Version: 1.00

LEGO® MINDSTORMS® NXT ARM7 Bluetooth® Interface Specification

TABLE OF CONTENTS

HARDWARE INTERFACE

The Bluetooth® chip from CSR, named BlueCore™, contains all the necessary hardware to run a completely self-contained Bluetooth node. A 16-bit integrated processor runs the BT-Stack implementation from CSR called BlueLab. This firmware integrates a user programmable VM-task enabling us to control the BT node and run small amounts of application code. We have integrated a command interpreter in the VM that decodes and responds to commands received through the UART.

Our VM-code is a full implementation of both the BT SPP-A and SPP-B profiles. The two SPP profiles differ in the way a connection is established with remote BT-nodes: SPP-A is used when the local BlueCore™ chip is the connection initiator while SPP-B is used when the remote node initiates the connection.

BlueCore™ uses "stream mode" to exchange data at a rate of <= 220K baud after a connection is established. This effectively emulates a serial cable between the two connected BT-nodes. The UART is used in both stream mode and command mode (which is used to control the VM application within BlueCore™ and by extension, the Bluetooth functionality within the NXT).
The state of the UART channel is controlled by two interface signals (ARM7__CMD & BC4__CMD). The program runs on BlueLab version 3.2.

The main NXT processor that controls our user interface provides drivers for peripherals and runs user code.

CONTROL SIGNALS

Reset:	Active low signal initiated by the ARM7 that resets the BlueCore™ chip.
ARM7__CMD:	Active high signal that signals the state of the UART channel seen from the ARM7. Input at BlueCore™ PIO(11).
BC4__CMD:	Active high signal that signals the state of the UART channel seen from BlueCore™. Output at BlueCore™ PIO(10).

For further details, see the UART Interface States section below.

SPI INTERFACE

The SPI enables firmware updates of the BlueCore™ chip through CSR's DFU-algorithm. The SPI signals are shared with the display (except for the active low control signal).

UART INTERFACE

High-speed full-duplex interface with handshake signals (RTS & CTS).

UART settings used both in stream and command modes:

- 460.8K Baud
- 8 Bit
- No Parity
- One stop bit

UART INTERFACE STATES

STREAM MODE

BlueCore™ is fully transparent although it is controlled by UART in stream mode. It is, therefore, up to the higher firmware-levels running in the ARM7 and the remote node to pack, unpack, and interpret the data.

VM COMMAND MODE

Command mode is initiated by request from either the ARM7 or BlueCore™ chips. All data is packed, unpacked, and interpreted in accordance with the protocol definition.

<u>State transitions</u>
Below is shown how the ARM7 transitions the BlueCore™ chip and the UART interface from stream mode to command mode and back and vice-versa.

** 1. Create stream connection **

INIT:	BC4__CMD low and ARM__CMD low
ARM:	OpenStream
BC4:	Open stream ARM->RADIO
BC4:	Set BC4__CMD high
ARM:	Set ARM__CMD high
BC4:	Open stream RADIO->ARM

** 2. BC4 closes stream **

INIT:	BC4__CMD high and ARM__CMD high
BC4:	Close stream RADIO->ARM
BC4:	Set BC4__CMD low
ARM:	Set ARM__CMD low
BC4:	Close stream ARM->RADIO
BC4:	Send Telegram

** 3. ARM close stream **

INIT:	BC4__CMD high and ARM__CMD high
ARM:	Set ARM__CMD low
BC4:	Close stream RADIO->ARM
BC4:	Close stream ARM->RADIO
BC4:	Set BC4__CMD low
ARM:	Send Telegram

COMMAND MESSAGES BETWEEN BLUECORE^{TM} & ARM7

Known Bluetooth addresses and their user-friendly names are stored in the BlueCore™ chip, enabling fast connections.

Figures 1 and 2 below show message and state diagrams for command messages sent between the BlueCore™ and ARM7 processors. All of the command messages and their respective replies are described in further detail later in this document.

