Tipro HID API

Communication with Tipro Controller over HID Interface

REFERENCE MANUAL

Operating and Programming Instructions

Tipro_HID_API_08.doc
Version 08
October 13, 2014

TIPRO
keyboards focusing on your future needs.

TABLE OF CONTENTS

A. INTRODUCTION

HID API is Tipro's Application Programming Interface.

By default the communication between Tipro hardware and Application software is one-way (if we ignore all low-level protocols); HID Keyboard events are sent on actions (e.g. PTT press), which can be defined in the configuration software ChangeMe.

Tipro HID API is needed in the following cases:

a) Change settings of Tipro Device (e.g. microphone sensitivity) from within the application software
b) Query a Tipro Device (e.g. get volume setting of loudspeakers in speakerbox)

Tipro HID API can be used in combination with Tipro devices that support HID Telephony to register a callback function and in this way to be notified of events (e.g. handset pick-up or PTT key pressed)

HID Telephony can also be used without HID API, if the application software supports HID Telephony devices directly.

Requirements HID API:

- Operating system: Windows XP and later
- Tipro Device with controller version 05.xx.31 and above

Requirements HID Telephony:

- Tipro Device with support for HID Telephony:
    - Handsets: TM-HTx
    - Speakerbox: TM-FxT
    - BeFREE 10

Note: HID Telephony is supported on a module base, not controller based. E.g. a Terminal comprising a BeFREE15, a handset –HTA- and a Speakerbox –FSU- can be accessed by the HID API if the controller firmware (in the BeFREE15) is 05.xx.31or higher. HID Telephony specific HID API functions are available only for the handset

B. SUPPORTED COMMAND SET

COMMAND			Directed to	SHORT DESCRIPTION
NAME	ARG	ANS		
Detect devices	-	-	Controller	Detect Tipro HID API supported devices
Enumerate modules	-	-	Controller	Enumerate TiproBus modules
Set Luminance	1	-	BeFREE15	Sets the luminance of the LCD screen.
Touchscreen disable	-	-	BeFREE15	Disables touchscreen.
Touchscreen enable	-	-	BeFREE15	Enables touchscreen.
Set LEDs	1	-	BeFREE15	Sets the user-defined LED state.
Set External Device Level	2	-	Speaker Box	Sets the volume of the handset / headset
Get External Device Level	1	1	Speaker Box	Returns the volume of the handset / headset
Set Sbx LED State	3	-	Speaker Box	Sets the LED state of the illuminated keys
Get Sbx LED State	2	1	Speaker Box	Returns the LED state of the illuminated keys
Set Speaker Level	2	-	Speaker Box	Sets the level of the handsfree speakers
Get Speaker Level	1	1	Speaker Box	Returns the level of the handsfree speakers
Get External Headset State	1	1	Speaker Box	Returns the state of the headset connected to Speaker Box module (connected/disconnected)
Mute Microphone	2	-	Speaker Box	Mutes/un-mutes the microphone signal
Set Microphone Level	2	-	Speaker Box	Sets the level of the microphone signal
Get Microphone Level	1	1	Speaker Box	Returns the level of the microphone signal
Set Microphone Level Ex	3	-	Speaker Box	Sets the level of the microphone signal for specific device
Set Microphone Optimal Distance	2	-	Speaker Box	Sets the optimal distance for the hands-free microphone
Get Microphone Optimal Distance	1	1	Speaker Box	Returns the optimal distance of the hands-free microphone
Set Microphone Threshold	2	-	Speaker Box	Sets the microphone threshold
Get Microphone Threshold	1	1	Speaker Box	Returns the microphone threshold
Set Microphone Compression	2	-	Speaker Box	Sets the microphone compression
Get Microphone Compression	1	1	Speaker Box	Returns the microphone compression
Set Microphone Threshold Ex	3	-	Speaker Box	Sets the microphone threshold for specific device
Set Microphone Compression Ex	3	-	Speaker Box	Sets the microphone compression for specific device
Speakerbox Go Online	1	-	Speaker Box	Puts Speaker Box to On-line mode
Is Speakerbox Online	1	1	Speaker Box	Returns On-line / Off-line mode
Set Alert Line	2	-	Speaker Box	Sets the function (mode) of the alert line
Get Alert Line	1	1	Speaker Box	Returns the function (mode) of the alert line
Set Active Device	2	-	Speaker Box	Sets the device where both speaker and microphone signals will be switched to.
BF10 Set Luminance	1	-	BeFREE10	Sets the luminance of the stripe and LCD screen.
BF10 Touchscreen disable	-	-	BeFREE10	Disables touchscreen.
BF10 Touchscreen enable	-	-	BeFREE10	Enables touchscreen.
BF10 Set Speaker Level	1	-	BeFREE10	Sets the level of the BF10 speakers
BF10 Get Speaker Level	-	1	BeFREE10	Returns the level of the BF10 speakers
BF10 Mute Microphone	1	-	BeFREE10	Mutes/un-mutes the microphone signal

