Category: **protocol description:**

Description of the BT3/6 or BT12 Bluetooth communication protocol

Project Name: **BT3/6 - BT12**

Company:

CORSCIENCE GmbH & Co. KG
Henkestraße 91
91052 Erlangen
GERMANY
www.corscience.de

Phone +49 9131 977986-0
Fax +49 9131 977986-58

General information: info@corscience.de
Technical questions: service@corscience.de

Written by: S. Rennecke **Date:** 10.02.09 **Version:** 1.7		**Checked by:** H.Troff (HTR) **Date**: 17.02.2009	
Index	Change	Date:	Name/Abbreviation
1.1	Documentation of 5.2.1, 5.2.2, and 5.2.6 revised.	21.11.2007	SRE
1.2	Command numbers in titles included, References in section 5.2.11, 0, and 5.2.14, Command number for SWITCH_BT_ROLE_CFM corrected.	11.04.2008	SRE
1.3	4.1.1 buffer Byte not implemented, 4.1.4 serial no. changed to 5 Bytes, 4.1.5 Operation cycles deleted, 4.1.6 length deleted, derivate deleted, description changed, firmware changed, version optional, examples added for general commands,5.2.5 text added, 5.2.5.1 calculation added.,5.2.5 description added	28.04.2008	KSE
1.4	The protocol version has been increased from 2 to 3 (section 4.2.1). Description of the command “START_STOP_ECG_TRANSMISSION” revised. Description of the command “START_STOP_ECG_TRANSMISSION_ADVANCED” added (section 5.2.5). Description of the command “ECG_DATA_TRANSMISSION_ADVANCED” added (section 5.2.7). Description for the field “packet type” in the Monitor Byte 2 added(section 5.2.7.1)	09.09.2008	SRE
1.5	Payload field “Signal Detection” added to the command ECG_DATA_TRANSMISSION_ADVANCED see section 5.2.7 Description of the signal detection Bytes in section 5.2.7.3 added.	13.11.2008	SRE
1.6	Some explanation for the signal detection bytes in section 5.2.7.3 added.	28.11.2008	SRE
1.7	Overview of commands completed (4.1 and 5.1); explanations of the commands at the sections 4.2.5, 4.2.6, 5.2.3, 5.2.6, and 0 revised; list of examples for the request command completed.	10.02.2009	SRE

TABLE OF CONTENTS

1 Brief document description

The protocol is designed for simple and memory-saving implementation in a microcontroller. The overhead was kept as low as possible. The principle of this protocol is basically modelled on PPP (point-to-point protocol), which is often used to establish modem connections. Escape sequences are used to filter out reserved bytes (Start flag, end flag, escape flag) from the data stream (so-called octet stuffing). The octet stuffing in the transmitter occurs after the checksum calculation. For the checksum formation in the transmitter, the start flag and end flag are not included, of course. In the receiver, inverse octet stuffing is applied first, before the data stream is saved or further processed. In the receiver, the checksum algorithm is run over the entire packet (not including the start flag and end flag). The result must then be 0.

2 Packet structure

The packet is structured according to the table below.

Start flag	Packet number	Command	Payload	Checksum	End flag
0xFC	1 byte	2 byte	x bytes	2 bytes	0xFD

Example: Command Protocol

Start	Packet No.	Command Protocol	Protocol Version	Max. Payload length	Checksum	End
FC	01	00 01	02	DC 00	2E 03	FD

2.1 Description of individual fields

The individual fields of the above overview are described in more detail below.

2.1.1 Start flag

The start flag, which has a length of one byte and the fixed value **0xFC** is the unmistakable start character of a packet. This character may not appear again in the rest of the packet to avoid confusion. To prevent this from happening, octet stuffing (see 3) is used. It is necessary to send the start flag. If not, the packet will not be confirmed.

2.1.2 Packet number

The packet number field has a length of one byte and can therefore sweep through numbers from 0 to 255. The packets are consecutively numbered, separately from both sides. Once 255 have been reached, the next packet starts again with 0. The numbers are given separately from both sides and must not be the same. The packets are numbered so that missing packets can be detected, received packets can be confirmed, undesired packets can be rejected and defective packets be resent.

