ANALOG DEVICES

USB AUDIO CLASS DEVICE DRIVER

DATE: FEBRUARY 09, 2009

Table of Contents

List of Tables

Document Revision History

Date	Description of Changes
February 09, 2009	Initial Version

,

Table 1 – Revision History

1. Overview

This document describes the usage of the USB Audio Class Driver. This driver has been tested on the BF526, BF527, BF548 EZKit Lite development boards.

The EZKit is connected to a PC via a USB cable. Once connected the PC detects the device, enumerates it. When enumeration is completed the device appears as an USB audio device.

2. Files

The files listed below comprise the device driver API and source files.

2.1 Include Files

The driver sources include the following include files:

- <services/services.h> This file contains all definitions, function prototypes etc. for all the System Services.
- <drivers/adi_dev.h> This file contains all definitions, function prototypes etc. for the Device Manager and general device driver information.
- <drivers\usb\usb_core\adi_usb_objects.h> This file contains various device objects that are buliding blocks for the usb system.
- <drivers\usb\usb_core\adi_usb_core.h> This file contains USB core layer definitions.
- <drivers/usb/class/pheripheral/audio/adi_usb_audio_class.h> This file contains all commands, event and return codes specific to the Audio Class Driver.
- <drivers/usb/class/pheripheral/audio/adi_usb_audio.h> This file contains the USB audio structure definitions

2.2 Source Files

The driver sources are contained in the following files, as located in the default installation directory:

- <Blackfin/lib/src/drivers/usb/class/pheripheral/audio/adi_usb_audio_class.c> This file contains source code for the Audio Class Driver. All code is written in 'C'.

3. Lower Level Drivers

The Audio Class Driver is part of the ADI USB stack. Please see the USB Core and USB Device driver documentation or more information.

4. Resources Required

Device drivers typically consume some amount of system resources. This section describes the resources required by the device driver.

Unless explicitly noted in the sections below, this device driver uses the System Services to access and control any required hardware. The information in this section may be helpful in determining the resources this driver requires, such as the number of interrupt handlers or number of DMA channels etc., from the System Services.

Because dynamic memory allocations are not used in the Device Drivers or System Services, all memory used by the Device Drivers and System Services must be supplied by the application. The Device Drivers and System Services supply macros that can be used by the application to size the amount of base memory and/or the amount of incremental memory required to support the needed functionality.

4.1 Interrupts

The USB Audio Device Driver does not use any Interrupt resources.

4.2 DMA

The USB Audio Device Driver does not use any DMA resources.

4.3 Timers

The USB Audio Device Driver does not use any Timer resources.

4.4 Real-Time Clock

The USB Audio Device Driver does not use any RTC resources.

4.5 Programmable Flags

The USB Audio Device Driver does not use any Programmable Flag resources.

4.6 Pins

The USB Audio Device Driver does not use any Pin resources.

5 Supported Features of the Device Driver

This section describes what features are supported by the device driver.

5.1 Directionality

The driver supports the dataflow directions listed in the table below.

ADI_DEV_DIRECTION	Description
ADI_DEV_ DIRECTION_BIDIRECTIONAL	Supports both the reception of data and transmission of data through the device.

Table 2 – Supported Dataflow Directions

5.2 Dataflow Methods

The driver supports the dataflow methods listed in the table below.

ADI_DEV_MODE	Description
ADI_DEV_MODE_CHAINED	Supports the chained buffer method

Table 3 – Supported Dataflow Methods

5.3 Buffer Types

The driver supports the buffer types listed in the table below.

- ADI_DEV_1D_BUFFER
  - Linear one-dimensional buffer
  - pAdditionalInfo – ignored

5.4 Command IDs

This section enumerates the commands that are supported by the driver. The commands are divided into three sections. The first section describes commands that are supported directly by the Device Manager. The next section describes common commands that the driver supports. The remaining section describes driver specific commands.

Commands are sent to the device driver via the adi_dev_Control() function. The adi_dev_Control() function accepts three arguments:

- DeviceHandle – This parameter is a ADI_DEV_DEVICE_HANDLE type that uniquely identifies the device driver. This handle is provided to the client in the adi_dev_Open() function call.
- CommandID – This parameter is a u32 data type that specifies the command ID.
- Value – This parameter is a void * whose value is context sensitive to the specific command ID.

The sections below enumerate the command IDs that are supported by the driver and the meaning of the Value parameter for each command ID.

5.4.1 Device Manager Commands

The commands listed below are supported and processed directly by the Device Manager. As such, all device drivers support these commands.

