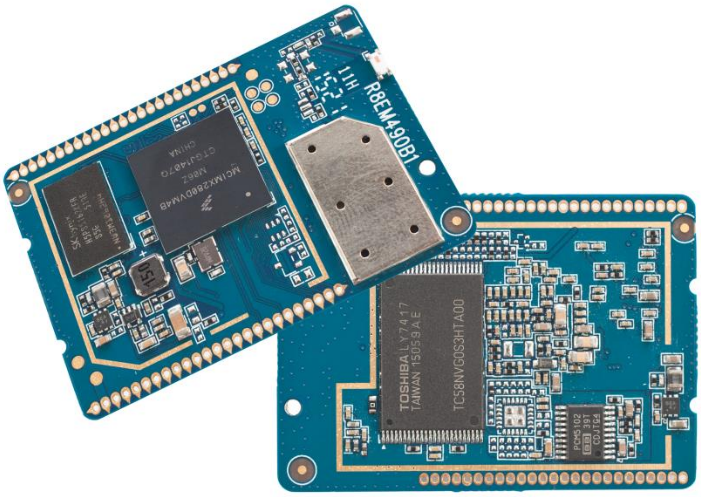
LinTech

Wireless Audio Module HBM10

Specification v4.4.3



Contents

Changes

Date	Version	Changes	Author
2017-04-18	4.4.3	Firmware 1. Fix low audio output level introduced in version 4.4.1 Documentation 1. Fix some notes about pull-up/pulld-downs in ch. 4	Jög Krause
2017-02-08	4.4.2	Firmware 1. Fix non-persistent SSID set by “setapmode” Documentation 1. Fix request and response parameters of “setdevicename” in ch. 15.1.2 2. Fix request parameter and example in ch. 15.5.14	Jörg Krause
2017-01-18	4.4.1	Firmware 1. Fix wifi after waking up from standby mode 2. Improved network performance 3. Improved playback of 24-bit high resolution audio files Documentation 1. Remove note about supported sampling rate for wifi in ch. 2 2. Fix iperf3 port number in ch. 12.1 3. Update iperf3 example in ch. 12.1	Jörg Krause
2016-12-21	4.4.0	Firmware 1. Add Spotify Connect combatibility 2. Bump to latest Linux Kernel 4.9 Documentation 1. Specify supported sample rated for wireless networks as “up to 96 kHz” in ch. 2 2. Specify power consumption in standby mode as “<25 mA” in ch. 2	Jörg Krause
2016-11-30	4.3.0	Firmware 1. Add standby mode for low power consumption 2. Improved switching between AirPlay and UPnP/Radio 3. Improved AUDIO_STATUS pin behavior to be more accurate when an PCM stream is opened and closed 4. Improved wifi network performance 5. New HTTP-API v16 • Advanced AP mode: ○ Setup an encrypted AP ○ Change SSID and channel • Support for multiple network configurations • Set a favourite playlist • Add actions “playpause”, “next”, “prev” to navigate through a playlist 6. Discard toggling between AP and station mode using pin 1 and enable it for factory reset solely 7. Fix Ethernet connection issue after factory reset or firmware update	Jörg Krause

		8. Fix missing field “name” when calling the radio “query” command after a restart 9. Fix missing download status when issueing a “querystatus” command 10. Fix I2S clock for sample rates above 96 kHz Documentation 1. Fix Audio reference schematic in ch. 6.1. 2. Add note about recommending a pull-down resistor for not connected input pins in ch. 4. 3. Fix request description for “getfavouritestation” of HTTP API endpoint radio. 4. Rename pin 1 to KEY_FACTORY_RESET in ch. 4. 5. Update AUDIO_STATUS description in ch. 7.4 6. New ch. 7.5 “Standby Mode” 7. Improved network configuration description and moved into ch. 9 8. Add notes about autostarting a favourite station or playlist in ch. 10	
2016-07-25	4.2.1	Firmware 1. Fix not going into AP mode if configured network is not found 2. Fix compatibility with “HTC Connect” 3. Fix AUDIO_STATUS (pin 8) not going to low state after power-up Documentation 1. Add notes about network configuration in ch. 9	Jörg Krause
2016-06-27	4.2.0	Firmware 1. Improved volume control and AirPlay audio playback 2. Auto disable WLAN interface if Ethernet link is detected 3. New radio playback features: • Play and navigate with the multimedia keys through a playlist • Define a favourite station • Enhanced *query* command 4. New sound features: • Get and set the master volume • Get and set the volume for the status tones 5. Improved network toggle button handler 6. Measure network bandwith with iperf3 7. Fix some network issues Documentation 1. New HTTP API v1.5 2. Add note about volume control in ch. 7.2 3. Add note about delay concerning AUDIO_STATUS_PIN in ch. 7.4 4. Add note about PWM-driven LEDs in ch. 6.3	Jörg Krause
2016-05-02	4.1.2	Firmware 1. Fix duplicated Ethernet MAC addresses 2. Fix LED blinking 3. Fix I2S BLCK	Jörg Krause
2016-04-11	4.1.1	Firmware 1. Fix streaming issue with some mp3 playlists	Jörg Krause

		2. Fix streaming issue with Windows Media Player Documentation 1. Add note about supported sampling rates of WM8804 in ch. 7.2	
2016-03-21	4.1.0	Firmware 1. Add support for setting the brightness of the LEDs 2. Improve support for MP3 audio files 3. Improve update over the air 4. Fix factory reset sent via HTTP-API 5. Fix storing volume Documentation 1. Add section 15.5.13 to HTTP API	Jörg Krause
2016-03-02	4.0.1	Firmware 1. Fix HTTP command “Radio Stop” 2. Fix an AirPlay bug, causing unnecessary resynchronizations 3. Fix an incompatibility bug with some UPnP media control points	Jörg Krause
2016-02-29	4.0.0	Firmware 1. Add support for remote control of audio playback with keys 2. Improve performance and stability 3. Feature “Serial to WifiBridge” is now optional and not enabled by default Documentation 1. Another fix for the sample request in section 15.4.1	Jörg Krause
2016-02-15	3.3.6	Firmware 1. Fix storing volume changes followed by a power-cut 2. Fix wrong AirPlay volume after changing UPnP/Radio volume Documentation 1. Fix sample request in section 15.4.1	Jörg Krause
2016-01-27	3.3.5	Firmware 1. Improve switching between different audio streams by temporarily turning AirPlay off, when UPnP or radio playback is about to begin 2. *HBM10-ETH*: Fix UPnP not available on Ethernet Documentation 1. Fix some key names and descriptions in ch. 4	Jörg Krause
2016-01-15	3.3.4	Firmware 1. Fix radio playback after network reconnection 2. Fix ”Next Track”-Bug using AirPlay on iOS 9.2 3. Disable DHCP server in network client mode	Jörg Krause
2015-12-23	3.3.3	Firmware 1. Fix radio unintentionally starting a stream after reboot although radio stream was stopped	Jörg Krause
2015-12-14	3.3.2	Firmware 1. Fix UPnP stuttering in case a Control Point becomes suddenly unreachable 2. Fix audio noise in quite passages	Jörg Krause
2015-12-01	3.3.1	Firmware 1. HTTP API v1.3 2. Expose API version in Zeroconf service description	Jörg Krause

		3. Fix UART issue outputting characters twice 4. Fix UPnP not working on Windows 5. Fix AUDIO_STATUS for internet radio 6. Improve network management for wifi + ethernet 7. Fix missing HTTP response for action setconfig, in case of changing the device name only without setting a network configuration Documentation 1. HTTP API v1.3 2. Fix current consumption values 3. Improve reference schematics sketches	
2015-10-21	3.3.0	Firmware 1. Internet radio support 2. HTTPS support Documentation 1. New HTTP API v1.2 2. Module Pin Definition: Audio KEYS are optional 3. HTTP API: Add note to 15.4.2	Jörg Krause
2015-10-19	3.1.3a	Firmware 1. Fix another power-cut issue	Jörg Krause
2015-09-30	3.1.3	Firmware 1. Fix bricking issue in case of a power-cut during boot process 2. Fix hostname issue causing trouble with some routers 3. Improve HTTPs JSON parser robustness 4. Backport support for HTTP API v1.1 5. Bump Linux Kernel to 4.1 LTS Documentation 1. Align device name description in ch. 14 to with hostname fix. 2. New ch. 9	Jörg Krause
2015-08-18	3.1.2a	Firmware 1. Append last four digits of the mac address to wifi ssid Documentation 1. Fix default values in ch. 14 and 15	Jörg Krause
2015-08-06	3.2.0	Firmware 1. HBM10-ETH with Ethernet support 2. Serial to Wi-Fi bridge Documentation 1. New chapters: Ethernet, Pin Headers, Serial to WiFi Bridge, AT Commands 2. New HTTP API v1.1 3. Several improvements all over the places	Jörg Krause
2015-07-06	3.1.2	Firmware 1. Fix clock synchronization with AirPlay 2. Pin *AUDIO_STATUS* also works now for closing UPnP connections 3. Improve compatibility with WHAALE app 4. Change default device and SSID name to “HBM10” Documentation 1. Fix wrong values for MCLK frequencies in the table of ch. 7.4	Jörg Krause

2015-05-26	3.1.1	1. Fix non-working *AUDIO_STATUS* in certain cases when streaming with UPnP	Jörg Krause
2015-05-20	3.1.0	1. Add module pin *AUDIO_STATUS* (pin #8)	Jörg Krause
2015-05-19	3.0.0	1. Enable GPIOs 2. Add an icon for UPnP device rendering	Jörg Krause
2015-05-18	2.0.1	1. Fix issue when updating firmware with the iOS app	Jörg Krause
2015-05-04	2.0.0	1. Initial version	Jörg Krause

1 Introduction

The **HBM10** is a low-cost and powerful wireless audio System-on-Module (SOM) bundled on a small 34mm x 50mm PCB. Its complete reference design drastically reduces the development time to start your own application on a custom carrier board.

The **HBM10-ETH** is additionally equipped with an 10/100 Mbps Ethernet transceiver for extended possibilities to setup your network.

The integrated and ready to use Linux-powered software stack supports audio streaming wirelessly with AirPlay and UPnP/DLNA to all kind of home audio equipment including home theater systems, A/V receivers, radios, wireless speakers and portable music players.

Furthermore, with an HBM10 you can turn your home audio system into an wireless internet radio player.

1.1 Target Applications

- Network music stations
- HiFi-systems
- Light and sound systems
- iPod docks
- Portable audio system
- Boom-boxes
- Network audio loudspeakers
- Wireless media adapters
- Complete radio and audio products

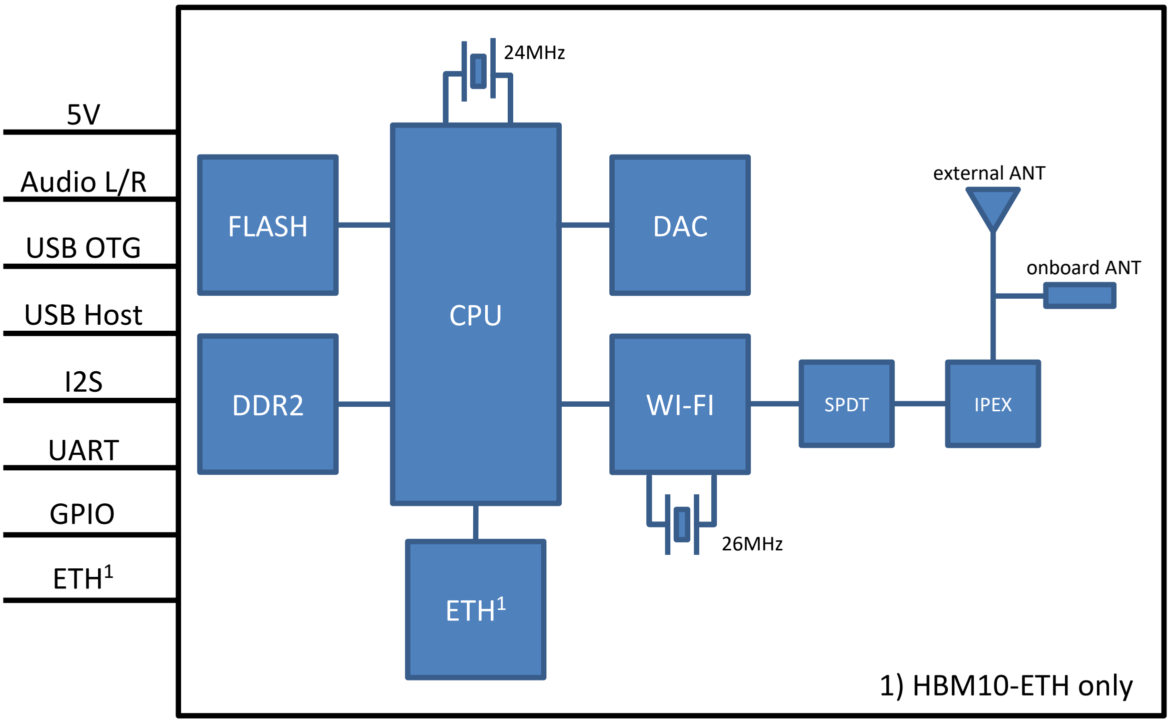
1.2 Software Features

- Linux Kernel 4.9
- Bootloader (USB Firmware recovery)
- Buildroot rootfs (pre-installed in NAND Flash)
- Audio Streaming Protocols:
    - AirPlay
    - UPnP/DLNA
    - OpenHome
    - Spotify Connect
    - HTTP
- Firmware Update over the Air
- Remote Control through HTTP API
- Internet radio player
- Audio playback control keys support
- Serial to Wifi Bridge *(optional)*

1.3 Board Features

	HBM10	HBM10-ETH
CPU	ARM9@454 MHz + Security Co-processor 128-bit AES hardware decryption	
RAM	512 Mbit DDR2	
Flash	1 Gbit NAND	
Ethernet	—	10/100 Mbps
Interfaces	I2S, I2C, UART, GPIOs	
Dimension	33.8mm x 49.5mm x 5mm	
Approvals & Certifications	CE, FCC, RoHS	

1.4 Module Block Diagram



2 System Specifications

Platform	OS	Linux 4.9
	CPU	ARM9 454 MHz + Security Co-processor 128-bit AES hardware decryption
	Wi-Fi	BCM43362
Memory	NAND FLASH	1 Gbit
	RAM	512 Mbit
Wi-Fi	Frequency Band	2.4 GHz
	Frequency Range	2.412 GHz ~ 2.484 GHz
	Channels	1 - 13
	Protocols	IEEE 802.11b IEEE 802.11g IEEE 802.11n
	Max data rates	*802.11b*: 11 Mbps *802.11g*: 54 Mbps *802.11n*: 150 Mbps
	Security	*Encryption*: None, WEP, WPA, WPA2 *Ciphers*: CCMP, TKIP
	Network Modes	Access Point Station
	Antenna	Onboard SMT antenna, 50 Ω (*default*) IPEX connector to external antenna (*optional*)
	EVM	*802.11n*: -30 dB
	Maximum transmit power	*802.11b*: 16 dBm, EVM: 28 % *802.11g*: 14 dBm, EVM: 28 % *802.11n*: 12 dBm, EVM: -30 db
	RSSI	*802.11b*: -90 dBm *802.11g*: -70 dBm *802.11n*: -70 dBm
Audio	Protocols	AirPlay, UPnP/DLNA, OpenHome, HTTP
	Formats	MP3, AAC, Vorbis, Opus, PCM, WMA, AC3, FLAC, ALAC, APE, WavPack
	Container	MP4, MKV, OGG, WAV, AIFF, ASF
	Audio Data Lengths	16 and 24 bit
	Sampling Frequency	8 to 192 kHz
	SNR	> 110dB
Interfaces	UART	1x
	USB 2.0	1x OTG 1x Host (optional)
	Audio	1x I2S 1x Line out (R, L)
	I2C	1x
	Power	*Input*: 5 V *Output*: 3.3 V

