How to Configure a USB Headset With Sample C Applications

1. Configuring Audio

Before a USB headset can be configured, the Virtual Audio Cape must be loaded with the playback devices configured and tested to playback audio.

a) Go through the audio guide by Brian Fraser to ensure that the BeagleBone Black is properly configured (http://www.cs.sfu.ca/CourseCentral/433/bfraser/other/AudioGuide.pdf).

b) Ensure that parts 1 (Installing Virtual Audio Cape), 2 (Configure and Play Audio), and 3 (Play PCM Audio from C) are completed before following with the rest of the guide.

The steps in the audio guide ensure that the audio cape is loaded, the ALSA sound library is compiled and placed in the appropriate folder, and that audio can be played from a simple C application.

2. Connecting the USB Headset

When the BeagleBone Black is configured for audio you will next need to connect a USB headset and configure the device.

a) Use the command “aplay –L” to list the currently configured playback devices:

```
root@cbologne-beagle:~# aplay -L
        null
           Discard all samples (playback) or generate zero samples (capture)
        default:CARD=EVM
           DA830 EVM,
           Default Audio Device
        sysdefault:CARD=EVM
           DA830 EVM,
           Default Audio Device
```

Without a headset, two devices are automatically configured for audio playback (null, and CARD= EVM). Null will send the audio playback to no device, resulting in the audio not being played back [1]. CARD=EVM is the audio jack on the Zen Cape and can be used to playback audio through a pair of speakers or a headset with an audio jack connector instead of USB. The aliases default and sysdefault can be used to playback audio on using the Zen Cape audio jack without specifying the full device name.

b) Next plug in a USB headset into the USB port on the BeagleBone Black (opposite side of the Ethernet Port) and reboot the BeagleBone Black to force Linux to detect the new USB device and load the appropriate drivers.

c) Once the OS has rebooted, check that the new USB device has been loaded with "dmesg | grep usb". If you see an output similar to the following, the device was detected and configured as a USB audio device:

```
[    1.122282] usb 1-1: new full-speed USB device number 2 using musb-hdrc
[    1.241996] usb 1-1: skipped 7 descriptors after interface
[    1.242018] usb 1-1: skipped 2 descriptors after interface
[    1.242032] usb 1-1: skipped 1 descriptor after endpoint
[    1.242044] usb 1-1: skipped 2 descriptors after interface
[    1.242055] usb 1-1: skipped 1 descriptor after endpoint
[    1.242066] usb 1-1: skipped 1 descriptor after interface
[    1.242162] usb 1-1: default language 0x0409
[    1.242296] usb 1-1: udev 2, busnum 1, minor = 1
[    1.242311] usb 1-1: New USB device found, idVendor=0d8c, idProduct=000e
[    1.242323] usb 1-1: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[    1.242335] usb 1-1: Product: Generic USB Audio Device
[    1.242780] usb 1-1: usb_probe_device
[    1.242798] usb 1-1: configuration #1 chosen from 1 choice
[    1.242972] usb 1-1: adding 1-1:1.0 (config #1, interface 0)
[    1.243171] usb 1-1: adding 1-1:1.1 (config #1, interface 1)
[    1.243400] usb 1-1: adding 1-1:1.2 (config #1, interface 2)
[    1.243537] usb 1-1: adding 1-1:1.3 (config #1, interface 3)
[    1.243678] usbhid 1-1:1.3: usb_probe_interface
[    1.243693] usbhid 1-1:1.3: usb_probe_interface - got id
[    3.625123] rtusb init rt2870 --->
[    3.625245] usbcore: registered new interface driver rt2870
[    5.713734] snd-usb-audio 1-1:1.0: usb_probe_interface
[    5.713770] snd-usb-audio 1-1:1.0: usb_probe_interface - got id
[    5.796298] usbcore: registered new interface driver snd-usb-audio
```

d) With the device loaded, we can check that alsa has also detected the new device by examining the /proc/asound/ folder as follows [2]:

```
root@cbologne-beagle:~# ls /proc/asound/
    card0  cards   devices  hwdep  pcm  timers
    card1  Device  EVM      oss    seq  version
root@cbologne-beagle:~# cat /proc/asound/devices
  2: [ 1- 0]: digital audio playback
  3: [ 1- 0]: digital audio capture
  4: [ 1]   : control
  5: [ 0- 0]: digital audio playback
  6: [ 0- 0]: digital audio capture
  7: [ 0]   : control
 33:        : timer
```