- ADI_DEV_CMD_TABLE
  - Table of command pairs being passed to the driver
  - Value – ADI_DEV_CMD_VALUE_PAIR *
- ADI_DEV_CMD_END
  - Signifies the end of a command pair table
  - Value – ignored
- ADI_DEV_CMD_PAIR
  - Single command pair being passed
  - Value – ADI_DEV_CMD_PAIR *
- ADI_DEV_CMD_SET_SYNCHRONOUS
  - Enables/disables synchronous mode for the driver
  - Value – TRUE/FALSE

5.4.2 Common Commands

The command IDs described in this section are common to many device drivers. The list below enumerates all common command IDs that are supported by this device driver.

- ADI_DEV_CMD_SET_DATAFLOW_METHOD
  - Specifies the dataflow method the device is to use. The list of dataflow types supported by the device driver is specified in section 5.2.
  - Value – ADI_DEV_MODE enumeration

- ADI_DEV_CMD_SET_DATAFLOW
  - Enables/disables dataflow through the device
  - Value – TRUE/FALSE

5.4.3 Device Driver Specific Commands

The command IDs listed below are supported and processed by the device driver. These command IDs are unique to this device driver.

- ADI_USB_CMD_ENABLE_USB
  - This command enables USB communication with the host PC.
  - Value – NULL

- ADI_USB_CMD_CLASS_SET_CONTROLLER_HANDLE
  - This command is used to pass the controller device handle to the class driver.
  - Value – This is a void * whose value is the handle of the controller device.

- ADI_USB_CMD_CLASS_CONFIGURE
  - This command is to configure the class driver.
  - Value – This value should be zero.

- ADI_USB_AUDIO_CMD_IS_DEVICE_CONFIGURED
  - This command is called to determine if the device was configured by the host PC.
  - Value – This parameter is a pointer to a u32 which will upon completion contain the returned value TRUE for device configured or FALSE device not configured.

- ADI_USB_AUDIO_CMD_SET_VID
  - This command is used to set a user defined Vendor ID.
  - Value – This value should be the user defined Vendor ID for this device.
  - NOTE: THIS VALUE CAN BE OBTAINED FROM www.usb.org IF YOUR COMPANY DOES NOT ALREADY HAVE A VALID VENDOR ID.

- ADI_USB_AUDIO_CMD_SET_PID
  - This command is used to set a user defined Product ID.
  - Value – This value should be the user defined Product ID.

- ADI_USB_AUDIO_CMD_SET_PRODUCT_STRING
  - This command is used to set the user defined Product String.
  - Value – This should be a user defined Product String.

- ADI_USB_AUDIO_CMD_SET_ MANUFACTURER_STRING
  - This command is used to set the user defined Manufacturer String.
  - Value – This should be a user defined Manufacturer String.

- ADI_USB_AUDIO_CMD_SET_RX_BUFFER
  - This command is used to pass the working USB buffer down to the Audio Class Driver.
  - Value – This a void * whose value is the address of the buffer.

- ADI_USB_AUDIO_CMD_SET_PLAYBACK_CHANNELS
  - This command set the number of audio channels supported; the default value is 2 channels.
  - Value – This value can be 2, 4 or 6 channels.

- ADI_USB_AUDIO_CMD_DISABLE_MIXER
  - This command disables the internal Audio mixer structure. If this is disabled Line-In and or MIC will not appear in the analog out mixer.
  - Value – TRUE allow mixer and connections, FALSE disable mixer and connections. If true the Line-In control will appear in the analog out mixer.

- ADI_USB_AUDIO_CMD_MIC_IN_ADC_MIXER
  - If the mixer is enabled this will add the MIC control to the analog out mixer.
  - Value – TRUE add MIC control to the analog out mixer, FALSE MIC control does not appear in the analog out mixer.

- ADI_USB_AUDIO_CMD_ENABLE_MIC_RECORD_CTL
  - This command enables the MIC record control
  - Value – TRUE enable MIC control, FALSE disable MIC record control.

- ADI_USB_AUDIO_CMD_ENABLE_LINEIN_RECORD_CTL
  - This command enables the Line-In record control
  - Value – TRUE enable Line-In control, FALSE disable Line-In record control.