2.1.3 Command

The command field is two bytes long and contains the command. (LSB first)

2.1.4 Payload

The actual data have to be transmitted. This field can contain other fields, depending on the command. The fields have a defined function within the command.

2.1.5 Checksum

The checksum has a length of two bytes. The parameters of the checksum are:

CRC16 (CCITT), polynomial 0x1021, start value 0xFFFF, LSB first.

The checksum is calculated in the transmitter over the entire packet with the exception of the **start flag (FC)**, **end flag (FD)** and **checksum**. The checksum must be calculated in the transmitter before octet stuffing.

In the **receiver**, the checksum is calculated over the entire packet with the exception of the **start-flag** and **end flag**. For a valid packet, the result must be **0**. The checksum test must be carried out after octet stuffing.

Example:

Command protocol 0x0100

CRC is calculated over **01 00 01 02 DC 00 (hex)** → CRC = 0x032E

Append the CRC LSB first: **2E 03 (hex)**

2.1.6 End flag

The end flag has a length of one byte and has the fixed value of **0xFD**, which is the unmistakable end character of a packet. To avoid confusion, this character may not appear again in the rest of the packet. To prevent this from happening, so-called octet stuffing is applied. It is necessary to send the end flag. If not, the packet will not be confirmed.

3 Octet stuffing

There are three special characters which may not appear by accident in the packet:

- Start flag 0xFC
- End flag 0xFD
- Escape flag 0xFE

These three characters must be filtered out of the data stream before sending the packet and be treated specially. If one of these characters appears in the data stream, an escape flag (FE) is sent first, and then the original byte is linked with 0x20 EXOR.

Characters within the packet:	**Are changed to the following when sent:**
0xFC	0xFE 0xDC
0xFD	0xFE 0xDD
0xFE	0xFE 0xDE

When an escape sequence is received by the receiver, this byte isn't saved, but the following byte is linked with 0x20 EXOR and saved.

The receiver decodes the escape sequence again:

Received characters:	**Decoded:**
0xFE 0xDC	0xFC
0xFE 0xDD	0xFD
0xFE 0xDE	0xFE

4 Description of general commands

4.1 Overview of the commands

Command	Number	Direction	Description
Protocol	0x0100	→	The protocol version and length of receive buffer
Firmware version	0x0150	→	The firmware version of the ECG device.
NACK	0x0300	→	Not acknowledge the last received packed
Reject	0x0400	→	Rejected packet given
Identification	0x0500	→	Identification data given
Maintenance	0x0600	→	Maintenance data given
Transmit data	0x07XX	→	Command sent by ECG device
Request	0x0800	←	Command requested from ECG device
Receive data	0x09XX	←	Command received by ECG device

4.2 Command description

The commands are described in more detail below, and the bytes in the payload are defined.

4.2.1 Command: Protocol (0x0100)

Command	Protocol version	Max. payload length (bytes)	Max no. of packets, which can be sent buffered by transmitter (not yet implemented in SW)
0x0100	1 byte	2 bytes	1 byte
0x0100	0x03	220	Not transmitted

Currently, the protocol version 0x03 is implemented. A command which is sent to the module must not exceed the maximum length 220 bytes. The protocol version is demanded with the command “Request”(0x0800). For more information see “Command: Request” below.

Example:

FC 00 00 01 02 DC 00 8E 46 FD

4.2.2 Command: NACK (0x0300)

Command	Packet number
0x0300	1 byte
0x0300	the packet number of the last received Packet

The reason for a NACK is a received packed with a wrong CRC value.

4.2.3 Command: Reject (0x0400)

Command	Packet number
0x0400	1 byte
0x0400	the packet number of the last received packet

The reason for a Reject can be:

- wrong number of arguments for the received command
- unknown command received

4.2.4 Command: Identification (0x0500)

Command	Manufacturer ID	Device ID (device type)	Serial number
0x0500	1 byte	1 byte	5 bytes
0x0500	0x01 = Corscience	0x05 = BT3/6 – BT12	

The identification data are demanded with the command Request. For more information, see "Command: Request" below.