COMMAND			Directed to	SHORT DESCRIPTION
NAME	ARG	ANS		
BF10 Set Microphone Level	1	-	BeFREE10	Sets the level of the microphone signal
BF10 Get Microphone Level	-	1	BeFREE10	Returns the level of the microphone signal
BF10 Set Microphone Optimal Distance	1	-	BeFREE10	Sets the optimal distance of the microphone
BF10 Get Microphone Optimal Distance	-	1	BeFREE10	Returns optimal distance of the microphone
BF10 Set Microphone Threshold	1	-	BeFREE10	Sets the microphone threshold
BF10 Get Microphone Threshol	-	1	BeFREE10	Returns the microphone threshold
BF10 Set Microphone Compression	1	-	BeFREE10	Sets the microphone compression
BF10 Get Microphone Compression	-	1	BeFREE10	Returns the microphone compression
BF10 Set PTT Key LED State	1	-	BeFREE10	Sets the LED state of illuminated PTT key
BF10 Set PTT Key LED State Ex	3	-	BeFREE10	Sets the operation of the illuminated PTT key with more parameters
BF10 Get PTT Key LED State	-	1	BeFREE10	Returns the LED state of the illuminated PTT Key
BF10 Set Analog Audio	1	-	BeFREE10	Sets the Analog audio options
BF10 Get Analog Audio	-	1	BeFREE10	Returns Analog audio options
Detect Telephony Devices	-	-	Telephony Device	Scans for telephony devices
Get Num Of Detected Telephony Devices	-	1	Telephony Device	Returns number of the connected telephony devices
Get Telephony Devices Product String	2	1	Telephony Device	Returns product string of the specific telephony device
Get Telephony Devices VID PID	1	2	Telephony Device	Returns vendor ID and product ID of the specific telephony device
Get Telephony Devices Manufacturer String	2	1	Telephony Device	Returns manufacturer string of the specific telephony device
Get Telephony Devices Path	2	1	Telephony Device	Returns the path of the specific telephony device
Register Telephony Callbacks	3	-	Telephony Device	Registers callbacks for key and status events
Register Telephony Callbacks Ex	3	-	Telephony Device	Registers callbacks for key and status events with support for classes
Stop Telephony Key Device	1	-	Telephony Device	Stops the processing if the callbacks for the specific device

C. COMMAND DESCRIPTION

C.1. General commands

These are general purpose commands, not directly targeted to a specific module. This set of commands might have to communicate with more modules over the TiproBus to get required data. It is expected that they will take more time to finish, but they are required to be called only at initialization phase and only if module specific commands are to be used later.

C.1.1. Detect devices

The command rescans the USB ports to detect if any Tipro devices supporting HID API are connected.

Syntax :

```
int HIDDetectDevices(void);
```

Return value:

Returns command status code

C.1.2. Enumerate modules

This command must be called before any commands that are directed to TiproBus modules are used. The command enumerates TiproBus modules and detects their supported command sets.

Syntax :

```
int HIDEnumerateModules (void);
```

Return value:

Returns command status code

C.2. BeFREE 15 Commands

These commands are directed to the controller inside of the BeFree module. Controller processes this commands internally, so they are generally very quickly executed.

C.2.1. Set Luminance

Sets the relative luminance of the LCD screen in 20 steps of approximately 5%. Valid range is from 0 (darkest) to 20(brightest).

Syntax :

```
int HIDSetLuminance(int nLum);
```

Parameters:

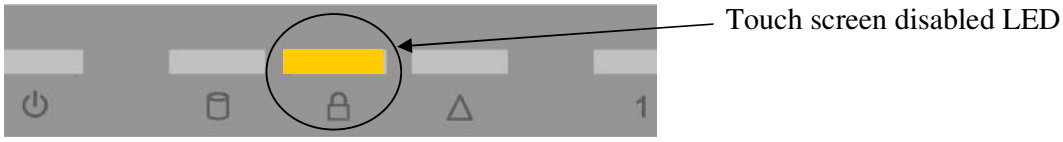
nLum Relative luminance (0-20)

Return value:

Returns command status code

C.2.2. Touchscreen disable

Disables the touchscreen. Touch screen disabled LED automatically lights up when touchscreen is disabled.



Syntax :

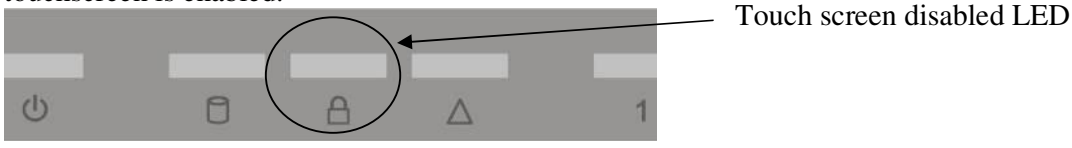
```
int HIDTouchScreenDisable(void);
```

Return value:

Returns command status code

C.2.3. Touchscreen enable

Enables the touchscreen. Touch screen disabled LED automatically turns off when touchscreen is enabled.



Syntax :

```
int HIDTouchScreenEnable(void);
```

Return value:

Returns command status code

C.2.4. Set LEDs

Sets the user defined LEDs. Each bit of the parameter corresponds to a LED on a BeFree as shown on picture below. When issuing this command the LED mode will automatically switch to user defined, overwriting any previously shown information (except error code).