- ADI_USB_AUDIO_CMD_SET_PBK_SAMPLE_RATE_48K_ONLY
  - This command sets class driver to support 48Khz playback ONLY
  - Value - NULL

- ADI_USB_AUDIO_CMD_SET_PBK_SAMPLE_RATE_96K_ONLY
  - This command sets class driver to support 96Khz playback ONLY
  - Value - NULL

- ADI_USB_AUDIO_CMD_SET_PBK_SAMPLE_RATE_96K_48K
  - This command sets class driver to support 96Khz or 48Khz playback ONLY
  - Value - NULL

- ADI_USB_AUDIO_CMD_SET_MIC_RECORD_SELECT
  - This command sets the MIC as the default record control if the MIC record control is enabled.
  - Value - NULL

- ADI_USB_AUDIO_CMD_SET_LINEIN_RECORD_SELECT
  - This command sets the Line-In as the default record control if the Line-In record control is enabled.
  - Value - NULL

- ADI_USB_AUDIO_CMD_RETURN_CTL_DATA_1
  - This command is used to write one byte to the control channel Endpoint Zero.
  - Value - Control data value to return.

- ADI_USB_AUDIO_CMD_ RETURN_CTL_DATA_2
  - This command is used to write two bytes to the control channel Endpoint Zero.
  - Value - Control data value to return.

- ADI_USB_AUDIO_CMD_SET_16BIT_PBK_RESOLUTION
  - This command is used to set the of effectively used bits from the available bits in an audio subframe to 16 bits.
  - Value - NULL.

- ADI_USB_AUDIO_CMD_ SET_32BIT_PBK_RESOLUTION
  - This command is used to set the of effectively used bits from the available bits in an audio subframe to 32 bits.
  - Value - NULL.

5.5 Callback Events

This Audio Class driver sends callback events back to the client application. This section defines the events handled by the application.

- ADI_USB_EVENT_DATA_TX
  - Notifies application callback function that the data sent to the host has completed. In this case this event happens when the record buffer has successfully been transmitted to the host.
  - Value – Not used.

- ADI_USB_EVENT_DISCONNECT
  - Notifies application callback function that the usb cable has been unplugged.
  - Value – Not used.

- ADI_DEV_EVENT_BUFFER_PROCESSED
  - Notifies application callback function that we have a buffer filled with data packets from the host. The application if needed can process this buffer before handing it off to the codec.
  - Value – data buffer

- ADI_USB_AUDIO_EVENT_PLAYBACK_STARTED
  - Notifies application callback function that audio playback has started. The application should enable data flow on the audio codec.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_PLAYBACK_STOPPED
  - Notifies application callback function that audio playback has stopped. The application should disable data flow on the audio codec.
  - Value – Not used.
  -

- ADI_USB_AUDIO_EVENT_RECORD_STARTED
  - Notifies application callback function that audio record has started. The application should enable data flow on the audio codec.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_RECORD_STOPPED
  - Notifies application callback function that audio record has stopped. The application should disable data flow on the audio codec.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_SET_MASTER_VOLUME
  - Notifies application callback function that the master volume has been changed. The application should update the audio codec master volume register.
  - Value – master volume value.

- ADI_USB_AUDIO_EVENT_SET_MASTER_MUTE
  - Notifies application callback function that the master mute value has changed. The application should update the audio codec master mute register.
  - Value – 0 - Un-Mute, 1 - Mute.

- ADI_USB_AUDIO_EVENT_SET_LINEIN_VOLUME
  - Notifies application callback function that the line-in volume has changed. The application should update the audio codec line-in volume register.
  - Value – line-in volume value.

- ADI_USB_AUDIO_EVENT_SET_LINEIN_MUTE
  - Notifies application callback function that the line-in mute value has changed. The application should update the audio codec line-in mute register.
  - Value – 0 - Un-Mute, 1 - Mute.

- ADI_USB_AUDIO_EVENT_SET_MIC_VOLUME
  - Notifies application callback function that the microphone volume has changed. The application should update the audio codec microphone volume register.
  - Value – mic volume value.

- ADI_USB_AUDIO_EVENT_SET_MIC_MUTE
  - Notifies application callback function that the microphone mute value has changed. The application should update the audio codec microphone mute register.
  - Value – 0 - Un-Mute, 1 - Mute.

- ADI_USB_AUDIO_EVENT_SET_LINEIN_SELECT
  - Notifies application callback function that line-in is selected as the record source. The application should update the audio codec record select register.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_SET_MIC_SELECT
  - Notifies application callback function that microphone is selected as the record source. The application should update the audio codec record select register.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_SET_RECORD_MASTER_GAIN
  - Notifies application callback function that the record gain has changed. The application should update the audio codec record master gain register.
  - Value – record gain value.