MESSAGE DIAGRAM

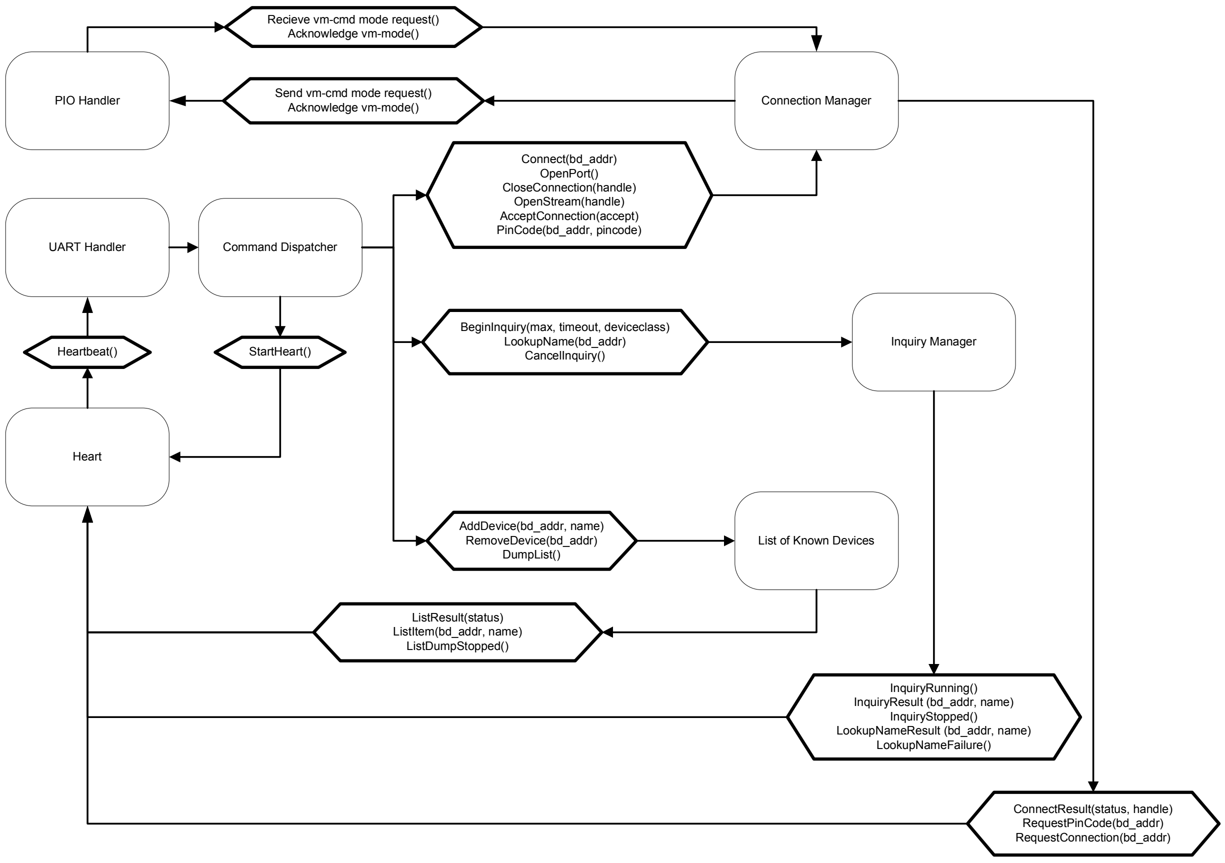

Figure 1: Block diagram for communication between the BlueCore^{TM} and ARM7 processors

