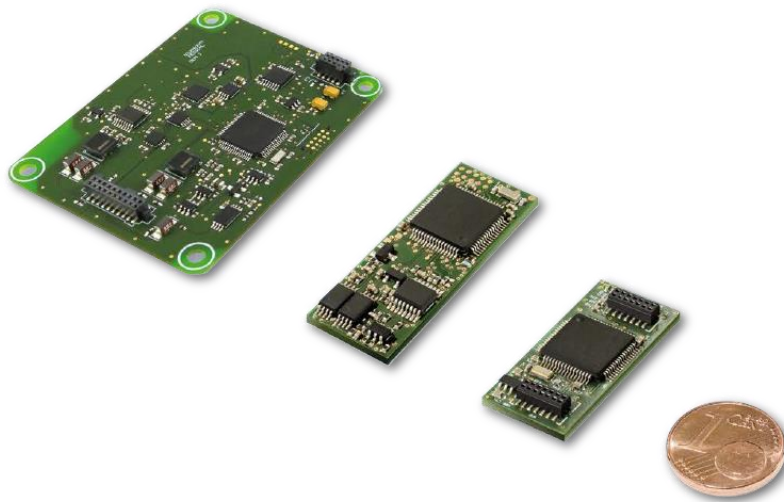


Communication Protocol

SMARTsat® OEM I/ II/ III



COMPANY CONFIDENTIAL

This document is the property of Bluepoint MEDICAL GmbH & Co. KG and is protected by German and international copyright law, and may not, without the written permission by Bluepoint MEDICAL GmbH & Co. KG, be given to any third parties.

Table of Contents

1	Transmission characteristics.....	4
1.1	Physical layer	4
1.2	Data-link layer.....	4
1.3	Host transmission wake-up sequence.....	5
2	Data packet structure	6
2.1	Data packet description	6
2.2	Read frames (Data sent by SMARTsat®).....	6
2.2.1	Start and end flags.....	6
2.2.2	Byte stuffing	6
2.2.3	CRC16 data verification	7
2.2.4	SMARTsat® data	7
2.2.4.1	Frame counter:	7
2.2.4.2	Data channel ID.....	8
2.2.4.3	Identifier	8
2.2.4.4	Data values	8
2.3	Commands sent by host.....	8
2.3.1	Build frame	8
2.3.2	Command.....	8
2.3.3	Change mode	9
3	Common device channel 0x01	10
4	Error channel 0x02	11
5	SMARTsat® channel 0x10	12
6	Examples.....	17
6.1	Change baud rate	17
6.2	UART send / receive in C	17
6.3	Build frame to switch on Raw Plethysmogram (RP) in C	19
7	Abbreviations.....	20
8	Revision History	21

SMARTsat® Communication Protocol

Document number: O-07-00-002

Document Revision: 11

Released: 04/2020

Communication Protocol Rev.: 11

The information in this document is subject to change.

Copyright © 2020 bluepoint MEDICAL GmbH & Co. KG. All Rights Reserved.

This document contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without prior written consent of bluepoint MEDICAL GmbH & Co. KG.

Contact information

bluepoint MEDICAL GmbH & Co. KG

An der Trave 15

23923 Selmsdorf

Germany

Phone: +49-(0)-38823-5488-0

Fax: +49-(0)-38823-5488-29

E-mail: info@bluepoint-medical.com

Web: www.bluepoint-medical.com

1 Transmission characteristics

1.1 Physical layer

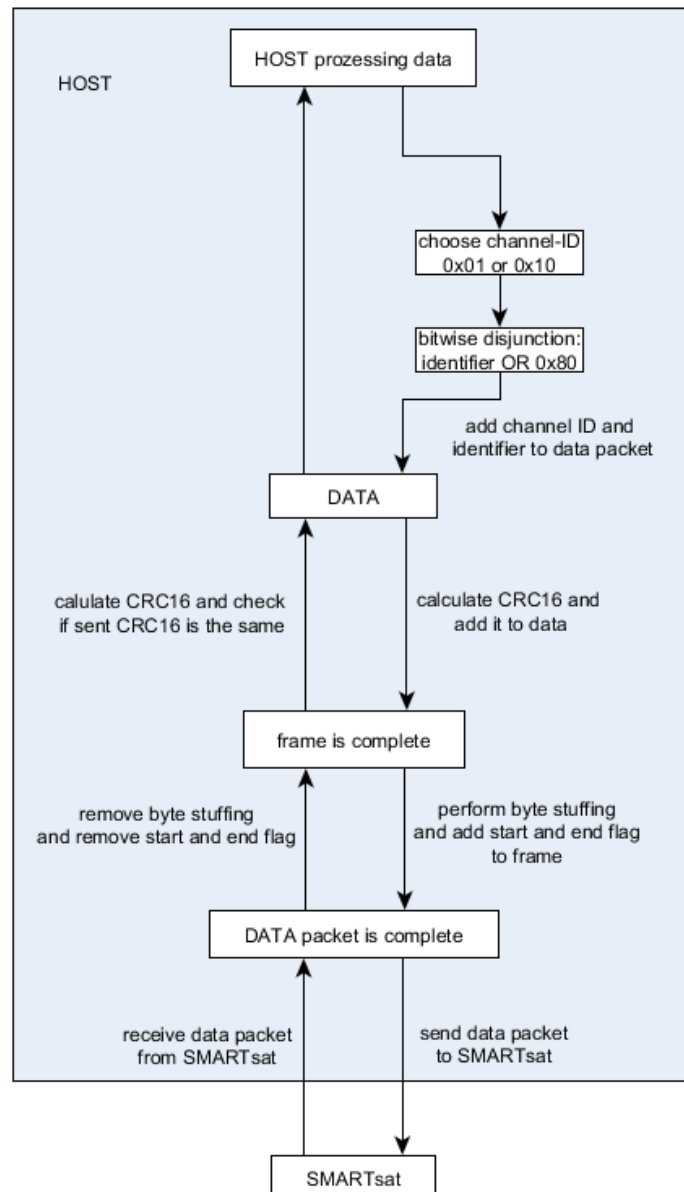
Serial UART interface:

- Baud rate is variable, default: 115200 Bd
- 1 start bit, 8 data bits, 1 stop bit, no parity bits (8-N-1)
- No hard or software handshake is used

1.2 Data-link layer

The data sent to the SMARTsat® module and received from it, is transmitted as packet. Each packet consists of a number of bytes. In general the SMARTsat® protocol is based on the „Serial Line Internet Protocol“.