- ADI_USB_AUDIO_EVENT_GET_MASTER_VOLUME
  - Notifies application callback function that the host is requesting the master volume value. The application should get the master volume value from the audio codec and write it to the host.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_GET_MASTER_MUTE
  - Notifies application callback function that the host is requesting the master mute value. The application should get the master mute value from the audio codec and write it to the host.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_GET_LINEIN_VOLUME
  - Notifies application callback function that the host is requesting the line-in volume value. The application should get the line-in volume value from the audio codec and write it to the host.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_GET_LINEIN_MUTE
  - Notifies application callback function that the host is requesting the line-in mute value. The application should get the line-in mute value from the audio codec and write it to the host.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_GET_MIC_VOLUME
  - Notifies application callback function that the host is requesting the microphone volume value. The application should get the microphone volume value from the audio codec and write it to the host.
  - Value – Not used.

- ADI_USB_AUDIO_EVENT_GET_MIC_MUTE
  - Notifies application callback function that the host is requesting the microphone mute value. The application should get the microphone mute value from the audio codec and write it to the host.
  - Value – Not used.

5.5.1 Device Driver Specific Events

The events listed below are supported and processed by the Audio Class device driver.

- **Control channel events (endpoint zero)**

  - ADI_USB_EVENT_SET_CONFIG
    - Notifies callback function that audio device enumeration by the host is complete. Device is ready to use. This is an Endpoint Zero event.
    - Value – Not used.

  - ADI_USB_EVENT_SET_INTERFACE
    - Notifies callback function that the host is requesting an interface change. This usually happens when the host is starting or stopping playback or record. This is an Endpoint Zero event.
    - Value – this is the interface number.

  - ADI_USB_EVENT_SETUP_PKT
    - This is a device setup packet event. Here we intercept and process audio specific setup packets.
    - Value – buffer that contains the audio setup packet.

  - ADI_USB_EVENT_VBUS_TRUE
    - Notifies callback function that the cable is unplugged.
    - Value - This value is the CallbackParameter value that was supplied in the buffer that was passed.

  - ADI_USB_EVENT_RESUME
    - Notifies callback function that host is initiating a resume.
    - Value - This value is the CallbackParameter value that was supplied in the buffer that was passed.

- ADI_USB_EVENT_SUSPEND
  - Notifies callback function that host is initiating a suspend.
  - Value - This value is the CallbackParameter value that was supplied in the buffer that was passed.

- ADI_USB_EVENT_DISCONNECT
  - Notifies callback function that host is initiating a disconnect due to the USB cable being unplugged. The Audio Class driver passes this callback up to the application. It also clears all of its active endpoint buffer queues.
  - Value – Not used

- ADI_USB_EVENT_RX_COMPLETE
  - Notifies callback function that host is sending data via the control channel.
  - Value - buffer containing the control information to process.

- **Standard endpoint events**

- ADI_USB_EVENT_DATA_TX
  - Notifies callback function that transmit has completed. This is passed up to the application for processing.
  - Value – Not used.

- ADI_USB_EVENT_DATA_RX
  - Notifies callback function that we have received a data packet from the host. The packet is then added to our buffer queue.
  - Value – buffer containing data sent from the host.

- ADI_USB_EVENT_PKT_RCVD_NO_BUFFER
  - Notifies callback function that we have received a data packet from the host there is no buffer setup fro it. This is passed up to the application for processing.
  - Value – Not used.

5.6 Return Codes

All API functions of the device driver return status indicating either successful completion of the function or an indication that an error has occurred. This section enumerates the return codes that the device driver is capable of returning to the client. A return value of ADI_DEV_RESULT_SUCCESS indicates success, while any other value indicates an error or some other informative result. The value ADI_DEV_RESULT_SUCCESS is always equal to the value zero. All other return codes are a non-zero value.

The return codes are divided into two sections. The first section describes return codes that are common to many device drivers. The next section describes driver specific return codes. The client should prepare to process each of the return codes described in these sections.

Typically, the application should check the return code for ADI_DEV_RESULT_SUCCESS, taking appropriate corrective action if ADI_DEV_RESULT_SUCCESS is not returned. For example:

```
if (adi_dev_Xxxx(…) == ADI_DEV_RESULT_SUCCESS) {
      // normal processing
} else {
      // error processing
}
```

5.6.1 Common Return Codes

The return codes described in this section are common to many device drivers. The list below enumerates all common return codes that are supported by this device driver.