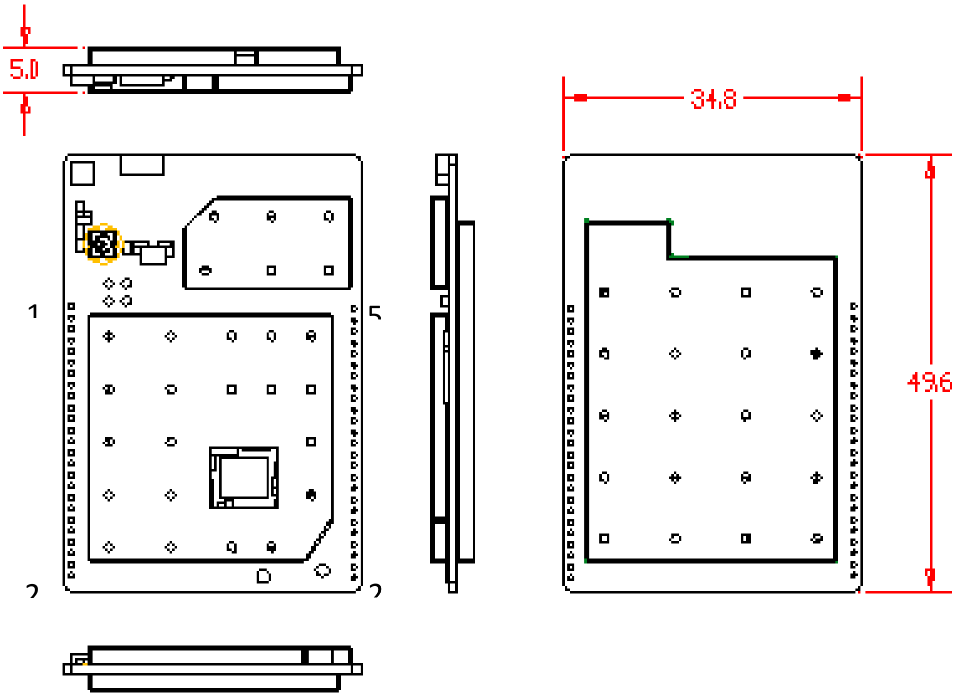
	Reset		1x
	GND		4x
	LED		2x
	KEYS		1x
	LRADC		1x
	ANT		1x
	GPIO		Up to 14x
Environment	Operation Temperature		-10 to 70 ℃
	Operation Humidity		10 to 90 %
	Storage Temperature		-40 to 100 ℃
	Storage Humidity		5 to 95 %
Performance	Boot Strap		~15 Sec
	Power Dissipation	Streaming	2.2 W
		Idle	1.9 W
	Current Consumption	Active	typ: 200 mA @ 5V, max: 800 mA @ 5 V
		Suspend	<25 mA @ 5V
Operating Condition	VDD		5 V ± 5%
	VDD_DAC		3.3 V ± 3%

3 Mechanical Specifications

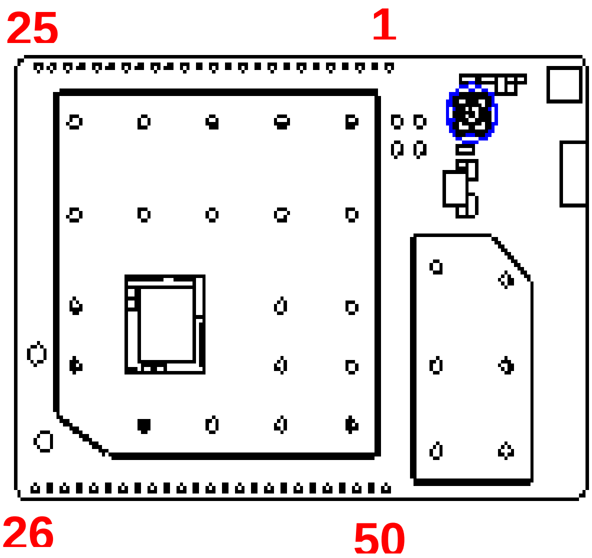
3.1 Dimension

Parameter Typical Units: 50pins

- Dimension (LxWxH): 34.8 x 49.6 x 5 mm
- Dimension tolerances: ±0.2 mm



3.2 Footprint View



4 Module Pin Definition

Pin	Name	I/O	Function	Notes
1	KEY_FACTORY_RESET	I	Factory reset	7
2	HW RESET	I	HW_RESET	
3	FEC_LED	O	ETH	4
4	GND	#	DIGITAL GROUND	
5	GPIO12	I/O	GPIO	2
6	GPIO13	I/O	GPIO	
7	SUSPEND	I	Suspend to RAM	7
8	AUDIO_STATUS	O	Audio status	6
9	KEY_VOLUME_UP	I	Key – Volume up	5
10	GPIO1	I/O	GPIO	2
11	GPIO2	I/O	GPIO	
12	GPIO3	I/O	GPIO	
13	GPIO4	I/O	GPIO	
14	GND	#	DIGITAL GROUND	
15	GPIO5	I/O	GPIO	1, 2
	LRADC	I	Low-rate ADC	1, 3
16	GPIO6	I/O	GPIO	1, 2
	UART0_TX	O	UART – Tx	1
17	GPIO7	I/O	GPIO	1, 2
	UART0_RX	I	UART – Rx	1
18	GPIO8	I/O	GPIO	1, 2
	UART0_CTS	O	UART – CTS	1
19	GPIO9	I/O	GPIO	1, 2
	UART0_RTS	I	UART – RTS	1
20	VDDIO_3V3	#	I/O voltage for GPIO	
21	LED1	O	Status LED 1	
22	LED2	O	Status LED 2	
23	GPIO10	I/O	GPIO	1, 2
	I2C_SDA	I/O	I2C – SDA	1, 3
24	GPIO11	I/O	GPIO	1, 2
	I2C_SCL	I/O	I2C – SCL	1, 3
25	GND	#	GND	
26	BATTERY	#	Battery input	3
27	USB_5V	#	USB 5V IN	
28	AGND	#	ANALOG GND	
29	LINE_OUT_R	O	Line out Right	
30	LINE_OUT_L	O	Line out Left	
31	KEY_VOLUME_DOWN	I	Key – Volume down	5
32	I2S_LRCLK	O	I2S – LRCLK	
33	I2S_MCLK	O	I2S – MCLK	
34	I2S_BCLK	O	I2S – BITCLK	
35	I2S_DOUT	O	I2S – DOUT	
36	I2S_DIN	I	I2S – DIN	
37	FEC_A3V3	#	ETH – 3V3 supply	4
38	ETH0_RXP	I	ETH – RXP	4
39	ETH0_RXN	I	ETH – RXN	4
40	ETH0_TXP	O	ETH – TXP	4

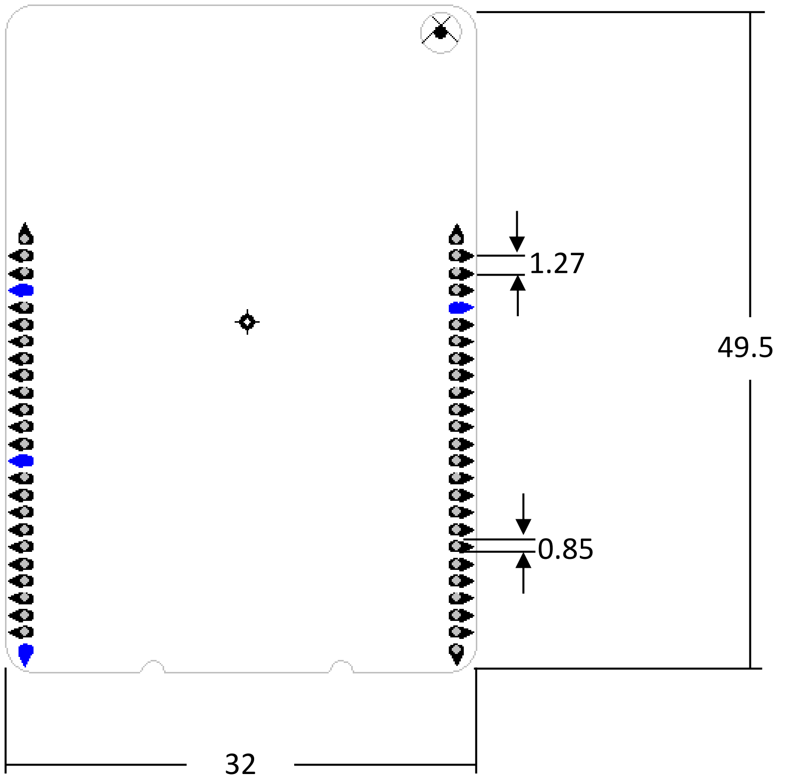
41	ETH0_TXN	O	ETH – TXN	4
42	USB0_DM	I/O	USB OTG – D-	
43	USB0_DP	I/O	USB OTG – D+	
44	USB1_DM	I/O	USB Host – D-	3
45	USB1_DP	I/O	USB Host – D+	3
46	GND	#	DIGITAL GND	
47	KEY_PLAY_PAUSE	I	Key – Play/Pause	5
48	KEY_STOP	I	Key – Stop	5
49	KEY_NEXT	I	Key – Next	5
50	KEY_PREV	I	Key – Previous	5

Notes:

1. Pins 15 – 19 and pins 23 – 24 are multiplexed
2. All GPIOs can be fully customized, see ch 10.4 Customization GPIOs,
3. Optional, not enabled by default
4. Only available on module HBM10-ETH
5. If used, a 10 kΩ pull-up resistor is recommended (see ch. 6.9). If not used, a 10 kΩ pull-down resistor is recommended instead.
6. If used, a 10 kΩ pull-down resistor is recommended.
7. If used, a 10 kΩ pull-up resistor is recommended.

5 Application Information

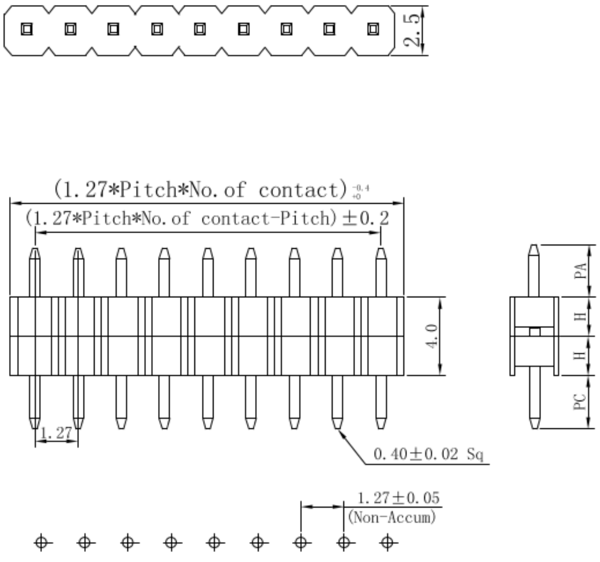
5.1 Recommended host circuit board PCB pattern



Recommended Host (customer) PCB Pattern

5.2 Pin Header

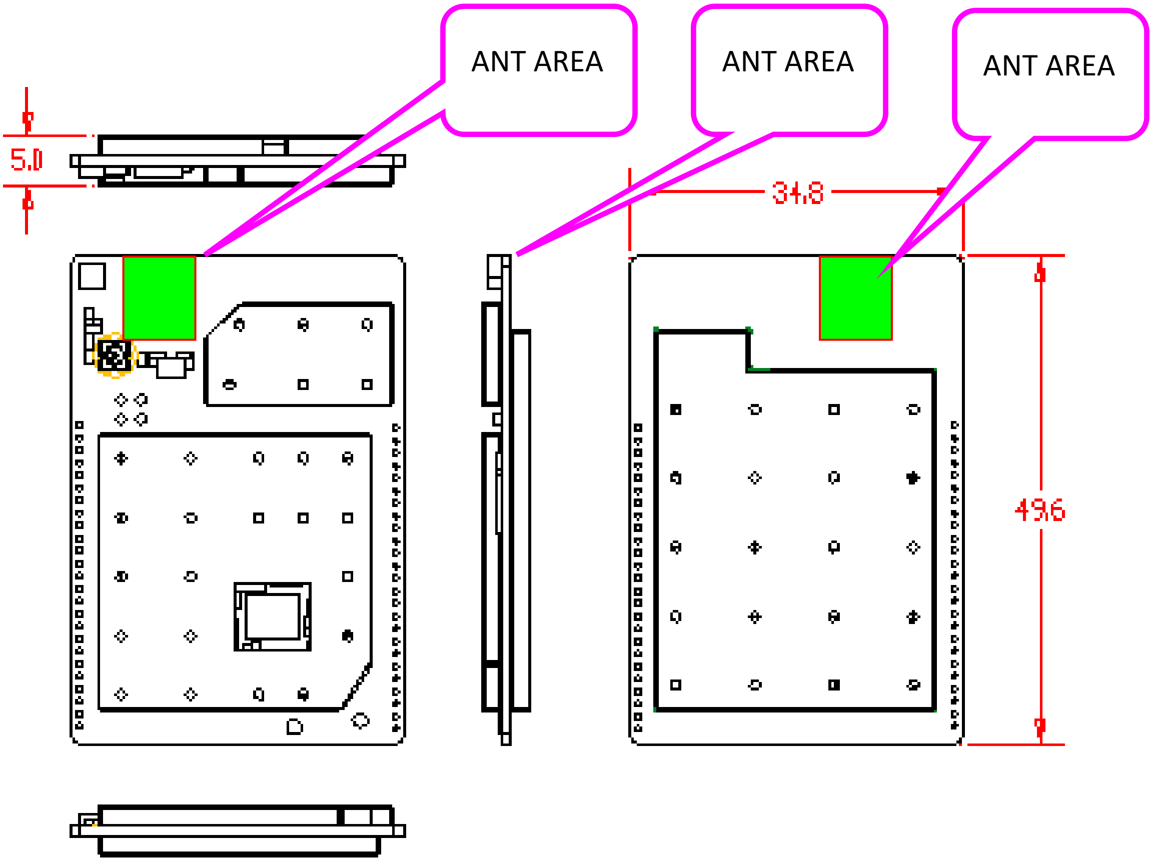
A pin header with a pitch of 1.27 mm is required.


2.5
(1.27*Pitch*No. of contact)
(1.27*Pitch*No. of contact-Pitch)±0.2
4.0
1.27
0.40±0.02 Sq
1.27±0.05
(Non-Accum)
PA
H
H
PC

Plastic Base	
Type	
H	2.0
PA	2.1
PC	1.6

5.3 Host PCB layout recommendations

The HBM10 module has an onboard antenna. Please make sure that the radio can achieve its best RF performance.



Recommended Host Circuit Board Design underneath the Module

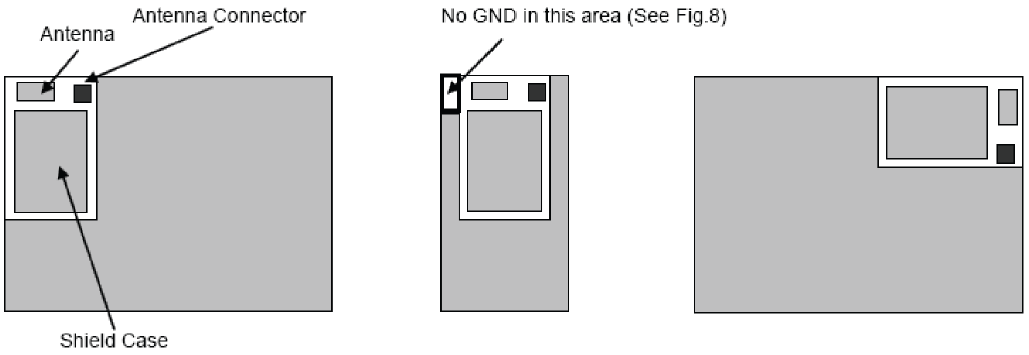
Notes

1. **Due to the surface mount antenna on the module, the green area on all layers of the customer circuit board should be free of any metal objects. Specifically, there should be no ground plane, traces or metal shield case.**
2. **The wireless signal including Wi-Fi applications is mostly affected by the surrounding environment, such as trees, and other obstacles. Metal absorbs a certain radio signal. In practical application, the data transmission distance is affected.**
3. **Please do not use metal housings.**

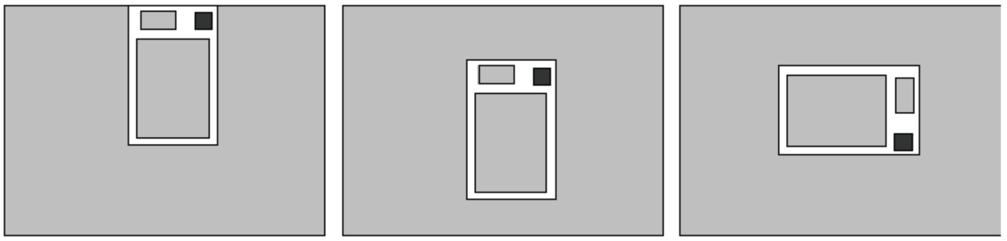
5.4 Module placement

For optimum EIRP, the customer is advised to use the recommended module placement on their host circuit board (see below).

