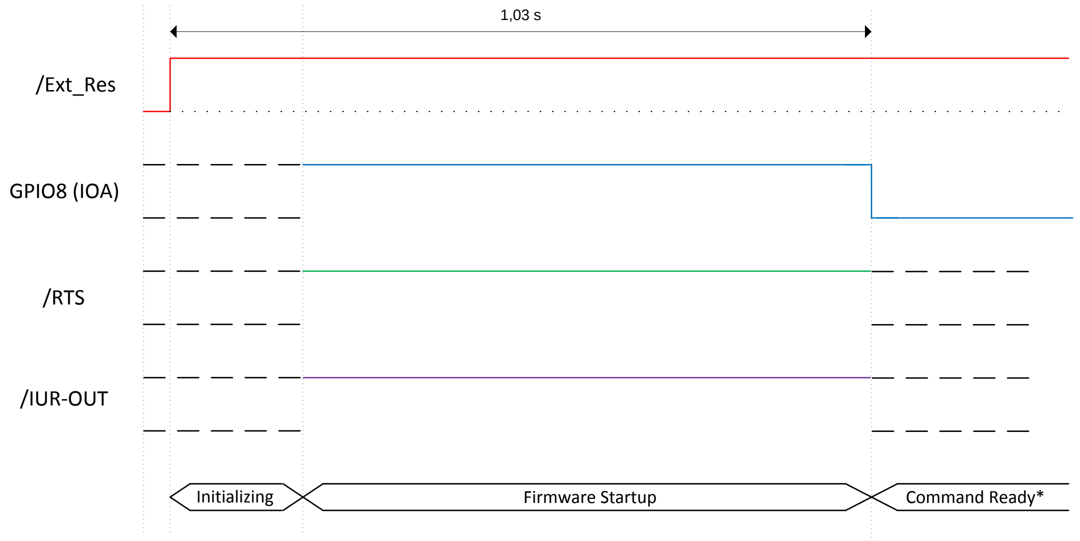
For a detailed description of the commands refer to the *BlueMod+SR AT Command Reference [1]*.

All referenced documents can be downloaded from:

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

2 Startup Timing

The following diagram shows the startup timing of the BlueMod+SR based on SPP firmware version 1.531.



(*) The firmware is command ready 1.03 seconds after the reset has been released and when GPIO8 (IOA) and /RTS are 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]*.

3 Pairable and Bondable Mode

In general we destinguish 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), LE Mode: BlueMod+SR advertises TIO as “functional” BR Mode: BlueMod+SR reject authentication request / pairing
1	Pairing, LE Mode: BlueMod+SR advertises TIO as “bondable and functional” BR Mode: BlueMod+SR accept authentication request / 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.

4 Security

For security reasons it is necessary to be able to recognize other Bluetooth devices and control the access to the local Bluetooth device.

The “Secure Simple Pairing” (SSP) is the headline feature of Bluetooth 2.1 and the improved experience of the pairing procedure.

SSP is mandatory for Bluetooth 2.1 devices and cannot be switched off.

A Bluetooth 2.1 device may only use “legacy pairing” to interoperate with a Bluetooth 2.0 or earlier device.

The pairing process can be triggered from the user to create a bond (AT+BND) or automatically when connecting to a service of another Bluetooth device (ATD…).

SSP is configurable using the parameters for I/O capabilities (AT+BIOCAP) and a man in the middle protection (AT+BMITM).

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

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 SSP.

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

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.

4.1 Classic Bluetooth (BR/EDR)

When using Classic Bluetooth SSP 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.
Compared to the legacy pairing with a fixed PIN, the security level provided by this method is much higher.

- **Numeric Comparison:**

If both devices have a display and both sides can accept a “Yes/No” user input, they may use Numeric Comparison. This method displays a six digit numeric code on each device. The user shall compare the numbers to ensure they are identical. If the comparison succeeds, the user(s) shall confirm pairing on the device(s) that can accept an input.
This method provides MITM protection, assuming the user confirms on both devices and actually performs the comparison properly.

- **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.
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.

- **Legacy Pairing: (Bluetooth 2.0 compatible pairing mechanism)**

In this case both devices needs to enter the same PIN code of minimum four digits. The BlueMod+SR uses the PIN code saved in the parameter AT+BPIN (compare: *BlueMod+SR AT Command Reference [1]*).

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+SR and the DTE are listed.

Remote device / BM+SR	Display only	Display Yes/No	Keyboard only	No input no output
Display only AT+BIOCAP=0	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*	Numeric comparison (both displayed, one automatic confirm) *No MITM protection*	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>	Just Works (Numeric comparison, both automatic confirmation) *No MITM protection*
Display Yes/No AT+BIOCAP=1	Numeric comparison (both displayed, one automatic confirm) *No MITM protection*	Numeric comparison (both displayed, both confirm) *MITM protection* SSPCONF <BT addr> <passkey> ? AT+BSSPCONF <BT addr>, 1	Passkey entry (one display, one input) *MITM protection* SSPPIN <BT addr> <passkey>	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*
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 (numeric comparison, both automatic confirmation) *No MITM protection*
No input no output AT+BIOCAP=3	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*	Just Works (numeric comparison, both automatic confirmation) *No MITM protection*

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

The following flow charts will give an example for the different SSP authentication methods “just works”, “numeric comparision” and “passkey entry” within an active outgoing call request from the BlueMod+SR.

Onto these flow charts the local Bluetooth device (*BT device A*) and the destination side (*BT device B*) are configured with a BlueMod+SR. The destination can be changed to each other Bluetooth 2.1 device.

The “*Application*” part will simulate the device at the end (DTE) which communicates to the local Bluetooth device with configuration commands.

The box called “*AIR*” will signal which part of communication will be transmitted over Bluetooth to the destination side.

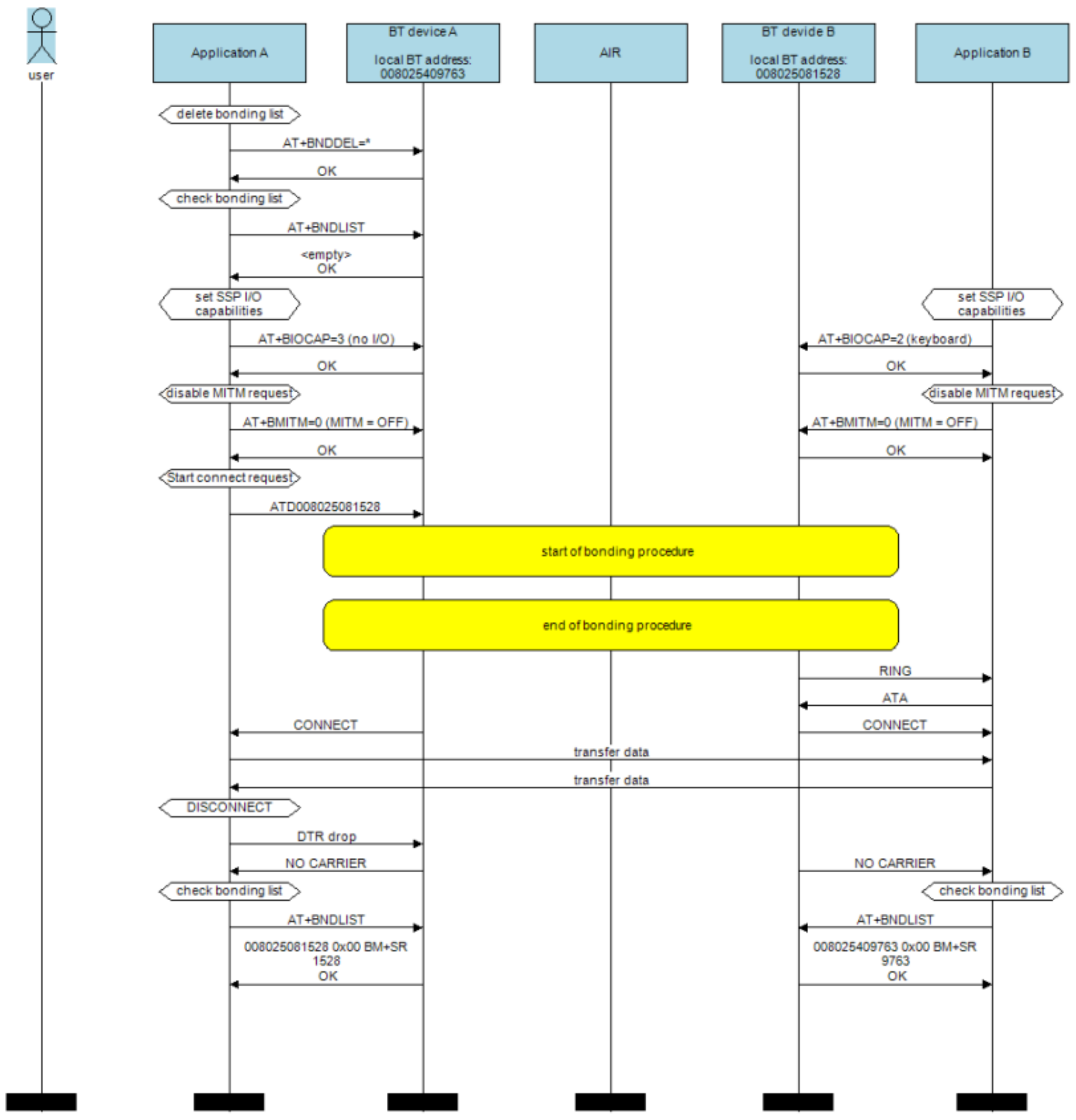
The interesting part of the bonding procedure is placed between the yellow boxes “*start of bonding procedure*” and “*end of bonding procedure*”.

All serial commands between the “*Application A/B*” and the “*BT device A/B*” out of the bonding procedure are used for further configuration of SSP.

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

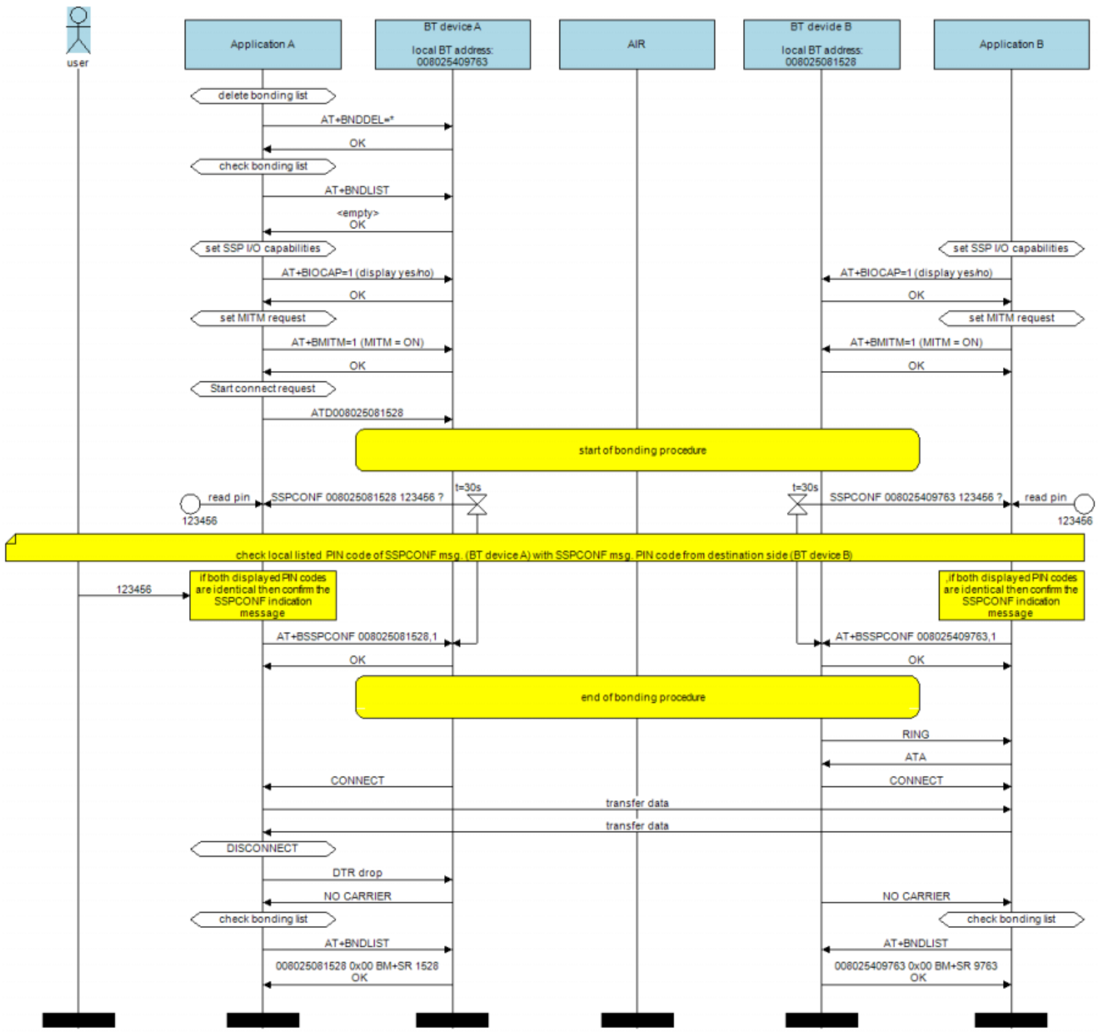
4.1.1 Connection Example “Just Works”

with I/O capabilities combination “no I/O” and “keyboard”