BLUECORE^{TM} STATE DIAGRAM

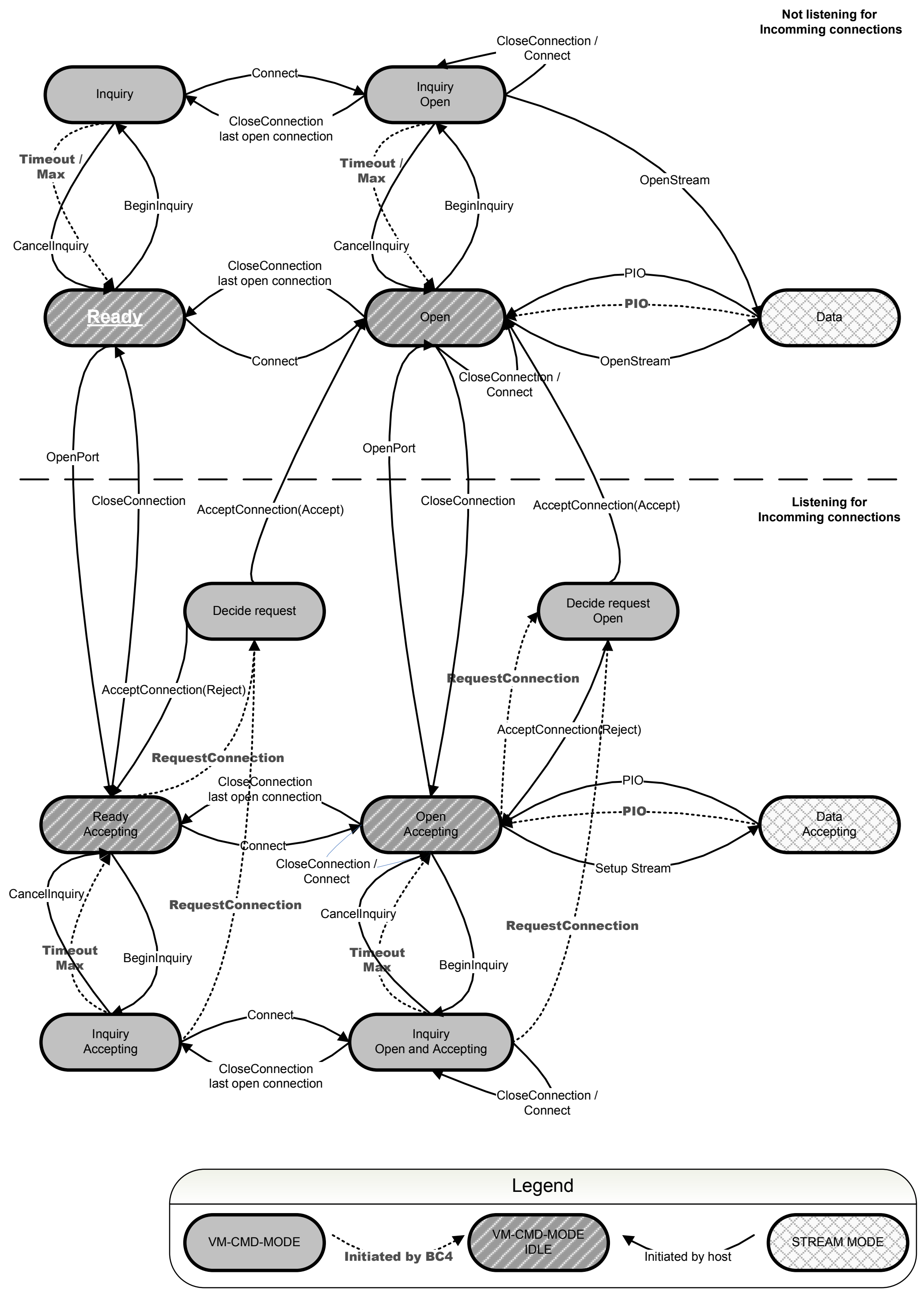

Figure 2: State diagram for communication between the BlueCore^{TM} and ARM7 processors

OPERATING MODES

Two operating modes are defined:

- STREAM__BREAKING__MODE
- DONT__BREAK__STREAM__MODE

The modes specify how incoming events are handled. Events that should be handled by the BlueCore™ chip include:

- connection requests,
- pairing requests, and
- disconnection

The operating mode is set using the SetOperatingMode telegram while the current setting can be read using the GetOperatingMode telegram. The setting is saved in the persistent storage and is auto-applied at start-up. The SetFactorySettings telegram also resets this setting.

STREAM__BREAKING__MODE

In this mode, an open stream will be closed by the BlueCore™ chip in the event of an incoming event.

DONT__BREAK__STREAM__MODE

In this mode, the BlueCore™ chip will not break any streams unless the connection currently streaming is closed.

	STREAM_BREAKING_MODE	**DON'T_BREAK_STREAM_MODE**
Connection request	1) If in stream_mode then break the stream 2) Send a ConnectionRequest telegram	1) If in stream_mode then reject the request 2) Else if remote device in device list then send ConnectionRequest telegram 3) Else reject the request
Pairing request	1) If in stream_mode then ignore the request 2) Else send PinCodeRequest telegram	1) If in stream_mode then ignore the request 2) Else send PinCodeRequest telegram
Disconnection	1) If in stream_mode then break the stream 2) Send CloseConnectionResult telegram	1) If in stream_mode with current stream then close stream and send CloseConnectionResult telegram 2) Else if in stream_mode then wait for cmd_mode and send CloseConnectionResult 3) Else send CloseConnectionResult

COMMAND MESSAGE CODING

MESSAGE STRUCTURE

Encoding and decoding of command mode telegrams is handled at byte level. This is necessary because of the different interpretation of variables longer than 8 bits by the two processors. The two also use different endians: ARM7 uses little endian and BlueCore™ big endian.

MESSAGE WRAPPING

lLengthlMessage typelMessage contentlSUM High bytelSUM Low bytel

Length:	Ubyte, Length of the complete telegram (excluding length)
Message type:	Ubyte, Defined by enumeration of all message types
Message content:	Defined for each message in the entries below
SUM:	Uint16, Negated sum of all previous bytes => Message ID + Message content + SUM = 0

COMMAND MESSAGES (ARM7 => BLUECORE™)

00 BEGININQUIRY

Parameters:	uint8 max__devices, uint16 timeout, uint32 class__of__device
In reply to:	None
Return messages:	InquiryRunning, InquiryResult and InquiryStopped

This command starts the inquiry process. It is acknowledged by an InquiryRunning message from the BlueCore™ chip. The inquiry can be cancelled by sending a CancelInquiry message. The parameters are transferred directly to the corresponding parameters of the BlueLab inquiry command. The following is copied from the BlueLab documentation:
"The time the inquiry is performed for is in fact timeout * 1.28 seconds. The allowed values of timeout are in the range 0x01 to 0x30. This corresponds to an inquiry timeout range of 1.28 to 61.44 seconds."

For every device found, an InquiryResult message is sent to the host. The inquiry will run until max__devices devices have been found or timeout is reached. An InquiryStopped message indicates that the inquiry has ended. Switching to stream mode will cancel inquiry without indication.

Telegram:
| 10 | MSG__BeginInquiry | max__devices | timeout [Hi] | timeout[Lo] | class__of__device[hi] | class__of__device[hi-1] | class__of__device[hi-2] | class__of__device[lo] | SUM[Hi] | SUM[Lo] |

01 CANCELINQUIRY

Parameters:	None
In reply to:	None
Return messages:	InquiryStopped

This command stops the inquiry process. It is acknowledged by an InquiryStopped message from the BlueCore™ chip. It cannot be guaranteed that no InquiryResult messages will be sent between the CancelInquiry and the InquiryStopped message but they will be kept at a minimum. Switching to stream mode will cancel inquiry without indication.

Telegram:
| 3| MSG__CancelInquiry | SUM[Hi] | SUM[Lo] |

02 CONNECT

Parameters:	bdaddr device__address
In reply to:	None
Return messages:	ConnectResult, RequestPinCode

This message indicates to the BlueCore™ chip that the host wants to connect to a remote device. The parameter specifies the Bluetooth device address of the remote device. In reply to this message a ConnectResult message is returned indicating the success or failure of the connect operation. After a successful operation an OpenStream message can be used to switch to stream mode. To close a connection the host can send a CloseConnection message.