The SMARTsat® module data packets are sent continuously. The package structure is described in section 2.2. The host command packets are sent occasionally as needed. The package structure is similar to the SMARTsat® module data packets, however no frame counter is sent and the identifier is linked with the attribute 0x80 OR (see 2.3). The figure below presents the data flow.



1.3 Host transmission wake-up sequence

Each command send from host to SMARTsat® OEM must be initiated by a **Wake-Up byte**.

This Wake-Up byte is necessary to ensure that no data is lost while the module is in a power save mode. The impressive low power consumption of the SMARTsat® OEM III is achieved due to this feature.

At the beginning of each command the host therefore needs to send one byte of data as Wake-Up byte to SMARTsat® (for example 0x77).

To start communication, send a wake-up byte (e.g. 0x77). Then send the data frame within 1 – 10ms after sending the wake-up byte.

For PC-applications e.g. written in C# a stopwatch timer is required to ensure the delay time stays within specifications (thread sleep methods are inaccurate).

After the delay time the transmission is completed by sending the data frame.

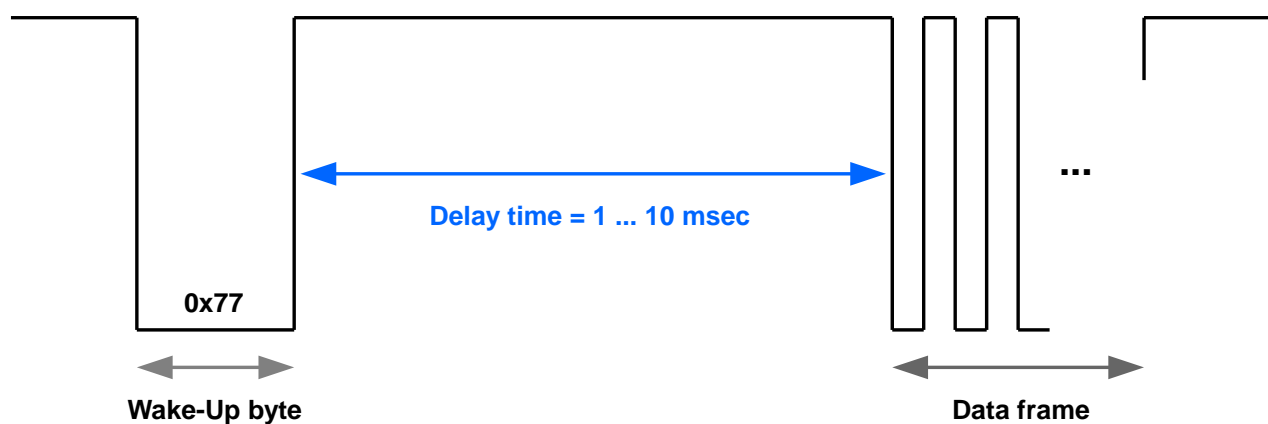


Figure 1-1: Timing of wake-up sequence

Example on how to request the device ID:

```
UART_Send(0x77);           // Send Wake-up byte
WaitMs(1);                 // Wait 1-10 ms
UART_Send(0xA8);           // Send command
UART_Send(0x01);           //
UART_Send(0x02|0x80);       //UART_Send(0x41);      //
UART_Send(0x80);           //
UART_Send(0xA8);           //
```

NOTE: Depending on baud rate a limited number of commands send by the host can be evaluated each second (at 9600 Bd smallest number). The *receive buffer overflow flag* is send, if this limit is reached.

2 Data packet structure

Each data packet received from SMARTsat® or command sent to the SMARTsat® has the following structure:

Packet			
Start Flag	Frame		End Flag
Start Flag	Data	CRC16	End Flag

2.1 Data packet description

Start Flag	Data	CRC16	End Flag
------------	------	-------	----------

Name	Description	Size
Start flag	Indicates the beginning of a frame: 0xA8	1 byte
Data	Data field	variable
CRC16	CRC16 data verification as indicated below from all characters in the field 'Data'. Send: High byte first	2 bytes
End flag	Indicates the end of a packet: 0xA8	1 byte

2.2 Read frames (Data sent by SMARTsat®)

2.2.1 Start and end flags

Start and end flags are used to find single frames in the data stream.

Start flag: 0xA8	Frame		End flag: 0xA8
Start flag: 0xA8	Data	CRC16	End flag: 0xA8

Both flags have the value **0xA8**. Find the bytes between the **0xA8** in the data stream to identify a frame. If there is no data between the **0xA8**, it is the start of a frame. In the next step the frame requires de-stuffing (see section 2.2.2)

2.2.2 Byte stuffing

Flag	Frame	Flag
------	-------	------

SMARTsat® uses a byte-stuffing algorithm on each frame to ensure that no 0xA8 is transmitted within the frame.

The reserved value 0xA8 is used for the start and end flags. The second reserved character is the control byte with value 0xA9. If in the block “data” or “CRC16” a byte is equal to 0xA8 or 0xA9, the byte is coded and replaced as follows:

- 0xA8 \triangleq 0xA9 0x88
- 0xA9 \triangleq 0xA9 0x89

The host can then clearly detect where one packet ends and the next one begins because **0xA8** is reserved to the start and end flags and will never show up within the frame.

If in the received data stream a control byte is detected, it is to be ignored and on the following byte the bitwise disjunction “byte OR 0x20” is execute (de-stuffing).