Example:

FC 01 00 05 01 05 31 30 30 37 30 00 D8 FD

A Bluetooth device produced by Corscience serves at least one RFCOM Services. You can identify the type of the device by reading out the Service Record. So it isn't necessary to open a RFCOM connection to identify the device. We use the attribute 0x0101 (Service Description) to store our ManufacturerID and the DeviceID as one 16Bit Value (0x0501). The service record sequence with this attribute is shown below:

```
…0x09, /* Attribute: ServiceDescription(0x0101) */
        0x01,
        0x01,
        0x09, /* 16Bit Value is following (LSB first)*/
              0x01, /* ManufacturerID (0x01) */
              0x05, /* DeviceID (0x05) */ …
```

For more information of the Service Discovery Protocol (SDP) read the Bluetooth specification.

4.2.5 Command: Maintenance (0x0600)

Command	Maintenance-Data
0x0600	2 bytes
0x0600	Selftest Status (Lsb first)

After turning on the ECG device, the integrated self test is started (max. 5sec). The self test will not be executed again. The result of the self test is transmitted with this command. It can be requested with the Command: Request (0x0800). For more information, see section 4.2.7 below.

The bits in the self test status bytes are shown below. The red zero marked bits are reserved.

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
0	0	0	0	0	0	0	Buzzer	RAM	int_FLASH	PLD/ADCs	BT	Power_ADC	PACER_ADC	ext Flash	LCD
MSB = 0x01								LSB = 0xFF							

Fig. 1 Self test status bytes

Example:

FC 02 00 06 FF 01 F1 F4 FD

If the corresponding bits in the self test status bytes are one, the self test was successful for the corresponding unit. If the self test fails, no communication to the ECG device is possible.

4.2.6 Command: Firmware version (0x0150)

Command	Firmware	Version	Development version (optional)
0x0150	7 bytes	1 byte	2 bytes
0x0150	e.g.: “CS10021”	e.g.: “A”, “B”, “C”…	e.g.: “01”, “02”…

Example:
Working on release of the version “CS10021F” in the fifth version of development, the firmware string is “CS10021E05”
If the firmware version - F is released the firmware version is “CS10021F”

Example:
FC 03 50 01 43 53 31 30 30 32 31 46 A2 52 FD

This command can be requested with the Command: Request (0x0800). For more information, see section 4.2.7.

4.2.7 Command: Request (0x0800)

Command	Command request (LSB First)
0x0800	2 byte

The Request command serves to inquiry the protocol version, identification data, maintenance data, Bluetooth Clock…

Example: Protocol version inquiry

Start flag	Packet number	Command	Payload – Protocol command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x00 0x01	0xDD 0x02	0xFD

Example: Firmware version inquiry

Start flag	Packet number	Command	Payload – Firmware version command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x50 0x01	0x62 0x0C	0xFD

Example: Identity inquiry

Start flag	Packet number	Command	Payload – Identification command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x00 0x05	0x59 0x42	0xFD

Example: Maintenance data inquiry

Start flag	Packet number	Command	Payload – Maintenance command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x00 0x06	0x3A 0x72	0xFD

Example: Bluetooth Clock inquiry

Start flag	Packet number	Command	Payload – READ_BD_CLOCK_CFM command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x05 0x07	0xEE 0x9D	0xFD

Example: ECG device configuration inquiry

Start flag	Packet number	Command	Payload – CONFIG_ECG_DEVICE_CFM command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x10 0x07	0x68 0x61	0xFD

Example: Medical Parameter configuration inquiry

Start flag	Packet number	Command	Payload – SET_MEDICAL_PARMETER_CFM command	Checksum	End flag
0xFC	0x01	0x00 0x08	0x16 0x07	0xCE 0xCB	0xFD

The responses to queries can be found under the commands: Protocol (section 4.2.1), Firmware version (section 4.2.6), Identification (section 4.2.4) Maintenance (section 4.2.5), READ_BD_CLOCK_CFM (section 5.2.3), CONFIG_ECG_DEVICE_CFM (section 5.2.11), and SET_MEDICAL_PARMETER_CFM (section 5.2.17) commands.

4.2.8 Command: Transmit Data (0x07XX)

Command	Data
0x07XX	x bytes

The commands from the ECG device are contained in the payload of a “Transmit Data” packet (0x07xx).

