



# Embedded USB Host Audio Class Driver User Guide

Version 2.20

For use with USB Host Audio Class Driver versions  
3.13 and above

## Table of Contents

<b>1. System Overview</b>	4
<b>1.1. Introduction</b>	5
<b>1.2. Feature Check</b>	6
<b>1.3. Packages and Documents</b>	7
<b>1.4. Change History</b>	8
<b>2. Source File List</b>	9
<b>3. Configuration Options</b>	10
<b>3.1. Audio Configuration Options</b>	10
<b>3.2. PCM Configuration Options</b>	12
<b>4. Application Programming Interface</b>	13
<b>4.1. Module Management</b>	13
usbh_audio_init	14
usbh_audio_start	15
usbh_audio_stop	16
usbh_audio_delete	17
<b>4.2. Device Management</b>	18
usbh_audio_change_mute	19
usbh_audio_change_volume	20
usbh_audio_find_first_terminal	21
usbh_audio_find_next_terminal	22
usbh_audio_play	23
usbh_audio_stop_play	24
usbh_audio_record	25
usbh_audio_stop_record	26
usbh_audio_present	27
usbh_audio_state	28
usbh_audio_get_stream_info	29
usbh_audio_get_port_hdl	30
usbh_audio_register_ntf	31
<b>4.3. Error Codes</b>	32
<b>4.4. Types and Definitions</b>	34
t_usbh_ntf_fn	34
Notification Codes	34
Terminal States	35

Terminal Group Macro .....	35
Stream Type .....	35
Invalid States .....	35
t_terminal_info .....	36
t_rw_data_info .....	36
t_stream_info .....	36
<b>4.5. Terminal Types .....</b>	<b>37</b>
Input Terminal Types .....	37
Output Terminal Types .....	37
Bidirectional Terminal Types .....	38
Telephony Terminal Types .....	38
External Terminal Types .....	39
Embedded Function Terminal Types .....	40
<b>5. Integration .....</b>	<b>41</b>
<b>5.1. OS Abstraction Layer (OAL) .....</b>	<b>41</b>
<b>5.2. PSP Porting .....</b>	<b>41</b>
<b>6. Sample Code .....</b>	<b>42</b>
<b>6.1. Initialization .....</b>	<b>42</b>
<b>7. Version .....</b>	<b>43</b>

# 1. System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- [Introduction](#) - describes the main elements of the module.
- [Feature Check](#) - summarizes the main features of the module as bullet points.
- [Packages and Documents](#) - the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- [Change History](#) - lists the earlier versions of this manual, giving the software version that each manual describes.

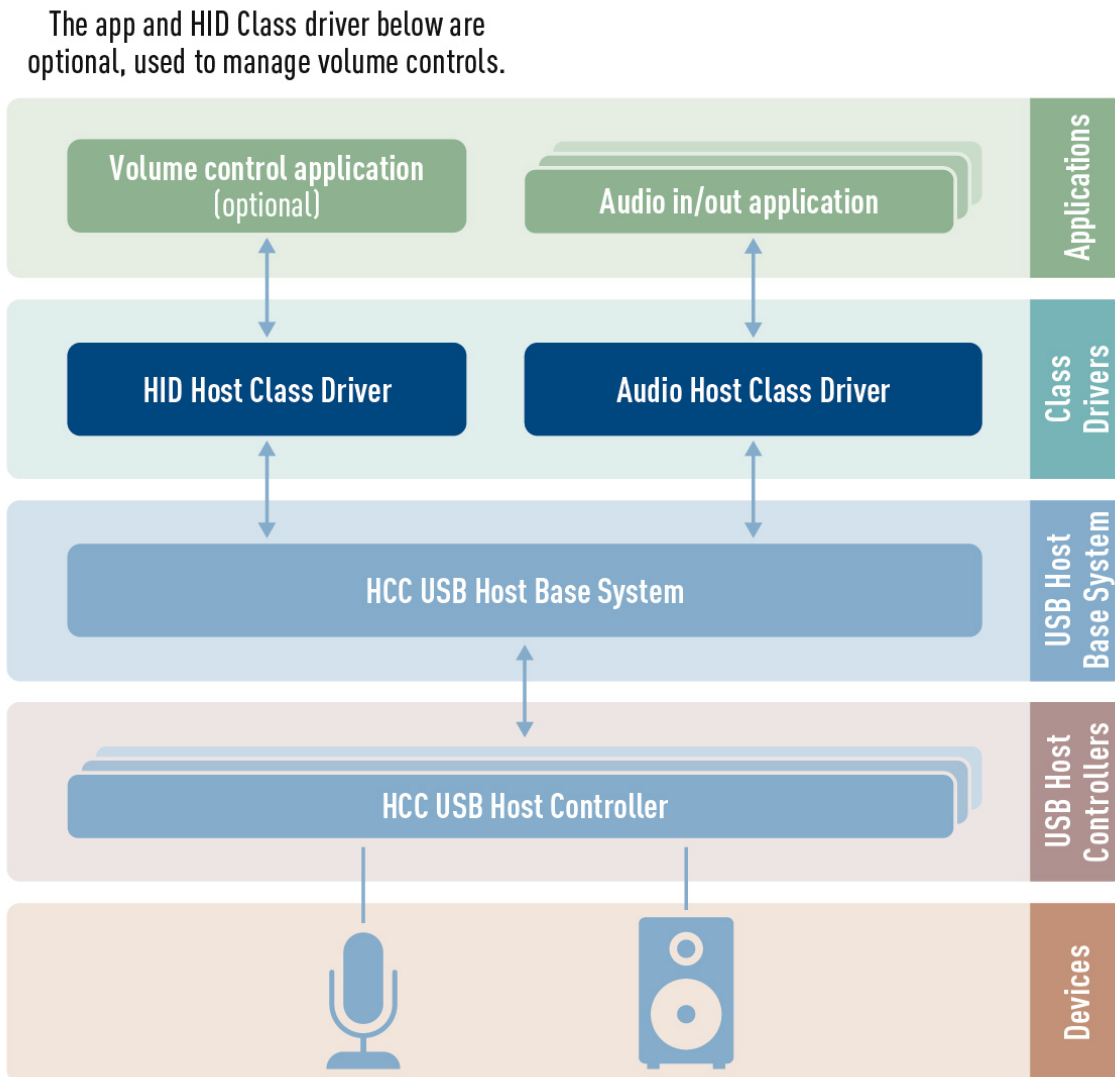
All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

