API Documentation

The Evaluation kit

1. Objective

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

This paper will describe all the configurations of the different communications protocols available on the chip, as well as, how to choose the right protocol for a given application. Then the document will explain the commands to control the Tactronik system.

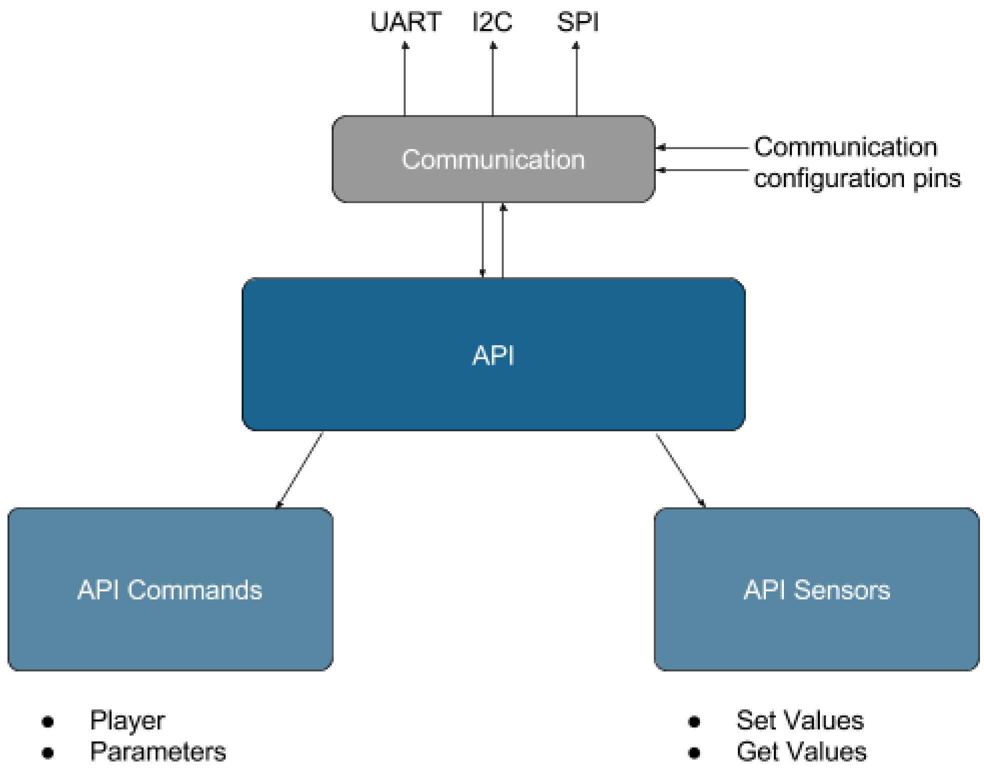
The description of this part will be completed with the document on the Tactronik Haptic Library.

2. APIs structure

The Tactronik API is designed to be work on multiple physical layer (UART/I2C/SPI) and is split in two parts: the command API and the sensor API.

The Command API is in charge of playing, configuring effects and giving board informations. The Sensor API is in charge of handling Tactronik physical and virtual sensor.

Below is a scheme to summarize the design:



3. Communication Protocols

Three communication protocols are available on the Tactronik chip, namely:

- UART
- SPI (Not available yet)
- I2C (Not available yet).

All above protocols are at the user's disposal but only one at a time can be used at runtime..

The communication protocols have been configured as follows:

3.1. UART

- Baudrate : 115200
- 8 Bits
- No parity
- 1 Stop bit
- No control flow

3.2. SPI

Not available yet

3.3. I2C

Not available yet

4. Commands Mapping & Structure

Within the Tactronik systems, numerous commands are available to control the haptic system.

APIs allow users to :

- get general information about the chip;
- control the activation/deactivation/loading of the Haptic Effects;
- get the integrated sensors values;
- provide external sensor values to the system and drive our algorithm with it.

To execute these commands, data must be sent according to our application interface.

4.1. General payload structure :

The entire communication with the Tactronik chip is performed through writing and reading data frames on the serial protocols. These frames must be structured as presented below.

4.1.1. Frame structure

Data transferred between Tactronik and the user's device are encapsulated in a frame with a header, and a checksum at the end. Frame structure and fields meaning are given below:

Data	Start Byte	Size	Type	Payload	Checksum
Size	8	5	3	n * 8	8

- **Start Byte:** defines the beginning of the communication between the devices.
- **Size:** Payload size
- **Type:** Api which will be used
    - 0x00 for the command API
    - 0x01 for the sensor API
- **Payload:** contains the data to be transferred.
- **Checksum:** is a byte used to verify the correctness of the transfer.

Each payload structures for every command will be detailed here below.

4.1.2. Checksum calculation

To assure that the transferred data hasn't been corrupted, an additional byte must be sent at the end of the communication. This byte helps each device to check the correctness of the data.

4.1.2.1. Calculation

This byte is calculated as a logic XOR on each byte of the frame, excluding the "Start Byte". The frame is considered as an array of char (8 bits). Some commands allow to transfer 16 or 32 bits of data. To calculate the checksum these values must be decomposed to n*8 bits values (ex : 0x1254 -> 0x12, 0x54).

```
uint8_t checksum = 0;
for(int i = 1; i < len(frame); i++)
{
        checksum ^= frame[i];
}
```

4.1.2.2. Example

For a sensor update, a correct frame could be :

Start Byte ; 0x19 ; 0x01 ; 0x0E ; 0x12 ; Checksum

Then, the corresponding checksum will be computed as :

Checksum = 0x19 ^ 0x01 ^ 0x0E ^ 0x12 = **0x04**

This way, the whole frame would be :

Start Byte ; 0x19 ; 0x01 ; 0x0E ; 0x12 ; **0x04**

4.2. Tactronik

This part will detail each command available on the version of the Tactronik system. For the moment, the "Start Byte" has been defined to be equal to **0xFF** (255).

