Version	Date	Name	Description of Change
0.0.12	22.11.2020	Lippach	Derived from Protocol description Holex
0.0.13	24.11.2010	Lippach	First revision after internal review
0.0.14	06.12.2020	Lippach	Second revision after internal review
0.0.15	16.12.2020	Lippach	Extensions regarding Service UUID’s
0.0.16	12.01.2021	Lippach	Definition of Product/Tool ID
0.0.17	17.02.2021	Lippach	Extension regarding SGTIN
0.0.18	19.03.2021	Lippach	Extension for SCAN-Response
0.0.19	14.04.2021	Lippach	Serial Number in Local Name
0.0.20	08.06.2021	Lippach	Extension Identification flag
0.0.21	11.06.2021	Lippach	Identification flags bitwise
1.0.0	08.08.2021	Lippach	Final revision after release of FW Version 1.0.0
1.0.1	27.04.2022	Lippach	Company Prefix in Advertisement corrected
1.0.2	27.04.2022	Lippach	Proposal for ProtocolType in Advertisement

HCT Protocol Description

V1.0.2

Inhalt

1 Glossary

Server	The server is the one who holds data and can offer parts of its data. In the context of BLE the Server will be the unit which represents the slave role. The Server will send the data if the client asks for or send data notifications if internal data have been changed and client is interested in.
Client BLE_Client	The client can send data to the server and asks for data in order to remain updated. The client will usually also receive data as notifications, if data have been changed.
BLE-Proxy-Library	Collection of C-based Header and source code files which help to serialize and deserialize the serial protocol to the BLE-Controller and the data flow from and to the BLE-clients.
Master	= BLE – Client / Central Device
Slave	= BLE – Server / Peripheral Device

2 Executive Summary

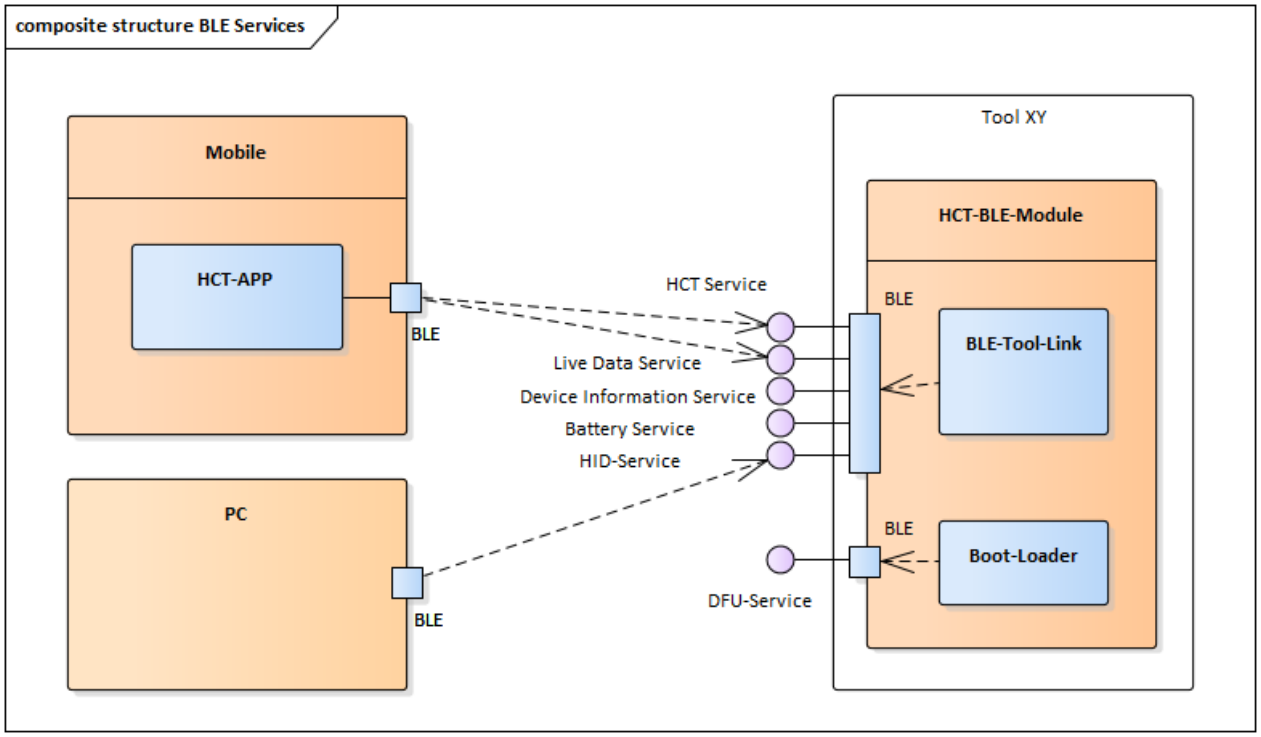
The current document describes the usage of the HCT-BLE-Module in order to be used in a tool for embedding it into the Hoffman-Connect-Tools infrastructure. It gives an overview about the BLE-Interfaces provided by the module, especially the HCT-Service-Protocol, as well as the interface between the Tool Application Module and the HCT-BLE-Module.
Detailed information about the API will be provided by another document.

3 Requirements

The following functional and non-functional requirements are the basis for the design of SW- and interface architecture.

- Functional Requirements
  - Writing/Reading and publishing of data
  - Creation, Reading, Writing and Deleting of list entries
  - Reading (and Writing) of entire lists
  - Sending live data asynchronously
  - Sending HID-Data asynchronously
- Non-Functional requirements
  - Encapsulation of BLE-Complexity towards the supplier of a tool
  - Simple interface to Supplier Controller
  - Notification when data has been changed
  - Multi-Client-Capability
  - Dynamic Advertisement
  - Dynamic Configuration of Services (e.g. HID)
  - Performance
  - Energy saving by optimized power consumption
  - Security → Dynamic Configuration of Security-Levels
  - Reducing of data transmission load to a minimum for energy saving

4 BLE-Interfaces of the HCT-BLE-MODULE



Currently there are five services: the HCT-Service, the Live Data Service and the HID-Service. It is planned to use the SIG adopted Battery- and Device Information Service and in future develop generic services that fit to the interoperability with other devices.
For Firmware Update the DFU-Service is used.

4.1 Advertising Structure of the HCT-BLE-Module

The advertising data (28 Byte) compiled by the HCT-BLE-Module has the following structure, explained by the example:

02 01 06 03 03 12 18 0A 08 48 43 54 2D 54 57 30 31 32 08 FF A3 08 02 01 02 00 02 FE FE

# of bytes	Explanation	Value (Hex)	Content
2	Length and type	03 19	3 Byte Type: Appearance
2	Appearance	C1 03	0x03C1 → Keyboard
2	Length and type	02 01	2 Byte, Type: Flags
1	Flags (Capabilities, bit coded)	06	LE Limited Discoverable Mode →No LE General Discoverable Mode →Yes BR/EDR Not Supported →Yes Simultaneous LE and BR/EDR (Controller) → No Simultaneous LE and BR/EDR (Host) → No
2	Length and type	0A 08	10 Byte, Type: Shortened Local Name
9	Local Name	48 43 54 2D 54 57 30 31 32	ASCII Values for "HCT-<TT><VVV>" TT = Tool Type; VVV = Variant or Version "HCT-TW012", see [**Fehler! Verweisquelle konnte nicht gefunden werden.**]
2	Length and type	08 FF	8 Byte, Type: Manufacturer Specific Data
2	Company ID	A3 08	0x08A3 → (Hoffmann SE Manufacturer ID)
1	Brand Indication	00 .. FF	00: Unknown 01: Garant 02: Holex 03: tbd
1	Identification Flags	00 .. 03	Bit 0 = 0: APP-Connection is deactivated Bit 0 = 1: APP-Connection activated, Bit 1 = 1: Tool is in usage (e.g. when button is pushed)
1	ProtocolType	02	0x0 = unknown or not implemented 0x1 = HCT 1 (old DTW) 0x2 = HCT 2 (current Generation) 0x10 = Sylvac-Protocol 0x11 = Mahr-Protocol (0xFE = HCT 2 (current Generation Holex))
2	Reserve	FE FE	
2	Length and Type	02 0A	2 Byte, Type: TX-Power Level
1	TX-Power	00	0 dBm
31	Gesamt		

For the tool identification the scan response data must be used. This data can be acquired by a scan request from the central device, if the peripheral indicates that more advertising data are available.

Scan response data example

# of bytes	Explanation	Value (Hex)	Content
2	Length and type	03 03	3 Byte, Type: Complete List of 16-bit Service Class UUIDs (here only one will be listed)
2	UUID	12 18	0x1812 → HID Service UUID
2	Length and type	12 FF	18 Byte, Type: Manufacturer Specific Data
2	Company ID	A3 08	0x08A3 → (Hoffmann SE Manufacturer ID)
4	GTIN (EAN) GS1 Company Prefix	C6 FC 3D 00	0x3D FCC6 → “4062406” 7 decimal places
4	GTIN (EAN) Item Reference	ED 7E 01 00	0x00017EED →”98029” 5 decimal places
6	Serial Number	81 F9 52 BA 00 00	0x00BA52F981→“ 003126000001” 12 decimal places
9	Reserve	00	
31 Bytes			

4.1.1 GTIN (EAN)

The GTIN is an individual, purely numeric article number from the GS1 numbering system. It identifies each product uniquely worldwide.
It contains the company prefix which identifies the producer and the item reference which identifies the product type and its sub-types uniquely.
An additional serial number is added in a way, that it compiles a SGTIN, that is used by the HCT-App in order to distinguish between advertising devices of the same type by reading a DMC-Code which is lasered on a metal part of the tool, printed on a sticker or shown on the display of the tool.

4.1.2 DMC-Definition

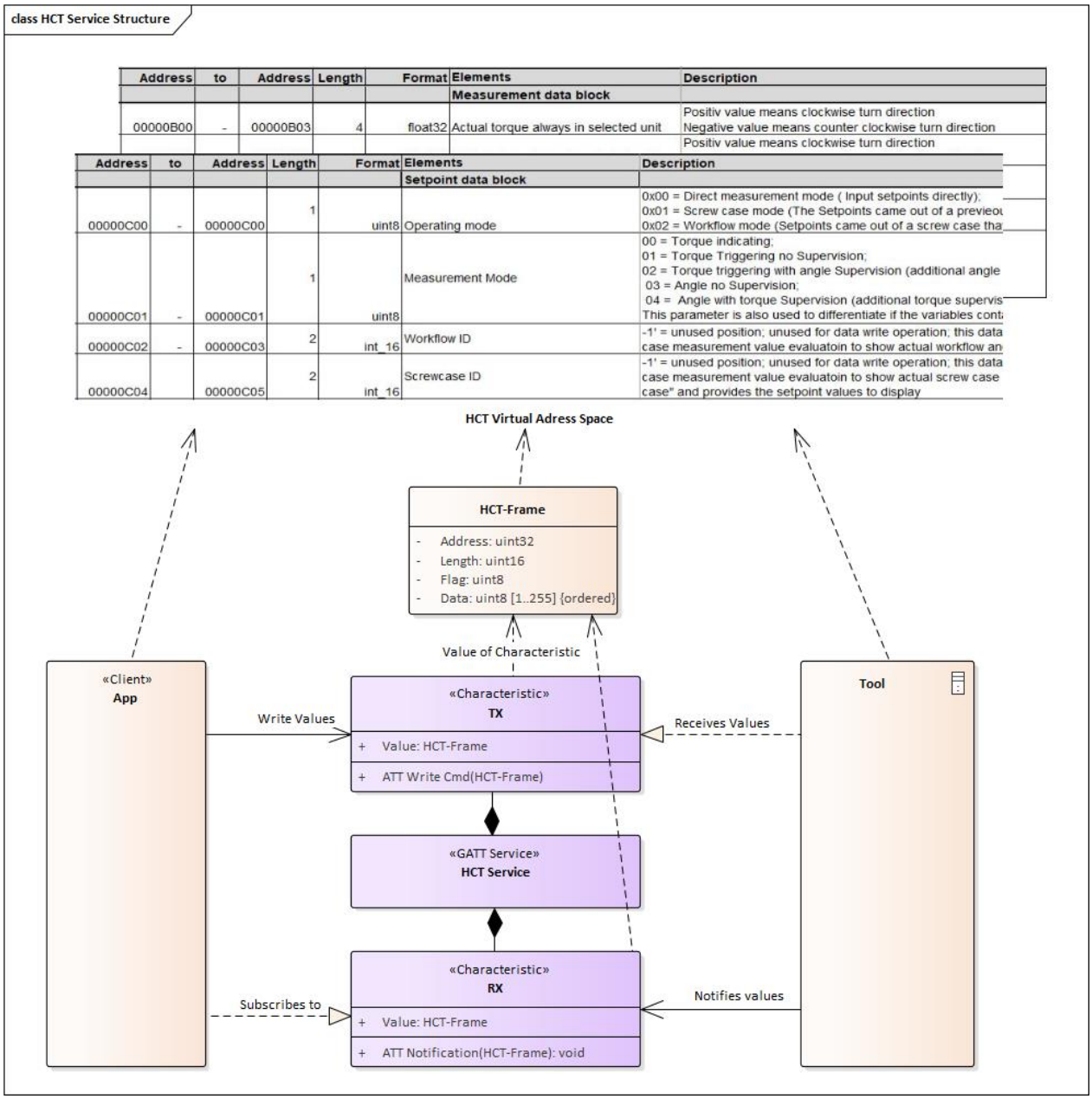
The Data Matric Code (DMC) shall be used for identifying the respective tool by the camera of a mobile device in order to establish the right connection.
The DMC-Code complies to the VDMA 34193.
Template and Example of a character string with leading FNC1 character and serialized identification number (SGTIN) for encoding in the GS1 DataMatrix
Template:
<FNC1>01 0**CCCCCCC RRRRR**2 21 **SSSSSSSSSSS**
C = Company Prefix with 7 numerical characters
R = Item Reference with 5 numerical characters
S = Serial Number with 12 numerical characters
Example with reference to the GTIN and Serial Number in Scan response data
<FNC1>01 0**4062406 98029**2 21 **003126000001**

4.1.3 Local Name – Identification for HID on Windows.

The “Local Name” in the HCT advertisement is the so called ‘shortened local name’.
It only gives an information about the tool-type or tool- characterization, not about its unique identification. It must comply to the HCT advertisement naming convention, defined in [**Fehler! Verweisquelle konnte nicht gefunden werden.**].
The full local name consists of the shortened local name and can have up to 30 characters, e.g. “HCT-<TT><VVV>-<Serial Number>”
whereby the possible name extension up to 20 characters shall contain a unique information that makes it possible to distinguish between two or more tools of the same type, especially when connected as a HID-device to Windows. It shall contain the serial number of the tool which is also part of the SGTIN of the DMC.
The complete name can be acquired by reading the device name characteristic after the connection has been established using GATT.