## 1.1. Introduction

This guide is for those who want to implement an embedded USB host class driver to control audio USB devices (speakers or microphones). The **usbh\_cd\_audio** package provides an audio host class driver for a USB stack. This is used for connecting audio devices over a USB link to a host.

The system structure is shown in the diagram below:



The lower layer interface is designed to use HCC Embedded's USB Host Interface Layer. This layer is standard over different host controller implementations; this means that the code is unchanged, whichever HCC USB host controller it is interfaced to. For detailed information about this layer, consult the [HCC USB Host Base System User Guide](#) that is shipped with the base system.

The package provides a set of Application Programming Interface (API) functions for controlling access to a device. These are described here, with separate sections for module and audio management.

## 1.2. Feature Check

The main features of the class driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Compatible with all HCC USB host controllers.
- Supports multiple devices connected simultaneously.
- Supports speakers for playing sound.
- Supports microphone for recording.
- Works with HCC USB HID class driver to manage volume control.
- Uses a system of callbacks for user-specified events.



## 1.3. Packages and Documents

### Packages

The table below lists the packages that you need in order to use this module:

Package	Description
<b>hcc_base_doc</b>	This contains the two guides that will help you get started.
<b>usbh_base</b>	The USB host base package. This is the framework used by USB class drivers to communicate over USB using a specific USB host controller package.
<b>usbh_cd_audio</b>	The USB device audio host class driver package described by this document.
<b>usbh_cd_hid</b>	The Human Interface Device (HID) module. This is only required if volume knob/mute button handling is required on the audio device.

### Documents

For an overview of HCC's embedded USB stacks, see [Product Information](#) on the main HCC website.

Readers should note the points in the [HCC Documentation Guidelines](#) on the HCC documentation website.

#### **HCC Firmware Quick Start Guide**

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

#### **HCC Source Tree Guide**

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

#### **HCC USB Host Base System User Guide**

This document defines the USB host base system upon which the complete USB stack is built.

#### **HCC Embedded USB Host Audio Class Driver User Guide**

This is this document.

#### **HCC Embedded USB Host HID Class Driver User Guide**

This document describes the Human Interface Device (HID) module.

## 1.4. Change History

This section describes past changes to this manual.

- To view this manual as a PDF, see [USB Host PDFs](#).
- For the history of changes made to the package code itself, see [History: usbh\\_cd\\_audio](#).

The current version of this manual is 2.20.

Manual version	Date	Software version	Reason for change
2.20	2020-05-19	3.13	New document template. Added USBH_AUDIO_ERR_CL_BUF_SMALL return code.
2.10	2019-10-18	3.11	Improved error reporting information in API.
2.00	2019-02-13	3.11	First online release.



## 2. Source File List

The following sections describe all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the [HCC Source Tree Guide](#). All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

**Note:** Do not modify any files except the configuration files.

### API Header File

The file `src/api/api_usbh_audio.h` should be included by any application using the system. This is the only file that should be included by an application using this module. For details of the API functions, see [Application Programming Interface](#).

### Configuration Files

The following files in the directory `src/config` contain all the configurable parameters of the system. Configure these as required.

