XeThru serial protocol

This document describes the serial protocol used by the XeThru sensor modules.

Version: - Firmware version 1.1.0-beta. - Document revision: F

Notation

The following notation is used in this document:

```
<X> = Single byte
[X] = Multiple bytes
[“abc”] = [0x61,0x62,0x63] = Ascii text
[X(i)] = 32 bit Integer, 4 bytes
[X(f)] = 32 bit Float, 4 bytes
```

Protocol format

Binary protocol using flag bytes and escaping.

Example: `<Start> + [Data] + <CRC> + <End>`

Flag	bytes
<Start>	0x7D
<End>	0x7E
<Esc>	0x7F

Byte ordering

The XeThru serial protocol assumes litle-endian byte order.

Data escaping

Escaping means that if the escape byte occurs in data, the next byte is not <Start>, <End> or <Esc>, but intended byte with same value as flags.

Example: `0x7D + 0x10 + 0x7F + 0x7E + 0x04 + 0xFF + 0x7E`

Here the byte 0x7E in the middle is intended, and should not be read as a flag. The 0x7E byte is prepended with the escape byte 0x7F. After parsing for escape bytes, the data becomes:

`0x7D + 0x10 + 0x7E + 0x04 + 0xFF + 0x7E`

Checksum

Calculated by XOR’ing all bytes from `<Start> + [Data]`. Note that the CRC is done after escape bytes is removed. This means that CRC is also calculated before adding escape bytes.

How it works

When powering up the sensor module for the first time, it enters an idle mode. In this mode, the user can choose module behaviour by loading the desired application. After loading the application, it can be configured by sending application level commands. Finally, after configuring the application, the user can send a command to start it.

Next time the module is powered, it will automatically load previous configuration and resume operation.

If you want to change sensor module behaviour, reset the module and start again. Once the application is running, it is not possible to change parameters or application without performing a reset first.

Typical configuration workflow:

`Reset module --> Idle mode --> Load application --> Load parameters --> Run application`

USB connection When connecting to the module directly using USB, use the following procedure to ensure proper behaviour:

1. Reset module. Wait for ACK and immediately disconnect from the virtual serial port provided by the USB.
2. Module enters Idle mode
3. Load application
4. Load parameters
5. Run application

See sections on `Reset module` and `Ping` for more information.

Module level

Commands that control the module at a top level.

Reset module

Use this command to completely reset the sensor module. After the module has responded with ACK, it will wait for 0.5 seconds before it actually starts the reset procedure. This gives the host time to disconnect from the serial connection prior to the module reset, which is necessary when the USB interface is used.

Example: `<Start> + <XTS_SPC_MOD_RESET> + <CRC> + <End>`

Response: `<Start> + <XTS_SPR_ACK> + <CRC> + <End>`

Protocol codes:

Name	Value
XTS_SPC_MOD_RESET	0x22
XTS_SPR_ACK	0x10

After the module resets, it sends a set of system messages to inform the host about the bootup status. At first startup, the XTS_SPRS_BOOTING message is sent. Then, after the module booting sequence is completed and the module is ready to accept further commands, the XTS_SPRS_READY command is issued.

Message: `<Start> + <XTS_SPR_SYSTEM> + [Responsecode(i)] + <CRC> + <End>`

Protocol codes:

Name	Value
XTS_SPR_SYSTEM	0x30

Responsecode	Value
XTS_SPRS_BOOTING	0x10
XTS_SPRS_READY	0x11

Ping

The ping command can be used to check connection to the module, and verify module readyness.

During the module boot procedure, when connecting using USB, it can be difficult to make sure you are able to receive the system status messages saying that the module is ready. In that case, the PING command comes in handy, where you can ask the module if it is ready to receive commands.

