API Documentation

Tactronik™

1 Objective **4**

2 Simple Message Protocol Library **5**
2.1 Serial Protocol Description 5
2.2 Checksum computation 5
2.3 Frame Example : 6
2.4 Message protocol description 6
2.4.1 Message 6
2.4.2 Header 7
2.4.3 Payload 7
2.4.4 Argument 7
2.4.5 String 8
2.5 Notes about AVR port 8

3 Tactronik UART Protocol Library **9**
3.1 Message List and ids : 9
3.1.1 ACK Message 10
3.1.2 ERROR Message 10
3.1.3 LOAD Message 10
3.1.4 PLAY Message 11
3.1.5 STOP Message 11
3.1.6 GET_VERSION Message 11
3.1.7 GET_PARAMETER Message 12
3.1.8 SET_PARAMETER Message 12
3.1.9 BIND_EFFECT Message 12
3.1.10 GET_SENSOR_VALUE Message 13
3.1.11 SET_SENSOR_VALUE Message 13
3.1.12 RESP_VERSION Message 14
3.1.13 RESP_PARAMETER Message 14
3.1.14 RESP_SENSOR Message 14
3.2 Note for long Messages 15

4 Revision History **16**

1 Objective

This document explains how to use the API provided by the Tactronik chip in order to access and communicate with it.

This API is composed of two parts, the *libsmp*, which stands for Simple Message Protocol Library, and the *libtup*, for Tactronik UART Protocol Library.

The *libsmp* aims to provide a simple protocol to exchange messages between two peers over a serial line. Those messages are identified by an ID, chosen by the user application, and contains 0 or more typed arguments.

The *libtup* defines the messages to control the Tactronik™ system designed by Actronika.

The current supported platforms of these libraries are :

- GNU / Linux (Posix systems should work as well)
- Windows
- ARM
- AVR (Atmega 328p and Atmega2560, see notes).

The *libsmp* can be found here : https://github.com/ActronikaSAS/libsmp

The *libtup* can be found here : https://github.com/ActronikaSAS/libtup

We advise our clients to use these libraries if possible instead of reimplementing the whole protocol.

2 Simple Message Protocol Library

2.1 Serial Protocol Description

It uses a framing protocol to ensure that data transferred over a serial protocol are correct.

The frame description is as below :

	Frame			
	Start Byte	Payload	Checksum	End Byte
Size (bytes)	1	N	1	1

- Start Byte : This byte is used to determinate the beginning of a frame.
- Payload : 0 or more bytes, it describes the payload to be sent to a peer. In the payload, reserved bytes (Start_byte, End_byte and Esc_byte) are escaped with Esc_byte.
- Checksum : checksum of the Payload, it can be escaped and is calculated using an algorithm described in 'Checksum computation' section.
- End Byte : Byte used to close the frame.

The reserved bytes are defined as below :

- Start Byte : 0x10
- End Byte : 0xFF
- Esc Byte : 0x1B

2.2 Checksum computation

The checksum is calculated with a simple algorithm, we apply XOR operations over each bytes of the payload :

```
uint8_t checksum = 0;
for(int i = 0; i < payload_size; i++)
{
	checksum ^= frame[i];
}
```

2.3 Frame Example :

Here is an example of a complete frame :

Considering that we want to send this payload with *libsmp* :

Payload				
0x33	0x1A	0xFE	0x10	0xC0

The full frame will be :

Frame								
Start Byte	Payload						Checksum	End Byte
0x10	0x33	0x1A	0xFE	0x1B	0x10	0xC0	0x07	0xFF

You can see that we wanted to send a 0x10 in the payload, the complete frame will include an escape byte in the payload before the reserved byte.

2.4 Message protocol description

A message sent over a serial communication with the Simple Message Protocol Library is defined that way :

2.4.1 Message

	Message	
	Header	Payload
Size (bytes)	8	N

- Header : The message header, described as below
- Payload : Content of the message

2.4.2 Header

	Header	
	ID	Size
Size (bytes)	4	4

This header identifies the message with an ID and indicates the size of the following payload.

- ID : The message ID
- Size : Payload Size

Both of these datas are Little-endian encoded.

2.4.3 Payload

The payload is a sequence of arguments with any padding between them. The size of each arguments depends on their types. If the message has no arguments, there is no payload in it.

Payload				
ARG1	ARG2	ARG3	...	ARGN

2.4.4 Argument

	Argument	
	Type	Data
Size (bytes)	1	N

- Type : This is the type of the arguments, that can be :
  - 0x01 : U8, 8 bits unsigned integer, size : 1 byte
  - 0x02 : I8, 8 bits signed integer, size : 1 byte
  - 0x03 : U16, 16 bits unsigned integer, size : 2 bytes
  - 0x04 : I16, 16 bits signed integer, size : 2 bytes
  - 0x05 : U32, 32 bits unsigned integer, size : 4 bytes
  - 0x06 : I32, 32 bits signed integer, size : 4 bytes
  - 0x07 : U64, 64 bits unsigned integer, size : 8 bytes
  - 0x08 : I64, 64 bits signed integer, size : 8 bytes
  - 0x09 : STR, C String nul-terminated
- Data : N bytes, encoded argument, in little-endian.

2.4.5 String

	String			
	0x09	Size	String content	0x00
Size (bytes)	1	2	N	1

- 0x09 : String type
- Size : String size, including nul-character, little-endian encoded
- String content : Data of the string
- 0x00: Nul-character

2.5 Notes about AVR port

AVR port is quite special as we don't have any OS on this platform. The serial device directly implements a driver for AVR UARTs modules.

As these systems don't have much RAM, it may need to tweak all buffers size. It can be done with "meson configure".