File	Description
<code>config_usbh_audio.h</code>	<a href="#">Audio configuration options.</a>
<code>config_usbh_audio_pcm.h</code>	<a href="#">Pulse Code Modulation (PCM) configuration options.</a>

### Version File

The file `src/version/ver_usbh_audio.h` contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

### Source Code Files

These files are in the directory `src/usb-host/class-drivers/audio`. **These files should only be modified by HCC.**

File	Description
<code>pcm.c</code>	PCM conversion module. This is required to play a stream with characteristics different from those that the audio device supports.
<code>pcm.h</code>	PCM header file.
<code>usbh_audio.c</code>	Main audio code.
<code>usbh_audio.h</code>	Audio header file.
<code>usbh_audio_dsc.h</code>	Audio descriptor-related information and macros.
<code>usbh_audio_hid.c</code>	Audio Human Interface Device (HID) module. This is for volume and mute controls.
<code>usbh_audio_hid.h</code>	Audio HID header file.

## 3. Configuration Options

This section lists the available configuration options and their default values.

### 3.1. Audio Configuration Options

Set the following audio options in the file `src/config/config_usbh_audio.h`.

#### **USBH\_AUDIO\_TASK\_STACK\_SIZE**

The size of the audio task stack. The default is 4096.

#### **USBH\_AUDIO\_MAX\_UNITS**

The maximum number of audio units. The default is 1.

#### **USBH\_AUDIO\_MAX\_STREAMS**

The maximum number of simultaneous play/record streams. The default is 2.

#### **USBH\_AUDIO\_MAX\_AS\_IFC**

The maximum number of audio streaming interfaces on a device. The default is 4.

#### **USBH\_AUDIO\_MAX\_AS\_ALT\_IFC**

The maximum number of alternate audio streaming interfaces on a device. Alternate interfaces belong to audio streaming interfaces. The default is 20.

#### **USBH\_AUDIO\_MAX\_NODES**

The maximum number of nodes. Nodes are required if an entity on the audio control interface connects to multiple entities, for example if an input terminal is connected to multiple feature units.

This number defines how many entities of this type can be present in a path from input terminal to output terminal. Generally the required number of nodes on a path is zero, but sometimes it is 1 or 2. Leaving this number at the default 6 provides a safe margin.

#### **USBH\_AUDIO\_DSC\_BUFFER\_SIZE**

The descriptor buffer size for an audio device. This buffer is used to store (part of) the descriptor of all the audio control entities, the audio streaming descriptor and the audio streaming format descriptor. After enumeration, links between AC entities are built in this buffer; this is required in order to find the correct path between input and output terminals when playing or recording. The default is 512.

#### **USBH\_AUDIO\_HID\_ENABLE**

Set this to enable audio Human Interface Device (HID) support. This is required for audio devices with mute, volume and other controls. The default is 0.

## USBH\_AUDIO\_FEEDBACK\_RATE

The default feedback time in ms (if a feedback channel is present). The default is 100.

## USBH\_AUDIO\_OPTIMIZE\_RECORD

Set this to enable optimize record. The default is 0.

**Note:** This feature can cause 0.2% inaccuracy in the worst case, but on slow systems it can be very useful.

If this is enabled then remainder samples are dropped when recording. When the required number of frames (up to USBH\_AUDIO\_MAX\_FRAME\_PER\_CLBUF) has arrived at one cluster buffer, the system calls the user write function. With this option enabled, this is called once with the total available data in the buffer. Otherwise, if the number of samples/frame can vary (for example, 44.1 kHz), it is called as many times as there are frames in the cluster buffer.

For example, assume a 44100 sample rate: 44 samples will be requested for 9 frames and 45 in the 10th, giving a gap of 1 sample for 9 frames.

## USBH\_AUDIO\_PCM\_CONVERT\_ENABLE

Keep the default of 1 to enable PCM conversion capability. This is required if **usbh\_audio\_play()** is called with stream parameters different from those the device supports.

## USBH\_AUDIO\_MAX\_TASK\_LOOP

The maximum number of loops in a transfer task for buffer reading/writing when in polled mode. The default is 16.

Cluster buffers are required to hold samples during streaming. This buffer is allocated at run time using the **hcc\_mem** module so **config\_hcc\_mem.h** must be configured correctly.

## USBH\_AUDIO\_CL\_BUF\_CNT

The number of cluster buffers for streaming. Cluster is the term used to represent one sample for all channels. For example, if there are 6 channels and 16 bits/sample then the size of one cluster is  $6 \times 2 = 12$  bytes. The default is 4.

## USBH\_AUDIO\_CL\_BUF\_SIZE

The size of a cluster buffer in bytes. The default is 3240.

## USBH\_AUDIO\_MAX\_FRAME\_PER\_CLBUF

The maximum number of USB frames in a cluster buffer for streaming. The default is 8.