E.g.:

Packet sent by SMARTsat®	A8 FE 10 02 2A 32 3D 4B 5C 6C 7E 8F 9E A9 89 AF B0 AE A9 88 A1 00 80 A4 7E A8
Frame which must be de-stuffed	FE 10 02 2A 32 3D 4B 5C 6C 7E 8F 9E A9 89 AF B0 AE A9 88 A1 00 80 A4 7E
Frame after de-stuffing	FE 10 02 2A 32 3D 4B 5C 6C 7E 8F 9E A9 AF B0 AE A8 A1 00 80 A4 7E

After de-stuffing calculate the CRC according to section 2.2.3 to ensure data validity.

2.2.3 CRC16 data verification

The integrity of the data received is verified by calculating the CRC16 check value of the “DATA” field and comparing it with the CRC16 check value sent in the frame. 0xFFFF is the starting value.

Start flag	Data	CRC16	End flag
------------	------	-------	----------

Example for CRC16 (cyclic redundancy check) in C:

```
const unsigned short wCRCTableAbs[] =
{
    0x0000, 0xCC01, 0xD801, 0x1400, 0xF001, 0x3C00, 0x2800, 0xE401,
    0xA001, 0x6C00, 0x7800, 0xB401, 0x5000, 0x9C01, 0x8801, 0x4400,
};

unsigned short _Data2CRC16(char *pchMsg, unsigned short wDataLen )
{
    unsigned short i, wCRC = 0xFFFF;
    char chChar;

    for (i = 0; i < wDataLen; i++)
    {
        chChar = *pchMsg++;
        wCRC = wCRCTableAbs[(chChar ^ wCRC) & 15] ^ (wCRC >> 4);
        wCRC = wCRCTableAbs[((chChar >> 4) ^ wCRC) & 15] ^ (wCRC >> 4);
    }

    return wCRC;
}
```

2.2.4 SMARTsat® data

The DATA sent by the SMARTsat® has the following structure:

Flag	Data				CRC16	Flag
Flag	Frame counter	Data channel-ID	Identifier	Value	CRC16	Flag
1 byte	1 byte	1 byte	1 byte	various bytes	2 bytes	1 byte

2.2.4.1 Frame counter:

Each data packet sent from the SMARTsat® module is consecutively numbered (frame counter).

Flag	Frame counter	Data channel-ID	Identifier	Value	CRC16	Flag
------	---------------	-----------------	------------	-------	-------	------

The frame counter is used to detect data packets lost during data transmission. The frame counter is an unsigned 8 bit value ranging from 0 to 255. It begins with 0 and is incremented with each data block. After reaching the frame counter 255, the next number will be set to 0, and so on.

2.2.4.2 Data channel ID

The protocol differentiates the data by using a data channel ID.

Flag	Frame counter	Data channel-ID	Identifier	Value	CRC16	Flag
------	---------------	-----------------	------------	-------	-------	------

The following channels are available:

Data channel ID	Name	Description
0x01	Common device information	Identifies the module hardware and firmware
0x02	Error channel	List of errors detected by SMARTsat®
0x10	SMARTsat® channel	Channel for configuration of SMARTsat® and receiving the measurement results and parameters.

2.2.4.3 Identifier

The data identifier is used to differentiate the different module parameters and values sent within each data channel. Possible identifiers for each Data Channel are listed in the sections 3 to 5.

Flag	Frame counter	Data channel-ID	Identifier	Value	CRC16	Flag
------	---------------	-----------------	------------	-------	-------	------

2.2.4.4 Data values

The relevant information transported by the data package is coded in the VALUE of the Common device information channel (see section 3) and the SMARTsat® channel (see section 5).

Flag	Frame counter	Data channel-ID	Identifier	Value	CRC16	Flag
------	---------------	-----------------	------------	-------	-------	------

In case of the Error channel, the relevant information is coded in the IDENTIFIER (see section 4)

2.3 Commands sent by host

2.3.1 Build frame

Building frames is done in reverse order to reading the frames. First identify data to be sent (see section 5). Execute a bitwise disjunction with the identifier and 0x80.

Now calculate the CRC16 (section 2.2.3) to complete the frame.

Perform byte stuffing on the complete frame (section 2.2.2), thereafter add the start and end flag.

The command package is ready to be sent to the SMARTsat®.

2.3.2 Command

Commands sent by the host to SMARTsat® must be sent in the following data packet structure:

Flag	Data				CRC16	Flag
Flag	Data channel-ID	Identifier OR 0x80	Value	CRC16	Flag	

The time between two consecutive commands sent to SMARTsat® has to be larger than 100ms. The SMARTsat® confirms the command. Send the command again, if the command is not confirmed within 100ms.

The data included in a command package is described in the table below.

Item	Description for commands sent by host (refer to section 2.2.4 for description of the SMARTsat® data structure)
Frame counter	NOTE: No frame counter is included in the command packages sent by the host to SMARTsat®
Data channel ID	Commands sent by the host and data sent by SMARTsat® use the same data channel ID. Refer to section 2.2.4.2.
Data identifier	For commands sent by the host a bitwise disjunction of the data identifier and the attribute 0x80 shall be performed. Possible identifiers for each Data Channel are listed in the sections 3 to 5. e.g. host identifier = 0x31 0x80 = 0xB1 NOTE: For messages from SMARTsat® to the host (Answer), the Identifier has no attribute and the relevant information follows in further bytes.
Command value	A command is sent by the host to SMARTsat® to change a mode or to ask for the current setting. A VALUE is included in the data package, depending on the type of request.

At any time the host can ask for the current setting within the SMARTsat® channel or request common device information. In this case no VALUE is included in the package if the command value size is defined as 0 Byte:

Flag	Data channel-ID	Identifier OR 0x80	CRC16	Flag
------	-----------------	---------------------------	-------	------

Example in HEX - Request which sensor type is connected:

A8 10 86 D2 8D A8

Also see section 3 and 5, column: “Host command value size” is set to 0 Byte in case an identifier can be requested. If set to n.a., it is not possible to request the identifier. If the “command value size” is > 0 Byte, see section 2.3.3 for more detail.