Location in x-y plane

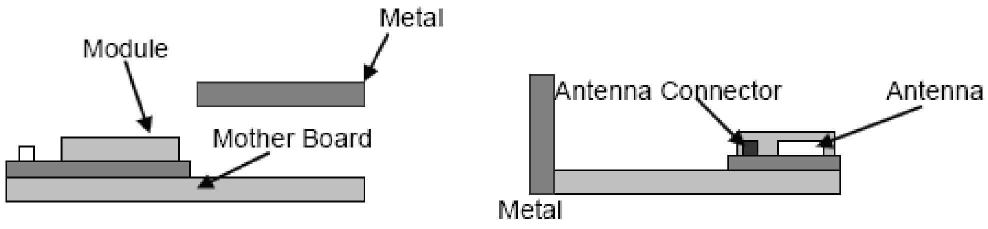


Recommended Placement in xy-plane

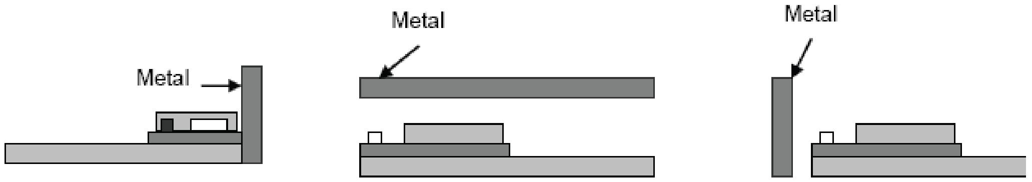


Locations Not Recommended in xy-plane

Location in z-plane



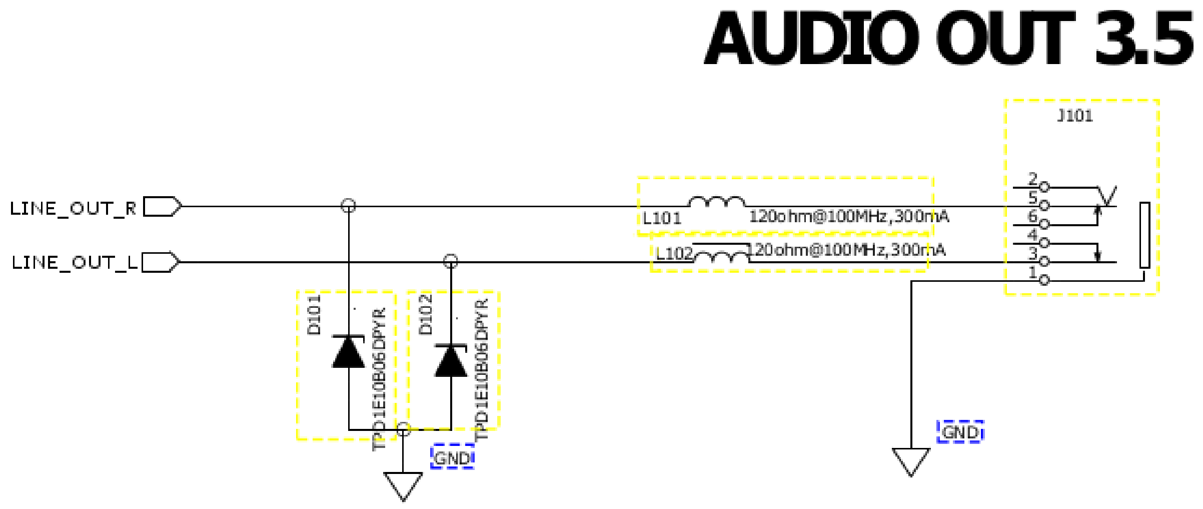
Recommended Locations in z-plane



Locations Not Recommended in xy-plane

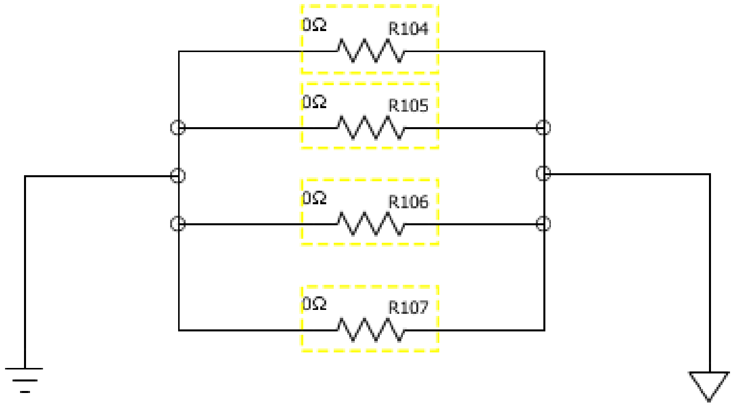
6 Reference schematic

6.1 Audio



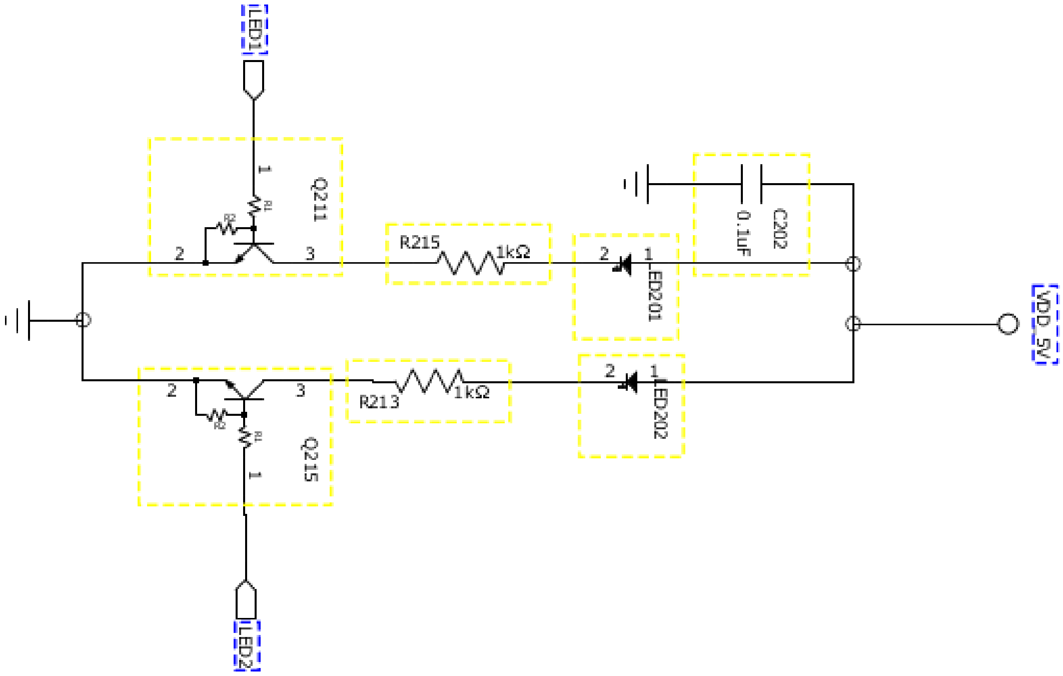
Audio out 3.5mm

6.2 Ground



AGND and GND must be at different planes

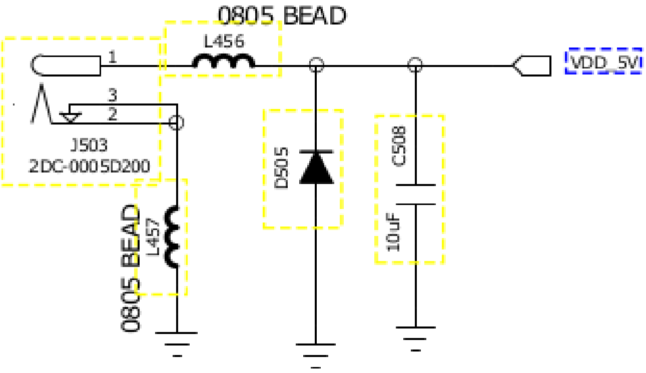
6.3 LED



LED

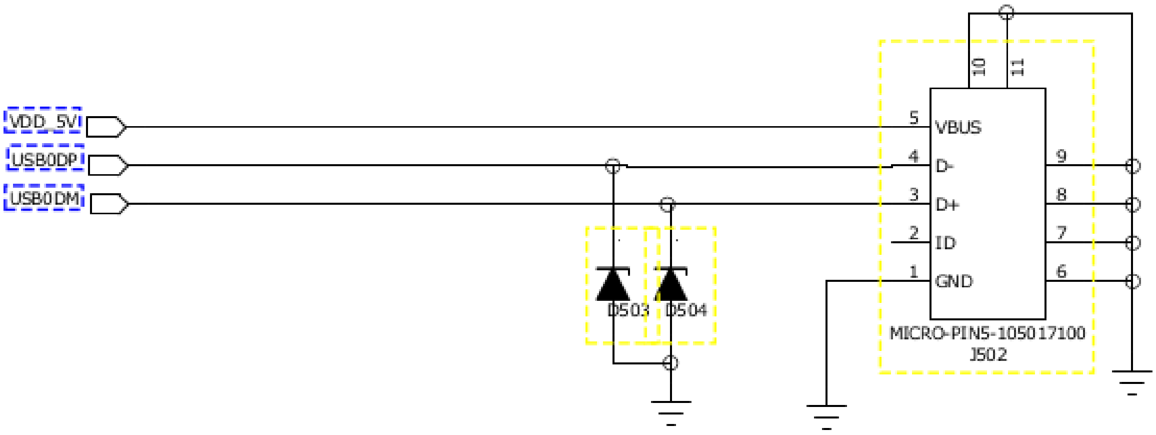
Note that the LEDs are PWM driven.