Example: `<Start> + <XTS_SPC_PING> + [XTS_DEF_PINGVAL(i)] + <CRC> + <End>`

Response: `<Start> + <XTS_SPR_PONG> + [Pongval(i)] + <CRC> + <End>`

Protocol codes:

Name	Value	Description
XTS_SPC_PING	0x01	Ping command code
XTS_DEF_PINGVAL	0xeeaaeaae	Ping seed value
XTS_SPR_PONG	0x01	Pong responce code
XTS_DEF_PONGVAL_READY	0xaaeeaeea	Module is ready
XTS_DEF_PONGVAL_NOTREADY	0xaeeaeeaa	Module is not ready

Load application

Loads the desired sensor module application.

Example: <Start> + <XTS_SPC_MOD_LOADAPP> + [AppID(i)] + <CRC> + <End>

Response: <Start> + <XTS_SPR_ACK> + <CRC> + <End>

Protocol codes:

Name	Value
XTS_SPC_MOD_LOADAPP	0x21
XTS_SPR_ACK	0x10

AppID values can be found in the Application section (XTS_ID_APP_x).

Execute application

After the application is loaded, it can be configured using 'Application level' commands (see below). Then the application is executed by setting the module mode.

Name	Value	Description
XTS_SM_RUN	0x01	Sensor module in running mode.
XTS_SM_IDLE	0x11	Idle mode. Sensor module ready but not active.

Example: <Start> + <XTS_SPC_MOD_SETMODE> + <XTS_SM_RUN> + <CRC> + <End>

Response: <Start> + <XTS_SPR_ACK> + <CRC> + <End>

Protocol codes:

Name	Value
XTS_SPC_MOD_SETMODE	0x20
XTS_SPR_ACK	0x10

LED control

Use this command to choose the behaviour of the sensor LED. There are three levels of LED operations, Off, Simple and Full. Different applications may use the LED differently, but in general the three levels will behave like this: - Off: Very simple LED indicator during startup and initialization. LED is always off in operating mode. - Simple: More indication during startup and initialization. Simple indication during operation, e.g. fixed indication or subtle blinking. - Full: Full indication during startup and initialization. Extensive use of blinking and colors to indicate sensor state and if possible values.

Example: <Start> + <XTS_SPC_MOD_SETLEDCONTROL> + <Mode> + <Reserved> + <CRC> + <End>

Response: <Start> + <XTS_SPR_ACK> + <CRC> + <End>

Protocol codes:

Name	Value
XTS_SPC_MOD_SETLEDCONTROL	0x24
XTS_SPR_ACK	0x10

Mode	Value
XT_UI_LED_MODE_OFF	0
XT_UI_LED_MODE_SIMPLE	1
XT_UI_LED_MODE_FULL	2

Error handling

If the module receives a command it does not understand, it will send an error message.

Example: <Start> + <XTS_SPR_ERROR> + [Errorcode(i)] + <CRC> + <End>

Protocol codes:

Name	Value
XTS_SPR_ERROR	0x20

Errorcode	Value	Description
XTS_SPRE_NOT_RECOGNIZED	0x01	Command not recognized
XTS_SPRE_CRC_FAILED	0x02	Checksum failed
XTS_SPRE_APP_INVALID	0x20	Command recognized, but invalid

Application level

Generic application commands

Set Detection Zone Set the desired detection zone.

Example: `<Start> + <XTS_SPC_APPCOMMAND> + <XTS_SPCA_SET> + [XTS_ID_DETECTION_ZONE(i)] + [Start(f)] + [End(f)] + <CRC> + <End>`

Response: `<Start> + <XTS_SPR_ACK> + <CRC> + <End>`

Protocol codes:

Name	Value
XTS_SPC_APPCOMMAND	0x10
XTS_SPCA_SET	0x10
XTS_ID_DETECTION_ZONE	0x96a10a1c
XTS_SPR_ACK	0x10

Set Sensitivity Set the desired sensitivity.

Example: `<Start> + <XTS_SPC_APPCOMMAND> + <XTS_SPCA_SET> + [XTS_ID_SENSITIVITY(i)] + [Sensitivity(i)] + <CRC> + <End>`

Response: `<Start> + <XTS_SPR_ACK> + <CRC> + <End>`

Sensitivity value must be between 0 (low sensitivity) and 9 (high sensitivity).