2.3.3 Change mode

Different measurement modes are available. The host can select a mode by setting the relevant mode VALUE (see section 5, column “Host command value size” at least 1 Byte).

Flag	Data channel-ID	Identifier OR 0x80	Value	CRC16	Flag
------	-----------------	---------------------------	-------	-------	------

A summary of selectable measurement modes in SMARTsat® channel 0x10 are listed in the table below.

Mode Name	Identifier at SMARTsat® channel 0x10 *
Response time setting (SpO ₂ and Pulse Rate)	0x10
Pulse rate mode	0x12
Status information send frequency	0x17
Auto Scaled Plethysmogram (ASP) on/off	0x18
Raw Plethysmogram (RP) on/off	0x19
Sample Rate	0x1A
Baud rate setting	0x31

* Refer to section 5 for a detailed description of possible settings.

3 Common device channel 0x01

Identifier	Value	SMARTsat® data value size	Host command value size
0x01	Protocol version as string E.g. "rev. 10"	variable	0 Bytes
0x02	Device identification (module ID) as string E.g. "01" = SMARTsat® OEM I "03" = SMARTsat® OEM III "07" = SMARTsat® OEM II	variable	0 Bytes
0x03	Firmware version as string E.g. "BM.03.B36.A24.1B"	variable	0 Bytes
0x04	Hardware version as string E.g. "V3.3.1 Rev.B"	variable	0 Bytes
0x05	Serial number as string E.g. "1828320001"	10 Bytes	0 Bytes
0x06	Sent once at device start-up NOTE: at repeated reset of the module the host is recommended to display message "Device Defective". A typical reason for reset of the module is an unstable or fluctuating supply voltage	0 Bytes	NA

NOTE: Each time the module is switched on (also after reset) it will send the start-up frame, firmware version and serial number.

Example in HEX at start-up for FW "BM.03.B36.A24.1B" and SN "1828320001":

Start-up frame: A8 00 01 06 52 F0 A8

FW: A8 01 01 03 42 4D 2E 30 33 2E 42 33 36 2E 41 32 34 2E 31 42 5C 52 A8

SN: A8 02 01 05 31 38 32 38 33 32 30 30 30 31 8F 57 A8

4 Error channel 0x02

If the SMARTsat® module receives faulty data packets or detects an internal module error, the corresponding error message will be sent to the host. The SMARTsat® module does not accept any messages on this channel.

Error messages are transmitted for the duration of error occurrence at 1 Hz on the data channel 0x02 according to the structure below.

Flag	Frame counter	0x02	Identifier	CRC16	Flag
------	---------------	------	------------	-------	------

The errors are listed in the table below.

Error channel 0x02:

Identifier	Error	Description
0x01	Unknown channel	SMARTsat received a command with unknown channel.
0x02	Unknown identifier	SMARTsat received a command with unknown identifier.
0x03	Invalid value	SMARTsat received a command with invalid value (e.g. a not defined baud rate).
0x04	Selected baud rate is too slow	For activation of the Raw Plethysmogram (RP) set at minimum baud rate of 115200 Bd.
0x05	Receive buffer overflow	SMARTsat is receiving too much data
0x06	Frame corrupt, CRC error	SMARTsat is receiving data packets with Frame corrupt or CRC error
0x07	Sensor Error: Red LED defective	The red LED at the sensor is defective. NOTE: status information 'Sensor defective' (Byte[0] Bit: 1 in channel 0x10, identifier 0x01) is sent at the same time .
0x08	Sensor Error: Infrared LED defective	The infrared LED at the sensor is defective. NOTE: status information 'Sensor defective' (Byte[0] Bit: 1 in channel 0x10, identifier 0x01) is sent at the same time .
0x09	Sensor Error: Photodiode defective	The detector at the sensor is defective. NOTE: status information 'Sensor defective' (Byte[0] Bit: 1 in channel 0x10, identifier 0x01) is sent at the same time .
0x0A	Sensor Error: Short circuit	Short circuit occurred in the sensor cable or connector. Measurement is interrupted. Disconnect sensor to reset the error.
0x10	Boot error	The boot-test during module switch-on failed.
0x11	Self-test error	During module switch-on a self-test is performed. This error is sent if the self-test failed
0x12	Buffer overflow	An internal buffer overflow error occurred and module restarts after sending the error message. NOTE: if the error message repeats, display a technical error message for at least 1 second upon occurrence of the error
0x13	Command to switch on Auto Scaled Plethysmogram (ASP) is not accepted	Raw Plethysmogram (RP) and Auto Scaled Plethysmogram (ASP) cannot be switched on simultaneously. First switch off the RP before switching on the ASP.

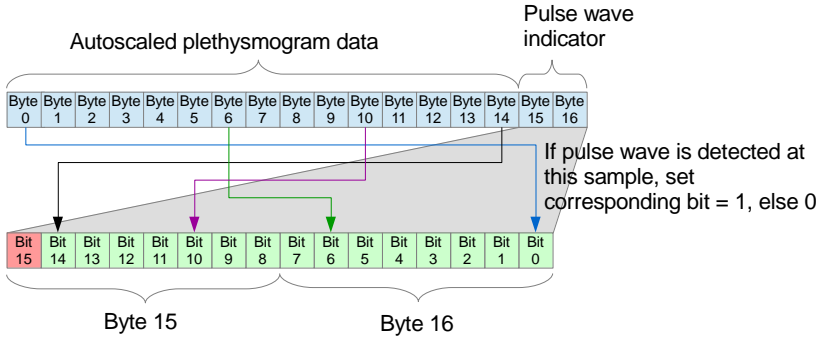
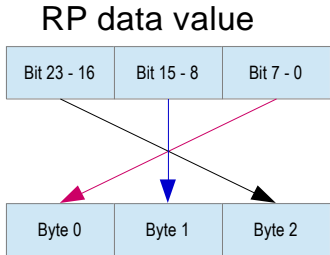
Example in HEX:

The following package is sent by SMARTsat® if the host sends a command with unknown identifier:

A8 53 02 02 70 01 A8


5 SMARTsat® channel 0x10

Identifier	Value (*for description to build the Host Command see section 2.3.2)	SMARTsat®		* Host Com. Value size
		Send freq.	Size	
0x01	<p>Status information (if Bit is set to 1)</p> <p>Byte[0]:</p> <p>Bit 0: Sensor disconnected</p> <p>Bit 1: Sensor defective</p> <p>Bit 2: Wrong sensor</p> <p>Bit 3-7: Reserved</p> <p>Byte[1]:</p> <p>Bit 0: Probe off</p> <p>Bit 1: Searching for pulse</p> <p>Bit 2: Pulse searching longer than 30 sec</p> <p>Bit 3: Low perfusion index (IR AC/DC ratio <1%)</p> <p>Bit 4: Low transmission</p> <p>Bit 5-6: Reserved</p> <p>Bit 7: Loss of pulse (no value is displayed; typically due to prolonged bad signal quality)</p> <p>Alarm monitors should give at least medium priority alarm if this bit is set</p> <p>Byte[2]:</p> <p>Bit 0: Ambient light</p> <p>Bit 1: Interferences detected</p> <p>Bit 2: Motion artifacts</p> <p>Bit 3: Vital parameter out of range</p> <p>Bit 4: Supply voltage out of range</p> <p>Bit 5 - 7: Reserved</p> <div><div><div>Byte 0</div><div>Bit 0Bit 1Bit 2Bit 3 - 7</div><div>Sensor disconnected Sensor defective Wrong sensor Reserved</div></div><div><div>Byte 1</div><div>Bit 0Bit 1Bit 2Bit 3Bit 4Bit 5 - 6Bit 7</div><div>Finger out Searching for pulse Searching pulse >30s Low perfusion index Low transmission Reserved Loss of pulse</div></div><div><div>Byte 2</div><div>Bit 0Bit 1Bit 2Bit 3Bit 4Bit 5 - 7</div><div>Ambient lighth Interferences detected Motion artifacts Vital param. out of range Supply voltage out of range Reserved</div></div></div> <p>The SMARTsat® module sends the status information setting continuously five times each second (5 Hz).</p> <p>Sending frequency can be changed to 1 Hz (see identifier: 0x17).</p> <p>Byte[0] is sent first.</p>	5 Hz (default) or 1 Hz	3 Bytes	0 Bytes

Identifier	Value (*for description to build the Host Command see section 2.3.2)	SMARTsat®		* Host Com. Value size
		Send freq.	Size	
0x02	<p>Auto Scaled Plethysmogram (ASP) with pulse beep indicator</p> <p>Byte [0-14]: The Auto Scaled Plethysmogram (ASP) data block with 15 samples. Resolution: 8-bit at 75Hz.</p> <p>Byte [15-16]: The pulse beep indicator is represented by an array of 15 bits (Byte 15 and 16) corresponding to the 15 plethysmogram data points in Byte 0 to 14. If a heartbeat is detected in sample 0 ... 14, the corresponding bit is set to 1, otherwise the bit is 0.</p> <p>The high byte is sent first as indicated below. The pulse beep indicator with reference to the plethysmogram samples is graphically described below:</p>  <p>The SMARTsat® module sends the plethysmogram continuously five times each second (5 Hz). Sending can be switched off (see identifier: 0x18)</p>	5 Hz (default) / 0 Hz	17 Bytes	n. a.
0x03	<p>Raw Plethysmogram (RP)</p> <p>Byte[0-2]: Raw Plethysmogram (RP) data block with one sample (data point). Each plethysmogram data sample is an unsigned 24 bit value represented by three bytes. The lowest byte (Byte 0) is sent first. The RP data send order is described below:</p>  <p>The time resolution of the plethysmogram is defined by the selected sample rate setting (0x1A). It can be set to 75Hz or 300 Hz sample rate The default setting is 0 Hz (off) The data samples are sent at a constant delay of:</p> <ul style="list-style-type: none"> - 13.3 ms in 75Hz sample rate - 3.3 ms in the 300Hz sample rate <p>No data is sent if the "Probe off" status is active.</p>	0 Hz (default) / 75 Hz / 300 Hz	3 Bytes	n. a.

Identifier	Value (*for description to build the Host Command see section 2.3.2)	SMARTsat®		* Host Com. Value size
		Send freq.	Size	
0x04	<p>Results and indicators</p> <div> <div> <p>SpO₂ value in % [0-100], 0xFF if no value</p> <p>Pulse rate in bpm [0-300], 0xFFFF if no value</p> <p>Perfusion Index in ‰ [0-200], 0xFFFF if no value</p> <p>Signal quality in % [0-100], 0xFF if no value</p> <p>Measurement settings</p> </div> <div> <p>Byte 0</p> <p>Byte 1 – 2 Hi Lo</p> <p>Byte 3 – 4 Hi Lo</p> <p>Byte 5</p> <p>Byte 6</p> </div> <div> <p>Bit 0</p> <p>Bit 1</p> <p>Bit 2</p> <p>Bit 3</p> <p>Bit 4</p> <p>Bit 5</p> <p>Bit 6</p> <p>Bit 7</p> </div> <div> <p>Response time: stable</p> <p>Response time: standard</p> <p>Response time: sensitive</p> <p>Response time: 8-Beat averaging</p> <p>Response time: 4-Beat averaging</p> <p>Pulse Rate mode: standard (30 - 240bpm)</p> <p>Pulse Rate mode: EPR Mode (20 - 300bpm)</p> <p>Is new measurement (SpO₂ and pulse rate value new. In general new values are calculated each second. The longest Data Update Period is 28sec. However the vitalparameters may change without setting this flag; then there was no actual new result, the value came from the history.)</p> </div> </div> <p>Byte[0]: SpO₂ Value [1...100 %], 0xFF = no value</p> <p>Byte[1-2]: Pulse rate [1...300 bpm], 0xFFFF = no value, (Hi+Lo)</p> <p>Byte[3-4]: Perfusion Index PI= I_{AC}/I_{DC} [1...200 ‰], 0xFFFF = no value, (Hi+Lo), (1 ‰ = 0.1 %) NOTE: PI<0.1 % is set to 0.1 %</p> <p>Byte[5]: Signal quality [1...100 %], 0xFF = no value</p> <p>The order of sending is as follows: SpO₂, pulse rate, pulsation strength, signal quality.</p> <p>Byte[6]: Measurement settings (active if respective bit is set to 1)</p> <p>Bit 0: Response time: stable</p> <p>Bit 1: Response time: standard</p> <p>Bit 2: Response time: sensitive</p> <p>Bit 3: Response time: 8-Beat averaging</p> <p>Bit 4: Response time: 4-Beat averaging</p> <p>Bit 5: Pulse Rate mode: Standard mode (30 – 240 bpm)</p> <p>Bit 6: Pulse Rate mode: EPR Mode (20 – 300 bpm)</p> <p>Bit 7: New measurement (SpO₂ or pulse rate value new. In general new values are calculated each second. The longest Data Update Period is 28sec)</p> <p>The high bytes are sent first.</p> <p>The SMARTsat® module sends this data continuously one time each second (1 Hz).</p>	1 Hz	7 Bytes	0 Bytes
0x05	Reserved	-	-	-