Telegram:
| 10| MSG__Connect | bdaddr.lap[hi] | bdaddr.lap[hi-1] | bdaddr.lap[hi-2] | bdaddr.lap[Lo] | bdaddr.uap | bdaddr.nap[Hi] | bdaddr.nap[Lo] | SUM[Hi] | SUM[Lo] |

03 OPENPORT

Parameters	None
In reply to:	None
Return messages:	OpenPortResult

To begin accepting connections from the outside, send this message to the BlueCore™ chip. This command is acknowledged by an OpenPortResult message indicating success or failure. After a successful operation, the BlueCore™ chip can send RequestConnection messages. Sending a ClosePort message to the BlueCore™ chip closes the port.

Telegram:
l 3l MSG_OpenPort l SUM[Hi] l SUM[Lo] l

04 LOOKUPNAME

Parameters:	bdaddr device_address
In reply to:	None
Return messages:	LookupNameFailure and LookupNameResult

Tells the BlueCore™ chip to look up and return the friendly name of a remote device. The result of this command will be returned in a LookupNameResult message or in case of failure, a LookupNameFailure.

Telegram:
l 10l MSG_LookupName l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l SUM[Hi] l SUM[Lo] l

05 ADDDEVICE

Parameters:	bdaddr device_address, char [16] name, uint32 class_of_device
In reply to:	None
Return messages:	ListResult

Adds or updates a device entry in the list of known devices. A ListResult message indicates success or failure of the operation.

Telegram:
l 30 l MSG_AddDevice l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l char [16] name l class_of_device[hi] l class_of_device[hi-1] l class_of_device[hi-2] l class_of_device[lo] l SUM[Hi] l SUM[Lo] l

06 REMOVEDEVICE

Parameters:	bdaddr device_address
In reply to:	None
Return messages:	ListResult

This message can be sent to erase a device from the list of known devices. A ListResult message acknowledges the operation.

Telegram:
l 10l MSG_RemoveDevice l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l SUM[Hi] l SUM[Lo] l

07 DUMPLIST

Parameters:	None
In reply to:	None
Return messages:	ListItem and ListDumpStopped

Send this message to retrieve the list of known devices. The items on the list will be sent one by one in ListItem messages. When the last item has been sent, a ListDumpStopped message is sent.

Telegram:
l 3l MSG__DumpList l SUM[Hi] l SUM[Lo] l

08 CLOSECONNECTION

Parameters:	uint8 handle
In reply to:	None
Return messages:	CloseConnectionResult

This message closes a connection an active connection or an open port. The handle is given by a ConnectResult message sent to the host in reply to Connect messages. The success or failure of the command is returned in a CloseConnectionResult message.

Telegram:
l 4l MSG__CloseConnection l handle l SUM[Hi] l SUM[Lo] l

09 ACCEPTCONNECTION

Parameters:	uint8 accept
In reply to:	RequestConnection
Return messages:	ConnectResult and RequestPinCode

This message is used to indicate whether the BlueCore™ chip should accept an incoming connection. The message should be sent in reply to a RequestConnection message. The accept parameter should be set to 1 if the connection is accepted and 0 if it is not. A ConnectResult or RequestPinCode message will be sent in response to this message.

Telegram:
l 4l MSG__AcceptConnection l accept l SUM[Hi] l SUM[Lo] l

0A PINCODE

Parameters:	bdaddr bd__addr, char [16] pin__code
In reply to:	RequestPinCode
Return messages:	PinCodeAck

This message is used to send a pin code entered by the user on to the BlueCore™ chip. This message should be used in response to a RequestPinCode message. The pin__code parameter is to be null-terminated if the pin__code is shorter than 16 chars. If none of the 16 chars is null, the pin code is assumed to be 16 chars long.

Telegram:
l 26l MSG__PinCode l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l char [16], pin__code l SUM[Hi] l SUM[Lo] l

0B OPENSTREAM

Parameters:	uint8 handle
In reply to:	None
Return messages:	A switch to stream mode is signaled on the PIO-pins

This message will set up a stream to the connection indicated by the handle parameter. After this call, the UART will go into stream mode. The only way to break stream mode is to signal on the PIO-pins. If the call fails, we are in trouble. The host should reset the BlueCore™ chip after a timeout period. Note that this call cancels a running inquiry.

Telegram:
| 4| MSG__OpenStream | handle | SUM[Hi] | SUM[Lo] |

0C STARTHEART

Parameters:	None
In reply to:	None
Return messages:	Heartbeat

This message indicates that the host wants to receive heartbeat signals. The Heartbeat message will be sent in reply and again every time the UART has been idle for X msec.

Telegram:
| 3| MSG__StartHeart | SUM[Hi] | SUM[Lo] |

1C SETDISCOVERABLE

Parameters:	uint8 visible
In reply to:	None
Return messages:	SetDiscoverableAck

This message will enable or disable inquiry scanning. If the visible parameter is set to 1, the BlueCore™ chip will answer incoming inquiries. If visible is set to 0, the BlueCore™ chip will not answer, rendering the BlueCore™ chip invisible to inquiries. This does not affect the ability to accept incoming connections.