As it has been mentioned before there are two kinds of API in the Tactronik system, the one which handles the commands and the one which manages the sensors.

4.2.1. Command API (0x00)

The type field of frame should be set 0x00 to select "Command API"

Data	Start Byte	Size	Type	API Command Payload	Checksum
Size (bits)	8	5	3	1 + n * 8	8
Value	0xFF	N	0x03	...	0xXX

The API Command payload contains the following fields:

Data	API Command Payload	
	Command Id	Command Payload
Size (bits)	8	n * 8

4.2.1.1. Table of commands

Command ID	Command
0x00	Stop
0x01	Play
0x02	Load
0x03	GetVersion
0x04	SetParameter
0x05	*“Reserved”*
0x06	BindEffectToActuator

4.2.1.2. Stop

Stop the given effect on the actuators it has been binded to.

Data	Command Id	Command Payload
		Effect ID
Size (bits)	8	8
Value	0x00	0xXX

- **Effect ID:** The effect ID

4.2.1.3. Play

This command plays the effect given in parameter.

Please note that for now, the firmware can only manage one effect at the same time.

Data	Command Id	Command Payload
		Effect ID
Size (bits)	8	8
Value	0x01	0xXX

- **Effect ID:** The effect ID

4.2.1.4. Load

Load an effect from the Haptic Effects Library to the desired location.

Data	Command Id	Command Payload	
		Effect ID	Effect Library id
Size (bits)	8	8	16
Value	0x02	0xXX	0xXXXX

- **Effect ID:** The desired effect id after load.
- **Effect Library ID:** The id of the effect in the effects library. See “Haptic Effects Library” documentation for the list of available effects.

4.2.1.5. GetVersion

Get the firmware version flashed on the chip.

Data	Command Id	Command Payload
Size (bits)	8	0
Value	0x03	

The firmware will reply with the following payload:

Data	Command Id	Command Payload
Size (bits)	8	N
Value	0x03	Version

- **Version:** String representation of the firmware version. Not nul-terminated.

4.2.1.6. SetParameter

Set a parameter of an effect.

Data	Command Id	Command Payload		
		Effect ID	Parameter ID	Parameter Value
Size (bits)	8	8	8	32
Value	0x04	0xXX	0xXX	0xXXXXXXXX

- **Effect ID:** The effect id
- **Parameter ID:** The parameter ID. See “Haptic Effects Library” documentation for parameter id for each effect.
- **Parameter Value:** The value to set.

4.2.1.7. BindEffectToActuator

Bind a loaded effect to the desired actuators. An effect can be binded to the first actuator, the second actuator or both of them.

Data	Command Id	Command Payload	
		Effect ID	Binding value
Size (bits)	8	8	8
Value	0x06	0xXX	0xXX

- **Effect ID:** The effect id
- **Binding value:** flags representing the binding between effects and actuators. For now, only the 2 LSBs (0bXXXXXXXXAB) are used:

A	B	
0	0	Unbind the effect
0	1	Bind to Actuator 0
1	0	Bind to Actuator 1
1	1	Bind to the two actuators

4.2.2. Sensors API (0x01)

The type field of frame should be set to 0x01 to select “Sensor API”

Data	Start Byte	Size	Type	API Sensor Payload	Checksum
Size (bits)	8	5	3	1 + n * 8	8
Value	0xFF	N	0x01	...	0xXX

4.2.2.1. SetSensorsValues

Update sensor values used in our algorithms given as { SensorID, SensorValue} pairs.

Data	API Sensor Payload						
	R/W	Number of Sensor	Sensor 1 ID	Sensor 1 Value	...	Sensor N ID	Sensor N Value
Size (bits)	1	7	8	16	...	8	16
Values	0x00	N	0xXX	0xXXXX	...	0xXX	0xXXXX

- **R/W:** Shall be set to 0x00.
- **Number of Sensor:** The number of sensor to set
- **Sensor 1 ID:** Id of the first sensor to set
- **Sensor 1 Value:** Value of the first sensor to set
- **...**
- **Sensor N ID:** Id of the last sensor to set
- **Sensor N Value:** Value of the last sensor to set

4.2.2.2. GetSensorsValues

Get values of given sensors.

Data	API Sensor Payload				
	R/W	Number of Sensor	Sensor 1 ID	...	Sensor N ID
Size (bits)	1	7	8	...	8
Values	0x01	N	0xXX	...	0xXX

- **R/W:** Shall be set to 0x01.
- **Number of Sensor:** The number of sensor to get
- **Sensor 1 ID:** Id of the first sensor to get the value of
- **...**
- **Sensor N ID:** Id of the last sensor to get the value of

For each value the user wants to get, a single frame will be sent by firmware containing the id of the sensor and its value. The payload is represented below:

Data	API Sensor Payload			
	R/W	Number of Sensor	Sensor ID	Sensor Value
Size (bits)	1	7	8	16
Values	0x01	1	0xXX	0xXXXX

- **R/W:** will be set to 0x01.
- **Number of Sensor:** will be 0x01
- **Sensor ID:** Id of the sensor.
- **Sensor ID:** Value of the sensor.

5. Revision History

Date	Ver.	Ref.	Description	Author	Checked by	Approved by
16/04/2017	1.0	TAEKAD0001	Tactronik - API Documentation	Thomas Begeot	Rafal Pijewski	