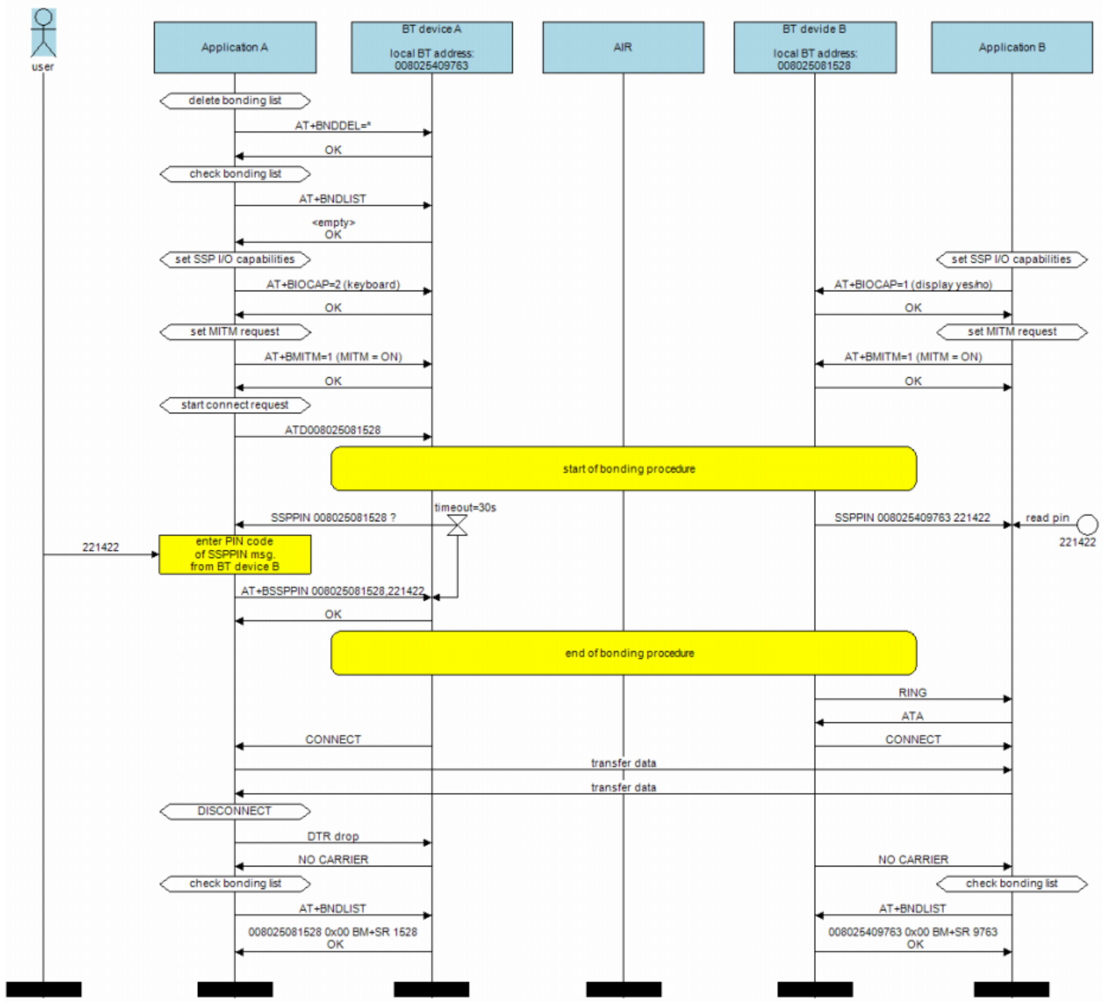
4.1.2 Connection Example “Numeric Comparison”

with I/O capabilities combination “display yes/no” on both sides


user
Application A
BT device A local BT address: 008025409763
AIR
BT devide B local BT address: 008025081528
Application B
delete bonding list
AT+BNDDEL=*
OK
check bonding list
AT+BNDLIST
<empty> OK
set SSP I/O capabilities
AT+BIOCAP=1 (display yes/no)
OK
set MITM request
AT+BMITM=1 (MITM = ON)
OK
Start connect request
ATD008025081528
start of bonding procedure
read pin
123456
SSPCONF 008025081528 123456 ?
t=30s
SSPCONF 008025409763 123456 ?
check local listed PIN code of SSPCONF msg. (BT device A) with SSPCONF msg. PIN code from destination side (BT device B)
if both displayed PIN codes are identical then confirm the SSPCONF indication message
AT+BSSPCONF 008025081528,1
AT+BSSPCONF 008025409763,1
end of bonding procedure
RING
ATA
CONNECT
transfer data
DISCONNECT
DTR drop
NO CARRIER
008025081528 0x00 BM+SR 1528 OK
008025409763 0x00 BM+SR 9763 OK

4.1.3 Connection Example “Passkey Entry”

with I/O capabilities combination “keyboard” and “display yes/no”



4.2 Bluetooth Low Energy (BLE)

When using Bluetooth Low Energy SSP 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.

- **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.
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+SR and the DTE are listed.

With Bluetooth Low Energy it is also possible to allow Terminal I/O connections without security procedures. This can be achieved by setting parameter AT+LETIO=2.

BM+SR \ Remote device	**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+SR output message	SSPPIN <BT addr> ? (example)
Blue color:	BM+SR 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 Telit’s Terminal I/O Utility app to the BlueMod+SR (see also the connection example in chapter 5.6.1).

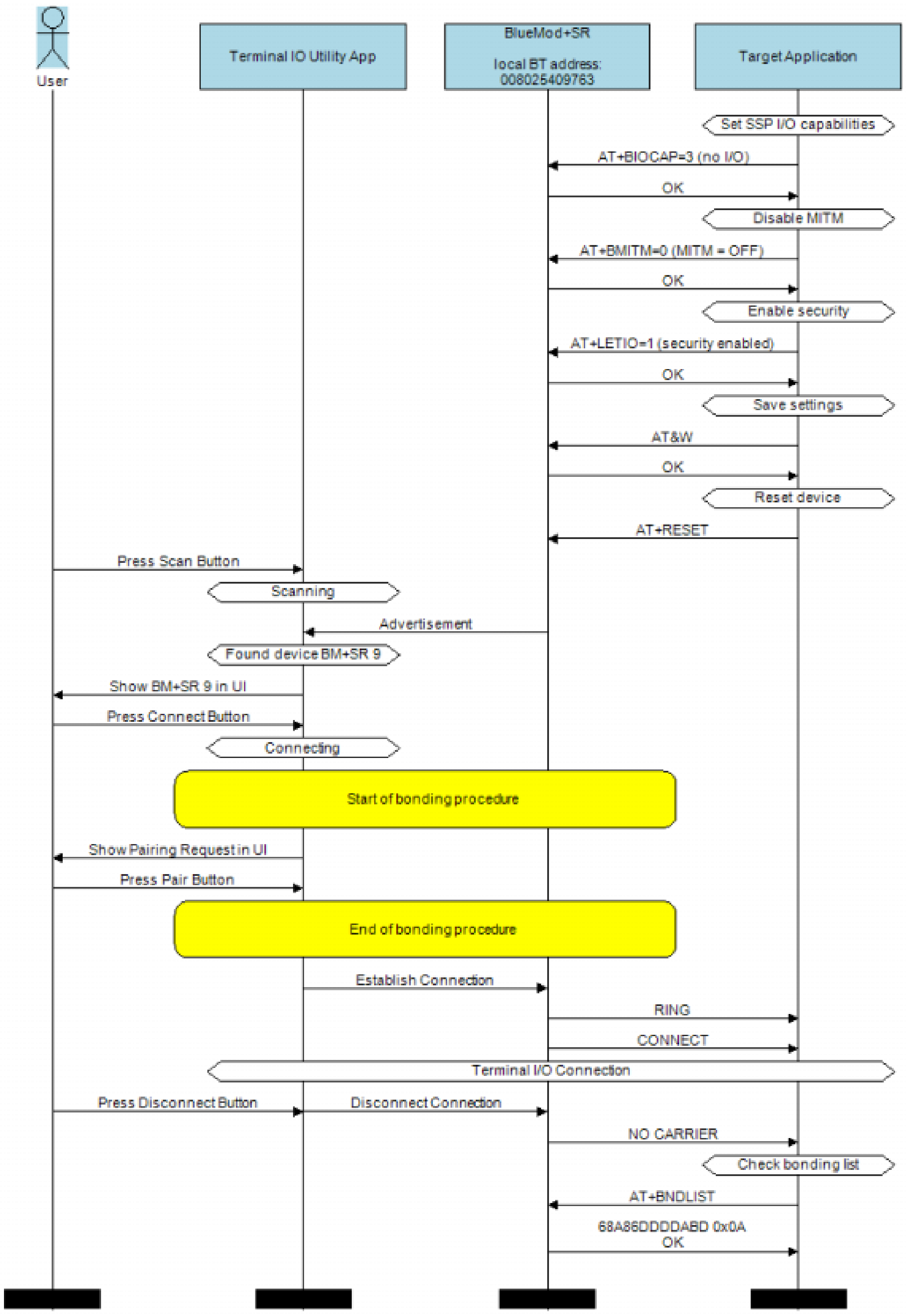
The “*Target Application*” part will simulate the device at the end (DTE) which communicates to the BlueMod+SR with configuration commands.

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+SR*” out of the bonding procedure are used for further configuration of SSP.

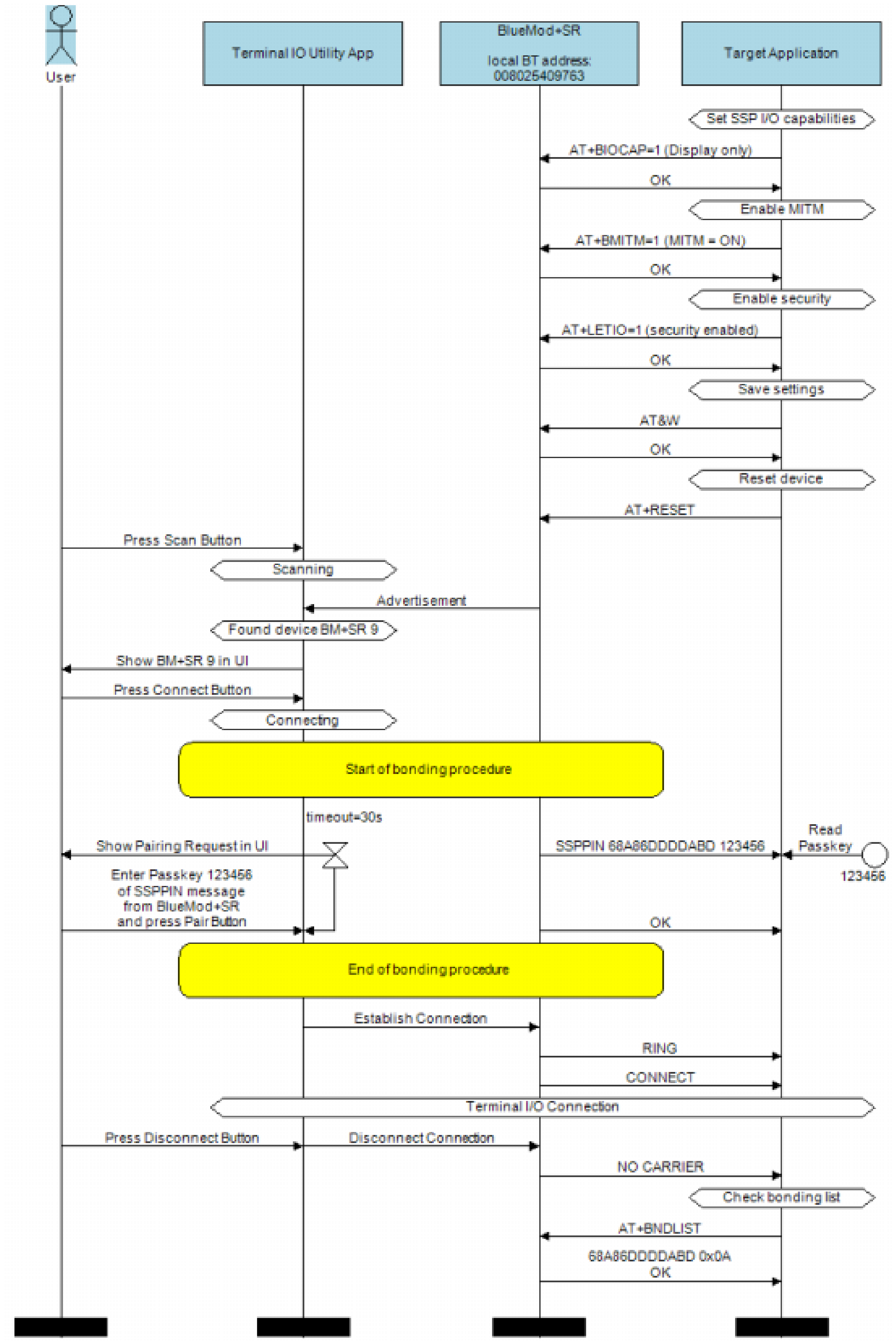
The configuration commands and responses within the flow charts are described in the *BlueMod+SR AT Command Reference [1]*.