Syntax :

```
int HIDSetLeds(int nLEDs);
```

Parameters:

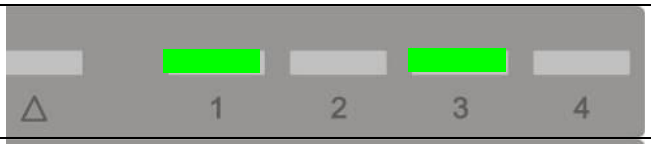
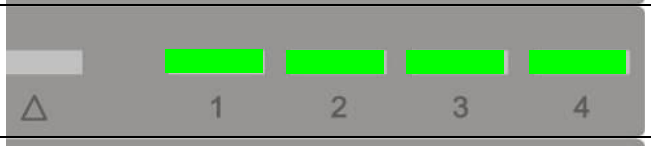
nLEDs LEDs to turn ON/OFF (0x00-0x0F)

Return value:

Returns command status code

Examples:

Sending command *HIDSetLeds(5)* would result in turning on the LED 1 and 3. Sending command *HIDSetLeds(15)* would result in turning on all LEDs. To turn off all LEDs command *HIDSetLeds(0)* should be sent.

HIDSetLeds(5)	
HIDSetLeds(15)	
HIDSetLeds(0)	

C.3. Speakerbox commands

These commands can be used to control any SpeakerBox module successfully enumerated on TiproBus. If no SpeakerBox is present the commands will return corresponding error. These commands are sent over the TiproBus and might take approximately 100ms to complete.

C.3.1. Set External Device Level

Sets the level of the connected headset/handset device.

Syntax :

```
int HIDSetExternalDeviceLevel(int nFsNum, int nLevel);
```

Parameters:

nFsNum Speakerbox module reference number looking from the left side of the configuration.

nLevel Level of the handset/headset device:

- SBX_HANDSET_HEADSET_LEVEL_9DB
- SBX_HANDSET_HEADSET_LEVEL_6DB
- SBX_HANDSET_HEADSET_LEVEL_3DB
- SBX_HANDSET_HEADSET_LEVEL_0DB
- SBX_HANDSET_HEADSET_LEVEL_M6DB
- SBX_HANDSET_HEADSET_LEVEL_M12DB
- SBX_HANDSET_HEADSET_LEVEL_M18DB
- SBX_HANDSET_HEADSET_LEVEL_M24DB
- SBX_HANDSET_HEADSET_LEVEL_M30DB
- SBX_HANDSET_HEADSET_LEVEL_M36DB
- SBX_HANDSET_HEADSET_LEVEL_M42DB
- SBX_HANDSET_HEADSET_ILLEGAL

Return value:

Returns command status code

C.3.2. Get External Device Level

Returns the the current level of the connected headset/handset device.

Syntax :

```
int HIDGetExternalDeviceLevel(int nFsNum, int *nLevel);
```

Parameters:

nFsNum Speakerbox module reference number looking from the left side of the configuration.

***nLevel**	Current level of the handset/headset device: - SBX_HANDSET_HEADSET_LEVEL_9DB - SBX_HANDSET_HEADSET_LEVEL_6DB - SBX_HANDSET_HEADSET_LEVEL_3DB - SBX_HANDSET_HEADSET_LEVEL_0DB - SBX_HANDSET_HEADSET_LEVEL_M6DB - SBX_HANDSET_HEADSET_LEVEL_M12DB - SBX_HANDSET_HEADSET_LEVEL_M18DB - SBX_HANDSET_HEADSET_LEVEL_M24DB - SBX_HANDSET_HEADSET_LEVEL_M30DB - SBX_HANDSET_HEADSET_LEVEL_M36DB - SBX_HANDSET_HEADSET_LEVEL_M42DB - SBX_HANDSET_HEADSET_ILLEGAL

Return value:

Returns command status code

C.3.3. Set Sbx LED State

Sets the LED state of the illuminated keys.

Syntax :

```
int HIDSetSbxLEDState(int nFsNum, int nKey, int nState);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nKey	Illuminated key reference number looking from the left side of the module
nState	State of the LED to set: - LED_ON - LED_OFF

Return value:

Returns command status code

C.3.4. Get Sbx LED State

Returns the LED state of the illuminated keys.

Syntax :

```
int HIDGetSbxLEDState(int nFsNum, int nKey, int
*nState);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nKey	Illuminated key reference number looking from the left side of the module
***nState**	State of the LED: - LED_ON - LED_OFF

Return value:

Returns command status code

C.3.5. Set Speaker Level

Sets the level of the handsfree speakers.

Syntax :

```
int HIDSetSpeakerLevel(int nFsNum, int nLevel);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nLevel	- Sets the speaker level to one of the predefined values (valid range is from 0 to 20)

Return value:

Returns command status code

C.3.6. Get Speaker Level

Returns the level of the handsfree speakers.

Syntax :

```
int HIDGetSpeakerLevel(int nFsNum, int *nLevel);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nLevel**	Current level of the handsfree speakers

Return value:

Returns command status code

C.3.7. Get External Headset State

Returns the state of the headset device connected to the Speakerbox module.

Syntax :

```
int HIDGetExternalHeadsetState(int nFsNum, int *nState);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nState**	State of the headset: - HEADSET_CONNECTED - HEADSET_DISCONNECTED

Return value:

Returns command status code

C.3.8. Mute Microphone

Mutes/un-mutes the microphone signal.

Syntax :

```
int HIDMuteMicrophone(int nFsNum, int nMute);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nMute	State of the microphone: - SBX_MIC_ACTIVE - SBX_MIC_MUTE

Return value:

Returns command status code

C.3.9. Set Microphone Level

Sets the level of the microphone signal (microphone sensitivity).