Protocol codes:

Name	Value
XTS_SPC_APPCOMMAND	0x10
XTS_SPCA_SET	0x10
XTS_ID_SENSITIVITY	0x10a5112b
XTS_SPR_ACK	0x10

Respiration application (RESP)

RESP status Outputs the status of the RESP application, with data when available.

Example: `<Start> + <XTS_SPR_APPDATA> + [XTS_ID_RESP_STATUS(i)] + [Counter(i)] + [StateCode(i)] + [StateData(i)] + [Distance(f)] + [Movement(f)] + [SignalQuality(i)] + <CRC> + <End>`

StateCode values:

StateCode	Value	Description	StateData
XTS_VAL_RESP_STATE_BREATHING	0	Valid RPM detected	Current RPM value
XTS_VAL_RESP_STATE_MOVEMENT	1	Detects motion, but can not identify breath	0
XTS_VAL_RESP_STATE_MOVEMENT_TRACKING	2	Detects motion, possible breathing	0
XTS_VAL_RESP_STATE_NO_MOVEMENT	3	No movement detected	0
XTS_VAL_RESP_STATE_INITIALIZING	4	No movement detected	0
Reserved	5		0
XTS_VAL_RESP_STATE_UNKNOWN	6	Undefined state.	0

Output:

- StateData: RPM, respirations per minute (Breathing state only).
- Distance: Distance to where respiration is detected (Breathing state only).

- Movement: Relative movement of the respiration, in mm (Breathing state only).
- SignalQuality: A measure of the signal quality giving respiration. Typically used to identify if the sensor is positioned correctly. Value from 0 to 10 where 0=low and 10=high. (Breathing state only).

Protocol codes:

Name	Value
XTS_ID_APP_RESP	0x1423a2d6
XTS_SPR_APPDATA	0x50
XTS_ID_RESP_STATUS	0x2375fe26

Sleep application (Sleep)

Sleep status Outputs the status of the Sleep application, with data when available. Example: `<Start> + <XTS_SPR_APPDATA> + [XTS_ID_SLEEP_STATUS(i)] + [Counter(i)] + [StateCode(i)] + [StateData(f)] + [Distance(f)] [SignalQuality(i)] + [MovementSlow(f)] + [MovementFast(f)] <CRC> + <End>`

StateCode values:

StateCode	Value	Description	StateData
XTS_VAL_RESP_STATE_BREATHING	0	Valid RPM detected	Current RPM value
XTS_VAL_RESP_STATE_MOVEMENT	1	Detects motion, but can not identify breath	0
XTS_VAL_RESP_STATE_MOVEMENT_TRACKING	2	Detects motion, possible breathing	0
XTS_VAL_RESP_STATE_NO_MOVEMENT	3	No movement detected	0
XTS_VAL_RESP_STATE_INITIALIZING	4	No movement detected	0
Reserved	5		0
XTS_VAL_RESP_STATE_UNKNOWN	6	Undefined state.	0

Output:

- StateData: RPM, respirations per minute (Breathing state only).
- Distance: Distance to where respiration is detected (Breathing state only).
- SignalQuality: A measure of the signal quality giving respiration. Typically used to identify if the sensor is positioned correctly. Value from 0 to 10 where 0=low and 10=high. (Breathing state only).
- MovementSlow: A measure of movement with long integration time.
- MovementFast: A measure of movement with short integration time.

Protocol codes:

Name	Value
XTS_ID_APP_SLEEP	0x00f17b17
XTS_SPR_APPDATA	0x50
XTS_ID_SLEEP_STATUS	0x2375a16c

Baseband IQ output Outputs the baseband I and Q data of the application.