4.2 General description of the HCT-Service-Protocol

The protocol itself in fact describes the access to a common process data interface. This process data interface is represented by a virtual memory. The virtual memory is divided into different blocks which contain data to a specific meaning or function, like structures or classes, for example measurement data or setpoint data. The meaning and structure of the data at the specific addresses in the data blocks is currently described in a generic device- xml data and device specific data -xml file which extends the generic data. This data is individual for each device and depends on the device capabilities. The start addresses of the respective blocks, also called base addresses or base identifier, will be used as a reference into the internal memory structure, whereby, depending on the development environment the respective block can be implemented as class or structure and assigned to its base address.
XML or equivalent excel-file can give a description what kind of data is available from the device.



The figure above shall illustrate the basic structure and philosophy of the HCT Service that consists of two proprietary GATT-Characteristics, the HCT-TX and HCT-RX characteristics. Both contain a value of the type *HCT-Frame* which is in fact a structure, that points to a certain element and number of subsequent elements of the virtual address space or parts of it and indicates whether these data needs to be transferred from the client to the server or vice versa. This frame shall be transferred, either by a Write Command of the TX-Characteristic from the Client (App) to the Server (Tool) or by a Notification of the value (HCT-Frame) of the RX-Characteristic from the Client to the Server.
In order to access this virtual memory and its blocks or classes and manipulate them, some generic methods are available which are described in the next chapter.

HCT_PROTOCOL_SERVICE_UUID	1015**0100**-34bb-41c7-bc33-78855b62d28e	Description and Access Method
HCT_PROTOCOL_TX_UUID	1015**0101**-34bb-41c7-bc33-78855b62d28e	Data transfer from Client (central device) to the Server (peripheral device) **ATT Cmd Write** (no Request)
HCT_PROTOCOL_RX_UUID	1015**0102**-34bb-41c7-bc33-78855b62d28e	Data transfer from Server (peripheral device) to the client (central device) **ATT Notification**

4.3 The HCT BLE Protocol-Frame

The protocol frame which also describes the structure of the HCT BLE TX and RX-characteristics consists of:

HEADER			**DATA**
ADDRESS	**LENGTH**	**FLAG**	**DATA**
4 Byte	2 Byte	1 Byte	MAX:256 Byte

ADDRESS:
The address is in fact a virtual address into the process data space. The Address space is divided into 256 Byte pages. Only one page can be written at once in a standard transmission procedure. The address map has some generic blocks respectively classes but can be extended by tool specific information. The available data content is in the device specific documentation.

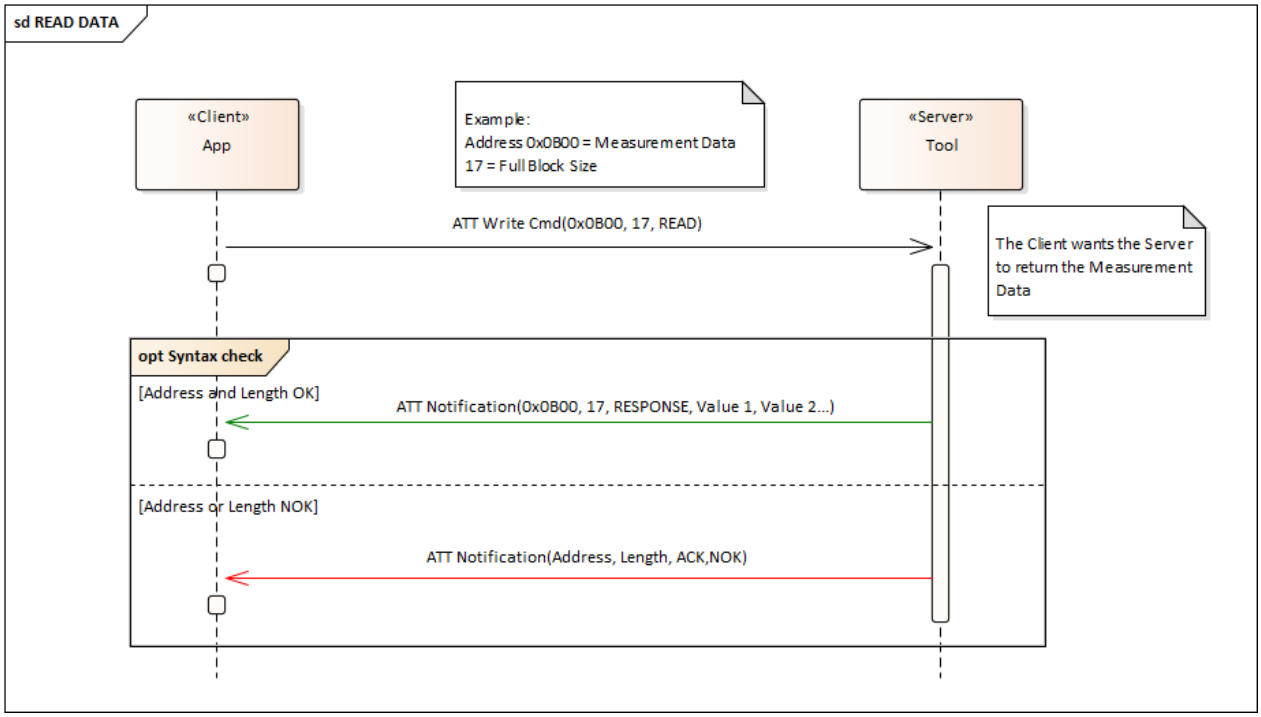
LENGTH:
The Length gives the amount of DATA in byte that should/could be transferred, without header.

FLAG:
The Flag indicates the type of transmission.

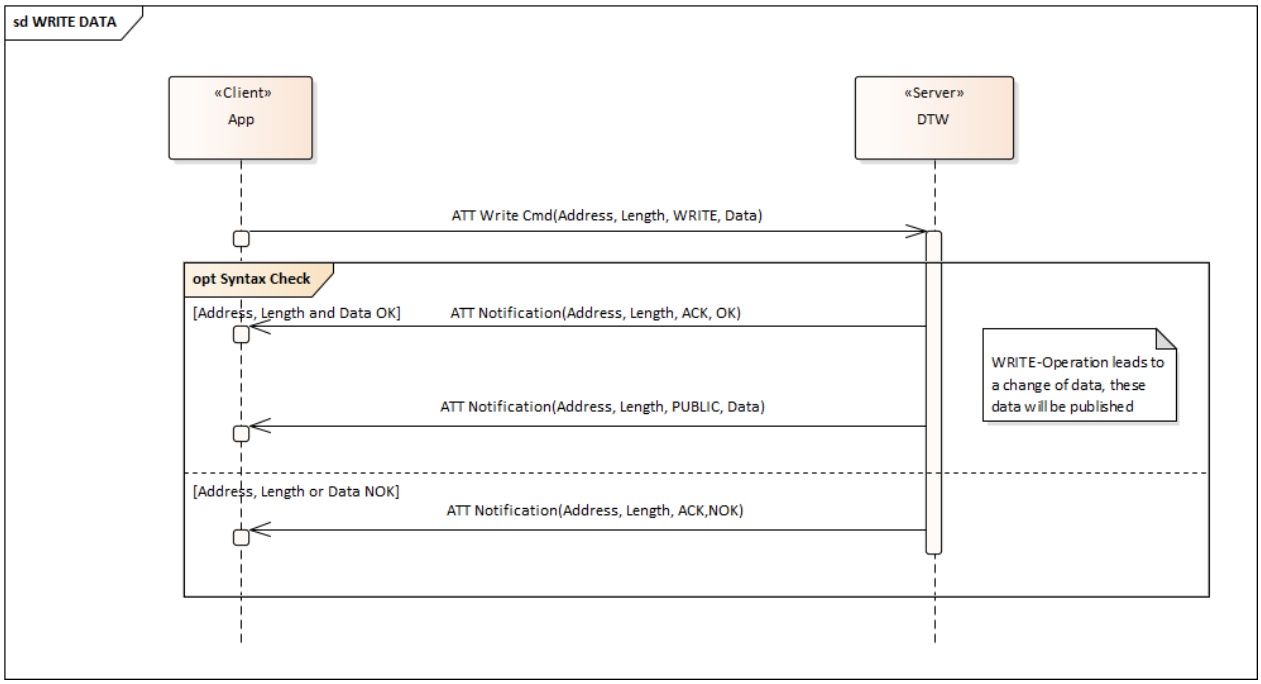
FLAG - Byte			
Value	Short name	Explanation	Reaction of Receiver
0x00	READ	This is a query for data from the client to be returned, from the server	RESPONSE or ACKNOWLEDGE with NOK or other Error code

0x01	WRITE	Write/Take over Data: The server takes over the data sent by the client.	ACKNOWLEDGE
0x02	ACKNOWLEDGE	Acknowledge message, message contains ACK/NACK or other Error Codes	NO
0x03	RESPONSE	Answer to a READ request	NO
0x04	PUBLISH	Spontaneous sent after a data change	NO
0x05…0xFF	RESERVE	Reserved	

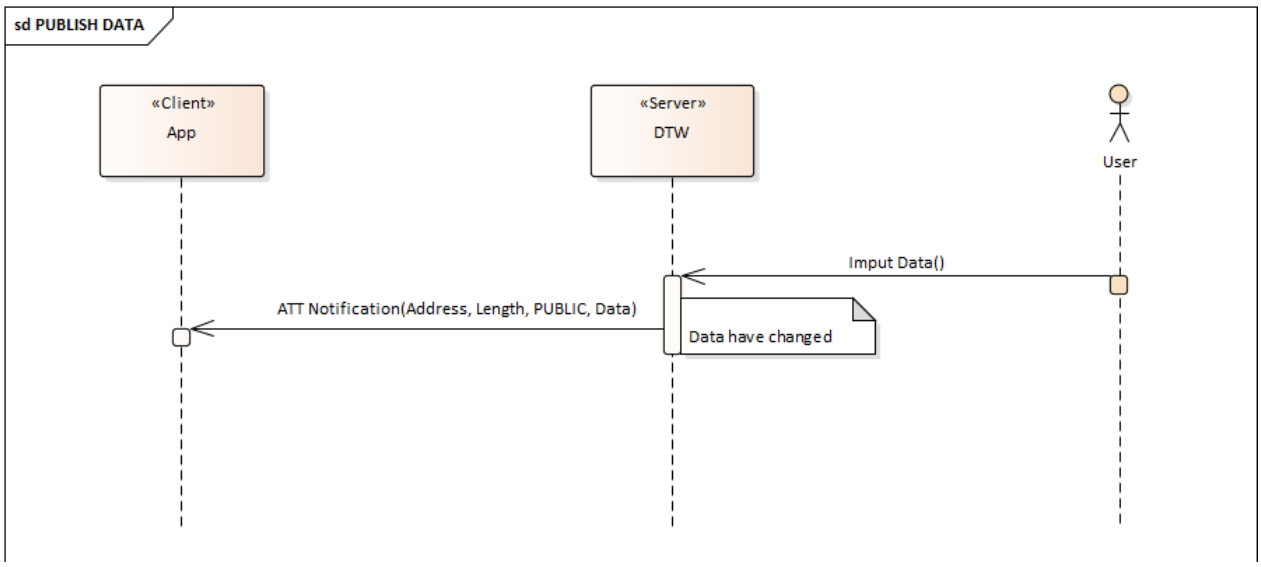
For a better understanding of the Flag-mechanism, see figures below.



Reading of data will be done with setting the READ-Flag, the server can either return the requested data or set the Acknowledge-Flag and one data byte with the Acknowledge code if the parameters from the client (address and length) where not OK and data cannot be transferred.