6.4 5V-Power in



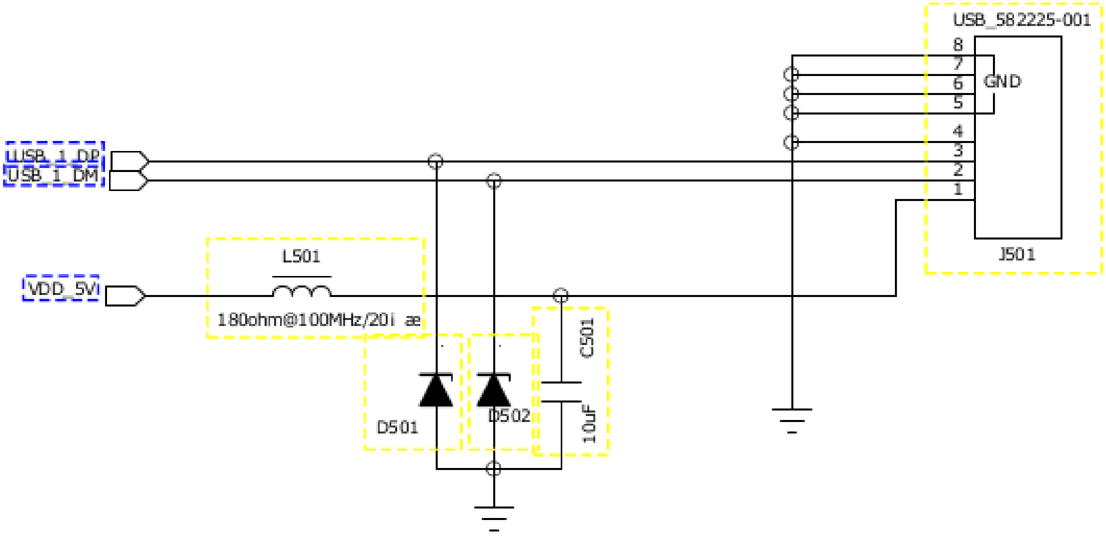
5V-Power in

6.5 USB OTG



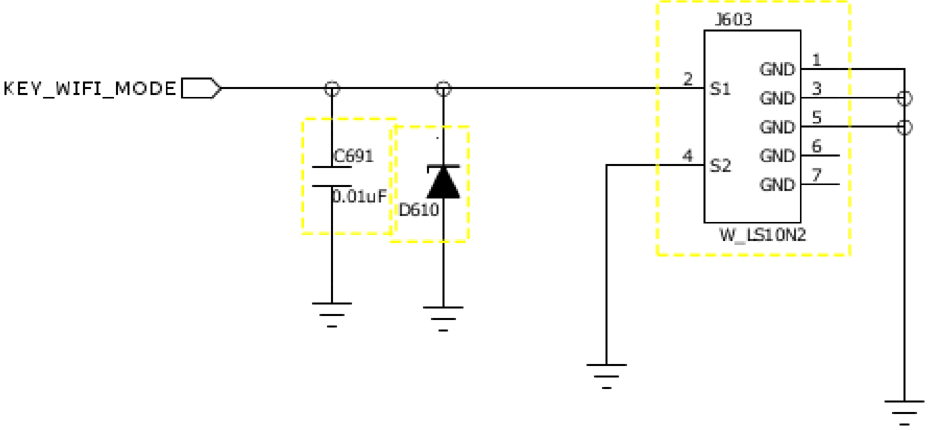
USB OTG

6.6 USB Host



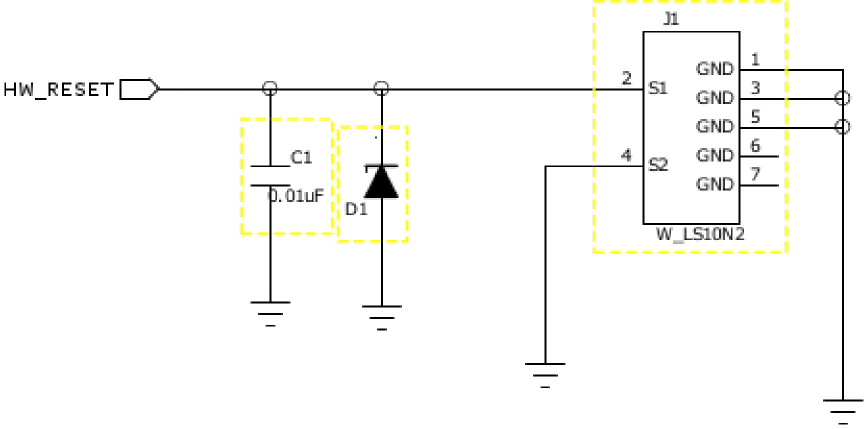
USB Host

6.7 Key WiFi-Mode



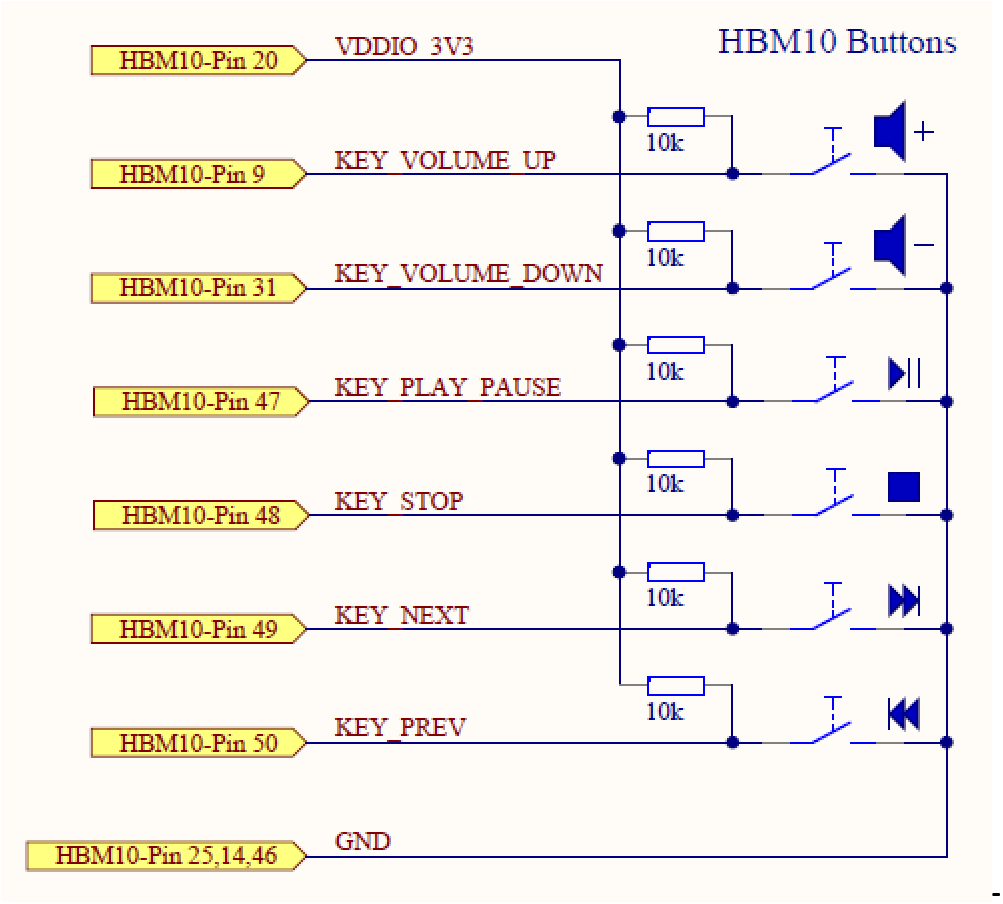
Key WiFi-Mode

6.8 HW-Reset



Key HW-Reset

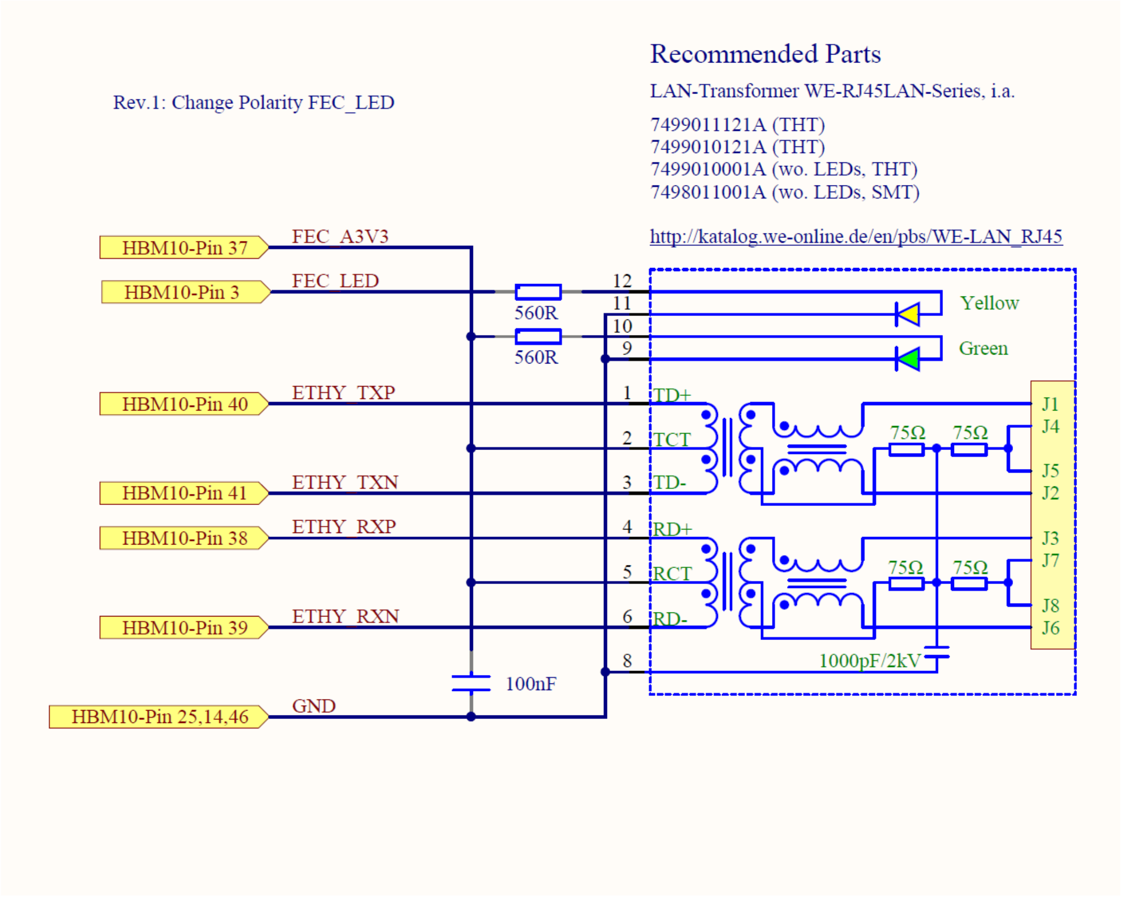
6.9 Audio Control Keys



Keys Audio Control

6.10 Ethernet

NOTE: Ethernet is only available for the HBM10-ETH module.



7 Reference design

The LinTech GmbH provides a complete reference design for using the HBM10 on a custom carrier board. The use of this reference design is the cost effective alternative to own development of your professional Hi-Fi audio applications.

Key features:

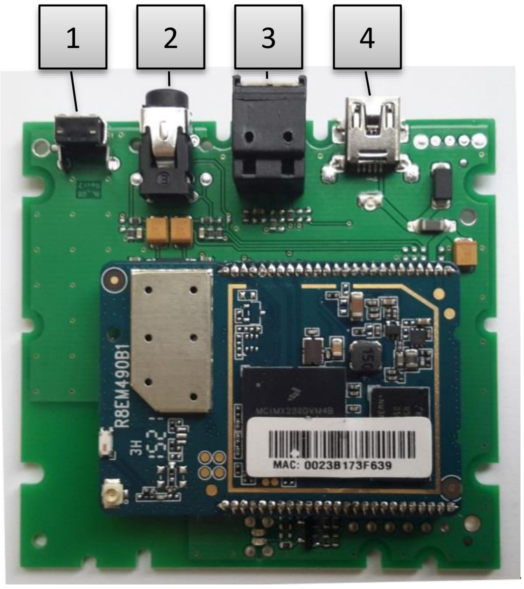
- stream music from various audio sources over WLAN
- remote control and configuration with apps

Supported audio streaming protocols:

- AirPlay
- UPnP/DLNA
- OpenHome

7.1 Hardware

	Port	I/O	Spec
1	Key	I	Factory reset (long pressed > 3s)
2	Audio Line-Out **analog**	O	Audio jack 3.5mm
3	Audio Line-Out **digital**	O	Optical S/PDIF (TOS Link)
4	Power supply 5V, 800mA	I	3,5mm DC 5V

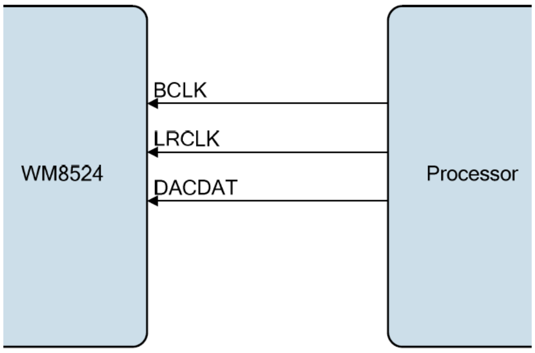


Dimensions: 81mm × 70mm

7.2 Audio Output

The reference design features a Wolfson WM8524 stereo DAC with integral charge pump and hardware control interface together with a Wolfson WM8804 S/PDIF transceiver. The analogue output level for the reference design is set to **2V$_{rms}$** typical for 0dBFS.

The Serial Audio Interface (SAIF) interface of the i.MX28 processor transmits the PCM audio data to the WM8524 and WM8804 both operating in slave mode:



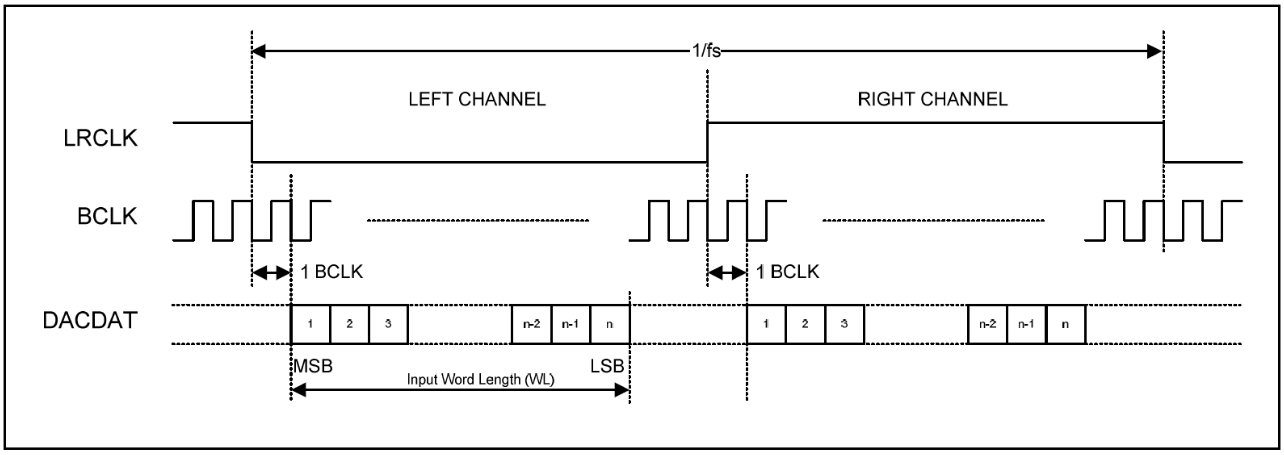
DAC operating in slave mode

I2S Audio Format

The supported interface format for PCM audio data transmission is I2S. The MSB is sent first. A word length of 24 bits is used.

Audio data for each stereo channel is clocked with the BCLK signal. Data is time multiplexed with the LRCLK, indication whether the left or right channel data is present. The LRCLK is also used as a timing reference to indicate the beginning or end of the data words. The minimum number of BCLKs per LRCLK period is two times the number of 24 bits.

The MSB of the output data changes on the first falling edge of BCLK following an LRCLK transition, and may be sampled on the next rising edge of BCLK. LRCLK is low during the left samples and high during the right samples.



I2S Audio Format

The Audio Interface supports a MCLK to LRCLK ratio of 192*fs and 384*fs and sampling rates of 8kHz to 192KHz[1]. The BCLK base rate is 48*fs.

Sampling Rate (kHz) LRCLK	MCLK (MHz)		BCLK (MHz)
	192*fs	**384*fs**	**48*fs**
8	–	3.072	0.384
32	–	12.288	1.536
44.1	–	16.9344	2.1168
48	–	18.432	2.304
88.2	–	33.8688	4.2336
96	–	36.864	4.608
176.4	33.8688	–	8.4672
192	36.864	–	9.216

MCLK Frequencies and Audio Sample Rates

Volume Control

The audio volume is controlled by a software control with a dynamic range from -57.2 to -6.2 dB and a resolution of 256 corresponding to a step of 0.2 dB.

The volume is stored periodically. However, to make sure a volume change withstands an abrupt power-cut a delay of 10 seconds is necessary.

7.3 Status LEDs

Two status LEDs are used to indicate the current device status – a blue and a red led:

Status	LED	
	Blue	**Red**
Bootloader initializes hardware	On	On
Bootloader starts Linux	On	Off

[1] The WM8804 S/PDIF transceiver supports sampling rates of 32kHz to 192KHz.

Booting Linux	Regular Double Flash	Off
Device Mode		
AP mode	Off	On
Client mode	On	Off
Connecting to network	Regular Single Flash	Off

7.4 Audio Status

Note: It is recommended to connect a 10kΩ pull-down resistor to this pin.

The pin AUDIO_STATUS (pin #8) represents the audio output status of the module:

Level	Function
Low	No audio output
High	Audio output PCM device is active

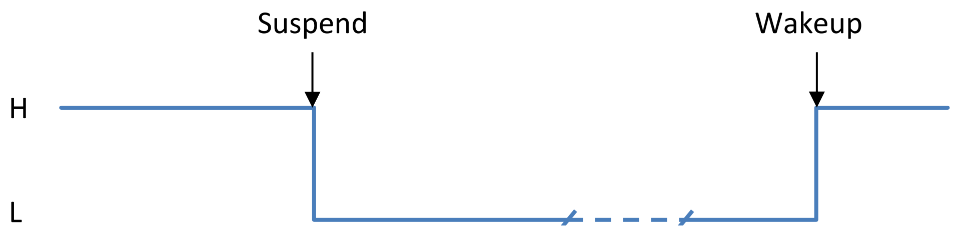
The level toggles from low to high state when a streaming application access the audio output PCM device, e.g. the “Play” button is pressed on a remote device. The level toggles from high to low when a streaming application releases the audio output PCM device, e.g. audio streaming is stopped by pressing the “Stop” button.

Note, that between switching tracks the state might went from high to low for a short period of time. Furthermore, it might take time some time after a track is finished until the state goes to low. This behavior is very dependent of the streaming application and might very between UPnP and AirPlay.

7.5 Standby mode

Note: It is recommended to connect a 10kΩ pull-up resistor to this pin.

The pin SUSPEND (pin #7) can be used to put the module into a “Standby” mode to reduce power consumption. In this mode, the CPU is placed in the Wiat-For-Interrupt (WFI) state, and the DRAM is placed in Self-Refresh mode.



Use of pin #7 to suspend and wakeup

To suspend the module a transition from high to low on pin #7 is necessary. The module will thereafter stop all streaming applications and disconnect itself from the network before shutting down the Wi-Fi and the internal DAC chip.

To wakeup the module a transition from low to high on pin #7 is necessary. The module will thereafter turn on the Wi-Fi and internal DAC chip and starts the network connection procedure. If no known network is found the module will switch to AP mode. After the network setup is done all streaming applications will be started again.

8 Remote Control

For easy remote device configuration the HBM10 module runs an unsecured HTTP server as well as a secured HTTPS server. The following table shows the network configuration:

Protocol	IP		Port
	Access Point	Network Client	
HTTP	192.168.2.1	DHCP	8989
HTTPS	192.168.2.1	DHCP	4949

For connecting to the HTTPS server, the client needs to accept the self-signed certificate. The recommended security protocol is TLS v1.2, the recommended cipher is “AES256-GCM-SHA384”. You can test the secured connection with the OpenSSL client tool, e.g:

```
openssl s_client –tlsv1_2 –cipher AES256-GCM-SHA384 –connect 192.168.2.1:4949
```

8.1 HTTP API Documentation

The HTTP API allows you to interact with a HBM10 module connected to a remote control through HTTP requests.

This documentation represents version 1.5 of the HBM10 HTTP API. Accordingly, all API endpoints will be prefixed with “`api/v16`”. Additional versions may be introduced in the future, and will be accompanied by a different prefix. To ease maintenance of customer HTTP clients HBM10’s HTTP server provides backward compatibility to previous API versions.

The HTTP server is accessed through POST requests to one of the following endpoints:

API endpoint	Description
`configure.action`	Device configuration
`upgrade.action`	Firmware update over the air
`radio.action`	Radio station playback
`leds.action`	LEDs control

Parameters has to be passed into these endpoints through the request body as a JSON object. The Appendix A includes a complete reference for all supported request.

8.2 AirLino® Configurator App

The AirLino® Configurator App is an easy to use configuration tool designed to run on iOS and Android mobile devices. Currently, the app implements the HBM10 HTTP API v1.0 to control the device remotely and can be used to perform all necessary steps to use the HBM10:

- integrate the device into an existing wireless network
- change the device name
- update the firmware over the air
- reset to factory settings
- radio station playback control

Once you have integrated your devices into an existing wireless network you are able to enjoy an unique listening experience by one or more devices at the same time - wirelessly!

How it works:

Connect your mobile device (iOS or Android) under Settings > Wi-Fi to the audio receiver network named “HBM10-xxxx”, where xxxx signifies the last for hexadecimal digits of the MAC address, (e.g. “HBM10-A54C”)

1. Start the “AirLino® Configurator” app and select the device named “HBM10”
2. If desired, you can provide the audio receiver with an individual device name or select one of the default preset options.
3. Select your home network name from the scanned network list to integrate the HBM10 module into your network. After receiving an IP address from the home networks base station the Module is now available on your wireless network.
4. You can now close the app and return with your phone back to your home network.

9 Network

9.1 Network initialization

The module can operate in access point (AP) mode or station mode. By default, if not configured, the module will boot into AP mode, except in case of the HBM10-ETH with Ethernet cable plugged-in - in that case it will boot into station mode. Without Ethernet, the module can be easily integrated into an existing network with a few HTTP requests as described in ch. 15.

Furthermore, the device will boot into station mode, if any of the configured wireless networks is found when scanning the environment and the corresponding access point accepted the authentification.

After successful authentification the internal DHCP client will request a dynamic IP, so it is necessary that a DHCP server is running on the network. When requesting the DHCP client will use a distinct hostname for identifying itself on the network. The hostname is based on the model name and the last four digits of the MAC address, e.g. HBM10-9710.

The network configuration state is indicated by the modules status LEDs. When starting the network configuration process the *Status LED 2* (pin #22) turns off and the *Status LED 1* (pin #21) begins to flash periodically. If the authentification is successful and the module receives a dynamic IP from the DHCP server, the network configuration process is assumed as successful and the *Status LED 1* will turn on permanently.