Syntax :

```
int HIDSetMicrophoneLevel(int nFsNum, int nLevel);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nLevel	Microphone sensitivity setting: - SBX_MIC_LEVEL_P7 - SBX_MIC_LEVEL_P6 - SBX_MIC_LEVEL_P5 - SBX_MIC_LEVEL_P4 - SBX_MIC_LEVEL_P3 - SBX_MIC_LEVEL_P2 - SBX_MIC_LEVEL_P1 - SBX_MIC_LEVEL_0 - SBX_MIC_LEVEL_M1 - SBX_MIC_LEVEL_M2 - SBX_MIC_LEVEL_M3 - SBX_MIC_LEVEL_M4 - SBX_MIC_LEVEL_M5 - SBX_MIC_LEVEL_M6 - SBX_MIC_LEVEL_M7

Return value:

Returns command status code

C.3.10. Get Microphone Level

Returns the level of the microphone signal (microphone sensitivity).

Syntax :

```
int HIDGetMicrophoneLevel(int nFsNum, int *nLevel);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nLevel**	Microphone sensitivity setting: - SBX_MIC_LEVEL_P7 - SBX_MIC_LEVEL_P6 - SBX_MIC_LEVEL_P5 - SBX_MIC_LEVEL_P4 - SBX_MIC_LEVEL_P3 - SBX_MIC_LEVEL_P2 - SBX_MIC_LEVEL_P1 - SBX_MIC_LEVEL_0 - SBX_MIC_LEVEL_M1 - SBX_MIC_LEVEL_M2 - SBX_MIC_LEVEL_M3

- SBX_MIC_LEVEL_M4
- SBX_MIC_LEVEL_M5
- SBX_MIC_LEVEL_M6
- SBX_MIC_LEVEL_M7
- SBX_MIC_LEVEL_ILLEGAL

Return value:

Returns command status code

C.3.11. Set Microphone Level Ex

Sets the level of the microphone signal (microphone sensitivity).

Syntax :

```
int HIDSetMicrophoneLevelEx(int nFsNum, int nLevel, int nDevice);
```

Parameters:

nFsNum Speakerbox module reference number looking from the left side of the configuration.

nLevel Microphone sensitivity setting:

- SBX_MIC_LEVEL_P7
- SBX_MIC_LEVEL_P6
- SBX_MIC_LEVEL_P5
- SBX_MIC_LEVEL_P4
- SBX_MIC_LEVEL_P3
- SBX_MIC_LEVEL_P2
- SBX_MIC_LEVEL_P1
- SBX_MIC_LEVEL_0
- SBX_MIC_LEVEL_M1
- SBX_MIC_LEVEL_M2
- SBX_MIC_LEVEL_M3
- SBX_MIC_LEVEL_M4
- SBX_MIC_LEVEL_M5
- SBX_MIC_LEVEL_M6
- SBX_MIC_LEVEL_M7

nDevice Device microphone to set :

- SBX_DEVICE_HANDSFREE
- SBX_DEVICE_HANDSET_HEADSET

Return value:

Returns command status code

C.3.12. Set Microphone Optimal Distance

Sets the optimal distance for the hands-free microphone.

Syntax :

```
int HIDSetMicrophoneOptimalDistance(int nFsNum, int nDistance);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nDistance	Microphone optimal distance: - SBX_MIC_DISTANCE_VERY_SHORT - SBX_MIC_DISTANCE_SHORT - SBX_MIC_DISTANCE_MEDIUM - SBX_MIC_DISTANCE_LONG

Return value:

Returns command status code

C.3.13. Get Microphone Optimal Distance

Returns the optimal distance for the hands-free microphone.

Syntax :

```
int HIDGetMicrophoneOptimalDistance(int nFsNum, int *nDistance);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nDistance**	Microphone optimal distance: - SBX_MIC_DISTANCE_VERY_SHORT - SBX_MIC_DISTANCE_SHORT - SBX_MIC_DISTANCE_MEDIUM - SBX_MIC_DISTANCE_LONG - SBX_MIC_DISTANCE_ILLEGAL

Return value:

Returns command status code

C.3.14. Set Microphone Threshold

Sets the microphone noise gate threshold.

Syntax :

```
int HIDSetMicrophoneThreshold(int nFsNum, int nThreshold);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nThreshold	Microphone noise gate threshold. - SBX_MIC_THRESHOLD_VERY_LOW - SBX_MIC_THRESHOLD_LOW - SBX_MIC_THRESHOLD_MID - SBX_MIC_THRESHOLD_HIGH

Return value:

Returns command status code

C.3.15. Get Microphone Threshold

Returns the microphone noise gate threshold.

Syntax :

```
int HIDGetMicrophoneThreshold(int nFsNum, int *nThreshold);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nThreshold**	Microphone noise gate threshold. - SBX_MIC_THRESHOLD_VERY_LOW - SBX_MIC_THRESHOLD_LOW - SBX_MIC_THRESHOLD_MID - SBX_MIC_THRESHOLD_HIGH - SBX_MIC_THRESHOLD_ILLEGAL

Return value:

Returns command status code

C.3.16. Set Microphone Compression

Sets the microphone compression.