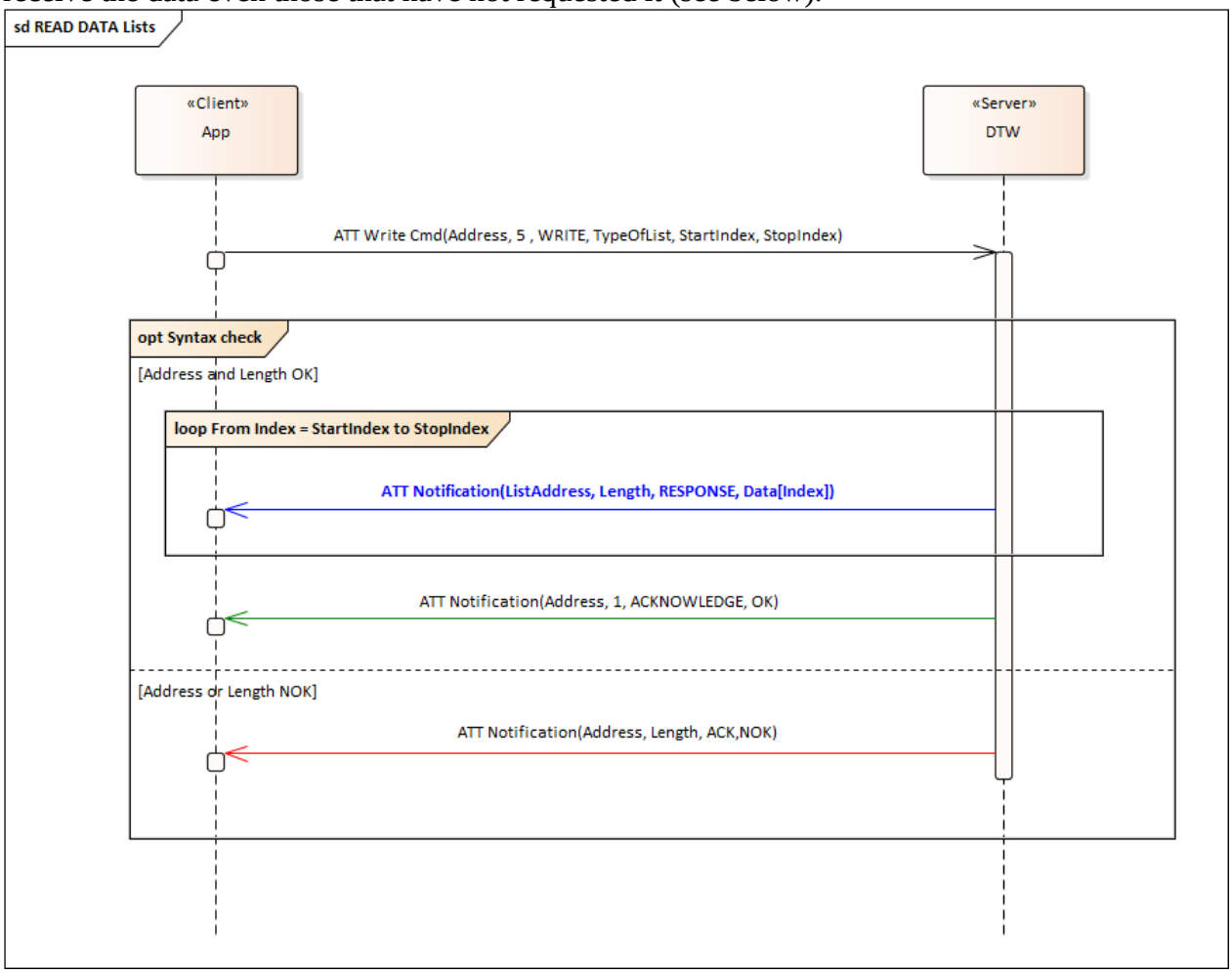
By setting the WRITE-Flag the client wants the server to take over its data. If the parameters are checked successfully the server will apply the data which will be acknowledged. If acknowledged with OK, the client knows that his data has been applied. In a multiple client connection where another client also subscribed for possible changes of data a WRITE operation must lead to a PUBLISH operation once these data are different from the current content in the moment of receiving the data.
If the data will lead to an internal status change or a change of other data, these changed data sets must also be published in order to retain the consistency. In case the parameter check failed the client will get a message with an Acknowledge NOK or a dedicated error number.



Data also will be published if the user has made changes on it, for example input a new measurement mode via buttons on the tool.

There is currently a special treatment in reading list entries. After the query for certain list entries has been received the server will send these entries one by one having the RESPONSE flag set.

In a multiple client environment, this should be avoided, since usually all connected clients receive the data even those that have not requested it (see below).



The Acknowledge to the WRITE action, that triggered the query and led to the RESPONSE of data will be sent after the data have been send. The WRITE/ACKNOWLEDE pair effectively encapsulates the RESPONSE data sets and signals to the client that the batch of responses has finished.

Data:
In the context of the GATT characteristic the data are unspecified, they only will be specified by knowing the dedicated structure behind the ADDRESS-information of the virtual memory.
How the data should be interpreted by the device is described in the device specific documentation.
In case of a message with the READ-flag set, client wants to retrieve data from the service and no data is sent. Otherwise at least one byte is necessary, for example if the message indicates an acknowledge to a writing of data, by sending the WRITE-Flag.

4.4 Virtual Address Map

The virtual address map is structured in certain data sets or blocks which represent a certain classification of the data. Every block has a certain start address, also called base address which can also be seen as an identifier for the specific data structure, block or class. Some of these blocks have a predefined base address which cannot be changed. Within the address range of the blocks sub structures can be defined, depending on the device specifics.

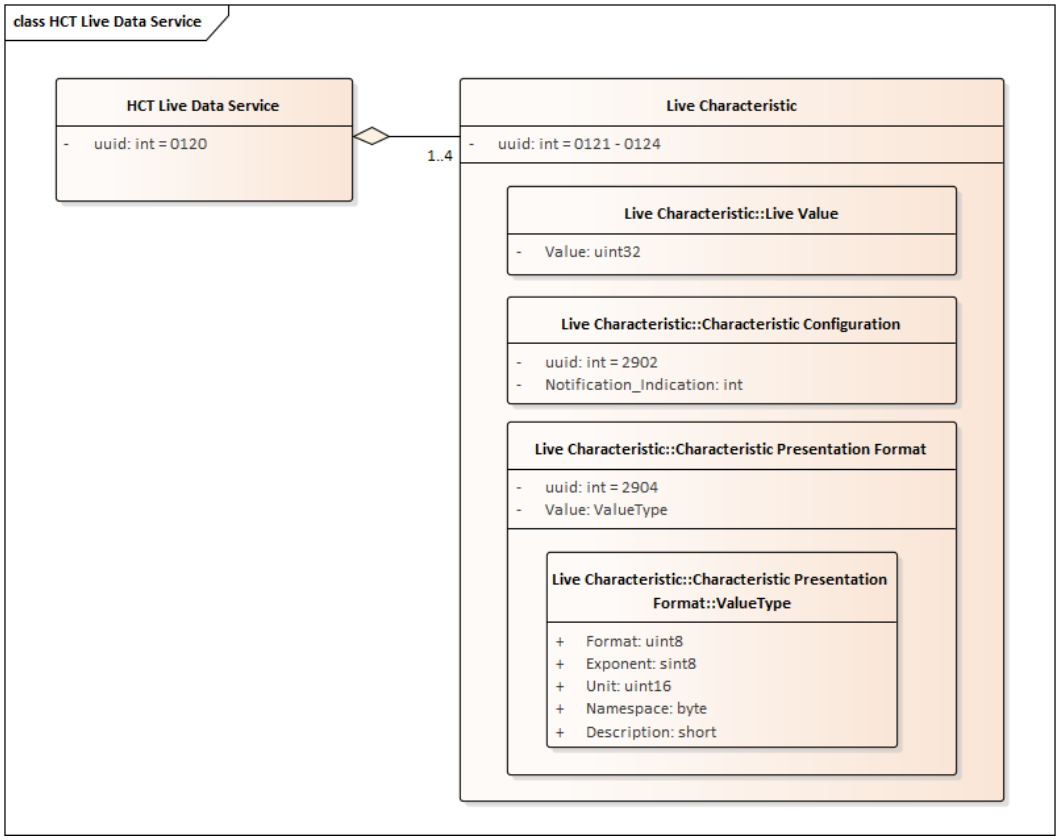
Address range		Description
0x00000000	0x000000FF	Standardized data block with device information. Description of that block is in the device specification. Even small devices/microcontrollers can handle this message - Includes e.g. information about device type
0x00000100	0x000FFFFF	Description of that block is in the device specification. Device specific data blocks - Containing the measurement data and all parameters for configuration
0x00100000	0x0010FFFF	Reserved – (Record Access Control Point (RACP) for LIST ACCESS) – SET
0x00110000	0x001^1FFFF	Reserved – (Record Access Control Point (RACP) for LIST ACCESS) – RESPONSE
0x00120000	0x001FFFFF	
0x00200000	0x0020FFFF	Address range for device specific “special” communication capabilities. E.g. BLE-characteristics for “Screen-Mirroring”, TCP/IP, UART
0x00210000	0x002FFFFF	SIG Defined Characteristics - Keyboard (Max. buffer length 100 Bytes) - Batterie (Resolution 1%) - Bond management - Date/Time - Digital I/O (n Bit ?) …
0x00300000	0x003FFFFF	- Reserved

For the 1st version there are fixed services and characteristics for BLE and only a subset of the described capabilities are available.
They are described in the Software_communication_protocol_definition_V1.1.1.xls

The use of a 4-byte address gives the possibility to have direct access to a data range of 4.294.967.295 bytes. This “virtual memory” space can be used for user accessible data and internals like updates and system configurations.

4.5 HCT Live Data Service

The HCT Live Data Service contains currently four characteristics which can be used for notifying live values. Clients must subscribe to the characteristics descriptor 2902.
Each characteristic contains a presentation format descriptor in which the format, exponent and unit of the live value is accessible.



The value – type is currently four octets which can be used as a standard 32 bit unsigned or signed value with an exponent to represent the decimals of the value, or as a float32 value. This should be described by using the SIG specified Presentation Format.

5 Interface from Tool Application Module to HCT-BLE-Module

For BLE communication the SoC from Nordic Semiconductor is used.

5.1 Embedding of the HCT-BLE-Module in a Tool - Conceptual View

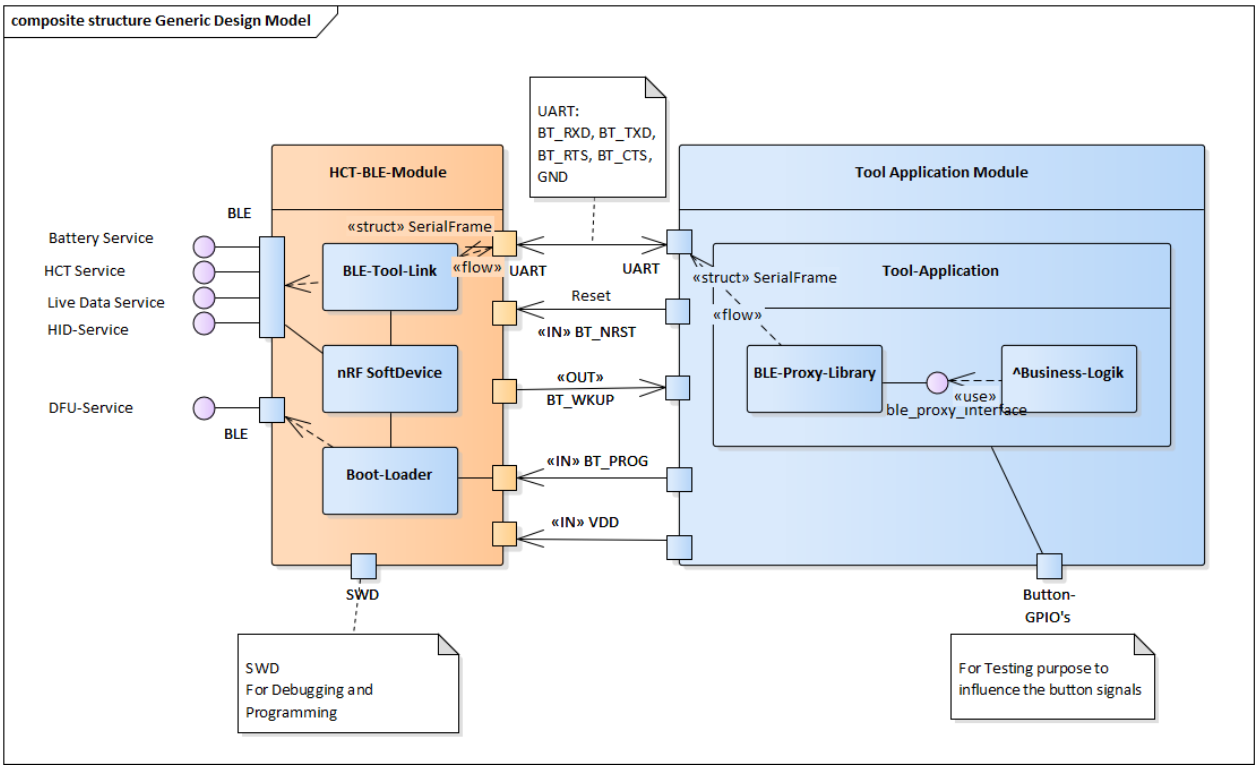


Figure 1:Conceptual View

As shown in Figure 1 the system is derived in two modules, the HCT-BLE- and the Application Module which communicate via the UART interface. Data transferred to the HCT-BLE-Module will be linked by the BLE-Tool-Link to the respective BLE-Services and characteristics.

5.2 Necessary HW Interfaces from and to the HCT-BLE-Module

5.2.1 Serial Interface

- BT_RX
- BT_TX
- BT_RTS →flow control
- BT_CTS →flow control
- GND
- VCC
- 115200 baud

5.2.2 GPIO's (From App-Module to HCT-BLE-Module)

- BT_VDD → Enable/switch on the BT-Controller
- BT_NRST → Reset the BT-Controller (low active)
- BT_PROG → Signal BT-Controller to activate DFU-Service for Download

- BT_WKUP →Signal from BT to Tool controller to wake up tool controller if a new BLE message has arrived and needs to be transferred via UART

For all lines additional scratch pads for connecting additional lines to the PC instead of the BLE-controller are necessary to test the BLE_Controller by itself and independently.