- ADI_DEV_RESULT_SUCCESS
  - The function executed successfully.
- ADI_DEV_RESULT_NOT_SUPPORTED
  - The function is not supported by the driver.
- ADI_DEV_RESULT_DEVICE_IN_USE
  - The requested device is already in use.
- ADI_DEV_RESULT_NO_MEMORY
  - There is insufficient memory available.
- ADI_DEV_RESULT_BAD_DEVICE_NUMBER
  - The device number is invalid.
- ADI_DEV_RESULT_DIRECTION_NOT_SUPPORTED
  - The device cannot be opened in the direction specified.
- ADI_DEV_RESULT_BAD_DEVICE_HANDLE
  - The handle to the device driver is invalid.
- ADI_DEV_RESULT_BAD_MANAGER_HANDLE
  - The handle to the Device Manager is invalid.
- ADI_DEV_RESULT_BAD_PDD_HANDLE
  - The handle to the physical driver is invalid.
- ADI_DEV_RESULT_INVALID_SEQUENCE
  - The action requested is not within a valid sequence.
- ADI_DEV_RESULT_ATTEMPTED_READ_ON_OUTBOUND_DEVICE
  - The client attempted to provide an inbound buffer for a device opened for outbound traffic only.
- ADI_DEV_RESULT_ATTEMPTED_WRITE_ON_INBOUND_DEVICE
  - The client attempted to provide an outbound buffer for a device opened for inbound traffic only.
- ADI_DEV_RESULT_DATAFLOW_UNDEFINED
  - The dataflow method has not yet been declared.
- ADI_DEV_RESULT_DATAFLOW_INCOMPATIBLE
  - The dataflow method is incompatible with the action requested.
- ADI_DEV_RESULT_BUFFER_TYPE_INCOMPATIBLE
  - The device does not support the buffer type provided.
- ADI_DEV_RESULT_CANT_HOOK_INTERRUPT
  - The Interrupt Manager failed to hook an interrupt handler.
- ADI_DEV_RESULT_CANT_UNHOOK_INTERRUPT
  - The Interrupt Manager failed to unhook an interrupt handler.
- ADI_DEV_RESULT_NON_TERMINATED_LIST
  - The chain of buffers provided is not NULL terminated.
- ADI_DEV_RESULT_NO_CALLBACK_FUNCTION_SUPPLIED
  - No callback function was supplied when it was required.
- ADI_DEV_RESULT_REQUIRES_UNIDIRECTIONAL_DEVICE
  - Requires the device be opened for either inbound or outbound traffic only.
- ADI_DEV_RESULT_REQUIRES_BIDIRECTIONAL_DEVICE
  - Requires the device be opened for bidirectional traffic only.

5.6.2 Device Driver Specific Return Codes

There are no Audio Class specific return codes.

6 Opening and Configuring the Device Driver

This section describes the default configuration settings for the device driver and any additional configuration settings required from the client application.

6.1 Entry Point

When opening the device driver with the adi_dev_Open() function call, the client passes a parameter to the function that identifies the specific device driver that is being opened. This parameter is called the entry point. The entry point for this driver is listed below.

- ADI_USB_Device_AudioClass_Entrypoint

6.2 Default Settings

The table below describes the default configuration settings for the device driver. If the default values are inappropriate for the given system, the application should use the command IDs listed in section 5.4.3 to configure the device driver appropriately. Any configuration settings not listed in the table below are undefined.

Item	Default Value
Playback channels	2
Mixer enabled	TRUE
Microphone default record source	TRUE
Microphone in mixer	TRUE
Line-in in mixer	TRUE
Playback sample rate	48Khz
Playback sample resolution	16 bits
Line in record control enabled	TRUE
Microphone record control enabled	TRUE

Table 4 – Default Settings

6.3 Additional Required Configuration Settings

In addition to the possible overrides of the default driver settings, the device driver requires the application to specify the additional configuration information listed in the table below.

Item	Possible Values	Command ID
Dataflow method	See section 5.2	ADI_DEV_CMD_SET_DATAFLOW_METHOD
Enable Dataflow	See section 5.2	ADI_DEV_CMD_SET_DATAFLOW
Set user defined vendor ID	See section 5.4.3	ADI_USB_AUDIO_CMD_SET_VID
Set user defined product ID	See section 5.4.3	ADI_USB_AUDIO_CMD_SET_PID
Set user defined product string	See section 5.4.3	ADI_USB_AUDIO_CMD_SET_PRODUCT_STRING
Enable USB	See section 5.4.3	ADI_USB_CMD_ENABLE_USB
Pass the handle of the controller to the class driver	See section 5.4.3	ADI_USB_CMD_CLASS_SET_CONTROLLER_HANDLE

Item	Possible Values	Command ID
Configure the class driver	See section 5.4.3	ADI_USB_CMD_CLASS_CONFIGURE
Pass the user buffer to the class driver	See section 5.4.3	ADI_USB_AUDIO_CMD_SET_RX_BUFFER

Table 5 – Additional Required Settings

7 Hardware Considerations

None