4.2.9 Command: Receive Data (0x09XX)

Command	Data
0x09XX	x bytes

The commands to the ECG device are contained in the payload of a “Receive Data” packet (0x09xx).

5 Description of ECG device specific commands

5.1 Overview of the commands

CONFIG_ANALOG_REQ	0x0901	ADC configuration
CONFIG_ANALOG_CFM	0x0701	Confirms ADC configuration
READ_BD_CLOCK_CFM	0x0705	Confirms the BT-Clock of the device
START_STOP_ECG_TRANSMISSION	0x0905	Starts and stops ECG transmission
ECG_DATA_TRANSMISSION	0x0724	Transmits ECG data
START_STOP_ECG_TRANSMISSION_ADVANCED	0x0927	Starts and stops ECG transmission with advanced ECG data packets
ECG_DATA_TRANSMISSION_ADVANCED	0x0727	Transmits ECG data in advanced data packets.
SWITCH_BT_ROLE_REQ	0x0907	Request an BT role switch
SWITCH_BT_ROLE_CFM	0x0707	Confirm the role switch request
CONFIG_ECG_DEVICE_REQ	0x0910	Device configuration
CONFIG_ECG_DEVICE_CFM	0x0710	Confirms device configuration
TEST_BEEPER_VOLUME_REQ	0x0913	Tests beeper volume
TEST_BEEPER_VOLUME_CFM	0x0713	Confirms audio output
FLASH_DATA_REQ	0x0915	Read data on flash
FLASH_EMPTY_CFM	0x0715	Flash content transmitted
SET_MEDICAL_PARAMETER_REQ	0x0916	Sets thresholds for heart rate, configures pace maker recognition
SET_MEDICAL_PARAMETER_CFM	0x0716	Confirms 0x916

5.2 Command descriptions

5.2.1 CONFIG_ANALOG_REQ (0x0901)

Command to configure the sampling rate and the active ECG channels.

Command	Channel	Sampling rate
0x0901	1 Byte	1 Byte
0x0901	0x01 – Transmit channel II and III (for 3- and 6-lead ECGs) 0x02 – Transmits II, III, V1, V2, V3, V4, V5, V6 (for 12-lead EKG)	0x01 – 100 Hz 0x05 – 500 Hz

5.2.2 CONFIG_ANALOG_CFM (0x0701)

This command confirms the ADC configuration.

Command	Channel	Sampling rate
0x0701	1 Byte	1 Byte
0x0701	0x01 – Transmit channel II and III (for 3- and 6-lead ECGs) 0x02 – Transmits II, III, V1, V2, V3, V4, V5, V6 (for 12-lead EKG)	0x01 – 100 Hz 0x05 – 500 Hz

5.2.3 READ_BD_CLOCK_CFM (0x0705)

This command confirms the Bluetooth Clock of the ECG device.

Command	Bluetooth Clock
0x0705	4 Byte
0x0705	Bluetooth Clock of the ECG device, LSB first.

This command can be requested with the Command: Request (0x0800). For more information, see section 4.2.7.

5.2.4 START_STOP_ECG_TRANSMISSION (0x0905)

Command	Start or Stop
0x0905	1 Byte
	0x00 – Stop 0x01 - Start

To start the transmission a “CONFIG_ANALOG_REQ (5.2.1)” command has to be sent first. After starting the transmission ECG data packets (0x0724) are sent, as described in section 5.2.6. During an ECG data transmission, the connection could be lost. In this case the device is able to store packets couldn’t be sent.

To receive missing packets, you can use the command FLASH_DATA_REQ (5.2.14) after the transmission is stopped (command 0x0905 with parameter 0x01).

During the ECG data transmission, only the commands “START_STOP_ECG_TRANSMISSION” with the parameter “Stop” and “Command: Request with the Parameter READ_BD_CLOCK_CFM” (section 4.2.7) are allowed. Other commands will be ignored.

5.2.5 START_STOP_ECG_TRANSMISSION_ADVANCED (0x0927)

Command	Start or Stop
0x0927	1 Byte
	0x00 – Stop 0x01 - Start

To start the ecg data transmission a “CONFIG_ANALOG_REQ (5.2.1)” command has to be sent first. After starting the transmission advanced ECG data packets (0x0727) are sent, as described in section 5.2.7. During an ECG data transmission, the connection could be lost. In this case the device is able to store packets couldn’t be sent.