If authentification with the access point or the DHCP request fails, the network configuration process is assumed as unsuccessful and the module will turn back into AP mode. In this case the *Status LED 1* will stop flashing and the *Status LED 2* will turn on permanently.

Some of the possible reasons for an unsuccessful network connection are:

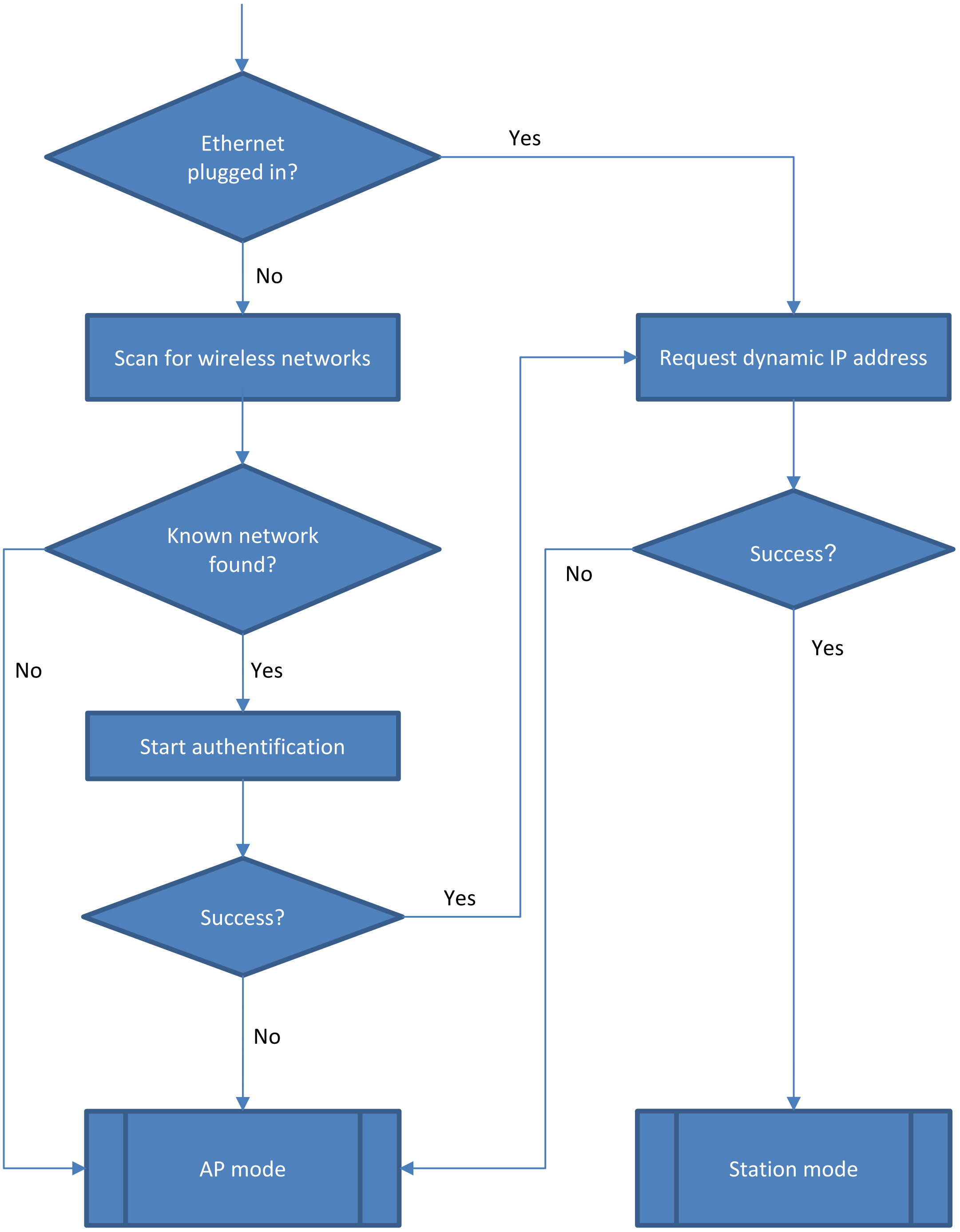
- a wrong network key
- the DHCP client does not receive an IP address within 60 seconds
- the connection to the networks access point is somehow lost

In case of a successful connection to an access point, the module is integrated into the network and can be accessed by the IP address provided by the networks DHCP. The current IP address can be easily obtained by resolving the hostname using *Zeroconf*.

Note, that if the module is powered-up, but no known network is found, the modules switches into AP mode.

Note, that if the module is powered-up and connects successfully to a network and loses the connection to the access point later, the *Status LED 1* (pin #21) turns off and the module waits for the access point to become visable again. Beware, that is might need up to 180 seconds until the module connects to the access point again.

The following charts shows the basic flow when the modules starts the network process.



Startup network procedure

9.2 Zeroconf

The module announces its HTTP service using DNS-SD[2]. The service type‘s name is _dockset._tcp. The instance name is the same as the device name. Note, if not set by the user, the device name defaults to concatenating the model name and the last for the digits of the MAC address with a hyphen, e.g. HBM10-BDDF.

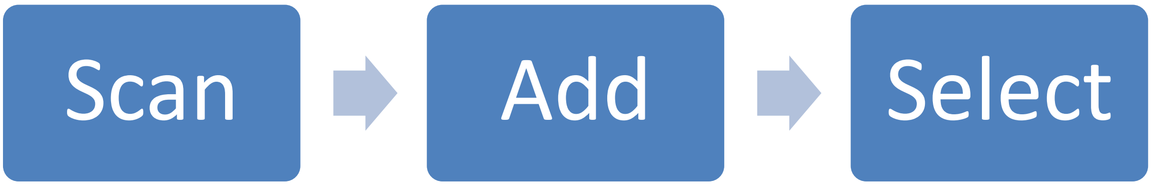
The modules response to an query with a SRV, TXT and address (type A) record. While the SRV record includes the port number of the HTTP server, the address records contains the IPv4 address of the module. The TXT records contains additional informations:

TXT record	Description	Example
api	The HTTP-API version.	v16
model	The model type.	HBM10
swver	The firmware version.	4.4.0

Using the IPv4 address, the port number and the api version string obtained from the mDNS response, the module can easily be configured using the HTTP API as described in ch. 15.

9.3 Network configuration

Setting up the wireless network only needs a few HTTP commands. All necessary commands are described in ch. 15.3. The basic workflow is as following:



The scan result contains all BSSID[3] found including additional information like authentification and encryption parameters. To connect to one of the networks from the scan results only the SSID and, in case of a WPA/WPA2 encrypted access point, the pre-shared key is required. All authentification and encryption parameters are determined automatically. After adding the SSID to the list of networks, the corresponding network can be selected by using the ID returned by „Add“.

To ease the usage of mobile devices who are changing their location often and therefore needs to handle different network setups the HBM10 allows the user to configure multiple networks. The configuration of multiple networks follows the basic workflow. Besides “Add” and “Select” the commands “List” and “Remove” can be used to manage the stored networks.

[2] Domain name service – service discovery

[3] Basic Service Set Identification

If the current network is removed, the module will perform a new scan and will search for any other enabled networks from the stored list. If any network is found, the module will associate with the access point having the best signal strength, otherwise it will switch to AP mode. If all networks are removed from the list, the module will switch to AP mode, too.

Note, that the IDs returned by the “List” command might change after issueing a “Remove” command, so it is recommend to run the “List” command afterwards.

10 Internet radio

The HBM10 module offers the ability to listen to streaming audio of radio stations worldwide.

10.1 Features

The HBM10 radio player's main features are:

- play HTTP streams encoded with MP3 or Ogg/Vorbis
- group radio stations in playlists and store them on the device
- autostart with a favourite radio station a playlist
- remote control through the HTTP API[4]

10.2 Playback

To make the HBM10 playback a radio station is very easy. You just need to request a valid URL[5] and optionally a station name with the **play** command to the HBM10 radio player. The **stop** command immediately stops the radio stream.

If the module is powered-down while it has been playing a radio station, it will restart the stream automatically the next power-up cycle.

To always start with a radio station after powering up the module, a favourite station can be defined using the **setfavouritestation**.

The current playback status can be retrieved with the **query** command.

10.3 Playlists

Radio stations can be group into playlists and are stored directly on the HBM10 module. So you have access to all your playlist even if you change the remote client.

A playlist is stored with the **saveplaylist** command. Every playlist has an unique ID in the range of 0 to 128. You can give each list a short description, e.g. “News” or “Rock”. Each playlist can contain up to 128 radio stations.

A playlist can be fetched with the **getplaylist** command. The playlist is specified by its ID. If the playlist is present, the radio player will return the corresponding playlist with its description and a list of stations.

The command **playplaylist** starts the playback of a previously stored playlist.

If the module is powered-down while it has been playing a playlist, it will restart the playlist automatically the next power-up cycle.

[4] See the Appendix for a full list and description of the provided HTTP API commands.

[5] Note that the player does not support multimedia playlists like M3U or PLS – however the appropriate URL can be extracted from such files.

To always start with a playlist after powering up the module, a favourite playlist can be defined using the **setfavouriteplaylist**.

A playlist is removed from the HBM10 with the **rmplaylist** command.

11 Audio playback control

The HBM10 module can be extended with keys to control the audio playback:

- Volume Up
- Volume Down
- Play/Pause
- Stop
- Next
- Previous

11.1 AirPlay Remote Control

Enabling remote control for AirPlay requires an AirPlay server supporting the Digital Audio Control Protocol (DACP). Once a key event has been triggered on the module, e.g. the “Stop” key was pressed by the user, an appropriate HTTP request is sent from the AirPlay client to the AirPlay server. Depending on the network connectivity it may need several dozen to hundred milliseconds until the server has received and processed the command sent by the client – for example in case of a “Stop” command, the server will end the playback stream.

11.2 UPnP Remote Control

Besides controlling the audio playback with the keys any compliant UPnP media control point, e.g. BubbleUPnP can be used to. Therefor the UPnP media control point has to be connected to the HBM10 who is acting as a UPnP media renderer.

11.3 Spotify Connect Remote Control

Remote Control of a Spotify Player is possible if the player is connected to a HBM10 device.

Once a key event has been triggered on the module, e.g. the “Next” key was pressed by the user, an appropriate command is sent from the Spotify Connect client to the Spotify Player. Depending on the network connectivity it may need up to several hundred of milliseconds until the player has received and processed the command sent by the client.

Note, that the Spotify Player does handle a “Stop” command actually as a “Pause” command.

12 Network Tools

One of the most common network problems is insufficient or unreliable bandwidth. Bandwith limitation can cause packet loss, delays, and jitters. In addition, if the required sending and receiving bit rates exceed the bandwith limitations of the network, network congestion will occur and eventually results in a poor audio experience.

12.1 iPerf3

iperf is a commonly used network testing tool to measure the bandwidth of a network.

The HBM10 module runs an iperf3 server on port 5201. Note that the server has to be enabled using the HTTP API as described in section 15.8.

Running an iperf3 client on another device in the same network allows to measure the bandwidth between the two endpoints. The example below shows the results of an iPerf3 test running between a client and the HBM10 module measuring the TCP bandwidth in an IEEE802.11n network.

```
$ iperf3 -c 192.168.1.148 -i 1 -t 10
Connecting to host 192.168.1.148, port 5201
[  4] local 192.168.1.157 port 33008 connected to 192.168.1.148 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  2.07 MBytes  17.3 Mbits/sec    0    113 KBytes
[  4]   1.00-2.00   sec  1.93 MBytes  16.2 Mbits/sec    0    189 KBytes
[  4]   2.00-3.00   sec  1.80 MBytes  15.1 Mbits/sec    0    235 KBytes
[  4]   3.00-4.00   sec  1.43 MBytes  12.0 Mbits/sec    9    185 KBytes
[  4]   4.00-5.00   sec  1.55 MBytes  13.0 Mbits/sec    0    212 KBytes
[  4]   5.00-6.00   sec  1.55 MBytes  13.0 Mbits/sec    0    225 KBytes
[  4]   6.00-7.00   sec  1.18 MBytes  9.90 Mbits/sec   20    168 KBytes
[  4]   7.00-8.00   sec  1.43 MBytes  12.0 Mbits/sec    0    184 KBytes
[  4]   8.00-9.00   sec  1.62 MBytes  13.6 Mbits/sec    0    189 KBytes
[  4]   9.00-10.00  sec  1.12 MBytes  9.38 Mbits/sec   26    151 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  15.7 MBytes  13.1 Mbits/sec   55             sender
[  4]   0.00-10.00  sec  14.5 MBytes  12.2 Mbits/sec                  receiver

iperf Done.
```

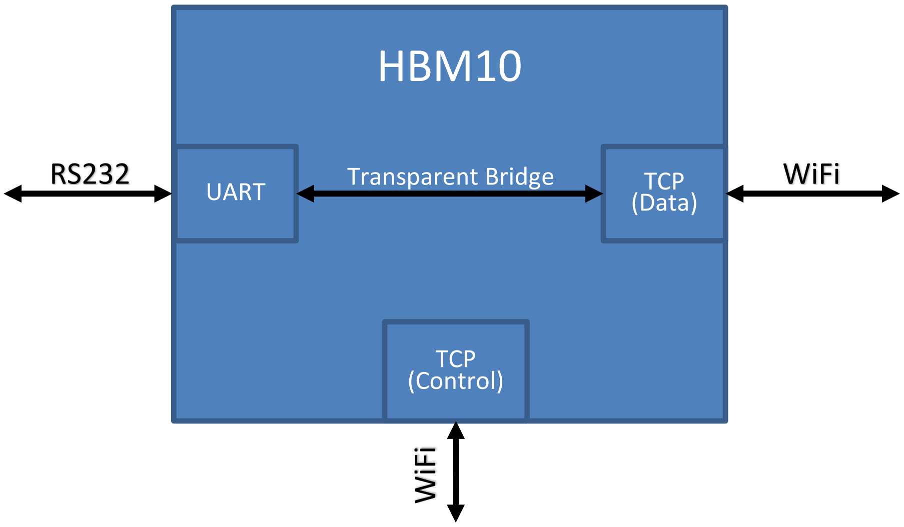
13 Serial to WiFi Bridge

NOTE: This feature is optional and not enabled by default. Please contact the LinTech support team: lintech@lintech.de.

The HBM10 module can be used as a transparent bridge to carry serial (UART) traffic over an 802.11 wireless link. AT commands as described in the AT Command Reference are used to manage the configuration.

13.1 Block Diagram

The HBM10 is running a TCP Data Server listening on port 8990 and a TCP Control Server listening on port 8991.



The Data Server transparently bridges data between the UART interface and the TCP port 8990. The data is sent to the other interface as received – no processing or formatting is done. The default setting for the UART interface is 9600 8N1.

The Control Server listens on TCP port 8991 for incoming AT commands as described in the AT Command Reference.

13.2 Workflow

The UART interface is opened and ready to user after the board powers up. If present, the UART interface uses the serial settings stored by the user otherwise the default settings for initialization.

To use the UART interface as a transparent bridge a TCP client has to establish a connection to the Data Servers port. Any data sent to the UART interface before a TCP connection is established will be lost.

If any configuration is requested, a TCP client has to establish a connection to the Control Servers port. This can be done at any point of time after the board powers up.

14 Customization

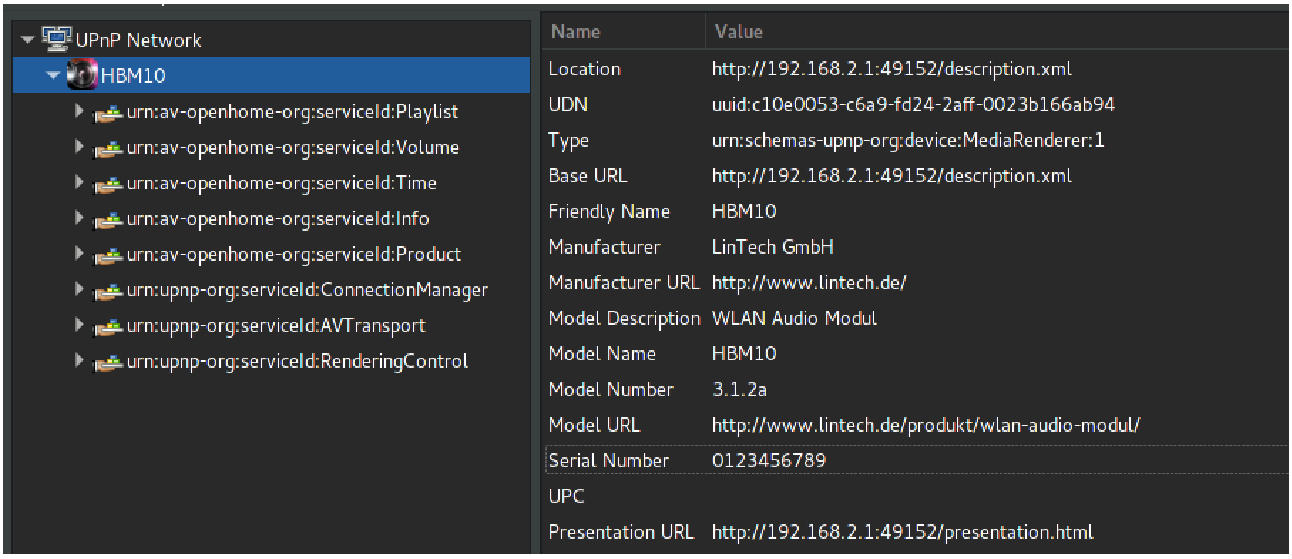
The device can be customized to represent your application. For customizing any of the values please contact the LinTech support team: lintech@lintech.de.