Since the directory now has two cards (0 and 1) each with a digital audio payback and capture device, the USB headset has been loaded by the OS and configured by the ALSA sound controller.

e) Next, we can ensure that the device is configured for playback by checking the playback interfaces with "aplay –L":

```
root@cbologne-beagle:~# aplay -L
    null
        Discard all samples (playback) or generate zero samples (capture)
    default:CARD=EVM
        DA830 EVM,
        Default Audio Device
    sysdefault:CARD=EVM
        DA830 EVM,
        Default Audio Device
    default:CARD=Device
        Generic USB Audio Device, USB Audio
        Default Audio Device
    sysdefault:CARD=Device
        Generic USB Audio Device, USB Audio
        Default Audio Device
    front:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        Front speakers
    surround40:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        4.0 Surround output to Front and Rear speakers
    surround41:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        4.1 Surround output to Front, Rear and Subwoofer speakers
    surround50:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        5.0 Surround output to Front, Center and Rear speakers
    surround51:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        5.1 Surround output to Front, Center, Rear and Subwoofer speakers
    surround71:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        7.1 Surround output to Front, Center, Side, Rear and Woofer speakers
    iec958:CARD=Device,DEV=0
        Generic USB Audio Device, USB Audio
        IEC958 (S/PDIF) Digital Audio Output
```

Since we now have a new device (CARD=Device, DEV=0), the USB headset has been configured and is ready to be used for playback. Once initially configured, the USB device will often automatically load the driver when it is plugged into the BeagleBone Black without rebooting the device.

Troubleshooting:

- If you don't see the USB headset as one of the devices available in aplay, ensure that the device has been loaded correctly via dmesg. Try rebooting the board and confirm that the drivers have been loaded correctly.
- To test which of the devices is the USB headset, use the command “speaker-test -D <device name>” to play test sound via the desired device.
- Microphone capture can be tested using the command “arecord -D <device name>”

3. ALSA asoundlib.h General Notes

The ALSA C sound library can be used to access the USB headset's speaker and microphone from a C application. A few general concepts should be understood before attempting to capture and playback audio on the headset.

a) Selecting Audio Devices via asoundlib

The device names in part **2e)** can be used in asoundlib to specify the sound device for either playback or capture. For example we can use “default” or “sysdefault:CARD=Device” to indicate that CARD=EVM should be used for playback. However, this is problematic with multiple USB headsets since each manufacturer has a different device name and the default devices are configured to access the audio jacks on the Zen Cape. To get around this, we can use the device names of the following format:

“hw:1,0” or “plughw:1,0”

The names break down as follows:

“hw/plughw:” specifies whether the device will be configured to automatically convert data into a format supported by the hardware (plughw) or to not perform the automatic conversion (hw), forcing the application to check the supported formats and features before configuring the device [3].

“1,0:” the first number specifies the audio card being used, while the second specifies the device number on that card [3]. Refer to steps **2d)** and **2e)** to ensure you are using the correct audio card and device for playback and capture.

b) Storing Audio Frames

Since a USB headset is often used to capture and playback multi-channel audio, the manner in which the audio channels are stored need to be taken into consideration. Each sound frame consists of one sample (one short) per channel, requiring a conversion from the number of shorts to the number of frames when interfacing with asoundlib’s functions that access the audio hardware [4]. This is in contrast to the sample code in the sound guide, which assume mono playback (one short per frame).