To receive missing packets, you can use the command FLASH_DATA_REQ (5.2.14) after the transmission is stopped (command 0x0905 with parameter 0x01).

During the ECG data transmission, only the commands “START_STOP_ECG_TRANSMISSION_ADVANCED” with the parameter “Stop” and “Command: Request with the Parameter READ_BD_CLOCK_CFM” (section 4.2.7) are allowed. Other commands will be ignored.

5.2.6 ECG_DATA_TRANSMISSION (0x0724)

Start Flag	Packet-No. Bit 0-7	Com mand	Packet-No. Bit 8-14	Packet- No. Bit 15-21	Pulse	Monitor Byte1	Monitor Byte2	Pay-load	Check sum	End Flag
0xFC	1 Byte	0x0724	1Byte	1Byte	1Byte	1Byte	1Byte	x Bytes	2 Bytes	0xFD

Packet No: The ECG data packet (0x0724) contains a 22 bit packet number which is split into 3 Bytes. The first packet number byte contains Bit 0-7, the second byte contains Bit 8-14 and the last byte contains Bit 15-21 of the packet number. In both of the higher two bytes the highest bit is set to zero. For transmitting battery, heart rate and electrode contact information “Monitor Byte 1” and “Monitor Byte 2” are used. The payload field includes compressed ECG data.

For the description of the fields **Pulse, Monitor Bytes1/2**, and **Payload** see the next section (section 5.2.7). These fields are the same in both ECG data transmission packet types.

5.2.7 ECG_DATA_TRANSMISSION_ADVANCED (0x0727)

Start Flag	Packet No.	Com mand	Time Stamp Bit 6-0	Time Stamp Bit 13-7	Time Stamp Bit 20-14	Time Stamp Bit 27-21	Pulse	Monitor Byte1	Monitor Byte2	Pay-load	Signal detection	Check sum	End Flag
0xFC	1 Byte	0x0727	1Byte	1Byte	1Byte	1Byte	1Byte	1Byte	1Byte	x Bytes	2 Bytes	2Bytes	0xFD

Packet No: The Packet number for the advanced ECG data packet is 8 bit long.

Time Stamp: The advanced ECG data packet provides the 28 bit long Time Stamp. The time stamp increases by one each 2 ms. The time stamp starts up to count from zero on every received START_STOP_ECG_TRANSMISSION_ADVANCED command (see section 5.2.5 – command 0x0927). The highest bits of each Time Stamp bytes are set to zero. So we avoid needless octet stuffing.

Each Time stamp represents the offset in 2ms ticks to the point when starting the data transmission. The time stamp delivers the time, when the first ECG data set in the payload field was sampled. The time stamp in packets where no payload is, delivers the time when the ECM packet was finished.

The following figure illustrates the association between the time stamp and the ECG and ECM packet types.

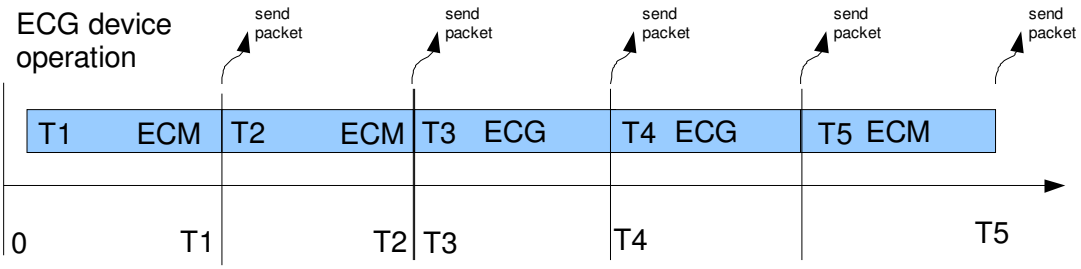

Figure 1 Time Stamp

Pulse: The ECG device provides a pulse rate calculation algorithm. Each ECG data packet contains the Pulse byte, which delivers the pulse in bpm. Of course the Pulse byte does not update if ECM is running.