SWD-Interface with TAG-Connect 6 Pin (Pinning + Footprint will be provided)

- SWD-IO
- SWD-CLK
- Reset

5.2.3 Interface Required for Testing Purpose on the Tool-Controller

- TAG-Connect for programming the processor

Option 1:

- Serial interface for additional Test-Input in order to simulate HW-Peripherals such as Buttons, Strain-Gauge, …

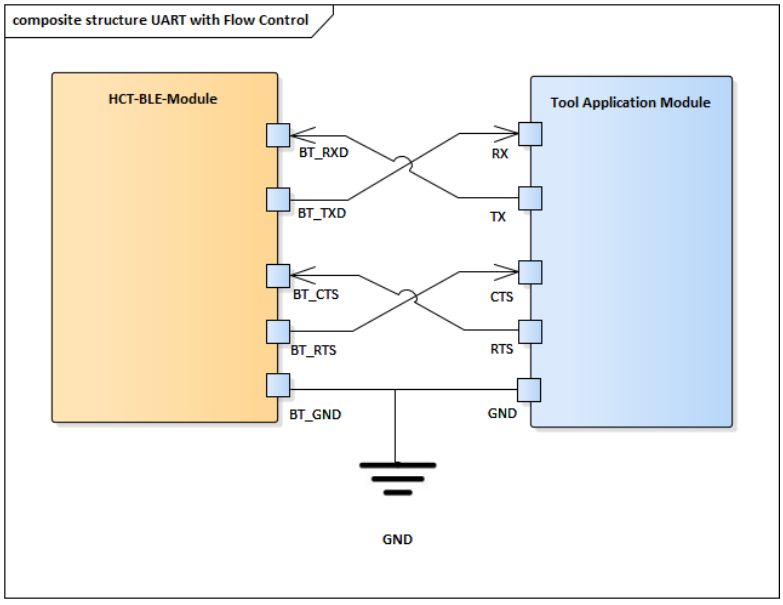
Option 2:

- Additional Scratch Pads adding the possibility for actuating the Button-Signals automatically

5.2.4 UART-HW Interface Description

Following features will be available:

- 115200 baud
- 8 Data-Bits, 1 Stop-Bit, No parity
- Hardware flow control – RTS /CTS,
- BT_CTS and BT_RTS signals are active low



5.3 UART-SW Interface Description

5.3.1 The UART-Data Frame

The data frame of the UART Protocol is in fact a specialization of the HCT-BLE-Protocol-Frame described in 4.3. It is extended by two elements, called link handles. The meaning of the link-handles is described in a dedicated chapter.

UART DATA-FRAME

Direct Link-Handle	Public Link-Handle	HCT Frame				
		ADDRESS	**LENGTH**	**FLAG**	**DATA**	**CRC N.A.**

5.3.2 Definition and Meaning of the Link-Handles

For use in the serial interface the BLE-Protocol frame is extended by two additional parameters in the front.
Internal communication between the Tool-Application and the BLE-Tool-Link is necessary, for example for setting up the BLE-connection. At the same time the BLE-Controller manages multiple BLE-connection links to several clients. Link handles represent these connections and are used to distinguish between them.
A general feature of the BLE GATT characteristic is, that all clients who have registered with a characteristic will always receive a notification message when the characteristic is changed.
The following examples illustrate the usage of link handles in typical situations in order to save communication load. The protocol is based on the publisher subscriber principle. If a client sends data to the server, the server applies these data. Then the client receives an ACKNOWLEDGE with necessary OK or NOK information. Supported by the link handles all other clients subscribed to the RX-characteristic will NOT receive the acknowledge information.
In a similar situation a client requests data by using the READ-operation. Using the link handles it is avoided that other clients receive the respective RESPONSE data without having requested it. In another use case data has changed by a WRITE-action of one certain client. This data must be published to all other subscriber clients. Using the link handles this (known) information is not send to the client that has written the data.

Direct Link-Handle:

Type	Value- Range	Description
Uint8	0	Used for direct communication between BLE-Controller and Tool-Controller
	0<n<0xFF	Any link number; added by the BLE-Controller and initiated indirectly from a BLE-Client by a READ or WRITE-Message, the handle must be repeated from Tool-Controller side, when answered with RESPONSE or ACKNOWLEDE
	0xFF	Currently not supported, Reserve

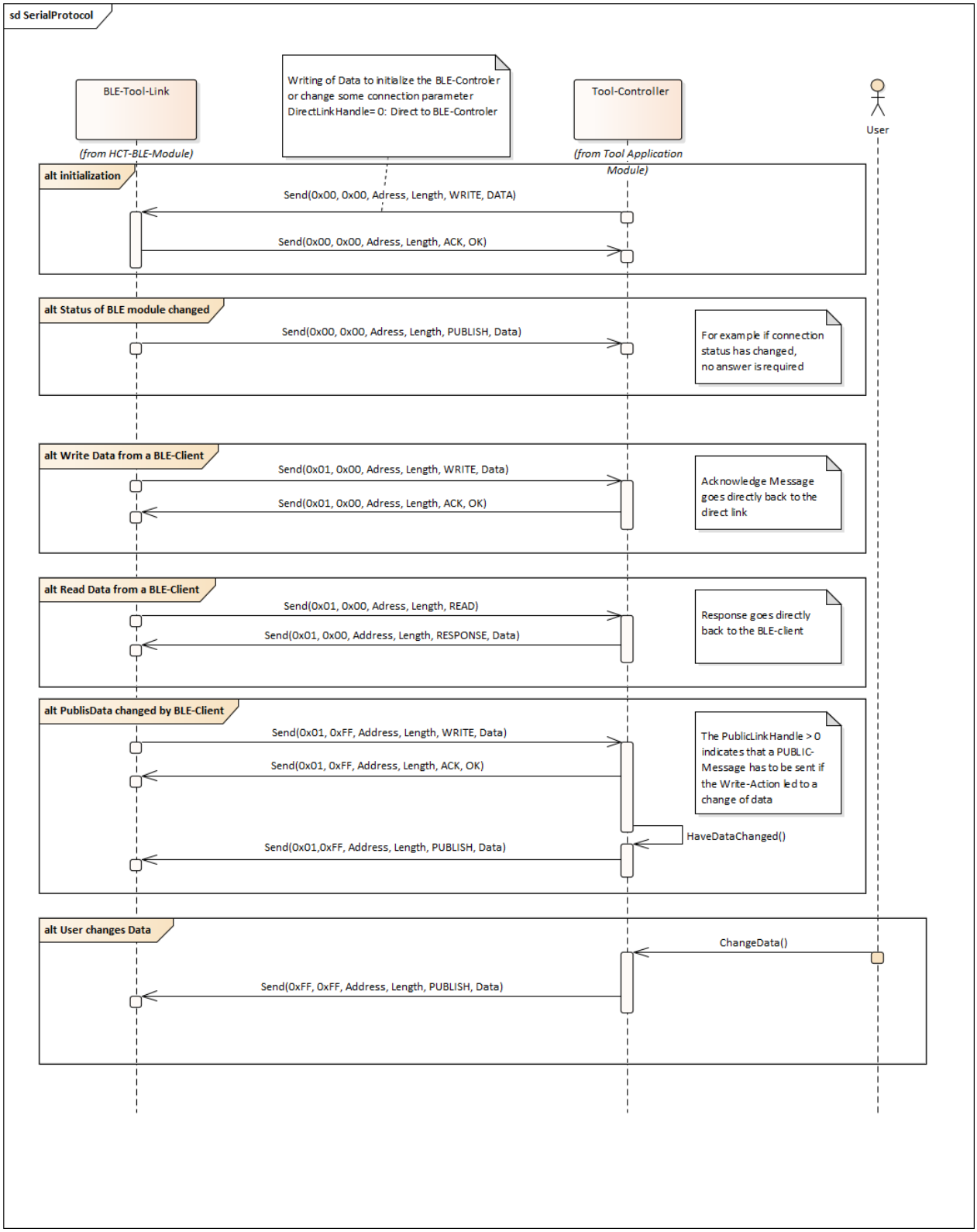
Public Link-Handle:

Type	Value- Range	Description
Uint8	0	No publication
	0<n<=0xFF	Any link number; added by the BLE-Controller and initiated indirectly from a BLE-Client by a WRITE-Message if another client has subscribed for the same characteristic. The handle must be repeated from Tool-Controller side, when answered with RESPONSE, ACKNOWLEDE.
	0xFF	If data change is initiated by the tool-controller and PUBLISH-Message is sent in order to inform the BLE-Controller to publish it to all subscribers of the respective characteristic.

5.3.3 Optional CRC-Check

To harden the communication against distortions like electric and/or magnetic fields especially at higher transmission rates a checksum can be used optionally.
UART doesn't provide an inherent error correction neither a checksum to correct or detect errors during the transmission. The BLE module therefore has a configurable checksum mechanism to include a standardized checksum value at the end of the protocol frame in a UART transmission. The checksum is placed directly after the last byte of the transmission payload.
CRC will be configurable, in the current released version (1.0.0) it is not supported.

5.3.4 Handling of Link-Handles

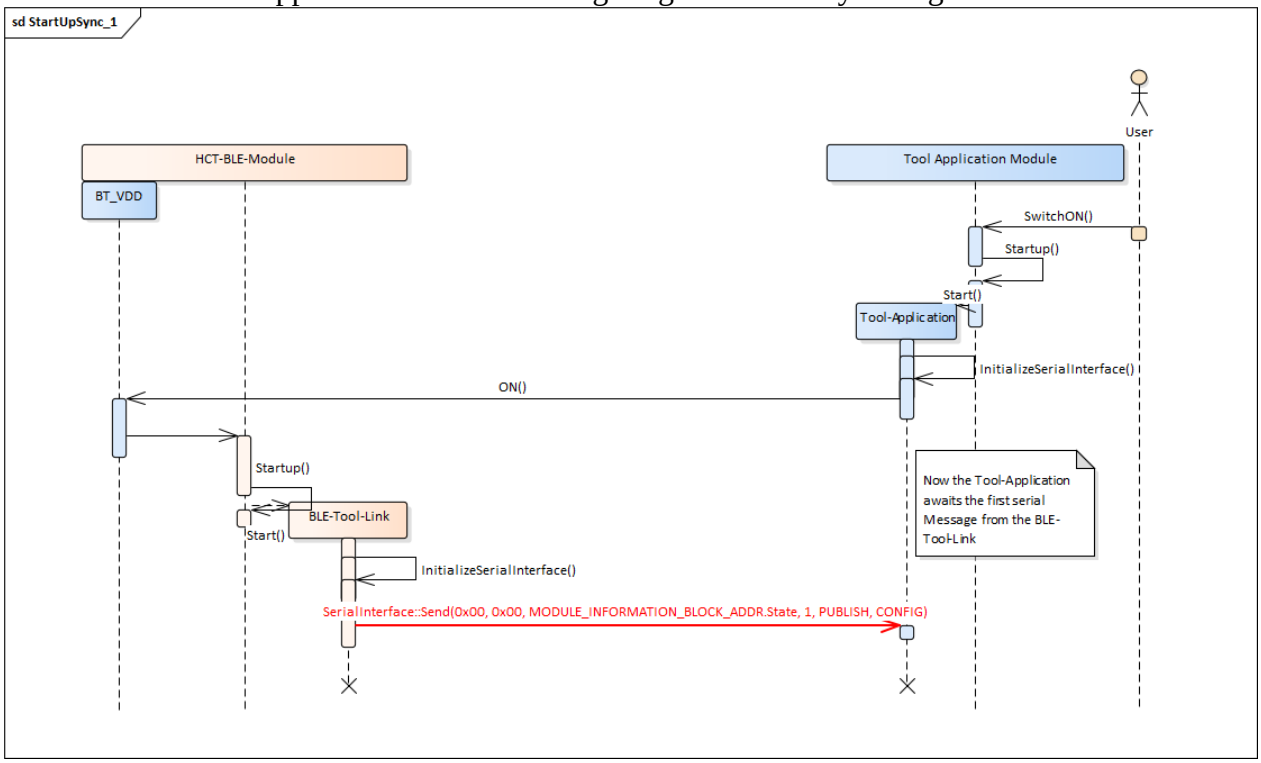


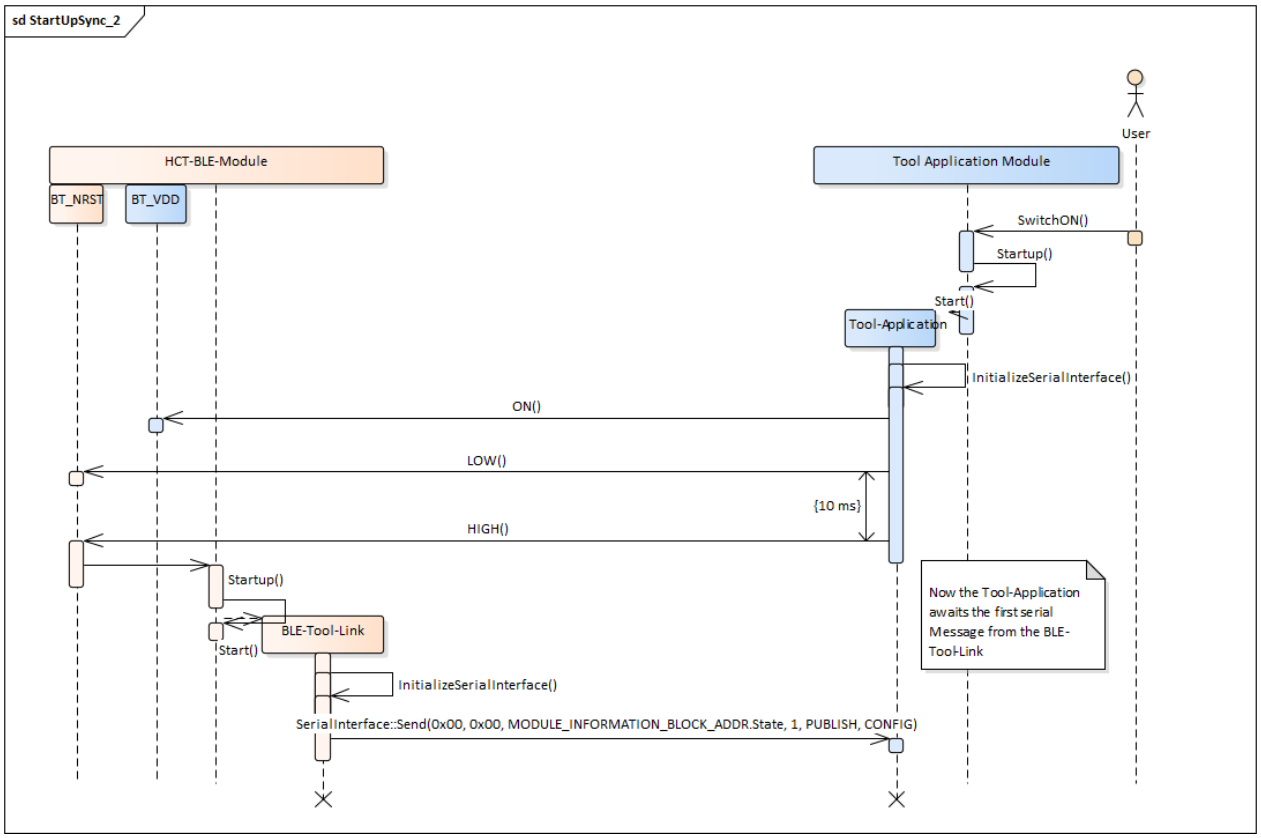
6 Use Cases

In the following chapter some use cases shall be described by the help of sequence diagrams. These sequence diagrams visualize a possible interaction between the Tool-Application Module and the HCT-BLE-Module which are not fixed yet and shall be discussed in order to find the best solution.