14.1 Device

The device parameters are mainly used for service description, e. g. by the UPnP protocol.

Parameter	Default	Description
Name	HBM10	A friendly name for identifying the device within the network.
Manufacturer		
Name	LinTech GmbH	The manufacturer name for the device.
URL	http://www.lintech.de	The manufacturer URL.
Model		
Name	HBM10	A model name for the device.
Description	WLAN Musikempfänger	A short description of the model.
URL	http://www.lintech.de/produkt/airlino-wlan-airplay-dlna-musikempfaenger/	The model URL.

These values are advertised by the device and can be viewed by most UPnP control points:



- UPnP Network
  - HBM10
    - urn:av-openhome-org:serviceId:Playlist
    - urn:av-openhome-org:serviceId:Volume
    - urn:av-openhome-org:serviceId:Time
    - urn:av-openhome-org:serviceId:Info
    - urn:av-openhome-org:serviceId:Product
    - urn:upnp-org:serviceId:ConnectionManager
    - urn:upnp-org:serviceId:AVTransport
    - urn:upnp-org:serviceId:RenderingControl

Name	Value
Location	http://192.168.2.1:49152/description.xml
UDN	uuid:c10e0053-c6a9-fd24-2aff-0023b166ab94
Type	urn:schemas-upnp-org:device:MediaRenderer:1
Base URL	http://192.168.2.1:49152/description.xml
Friendly Name	HBM10
Manufacturer	LinTech GmbH
Manufacturer URL	http://www.lintech.de/
Model Description	WLAN Audio Modul
Model Name	HBM10
Model Number	3.1.2a
Model URL	http://www.lintech.de/produkt/wlan-audio-modul/
Serial Number	0123456789
UPC	
Presentation URL	http://192.168.2.1:49152/presentation.html

14.2 Wi-Fi

Currently, the SSID (Service Set Identifier) is the only customizable Wi-Fi parameter.

Parameter	Default	Description
SSID	HBM10-XXXX	The SSID (Service Set Identifier) name used as Access Point appended with the last four digits of the MAC address, eg. HBM10-A97B.

14.3 Audio

The default status tones used to signal the boot finished and the network connection event can be turned off.

Parameter	Default	Description
Status Tones	on	Turn status tones on/off.

14.4 GPIOs

The HBM10 module offers up to 15 GPIOs. For each pin the behavior can be customized.

Parameter	Default	Description
Direction	Input	Configure the GPIO as input or output.
Value	0	The value to drive for GPIOs configured as output: • 0 = Low • 1 = High
Pull Up Resistor	Disabled	Enables/disables integrated on-chip pull up resistors.
Voltage	3.3V	Select between: • 1.8V • 3.3V
Drive Strength	Low	Select between: • Low • Medium • High
IRQ Enable	Disabled	Enable/Disable interrupts

Level/Edge Sensivity	Edge detection	Select between: - Edge detection - Level detection
Polarity	Low or falling edge	Select between: - Low or falling edge - High or rising edge

15 HTTP API

The current HTTP API version is v16. The API is valid for both the HTTP and the HTTPS server.

The HTTP API allows you to interact with an HBM10 module through HTTP requests. Accordingly, all API endpoints will be prefixed with “`api/v16`”. Additional versions may be introduced in the future, and will be accompanied by a different prefix.

The HTTP server is accessed through HTTP requests to one of the following endpoints:

API Endpoint	Description
device.action	Device configuration
network.action	Network configuration
upgrade.action	Firmware update over the air
radio.action	Internet radio control
leds.action	LEDs control
sound.action	Sound control
iperf.action	iPerf3 control
configure.action *[deprecated]*	Device and network configuration

Parameters has to be passed into these endpoints through the request body as a JSON object.

HTTP Request Rules

For proper operation the following requirements has to be met:

- All HTTP request are `'POST'` requests.
- The `'Content-Type'` is `'application/json'`.
- The `'charset'` is UTF-8.

The syntax for the body parameters description follows the rules for the Extended Backus–Naur Form (EBNF):

- Words inside double quotes (" ... ") represent terminal strings.
- Square brackets ([...]) surround optional items.
- Curly brackets ({ ... }) surround items that can repeat zero or more times.
- A vertical line (|) seperates alternatives.
- `String = ? all visible ASCII characters ?`

Sample Requests using HTTPS

The sample requests use the network tool “curl” to provide examples to connect to the HBM10 with the HTTP protocol. To connect the module using HTTPS curl has to be called with the parameter “**-k**” and the URL should be **https**://<IP>:**4949**/..., e.g.

```
curl -k –H 'Content-Type: application/json; charset=UTF-8'
     –d '{"action": "query"}'
     –X POST https://1921.168.2.1:4949/api/v16/configure.action
```

15.1 Device

15.1.1 Get Device Information

Description

Get the information about the device.

Method

POST

URL

http://<IP>:8989/api/v16/device.action

Request Parameters

Parameter	Description	Value
action	Action type	"info"

Response Parameters

Parameter	Description	Value
model	Model name	String
devicename	Friendly name of the device	String
firmware	Firmware version	String
hardware	Hardware revision	String

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "info"}'                                                 \
     -X POST http://192.168.2.1:8989/api/v16/device.action
```

Sample Response

```
{
  "model": "HBM10",                          // Model name
  "devicename": "Wohnzimmer",                // Device name
  "firmware": "4.3.0"                        // Firmware version
  "hardware": "R8EM49490B1"                  // Hardware revision
}
```

15.1.2 Set Device Name

Description

Set a friendly name for the device.

Method

POST

URL

http://<IP>:8989/api/v16/device.action

Request Parameters

Parameter	Description	Value
action	Action type	`"setdevicename"`
devicename	Friendly name of the device	`String`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>New device name was set.</td></tr>
<tr><td>error</td><td>New device name could not be set.</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                 \
     -d '{"action": "setdevicename", "devicename": "Wohnzimmer"}'       \
     -X POST http://192.168.2.1:8989/api/v16/device.action
```

Sample Response

```
{ "returncode": "success" }
```

15.1.3 Factory Reset

Description

Reset the device to factory settings.

Method

POST

URL

http://<IP>:8989/api/v16/device.action

Request Parameters

Parameter	Description	Value
action	Action type	"factoryreset"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "factoryreset"}'                                          \
     -X POST http://192.168.2.1:8989/api/v16/device.action
```

Sample Response

```
{ "returncode": "success" }
```

15.1.4 Reboot

Description

Reboot the device.

Method

POST

URL

http://<IP>:8989/api/v16/device.action

Request Parameters

Parameter	Description	Value
action	Action type	"reboot"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "reboot"}'                                                \
     -X POST http://192.168.2.1:8989/api/v16/device.action
```

Sample Response

```
{ "returncode": "success" }
```

15.2 Network

15.2.1 Get Network Information

Description

Get the network configuration.

Method

POST

URL

http://<IP>:8989/api/v16/network.action

Request Parameters

Parameter	Description	Value
action	Action type	"info"

Response Parameters

Parameter	Description	Value
wlan	*Wireless network*	Object | Null
└ **mac**	└ MAC address	String
└ **mode**	└ Wi-Fi operation mode	"AP" | "station"
└ **ssid**	└ Service Set Identifier	String
└ **encryption**	└ Encryption type	"NONE" | "WEP" | "WPA" | "WPA2-PSK"
└ ***[inet]***	└ *List of inet objects*	*List*
eth	*Ethernet network*	Object | Null
└ **mac**	└ MAC address	String
└ ***[inet]***	└ *List of inet objects*	*List*

JSON Objects

inet

Parameter	Description	Value
family	Internet address family type	"IPv4" | "IPv6"
address	IP address	String
netmask	Netmask	String

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "info"}'                                                  \
     -X POST http://192.168.2.1:8989/api/v16/network.action
```

Sample Response

Sample response for WLAN interface being up and the Ethernet interface being down:

```
{
  "wlan": {                                        // WLAN is up
      "mac": "00:23:b1:66:ab:94"                   // MAC address
      "mode": "AP",                                // AP mode
      "ssid": "HBM10-AB94",                        // SSID
      "channel": 11,                               // Channel
      "encryption": "NONE",                        // No encryption
      "inet": [                                    // Internet addresses
                  {
                      "family": "IPv4",            // IPv4
                      "address": "192.168.2.1"     // IP address
                      "netmask": "255.255.255.0"   // Netmask
                  }
              ]
   }
}
```

Sample response for WLAN interface being down and the Ethernet interface being up:

```
{
   "eth": {                                        // Ethernet is up
      "mac": "00:23:b1:a4:35:9e"                   // MAC address
      "inet": [                                    // Internet addresses
                  {
                      "family": "IPv4",            // IPv4
                      "address": "192.168.178.102" // IP address
                      "netmask": "255.255.255.0"   // Netmask
                  }
              ]
   }
}
```

15.2.2 Get AP Mode

Description

Get informations about the current AP mode configuration.

Method

POST

URL

http://<IP>:8989/api/v16/network.action

Request Parameters

Parameter	Description	Value
action	Action type	`"getapmode"`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>ssid</b></td><td>Current Service Set Identifier</td><td><code>String</code></td></tr>
<tr><td><b>encrypt</b></td><td>Current encryption mode
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>0</td><td>No encryption (open network)</td></tr>
<tr><td>1</td><td>WPA2-PSK encryption</td></tr>
</table></td><td><code>Enum</code></td></tr>
<tr><td><b>channel</b></td><td>Current channel used by the AP,
<i>1 <= channel <= 13.</i></td><td><code>Integer</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "getapmode"}'                                             \
     -X POST http://192.168.2.1:8989/api/v16/network.action
```

Sample Response

```
{
   "ssid": "HBM10-AB94",                         // SSID
   "channel": 11,                                // Channel
   "encrypt": 0",                                // No encryption
}
```

15.2.3 Set AP Mode

Description

Set-up a new AP mode configuration.

By default, the module operates in AP mode without any encryption. If this is not desirable, the AP mode can be configured to operate with WPA2-PSK encryption.

Additionally, the SSID and the channel can be changed.

Method

POST

URL

http://<IP>:8989/api/v16/network.action

Request Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>action</b></td><td>Action type</td><td>"setapmode"</td></tr>
<tr><td><b>ssid*</b></td><td>Service Set Identifier</td><td>String</td></tr>
<tr><td><b>encrypt*</b></td><td>Encryption mode
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>0</td><td>No encryption (open network)</td></tr>
<tr><td>1</td><td>WPA2-PSK encryption</td></tr>
</table></td><td>Enum</td></tr>
<tr><td><b>psk*</b></td><td>Pre-shared key whereas the psk length is limited to:
<i>8 &lt;= len(psk) &lt;= 63</i>
Ignored if encryption is set to 0 otherwise required.</td><td>String</td></tr>
<tr><td><b>channel*</b></td><td>Channel the AP will operate on with
<i>1 &lt;= channel &lt;= 13.</i></td><td>Integer</td></tr>
</table>

*) If any of these values is omitted the AP will keep its current setting.

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>AP mode configured successfully</td></tr>
</table></td><td>"success" | "error"</td></tr>
</table>

	error	AP mode could not be configured	

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "setapmode",                                              \
          "ssid": "AP_Wohnzimmer",                                            \
          "encrypt": 1,                                                       \
          "psk": "mysecretpresharedkey"                                       \
          }'                                                                  \
     -X POST http://192.168.2.1:8989/api/v16/network.action
```

Sample Response

```
{ "returncode": "success" }
```

15.3 Wi-Fi

15.3.1 Scan

Description

Get a list of scanned networks.

Method

POST

URL

http://<IP>:8989/api/v16/wifi.action

Request Parameters

Parameter	Description	Value
action	Action type	`"scan"`

Response Parameters

Parameter	Description	Value
[network]	List of network objects	`List`

JSON Objects

network

Parameter	Description	Value
channel	Wireless channel	`Integer`
ssid	Service Set Identifier	`String`
bssid	Basic Service Set Identification	`String`
signal	Signal level	`Integer`
wpa*	*WPA*	*`Object	Null`*
└ **auth**	└ Authentification	`String`
└ **group_cipher**	└ Group cipher	`String`
└ **pairwise_ciphers**	└ Pairwise ciphers	`String`
rsn*	*WPA2*	*`Object	Null`*
└ **auth**	└ Authentification	`String`
└ **group_cipher**	└ Group cipher	`String`
└ **pairwise_ciphers**	└ Pairwise ciphers	`String`

Note that the parameters *wpa* and *rsn* can be omitted if the corresponding security protocol is not supported by the access point.

Sample Request

```
curl –H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "scan"}'                                           \
     –X POST http://192.168.2.1:8989/api/v16/wifi.action
```

Sample Response

```
[
   {                                         // Encyption with WPA2-PSK
      "channel": 1,
      "ssid": "Lintech1",
      "bssid": "80:2a:a8:51:a4:d1",
      "signal": -8,
      "rsn": {
         "auth": "PSK",
         "pairwise_ciphers": "CCMP",
         "group_cipher": "CCMP"
      }
   },
   {                                         // Encryption with WPA or WPA2-PSK
      "channel": 4,
      "ssid": "Lintech2",
      "bssid": "00:14:6c:53:4f:52",
      "signal": -78,
      "wpa": {
         "auth": "PSK",
         "pairwise_ciphers": "CCMP TKIP",
         "group_cipher": "TKIP"
      },
      "rsn": {
         "auth": "PSK",
         "pairwise_ciphers": "CCMP TKIP",
         "group_cipher": "TKIP"
      }
   },
   {                                         // No encryption (Open Network)
      "channel": 8,
      "ssid": "HBM10-BDDF",
      "bssid": "00:90:4c:07:71:12",
      "signal": -24,
   }
]
```

15.3.2 List Networks

Description

List stored Wi-Fi networks.

Method

POST

URL

http://<IP>:8989/api/v16/wifi.action

Request Parameters

Parameter	Description	Value
action	Action type	`"listnetworks"`

Response Parameters

Parameter	Description	Value
[network]	List of configured network objects	`List`

JSON Objects

network

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>id</b></td><td>Network ID</td><td><code>Integer</code></td></tr>
<tr><td><b>ssid</b></td><td>Service Set Identifier</td><td><code>String</code></td></tr>
<tr><td><b>state</b></td><td>State
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Disabled network</td></tr>
<tr><td>1</td><td>Enabled network</td></tr>
<tr><td>2</td><td>Current network</td></tr>
</table>
</td><td><code>Enum</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                   \
     -d '{"action": "listnetworks" }'                                     \
     -X POST http://192.168.2.1:8989/api/v16/wifi.action
```

Sample Response

```
[  { "id": 1, "ssid": "Lintech1", "state": 2 },
   { "id": 2, "ssid": "Lintech2", "state": 0 },
   { "id": 3, "ssid": "Lintech3", "state": 1 }]
```

15.3.3 Add Network

Description

Add a new network configuration.

Method

POST

URL

http://<IP>:8989/api/v16/wifi.action

Request Parameters

Parameter	Description	Value
action	Action type	`"addnetwork"`
ssid	Service Set Identifier	`String`
psk*	Pre-shared key	`String | Null`

Note that the parameter *psk* is only necessary for encrypted networks. For unencrypted networks it can be omitted.