Syntax :

```
int HIDSetMicrophoneCompression(int nFsNum, int
nCompression);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nCompression	Microphone noise gate threshold. - SBX_MIC_COMPRESSION_OFF - SBX_MIC_COMPRESSION_LOW - SBX_MIC_COMPRESSION_MID

Return value:

Returns command status code

C.3.17. Get Microphone Compression

Returns microphone compression settings.

Syntax :

```
int HIDGetMicrophoneCompression(int nFsNum, int
*nCompression);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nCompression**	Microphone noise gate threshold. - SBX_MIC_COMPRESSION_OFF - SBX_MIC_COMPRESSION_LOW - SBX_MIC_COMPRESSION_MID - SBX_MIC_COMPRESSION_ILLEGAL

Return value:

Returns command status code

C.3.18. Set Microphone Threshold Ex

Sets the microphone noise gate threshold for specific device.

Syntax :

```
int HIDSetMicrophoneThresholdEx(int nFsNum, int
nThreshold, int nDevice);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nThreshold	Microphone noise gate threshold. - SBX_MIC_THRESHOLD_VERY_LOW - SBX_MIC_THRESHOLD_LOW - SBX_MIC_THRESHOLD_MID - SBX_MIC_THRESHOLD_HIGH
nDevice	Device microphone for which threshold will be set : - SBX_DEVICE_HANDSFREE - SBX_DEVICE_HANDSET_HEADSET

Return value:

Returns command status code

C.3.19. Set Microphone Compression Ex

Sets the microphone compression for specific device.

Syntax :

```
int HIDSetMicrophoneCompressionEx(int nFsNum, int nCompression, int nDevice);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nCompression	Microphone noise gate threshold. - SBX_MIC_COMPRESSION_OFF - SBX_MIC_COMPRESSION_LOW - SBX_MIC_COMPRESSION_MID
nDevice	Device microphone for which compression will be set : - SBX_DEVICE_HANDSFREE - SBX_DEVICE_HANDSET_HEADSET

Return value:

Returns command status code

C.3.20. Speakerbox Go Online

Puts speakerbox to on-line mode.

Syntax :

```
int HIDSpeakerboxGoOnLine(int nFsNum);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.

Return value:

Returns command status code

C.3.21. Is Speakerbox Online

Returns if the Speakerbox is either in on-line or off-line mode.

Syntax :

```
int HIDIsSpeakerboxOnline (int nFsNum, int *nOnline);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nOnline**	If speakerbox is in on-line mode 1 is returned, 0 otherwise.

Return value:

Returns command status code

C.3.22. Set Alert Line

Sets the function (mode) of the alert audio channel. This function only works if the speakerbox is put to on-line mode.

Syntax :

```
int HIDSetAlertLine(int nFsNum, int nMode);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nMode	Mode of the alert audio channel: - SBX_ALERT_AS_ALERT (used as ALERT) - SBX_ALERT_AS_VOICE (used as VOICE)

Return value:

Returns command status code

C.3.23. Get Alert Line

Returns the function (mode) of the alert audio channel. This function only works if the speakerbox is put to on-line mode.

Syntax :

```
int HIDGetAlertLine(int nFsNum, int *nMode);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
***nMode**	Mode of the alert audio channel: - SBX_ALERT_AS_ALERT (used as ALERT) - SBX_ALERT_AS_VOICE (used as VOICE) - SBX_ALERT_ILLEGAL

Return value:

Returns command status code

C.3.24. Set Active Device

Sets the device where both speaker and microphone signals will be switch to. This function only works if the speakerbox is put to on-line mode.

Syntax :

```
int HIDSetActiveDevice (int nFsNum, int nDevice);
```

Parameters:

nFsNum	Speakerbox module reference number looking from the left side of the configuration.
nDevice	Device to switch to: - SBX_DEVICE_HANDSFREE - SBX_DEVICE_HANDSET - SBX_DEVICE_HEADSET

Return value:

Returns command status code

C.4. BeFREE 10 Commands

These commands can be used to control BF10 operation. Both controller module and audio module are supported. Speakerbox commands can be used for backwards compatibility where functionality allows it. If BF10 module is not present the commands will return corresponding error.

C.4.1. BF10 Set Luminance

Sets the luminance for both stripe LED and LCD backlight.

Syntax :

```
int HIDBF10SetLuminance (int nIndex);
```

Parameters:

nIndex	Luminance index (0 – darkest, 10 – brightest).

Return value:

Returns command status code

C.4.2. BF10 Touchscreen disable

Disables touchscreen.

Syntax :

```
int HIDBF10TouchscreenDisable (void);
```

Return value:

Returns command status code

C.4.3. BF10 Touchscreen enable

Enables touchscreen.

Syntax :

```
int HIDBF10TouchscreenEnable (void);
```

Return value:

Returns command status code

C.4.4. BF10 Set Speaker Level

Sets the level of the BF10 speakers.

Syntax :

```
int HIDBF10SetSpeakerLevel(int nLevel);
```

Parameters:

nLevel	Sets the speaker level to one of the predefined values (valid range is from 0 to 20)

Return value:

Returns command status code

C.4.5. BF10 Get Speaker Level

Returns the level of the BF10 speakers.

Syntax :

```
int HIDBF10GetSpeakerLevel(int *nLevel);
```

Parameters:

***nLevel**	Current level of the BF10 speakers

Return value:

Returns command status code

C.4.6. BF10 Mute Microphone

Mutes/un-mutes the microphone signal.