## 3.2. PCM Configuration Options

Set the following Pulse Code Modulation (PCM) conversion options in the file **src/config/config\_usbh\_audio\_pcm.h**.

### **PCM\_MAX\_CONVERSIONS**

The maximum number of simultaneous conversions. The default is 2.

### **PCM\_PRECISION**

The conversion precision. If the sample rate differs, stretching or dropping of samples may be required. Increase this number for more precise conversion. The default is 2.

### **PCM\_BUF\_SIZE**

The size of the buffer which raw PCM data is read to. The default is 4096.

### **PCM\_USE\_MEMCPY**

Keep the default of 1 to use the internal PSP memcopy, **psp\_memcpy()**, or set this to 0 to copy directly from the code.

## 4. Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

### 4.1. Module Management

The functions are the following:

Function	Description
<b>usbh_audio_init()</b>	Initializes the module and allocates the required resources.
<b>usbh_audio_start()</b>	Starts the module.
<b>usbh_audio_stop()</b>	Stops the module.
<b>usbh_audio_delete()</b>	Deletes the module and releases the resources it used.

## usbh\_audio\_init

Use this function to initialize the class driver and allocate the required resources.

**Note:** You must call this before any other function.

### Format

```
int usbh_audio_init ( void )
```

### Arguments

None.

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_audio\_start

Use this function to start the module.

**Note:** You must call **usbh\_audio\_init()** before this function.

### Format

```
int usbh_audio_start ( void )
```

### Arguments

None.

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_ERR_RESOURCE	OS resource error.



## usbh\_audio\_stop

Use this function to stop the module.

### Format

```
int usbh_audio_stop ( void )
```

### Arguments

None.

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_ERR_RESOURCE	OS resource error.

## usbh\_audio\_delete

Use this function to delete the module and release the associated resources.

### Format

```
int usbh_audio_delete ( void )
```

### Arguments

None.

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_ERR_RESOURCE	OS resource error.

## 4.2. Device Management

The functions are the following:

Function	Description
<b>usbh_audio_change_mute()</b>	Toggles the current mute state.
<b>usbh_audio_change_volume()</b>	Changes the volume.
<b>usbh_audio_find_first_terminal()</b>	Gets the first terminal type for playing/recording.
<b>usbh_audio_find_next_terminal()</b>	Gets the next terminal type for playing/recording.
<b>usbh_audio_play()</b>	Starts audio play on a terminal.
<b>usbh_audio_stop_play()</b>	Stops playing on a terminal.
<b>usbh_audio_record()</b>	Starts audio recording on a terminal.
<b>usbh_audio_stop_record()</b>	Stops recording on a terminal.
<b>usbh_audio_present()</b>	Checks whether an audio device is connected.
<b>usbh_audio_state()</b>	Gets the current state of the terminal.
<b>usbh_audio_get_stream_info()</b>	Gets terminal stream information.
<b>usbh_audio_get_port_hdl()</b>	Gets the port handle.
<b>usbh_audio_register_ntf()</b>	Registers a notification function for a specified event type.

## usbh\_audio\_change\_mute

Use this function to toggle the current mute state.

### Format

```
int usbh_audio_change_mute ( t_terminal_info * p_tinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <b>usbh_audio_find_first_terminal()</b></li><li>• <b>usbh_audio_find_next_terminal()</b>.</li></ul>	<a href="#">t_terminal_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_NOTSUPPORTED	The operation is not supported.

## usbh\_audio\_change\_volume

Use this function to change the volume.

The physical volume change command is only sent if the streaming interface supports volume control and the stream is active (playing/recording is in progress).

### Format

```
int usbh_audio_change_volume (  
    t_terminal_info * p_tinf,  
    int volume )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <b>usbh_audio_find_first_terminal()</b></li><li>• <b>usbh_audio_find_next_terminal()</b>.</li></ul>	<a href="#">t_terminal_info</a> *
volume	The volume relative to the current volume (for example, 1000 or -1000).	int

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_NOTSUPPORTED	The operation is not supported.

## usbh\_audio\_find\_first\_terminal

Use this function to get the first terminal type for playing/recording.

This function fills a terminal information structure (p\_tinf) that can be used for any further audio functions. This returns a [terminal type](#).

### Format

```
int usbh_audio_find_first_terminal (
    t_usbh_unit_id    uid,
    uint8_t          mode,
    t_terminal_info * p_tinf,
    uint16_t *       p_terminal_type )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
mode	T_ST_PLAY or T_ST_RECORD.	uint8_t
p_tinf	On return, a pointer to the terminal information (allocated in user space).	<a href="#">t_terminal_info</a> *
p_terminal_type	On return, the type of terminal found for the specific mode.	uint16_t *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_NOT_PRESENT	There is no such terminal connected.

## usbh\_audio\_find\_next\_terminal

Use this function to get the next terminal type for playing/recording.

This function fills a terminal information structure (p\_tinf) that can be used for any further audio functions. This returns a [terminal type](#).

### Format

```
int usbh_audio_find_next_terminal (
    t_usbh_unit_id    uid,
    uint8_t           mode,
    t_terminal_info * p_tinf,
    uint16_t *        p_terminal_type )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
mode	T_ST_PLAY or T_ST_RECORD.	uint8_t
p_tinf	On return, a pointer to the terminal information (allocated in user space).	<a href="#">t_terminal_info</a> *
p_terminal_type	On return, the type of terminal found for the specific mode.	uint16_t *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_NOT_PRESENT	There is no such terminal connected.