4.2.1 Connection Example Terminal I/O “Just Works”



4.2.2 Connection Example Terminal I/O “Passkey Entry”

with I/O capabilities “display only”



5 Terminal I/O over Bluetooth Low Energy

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

5.1 GAP role

BlueMod+SR supports the two GAP roles peripheral and central.

A peripheral is a device that advertises by using connectable advertising packets. Normally it becomes a slave once connected.

A central is a device that initiates connections to peripherals. Normally it becomes master when connected.

The GAP role needs to be set according to the required connection scenario.

To set the GAP role use the parameter LEROLE described in the *BlueMod+SR AT Command Reference [1]*.

5.2 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+SR AT Command Reference [1]*.

5.3 Connection Interval

The connection interval is a time that determines how often 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+SR AT Command Reference [1]*.

5.4 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+SR AT Command Reference [1]*.

5.5 Local Device Name

To set the local device name, use the parameter BNAME described in the *BlueMod+SR 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.

In the LE SCAN_RESP message the length of the name is truncated to 7 bytes, due to the maximum size of the data packet (there are insufficient bytes available to include the full name).

In addition to the name delivered in the advertising the BlueMod+SR 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.

5.6 Connection Examples

This chapter describes some Terminal I/O connection scenarios using BlueMod+SR.

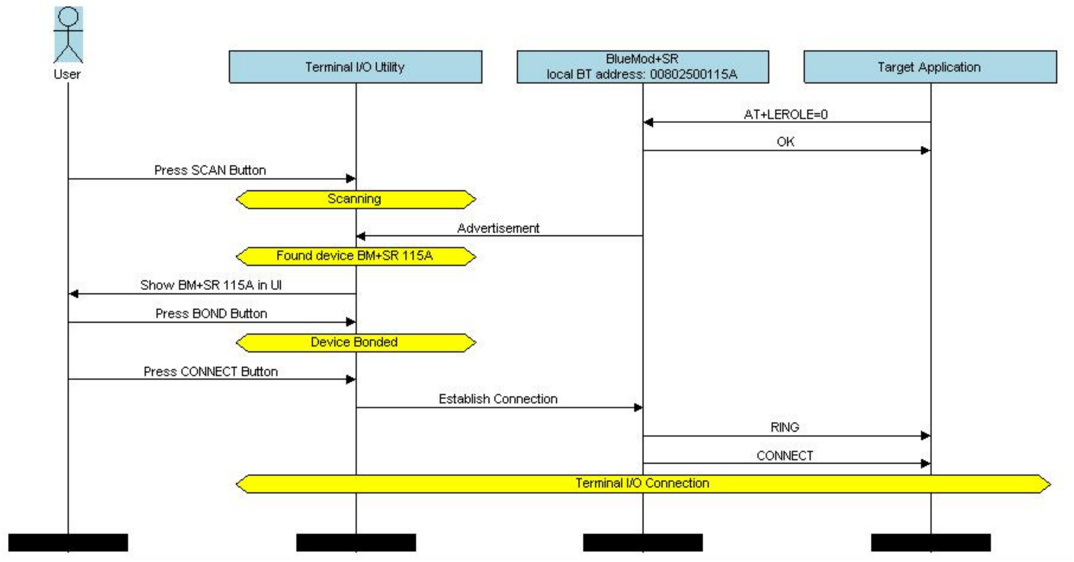
5.6.1 iPhone to BlueMod+SR

To establish a Bluetooth Low Energy connection from the iPhone to the BlueMod+SR the "Terminal IO Utility" App from Telit 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.

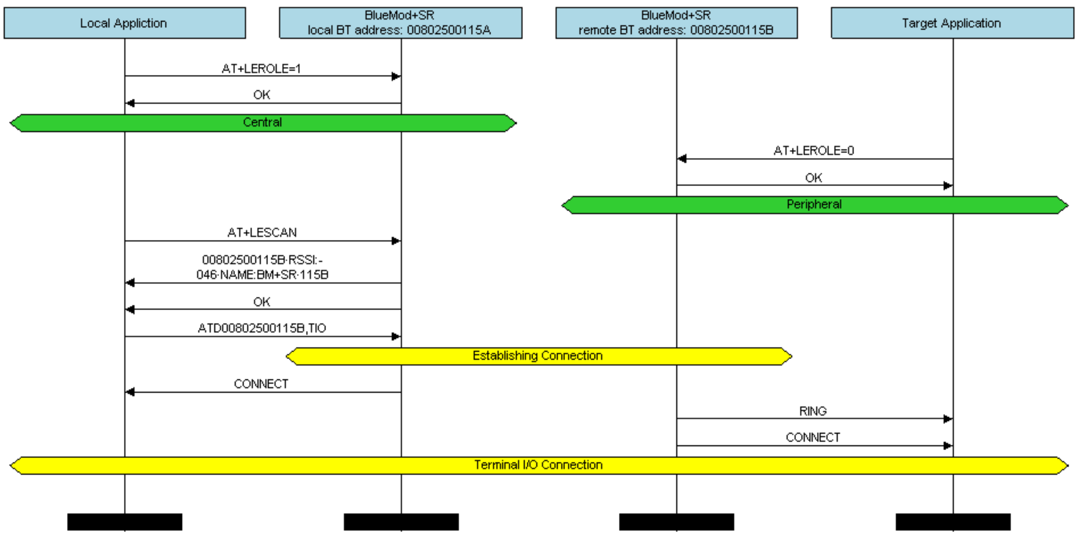
Note: The "Terminal IO Utility" App provided in the Apple store supports Terminal I/O V2 and requires firmware version 1.4xx and higher. It is not compatible with older firmware version 1.3xx and lower.



The BlueMod+SR has to be set to peripheral role because the Terminal IO Utility App on the iPhone has the central role.

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

5.6.2 BlueMod+SR to BlueMod+SR



The local device is configured as central role, the remote device is configured as peripheral role.

The local device scans for peripheral devices and establishes the connection with the command ATD.

As soon as the connection is established data can be sent from local to remote BlueMod+SR and vice versa.

6 UART Interface Control Protocol (UICP)

6.1 General Protocol Description

Telit’s 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.

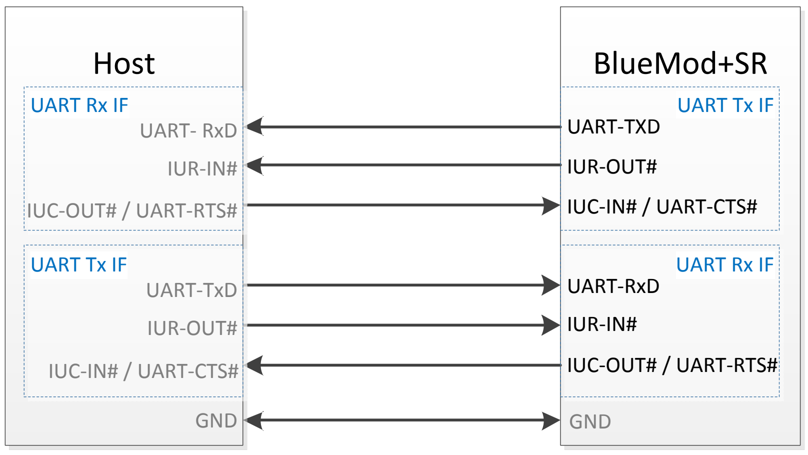
6.2 Requirements of Using UICP on BlueMod+SR

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+SR 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+SR the configuration parameter AT+UICP=1 needs to be set.

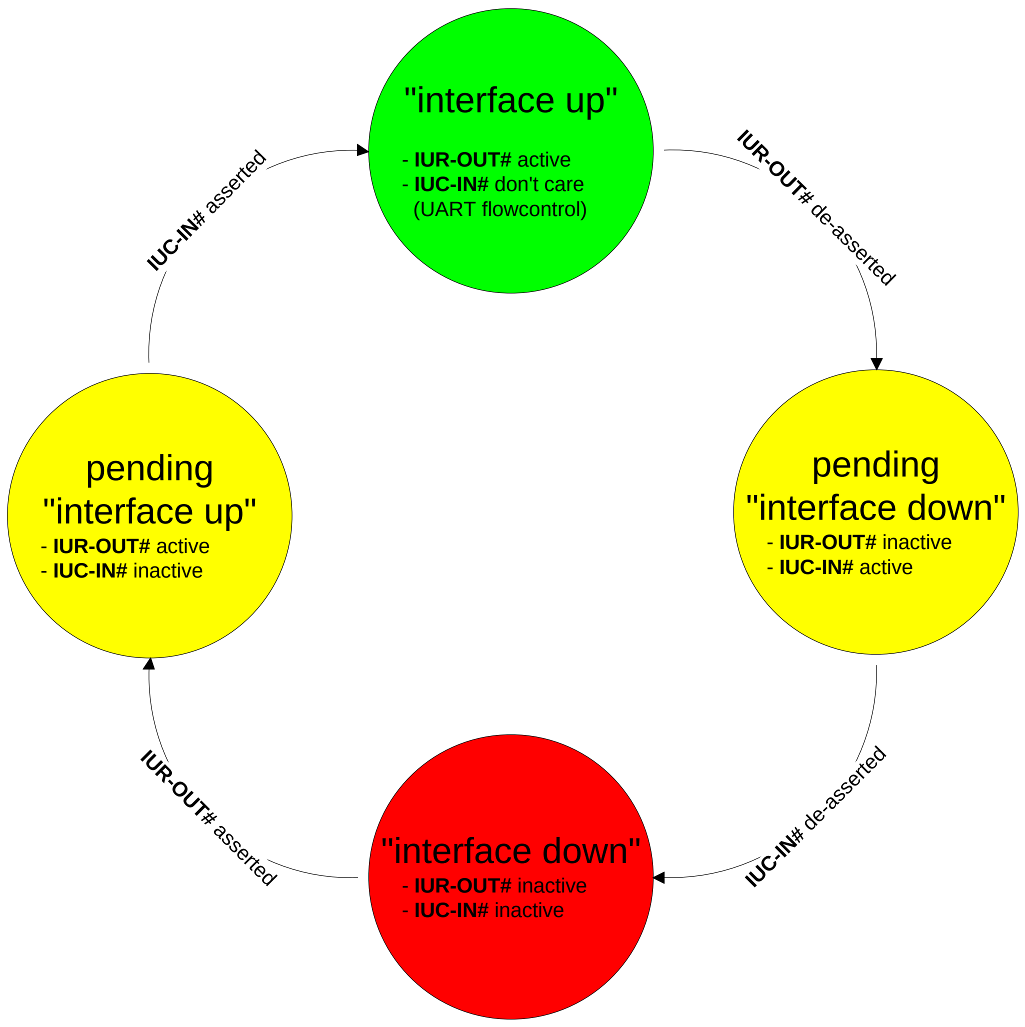
6.3 Connection Example between BlueMod+SR and Host Controller



Further information about the BlueMod+SR UART interface is described in the document *BlueMod+SR Hardware Reference [5]*.

6.4 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

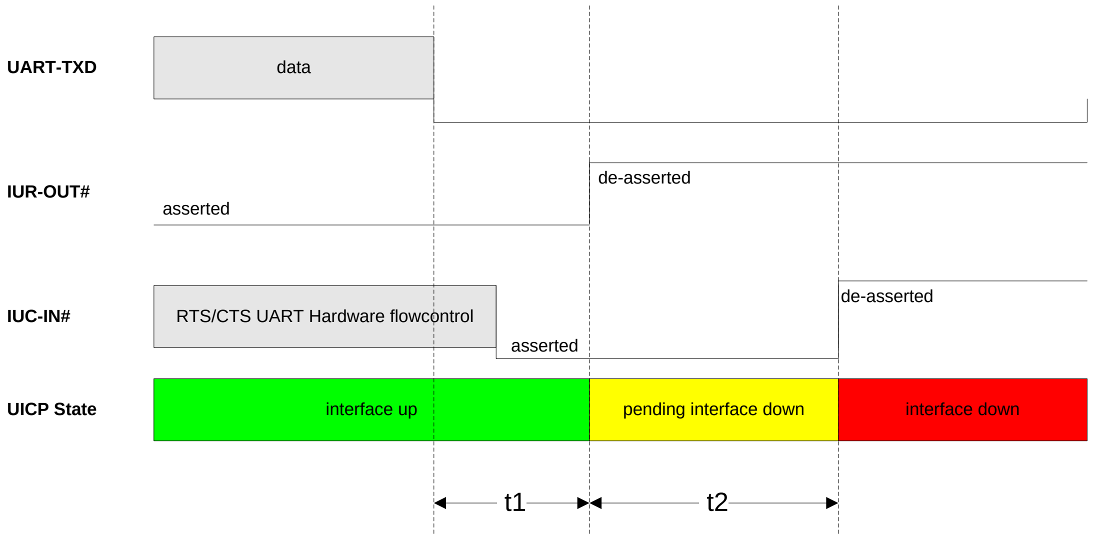
Note: All data received before the interface up state has been achieved shall be seen as invalid data and shall be discarded.