Monitor Byte1/2: The Monitor Bytes deliver some status information of the ECG device. These Bytes are described more in detail in section 5.2.7.1.

Payload: The payload contains the ECG data sets in a compressed format. The payload structure is described more in detail in section 0.
During a running electrode contact measurement (ECM) the payload doesn’t contain any ECG data.

At the beginning of an ECG data transmission automatically starts an ECM. So the first data packets don't contain ECG data in the payload field. The ECM is running until all electrodes are connected to the patient.
During ECG measurement the ECM starts automatically again, if one of the extremity leads (R, L, F, N) has bad contact. During the ECM no ECG data can be transmitted.

5.2.7.1 Structure 'Monitor Byte 1' and 'Monitor Byte 2'

Monitor Byte1								Monitor Byte2							
P	BAT1	BAT2	HRU	HRL	L	R	F	packet type	N	V6	V5	V4	V3	V2	V1

Pacer detected P (has to be activated (see 5.2.16))

0 No Pacer impulse detected

1 A Pacer impulse was detected

Battery

BAT1	BAT2	
0	0	critical empty
0	1	empty
1	0	okay
1	1	full

Heart rate

HRU	HRL	
0	0	no heart rate limit exceed
0	1	lower heart rate limit exceed
1	0	upper heart rate limit exceed
1	1	not possible

Electrode contact L,R,F,N, V1-V6

1 Electrode has contact

0 Electrode has no contact

packet type

0 BT12[1]

1 BT3/6[1]

[1] This field is only supported in command 0x0727. In command 0x0724 it is set always zero.

5.2.7.2 Payload structure

Depending on the configurations the ECG payload consists of frames with channel II, III, V1…V6 (12 lead ECG) or II and III for 3-/6-Channel ECG.

Frame structure for 12 lead ECG

II	III	V1	V2	V3	V4	V5	V6

Payload: II, III, V1…V6, II, III, V1…V6, II, III, V1…V6,…, II, III, V1…V6

Frame structure for 3-/6-lead ECG

II	III

Payload: II, III, II, III, II, III, II, III, II, III, II, III, II, III,… ,II, III

Each data value is 15 Bits (signed) long. If only 7 Bits within a data value are used, the ECG device transmits only one Byte for a value. The missing Bits need to be filled up by the ECG monitor. To differ between 1 Byte and 2 Byte values a mask Bit (LSB) within the first Byte is used.

The resolution of the ECG data is 2,63 µV/bit.(referred to the input)

2 Byte value:

Byte1								**Byte2**							
sign	B_{13}	B_{12}	B_{11}	B_{10}	B_9	B_8	1	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

1 Byte value:

Byte1							
sign	B_5	B_4	B_3	B_2	B_1	B_0	0

The Leads I, aVR, aVL and aVF are calculated:

I = LA-RA = I-III

II = LL - RA

III = LL - LA

aVR = RA - 0.5 (LA + LL) = 0.5 III - II

aVL = LA - 0.5 (LL + RA) = 0.5 II –III

aVF = LL - 0.5 (LA + RA) = 0.5 (II +III)

5.2.7.3 Signal Detection

R wave detected	**Pacer detected**
1Byte (send first)	1Byte (send last)
0x0 – no R wave detected	0x0 no Pacer detected
0x1-0xFF R wave detected in Dataset 0x1-0xFF	0x1-0xFF Pacer detected in Dataset 0x1-0xFF

The signal detection bytes show, whether a pacer or R-wave was detected in the data transmitted within this packet. If a signal was detected the position of the dataset within the packet, where the signal was detected, is given. The first position in packet is position = 1. The value zero signals, that no signal was detected. The pacer signal byte is updated only, if the pacer detection is activated (see section 5.2.16).

Note: These both bytes where also send if no ECG data were transmitted during the ECG device is in offline electrode measurement mode. But then both signal detection bytes where set to zero.

5.2.8 SWITCH_BT_ROLE_REQ (0x0907)

This command initiates a request that the ECG device tries to get the Bluetooth master role. The operation as master reduces the current consumption of the ECG device.

Command	**no payload**
0x0907	

5.2.9 SWITCH_BT_ROLE_CFM (0x0707)

This command confirms the request.