Simple C Program for Headset Capture and Playback

The following is a simple C program that reads 128 frames of data from the USB microphone and plays the audio back through the headset speakers (based on guides [3] and [5]):

```
#include <stdio.h>
#include <stdlib.h>
#include <alsa/asoundlib.h>
```

```
#define AUDIO_DEVICE "plughw:1,0"
#define SAMPLE_RATE   44100
#define NUM_CHANNELS  2
#define BUF_SIZE          128
static snd_pcm_t *input_handle;
static snd_pcm_t *output_handle;
//multiply by number of channels since a frame is one sample per channel
static short buf[BUF_SIZE * NUM_CHANNELS];

//local headers
static void initInput(void);
static void initOutput(void);
static void playAudio(void);

static void initInput(void){
	int err;
	snd_pcm_hw_params_t *hw_params;

	if ((err = snd_pcm_open (&input_handle, AUDIO_DEVICE, SND_PCM_STREAM_CAPTURE, 0)) < 0) {
		fprintf (stderr, "cannot open audio device %s (%s)\n",
					 AUDIO_DEVICE,
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}

	if ((err = snd_pcm_hw_params_malloc (&hw_params)) < 0) {
		fprintf (stderr, "cannot allocate hardware parameter structure (%s)\n",
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}

	if ((err = snd_pcm_hw_params_any (input_handle, hw_params)) < 0) {
		fprintf (stderr, "cannot initialize hardware parameter structure (%s)\n",
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}

	if ((err = snd_pcm_hw_params_set_access (input_handle, hw_params,
SND_PCM_ACCESS_RW_INTERLEAVED)) < 0) {
		fprintf (stderr, "cannot set access type (%s)\n",
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}

	if ((err = snd_pcm_hw_params_set_format (input_handle, hw_params, SND_PCM_FORMAT_S16_LE)) < 0) {
		fprintf (stderr, "cannot set sample format (%s)\n",
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}
	unsigned int rate = SAMPLE_RATE;
	if ((err = snd_pcm_hw_params_set_rate_near (input_handle, hw_params, &rate, 0)) < 0) {
		fprintf (stderr, "cannot set sample rate (%s)\n",
				 snd_strerror (err));
		exit(EXIT_FAILURE);
	}

	if ((err = snd_pcm_hw_params_set_channels (input_handle, hw_params, NUM_CHANNELS)) < 0) {
		fprintf (stderr, "cannot set channel count (%s)\n",
```

```
                snd_strerror (err));
        exit(EXIT_FAILURE);
    }

    if ((err = snd_pcm_hw_params (input_handle, hw_params)) < 0) {
        fprintf (stderr, "cannot set parameters (%s)\n",
                snd_strerror (err));
        exit(EXIT_FAILURE);
    }

    snd_pcm_hw_params_free (hw_params);

    if ((err = snd_pcm_prepare (input_handle)) < 0) {
        fprintf (stderr, "cannot prepare audio interface for use (%s)\n",
                snd_strerror (err));
        exit(EXIT_FAILURE);
    }
}
static void initOutput(void){
    // Open the PCM output
    int err = snd_pcm_open(&output_handle, AUDIO_DEVICE, SND_PCM_STREAM_PLAYBACK, 0);
    if (err < 0) {
        printf("Play-back open error: %s\n", snd_strerror(err));
        exit(EXIT_FAILURE);
    }
    // Configure parameters of PCM output
    err = snd_pcm_set_params(output_handle,
            SND_PCM_FORMAT_S16_LE,
            SND_PCM_ACCESS_RW_INTERLEAVED,
            NUM_CHANNELS,
            SAMPLE_RATE,
            1,                  // Allow software resampling
            50000);         // 0.05 seconds per buffer
    if (err < 0) {
        printf("Play-back configuration error: %s\n", snd_strerror(err));
        exit(EXIT_FAILURE);
    }
}

static void playAudio(void){
    snd_pcm_sframes_t frames = snd_pcm_writei(output_handle, buf, BUF_SIZE);
    // Check for errors
    if (frames < 0)
        frames = snd_pcm_recover(output_handle, frames, 0);
    if (frames < 0) {
        fprintf(stderr, "ERROR: Failed writing audio with snd_pcm_writei(): %li\n", frames);
        exit(EXIT_FAILURE);
    }
    if (frames > 0 && frames < BUF_SIZE)
        printf("Short write (expected %d, wrote %li)\n", BUF_SIZE, frames);

}

int main (int argc, char *argv[]) {
    int err;
    initInput();
    initOutput();

    for (int i = 0; i < 100000; ++i) {
```

```
		if ((err = snd_pcm_readi (input_handle, buf, BUF_SIZE)) != BUF_SIZE) {
			fprintf (stderr, "read from audio interface failed (%s)\n",
				 snd_strerror (err));
			exit(EXIT_FAILURE);
		}
		playAudio();
	}
	snd_pcm_close (input_handle);
	snd_pcm_close (output_handle);
	return EXIT_SUCCESS;
}
```