6.4.1 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

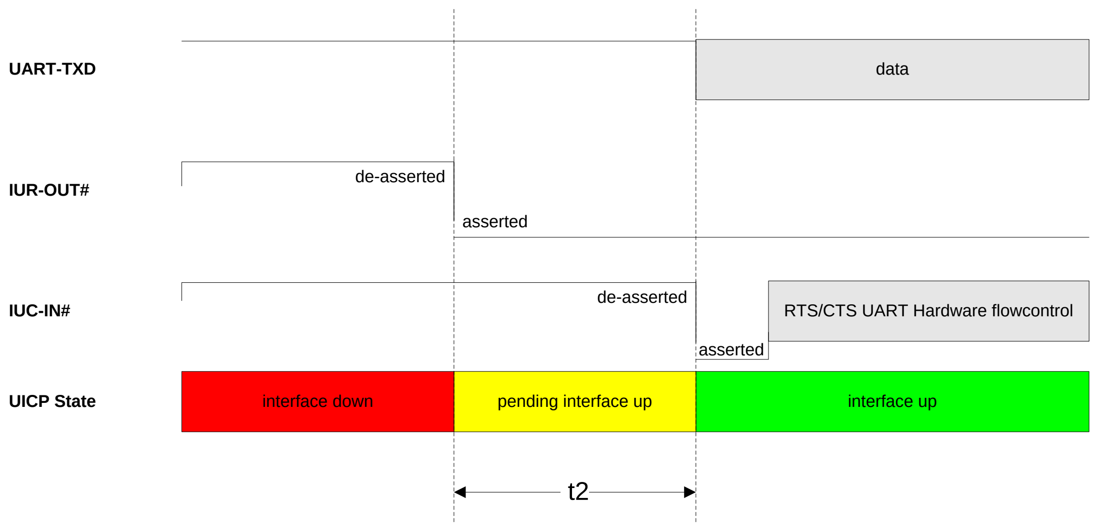
6.4.2 Drive from "interface down" to "interface up" State

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.



t1 >= 100ms (see this chapter)

t2 < 1s

6.5 Example of UICP Usage

The following examples shows the state change between the BlueMod+SR 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.

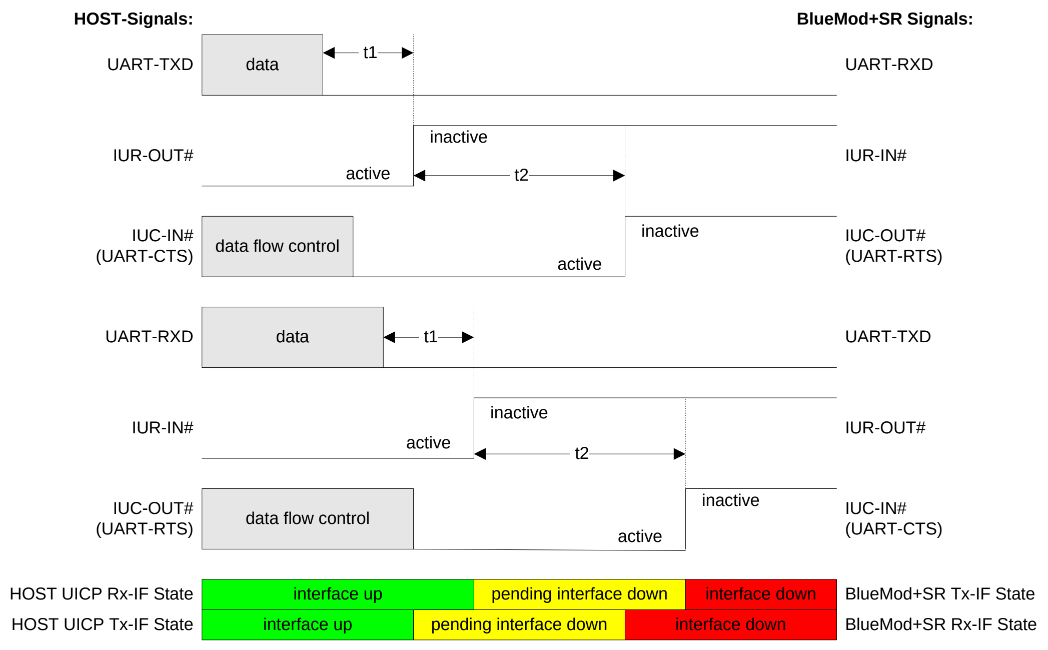
6.5.1 State Change from "interface up" to "interface down"

Host and BlueMod+SR 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+SR that it is allowed to go to “interface down” state by de-asserting IUR-OUT# signal.

Parallel to that UICP signalling from host to BlueMod+SR the BlueMod+SR 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+SR 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 signalling sequences proceed in parallel until host and BlueMod+SR interfaces are in “interface down” state.



6.5.2 State Change from "interface down" to "interface up"

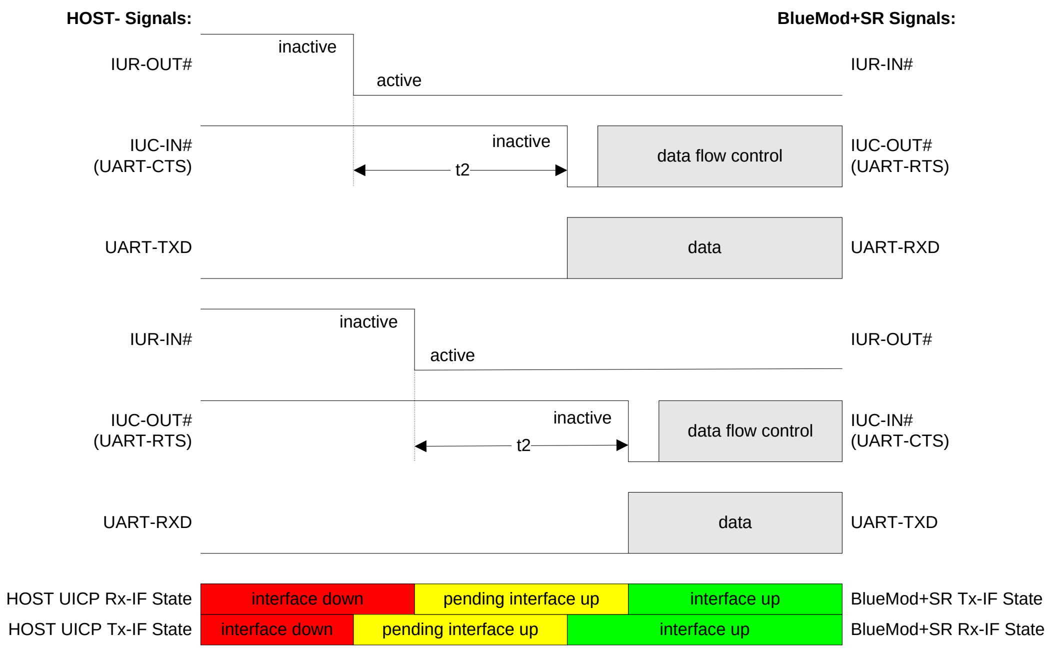
Host and BlueMod+SR are in the state “Interface down” and may have the MCU into some kind of power saving state.

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

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

The host and the BlueMod+SR 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”.

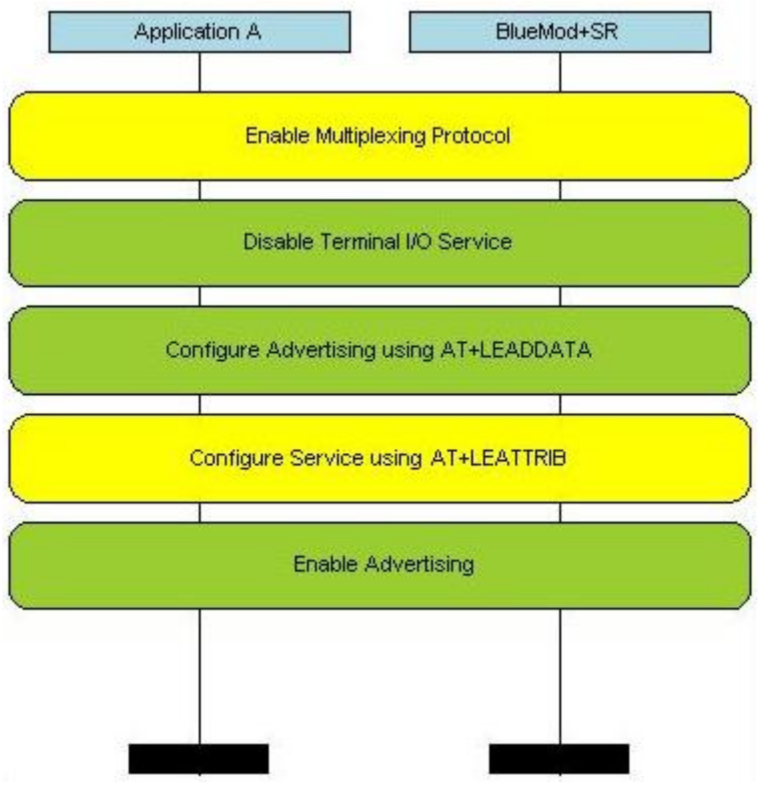
Both UICP signalling sequences proceed in parallel until host and BlueMod+SR interfaces are in “interface up” state and data can be exchanged bidirectional.



7 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+SR and define its own advertising. The services can exist in addition to the Terminal I/O service or standalone.

Using the GATT server commands implies the use of the multiplexing protocol.


Application A
BlueMod+SR
Enable Multiplexing Protocol
Disable Terminal I/O Service
Configure Advertising using AT+LEADDATA
Configure Service using AT+LEATTRIB
Enable Advertising

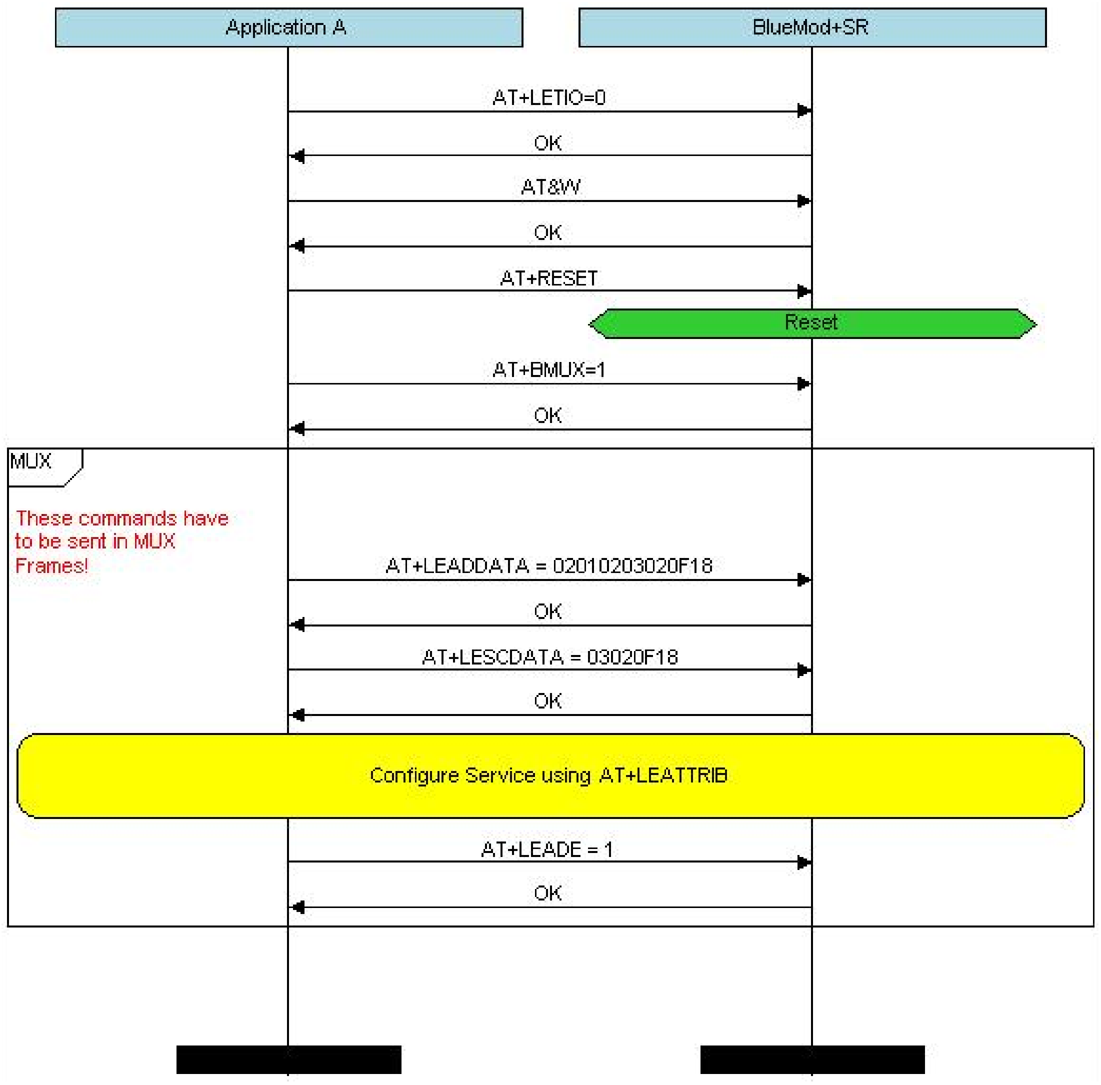
The steps to configure the advertising are optional (green). A GATT service can be used without being stated in the advertising (yellow). If the user wants a customized advertising all steps have to be done.