## usbh\_audio\_play

Use this function to start audio play on a terminal.

### Format

```
int usbh_audio_play (
    t_terminal_info * p_tinf,
    t_stream_info * p_sinf,
    t_rw_data_info * p_rwinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"> <li>• <b>usbh_audio_find_first_terminal()</b></li> <li>• <b>usbh_audio_find_next_terminal()</b>.</li> </ul>	<a href="#">t_terminal_info</a> *
p_sinf	A pointer to source/destination stream information.	<a href="#">t_stream_info</a> *
p_rwinf	A pointer to stream read/write information.	<a href="#">t_rw_data_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_ALLOCATION	A buffer cannot be allocated.
USBH_AUDIO_ERR_BADFORMAT	The input/output stream is not supported by the device. This may be because the sample rate is not processable, the number of channels required, and so on.
USBH_AUDIO_ERR_BUSY	The audio device is busy or there is no free stream to complete the operation.

## usbh\_audio\_stop\_play

Use this function to stop playing on a terminal.

### Format

```
int usbh_audio_stop_play ( t_terminal_info * p_tinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <b>usbh_audio_find_first_terminal()</b></li><li>• <b>usbh_audio_find_next_terminal()</b>.</li></ul>	<a href="#">t_terminal_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_audio\_record

Use this function to start audio recording on a terminal.

### Format

```
int usbh_audio_record (
    t_terminal_info * p_tinf,
    t_stream_info * p_sinf,
    t_rw_data_info * p_rwinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"> <li>• <b>usbh_audio_find_first_terminal()</b></li> <li>• <b>usbh_audio_find_next_terminal()</b>.</li> </ul>	<a href="#">t_terminal_info</a> *
p_sinf	A pointer to source/destination stream information.	<a href="#">t_stream_info</a> *
p_rwinf	A pointer to stream read/write information.	<a href="#">t_rw_data_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERR_ALLOCATION	A buffer cannot be allocated.
USBH_AUDIO_ERR_BADFORMAT	The input/output stream is not supported by the device. This may be because the sample rate is not processable, the number of channels required, and so on.
USBH_AUDIO_ERR_BUSY	The audio device is busy or there is no free stream to complete the operation.

## usbh\_audio\_stop\_record

Use this function to stop recording on a terminal.

### Format

```
int usbh_audio_stop_record ( t_terminal_info * p_tinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <b>usbh_audio_find_first_terminal()</b></li><li>• <b>usbh_audio_find_next_terminal()</b>.</li></ul>	<a href="#">t_terminal_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_audio\_present

Use this function to check whether an audio device is connected.

### Format

```
int usbh_audio_present ( t_usbh_unit_id uid )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

### Return Values

Return value	Description
0	No audio device is connected.
1	An audio device is connected.

## usbh\_audio\_state

Use this function to get the current state of the terminal.

### Format

```
int usbh_audio_state (
    t_terminal_info * p_tinf,
    uint8_t * p_state )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <a href="#">usbh_audio_find_first_terminal()</a></li><li>• <a href="#">usbh_audio_find_next_terminal()</a></li></ul>	<a href="#">t_terminal_info</a> *
p_state	A bitmask showing the <a href="#">terminal state</a> .	uint8_t *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_audio\_get\_stream\_info

Use this function to get terminal stream information.

Fill the *t\_stream\_info* structure with the desired streaming parameters. The system changes the values to the closest matching values that can be used on the streaming interface.

### Format

```
int usbh_audio_get_stream_info (  
    t_terminal_info * p_tinf,  
    t_stream_info *  sinf )
```

### Arguments

Parameter	Description	Type
p_tinf	A pointer to the terminal information structure. To get this, use one of the following functions: <ul style="list-style-type: none"><li>• <b>usbh_audio_find_first_terminal()</b></li><li>• <b>usbh_audio_find_next_terminal()</b>.</li></ul>	<a href="#">t_terminal_info</a> *
sinf	A pointer to the streaming information. On return this holds the possible (closest) parameters.	<a href="#">t_stream_info</a> *

### Return Values

Return value	Description
USBH_AUDIO_SUCCESS	Successful execution.
USBH_AUDIO_ERROR	Operation failed.



## usbh\_audio\_get\_port\_hdl

Use this function to get the port handle.