Syntax :

```
int HIDBF10MuteMicrophone(int nMute);
```

Parameters:

nMute	State of the microphone: - SBX_MIC_ACTIVE - SBX_MIC_MUTE

Return value:

Returns command status code

C.4.7. BF10 Set Microphone Level

Sets the level of the microphone signal (microphone sensitivity).

Syntax :

```
int HIDBF10SetMicrophoneLevel(int nLevel);
```

Parameters:

nLevel Microphone sensitivity setting:

- SBX_MIC_LEVEL_P7
- SBX_MIC_LEVEL_P6
- SBX_MIC_LEVEL_P5
- SBX_MIC_LEVEL_P4
- SBX_MIC_LEVEL_P3
- SBX_MIC_LEVEL_P2
- SBX_MIC_LEVEL_P1
- SBX_MIC_LEVEL_0
- SBX_MIC_LEVEL_M1
- SBX_MIC_LEVEL_M2
- SBX_MIC_LEVEL_M3
- SBX_MIC_LEVEL_M4
- SBX_MIC_LEVEL_M5
- SBX_MIC_LEVEL_M6
- SBX_MIC_LEVEL_M7

Return value:

Returns command status code

C.4.8. BF10 Get Microphone Level

Returns the level of the microphone signal (microphone sensitivity).

Syntax :

```
int HIDBF10GetMicrophoneLevel(int *nLevel);
```

Parameters:

***nLevel** Microphone sensitivity setting:

- SBX_MIC_LEVEL_P7
- SBX_MIC_LEVEL_P6
- SBX_MIC_LEVEL_P5
- SBX_MIC_LEVEL_P4
- SBX_MIC_LEVEL_P3
- SBX_MIC_LEVEL_P2
- SBX_MIC_LEVEL_P1
- SBX_MIC_LEVEL_0
- SBX_MIC_LEVEL_M1
- SBX_MIC_LEVEL_M2
- SBX_MIC_LEVEL_M3
- SBX_MIC_LEVEL_M4
- SBX_MIC_LEVEL_M5

- SBX_MIC_LEVEL_M6
- SBX_MIC_LEVEL_M7
- SBX_MIC_LEVEL_ILLEGAL

Return value:

Returns command status code

C.4.9. BF10 Set Microphone Optimal Distance

Sets the optimal distance for the hands-free microphone.

Syntax :

```
int HIDBF10SetMicrophoneOptimalDistance(int nDistance);
```

Parameters:

nDistance Microphone optimal distance:
- SBX_MIC_DISTANCE_VERY_SHORT
- SBX_MIC_DISTANCE_SHORT
- SBX_MIC_DISTANCE_MEDIUM
- SBX_MIC_DISTANCE_LONG

Return value:

Returns command status code

C.4.10. BF10 Get Microphone Optimal Distance

Returns the optimal distance for the hands-free microphone.

Syntax :

```
int HIDBF10GetMicrophoneOptimalDistance(int *nDistance);
```

Parameters:

***nDistance** Microphone optimal distance:
- SBX_MIC_DISTANCE_VERY_SHORT
- SBX_MIC_DISTANCE_SHORT
- SBX_MIC_DISTANCE_MEDIUM
- SBX_MIC_DISTANCE_LONG
- SBX_MIC_DISTANCE_ILLEGAL

Return value:

Returns command status code

C.4.11. BF10 Set Microphone Threshold

Sets the microphone noise gate threshold.

Syntax :

```
int HIDBF10SetMicrophoneThreshold(int nThreshold);
```

Parameters:

nThreshold	Microphone noise gate threshold. - SBX_MIC_THRESHOLD_VERY_LOW - SBX_MIC_THRESHOLD_LOW - SBX_MIC_THRESHOLD_MID - SBX_MIC_THRESHOLD_HIGH

Return value:

Returns command status code

C.4.12. BF10 Get Microphone Threshold

Returns the microphone noise gate threshold.

Syntax :

```
int HIDBF10GetMicrophoneThreshold(int *nThreshold);
```

Parameters:

***nThreshold**	Microphone noise gate threshold. - SBX_MIC_THRESHOLD_VERY_LOW - SBX_MIC_THRESHOLD_LOW - SBX_MIC_THRESHOLD_MID - SBX_MIC_THRESHOLD_HIGH - SBX_MIC_THRESHOLD_ILLEGAL

Return value:

Returns command status code

C.4.13. BF10 Set Microphone Compression

Sets the microphone compression.

Syntax :

```
int HIDBF10SetMicrophoneCompression(int nCompression);
```

Parameters:

nCompression Microphone noise gate threshold.
- SBX_MIC_COMPRESSION_OFF
- SBX_MIC_COMPRESSION_LOW
- SBX_MIC_COMPRESSION_MID

Return value:

Returns command status code

C.4.14. BF10 Get Microphone Compression

Returns microphone compression settings.

Syntax :

```
int HIDBF10GetMicrophoneCompression(int *nCompression);
```

Parameters:

***nCompression** Microphone noise gate threshold.
- SBX_MIC_COMPRESSION_OFF
- SBX_MIC_COMPRESSION_LOW
- SBX_MIC_COMPRESSION_MID
- SBX_MIC_COMPRESSION_ILLEGAL

Return value:

Returns command status code

C.4.15. BF10 Set PTT Key LED State

Sets the LED state of the illuminated PTT key.