7.1 Configuring the Advertising

It is not possible to include an additional UUID in the existing Terminal I/O advertising. If the user wants to set the a UUID of a service other than Terminal I/O in the advertising, the Terminal I/O service and advertising has to be disabled by setting AT+LETIO=0. After performing a reset the user can define its own advertising.

The advertising starts, as soon as it is enabled by AT+LEADE=1. All services shall be defined before enabling advertising, otherwise the device sends advertising without providing the appropriate services for a short time.

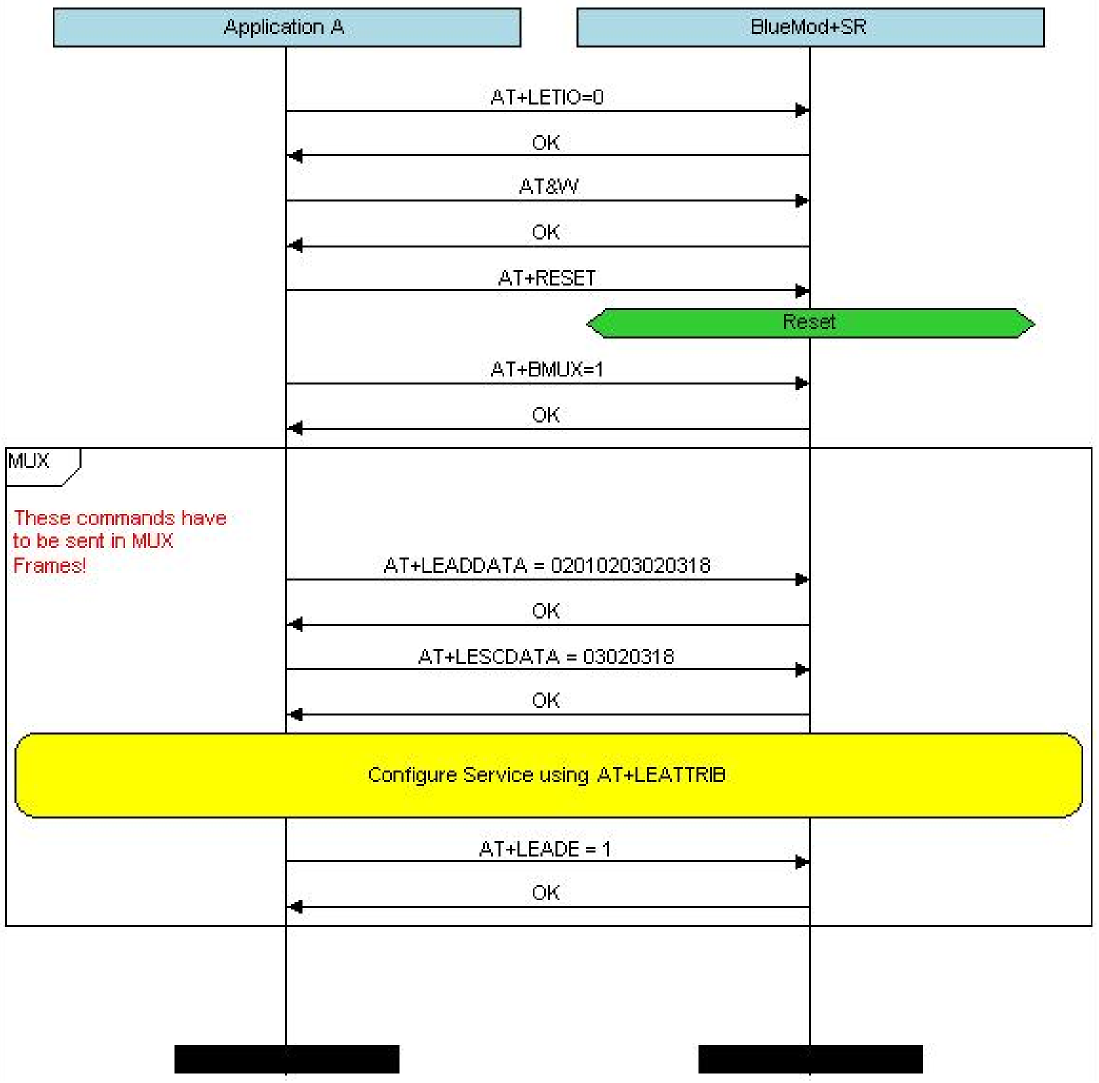
7.1.1 Battery Service


Application A
BlueMod+SR
AT+LETIO=0
OK
AT&W
OK
AT+RESET
Reset
AT+BMUX=1
OK
MUX
These commands have to be sent in MUX Frames!
AT+LEADDATA = 02010203020F18
OK
AT+LESCDATA = 03020F18
OK
Configure Service using AT+LEATTRIB
AT+LEADE = 1
OK

The resulting advertising contains the UUID of Battery Service (180F) and sets the flags to LE General Discoverable mode.

The resulting scan response contains the UUID of Battery Service only.

7.1.2 Proximity Profile


Application A
BlueMod+SR
AT+LETIO=0
OK
AT&W
OK
AT+RESET
Reset
AT+BMUX=1
OK
MUX
These commands have to be sent in MUX Frames!
AT+LEADDATA = 02010203020318
OK
AT+LESCDATA = 03020318
OK
Configure Service using AT+LEATTRIB
AT+LEADE = 1
OK

The resulting advertising contains the UUID of the Link Loss Service (1803) which is mandatory for Proximity profile and sets the Flags to LE General Discoverable mode.

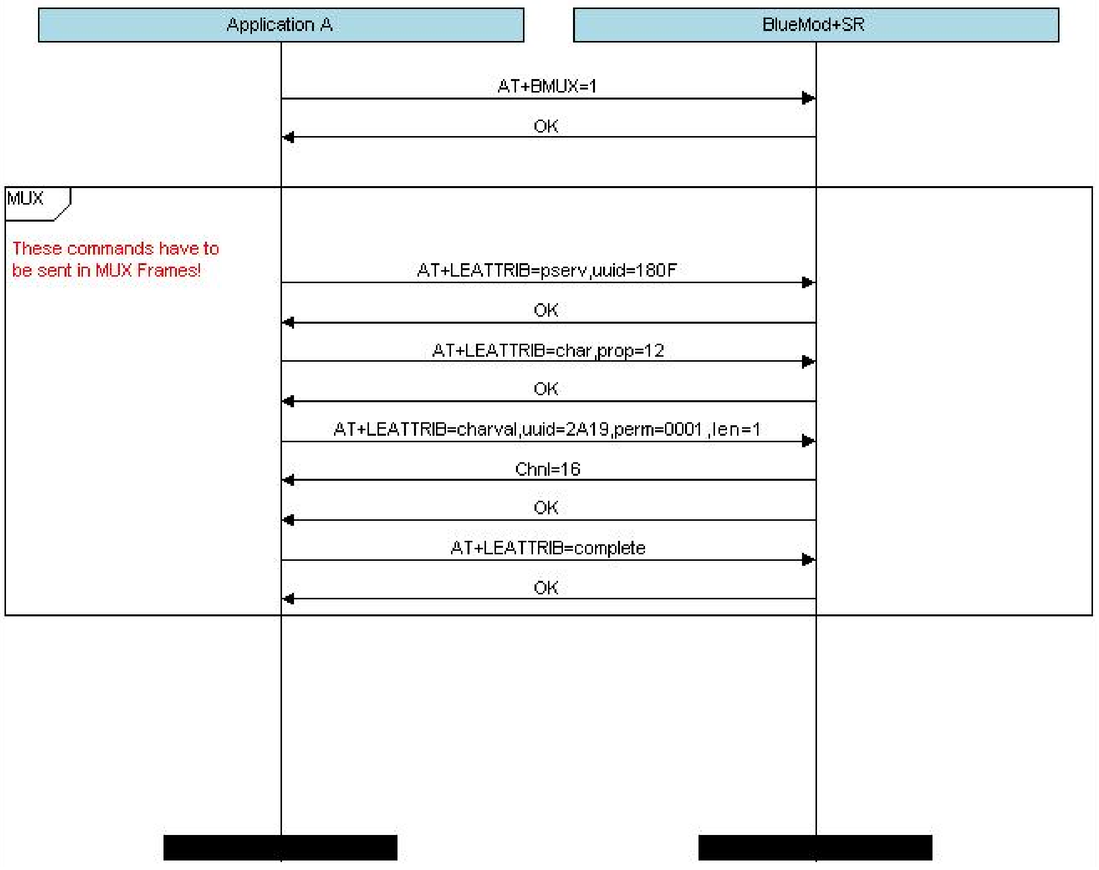
The resulting scan response contains the UUID of Link Loss Service only.

7.2 Configuring Services

The following examples shows how to setup the Battery service and Proximity Profile in BlueMod+SR.

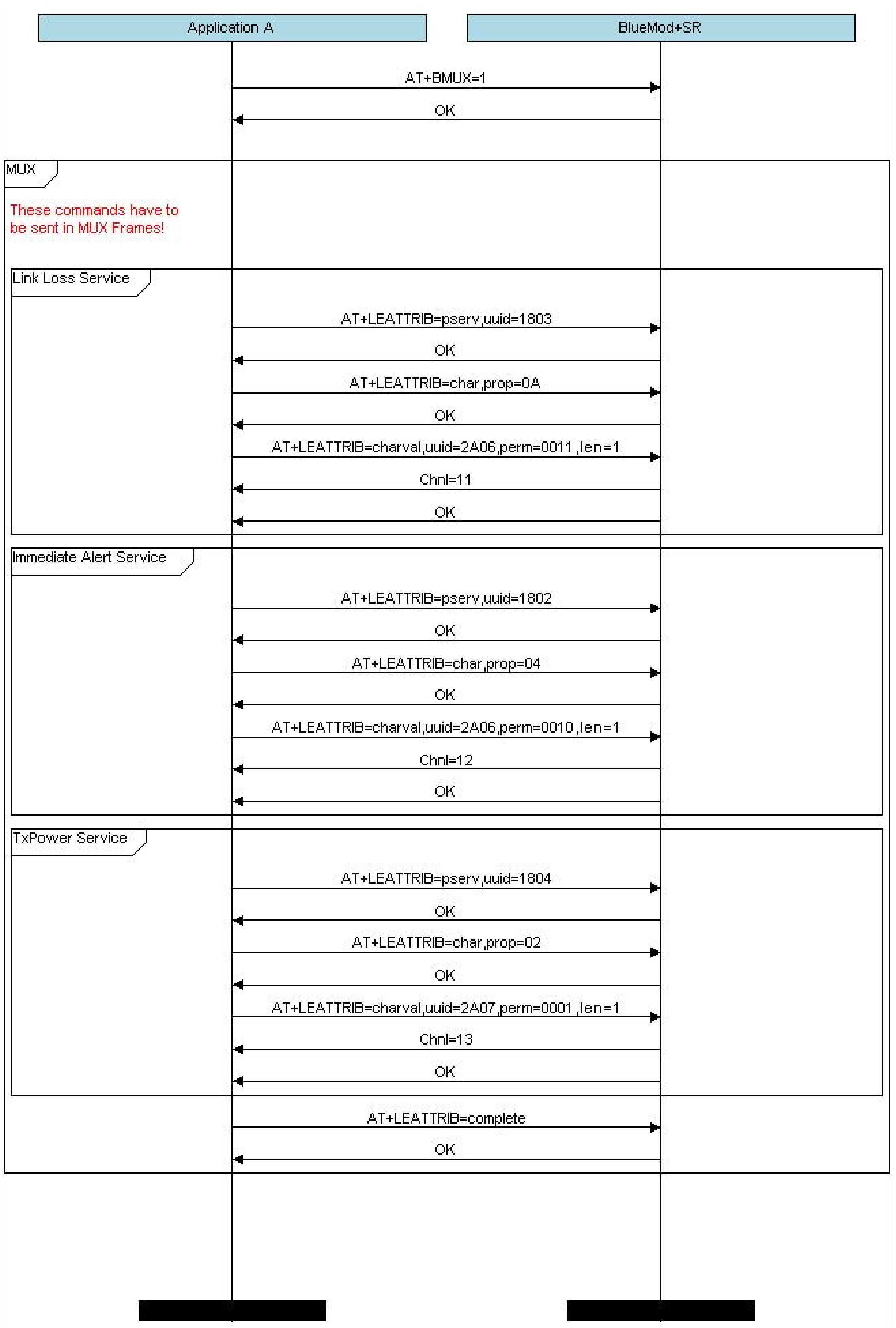
The multiplexing mode has to be activated before the command LEATTRIB can be used. After sending the LEATTRIB=complete command, no other services can be defined.