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Network was added to the configured network list.</td></tr>
<tr><td>error</td><td>Network could not be added to the configured network list.</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
<tr><td><b>Id</b></td><td>Network ID</td><td><code>Integer</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                    \
     -d '{"action": "addnetwork",                                           \
          "ssid": "Lintech1",                                               \
          "psk": "12345678"                                                 \
         }'                                                                 \
     -X POST http://192.168.2.1:8989/api/v16/wifi.action
```

Sample Response

```
{ "returncode": "success", "id": 1 }
```

15.3.4 Select Network

Description

Select a network from the network list to connect to.

Method

POST

URL

http://<IP>:8989/api/v16/wifi.action

Request Parameters

Parameter	Description	Value
action	Action type	`"selectnetwork"`
id	Network ID	`Integer`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Valid network ID, starting connection.</td></tr>
<tr><td>error</td><td>Network ID not valid.</td></tr>
</table>
</td><td>"success" | "error"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "selectnetwork",                                          \
          "id": "1" }'                                                        \
     -X POST http://192.168.2.1:8989/api/v16/wifi.action
```

Sample Response

```
{ "returncode": "success" }
```

15.3.5 Remove Network

Description

Remove a network from the configured network list. The corresponding ID must be fetched from calling the action "listnetworks" first.

Note, that the IDs returned by "listnetworks" might change after issueing a "removenetwork" command, so it is recommend to run this command afterwards again to get an updated network list.

Method

POST

URL

http://<IP>:8989/api/v16/wifi.action

Request Parameters

Parameter	Description	Value
action	Action type	"removenetwork"
id	Network ID	Integer

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Network is removed.</td></tr>
<tr><td>error</td><td>Network is not removed.</td></tr>
</table></td><td>"success" | "error"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "removenetwork", "id": 1 }'                               \
     -X POST http://192.168.2.1:8989/api/v16/wifi.action
```

Sample Response

```
{ "returncode": "success" }
```

15.4 OTA Upgrade

15.4.1 Firmware Update

Description

Start a firmware update by setting an URL from where the device will fetch the firmware. This only works in network client mode.

Method

POST

URL

http://<IP>:8989/api/v16/otaupgrade.action

Request Parameters

Parameter	Description	Value
action	Action type	`"seturl"`
otaaddress	URL where to fetch the update file from	`String`
filesize	File size of the update file	`String`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "seturl"                                                  \
          "otaaddress": "http://dl.lintech.de/download/upgrade/HBM10/3.3.6"   \
          "filesize": "25438208"}'                                            \
     -X POST http://192.168.1.159:8989/api/v16/otaupgrade.action
```

Sample Response

```
{ "returncode": "success" }                  // Update starts
```

15.4.2 Firmware Update Status

Description

Requests the firmware update status. This only works in network client mode.

NOTE: Querying the firmware update status should not be done in periods less then 1 second.

Method

POST

URL

http://<IP>:8989/api/v16/otaupgrade.action

Request Parameters

Parameter	Description	Value
action	Action type	`"querystatus"`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                  \
     -d '{"action": "querystatus"}'                                      \
     -X POST http://192.168.1.159:8989/api/v16/otaupgrade.action
```

Response Parameters

Parameter	Description
status	Status code: -1: update not running 1: downloading 2: download finished 3: updating 4: checksum verification 5: failure occurred
downfilename	File name currently downloading
downprogress	Download progress in percent
errinfo	Error code: -1: update not running 1: file not exist 2: download error 3: checksum error 4: flash error 5: update error

Sample Response

```
{
    "status": "1",                           // Status code
    "downfilename": "airlino.swu",           // Current downloading file name
    "downprogress": "44",                    // Download process in percent
    "errinfo": "-1"                          // Error code
}
```

15.5 Internet Radio

15.5.1 Play

Description

Starts to play a radio stream (MP3 or Ogg/Vorbis).

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"play"`
station	*Radio station*	
└ **name**	└ Station name	`String`
└ **url**	└ Station URL	`String`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Plays radio stream</td></tr>
<tr><td>error</td><td>Radio stream cannot be played, e.g. URL is not a valid</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "play",                                                    \
          "station": {                                                        \
              "name": "Inforadio",                                            \
              "url": "http://inforadio.de/livemp3"                            \
          } }'                                                                \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }                  // Plays the stream
```

15.5.2 Play/Pause

Description

Toggles between play and pause.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"playpause"`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{ "action": "playpause" }'                                     \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

15.5.3 Next

Description

Starts playing next station from a playlist.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	"next"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{ "action": "next" }'                                               \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

15.5.4 Previous

Description

Start playing previous station from a playlist.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	"prev"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                \
     -d '{ "action": "prev" }'                                          \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

15.5.5 Stop

Description

Stops playback of a radio stream.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	"stop"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{ "action": "stop" }'                                                \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

15.5.6 Query

Description

Query the current station.

If metadata is available for the current radio station it is appended to the station information. Note that the field *station.name* contains the string set through the HTTP-API, whereas the field *station.meta.name* contains a string fetched from the metadata.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"query"`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>state</b></td><td>Radio playback state<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>1</td><td>Not playing</td></tr><tr><td>2</td><td>Playing</td></tr><tr><td>3</td><td>Playing, but paused</td></tr></table></td><td><code>Integer</code></td></tr>
<tr><td><b>listid</b></td><td>Playlist identification number</td><td><code>Integer</code></td></tr>
<tr><td><b><i>station</i></b></td><td><i>Radio station</i></td><td><code>Object</code></td></tr>
<tr><td>└ <b>name</b></td><td>└ Station name</td><td><code>String</code></td></tr>
<tr><td>└ <b>url</b></td><td>└ Station URL</td><td><code>String</code></td></tr>
<tr><td>└ <b><i>meta</i></b></td><td>└ Station metadata</td><td><code>Object</code></td></tr>
<tr><td> └ <b>now_playing</b></td><td> └ Now playing</td><td><code>String</code></td></tr>
<tr><td> └ <b>name</b></td><td> └ Name</td><td><code>String</code></td></tr>
</table>

`listid` is only set if a playlist is played.

`meta` is only set if metadata are available for the current radio station.

Sample Request

```
curl –H 'Content-Type: application/json; charset=UTF-8'                       \
     –d '{ "action": "query" }'                                                \
     –X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{  "state": 2,
   "station": {
       "url": "http://stream.berliner-rundfunk.de/brf/mp3-128/internetradio",
       "name": "Berliner Rundfunk Livestream",
       "meta": {
           "name": "Berliner Rundfunk Livestream",
           "now_playing": "KARAT - DER BLAUE PLANET"
       }
   }
}
```

15.5.7 Get Playlist

Description

Fetch a playlist from the device’s internal memory.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"getplaylist"`
listid	Playlist identification number	`Integer`

Response Parameters

Parameter	Description	Value
description	Short description	`String`
stationlist	*List of stations*	
└ **[station]**	└ List of radio station objects	`List`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{ "action": "getplaylist", "listid": 1 }'                             \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

Playlist does not exists or is empty:

```
{ }   // empty JSON object
```

Playlist exists:

```
{
  "description": "News",
  "stationlist": [
      { "name": "Inforadio",
        "url": "http://inforadio.de/livemp3" },
      { "name": "Deutschlandfunk",
        "url": "http://dradio-ogg-dwissen-l.akacast.akamaistream.net/7/192/
135496/v1/gnl.akacast.akamaistream.net/dradio_ogg_dwissen_l" }
  ]
}
```

15.5.8 Save Playlist

Description

Store a radio station playlist to the device’s internal memory. A playlist with the same ID will be overwritten.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"saveplaylist"`
listid	List identification number	`Integer`
playlist	*Playlist*	
└ **description**	└ Short description	`String`
└ ***stationlist***	└ *List of stations*	
└ **[station]**	└ List of radio station objects	`List`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>playlist is stored in memory</td></tr>
<tr><td>error</td><td>playlist was <b>not</b> stored, e.g. due to a JSON parser error</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "saveplaylist",                                           \
          "listid": 1,                                                        \
          "playlist": {                                                       \
              "description": "News",                                          \
              "stationlist": [                                                \
                  { "name": "Inforadio",                                      \
                    "url": "http://inforadio.de/livemp3" },                   \
                  { "name": "Deutschlandfunk",                                \
                    "url": "http://dradio-ogg-dwissen-l.akacast.akamaistream.net
/7/192/135496/v1/gnl.akacast.akamaistream.net/dradio_ogg_dwissen_l" }         \
          ]}}'                                                                \
      –X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }
```

15.5.9 Remove Playlist

Description

Deletes a specific playlist from the internal memory.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"rmplaylist"`
listid	List identification number	`Integer`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>playlist is removed</td></tr>
<tr><td>error</td><td>playlist cannot be deleted, e.g. the list does not exist</td></tr>
</table></td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                    \
     -d '{ "action": "rmplaylist", "listid": 1 }'                          \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }
```

15.5.10 Play Playlist

Description

Request to play a specific playlist from the internal memory. Playback will start with the first entry in the list. It is possible to step forward in the list with KEY_NEXT and step back in the list with KEY_PREV. The playlist will be played in repeat mode: if KEY_NEXT is pressed while the last entry of the list is currently played, the first entry in the list will be played next and if KEY_PREV is pressed while the first entry of the list is currently played, the last entry in the list will be played.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"playplaylist"`
listid	List identification number	`Integer`
pos*	Position of the station in the list to start playback from, whereby: *1 <= pos <= 255.*	`Integer`

Note, the parameter *pos* is optional. If omitted the playback will start with the first station in the list.

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Playback is about to begin</td></tr>
<tr><td>error</td><td>playlist cannot be played, e.g. the list contains invalid an URL</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{ "action": "playplaylist", "listid": 1, "pos": 3 }'          \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }
```

15.5.11 Get Favourite Station

Description

Get the current favourite radio station.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"getfavouritestation"`

Response Parameters

Parameter	Description	Value
station	*Radio station*	
└ **name**	└ Station name	`String`
└ **url**	└ Station URL	`String`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                 \
     -d '{"action": "getfavouritestation" }                              \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "name": "Inforadio", "url": "http://inforadio.de/livemp3" }
```

15.5.12 Set Favourite Station

Description

Set a radio station as the favourite station. This station will always be played when the module is ready after power up and is connected to the network.

To remove the favourite station issue a request with no station object set.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"setfavouritestation"`
station	*Radio station*	
└ **name**	└ Station name	`String`
└ **url**	└ Station URL	`String`

If no station is set the current favourite station is removed.

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>success</td><td>Station was set as favourite</td></tr><tr><td>error</td><td>Station could not set as favourite</td></tr></table></td><td>"success" | "error"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                        \
     -d '{"action": "setfavouritestation",                                       \
          "station": {                                                          \
              "name": "Inforadio",                                              \
              "url": "http://inforadio.de/livemp3"                              \
         } }'                                                                   \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }
```

15.5.13 Get Favourite Playlist

Description

Get the current favourite playlist of radio stations.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	"getfavouriteplaylist"

Response Parameters

Parameter	Description	Value
listid	ID of the favourite playlist	Integer

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                        \
     -d '{"action": "getfavouriteplaylist" }'                                   \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "listid": 1 }
```

15.5.14 Set Favourite Playlist

Description

Set a playlist as the favourite playlist. This playlist will always be played when the module is ready after power up and is connected to the network.

Note that if a favourite station is set, too, the favourite playlist will be ignored.

To remove the favourite station issue a request with no station object set.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"setfavouriteplaylist"`
listid	ID of the favourite playlist	`Integer`

If no listid is set the current favourite playlist is removed.

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Playlist was set.</td></tr>
<tr><td>error</td><td>Playlist could not be set, e.g. no such playlist stored.</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                \
     -d '{"action": "setfavouriteplaylist",                            \
          "listid": 1 }'                                               \
     -X POST http://192.168.1.159:8989/api/v16/radio.action
```

Sample Response

```
{ "returncode": "success" }
```

15.6 LEDs Control

15.6.1 Get LEDs configuration

Description

Get the current LEDs configuration.

Method

POST

URL

http://<IP>:8989/api/v16/leds.action

Request Parameters

Parameter	Description	Value
action	Action type	`"get"`

Response Parameters

Parameter	Description	Value
brightness	LEDs brightness level with: 1 <= *brightness* <= 255	`Integer`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{ "action": "get" }'                                                 \
     -X POST http://192.168.1.159:8989/api/v16/leds.action
```

Sample Response

```
{ "brightness": 30 }
```

15.6.2 Set LEDs configuration

Description

Set the current LEDs configuration.

Method

POST

URL

http://<IP>:8989/api/v16/leds.action

Request Parameters

Parameter	Description	Value
action	Action type	`"set"`
brightness	LEDs brightness level with: 1 <= *brightness* <= 255	`Integer`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Success</td></tr>
<tr><td>error</td><td>An error occured</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                     \
     -d '{ "action": "set", "brightness": 30 }'                             \
     -X POST http://192.168.1.159:8989/api/v16/leds.action
```

Sample Response

```
{ "returncode": "success" }
```

15.7 Sound Control

15.7.1 Get Master Volume

Description

Get the current Master volume level.

Method

POST

URL

http://<IP>:8989/api/v16/sound.action

Request Parameters

Parameter	Description	Value
action	Action type	`"getmastervol"`

Response Parameters

Parameter	Description	Value
volume	Master volume level with: 0 <= *volume* <= 255	`Integer`

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{ "action": "getmastervol" }'                                        \
     -X POST http://192.168.1.159:8989/api/v16/sound.action
```

Sample Response

```
{ "volume": 255 }
```

15.7.2 Set Master Volume

Description

Set the Master volume to a new value.

Note that the volume is stored periodically. To make sure a volume change withstands an abrupt power-cut a delay of 10 seconds is necessary.

Method

POST

URL

http://<IP>:8989/api/v16/sound.action

Request Parameters

Parameter	Description	Value
action	Action type	`"setmastervol"`
volume	Master volume level with: 0 <= *volume* <= 255	`Integer`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Success</td></tr>
<tr><td>error</td><td>An error occured</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{ "action": "setmastervol", "volume": 255 }'                         \
     -X POST http://192.168.1.159:8989/api/v16/sound.action
```

Sample Response

```
{ "returncode": "success" }
```

15.7.3 Get Status Tones Volume

Description

Get the current volume level of the status tones.

Note that the volume is stored periodically. To make sure a volume change withstands an abrupt power-cut a delay of 10 seconds is necessary.

Method

POST

URL

http://<IP>:8989/api/v16/sound.action

Request Parameters

Parameter	Description	Value
action	Action type	"getstatusvol"

Response Parameters

Parameter	Description	Value
volume	Master volume level with: 0 <= *volume* <= 255	Integer

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                \
     -d '{ "action": "getstatusvol" }'                                 \
     -X POST http://192.168.1.159:8989/api/v16/sound.action
```

Sample Response

```
{ "volume": 0 }
```

15.7.4 Set Status Tones Volume

Description

Set the volume for the status tones to a new value.

Method

POST

URL

http://<IP>:8989/api/v16/sound.action

Request Parameters

Parameter	Description	Value
action	Action type	`"setstatusvol"`
volume	Status tones volume level with: 0 <= *volume* <= 255	`Integer`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Success</td></tr>
<tr><td>error</td><td>An error occured</td></tr>
</table>
</td><td><code>"success" | "error"</code></td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                 \
     -d '{ "action": "setstatusvol", "volume": 0 }'                     \
     -X POST http://192.168.1.159:8989/api/v16/sound.action
```

Sample Response

```
{ "returncode": "success" }
```

15.8 iPerf Control

15.8.1 Enable iPerf3 Server

Description

Start a iPerf3 server daemon. If enabled the daemon will be started on bootup.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	"enable"

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Success</td></tr>
<tr><td>error</td><td>An error occured</td></tr>
</table>
</td><td>"success" | "error"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                  \
     -d '{ "action": "enable" }'                                          \
     -X POST http://192.168.1.159:8989/api/v16/iperf.action
```

Sample Response

```
{ "returncode": "success" }
```

15.8.2 Disable iPerf3 Server

Description

Stop the iPerf3 server daemon. If disabled the daemon will not be started on bootup.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"disable"`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>returncode</b></td><td>Response message
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>success</td><td>Success</td></tr>
<tr><td>error</td><td>An error occured</td></tr>
</table></td><td>"success" | "error"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{ "action": "disable" }'                                            \
     -X POST http://192.168.1.159:8989/api/v16/iperf.action
```

Sample Response

```
{ "returncode": "success" }
```

15.8.3 Get Status Of iPerf3 Server

Description

Get the status if the iPerf3 server daemon is currently running.

Method

POST

URL

http://<IP>:8989/api/v16/radio.action

Request Parameters

Parameter	Description	Value
action	Action type	`"status"`

Response Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>enabled</b></td><td>Server status
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>true</td><td>Server is running</td></tr>
<tr><td>false</td><td>Server is not running</td></tr>
</table>
</td><td>Boolean</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                    \
     -d '{ "action": "status" }'                                           \
     -X POST http://192.168.1.159:8989/api/v16/iperf.action
```

Sample Response

```
{ "enabled": true }
```

15.9 Configure (*deprecated*)

Note that the configure API endpoint is deprecated since API v16 and may be removed in the future. As a replacement, the API endpoints “device”, “network” and “wifi” shall be used.