6.1 Startup-Synchronization between HCT-BLE-Module and Tool-Application Module

After startup both controllers must initialize their UART interface in order to start communication.
One possible way is, that after its own startup and initialization of the UART-Interface the Tool-Controller starts the BLE-Controller. Then it waits for the initialization of the BLE-Controller which sends is module status which signals, that it must be configured now.
After the BLE-Tool-Link has initialized the UART, it will send the state CONFIG in order to show that the Tool-Application can start with giving the necessary configuration information.





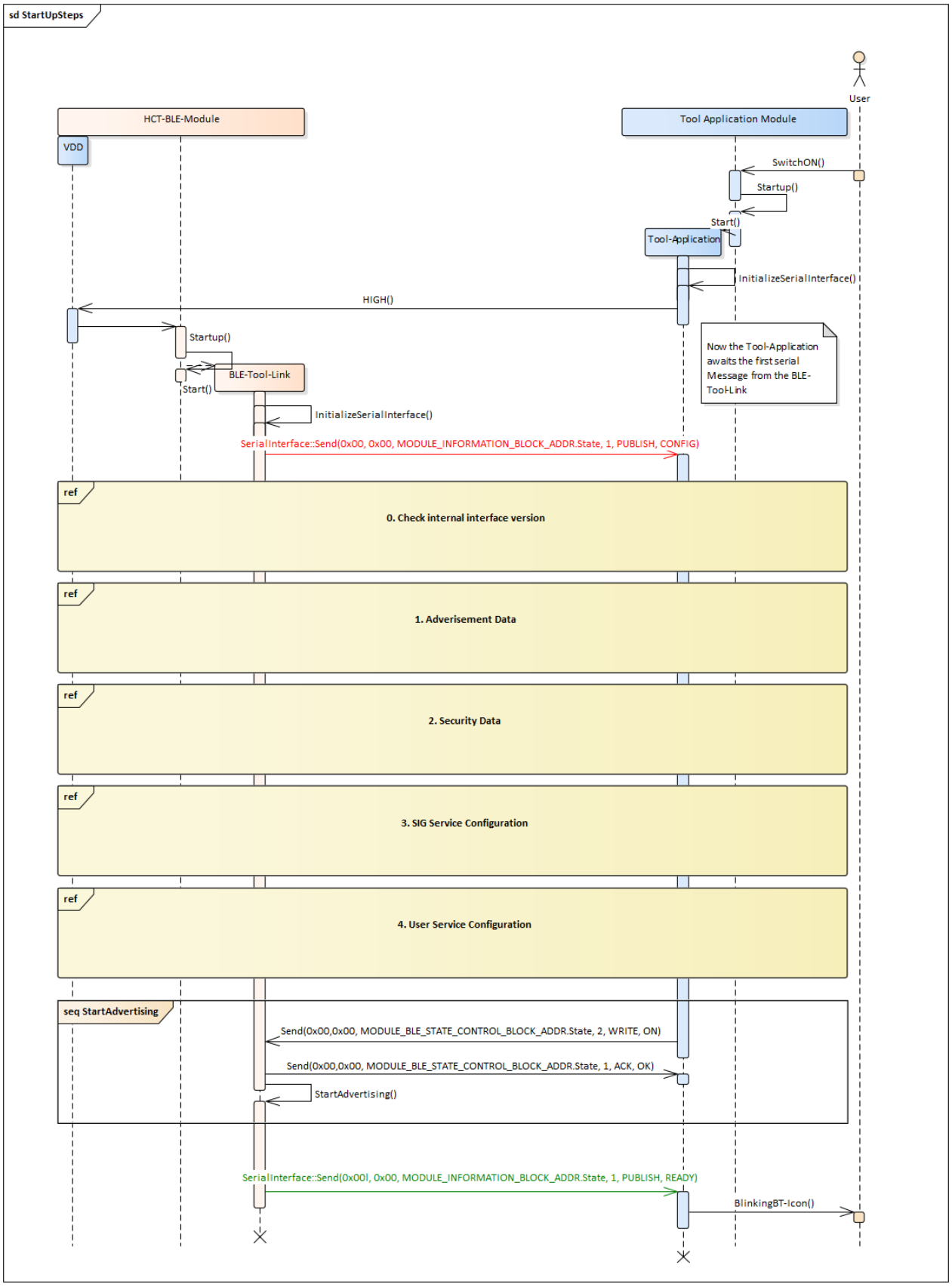
The second option is to use the BT_NRST additionally, see figure above. Since the BT_NRST is a software configurable PIN it must be configured in the Bootloader. That means, that the Bootloader must be able to start and configure the Pin in order to be recognized as a reset-pin.

6.2 Initialization of HCT-BLE-Module

The tool-application starts the BLE-specific configuration of the HCT-BLE-Module.
The compilation of the necessary data is supported by our BLE-Proxy-Library, see [2]. It contains the necessary data and only exposes data to be changed, i.e. tool specific data such as the tool name in the advertisement.
For a better understanding, the configuration is divided in five parts:

- Advertisement Data,
- Security Data,
- Configuration of predefined SIG-Services,
- Configuration of proprietary, user defined services.

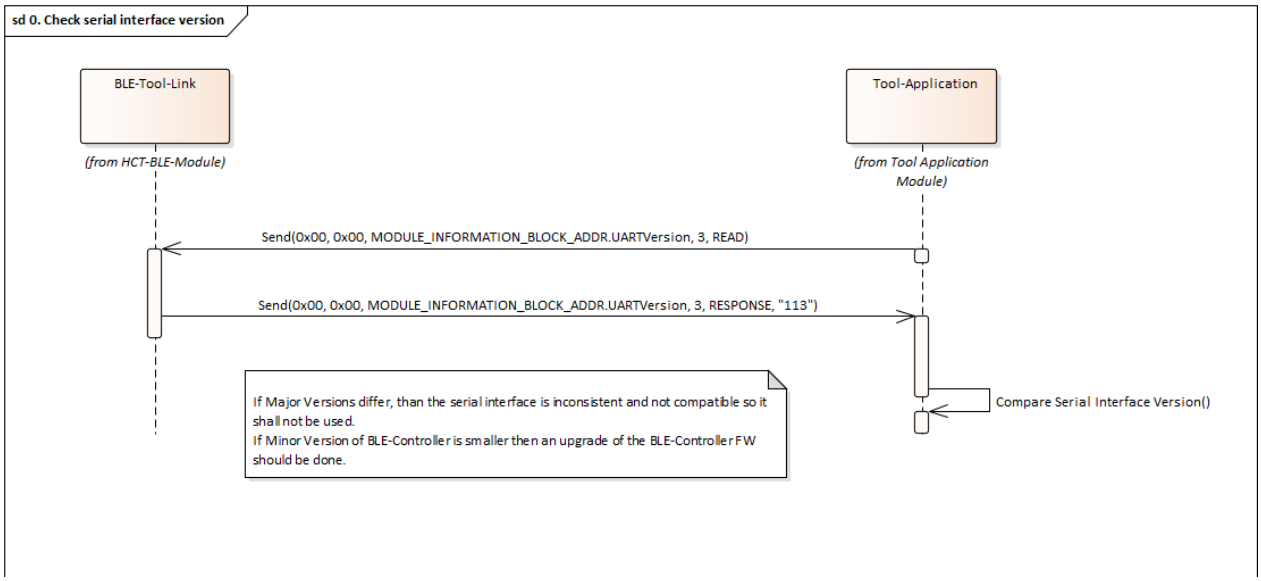
After having send this information the Tools SW will signalize the BLE-Controller that it can start advertising.


sd StartUpSteps
User
HCT-BLE-Module
Tool Application Module
VDD
SwitchON()
Startup()
Start()
Tool-Application
InitializeSerialInterface()
HIGH()
Startup()
Now the Tool-Application awaits the first serial Message from the BLE-Tool-Link
BLE-Tool-Link
Start()
InitializeSerialInterface()
SerialInterface::Send(0x00, 0x00, MODULE_INFORMATION_BLOCK_ADDR.State, 1, PUBLISH, CONFIG)
ref
0. Check internal interface version
ref
1. Adverisement Data
ref
2. Security Data
ref
3. SIG Service Configuration
ref
4. User Service Configuration
seq StartAdvertising
Send(0x00,0x00, MODULE_BLE_STATE_CONTROL_BLOCK_ADDR.State, 2, WRITE, ON)
Send(0x00,0x00, MODULE_BLE_STATE_CONTROL_BLOCK_ADDR.State, 1, ACK, OK)
StartAdvertising()
SerialInterface::Send(0x00l, 0x00, MODULE_INFORMATION_BLOCK_ADDR.State, 1, PUBLISH, READY)
BlinkingBT-Icon()

In the following section the Start-Up steps will be described in detail:

6.2.1 Check Serial Interface Version

In order to be able to exchange the internal data for setting up the BLE-Controller, the consistency of the data-structure interface must be ensured. Therefore the Application controller must query first the serial protocol version.

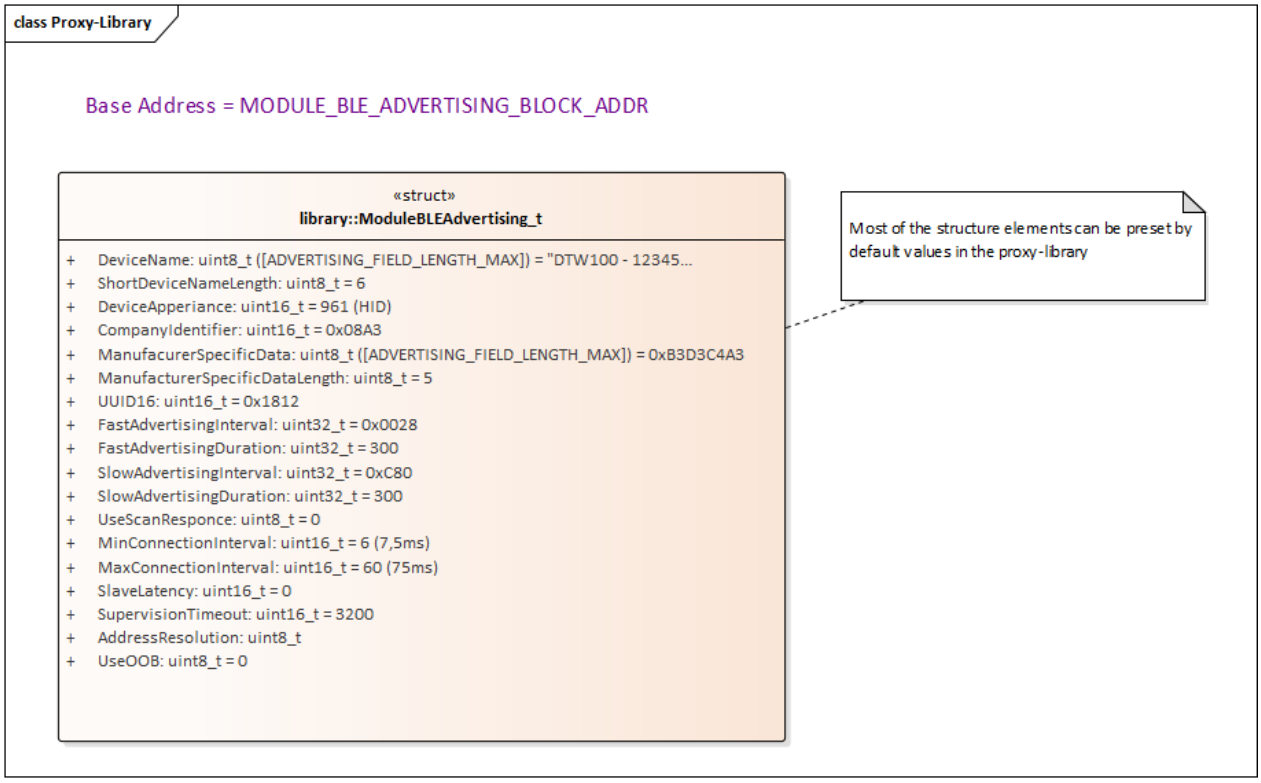


The version is divided into <Major>.<Minor>.<Step>., see Software_communication_protocol_definition_V1.1.1.
The Major-Version Number of the Interface should be the same between BLE-Tool-Link and Tool Application, as a difference indicates a not backward-compatible change between the structures. The Minor-Version of the BLE-Tool-Link should not be smaller than the Minor-Version of the Library in the Tool-controller. If the Minor-Version of the BLE-Tool-Link is higher it indicates a backward-compatible change that does not require an Upgrade of the FW, see also Table.