7.3 Battery Service


Application A
BlueMod+SR
AT+BMUX=1
OK
MUX
These commands have to be sent in MUX Frames!
AT+LEATTRIB=pserv,uuid=180F
OK
AT+LEATTRIB=char,prop=12
OK
AT+LEATTRIB=charval,uuid=2A19,perm=0001 ,len=1
Chnl=16
OK
AT+LEATTRIB=complete
OK

The Battery Service exposes the Battery Level characteristic.

7.4 Proximity profile


Application A
BlueMod+SR
AT+BMUX=1
OK
MUX
These commands have to be sent in MUX Frames!
Link Loss Service
AT+LEATTRIB=pserv,uuid=1803
OK
AT+LEATTRIB=char,prop=0A
OK
AT+LEATTRIB=charval,uuid=2A06,perm=0011 ,len=1
Chnl=11
OK
Immediate Alert Service
AT+LEATTRIB=pserv,uuid=1802
OK
AT+LEATTRIB=char,prop=04
OK
AT+LEATTRIB=charval,uuid=2A06,perm=0010 ,len=1
Chnl=12
OK
TxPower Service
AT+LEATTRIB=pserv,uuid=1804
OK
AT+LEATTRIB=char,prop=02
OK
AT+LEATTRIB=charval,uuid=2A07,perm=0001 ,len=1
Chnl=13
OK
AT+LEATTRIB=complete
OK

The proximity profile consists of the Link Loss service, the Immediate Altert Service and the TXPower Service.

8 NFC Handover

Since firmware version 1.500 the BlueMod+SR is able to communicate to a dynamic NFC tag which can be used for pairing with other NFC enabled devices (handover).

8.1 General Information

The BlueMod+SR supports the following NXP NFC tag types:

- NT3H1101 (1k bytes memory)
- NT3H1201 (2k bytes memory)

This dynamic NFC tag needs to be connected via I²C interface to the BlueMod+SR. For details please refer to the *BlueMod+SR Hardware Reference [5]*.

The NFC tag shares the I²C interface (configured to the fixed I²C address 0xAA) with the optional RSSI value distribution.

8.2 NFC Tag Records

The BlueMod+SR supports static handover for SPP and Terminal I/O connections. The BlueMod+SR uses only the “Simplified Tag Format”. For details please refer to the document *Bluetooth Secure Simple Pairing Using NFC [6]*.

Using static handover with NFC and Bluetooth BR/EDR or Bluetooth LE requires different NDEF records. The BlueMod+SR provides the following NDEF records and writes the selected parameter values to the dynamic NFC tag record field:

- First record: BR/EDR:
  - Mime type, Bluetooth address, EIR data, local name, CoD, 16 bit UUIDs
- Second record: BLE OOB data (included only if AT+LETIO is enabled):
  - Mime type, BLE Bluetooth address, LE role, appearance, local name, 16 bit UUIDs, 128 bit UUIDs

8.3 NFC Mode

AT+NFCMODE activates/de-activates the I²C interface between the BlueMod+SR and the NXP NFC tag.

Value	Description
0	NFC interface off
1	Automatic mode

Setting AT+NFCMODE=1 writes the NDEF records to the NFC tag and enables the NFC interface. As long as enabled it will also write the NDEF records to the NFC tag in the following situation:

- After power on (or software reset)
- Changing the configuration parameter used in one of the dynamic NFC tag record field:
  (AT+BCLASS, AT+BOSRV, AT+BNAME, AT+LEROLE, AT+LETIO)
- Loading the default values or last stored values if these will change one of the parameter used in the dynamic NFC tag record field:
  (AT&F, AT&F1, ATZ)

Note: The maximum count of write cycles for the NFC tag is specified in the manufacturer data sheet.

If there is no NFC activity the BlueMod+SR works as configured with parameters AT+BPSM and AT+BPAIRMODE.

After detecting NFC activity (field detect pin “FD” becomes active) the page scan mode and pairable mode are automatically enabled for 2 minutes (independent of the settings of AT+BPSM and AT+BPAIRMODE).

The BlueMod+SR handles all NFC activity whithout user interaction.

This mode requires a “Just Works” pairing scenario.

After the successful pairing or a timeout occurred the BlueMod+SR returns to the page scan mode and pairable mode as configured with AT+BPSM and AT+BPAIRMODE.

When setting AT+NFCMODE=0 all activity on the NFC interface is ignored. The BlueMod+SR behaves as no NFC tag is connected. A detecting NFC activity (field detect pin “FD” becomes active) is ignored by the BlueMod+SR.

The NFC tag keeps the last updated NFC tag record field configuration even if AT+NFCMODE is set to “0” or is powered off.

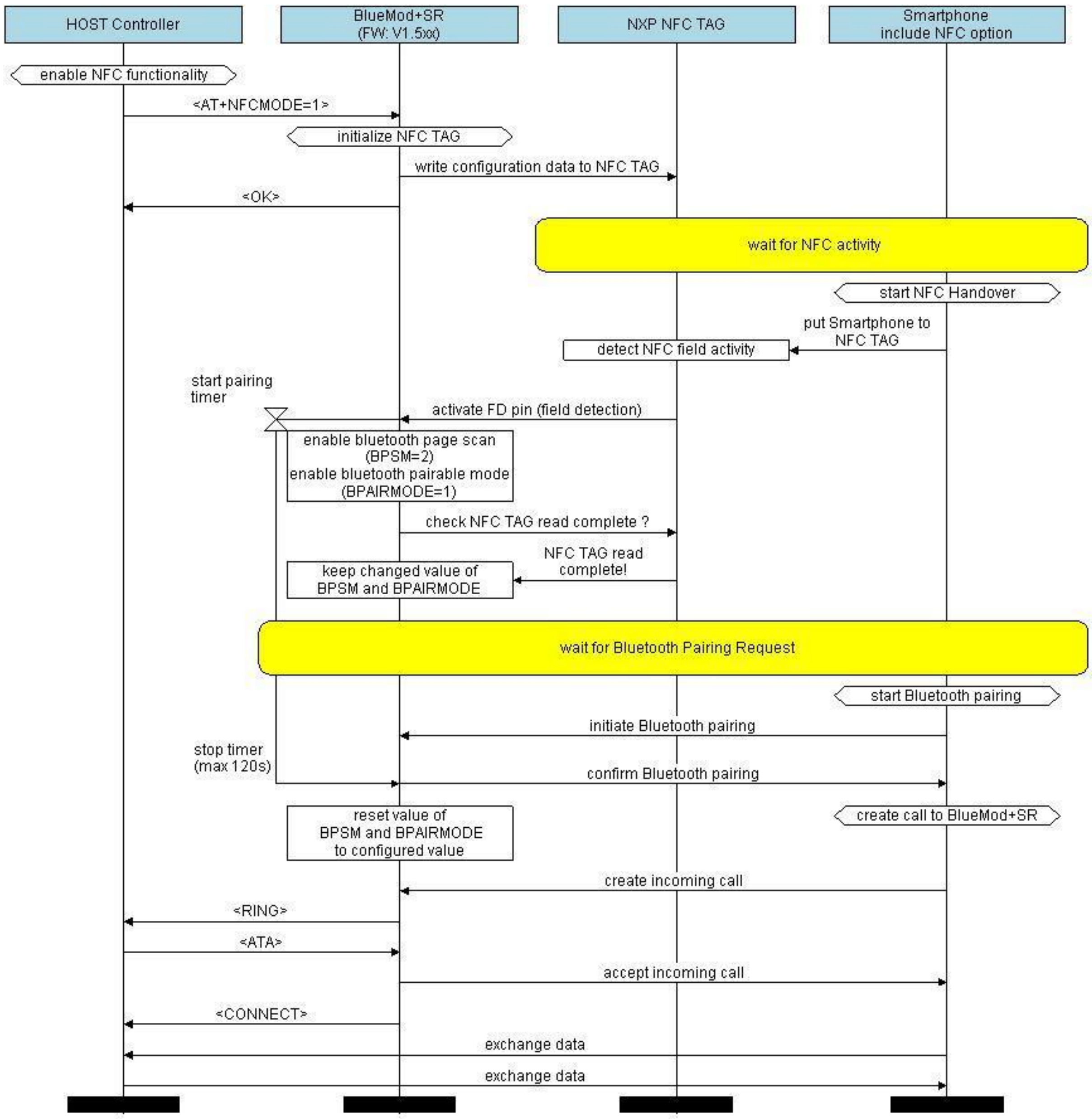
Removing the NFC tag from I²C interface does not erase the written information in the NFC tag record fields.

8.4 NFC Handover Example

The following flow chart will give an example for the NFC Handover between the BlueMod+SR module (include the I²C connected NFC tag) and the NFC supported destination.

On top of the flow chart the different devices are listed like “HOST Controller” who is connected to the UART interface of the BlueMod+SR module. The “NXP NFC TAG” is connected via I²C interface to the BlueMod+SR. The “Smartphone include NFC option” represents the destination.

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


HOST Controller
BlueMod+SR (FW: V1.5xx)
NXP NFC TAG
Smartphone include NFC option
enable NFC functionality
<AT+NFCMODE=1>
initialize NFC TAG
write configuration data to NFC TAG
<OK>
wait for NFC activity
start NFC Handover
put Smartphone to NFC TAG
detect NFC field activity
start pairing timer
activate FD pin (field detection)
enable bluetooth page scan (BPSM=2) enable bluetooth pairable mode (BPAIRMODE=1)
check NFC TAG read complete ?
NFC TAG read complete!
keep changed value of BPSM and BPAIRMODE
wait for Bluetooth Pairing Request
start Bluetooth pairing
initiate Bluetooth pairing
stop timer (max 120s)
confirm Bluetooth pairing
reset value of BPSM and BPAIRMODE to configured value
create call to BlueMod+SR
create incoming call
<RING>
<ATA>
accept incoming call
<CONNECT>
exchange data
exchange data

8.5 “NFC Utility” App for Android

To demonstrate the NFC handover feature Telit provides the “NFC Utility” app for Android devices.

App and demo source code are available on request.

For a detailed description please refer to the document *BlueEva+SR User Guide [7]*.

9 Autodial

Since firmware version 1.531 the BlueMod+SR supports the autodial feature. The autodial feature allows to set up to 3 different autodial targets and can be used for all kind of connections supported by the BlueMod+SR. The autodial feature performs autodialling for all targets with a round robin algorithm. Call timeout, delay timer and connection attempt counter are programmable.

The autodial procedure runs silently in the background. A successful connect is signalled with the CONNECT message. Enable the extended result codes (ATW1) to see the connected Bluetooth address.

The AT+DSET command is used to set or delete a dial string.

Parameter	Description
Number	Number of dial string to be set (1, 2 or 3)
Dial string	Dial string as used for ATD command

The AT+DPARAMS command is used to set the autodial parameters call timeout, delay timer and connection attempt counter.

Parameter	Description
Timeout	0…255 (default **10**): call timeout in seconds
Pause	0…255 (default **20**): delay between connection attempts in seconds
Count	**0** (default): endless retry 1...255: number of connection attempts

The call timer is started with the dial attempt. If the dial attempt didn’t succeed the delay timer is started. If the number of configured connection attempts is reached, the autodial procedure stops.

The AT+D command is used to manually start or stop the autodial procedure.

Value	Description
Start	Start an autodial procedure with configured parameters
Stop	Stop an active autodial procedure

The AT+D=? command is used to report the status of the autodial procedure.

Report type	Description
Status	Off: Autodial procedure not running CALL: During call timeout (defined in AT+DPARAMS) DELAY: During pause (defined in AT+DPARAMS) between connection attempts
Number	Number of active dial string
Count	Number of active dial sequence (0 if endless autodial is configured in AT+DPARAMS)

The AT+DMODE command is used to enable/disable the automatic autodial procedure. When set to 255 the autodial procedure starts automatically after power-on or reset.

Value	Description
0	Autodial off
255	Dial all valid entries round robin

9.1 Implications on Normal Operation

When the autodial procedure is started incoming calls are forbidden. Therefore the user has to assure that incoming calls are ignored (setting ATS0=0) and the device shall be non-connectable (setting AT+BPSM=0 and AT+LEROLE=1).

The host shall not issue dial, device and service discovery commands (ATD, AT+BINQ, AT+BINQSERV) while the autodial procedure is started.