Telegram:
| 4| MSG__SetDiscoverable | visible | SUM[Hi] | SUM[Lo] |

1D CLOSEPORT

Parameters:	uint8 handle
In reply to:	None
Return messages:	ClosePortResult

This message will close the port. Until it is removed, the handle should always be 03…

Telegram:
| 4 | MSG__ClosePort | handle | SUM[Hi] | SUM[Lo] |

21 SETFRIENDLYNAME

Parameters:	char [16] name
In reply to:	None
Return messages:	SetFriendlyName Ack

This message is used to set the friendly name of the local device. The name parameter is to be null-terminated if the name is shorter than 16 chars. If none of the 16 chars is null, the name is assumed to be 16 chars long.

Telegram:
l 19 l MSG__SetFriendlyName l char [16] name l SUM[Hi] l SUM[Lo] l

23 GETLINKQUALITY

Parameters:	uint8 handle
In reply to:	None
Return messages:	LinkQualityResult

This message requests a reading of the HCI link quality of a connection.

Telegram:
l 4l MSG__GetLinkQuality l handle l SUM[Hi] l SUM[Lo] l

25 SETFACTORYSETTINGS

Parameters:	None
In reply to:	None
Return messages:	SetFactorySettingsAck

This message is sent to clear the settings in the persistent storage. The BlueCore™ chip should be restarted after calling this function. Otherwise old values can be floating around the BlueCore™ chip causing unexpected behavior.

Telegram:
l 3l MSG__SetFactorySettings l SUM[Hi] l SUM[Lo] l

27 GETLOCALADDR

Parameters:	None
In reply to:	None
Return messages:	GetLocalAddrResult

This message will fetch the local Bluetooth device address.

Telegram:
l 3l MSG__GetLocalAddr l SUM[Hi] l SUM[Lo] l

29 GETFRIENDLYNAME

Parameters:	None
In reply to:	None
Return messages:	GetFriendlyNameResult

This message will fetch the friendly name of the local Bluetooth device.

Telegram:
l 3l MSG__GetFriendlyName l SUM[Hi] l SUM[Lo] l

2A GETDISCOVERABLE

Parameters:	None
In reply to:	None
Return messages:	GetDiscoverableResult

This message will fetch the status of the discoverable local Bluetooth devices.

Telegram:
l 3l MSG__GetDiscoverable l SUM[Hi] l SUM[Lo] l

2B GETPORTOPEN

Parameters:	None
In reply to:	None
Return messages:	GetPortOpenResult

This message will fetch the status of the local Bluetooth device port.

Telegram:
l 3l MSG__GetPortOpen l SUM[Hi] l SUM[Lo] l

2F GETVERSION

Parameters:	None
In reply to:	None
Return messages:	GetVersionResult

This message will fetch the version of the BlueCore™ code.

Telegram:
l 3l MSG__GetVersionOpen l SUM[Hi] l SUM[Lo] l

33 GETBRICKSTATUSBYTE

Parameters:	None
In reply to:	None
Return messages:	GetBrickStatusbyteResult

This message will fetch the status bytes from persistent storage.

Telegram:
l 3l MSG__GetBrickStatusbyte l SUM[Hi] l SUM[Lo] l

34 SETBRICKSTATUSBYTE

Parameters:	uint8 byte1, uint8 byte2
In reply to:	None
Return messages:	SetBrickStatusbyteResult

This message set the status bytes in the persistent storage.

Telegram:
l 5l MSG__SetBrickStatusbyte l byte1 l byte2 l SUM[Hi] l SUM[Lo] l

35 GETOPERATINGMODE

Parameters:	None
In reply to:	None
Return messages:	OperatingModeResult

This message gets the operating mode of the brick. See “36 SetOperatingMode” for a description of the modes.

Telegram:
l 3 l MSG__GetOperatingMode l SUM[Hi] l SUM[Lo] l

36 SETOPERATINGMODE

Parameters:	uint8 mode
In reply to:	None
Return messages:	OperatingModeResult

This message sets the operating mode of the brick. The mode should be one of:

```
typedef enum {
        STREAM__BREAKING__MODE,
        DONT__BREAK__STREAM__MODE
} OperatingMode;
```

Telegram:
l 4l MSG__SetOperatingMode l mode l SUM[Hi] l SUM[Lo] l

38 GETCONNECTIONSTATUS

Parameters:	None
In reply to:	None
Return messages:	ConnectionStatusResult

This message gets the connection status of the brick.

Telegram:
l 3 l MSG__GetConnectionStatus l SUM[Hi] l SUM[Lo] l

3A GOTODFUMODE

Parameters:	None
In reply to:	None
Return messages:	None

This message will cause a warm reboot into DFU boot mode.

Telegram:
l 3 l MSG__GotoDFUMode l SUM[Hi] l SUM[Lo] l

RESULT MESSAGES (BLUECORE^{TM} => ARM7)

0D HEARTBEAT

Parameters:	None
In reply to:	StartHeart, and a BlueCore™ initiated mode shift to command mode
Return messages:	None

After a StartHeart message is sent, this message is sent periodically. See the description of the StartHeart message for details.

Telegram:
l 3l MSG__Heartbeat l SUM[Hi] l SUM[Lo] l

0E INQUIRYRUNNING

Parameters:	None
In reply to:	BeginInquiry
Return messages:	None

Sent as acknowledgement of a BeginInquiry message. This message will be followed by zero or more InquiryResult messages and lastly by an InquiryStopped message.