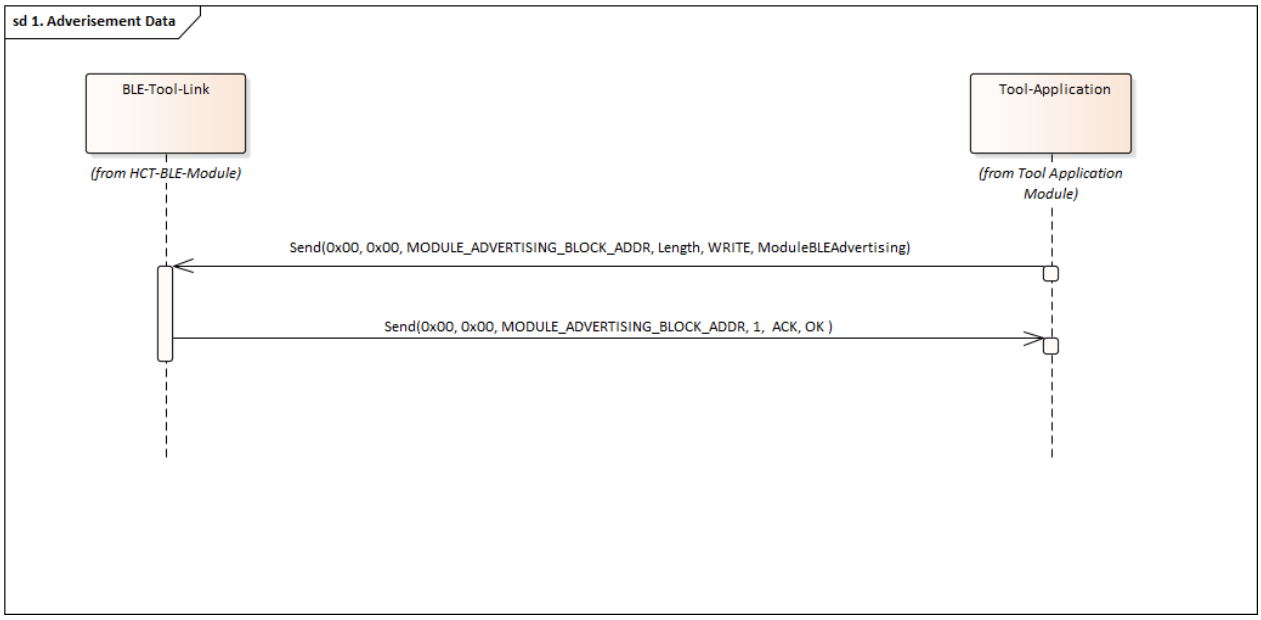
BLE-Tool-Link Serial Version		Tool-Application:: Proxy-Library-Version	Action required.
Major	>	Major	Serial Interface cannot be used due to inconsistency. Application Controller FW upgrade with newer Proxy-Library required or downgrade of the BLE-Controller FW-Version to the same Major (by using BT_PROG and BT_NRST Pin)
Major	<	Major	Serial Interface cannot be used due to inconsistency. BLE-Controller FW-Version must be upgraded by using BT_PROG and BT_NRST Pin
Minor	>	Minor	Backward compatible change/extension, upgrade of the Tool-Controller FW possible but not necessary

Minor	<	Minor	Serial interface can be used, but new extended functionality is not supported. Upgrade of BLE-Controller FW needed if extended functionality shall be used. BLE-Download can be initiated by the serial interface.
Step	<>	Step	Serial interface can be used. Changes which do not affect the structure and the compatibility. Will be used to trace some steps during the development or smaller bug fixes.

6.2.2 Advertisement Data

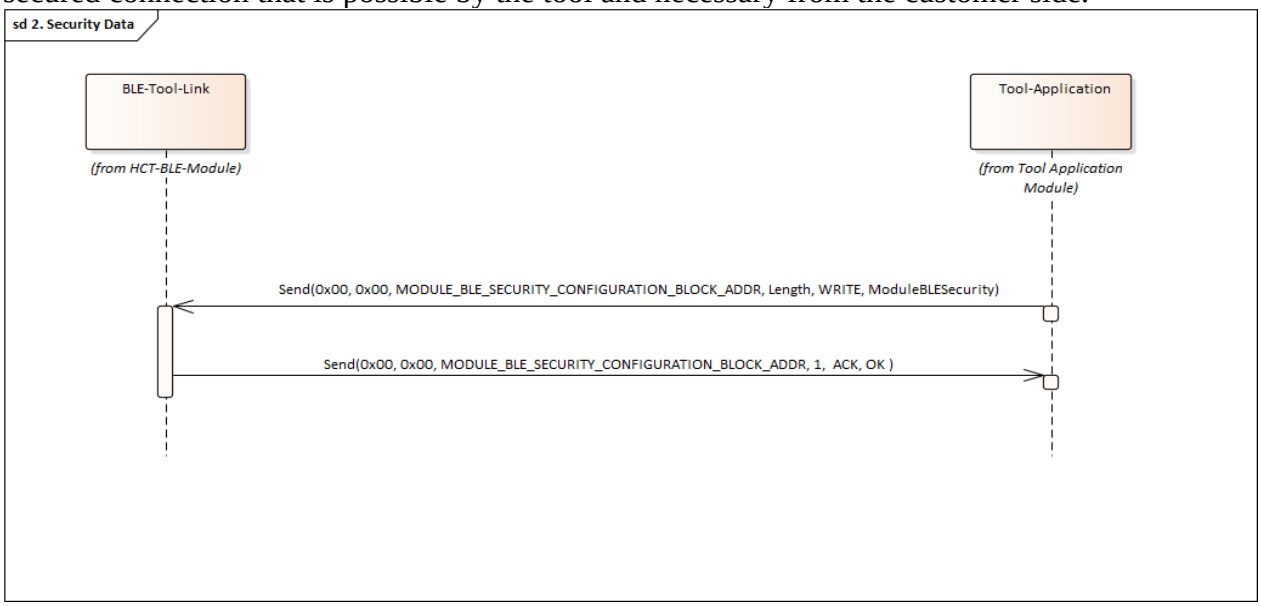


Advertisement configuration data are hosted in the virtual Hoffmann-Protocol address-space with its base address, see figure above or [1]. The structure can and will be preset with default values by the BLE-Proxy-Library, however some values need to be filled explicitly, for example the DeviceName.
The Tool-Application then must send the data via the serial protocol to the BLE-Tool-Link and will get an Acknowledge with OK, if data was taken over.



6.2.3 Security Data

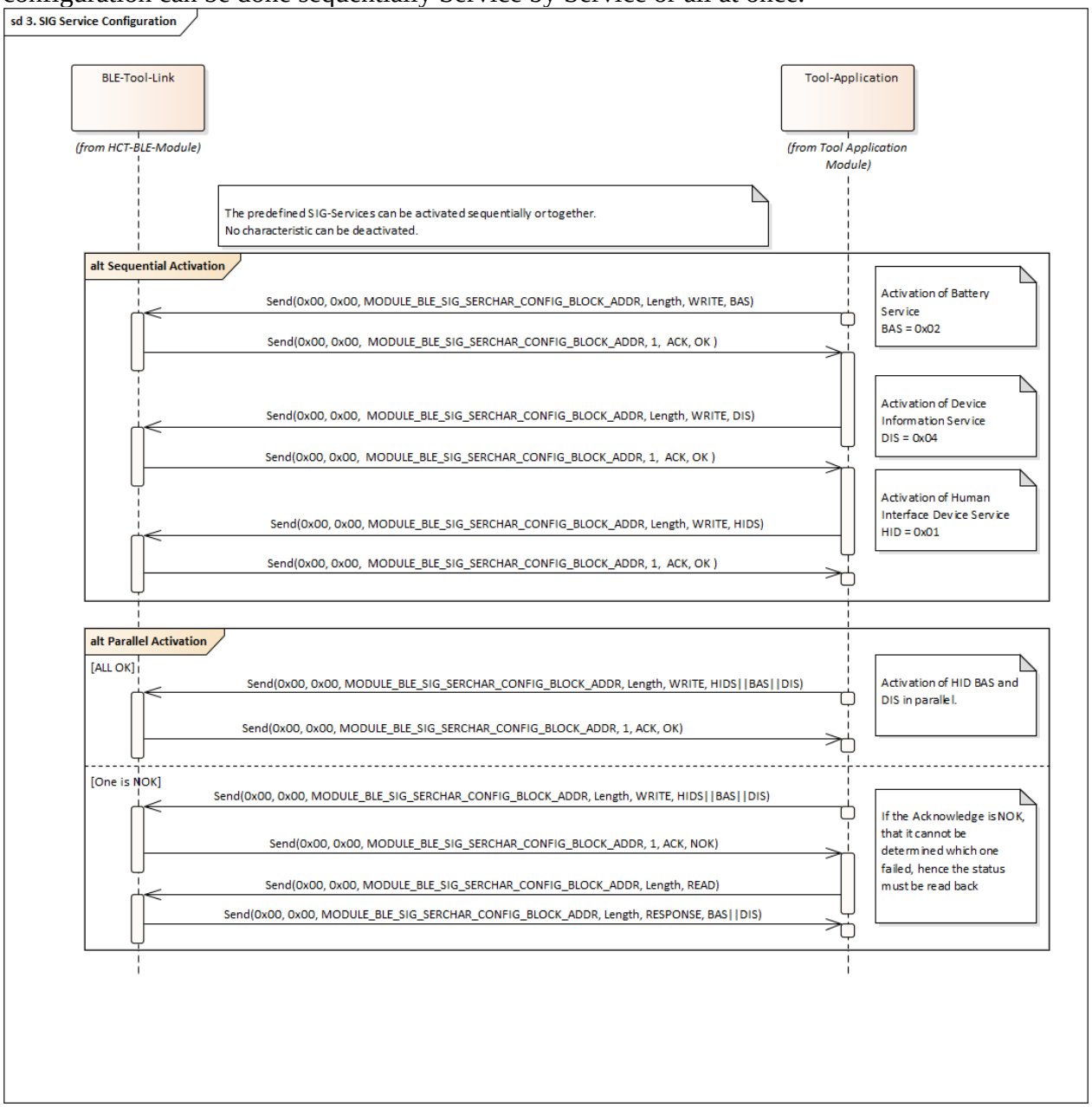
The current structure of the security data configuration block at address MODULE_BLE_SECURITY_CONFIGURATION_BLOCK_ADDR is verry open and offers a variety of possibilities to configure security options, however only some presets are helpfull. Here the intention is, to collect parameters, for example the type of address resolution (random, resolvable, private,…) or which authentication method must be chosen. The principle is the same as with the advertisement configuration data – the structure will be preset by the proxy-library and can be changed for a few dedicated parameters, depending on the type of authentication and secured connection that is possible by the tool and necessary from the customer side.



6.2.4 SIG Service Configuration

The Sig Service Configuration block can be found at MODULE_BLE_SIG_SERCHAR_CONFIG_BLOCK_ADDR.
SIG defined Services are preset by the BLE-Controller and don’t need additional parameters. For the tools, the Battery-Service (BAS), Device Information Service (DIS) and Human Interface Device Service (HIDS) will be available and can be activated by the configuration.

The control-parameter SerCharConfiguration_1_32 is a bitfield where each SIG-Service will be represented with a dedicated DEFINE. By setting the configuration and sending it to the BLE_Tool-Link it will setup the activated Services in the BLE-Stack.
Important to know is, that a Service can only be added to but not removed from the BLE-Stack. If for example the HID is not yet activated on the DTW it can startup and connect to a mobile application. If then later the HID is needed, it can be activated by sending the SIG Service Configuration block again with only the activated HID-Bit. If a service is supposed to be deactivated, that only can be done by performing shut down of all connections and a reset of the BLE-Controller and a restart of the configuration steps. As shown in the figure below, the configuration can be done sequentially Service by Service or all at once.