### Format

```
t_usbh_port_hdl usbh_audio_get_port_hdl ( t_usbh_unit_id uid )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

### Return Values

Return value	Description
The port handle	Successful execution.
USBH_PORT_HDL_INVALID	Invalid port handle.
Else	See <a href="#">Error Codes</a> .

## usbh\_audio\_register\_ntf

Use this function to register a notification function for a specified event type.

When a device is connected or disconnected, the notification function is called.

**Note:** It is the user's responsibility to provide any notification functions required by the application. Providing such functions is optional.

### Format

```
int usbh_audio_register_ntf (  
    t_usbh_unit_id    uid,  
    t_usbh_ntf        ntf,  
    t_usbh_ntf_fn     ntf_fn )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
ntf	The <a href="#">notification type</a> .	t_usbh_ntf
ntf_fn	The notification function to use when an event occurs.	<a href="#">t_usbh_ntf_fn</a>

### Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERR_INVALID	The ID is invalid.
Else	See <a href="#">Error Codes</a> .

## 4.3. Error Codes

The first table below shows the meaning of the audio-specific error codes, the second table shows the generic USB host error codes.

### Audio-specific Error Codes

Return Code	Value	Description
USBH_AUDIO_SUCCESS	0	Successful execution.
USBH_AUDIO_PENDING	1	The operation is not complete yet.
USBH_AUDIO_ERR_NOTSUPPORTED	2	The operation is not supported.
USBH_AUDIO_ERR_BADFORMAT	3	The input/output stream is not supported by the device. This may be because the sample rate is not processable, the number of channels required, and so on.
USBH_AUDIO_ERR_NOTPRESENT	4	No device is connected.
USBH_AUDIO_ERR_BUSY	5	The audio device is busy or there is no free stream to complete the operation.
USBH_AUDIO_ERR_ALLOCATION	6	Allocation error. This is returned if the cluster buffer cannot be allocated for playing/recording. This buffer is allocated at runtime from a pool that can be used directly by the host controller. In this way the system can avoid making extra copies before/after transmitting/receiving data.
USBH_AUDIO_ERR_CL_BUF_SMALL	7	The cluster buffer is too small. <code>USBH_AUDIO_CL_BUF_SIZE</code> ( <b>config_usbh_audio.h</b> ) is not large enough to handle the requested playback on the connected audio device. If this happens, increase <code>USBH_AUDIO_CL_BUF_SIZE</code> .
USBH_AUDIO_ERROR	8	General error occurred.

## Generic USB Host Error Codes

The generic USB host class driver error codes are as follows:

Return Code	Value	Description
USBH_SUCCESS	0	Successful execution.
USBH_SHORT_PACKET	1	IN transfer completed with short packet.
USBH_PENDING	2	Transfer still pending.
USBH_ERR_BUSY	3	Another transfer in progress.
USBH_ERR_DIR	4	Transfer direction error.
USBH_ERR_TIMEOUT	5	Transfer timed out.
USBH_ERR_TRANSFER	6	Transfer failed to complete.
USBH_ERR_TRANSFER_FULL	7	Cannot process more transfers.
USBH_ERR_SUSPENDED	8	Host controller is suspended.
USBH_ERR_HC_HALTED	9	Host controller is halted.
USBH_ERR_REMOVED	10	Transfer finished due to device removal.
USBH_ERR_PERIODIC_LIST	11	Periodic list error.
USBH_ERR_RESET_REQUEST	12	Reset request during enumeration.
USBH_ERR_RESOURCE	13	OS resource error.
USBH_ERR_INVALID	14	Invalid identifier/type (HC, EP HDL, and so on).
USBH_ERR_NOT_AVAILABLE	15	Item not available.
USBH_ERR_INVALID_SIZE	16	Invalid size.
USBH_ERR_NOT_ALLOWED	17	Operation not allowed.
USBH_ERROR	18	General error.

## 4.4. Types and Definitions

This section describes the notification function, the standard notification codes, and the terminal and stream types that are defined in the API header file.

### t\_usbh\_ntf\_fn

The **t\_usbh\_ntf\_fn** definition specifies the format of the notification function. It is defined in the USB host base system in the file **api\_usb\_host.h**.

Standard notification codes are defined in the USB host base system in the file **api\_usb\_host.h**. There are no additional notification codes for this module.

#### Format

```
int ( * t_usbh_ntf_fn )(
    t_usbh_unit_id  uid,
    t_usbh_ntf      ntf )
```

#### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
ntf	The <a href="#">notification code</a> .	t_usbh_ntf

### Notification Codes

The standard notification codes shown below are defined in the USB host base system in the file **api\_usb\_host.h**. This module has no specific notification codes of its own.

Notification	Value	Description
USBH_NTF_CONNECT	1	Connection notification code.
USBH_NTF_DISCONNECT	2	Disconnection notification code.