Telegram:
l 3l MSG__InquiryRunning l SUM[Hi] l SUM[Lo] l

0F INQUIRYRESULT

Parameters:	bdaddr device__address, char [16] name, uint32 class__of__device
In reply to:	BeginInquiry
Return messages:	None

For each device found in an inquiry, this message is sent from the BlueCore™ to the host. The message contains the device address of the device and the friendly name. If the friendly name is less than 16 chars long, the string is null-terminated; otherwise it is assumed to be 16 chars long. The class__of__device parameter contains the device class identifier as specified in the Bluetooth specifications.

Telegram:
l 30 l MSG__InquiryResult l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l char [16], name l class__of__device[hi] l class__of__device[hi-1] l class__of__device[hi-2] l class__of__device[lo] l SUM[Hi] l SUM[Lo] l

10 INQUIRYSTOPPED

Parameters:	None
In reply to:	BeginInquiry
Return messages:	None

This message indicates that an inquiry has ended. This may be because a timeout or that the maximum number of found devices has been reached. An inquiry will also end if the UART enters stream mode but this will not generate an InquiryStopped message.

Telegram:
l 3l MSG__InquiryStopped l SUM[Hi] l SUM[Lo] l

11 LOOKUPNAMERESULT

Parameters:	bdaddr device__address, char [16] name, uint32 class__of__device
In reply to:	LookupName
Return messages:	None

This message is sent in response to a LookupName command message. The message contains the device address and the friendly name of the device. If the friendly name is less than 16 chars long, the string is null-terminated; otherwise it is assumed to be 16 chars long.

The class of device is read from the device list. If the device is not on the list, zero is returned.

Telegram:
l 30l MSG__LookupNameResult l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l char [16], name l class__of__device[hi] l class__of__device[hi-1] l class__of__device[hi-2] l class__of__device[lo] l SUM[Hi] l SUM[Lo] l

12 LOOKUPNAMEFAILURE

Parameters:	bdaddr device__address
In reply to:	LookupName
Return messages:	None

This message is sent as response to a LookupName command message in case of failure.

Telegram:
l 10 l MSG__LookupNameFailure l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l SUM[Hi] l SUM[Lo] l

13 CONNECTRESULT

Parameters:	uint8 status, uint8 handle
In reply to:	Connect and OpenPort
Return messages:	None

This message is sent in response to Connect and OpenPort messages. The status parameter is 1 if the Connect or OpenPort operation was a success and 0 if it is not.

Telegram:
l 5l MSG__ConnectResult l status l handlel SUM[Hi] l SUM[Lo] l

14 RESETINDICATION

Parameters:	None
In reply to:	None
Return messages:	None

This message is sent to the host when the BlueCore™ chip is finished with its initialization.

Telegram:
l 3l MSG__ResetIndication l SUM[Hi] l SUM[Lo] l

15 REQUESTPINCODE

Parameters:	bdaddr device__address
In reply to:	None
Return messages:	PinCode

This message is sent if a remote device is requesting a pin code. The host should prompt the user for a pin code and return it in a PinCode message. The device__address parameter contains the device address of the remote device.

Telegram:
| 10| MSG__RequestPinCode | bdaddr.lap[hi] | bdaddr.lap[hi-1] | bdaddr.lap[hi-2] | bdaddr.lap[Lo] | bdaddr.uap | bdaddr.nap[Hi] | bdaddr.nap[Lo] | SUM[Hi] | SUM[Lo] |

16 REQUESTCONNECTION

Parameters:	bdaddr device__address
In reply to:	None
Return messages:	AcceptConnection

This message is sent to the host if a remote device wants to connect to the BlueCore™ chip. The host should respond by sending an AcceptConnection message indicating whether or not the connection should be accepted.

Telegram:
| 10| MSG__RequestConnection | bdaddr.lap[hi] | bdaddr.lap[hi-1] | bdaddr.lap[hi-2] | bdaddr.lap[Lo] | bdaddr.uap | bdaddr.nap[Hi] | bdaddr.nap[Lo] | SUM[Hi] | SUM[Lo] |

17 LISTRESULT

Parameters:	uint8 status
In reply to:	AddDevice or RemoveDevice
Return messages:	None

The status parameter is given by the following enum:

```
enum {
        LR__SUCCESS = 0x50,
        LR__COULD__NOT__SAVE,
        LR__STORE__IS__FULL,
        LR__ENTRY__REMOVED,
        LR__UNKNOWN__ADDR
};
```

LR__SUCCESS: Indicates that the operation was successful.
LR__COULD__NOT__SAVE: The entry could not be written to persistent storage. The BlueCore™ chip must be reset to activate defragmentation of the flash.
LR__STORE__IS__FULL: There are no empty slots in the list to save an entry to. Use RemoveDevice to create an open slot.
LR__ENTRY__REMOVED: The entry was successfully removed.
LR__UNKNOWN__ADDR: The list does not contain an entry with the provided Bluetooth device address.

Telegram:
| 4| MSG__ListResult | status | SUM[Hi] | SUM[Lo] |

18 LISTITEM

Parameters:	bdaddr device__address, char [16] name, uint32 class__of__device
In reply to:	DumpList
Return messages:	None

This message is sent from the BlueCore™ chip to the host for each device found on the list of known devices. The message contains the device address of the device and the friendly name. If the friendly name is less than 16 chars long the string is null-terminated; otherwise it is assumed to be 16 chars long.