Identifier	Value (*for description to build the Host Command see section 2.3.2)	SMARTsat®		* Host Com. Value size
		Send freq.	Size	
0x06	Sensor type Byte[0-1]: sensor type value = Byte[0]*0x100 + Byte[1] 0xFFFF /65535 = undefined sensor type (see status information in identifier 0x01for detailed reason e.g. sensor disconnected or wrong) 0x000A /10 = Type C Sensors (closed sensors): SC7500 SoftCap; SCM7500 SoftCap medium; SCP7500 SoftCap pediatric; SF7500 SoftFlap 0x0028 /40 = Type O Sensors (open sensors): W7500 SoftWrap; 10-AP Disposable adult; 10-PP Disposable pediatric; 10-IP Disposable infant 0x0032 /50 = Type E Sensors (ear sensors): EP7500 Ear Probe 0x005B /91 = Type N Sensors (neonatal sensors): 10-NP Disposable neonatal The high byte is sent first. NOTE: Customer specific firmware with customized Sensor Type Codes are listed in a separate document	on request	2 Bytes	0 Bytes
0x07-0x0F	Reserved	-	-	-
0x10	Response Time (RT) setting for SpO₂ and Pulse Rate 0x0 = get current setting 0x1 = stable 0x2 = standard (default) 0x3 = sensitive 0x4 = 8-Beat averaging 0x5 = 4-Beat averaging	on request	1 Byte	1 Byte
0x11	Reserved	-	-	-
0x12	Pulse Rate (PR) mode 0x0 = get current setting 0x1 = Standard Pulse Rate (SPR) mode (default): 30 to 240 bpm 0x2 = Enhanced Pulse Rate (EPR) mode : 20 to 300 bpm	on request	1 Byte	1 Byte
0x13 - 0x16	Reserved	-	-	-
0x17	Set send frequency of status information (value at identifier 0x01) 0x0 = get current setting 0x1 = 5 Hz (default) 0x2 = 1 Hz	on request	1 Byte	1 Byte
0x18	Set Auto Scaled Plethysmogram (ASP) 0x0 = get current setting 0x1 = ON (default) - 5 Hz send frequency 0x2 = OFF - 0 Hz send frequency	on request	1 Byte	1 Byte

Identifier	Value (*for description to build the Host Command see section 2.3.2)	SMARTsat®		* Host Com. Value size
		Send freq.	Size	
0x19	Switch Raw Plethysmogram (RP) ON/OFF 0x0 = get current setting 0x1 = ON - sent at selected frequency 0x2 = OFF (default) - sent at 0 Hz frequency Auto Scaled Plethysmogram (ASP) is switched off, when Raw Plethysmogram (RP) is switched on. <i>Note:</i> Error 0x13 is sent, if Auto Scaled Plethysmogram (ASP) is switched on while Raw Plethysmogram (RP) is on	on request	1 Byte	1 Byte
0x1A	Set Sample Rate (SR) 0x0 = get current setting 0x1 = 75 Hz sample rate 0x3 = 300 Hz sample rate	on request	1 Byte	1 Byte
0x1B - 0x2F	Reserved	-	-	-
0x30	SMARTsat® Software Reset	n. a	n. a.	0 Bytes
0x31	Baud rate setting 0x0 /0 = get current module baud rate 0x60 /96 = 9600 Bd 0x13 /19 = 19200 Bd 0x26 /38 = 38400 Bd 0x39 /57 = 57600 Bd 0x73 /115 = 115200 Bd (default) 0xE6 /230 = 230400 Bd Changing the baud rate takes approximately 1 second. NOTE: The default shipping baud rate of SMARTsat® is 115200 Bd. The host can change the baud rate. During this process the default value of 115200 Bd is overwritten by the new setting. Next time when SMARTsat® powers-up it starts with the last set baud rate setting.  The baud rate can only be changed 1000 times in total.	on request	1 Byte	1 Byte

6 Examples

6.1 Change baud rate

In this example, the host changes the baud rate of SMARTsat® module from 115200 Bd to 9600 Bd:

Command from host to SMARTsat® module (at 115200 Bd)

Start flag	Channel	Identifier	Value	CRC16 Hi	CRC16 Lo	End flag
0xA8	0x10	0xB1	0x60	0x7D	0x04	0xA8

Confirmation from SMARTsat® module to host (at 115200 Bd)

Start flag	Frame counter	Channel	Identifier	Value	CRC16 Hi	CRC16 Lo	End flag
0xA8	0x01	0x10	0x31	0x60	0xA5	0x15	0xA8

After approximately 1 second the new baud 9600 is valid.