Syntax :

```
int HIDBF10SetPTTKeyLEDState (int nState);
```

Parameters:

nState State/function of LED:
- PTT_LED_OFF
- PTT_LED_ON
- PTT_LED_BLINK

Return value:

Returns command status code

C.4.16. BF10 Set PTT Key LED State Ex

Sets the LED state of the illuminated PTT key and defines additional blink paramters.

Syntax :

```
    int   HIDBF10SetPTTKeyLEDStateEx   (int   nState,   int
nTBlinkOn, int nTBlinkOff);
```

Parameters:

nState	State/function of LED: - PTT_LED_OFF - PTT_LED_ON - PTT_LED_BLINK
nTBlinkOn	ON period for blink mode in ms.
nTBlinkOff	OFF period for blink mode in ms

Return value:

Returns command status code

C.4.17. BF10 Get PTT Key LED State

Return the LED state of the illuminated PTT key.

Syntax :

```
    int HIDBF10GetPTTKeyLEDState (int *nState);
```

Parameters:

***nState**	State/function of LED: - PTT_LED_OFF - PTT_LED_ON - PTT_LED_BLINK

Return value:

Returns command status code

C.4.18. BF10 Set Analog Audio

Sets the analog audio options.

Syntax :

```
int HIDBF10SetAnalogAudio (int nSettings);
```

Parameters:

nSettings Analog channel that is added to the output:
- SBX_ANALOG_AUDIO_OFF
- SBX_ANALOG_AUDIO_LEFT
- SBX_ANALOG_AUDIO_RIGHT
- SBX_ANALOG_AUDIO_BOTH

Return value:

Returns command status code

C.4.19. BF10 Get Analog Audio

Returns analog audio options.

Syntax :

```
int HIDBF10GetAnalogAudio (int *nSettings);
```

Parameters:

***nSettings** Analog channel that is added to the output:
- SBX_ANALOG_AUDIO_OFF
- SBX_ANALOG_AUDIO_LEFT
- SBX_ANALOG_AUDIO_RIGHT
- SBX_ANALOG_AUDIO_BOTH

Return value:

Returns command status code

C.5. Telephony Commands

These commands can be used to communicate with the telephony enabled modules. There could be more telephony modules connected to the same configuration / PC.

C.5.1. Detect Telephony Devices

Scans all USB ports and updates the list of connected telephony devices. This function should be called before using other telephony commands or when new device is connected to the system.

Syntax :

```
int HIDDetectTelephonyDevices (void);
```

Return value:

Returns command status code

C.5.2. Get Num Of Detected Telephony Devices

Returns the number of connected telephony devices.

Syntax :

```
int HIDGetNumOfDetectedTelephonyDevices (void);
```

Return value:

Returns number of connected telephony devices.

C.5.3. Get Telephony Device Product String

Returns telephony device product string.

Syntax :

```
int HIDGetTelephonyDeviceProductString(USHORT hidIx,
PVOID pString, ULONG lBufferLen);
```

Parameters:

hidIx	Telephony device index.
pString	Pointer to user defined buffer where the string will be returned to.
lBufferLen	Length of a supplied buffer.

Return value:

Returns command status code

C.5.4. Get Telephony Device VID PID

Returns telephony device vendor ID and product ID.

Syntax :

```
int HIDGetTelephonyDevice_VID_PID(USHORT hidIx, USHORT *nVID,
USHORT *nPID);
```

Parameters:

hidIx	Telephony device index.
nVID	Vendor ID.

nPID	Product ID.

Return value:

Returns command status code

C.5.5. Get Telephony Device Manufacturer String

Returns telephony device manufacturer string

Syntax :

`int` HIDGetTelephonyDeviceManufacturerString(USHORT hidIx, PVOID pString, ULONG lBufferLen);

Parameters:

hidIx	Telephony device index.
pString	Pointer to user defined buffer where the string will be returned to.
lBufferLen	Length of a supplied buffer.

Return value:

Returns command status code

C.5.6. Get Telephony Device Path

Returns telephony device path that can be used to access the device directly.

Syntax :

`int` HIDGetTelephonyDevicePath(USHORT hidIx, PVOID pString, ULONG lBufferLen);

Parameters:

hidIx	Telephony device index.
pString	Pointer to user defined buffer where the path will be returned to.
lBufferLen	Length of a supplied buffer.

Return value:

Returns command status code

C.5.7. Register Telephony Callbacks

Register callback functions that will be called when status of the selected telephony device has changed or event has been send over the telephony interface.

Syntax :

`int` HIDRegisterTelephonyCallbacks(USHORT hidIx, tfnCallback fnKeyCallback, tfnThreadStatus fnKeyStatusCallback);

Parameters:

hidIx	Telephony device index.
fnKeyCallback	Pointer to function that will be called when a new key with telephony content is captured.
fnKeyStatusCallback	Pointer to function that will be called when new status is captured.