Telegram:
l 26l MSG__ListItem l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l char [16] name l class__of__device[hi] l class__of__device[hi-1] l class__of__device[hi-2] l class__of__device[lo] l SUM[Hi] l SUM[Lo] l

19 LISTDUMPSTOPPED

Parameters:	None
In reply to:	DumpList
Return messages:	None

This message indicates that the list dump was completed.

Telegram:
l 3l MSG__ListDumpStopped l SUM[Hi] l SUM[Lo] l

1A CLOSECONNECTIONRESULT

Parameters:	uint8 status, uint8 handle
In reply to:	CloseConnection
Return messages:	None

This message is sent in response to CloseConnection messages. The status parameter is given by the following enum:

```
typedef enum
{
        /*! Successful disconnection.*/
        spp__disconnect__success,
        /*! Unsuccessful due to the link being lost.*/
        spp__disconnect__link__loss,
        /*! Unsuccessful due to no service level connection.*/
        spp__disconnect__no__slc,
        /*! Unsuccessful due to time out.*/
        spp__disconnect__timeout,
        /*! Unsuccessful for some other reason.*/
        spp__disconnect__error
} spp__disconnect__status;
```

Telegram:
l 5l MSG__CloseConnectionResult l status l handle l SUM[Hi] l SUM[Lo] l

1B PORTOPENRESULT

Parameters:	uint8 status, uint8 handle, uint8 ps__success
In reply to:	OpenPort
Return messages:	None

This message is the result of a PortOpen command. It will contain the status 1 if successful and 0 otherwise. Ps__Success is 0 if the port-open could not be written to persistent storage and 1 if it could. The port will be opened regardless of the persistent storage.

Telegram:
l 6 l MSG__PortOpenResult l status l handle l ps__success l SUM[Hi] l SUM[Lo] l

1E CLOSEPORTRESULT

Parameters:	uint8 status, uint8 handle, uint8 ps__sucess
In reply to:	ClosePort
Return messages:	None

This message is the result of an CloseOpen command. It will contain the status 1 if successful and 0 otherwise. Ps__Success is 0 if the port-close could not be written to persistent storage and 1 if it could. The port will be closed regardless of the persistent storage.

Telegram:
l 6 l MSG__ClosePortResult l status l handle l ps__success l SUM[Hi] l SUM[Lo] l

1F PINCODEACK

Parameters:	None
In reply to:	PinCode
Return messages:	None

This message is sent after a PinCode message is received. See the description of the PinCode telegram for details.

Telegram:
l 3 l MSG__PinCodeAck l SUM[Hi] l SUM[Lo] l

20 SETDISCOVERABLEACK

Parameters:	uint8 success
In reply to:	SetDiscoverable
Return messages:	None

This message is sent after a SetDiscoverable message is received. See the description of the SetDiscoverable telegram for details. Success is 0 if the name could not be written to persistent storage and 1 if it could. The discoverability will be changed regardless of the persistent storage.

Telegram:
l 3l MSG__SetDiscoverableAck l success l SUM[Hi] l SUM[Lo] l

22 SETFRIENDLYNAMEACK

Parameters:	uint8 success
In reply to:	SetFriendlyName
Return messages:	None

This message is sent after a SetFriendlyName message is received. See the description of the SetFriendlyName telegram for details. Success is 0 if the name could not be written to persistent storage and 1 if it could. The name will be changed regardless of the persistent storage.

Telegram:
l 4 l MSG__ SetFriendlyNameAck l success l SUM[Hi] l SUM[Lo] l

24 LINKQUALITYRESULT

Parameters:	uint8 quality
In reply to:	GetLinkQuality
Return messages:	None

This message contains the result of a GetLinkQuality message. The quality is an octet value ranging from 0x00 to 0xFF. If the value is high, the link quality is better.

Telegram:
l 4l MSG__GetLinkQuality l quality l SUM[Hi] l SUM[Lo] l

26 SETFACTORYSETTINGSACK

Parameters:	None
In reply to:	SetFactorySettings
Return messages:	None

This message is sent when the settings in the persistent storage have been cleared.

Telegram:
l 3l MSG__SetFactorySettingsAck l SUM[Hi] l SUM[Lo] l

28 GETLOCALADDRRESULT

Parameters:	bdaddr addr
In reply to:	GetLocalAddr
Return messages:	None

This message returns the local Bluetooth device address.

Telegram:
l 10 l MSG__GetLocalAddrResult l bdaddr.lap[hi] l bdaddr.lap[hi-1] l bdaddr.lap[hi-2] l bdaddr.lap[Lo] l bdaddr.uap l bdaddr.nap[Hi] l bdaddr.nap[Lo] l SUM[Hi] l SUM[Lo] l

2C GETFRIENDLYNAMERESULT

Parameters:	char name[16]
In reply to:	GetFriendlyName
Return messages:	None

This message returns the friendly name of the local Bluetooth device. If the name is shorter than 16 chars the name will be zero-padded.