## Terminal States

The possible terminal states are as follows:

State	Value	Description
T_ST_NONE	0	Inactive.
T_ST_PLAY	1	Playing.
T_ST_RECORD	2	Recording.
T_ST_END	4	Finished.

## Terminal Group Macro

The macro used to get the terminal group is:

Return Value	Default Setting
TERMINAL_GROUP( v )	( v ) & 0xFF00

## Stream Type

There is just one stream type:

Return Value	Default Setting	Description
AUDIO_TYPE_PCM	1	The only supported stream type.

## Invalid States

There are two invalid states, as follows:

Return Value	Default Setting	Description
INVALID_TERMINAL	0xFF	Invalid terminal type.
INVALID_NODE	0xFF	Invalid node type.

## t\_terminal\_info

The *t\_terminal\_info* structure describes a terminal:

Element	Type	Description
audio_device_pos	t_usbh_unit_id	Audio device position.
stream_ifc_pos	uint8_t	The stream interface position of the audio device.
usage_count	uint8_t	The usage count.
terminal	uint8_t	The terminal to search for.
i_terminal	uint8_t	The input terminal.
o_terminal	uint8_t	The output terminal.
node_cnt	uint8_t	The number of nodes in the nodes buffer.
nodes [USBH_AUDIO_MAX_NODES]	uint8_t	The nodes.

## t\_rw\_data\_info

The *t\_rw\_data\_info* structure describes read/write data access information:

Element	Type	Description
rw	t_rw_data	The read/write function.
par	void *	The read/write function parameter.
close	t_close_data	The close function.

## t\_stream\_info

The *t\_stream\_info* structure describes stream information:

Element	Type	Description
type	uint8_t	The <a href="#">stream type</a> .
sr	uint32_t	The sampling rate.
bps	uint8_t	The bits/sample.
channels	uint8_t	The number of channels.



## 4.5. Terminal Types

To find out which group a terminal belongs to, you can use the `TERMINAL_GROUP` macro. Provide the terminal type as a parameter to get a match to any `TGROUP_....`

For example:

```
TERMINAL_GROUP( terminal_type ) == TGROUP_OT
```

### Input Terminal Types

The input terminal types are as follows:

Return Value	Default Setting	Description
TGROUP_IT	0x0200	
TTYTYPE_IT_UNDEF	0x0200	Undefined type.
TTYTYPE_IT_MIC	0x0201	Microphone.
TTYTYPE_IT_DESKTOP_MIC	0x0202	Desktop microphone.
TTYTYPE_IT_PERSONAL_MIC	0x0203	Personal microphone.
TTYTYPE_IT_OMNI_DIR_MIC	0x0204	Omni-directional microphone.
TTYTYPE_IT_MIC_ARRAY	0x0205	Microphone array.
TTYTYPE_IT_PROC_MIC_ARRAY	0x0206	Processing microphone array.

### Output Terminal Types

The output terminal types are as follows:

Return Value	Default Setting	Description
TGROUP_OT	0x0300	
TTYTYPE_OT_UNDEF	0x0300	Undefined type.
TTYTYPE_OT_SPEAKER	0x0301	Speaker.
TTYTYPE_OT_HEADPHONES	0x0302	Headphones.
TTYTYPE_OT_HEAD_MOUNTED_DISP	0x0303	Head mounted display audio.
TTYTYPE_OT_DESKTOP_SPEAKER	0x0304	Desktop speaker.
TTYTYPE_OT_ROOM_SPEAKER	0x0305	Room speaker.
TTYTYPE_OT_COMM_SPEAKER	0x0306	Communication speaker.
TTYTYPE_OT_LOW_FREQ_SPEAKER	0x0307	Low frequency effects speaker.

## Bidirectional Terminal Types

The bidirectional terminal types are as follows:

Return Value	Default Setting	Description
TGROUP_BIDIR	0x0400	
TTYTYPE_BIDIR_UNDEF	0x0400	Undefined type.
TTYTYPE_BIDIR_HANDSET	0x0401	Handset.
TTYTYPE_BIDIR_HEADSET	0x0402	Headset.
TTYTYPE_BIDIR_SPKPHONE_NE	0x0403	No echo reduction speakerphone.
TTYTYPE_BIDIR_SPKPHONE_ES	0x0404	Echo-suppressing speakerphone.
TTYTYPE_BIDIR_SPKPHONE_EC	0x0405	Echo-canceling speakerphone.

## Telephony Terminal Types

The telephony terminal types are as follows:

Return Value	Default Setting	Description
TGROUP_TPH	0x0500	
TTYTYPE_TPH_UNDEF	0x0500	Undefined type.
TTYTYPE_TPH_PHONE_LINE	0x0501	Phone line.
TTYTYPE_TPH_TELEPHONE	0x0502	Telephone.
TTYTYPE_TPH_DOWN_LINE_PHONE	0x0503	Down line phone.