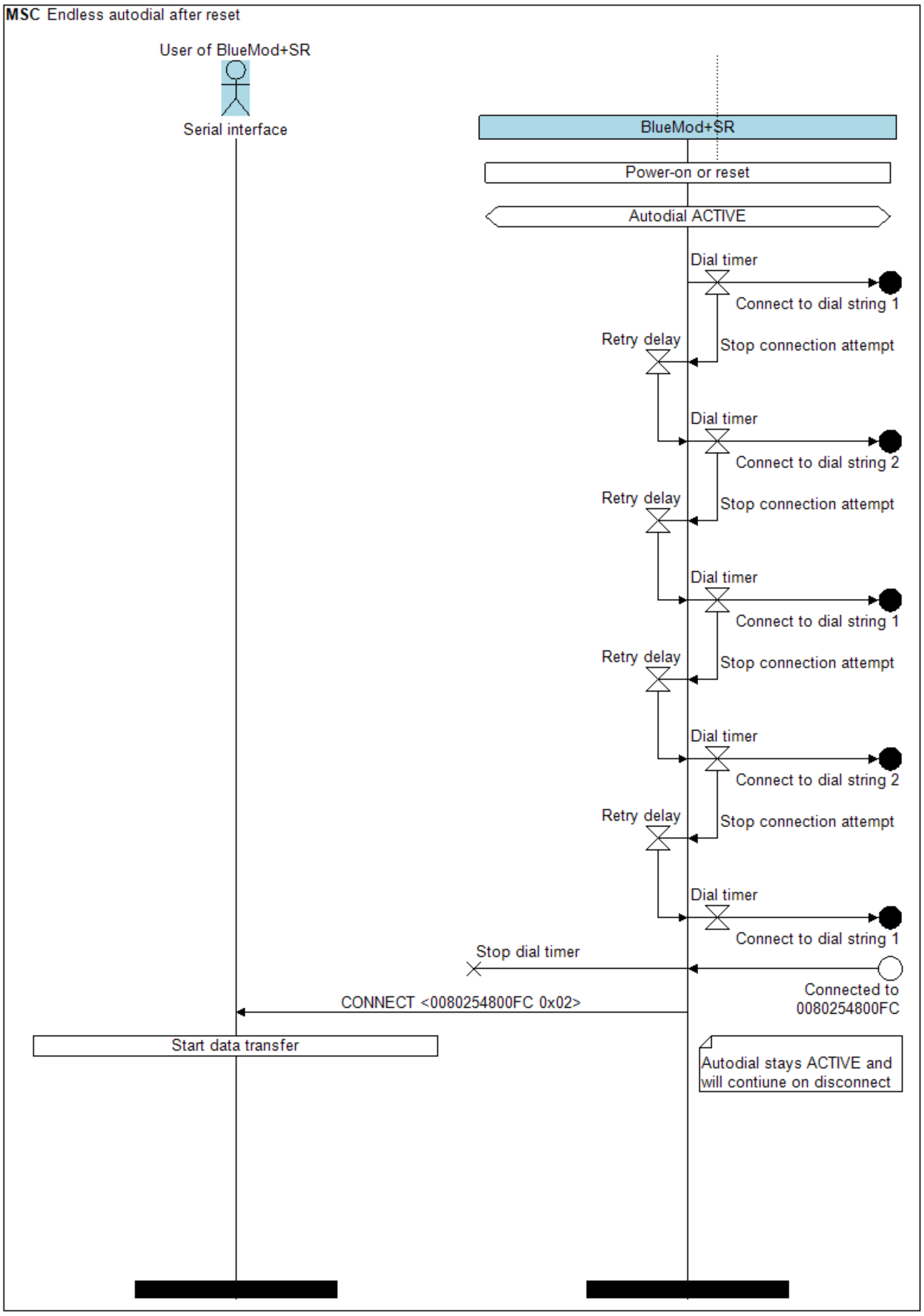
9.2 Connection Examples

9.2.1 Automatic Endless Autodial

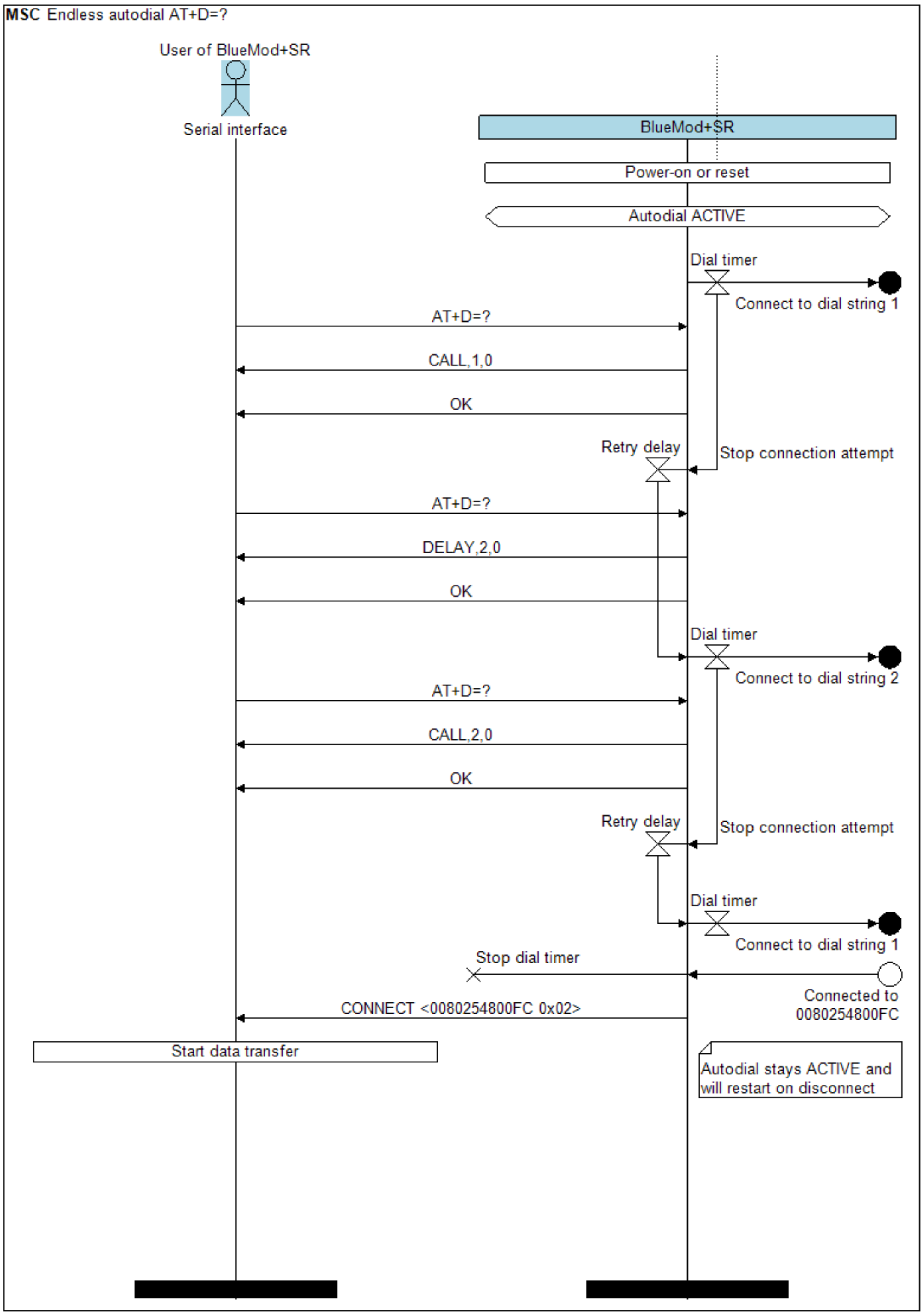
To setup an automatic and endless autodial procedure with 2 TIO peripherals the following settings are required:

Command	Description
AT+DSET=1,0080254800FC,TIO	Set dial string 1: TIO connection to device with Bluetooth address 0080254800FC
AT+DSET=2,008025540202,TIO	Set dial string 2: TIO connection to device with Bluetooth address 008025540202
AT+LEROLE=1	Set BlueMod+SR to central role to support outgoing TIO connections
AT+DPARAMS=10,20,0	Set call timeout to 10 seconds, call delay to 20 seconds and endless retry count
AT+DMODE=255	Activate automatic autodial after power-on/reset
ATW1	Enable extended result codes
AT&W	Store the settings non-volatile

The example is shown in the flow chart below.


MSC Endless autodial after reset
User of BlueMod+SR
Serial interface
BlueMod+SR
Power-on or reset
Autodial ACTIVE
Dial timer
Connect to dial string 1
Retry delay
Stop connection attempt
Dial timer
Connect to dial string 2
Retry delay
Stop connection attempt
Dial timer
Connect to dial string 1
Retry delay
Stop connection attempt
Dial timer
Connect to dial string 2
Retry delay
Stop connection attempt
Dial timer
Connect to dial string 1
Stop dial timer
Connected to 0080254800FC
CONNECT <0080254800FC 0x02>
Start data transfer
Autodial stays ACTIVE and will contiune on disconnect

The flow chart below shows the different states of the autodial procedure.

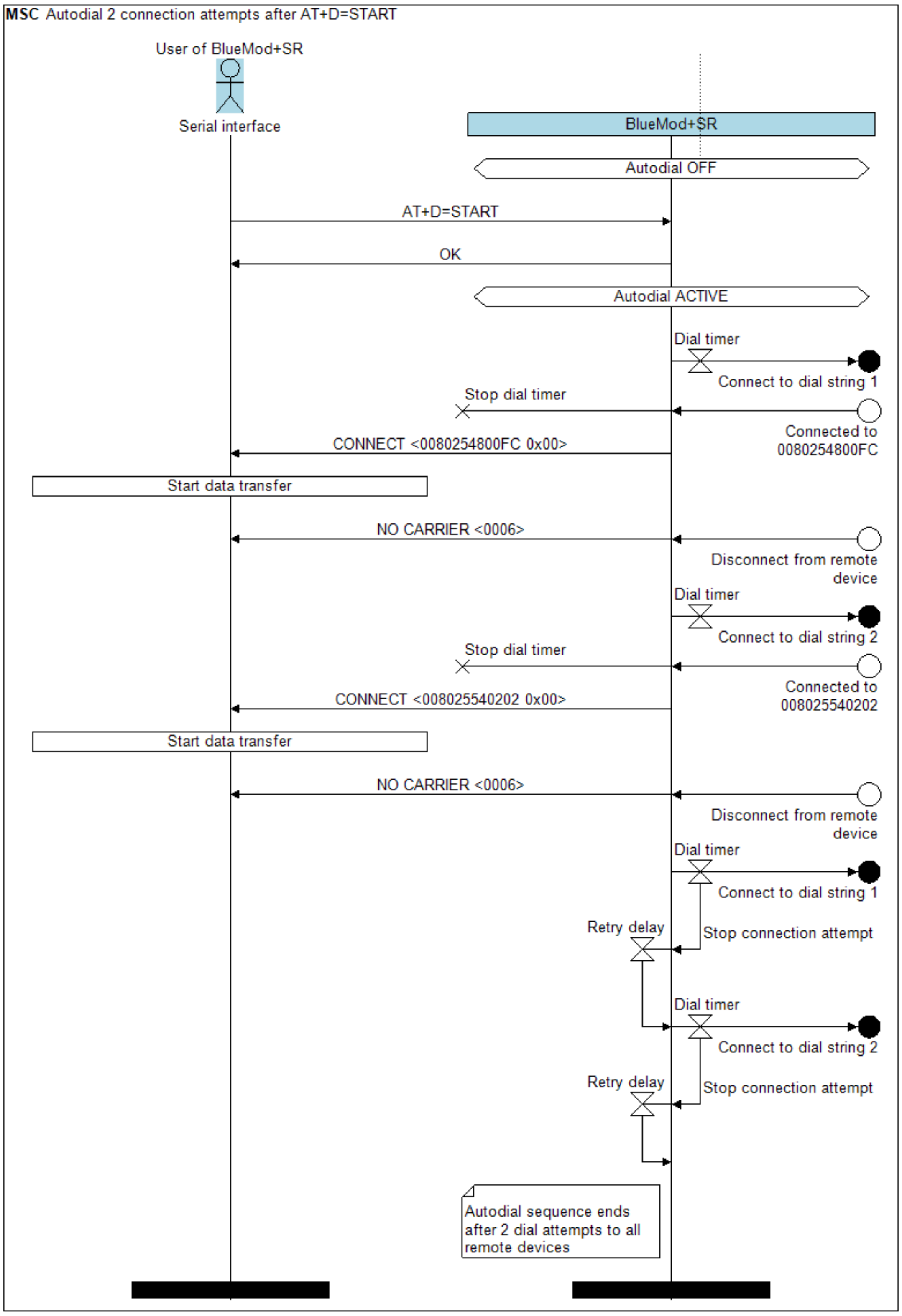


9.2.2 Manually Autodial with 2 Connection Atttempts

To manually start an autodial procedure with 2 connection attempts with 2 SPP targets the following settings are required:

Command	Description
AT+DSET=1,0080254800FC,SPP	Set dial string 1: SPP connection to device with Bluetooth address 0080254800FC
AT+DSET=2,008025540202,SPP	Set dial string 2: SPP connection to device with Bluetooth address 008025540202
AT+DPARAMS=15,10,2	Set call timeout to 15 seconds, call delay to 10 seconds and retry count to 2
AT+DMODE=0	Deactivate automatic autodial after power-on/reset
ATW1	Enable extended result codes
AT&W (optional)	Store the settings non-volatile
AT+D=Start	Start autodial procedure

The example is shown in the flow chart below.