The serial device name have the following format : "serialX" with X a number from 0 to the maximum UART device of the desired chip. For instance, the Atmega328p has only one uart device, which is "serial0".

The returned fd on these platforms is not a file descriptor, and so, it shouldn't be used outside of the *libsmp* functions.

3 Tactronik UART Protocol Library

Within the Tactronik systems, numerous commands are available to control the haptic system. The Tactronik UART Protocol Library describes those commands with messages provided by *libsmp*.

For each message sent by a peer to a Tactronik module, the client will receive a response, either an ACK, an ERROR or a message from the RESP_x family. If the client receive no message, it means that the frame was never received by the module, or it got corrupted.

Tactronik use the following UART configuration :

- Baudrate : 115200
- Data width : 8 bits
- Stop : 1 bit
- Parity : none
- Control Flow : none

The following part of this documents will describe the differents commands and the content of its messages.

3.1 Message List and ids :

- 1 - ACK
- 2 - ERROR

- 10 - LOAD
- 11 - PLAY
- 12 - STOP
- 13 - GET_VERSION
- 14 - GET_PARAMETER
- 15 - SET_PARAMETER
- 16 - BIND_EFFECT
- 17 - GET_SENSOR_VALUE
- 18 - SET_SENSOR_VALUE

- 100 - RESP_VERSION
- 101 - RESP_PARAMETER
- 102 - RESP_SENSOR

3.1.1 ACK Message

Direction :

Tactronik ➔ Client

Description :

Message sent by Tactronik when a command has been received and processed without error.

Arguments :

U32 : id of the acknowledged command

3.1.2 ERROR Message

Direction :

Tactronik ➔ Client

Description :

Message sent by Tactronik when an error occurs when processing a command.

Error codes are not yet defined.

Arguments :

U32 : id of the command which caused the error

U32 : error code

3.1.3 LOAD Message

Direction :

Client ➔ Tactronik

Description :

Command to load an effect from the Haptic Effects Library to an effect slot

Arguments :

U8 : the slot where the effect will be loaded

U16 : id of the effect in the library

Expected response :

ACK, ERROR

3.1.4 PLAY Message

Direction :

Client ➔ Tactronik

Description :

Command to play the effect in the given slot

Arguments :

U8 : slot of the effect to play

Expected response :

ACK, ERROR

3.1.5 STOP Message

Direction :

Client ➔ Tactronik

Description :

Command to stop the effect in the given slot

Arguments :

U8 : slot of the effect to stop

Expected response :

ACK, ERROR

3.1.6 GET_VERSION Message

Direction :

Client ➔ Tactronik

Description :

Command to get the current version of Tactronik

Arguments :

None

Expected Response :

RESP_VERSION

3.1.7 GET_PARAMETER Message

Direction :

Client ➔ Tactronik

Description :

Command to get parameter values of an effect

Arguments :

U8 : slot of the effect from which the client want to get the parameter values

N * U8 : ids of the parameters to get

Expected Response :

RESP_PARAMETER, ERROR

3.1.8 SET_PARAMETER Message

Direction :

Client ➔ Tactronik

Description :

Command to set parameters values of an effect loaded in a slot

Arguments :

U8 : slot of the effect from which the client want to set the parameter values

U8 : id of the first parameter to set

U32 : value of the first parameter to set

U8 : id of the second parameter to set

U32 : value of the second parameter to set

...

Expected Response :

ACK, ERROR

3.1.9 BIND_EFFECT Message

Direction :

Client ➔ Tactronik

Description :

Bind the effect loaded in the given slot to actuators

Arguments :

U8 : slot of the effect to bind.

U8 : Binding Flag

Expected Responses :

ACK, ERROR

The binding flags are :

0x0 : None (unbind)
0x1 : Bind to actuator 1
0x2 : Bind to actuator 2
0x3 : Bind to both actuators

3.1.10 GET_SENSOR_VALUE Message

Direction :

Client ➔ Tactronik

Description :

Command to get the value of a sensor

Arguments :

N * U8 : Sensor id from which the client want to get a value

Expected Response :

RESP_SENSOR, ERROR

3.1.11 SET_SENSOR_VALUE Message

Direction :

Client ➔ Tactronik

Description :

Command to set values of sensors

Arguments :

U8 : id of the first sensor to set
U16 : Value of the first sensor to set
U8 : id of the second sensor to set
U16 : Value of the second sensor to set
...

Expected Response :

ACK, ERROR

3.1.12 RESP_VERSION Message

Direction :

Tactronik ➔ Client

Description :

Response to a GET_VERSION command. It contains the Tactronik version.

Arguments :

STR : Tactronik version

3.1.13 RESP_PARAMETER Message

Direction :

Tactronik ➔ Client

Description :

Response to a GET_PARAMETER command. It contains the desired parameters values.

Arguments :

U8 : slot of the effect

U8 : id of the first requested parameter

I32 : value of the first requested parameter

U8 : id of the second requested parameter

I32 : value of the second requested parameter

...

3.1.14 RESP_SENSOR Message

Direction :

Tactronik ➔ Client

Description :

Response to a GET_SENSOR command which contains the desired sensor values

Arguments :

U8 : id of the first requested sensor

I16 : value of the first requested sensor

U8 : id of the second requested sensor

I16 : value of the second requested sensor

...

3.2 Note for long Messages

The commands that allows to send or ask for many values, like SET_SENSOR_VALUE or GET_PARAMETERS for instance. It is currently limited to 16 arguments maximum.

4 Revision History

Date	Ver.	Ref.	Description	Author	Checked by	Approved by
21/11/2017	2.0	TAAD0002	Update of documentation with the new Tactronik API based on libsmp & libtup	Thomas Begeot	Aurelien Zanelli	Rafal Pijewski