## External Terminal Types

The external terminal types are as follows:

Return Value	Default Setting	Description
TGROUP_EXT	0x0600	
TTYPE_EXT_UNDEF	0x0600	Undefined type.
TTYPE_EXT_ANALOG	0x0601	Analog connector.
TTYPE_EXT_DIGITAL	0x0602	Digital audio interface.
TTYPE_EXT_LINE	0x0603	Line connector.
TTYPE_EXT_LEGACY	0x0604	Legacy audio connector.
TTYPE_EXT_SPDIF	0x0605	S/PDIF interface.
TTYPE_EXT_1394_DA	0x0606	IEEE 1394 DA stream.
TTYPE_EXT_1394_DV	0x0607	IEEE 1394 DV stream soundtrack.
TTYPE_EXT_ADAT	0x0608	ADAT Lightpipe.
TTYPE_EXT_TDIF	0x0609	Tascam Digital Interface (TDIF).
TTYPE_EXTMADI	0x060A	Multiple Audio Digital Interface (MADI).

## Embedded Function Terminal Types

Return Value	Default Setting	Description
TGROUP_EMB	0x0700	
TTYPE_EMB_UNDEF	0x0700	Undefined type.
TTYPE_EMB_LEVEL_CNOISE	0x0701	Level calibration noise.
TTYPE_EMB_EQUALIZATION	0x0702	Equalization noise.
TTYPE_EMB_CD_PLAYER	0x0703	CD player.
TTYPE_EMB_DAT	0x0704	DAT.
TTYPE_EMB_DCC	0x0705	DCC.
TTYPE_EMB_COMPRESSED	0x0706	Compressed audio player.
TTYPE_EMB_ANALOG_TAPE	0x0707	Analog tape
TTYPE_EMB_PHONOGRAPH	0x0708	Phonograph
TTYPE_EMB_VCR	0x0709	VCR audio
TTYPE_EMB_VIDEO_DISC	0x070A	Video disc audio
TTYPE_EMB_DVD	0x070B	DVD audio.
TTYPE_EMB_TV_TUNER	0x070C	TV tuner audio.
TTYPE_EMB_SATELLITE_REC	0x070D	Satellite receiver audio.
TTYPE_EMB_CABLE_TUNER	0x070E	Cable tuner audio.
TTYPE_EMB_DSS	0x070F	DSS audio.
TTYPE_EMB_RADIO_REC	0x0710	Radio receiver.
TTYPE_EMB_RADIO_TR	0x0711	Radio transmitter.
TTYPE_EMB_MULTI_TRACK	0x0712	Multi-track recorder.
TTYPE_EMB_SYNTHESIZER	0x0713	Synthesizer.
TTYPE_EMB_PIANO	0x0714	Piano.
TTYPE_EMB_GUITAR	0x0715	Guitar.
TTYPE_EMB_DRUMS	0x0716	Drums/rhythm.
TTYPE_EMB_OTHER	0x0717	Other musical instrument.

## 5. Integration

This section specifies the elements of this package that need porting, depending on the target environment.

### 5.1. OS Abstraction Layer (OAL)

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The module uses the following OAL components:

OAL Resource	Number Required
Tasks	1
Mutexes	1
Events	1

### 5.2. PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The module makes use of the following standard PSP functions:

Function	Package	Component	Description
<b>psp_memcpy()</b>	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
<b>psp_memset()</b>	psp_base	psp_string	Sets the specified area of memory to the defined value.

The module makes use of the following standard PSP macros:

Macro	Package	Component	Description
PSP_RD_LE16	psp_base	psp_endianness	Reads a 16 bit value stored as little-endian from a memory location.
PSP_RD_LE24	psp_base	psp_endianness	Reads a 24 bit value stored as little-endian from a memory location.
PSP_WR_LE16	psp_base	psp_endianness	Writes a 16 bit value stored as little-endian to a memory location.
PSP_WR_LE24	psp_base	psp_endianness	Writes a 24 bit value stored as little-endian to a memory location.

## 6. Sample Code

This section shows example code for the class driver.

### 6.1. Initialization

This example shows the code used to initialize a USB host with the class driver.

```
/*
** Initialize USB host with Audio class driver.
*/

int usb_host_init ( void )
{
    int rc;
    rc = hcc_mem_init();

    /* Initialize the USB host module */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_init();
    }

    /* Initialize the specific USB host controller */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_hc_init( 0, usbh_stm32uh_hc, 0 );
    }

    /* Initialize the Audio class driver module */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_audio_init();
    }

    /* Start the Audio class driver */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_audio_start();
    }

    /* Start the USB host stack */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_start(); /* Start the USB host */
    }

    return rc;
} /* usb_host_init */
```

## 7. Version

Version 2.20

For use with USB Host Audio Class Driver versions 3.13 and above