Return value:

Returns command status code

tfnCallback callback function prototype:

`int` fnKeyCallbackEx (USHORT nID,USHORT nUsagePage ,USHORT nLinkUsage ,USHORT nUsage ,bool bPressed);

Callback function Parameters:

nID	ID of the telephony device (supporting more then one telephony devices on the same PC).
nUsagePage	Top collection usage page.
nLinkUsage	Usage of the specific link collection - 0x01 = Phone - 0x06 = Key Pad - 0x07 = Programmable Button
nUsage	Usage at the specific collection.
bPressed	Key pressed = TRUE or released = FALSE

tfnThreadStatus callback function prototype:

`int` fnKeyStatusCallback (USHORT nID, int nStatus, int nErrCode, int nRFUParam);

Callback function Parameters:

nID	ID of the telephony device (supporting more then one telephony devices on the same PC).
nStatus	Status of the thread.

nErrCode	Error code for reported status.
nRFUParam	Reserved for future use (not used in current version).

C.5.8. Register Telephony Callbacks Ex

Register callback functions that will be called when status of the selected telephony device has changed or event has been send over the telephony interface. This function allows callback to be used inside of the class by using std:bind. Visual Studio 2010 or later must be used in order to use this function, otherwise *HIDRegisterTelephonyCallbacks* should be used.

Syntax :

`int` HIDRegisterTelephonyCallbacksEx(USHORT hidIx, tfnCallbackEx fnKeyCallback, tfnThreadStatusEx fnKeyStatusCallback);

Parameters:

hidIx	Telephony device index.
fnKeyCallback	Pointer to function that will be called when a new key with telephony content is captured.
fnKeyStatusCallback	Pointer to function that will be called when new status is captured.

Return value:

Returns command status code

Example how to register callback function that is part of a class:

```
using namespace std::placeholders;
HIDRegisterTelephonyKeysCallbackEx(
    0,
    std::bind(&CExampleDlg::OnTelKeyPress,this,_1,_2,_3,_4,_5),
    NULL);
```

tfnCallbackEx callback function prototype:

`int` fnKeyCallbackEx (USHORT nID,USHORT nUsagePage ,USHORT nLinkUsage ,USHORT nUsage ,bool bPressed);

Callback function Parameters:

nID	ID of the telephony device (supporting more then one telephony devices on the same PC).

nUsagePage	Top collection usage page.
nLinkUsage	Usage of the specific link collection - 0x01 = Phone - 0x06 = Key Pad - 0x07 = Programmable Button
nUsage	Usage at the specific collection.
bPressed	Key pressed = TRUE or released = FALSE

tfnThreadStatusEx callback function prototype:

`int` fnKeyStatusCallback (USHORT nID, int nStatus, int nErrCode, int nRFUParam);

Callback function Parameters:

nID	ID of the telephony device (supporting more then one telephony devices on the same PC).
nStatus	Status of the thread.
nErrCode	Error code for reported status.
nRFUParam	Reserved for future use (not used in current version).

C.5.9. Stop Telephony Key Device

Stops background thread that catches and reports keys and status of telephony device.

Syntax :

`int` HIDStopTelephonyKeyDevice(USHORT hidIx);

Parameters:

hidIx	Telephony device index.

Return value:

Returns command status code

D. STATUS CODES

D.1. Common Status Codes

- API_ERR_None: **00**h

Everything is OK.

- API_ERR_WrongParameter: **01**h

Parameter out of valid range.

- API_ERR_InvalidHandleValue: **02**h

Nonexistent handle to a HID device.

- API_ERR_DeviceNotOpened: **03**h

Connection to HID device is not opened.

- API_ERR_GetFeature: **04**h

Communication error at GetFeature stage.

- API_ERR_SetFeature: **05**h

Communication error at SetFeature stage.

- API_ERR_Unknown: **06**h

Error appeared on lower level.

- API_ERR_CmdNotSupportedAtFWLevel: **07**h

This command is not supported in current firmware version.

- API_ERR_FWLevelNotAvailable: **08**h

Firmware doesn't support HID communication features required by the command.

- API_ERR_FWLevelChangeFailed: **09**h

Firmware doesn't support HID communication features.

- API_ERR_EnumerateModulesFailed: **10**h

Error during the TiproBus modules enumeration.

- API_ERR_ModulesNotEnumerated: **11**h

Enumeraion of TiproBus modules was not performed. *EnumerateModules* command must be called before any module specific command is used.

- API_ERR_NonexistantModule: **12**h

TiproBus module does not exist.

- API_ERR_CommunicationError: **20**h

Error in communication between controller and module.

- API_ERR_ResponseError: **21**h

Wrong response from module.

- API_ERR_UnexpectedResponse: **22**h

Unexpected response from module.

- API_ERR_UnsupportedParameter: **23**h

Parameter is not supported by the connected hardware (firmware).

- API_ERR_UnsupportedCommand: **24**h

Command is not supported by the connected hardware (firmware)

- API_ERR_NoStringDescriptor: **30**h

String descriptor was not reported or doesn't exist.

- API_ERR_VID_PID_NotReported: **31**h

VendorID and/or ProductID were not reported or don't exist.

- API_ERR_SBXMode: **40**h

Command returning this error requires Speakerbox to be in On-Line mode.

- API_ERR_GeneralLowLevelError: **E0**h

Error at lower level.

- API_ERR_NoDevicesDetected: **F1**h

No Tipro devices that support HID communication were detected.

- API_ERR_DeviceUnplugged: **F2**h

Tipro HID device was unplugged.

- API_ERR_DeviceNotConnected: **F3**h

Tipro HID device is not connected.

E. REFERENCES

[1] BeFREE V3.0 (Intel ATOM based) User Manual