Example: `<Start> + <XTS_SPR_APPDATA> + [XTS_ID_BASEBAND_IQ(i)] + [Counter(i)] + [NumOfBins(i)] + [BinLength(f)] + [SamplingFrequency(f)] + [CarrierFrequency(f)] + [RangeOffset(f)] + [SigI(f)] + ... + [SigQ(f)] + ... + <CRC> + <End>`

Name	Description
Counter	Incremental frame counter
NumOfBins	Number of bins in dataset
BinLength	Length in meters between each bin
SamplingFrequency	Chip sampling frequency in Hz
CarrierFrequency	Chip carrier frequency in Hz
RangeOffset	First range bin start in meters
SigI	Array of NumOfBins float values of the signal I-channel
SigQ	Array of NumOfBins float values of the signal Q-channel

Protocol codes:

Name	Value
XTS_SPR_APPDATA	0x50
XTS_ID_BASEBAND_IQ	0x0C

Enable message The output of baseband IQ can be turned on and off using the enable message.

Example: `<Start> + <XTS_SPC_DIR_COMMAND> + <XTS_SDC_APP_SETINT> + [XTS_SACR_OUTPUTBASEBAND(i)] + [Length(i)] + [EnableCode(i)] + <CRC> + <End>`

Response: `<Start> + <XTS_SPR_ACK> + <CRC> + <End>`

EnableCode:

Name	Value
XTS_SACR_ID_BASEBAND_OUTPUT_OFF	0x00
XTS_SACR_ID_BASEBAND_OUTPUT_IQ	0x01

Protocol codes:

Name	Value
XTS_SPC_DIR_COMMAND	0x90
XTS_SDC_APP_SETINT	0x71
XTS_SACR_OUTPUTBASEBAND	0x10
Length	1
XTS_SPR_ACK	0x10

Baseband Amplitude/Phase output Outputs the baseband amplitude and phase data of the application.

Example: `<Start> + <XTS_SPR_APPDATA> + [XTS_ID_BASEBAND_AMPLITUDE_PHASE(i)] + [Counter(i)] + [NumOfBins(i)] + [BinLength(f)] + [SamplingFrequency(f)] + [CarrierFrequency(f)] + [RangeOffset(f)] + [Amplitude(f)] + ... + [Phase(f)] + ... + <CRC> + <End>`

Name	Description
Counter	Incremental frame counter
NumOfBins	Number of bins in dataset
BinLength	Length in meters between each bin
SamplingFrequency	Chip sampling frequency in Hz
CarrierFrequency	Chip carrier frequency in Hz
RangeOffset	First range bin start in meters
Power	Array of NumOfBins float values of the signal power
Phase	Array of NumOfBins float values of the signal phase

Power is calculated using: `power(n) = i(n)^2 + q(n)^2 for n=[0..NumOfBins-1]`

Phase is calculated using: `phase(n) = atan2( q(n) / i(n) ) for n=[0..NumOfBins-1]`

Protocol codes:

Name	Value
XTS_SPR_APPDATA	0x50
XTS_ID_BASEBAND_AMPLITUDE_PHASE	0x0D

Enable message The output of baseband amplitude and phase can be turned on and off using the enable message.

Example: `<Start> + <XTS_SPC_DIR_COMMAND> + <XTS_SDC_APP_SETINT> + [XTS_SACR_OUTPUTBASEBAND(i)] + [Length(i)] + [EnableCode(i)] + <CRC> + <End>` Response: `<Start> + <XTS_SPR_ACK> + <CRC> + <End>`

EnableCode:

Name	Value
XTS_SACR_ID_BASEBAND_OUTPUT_OFF	0x00
XTS_SACR_ID_BASEBAND_OUTPUT_AMPLITUDE_PHASE	0x02

Protocol codes:

Name	Value
XTS_SPC_DIR_COMMAND	0x90
XTS_SDC_APP_SETINT	0x71
XTS_SACR_OUTPUTBASEBAND	0x10
Length	1
XTS_SPR_ACK	0x10

Disclaimer

The information in this document is provided in connection with Novelda products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Novelda products. EXCEPT AS SET FORTH IN THE NOVELDA TERMS AND CONDITIONS OF SALES LOCATED ON THE NOVELDA WEBSITE, NOVELDA ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL NOVELDA BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF NOVELDA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Novelda makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Novelda does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Novelda products are not suitable for, and shall not be used in, automotive applications. Novelda products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.