Simple C Function to Change Playback Volume

A C function to change the volume on a USB headset is very similar to the function presented in the sound guide, with the audio card changed from “default” to “hw:1” [6].

```
// Function copied from:
// http://stackoverflow.com/questions/6787318/set-alsa-master-volume-from-c-code
// Written by user "trenki".
void Audio_setVolume(int newVolume)
{
	// Ensure volume is reasonable; If so, cache it for later getVolume() calls.
	if (newVolume < 0 || newVolume > MAX_VOLUME) {
		printf("ERROR: Volume must be between 0 and 100.\n");
		return;
	}
	volume = newVolume;

	long min, max;
	snd_mixer_t *handle;
	snd_mixer_selem_id_t *sid;
	//Devices can be listed by using "amixer scontrols" command
	//http://alsa.opensrc.org/HowTo_access_a_mixer_control
	const char *card = "hw:1";
	const char *selem_name = "PCM";

	snd_mixer_open(&handle, 0);
	snd_mixer_attach(handle, card);
	snd_mixer_selem_register(handle, NULL, NULL);
	snd_mixer_load(handle);

	snd_mixer_selem_id_alloca(&sid);
	snd_mixer_selem_id_set_index(sid, 0);
	snd_mixer_selem_id_set_name(sid, selem_name);
	snd_mixer_elem_t* elem = snd_mixer_find_selem(handle, sid);

	snd_mixer_selem_get_playback_volume_range(elem, &min, &max);
	snd_mixer_selem_set_playback_volume_all(elem, volume * max / 100);

	snd_mixer_close(handle);
}
```

C Program Common Errors

The following are some common errors we encountered with USB headset audio during the course of our project along with the solutions we found:

EPIPE Broken Errors

Problem: These errors are thrown if an underrun occurs when trying to capture data from the microphone. This commonly happens when the thread capturing data via readi() is interrupted and the data in the microphone's hardware buffer isn't cleared out fast enough [7].

Solution: To resolve this problem, use the function snd_pcm_prepare(<capture_handle>) to clear the leftover data and prepare the device for capturing input [7].

Large Number of Valgrind Errors

Problem: If you encounter this issue, the likely cause is that the buffers allocated for the microphone or speaker data haven't taken into account the number of channels per frame [4].

Solution: Ensure that all buffers contain one short per channel and that this data is considered one frame in each of the asoundlib function calls (especially those that read and write data to the headset) [4].

References

[1] ALSA project, "ALSA project - the C library reference," ALSA, 2015. [Online]. Available: http://www.alsa-project.org/alsa-doc/alsa-lib/pcm_plugins.html. [Accessed 5 December 2015].

[2] J. P. Giraud, "ALSA," Debian, 8 November 2015. [Online]. Available: https://wiki.debian.org/ALSA. [Accessed 2015 5 December].

[3] M. Nagorni, "ALSA Programming HOWTO," Suse, 24 February 2010. [Online]. Available: http://users.suse.com/~mana/alsa090_howto.html. [Accessed 5 December 2015].

[4] J. C. Dutton, "Frames Periods," ALSA, 24 September 2006. [Online]. Available: http://www.alsa-project.org/main/index.php/FramesPeriods. [Accessed 5 December 2015].

[5] P. Davis, "A Tutorial on Using the ALSA Audio API," Equal Area, 2002. [Online]. Available: http://equalarea.com/paul/alsa-audio.html. [Accessed 5 December 2015].

[6] trenki, "Set ALSA master volume from C code," Stack Overflow, 22 July 2011. [Online]. Available: http://stackoverflow.com/questions/6787318/set-alsa-master-volume-from-c-code. [Accessed 5 December 2015].

[7] J. Tranter, "Introduction to Sound Programming with ALSA," Linux Journal, 30 September 2004. [Online]. Available: http://www.linuxjournal.com/node/6735/print. [Accessed 5 December 2015].