15.9.1 Get Device Status

Description

Get the current device and network configuration.

Method

POST

URL

http://<IP>:8989/api/v16/configure.action

Request Parameters

Parameter	Description	Value
action	Action type	`"query"`

Response Parameters

Parameter	Description	Value
networkinfo	*Wireless network information*	
└ ***apinfo***	└ *AP mode information*	`Wifi object`
└ ***clientinfo***	└ *Client mode information*	`Wifi object`
└ **wifimode**	└ Current wifi mode	`"ap" | "client"`
└ ***ipaddressinfo***	└ *IP information*	`IPAddress object`
ethinfo	*Ethernet network information*	
└ ***ipaddressinfo***	└ *IP information*	`IPAddress object`
deviceinfo	*Device information*	`DeviceInfo object`

JSON Objects

Wifi

Parameter	Description	Value
wifissid	Service Set Identifier	`String`
wifipwd	Authentification password	`String`

<table>
<tr><td><b>encrypt_type</b></td><td>Encryption type<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>NONE</td><td>No encryption</td></tr><tr><td>WEP</td><td>WEP encrypted</td></tr><tr><td>WPA</td><td>WPA2 or WPA encrypted</td></tr></table></td><td>"NONE" | "WEP" | "WPA"</td></tr>
<tr><td><b>encrypt_subtype</b></td><td>Encryption subtype<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>WPA2</td><td>WPA2 only</td></tr><tr><td>WPA</td><td>WPA only</td></tr><tr><td>WPA2 WPA</td><td>WPA2 and WPA</td></tr></table></td><td>"WPA2" | "WPA" | "WPA2 WPA"</td></tr>
<tr><td><b>group_cipher</b></td><td>Group cipher</td><td>String</td></tr>
<tr><td><b>pairwise_ciphers</b></td><td>Pairwise cipher</td><td>String</td></tr>
</table>

IPAddress

Parameter	Description	Value
type	IP address assignment method	"DHCP"
subnetmask	Subnet mask	Reserved
primarydns	Primary DNS	Reserved
seconddns	Second DNS	Reserved
gateway	Gateway	Reserved
ipaddress	IP address	Reserved

DeviceInfo

Note that ethaddress is only present for HBM10-ETH devices and was included in API v16.

Parameter	Description	Value
model	Model name	String
devicename	Friendly name of the device	String
softwarever	Firmware version	String
macaddress	Wifi MAC address	String
ethaddress*	Ethernet MAC address	String
configured	Wifi interface is integrated into a network	"true" | "false"
serialnumber	Serial number	Reserved
hardwarever	Hardware revision	String

airplaypwd	AirPlay password	Reserved
devicepwd	Device password	Reserved

Sample Request

```
curl –H 'Content-Type: application/json; charset=UTF-8'                \
     -d '{"action": "query"}'                                          \
     –X POST http://192.168.2.1:8989/api/v16/configure.action
```

Sample Response

AP mode

```
{
    "networkinfo": {
        "apinfo": {
            "wifissid": "HBM10-AB94",              // SSID
            "wifipwd": "",
            "encrypt_type": "",
            "encrypt_subtype": "",
            "group_cipher": "",
            "pairwise_ciphers": ""
        },
        "clientinfo": {
            "encrypt_type": "",
            "encrypt_subtype": "",
            "wifipwd": "",
            "group_cipher": "",
            "wifissid": "",
            "pairwise_ciphers": ""
        },
        "wifimode": "ap",                          // AP mode
        "ipaddressinfo": {
            "type": "",
            "subnetmask": "",
            "secondarydns": "",
            "gateway": "",
            "primarydns": "",
            "ipaddress": ""
        }
    },
    "ethinfo": {
        "ipaddressinfo": {
            "type": "DHCP",                        // DHCP (Dynamic IP)
            "subnetmask": "",
            "seconddns": "",
            "gateway": "",
            "primarydns": "",
            "ipaddress": "192.168.178.101"         // IPv4 address
        }
    },
    "deviceinfo": {
        "airplaypwd": "",
        "serialnumber": "2976295828",              // Serial number
        "hardwarever": "R8EM49490B1",               // Hardware version
        "macaddress": "00:23:b1:66:ab:94",          // MAC address
        "model": "HBM10",                           // Model name
        "devicepwd": "",
        "configured": "false",                      // Device is not configured
        "devicename": "HBM10",                      // Device name
        "softwarever": "4.2.0"                      // Software version
    }
}
```

Client mode

```
{
    "networkinfo": {
        "apinfo": {
            "encrypt_type": "",
            "encrypt_subtype": "",
            "wifipwd": "",
            "group_cipher": "",
            "wifissid": "",
            "pairwise_ciphers": ""
        },
        "clientinfo": {
            "wifissid": "My Home",                  // SSID
            "wifipwd": "",
            "encrypt_type": "WPA2-PSK",             // Encryption
            "encrypt_subtype": "",
            "group_cipher": "TKIP",                 // Group cipher
            "pairwise_ciphers": "CCMP"              // Pairwise cipher
        },
        "wifimode": "client",
        "ipaddressinfo": {
            "type": "DHCP",                         // DHCP (Dynamic IP)
            "subnetmask": "",
            "seconddns": "",
            "gateway": "",
            "primarydns": "",
            "ipaddress": ""
        }
    },
    "ethinfo": {
        "ipaddressinfo": {
            "type": "DHCP",                         // DHCP (Dynamic IP)
            "subnetmask": "",
            "seconddns": "",
            "gateway": "",
            "primarydns": "",
            "ipaddress": "192.168.178.101"          // IPv4 address
        }
    },
    "deviceinfo": {
        "airplaypwd": "",
        "serialnumber": "2976295828",               // Serial number
        "hardwarever": "R8EM49490B1",               // Hardware version
        "macaddress": "00:23:b1:66:ab:94",          // MAC address
        "model": "HBM10",                           // Model name
        "devicepwd": "",
        "configured": "true",                       // Configured
        "devicename": "HBM10",                      // Device name
        "softwarever": "4.2.0"                      // Software version
    }
}
```

15.9.2 Wifi Scan

Description

Get a list of scanned networks.

Method

POST

URL

http://<IP>:8989/api/v16/configure.action

Request Parameters

Parameter	Description	Value
action	Action type	"wifiscan"

Response Parameters

Parameter	Description	Value
aplist	*List of scanned networks*	
└ **[network]**	└ List of network objects	List of Network objects

JSON Objects

Network

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>bss</b></td><td>Basic Service Set Identification</td><td>String</td></tr>
<tr><td><b>ssid</b></td><td>Service Set Identifier</td><td>String</td></tr>
<tr><td><b>channel</b></td><td>Wireless channel</td><td>String</td></tr>
<tr><td><b>signal_level</b></td><td>Signal level</td><td>String</td></tr>
<tr><td><b>encrypt_type</b></td><td>Encryption type
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>NONE</td><td>No encryption</td></tr>
<tr><td>WEP</td><td>WEP encrypted</td></tr>
<tr><td>WPA</td><td>WPA2 or WPA encrypted</td></tr>
</table></td><td>"NONE" | "WEP" | "WPA"</td></tr>
<tr><td><b>encrypt_subtype</b></td><td>Encryption subtype
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>WPA2</td><td>WPA2 only</td></tr>
<tr><td>WPA</td><td>WPA only</td></tr>
</table></td><td>"WPA2" | "WPA" | "WPA2 WPA"</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>WPA2 WPA</td><td>WPA2 and WPA</td></tr></table></td><td></td></tr>
<tr><td><b>group_cipher</b></td><td>Group cipher</td><td>String</td></tr>
<tr><td><b>pairwise_ciphers</b></td><td>Pairwise cipher</td><td>String</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "wifiscan"}'                                             \
     -X POST http://192.168.2.1:8989/api/v16/configure.action
```

Sample Response

```
{
    "aplist": [
        {
            "encrypt_type": "NONE",                // Open network
            "encrypt_subtype": "",
            "ssid": "Some SSID 1",                 // SSID
            "group_cipher": "",
            "pairwise_ciphers": "",
            "bss": "00:12:23:34:45:56",            // MAC address
            "signal_level": "-46",                 // Signal level in dBm
            "channel": "6"                         // Channel
        },
        {
            "encrypt_type": "WPA",                 // Secured network
            "encrypt_subtype": "WPA2 WPA",         // WPA+WPA2
            "ssid": "Some SSID 2",                 // SSID
            "group_cipher": "TKIP",                // Group cipher
            "pairwise_ciphers": "CCMP TKIP",       // Pairwise cipher
            "bss": "00:14:6c:53:4f:52",            // MAC address
            "signal_level": "-24",                 // Signal level in dBm
            "channel": "6"                         // Channel
        }
    ]
}
```

15.9.3 Set Config

Description

Configure the device

Method

POST

URL

http://<IP>:8989/api/v16/configure.action

Request Parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Value</th></tr>
<tr><td><b>action</b></td><td>Action type</td><td>"setconfig"</td></tr>
<tr><td><b><i>deviceinfo</i></b></td><td><i>Device information</i></td><td></td></tr>
<tr><td>└ <b>devicename</b></td><td>└ Device name</td><td>String</td></tr>
<tr><td><b><i>networkinfo</i></b></td><td><i>Network information</i></td><td></td></tr>
<tr><td>└ <b>wifimode</b></td><td>└ Wifi mode</td><td>"client"</td></tr>
<tr><td>└ <b><i>clientinfo</i></b></td><td>└ <i>Network client configuration</i></td><td></td></tr>
<tr><td>└ <b>encrypt_type</b></td><td>└ Encryption type
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>NONE</td><td>No encryption</td></tr>
<tr><td>WEP</td><td>WEP encrypted</td></tr>
<tr><td>WPA</td><td>WPA2 or WPA encrypted</td></tr>
</table></td><td>"NONE" | "WEP" | "WPA"</td></tr>
<tr><td>└ <b>wifipwd</b></td><td>└ Wifi password</td><td>String</td></tr>
<tr><td>└ <b>wifissid</b></td><td>└ SSID name</td><td>String</td></tr>
<tr><td>└ <b><i>ipaddressinfo</i></b></td><td><i>IP address information</i></td><td></td></tr>
<tr><td>└ <b>type</b></td><td>└ IP address type</td><td>"DHCP"</td></tr>
</table>

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "setconfig",                                              \
          "deviceinfo": {"devicename": "Kitchen"},                            \
          "networkinfo": {"wifimode": "client",                               \
                          "clientinfo": {"encrypt_type": "WPA",               \
                                         "wifipwd": "12345678",               \
                                         "wifissid": "MyHomeNetwork"},        \
                          "ipaddressinfo": {"type": "DHCP"}}}'                \
     -X POST http://192.168.2.1:8989/api/v16/configure.action
```

Sample Response

```
{ "returncode": "success" }
```

15.9.4 Factory Reset

Description

Reset the device to factory settings.

Method

POST

URL

http://<IP>:8989/api/v13/configure.action

Request Parameters

Parameter	Description	Value
action	Action type	"resetdefault"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                       \
     -d '{"action": "resetdefault"}'                                          \
     -X POST http://192.168.2.1:8989/api/v16/configure.action
```

Sample Response

```
{ "returncode": "success" }
```

15.9.5 Reboot

Description

Reboot the device.

Method

POST

URL

http://<IP>:8989/api/v16/configure.action

Request Parameters

Parameter	Description	Value
action	Action type	"reboot"

Sample Request

```
curl -H 'Content-Type: application/json; charset=UTF-8'                      \
     -d '{"action": "reboot"}'                                                \
     -X POST http://192.168.2.1:8989/api/v16/configure.action
```

Sample Response

```
{ "returncode": "success" }
```

16 AT Commands Reference

NOTE: This feature is optional and not enabled by default. Please contact the LinTech support team: lintech@lintech.de.

16.1 AT Command Syntax

The “AT” or “at” prefix must be set at the beginning of each command line. To terminate a command line enter <CR>. Throughout this document, only the command lines are presented, <CR> is omitted intentionally.

Commands are usually followed by a response – <CR><LF>*response*<CR><LF>. Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.

The AT commands are case-insensitive and may be entered in either uppercase or lowercase letters and even can be mixed. Therefore, the following command lines are equivalent:

```
AT+UART?
at+uart?
At+Uart?
```

16.2 Result Codes

Result codes are messages sent from the Control Server to provide information about the execution of an AT command and the occurrence of an event. Two types of result codes are used:

- Final result codes
- Unsolicited result codes

A final result code marks the end of an AT command response. It is an indication that the Control Server has finished the execution of a command line. Two frequently used final result codes are **OK** and **ERROR**. Only one final result code will be returned for each command line.

The OK Final Result Code

The OK final result code indicates that a command line has been executed successfully by the Control Server. It always starts and ends with <CR><LF>.

The ERROR Final Result Code

The ERROR final result code indicates that an error occurs when the Control Server tries to execute a command line. After the occurrence of an error the Control Server will not process any remaining AT command. Like the OK final result code, the ERROR final result code always starts and ends with <CR><LF>. For common errors an error code follows the string “ERROR”, separated by a <SPACE> character, e.g.

```
ERROR 1
```

The following error codes are supported:

Error Code	Description
1	*Unknown command.* The command is not supported or contains a typo.
2	*Syntax error.* The command syntax is wrong, e.g. not all necessary parameters are set.
3	*Invalid range.* One or more parameters are out of range.

Unsolicited Result Codes

Unsolicited result codes are currently not used, but may be introduced with a new AT command.

16.3 Standard AT Commands

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td><b>AT</b></td><td><i>Test command.</i>
Response with <b>OK</b> when the control server is running.</td></tr>
<tr><td><b>A/</b></td><td><i>Repeat the last AT command.</i></td></tr>
<tr><td><b>ATE</b>[<echo>]</td><td><i>Echo command.</i>
<b>Parameters</b>:
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td><b>echo</b></td><td>Enum</td><td>0: Incoming characters will not be echoed.
1: Incoming characters will be echoed.</td></tr>
</table>
If <echo> is omitted, it defaults to 0.</td></tr>
<tr><td><b>AT&F</b></td><td><i>Factory defined configuration.</i>
All configuration settings impacted by the <b>AT&W</b> command are reset to their default value.</td></tr>
<tr><td><b>AT&W</b></td><td><i>Stores the current configuration settings in non-volatile memory.</i>
<b>Parameters impacted by AT&W command</b>:
<table>
<tr><th>Command</th><th>Parameter</th><th>Default</th></tr>
<tr><td><b>ATE</b></td><td><echo></td><td>0</td></tr>
<tr><td><b>UART</b></td><td><baud>
<data>
<parity>
<stop></td><td>9600
8
N
1</td></tr>
</table>
</td></tr>
</table>

16.4 Serial AT Commands

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td><b>AT+UART=</b>
&lt;baud>,<data>,
<parity>,<stop&gt;</td><td><i>Apply new UART settings.</i>
<b>Usage:</b>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td><b>baud</b></td><td>Integer</td><td><i>Supported baud rates:</i>
<table>
<tr><td>300</td><td>28800</td></tr>
<tr><td>1200</td><td>38400</td></tr>
<tr><td>2400</td><td>57600</td></tr>
<tr><td>4800</td><td>115200</td></tr>
<tr><td>9600</td><td>230400</td></tr>
<tr><td>14400</td><td>460800</td></tr>
<tr><td>19200</td><td>921600</td></tr>
</table>
<b>Note</b>: Other baud rates may work too, but are not supported.</td></tr>
<tr><td><b>data</b></td><td>Integer</td><td><i>Supported data bits:</i>
5, 6, 7 or 8</td></tr>
<tr><td><b>parity</b></td><td>Char</td><td><i>Supported parities:</i>
N: No parity
O: Odd parity
E: Even parity</td></tr>
<tr><td><b>stop</b></td><td>Integer</td><td><i>Supported stop bits:</i>
1 or 2</td></tr>
</table>
<b>Example:</b>
<code>AT+UART=9600,8,N,1</code></td></tr>
<tr><td><b>AT+UART?</b></td><td><i>Read current UART settings.</i>
<b>Response</b>:
<code>+UART=<baud>,<data>,<parity>,<stop&gt;</code>
<b>Example:</b>
<code>+UART=9600,8,N,1</code></td></tr>
</table>