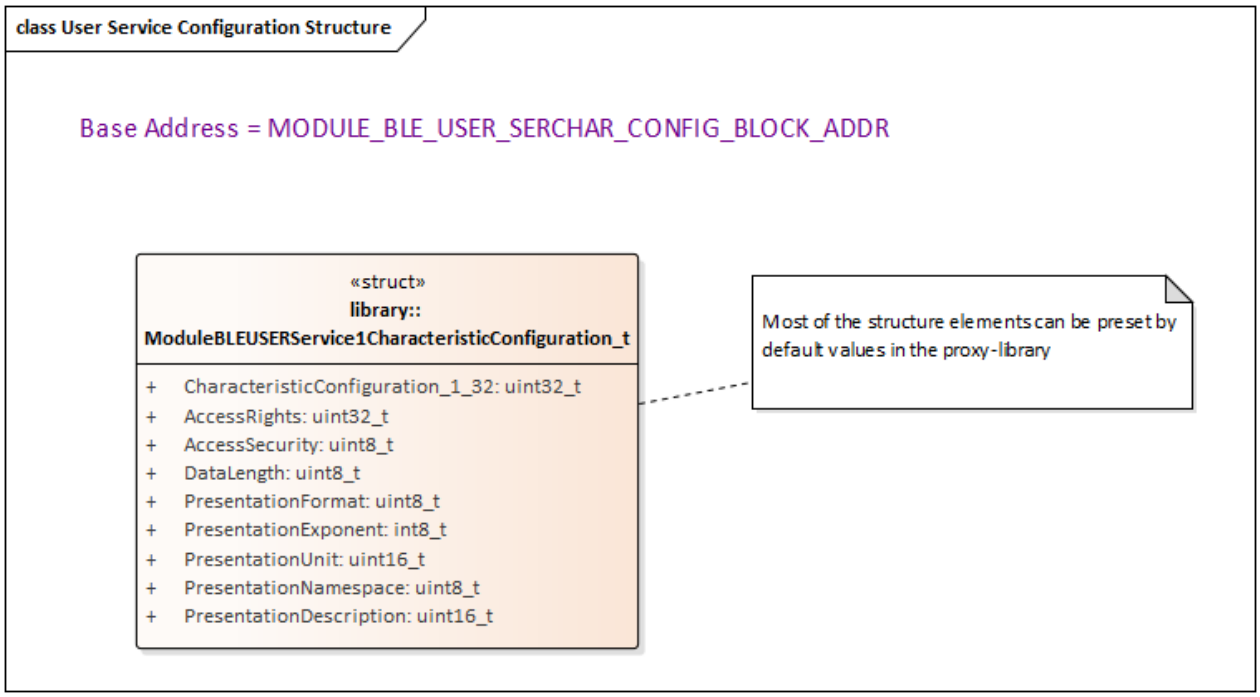


6.2.5 User Service Configuration

In the User Service Configuration the proprietary services and their characteristics will be activated. For each characteristic some more data has to be sent to the BLE-Tool-Link.
Note: Most of the data is already filled with default values by the BLE-Proxy-Library.
Every added characteristic included in a certain service will be represented in the bitfield CharacteristicConfiguration_1_32.
([UserServiceX]-> CharacteristicConfiguration_1_32)
The current proprietary services and characteristics are:

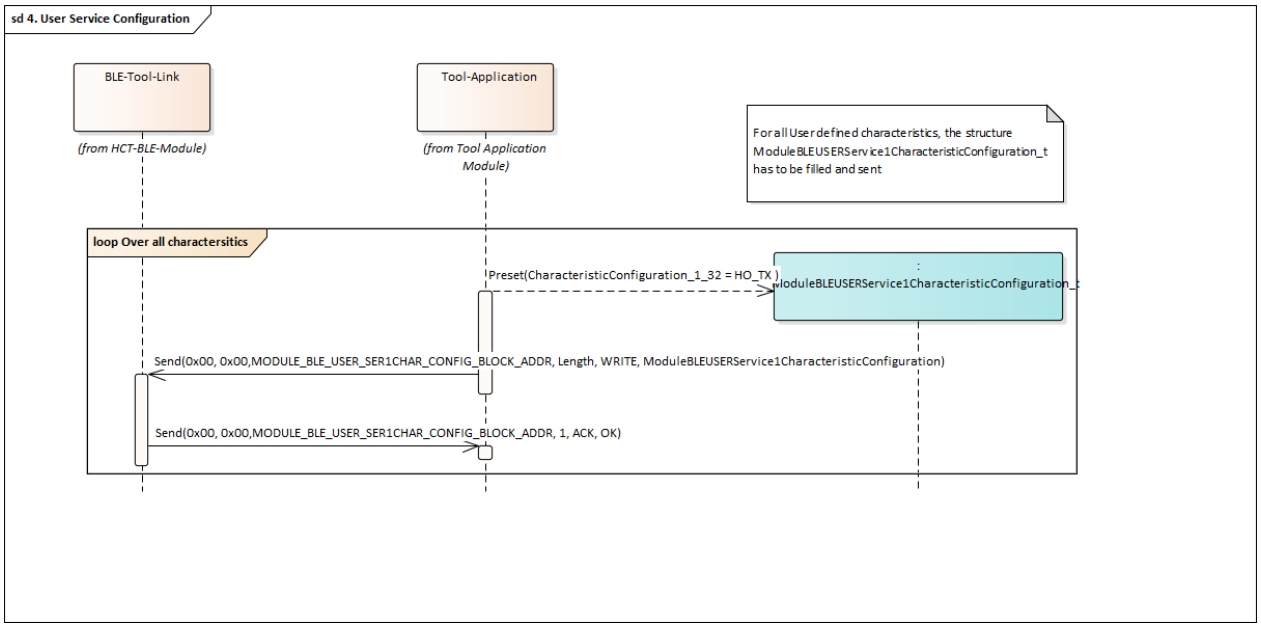
Characteristic description	**included in (service)**	**Currently used for**	**Value of the bit in bitfield in HEX**
Protocol TX (BLE Central Out -> peripheral In)	HOS	Communication data/command	1
Protocol RX (BLE Central In -> Peripheral Out)	HOS	Communication data/command	2
Measurement Characteristic 1	LDS	Actual Torque value live view	1
Measurement Characteristic 2	LDS	Peak Torque value live view	2
Measurement Characteristic 3	LDS	Actual Angle value live view	4
Measurement Characteristic 4	LDS	Peak Angle value live view	8

For each characteristic, the access right as well as the presentation format can be specified.



6.2.5.1 *Dynamic Length Configuration of User Service Characteristics*

Especially in regard to an optimized sending of live values via the serial interface there is a possibility to define a dedicated length and a virtual start address of these live values in the HCT protocol address space (`VirtualStartAddress`) for each user defined characteristic configuration. These values can now directly be placed sequentially in the virtual adress map of the protocol. So it is possible to send for example four values in one serial telegramm. In the dynamic configuration each subsequent characteristic value's virtual start adress must have an offset to the predecessor by exactly the length in bytes of the predecessor. The BLE-Tool link will not verify this.



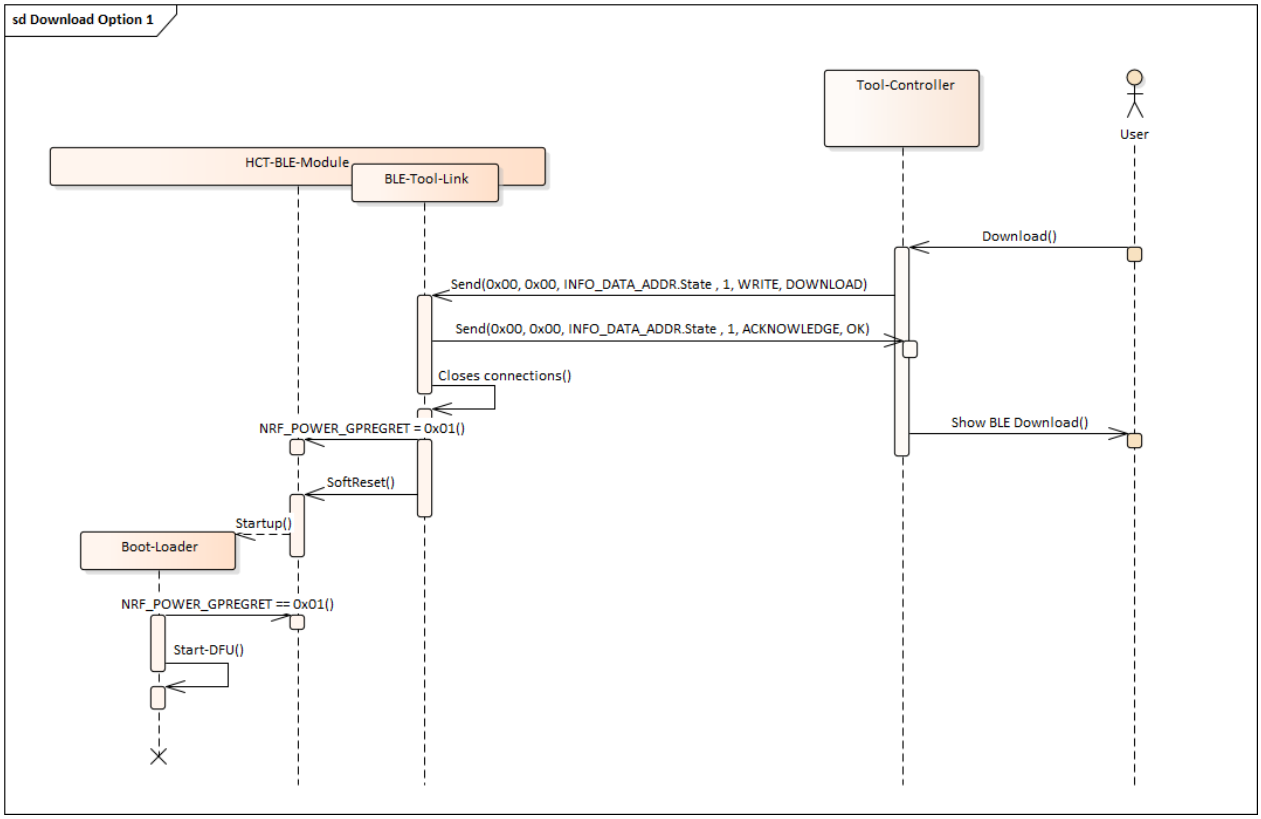
As shown in the figure above the configuration of the characteristics must be done sequentially, if the parameter set differs from the other characteristics. For the live characteristics torque or angle one set can be sent by bitwise OR-junction in the *CharacteristicConfiguration_1_32* .
In the BLE-Proxy-Library, the data will be preset by default values.

6.3 Firmware Update Initialization

The firmware update initialization can and must be done in one of two ways.

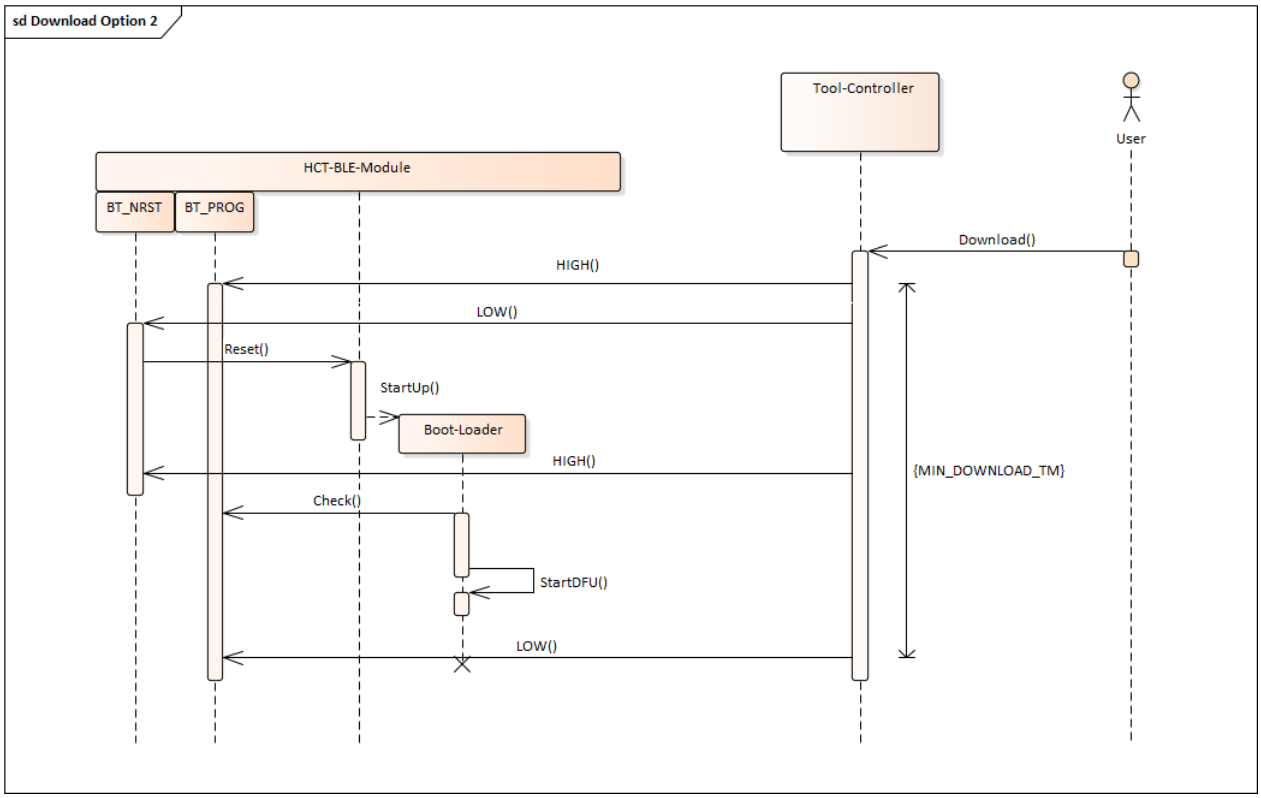
6.3.1 Method 1: Triggered by User via Serial Protocol

The first method can be used if the BLE-Tool-Link is already started up and working. The download can also be initiated by a client connected to the Hoffmann-Service.
If the BLE-Tool-Link is connected to the Tool-Controller's software, the user can trigger or initiate the BLE-Download via a button, an external tool or a certain menu point. The tool-SW then must send the described command by setting or requesting the DOWNLOAD – State. When the Tool-Controller gets the respective acknowledge, it can show the user by means of display or LED-sign that the BLE-download has been initiated and that the user now can connect to the advertised download-service, for example DFU-Service.
The figure shown visualizes the necessary sequence for initiating this process.