6.2 UART send / receive in C

```

/*
The following C language functions send and receive SMARTsat® packets.
They depend on two functions, send_char() and recv_char(), which send
and receive a single character over the serial line.
*/

/* SMARTsat special character codes
*/
#define END      0xA8 /* indicates end of packet */
#define ESC      0xA9 /* indicates byte stuffing */
#define ESC_END  0x88 /* ESC ESC_END means END data byte */
#define ESC_ESC  0x89 /* ESC ESC_ESC means ESC data byte */

/* SEND_PACKET: sends a packet of length "len", starting at
* location "p".
*/
void send_packet(char *p, int len)
{
    /* send an initial END character to flush out any data that may
    * have accumulated in the receiver due to line noise
    */
    send_char(END);

    /* for each byte in the packet, send the appropriate character sequence */
    while(len--) {
        switch(*p) {
            /* if it's the same code as an END character, we send a special two character code so as not
            we sent an END */
            case END:
                send_char(ESC);
                send_char(ESC_END);
                break;

            /* if it's the same code as an ESC character, we send a special two character code so as not
            we sent an ESC */
            case ESC:
                send_char(ESC);
                send_char(ESC_ESC);
                break;

            /* otherwise, we just send the character */
            default:
                send_char(*p);
        }
    }
}

```

```
        p++;
    }

    /* tell the receiver that we're done sending the packet */
    send_char(END);
}

/* RECV_PACKET: receives a packet into the buffer located at "p". If more than len bytes are received, the packet will be truncated.
Returns the number of bytes stored in the buffer. */

int recv_packet(char *p, int len)
{
    char c;
    int received = 0;

    /* sit in a loop reading bytes until we put together a whole packet. Make sure not to copy them into the packet if we run out of room. */
    while(1) {
        /* get a character to process */
        c = recv_char();

        /* handle bytestuffing if necessary */
        switch(c) {

            /* if it's an END character then we're done with the packet */
            case END:

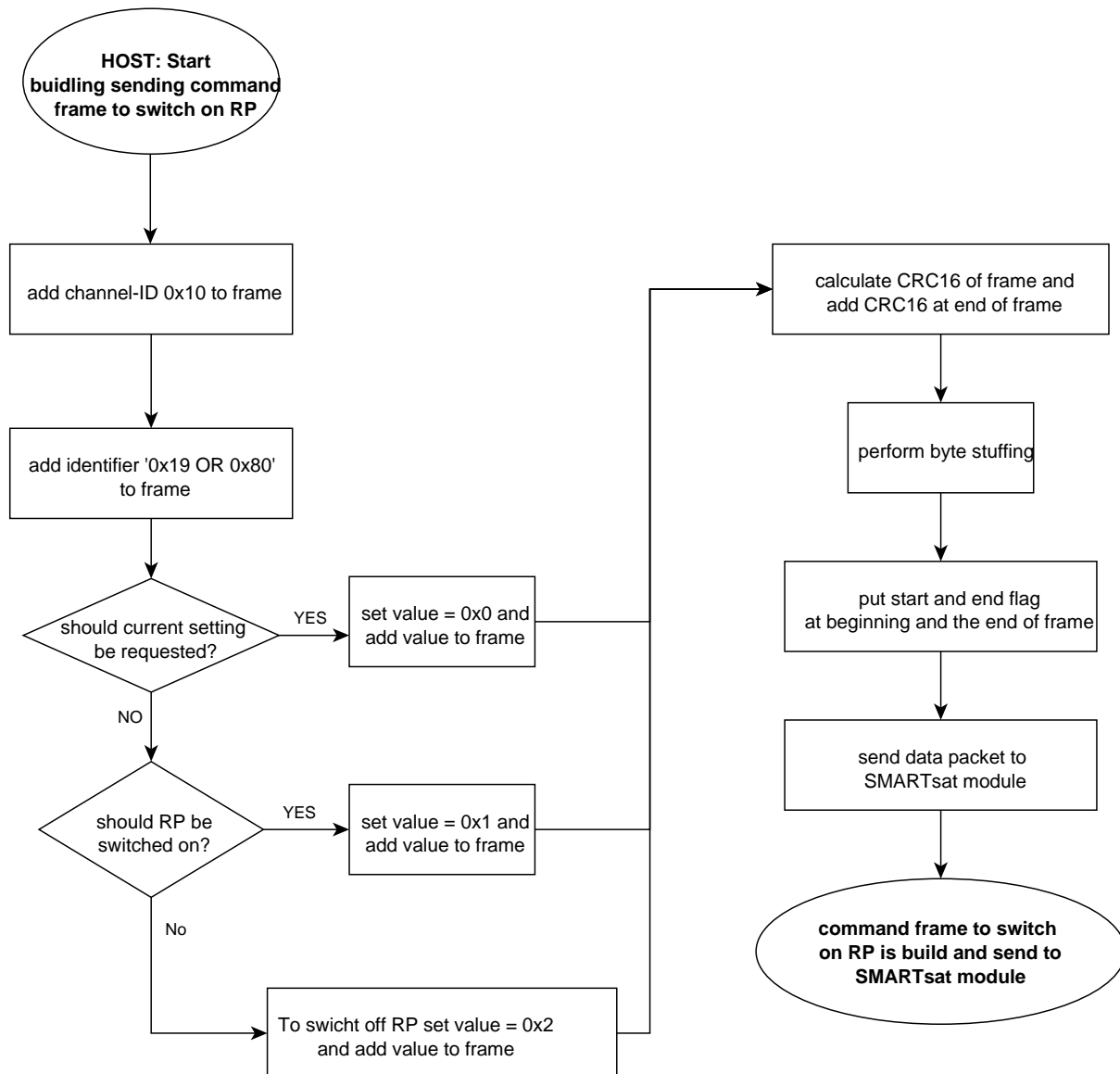
                /* a minor optimization: if there is no data in the packet, ignore it. This is meant to avoid bothering IP with all the empty packets
                generated by the duplicate END characters which are in turn sent to try to detect line noise. */
                if(received)
                    return received;
                else
                    break;

            /* if it's the same code as an ESC character, wait and get another character and then figure out what to store in the packet
            based on that. */
            case ESC:
                c = recv_char();

                /* if "c" is not one of these two, then we have a protocol violation. The best bet
                leave the byte alone and just stuff it into the packet*/
                switch(c) {
                    case ESC_END:
                        c = END;
                        break;
                    case ESC_ESC:
                        c = ESC;
                        break;
                }

                /* here we fall into the default handler and let it store the character for us */
            default:
                if(received < len)
                    p[received++] = c;
        }
    }
}
```