MSC Autodial 2 connection attempts after AT+D=START
User of BlueMod+SR
Serial interface
BlueMod+SR
Autodial OFF
AT+D=START
OK
Autodial ACTIVE
Dial timer
Connect to dial string 1
Stop dial timer
Connected to 0080254800FC
CONNECT <0080254800FC 0x00>
Start data transfer
NO CARRIER <0006>
Disconnect from remote device
Dial timer
Connect to dial string 2
Stop dial timer
Connected to 008025540202
CONNECT <008025540202 0x00>
Start data transfer
NO CARRIER <0006>
Disconnect from remote device
Dial timer
Connect to dial string 1
Retry delay
Stop connection attempt
Dial timer
Connect to dial string 2
Retry delay
Stop connection attempt
Autodial sequence ends after 2 dial attempts to all remote devices

10 Firmware Upgrade

The firmware upgrade will be done 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 system memory boot mode protocol on the host system.

10.1 Firmware Upgrade via Serial Interface

This chapter describes the firmware upgrade procedure for a BlueMod+SR via UART using a Windows PC.

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 software package provides a tool for uploading firmware into a BlueMod+SR via the serial interface.

For example a firmware version 1.103 will result in the executable file “SR_1_103_FWupdate.exe”.

10.1.1 Prerequisites for Serial Firmware Upgrade

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

Serial firmware update requires at least a 3-wire serial connection (UART-RXD, UART-TXD, GND) to the PC without flow control.

Pin BOOT0 (E-1) shall be pulled high to access the STM32 Bootloader.

You need to have a correct serial update file for your BlueMod+SR.

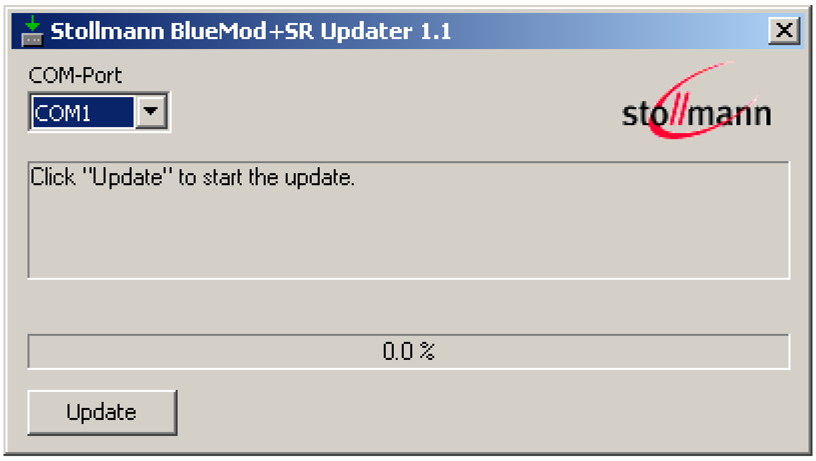
10.1.2 Stollmann BlueMod+SR Updater

Stollmann BlueMod+SR Updater serves as a tool for uploading a firmware file into a BlueMod+SR.

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

Several instances of Stollmann BlueMod+SR Updater may be started concurrently on one PC in order to update several BlueMod+SR in parallel.

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

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+SR please contact the support.

10.2 Firmware Upgrade using Host Platform in Embedded Environment

This chapter describes the steps to implement a firmware update using a host platform in an embedded environment.

10.2.1 Hardware Requirements

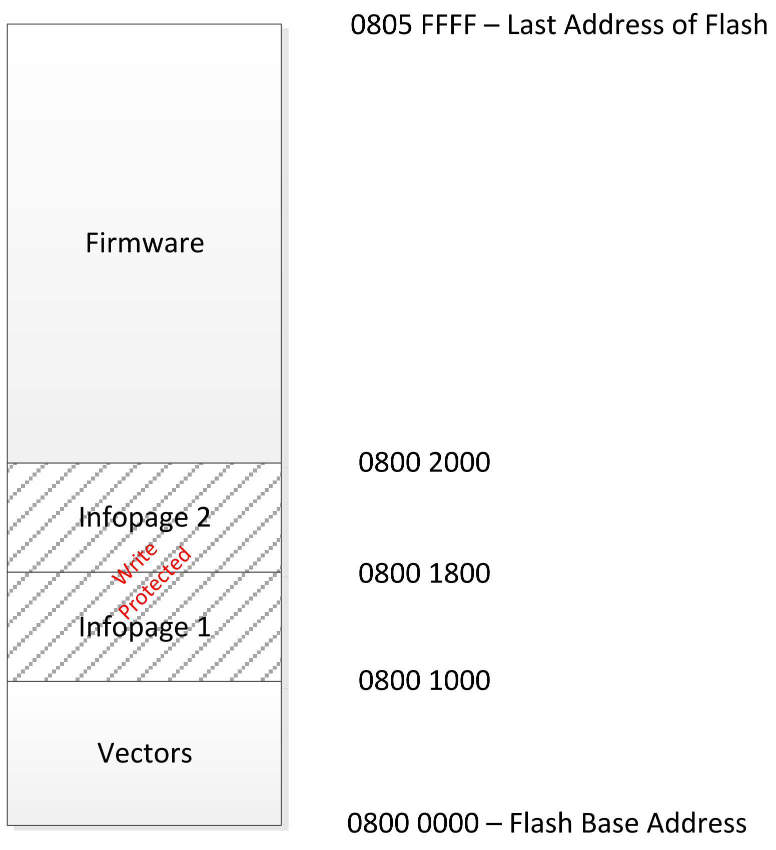
The host platform needs a UART connection to the BlueMod+SR (at least a 3-wire interface). Furthermore the pin BOOT0 (E1) needs to be pulled high to access the STM32 bootloader and the module shall be reset using the pin EXT-RES# (B1).

10.2.2 STM32 Bootloader

The STM32 bootloader can read and write the flash of the BlueMod+SR. For the full documentation of the STM32 bootloader we refer to the document [4] of STMicro.

10.2.3 Flash Memory Mapping

The firmware consists of two parts, vectors and firmware. Both parts shall be flashed to the right offset in flash.



The flash is partitioned in flash pages of 2k size.

The vectors.bin file shall be flashed to the base address 0800 0000.

Flash pages two and three (0800 1000 – 0800 1FFF) are write protected and shall not be deleted. This area contains the product specific information.

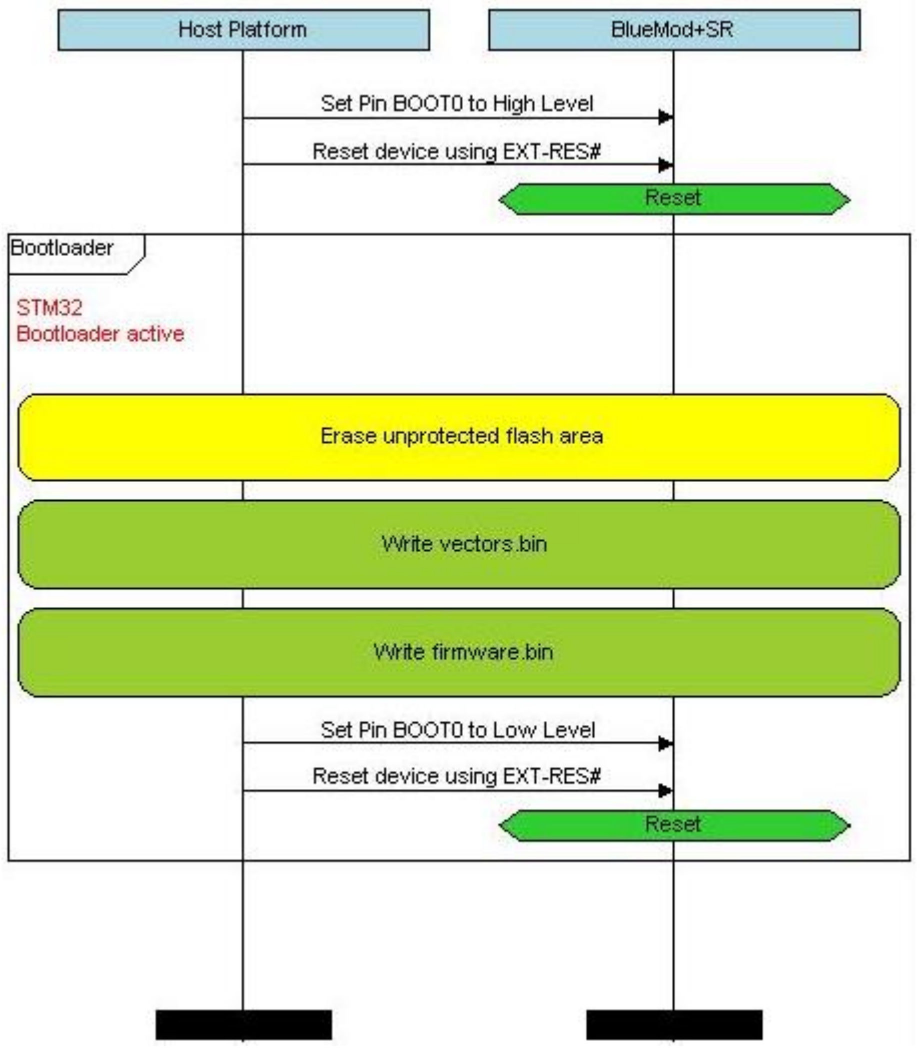
The firmware.bin file shall be flashed to the address 0800 2000.

Vectors and firmware area shall not be write protected after flashing.

The provided bin files are multiples of four bytes in size, as the STM32 bootloader demands.

10.2.4 Upgrade Process

The following steps are recommended to do a firmware update.


Host Platform
BlueMod+SR
Set Pin BOOT0 to High Level
Reset device using EXT-RES#
Reset
Bootloader
STM32
Bootloader active
Erase unprotected flash area
Write vectors.bin
Write firmware.bin
Set Pin BOOT0 to Low Level
Reset device using EXT-RES#
Reset

- BOOT0 Pin (E1) shall be pulled to high level to access the STM32 bootloader
- Reset shall be done using EXT-RES# (B1)
- After the reset the STM32 bootloader is active
- The unprotected area of the flash shall be erased using the erase command
- Vectors.bin shall be written to the flash
- Firmware.bin shall be written to the flash
- BOOT0 Pin (E1) shall be pulled to low level to leave the STM32 bootloader
- Reset shall be done using EXT-RES# (B1)
- The firmware is accessible in AT mode again, query the version number using ATI99 command

11 Related Documents

[1] BlueMod+SR AT Command Reference

[2] BlueMod+SR Startup Timing

[3] UICP+ UART Interface Control Protocol

[4] AN3155_USART_protocol_used_in_STM32_bootloader

[5] BlueMod+SR Hardware Reference

[6] Bluetooth Secure Simple Pairing Using NFC

[7] BlueEva+SR User Guide

12 History

Version	Release Date	By	Change description
r01d01	08.03.2013	hb	Initial version
r01d02	21.05.2013	hb	Added chapter 2 Startup Timing Addec chapter 5 Terminal I/O over Bluetooth Low Energy
r01	23.05.2013	hb	Added link to documentation in chapter 1 Removed description of AT Commands from chapter 3 Added BLE description instead Added description and link of Terminal IO Utility in chapter 3 Revised OS information for firmware upgrade in chapter 8 Revised formatting and typos
r02	05.11.2013	hb	Added chapter 7 GATT Configuration Revised chapter 6 UICP
r03	26.09.2014	ta	Added chapter 4 Security Added description about special behavior of BNAME with Bluetooth Low Energy
r04	29.09.2014	ta	Added chapter 10.2 Firmware Upgrade using Host Platform in Embedded Environment Added list of Related Documents
r05	21.10.2014	ta	Revised description of 6 UART Interface Control Protocol (UICP)
r06	16.01.2015	ta	Added chapter 3 Pairable and Bondable Mode Added chapter 8 NFC Handover
r07	10.09.2015	ta	Added startup timing in the document Improved drawings of state changes in UICP chapter Added note in UICP chapter that all data received before interface up state has been achieved shall be discarded
r08	27.11.2015	ta	Added startup timing of firmware version 1.531 Added new feature 9 Autodial
r09	26.05.2016	bg	Telit cover page added.

Telit Wireless Solutions GmbH	Phone: +49 (0)40 890 88-0
Mendelssohnstraße 15 D	Fax: +49 (0)40 890 88-444
22761 Hamburg	E-mail: ts-srd@telit.com
Germany	www.telit.com

SUPPORT INQUIRIES

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

www.telit.com

Telit

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

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

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

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

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

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

[04.2016]