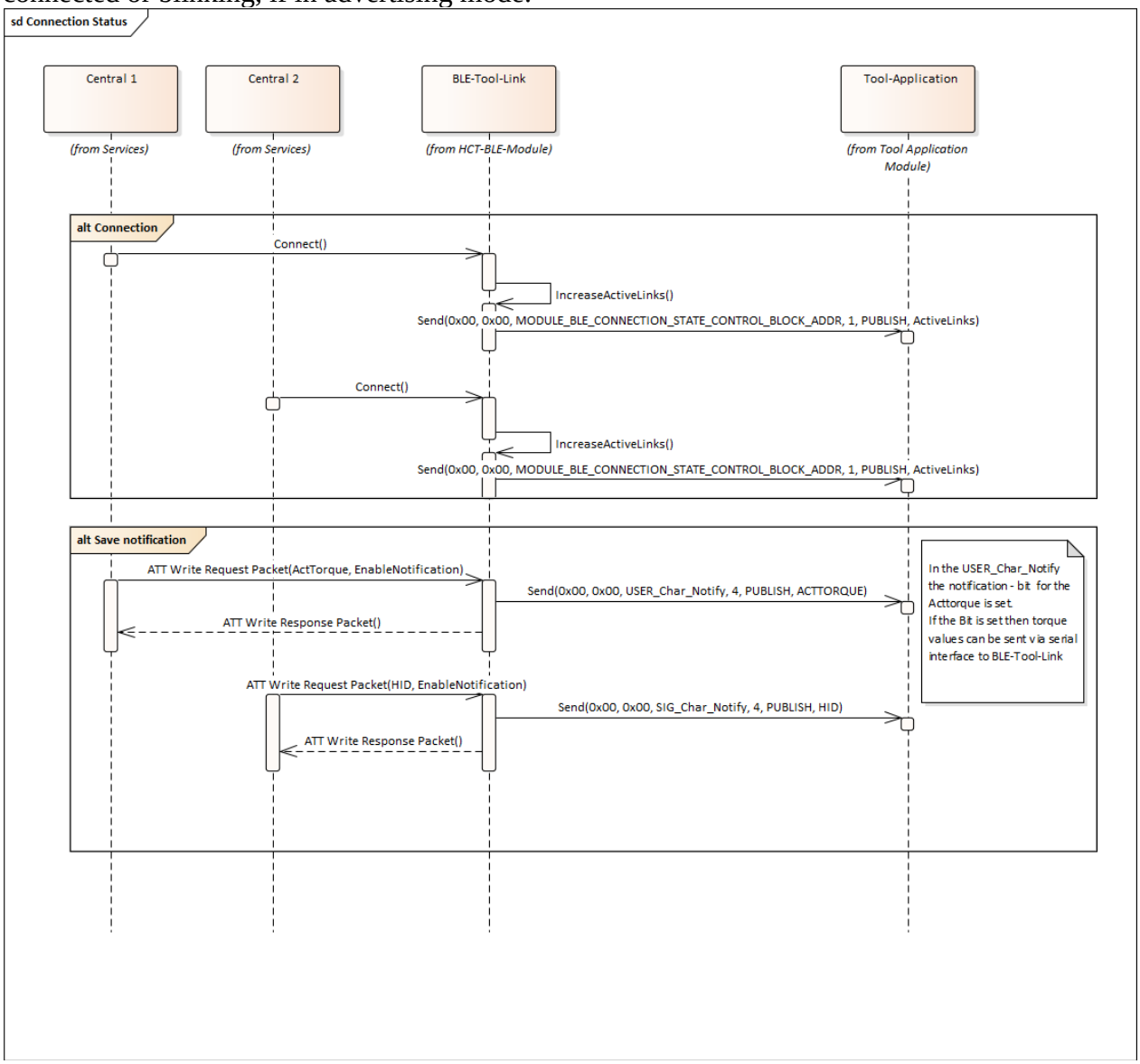
6.3.2 Method 2: Triggered by the User if no Application or a not Working or Corrupt Application is Loaded

In this case the application software on the HCT-BLE-Module might not be working. The reason could be that the application is corrupt and will crash after start, the bootloader cannot recognize this, or the trigger functionality of the serial interface is faulty, so that the application will not recognize the serial or BLE-trigger.
In such case the Tool-Controller must set a dedicated (BT_PROG)-Input and initiate a reset by switching the BT_NRST to LOW. In consequence the HCT-BLE-Module will start up and the started Boot -Loader will check the BT_PROG-Input. The initiating controller must hold this BT_PROG-Input to high for a minimum time-frame, here called MIN_Download_TM.



6.4 Connection Status

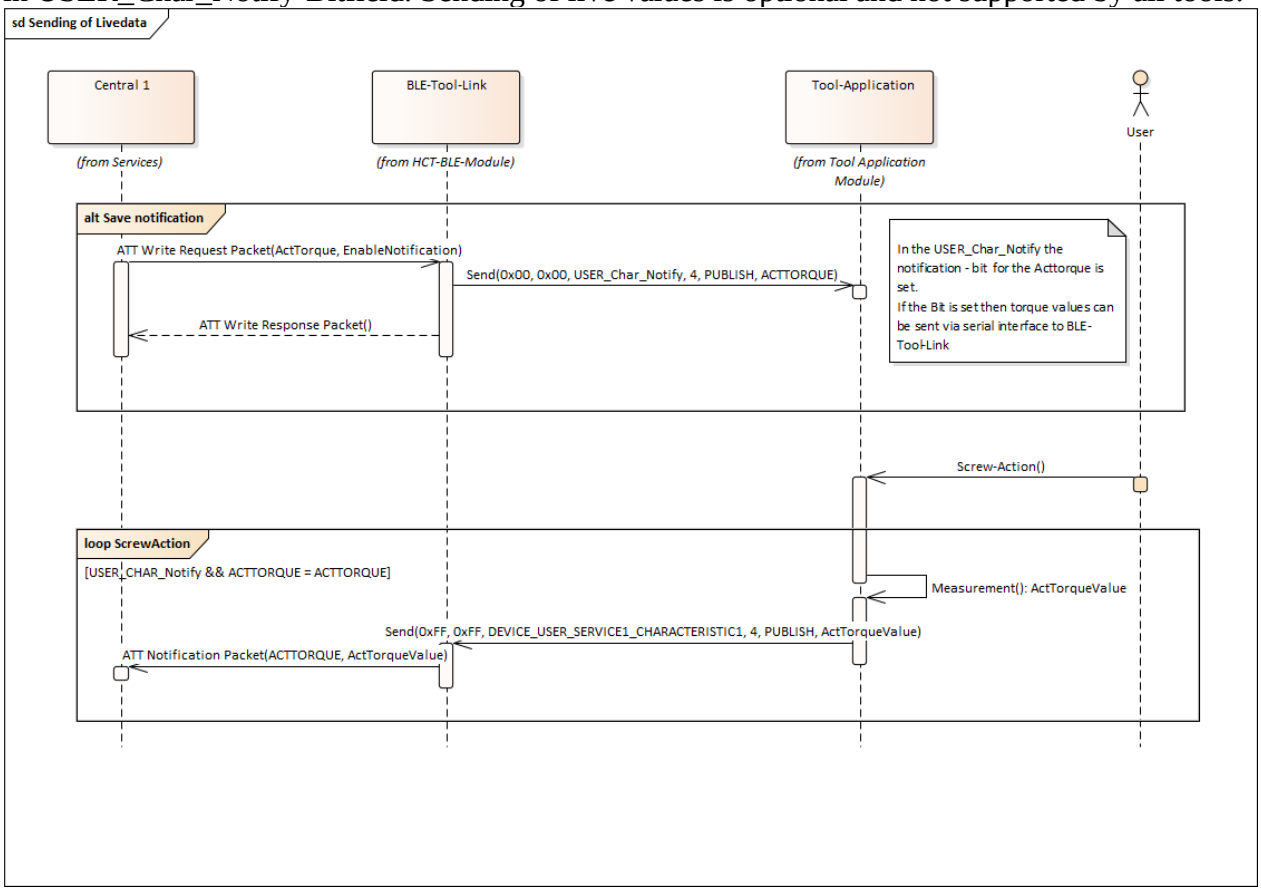
The connection status of the BLE-Tool-Link is represented in the ModuleBLEConnectionStateControl-Block at address `MODULE_BLE_CONNECTION_STATE_BLOCK_ADDR` of the virtual address space of the SW_Communication_Protocol.
The parameter ActiveLinks will be increased once a client connects to the device and will be decreased once a client disconnects. If a client subscribes to a configured SIG or User-defined characteristic for data notification the respective bit in the parameter SIG_Char_Notify- or User_Char_Notify parameter is set. Every change of it is published via the serial interface to the Tool-Application. The state is held consistently in the proxy-library. It is needed from the Tool-Application to show the connection state, for example the solid Bluetooth-Symbol when connected or blinking, if in advertising mode.



6.5 Live Data

Live data is sent by using the same protocol frame as the other data-commands.
For each characteristic the Client must subscribe to the notification using the Client Characteristic Configuration descriptor. BLE-Tool-Link sets the respective information in the

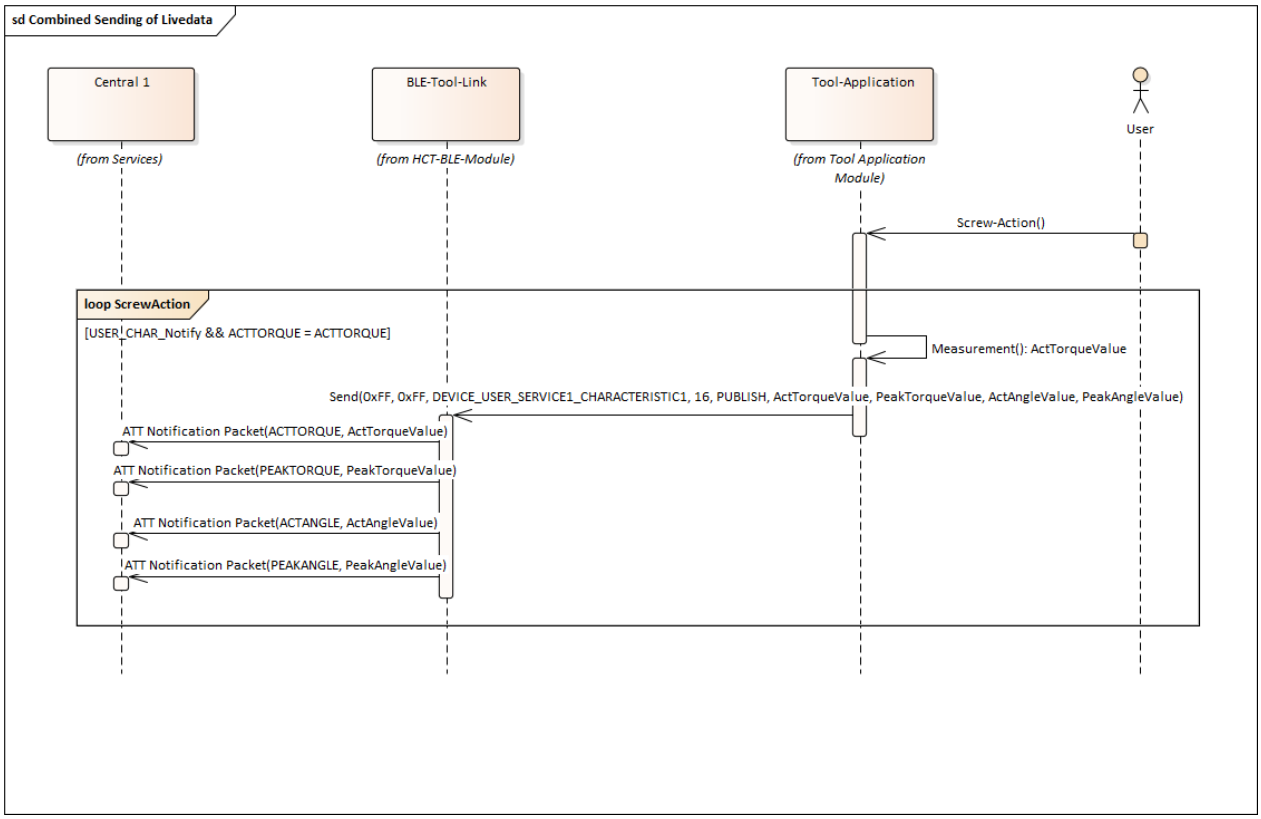
USER_Char_Notify-Bitfield of the ModuleBLEConnectionState-block. Indication is currently not supported but also not needed.
Goal is to prevent sending of Live-Values via the serial interface, if no client has subscribed to the respective characteristics.
If the user starts a measurement on the tool, the Live-Values will be sent depending on the set bit in USER_Char_Notify-Bitfield. Sending of live values is optional and not supported by all tools.



6.5.1 Combination of Sending of Live Values

In order to save overhead and to ensure a deterministic (equidistant time triggered) sending of for example up to four Torque- and Angle- live values in one serial message, the respective User-Characteristics must be configured as described in 6.2.5.1. If this is ensured, then the address of the first of the live values in the virtual address map must be used as the address in the serial protocol frame. Please note that the BLE-Tool-Link does the mapping of the subsequent values according to their virtual address and length information.
It sends all characteristic values when

- Subsequently received with the serial message from the Application Module and
- a subscription for the characteristic value is active.



6.6 Additional Use Cases that Will be Defined by Hoffmann

- Trigger Download via Hoffmann-Protocol
- HID Switch ON/OFF
- Switch OFF or reset BLE-Controller

7 Proxy-Library for Easier Integration

To make it easy for our suppliers to integrate BLE into their products, we offer an easy-to-integrate proxy library called BLE-Proxy-Library.
First, a fixed module configuration is used, later a dynamic configuration and a more general approach will be implemented.

8 Documents

1.	Address-Map for protocol description: Software_communication_protocol_definition_V2.3.0.xlsx
2.	BLE-Proxy-Library_description_Vx.x.x.docx
3.	BT advertising name concept of HCT tools.xls
4.	https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Descriptors/org.bluetooth.descriptor.gatt.characteristic_presentation_format.xml