Telegram:
l 19 l MSG__GetFriendlyNameResult l char [16] l SUM[Hi] l SUM[Lo] l

2D GETDISCOVERABLERESULT

Parameters:	uint8 discoverable
In reply to:	GetDiscoverable
Return messages:	None

The discoverable parameter will be 1 if the device is discoverable and 0 otherwise.

Telegram:
l 4l MSG__GetDiscoverableResult l discoverable l SUM[Hi] l SUM[Lo] l

2E GETPORTOPENRESULT

Parameters:	uint8 portIsOpen
In reply to:	GetDiscoverable
Return messages:	None

The portIsOpen parameter will be 1 if the port is open and 0 otherwise.

Telegram:
l 4l MSG__GetPortOpenResult l portIsOpen l SUM[Hi] l SUM[Lo] l

30 GETVERSIONRESULT

Parameters:	uint8 major, uint8 minor
In reply to:	GetVersion
Return messages:	None

This message contains the version number of the firmware implemented within the BlueCore™ chip.

Telegram:
l 5l MSG__GetVersionResult l major l minor l SUM[Hi] l SUM[Lo] l

31 GETBRICKSTATUSBYTERESULT

Parameters:	uint8 byte1, uint8 byte2
In reply to:	GetBrickStatusbyte
Return messages:	None

This message contains the status bytes from persistent storage.

Telegram:
l 5l MSG__GetBrickStatusbyteResult l byte1 l byte2 l SUM[Hi] l SUM[Lo] l

32 SETBRICKSTATUSBYTERESULT

Parameters:	uint8 success
In reply to:	SetBrickStatusbyte
Return messages:	None

The success parameter is given by the following enum, also used in ListResult:

```
enum
{
        LR_SUCCESS = 0x50,
        LR_COULD_NOT_SAVE
};
```

LR_SUCCESS: Indicates that the operation was successful.
LR_COULD_NOT_SAVE: The entry could not be written to persistent storage. The BlueCore™ chip must be reset to activate defragmentation of the flash.

Telegram:
| 3| MSG_SetBrickStatusbyteResult | success | SUM[Hi] | SUM[Lo] |

37 OPERATINGMODERESULT

Parameters:	uint8 mode
In reply to:	SetOperatingMode, GetOperatingMode
Return messages:	None

This message indicates the operating mode of the brick.

Telegram:
| 4| MSG_OperatingModeResult | mode | SUM[Hi] | SUM[Lo] |

39 CONNECTIONSTATUSRESULT

Parameters: uint8 status_handle0, uint8 status_handle1, uint8 status_handle2, uint8 status_handle3
In reply to: GetConnectionStatus
Return messages: None

This message indicates the status of the connection. Each status byte will contain a value in the enumeration:

```
typedef enum {
        CONN_READY,
        CONN_INITIALIZED,
        CONN_CONNECTED,
        CONN_CONNECTING,
        CONN_STREAM_OPEN
} ConnState;
```

Telegram:
| 10 | MSG_ConnectionStatusResult |3 x RESERVED | h0 | h1 | h2 | h3 | SUM[Hi] | SUM[Lo] |

C-CODE STANDARD FOR MESSAGE ID

This enumeration defines the message id numbering. The first message type has id 0.

ENUMERATION

```
enum MSG__TYPES
{
        MSG__BeginInquiry,
        MSG__CancelInquiry,
        MSG__Connect,
        MSG__OpenPort,
        MSG__LookupName,
        MSG__AddDevice,
        MSG__RemoveDevice,
        MSG__DumpList,
        MSG__CloseConnection,
        MSG__AcceptConnection,
        MSG__PinCode,
        MSG__OpenStream,
        MSG__StartHeart,
        MSG__Heartbeat,
        MSG__InquiryRunning,
        MSG__InquiryResult,
        MSG__InquiryStopped,
        MSG__LookupNameResult,
        MSG__LookupNameFailure,
        MSG__ConnectResult,
        MSG__ResetIndication,
        MSG__RequestPinCode,
        MSG__RequestConnection,
        MSG__ListResult,
        MSG__ListItem,
        MSG__ListDumpStopped,
        MSG__CloseConnectionResult,
        MSG__PortOpenResult,
        MSG__SetDiscoverable,
        MSG__ClosePort,
        MSG__ClosePortResult,
        MSG__PinCodeAck,
        MSG__SetDiscoverableAck,
        MSG__SetFriendlyName,
        MSG__SetFriendlyNameAck,
        MSG__GetLinkQuality,
        MSG__LinkQualityResult,
        MSG__SetFactorySettings,
        MSG__SetFactorySettingsAck,
        MSG__GetLocalAddr,
        MSG__GetLocalAddrResult,
        MSG__GetFriendlyName,
        MSG__GetDiscoverable,
        MSG__GetPortOpen,
        MSG__GetFriendlyNameResult,
        MSG__GetDiscoverableResult,
        MSG__GetPortOpenResult,
        MSG__GetVersion,
        MSG__GetVersionResult,
        MSG__GetBrickStatusbyteResult,
        MSG__SetBrickStatusbyteResult,
        MSG__GetBrickStatusbyte,
        MSG__SetBrickStatusbyte,
        MSG__GetOperatingMode,
        MSG__SetOperatingMode,
        MSG__SetOperatingModeResult,
        MSG__GetConnectionStatus,
        MSG__ConnectionStatusResult,
        MSG__gotoDFUMode
};
```