6.3 Build frame to switch on Raw Plethysmogram (RP) in C



```
uint32_t buildSwitchRawPlethFrame(uint8_t *fb, uint8_t rpSetting)
{
    uint16_t checksum;
    uint8_t *p = fb;

    // Channel
    *p++ = 0x10u;

    // Identifier
    *p++ = 0x19u | 0x80u;    // Message from host to SMARTsat --> bitwise disjunction of
                           // identifier and attribute 0x80

    // RP setting
    *p++ = rpSetting;

    // Checksum
    checksum = MWData2CRC16(fb, p - fb);    // Calculate CRC
    *p++ = (uint8_t) ((checksum >> 8) & 0xFFu);    // High byte
    *p++ = (uint8_t) (checksum & 0xFFu);    // Low byte

    return (p - fb);    // Return packet length
}

void switchOnRP(void)
{
    uint8_t frameBuffer[50];
    uint8_t frameLength;

    // Build frame to switch on RP
    // 0 = get current setting, 1 = switch on RP, 2 = Switch off RP
    frameLength = (uint8_t) buildSwitchRawPlethFrame(&frameBuffer[0], 1u);

    // Add transfer layer (Start/end flag, stuffing) and send frame
    send_packet(&frameBuffer[0], frameLength);
}
```

7 Abbreviations

n. a. = not applicable

EPR = Enhanced Pulse Rate

RP = Raw Plethysmogram

ASP = Auto Scaled Plethysmogram

IR = Infrared

AC/DC = Pulsatile part of the plethysmogram / Non-Pulsatile part of the plethysmogram

8 Revision History

Doc. Rev.	Effective Date	Prot. Rev.	Change description
11	2020-04-17	11	<ul style="list-style-type: none"> - Replace 150Hz sample rate by 300Hz sample rate (robust performance also at ripple on power supply) - rename “finger out”, to be “probe off” (status applies also to ear sensors) - update example in section 3 - Rework layout and format, , add document number
10	2019-10-16	10	<ul style="list-style-type: none"> - Add flowcharts and images to support understanding dataflow and structure. - Remove HRP mode (0x10) and replace with Raw Plethysmogram (RP), were applicable. - update example in section 6.3 - rename “pulsation strength” to be “perfusion index” - correct “undefined sensor” to 0xFFFF (before 0x00FF)
9	2017-06-29	9	Changes only available for OEM I and II: <ul style="list-style-type: none"> - Add Identifier 0x19 (Switch High Resolution Plethysmogram ON/OFF) - Add Identifier 0x1A (Set sample rate)
8	2017-01-31	8	<ul style="list-style-type: none"> - Add Identifier 0x0A (Sensor Error: Short circuit) to Error channel. Measurement is interrupted and error is sent together with “sensor defective status” if SMARTsat detects short at sensor LED. (Before measurement was interrupted and status “sensor defective” was sent).
7 - A	2016-09-06	7	<i>Note: change to document, communication protocol unchanged</i> <ul style="list-style-type: none"> - corrected typos - Add description to error channel (section 2)
7	2016-08-24	7	<ul style="list-style-type: none"> - Add Identifier 0x06 in Common device channel 0x01 - Add Identifiers 0x07, 0x08, 0x09 and 0x13 in Error channel 0x02 - Add Bit 7 at Identifier 0x01 of SMARTsat® channel 0x10 - Add Bit 7 at Byte 6 of Identifier 0x04 in SMARTsat® channel 0x10 - Update the Sensor Type ID (backward compatible) - Add information that HRP needs to be switched off before auto scaled pleth can be switched on again - update examples in HEX
6	2016-03-14	6	<i>Note: change to document, communication protocol unchanged</i> <ul style="list-style-type: none"> - Add description on how commands are send by host (section 2.3) - Add examples in HEX at “Common device channel”, “Error channel” - Add examples in C for “UART send/receive” and “build frame to switch on HRP”
5	2016-03-08	5	Add Features: <ul style="list-style-type: none"> - send single HRP data samples (3 Byte, before 45 Byte) - switch Auto scaled plethysmogram (ID 0x02) on/off - transmission frequency of status selectable from 5Hz to 1Hz - remove sending ID 0x10,0x12 at 1Hz, add sending these settings at ID 0x04,
4	2016-03-03	4	<i>Note: change to document, communication protocol unchanged</i> <ul style="list-style-type: none"> - Rename section 1, 2, 3, headers of tables - Add 1.1 Data link layer, add 1.2 Data package structure - Change images at “Auto scaled plethysmogram”, “HRP” - Add HEX values (not only decimal) to table under SMARTsat® channel
3	2016-02-19	3	<i>Note: change to document, communication protocol unchanged</i> <ul style="list-style-type: none"> - Changed: Format, length and examples for common device channel 0x01 - Add: HRP data send order, Pulse beep pointer bit positions, SpO2 send order, Status send order, Sensor type send order
2	2015-12-08	2	<ul style="list-style-type: none"> - Combined version for SMARTsat® OEM I – III (based on MW Protocol rev.2) - Removed: electrical properties of serial interface (part of Integration Guide) - Add: EPR 150 Hz, 300 Hz, example for destuffing, sensor types
1	2014-12-19	1	First version for Integration Guide based on MW Protocol rev. 1
0	2014-12-15	0	Initial version