Command	**role_slave**
0x0707	1Byte
	slave (0x01) master (0x00)

5.2.10 CONFIG_ECG_DEVICE_REQ (0x0910)

This command configures display mode, beeper volume, data logging, alarms und used power supply.

Command	Display Mode	Beeper VOL	DATA SAVE MODE_ON	Allow Acoustic Alarms	reserved	Power Supply – not active
0x0910	1 Byte	1Byte	1Byte	2 Bytes	1Byte	1Byte
	0x01 – full 0x02 - reduced	0x00 – OFF[1] … 0x05 - LOUD	0x00 – FALSE 0x01 - TRUE	send LSB first see the following table	0x00	0x01 – AKKU 0x02 - Battery

Alarms	Value	active/not active[1]
R-WAVE DETECTED	0x0001	active
ELEKTRODE NOT CONNECTED	0x0002	active
PULSE OUT OF RANGE	0x0004	active
CONNECTION_LOST_ALARM	0x0008	always active

[1]Note: activated acoustic alarms are not sounded, if the beeper volume is set to zero.

An alarm is active, if the appropriate bit is set. For example 0x0003 activates an acoustic signal for every R-Wave detected and every open electrode.

5.2.11 CONFIG_ECG_DEVICE_CFM (0x0710)

This command confirms the configuration.

Command	Display Mode	Beeper VOL	DATA SAVE MODE_ON	Allow Acoustic Alarms	reserved	Power Supply
0x0710	1 Byte	1Byte	1Byte	2 Bytes	1Byte	1Byte
	0x01 – full 0x02 - reduced	0x00 – OFF … 0x05 - LOUD	0x00 – FALSE 0x01 - TRUE	send LSB first see in 0 (note: Bit3 is always set)	reserved	0x01 – AKKU 0x02 - Battery

This command can be requested with the Command: Request (0x0800). For more information, see section 4.2.7.

5.2.12 TEST_BEEPER_VOLUME_REQ (0x0913)

This command is only for testing the configurable beeper volumes. Please use command 0 for setting beeper volume permanent.

Command	Beeper VOL
0x0913	1 Byte
	0x00 – OFF … 0x05 - LOUD

5.2.13 TEST_BEEPER_VOLUME_CFM (0x0713)

This is the confirmation for the Test Beeper Volume Request command.

Command	Beeper VOL
0x0713	1 Byte
	0x00 – OFF … 0x05 - LOUD

5.2.14 FLASH_DATA_REQ (0x0915)

If the Bt3/6 - 12 device losts its connection to the monitor, it stores data on a flash memory. This command reads out all buffered data. To enable this feature, please use command 0.

Command	no payload
0x0915	

5.2.15 FLASH_EMPTY_CFM (0x0715)

This command indicates that all data from flash are transmitted.

Command	no payload
0x0715	

5.2.16 SET_MEDICAL_PARAMETER_REQ (0x0916)

This command defines thresholds for heart rate alarms and configures pacemaker detection.

Command	upper heart rate	lower heart rate	pacemaker parameters	
0x0916	1 Byte	1Byte	1Byte	1 Byte reserved
	must be higher than lower heart rate threshold	must be lower than upper heart rate threshold	0x00 pm inactive 0x01 pm active	don't send this Byte

5.2.17 SET_MEDICAL_PARAMETER_CFM (0x0716)

This command confirms SET_MEDICAL_PARAMETER_REQ.

Command	upper heart rate	lower heart rate	pacemaker parameters	
0x0716	1 Byte	1Byte	1Byte	1 Byte reserved
	must be higher than lower heart rate threshold	must be lower than upper heart rate threshold	0x00 pm inactive 0x01 pm active	don't send this Byte

This command can be requested with the Command: Request (0x0800). For more information, see section 4.2.7.

6 Connection establishment

The Bluetooth connection is protected with the PIN "**1111**". A pairing by using this PIN is needed.

After the establishment of a Bluetooth connection, you can use the following sequence of commands to receive ECG data.

No.	ECG device	Direction	Monitor
1		←	Request Identification
2	Send Identification	→	
3		←	Request Protocol version
4	Send Protocol	→	
5		←	Config ECG channels
6	Confirm configuration	→	
7		←	Start
8 till x	ECG data	→	
x + 8		←	Stop