Installing PortAudio on a BeagleBone Green

PortAudio
Portable Cross-platform Audio I/O

Written for Kernel 4.9+
Version note: This guide targets boards which have the cape management built into UBoot, not the Linux Cape Manager (which is now removed).
Last Update: 27 Nov 2019
Team: Tin Can Telephones

This document guides the user through:

1. Explaining PortAudio
2. Why PortAudio can be used for Audio Programming in C
3. How to install and use PortAudio
4. Troubleshooting common errors

Formatting:

1. Host (desktop) commands starting with $ are Linux console commands:
   ```
   $ echo "Hello world"
   ```
2. Target (board) commands start with #:
   ```
   # echo "On embedded board"
   ```
3. All commands are case sensitive.

Introduction

PortAudio (http://www.portaudio.com/) is a portable library that exposes a convenient stream API for manipulating audio, in both the record and playback directions. It is straightforward to compile and include in your C project. After *#include*-ing, it is simple to configure for your target device. The library is compatible with ALSA, so if you have completed the Audio Guide, you are half way to finishing this guide. We successfully used PortAudio in our VoIP project.

Why use PortAudio?

ALSA (asoundlib) is not a very (student/user)-friendly library; you cannot simply *#include* and start reading/writing audio. Runtime configuration is esoteric and, worst of all, undocumented. The API is labyrinthine and poorly documented. This means that if you want to do something more complicated than play a sine wave or adjust your input/output volume, you must sift through thousands of lines of example code.

You can alleviate these issues by using **PortAudio**.

Installation

Prerequisites

- ALSA must be installed and configured on the target (the BeagleBone).
- The target must have internet access enabled.
- The target must have SSH or screen access.
- The target must have NFS set up and `/mnt/remote` mounted.

Install libasound2-dev on the target

Run the following command on the target machine to install the appropriate sound library:

```
     # sudo apt-get install libasound2-dev
```

Set Up PortAudio on the target

Run the following commands on the target machine to download, make, and properly configure the PortAudio library:

```
# cd /mnt/remote/
# wget http://portaudio.com/archives/pa_stable_v190600_20161030.tgz
# tar xvzf pa_stable_v190600_20161030.tgz
# cd ./portaudio
# ./configure && make
```

Important: Ensure that PortAudio is correctly configured by checking the configuration summary.

You should see output like this:

```
Configuration summary:

  Target ...................... armv7l-unknown-linux-gnueabihf
  C++ bindings ................ no
  Debug output ................ no

  ALSA ........................ yes
  ASIHPI ...................... no



  OSS ......................... yes
  JACK ........................ no
```

It is vital that ALSA is **yes**. Otherwise, libasound2-dev has not been correctly installed on the target.

Copy binary and headers to project folder

After running `make`, run the following commands (within the PortAudio archive directory) to duplicate the PortAudio prebuilt static library file (`libportaudio.a`) and the header file (`portaudio.h`) into your own project:

```
# cp lib/.libs/libportaudio.a /location/of/your/project/
# cp include/portaudio.h /location/of/your/project/include/
```

Now you can `#include` the `portaudio.h` file in your source code without errors or warnings at compile time.

If you would like to cross-compile from the host machine to the target machine, then copy these two files into your own project directory on the host machine.

To build your final executable, your example gcc command should look similar to the following:

```
# gcc main.c libportaudio.a -lm -lasound -pthread -o YOUR_BINARY
```

If you can compile with no errors, then you have correctly compiled PortAudio.

Troubleshooting

Fixing runtime error messages

PortAudio is not smart enough to figure out which devices are actually read/writeable by itself. It may attempt to access devices that are exposed by ALSA but are not actually used. In this case, during runtime, calling `Pa_Initialize()` will produce an avalanche of errors.

This can be fixed by correctly configuring `alsa.conf` located on the target at: `/usr/share/alsa/alsa.conf` .

Comment out the following lines in the configuration file, and run again:

```
pcm.front cards.pcm.front
pcm.rear cards.pcm.rear
pcm.center_lfe cards.pcm.center_lfe
pcm.side cards.pcm.side
pcm.surround21 cards.pcm.surround21
pcm.surround40 cards.pcm.surround40
pcm.surround41 cards.pcm.surround41
pcm.surround50 cards.pcm.surround50
pcm.surround51 cards.pcm.surround51
pcm.surround71 cards.pcm.surround71
pcm.iec958 cards.pcm.iec958
pcm.spdif iec958
pcm.hdmi cards.pcm.hdmi
pcm.modem cards.pcm.modem
pcm.phoneline cards.pcm.phoneline
```

If you still see error messages such as:

```
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM surround71
```

You should just note down which device (in this case “surround71”), then comment out the corresponding line in the config file. They are all grouped together, so it’s easy to find. Repeat this process until there are no more runtime errors.

Congratulations, you are done.

ALSA Configuration

If alsa.conf is still not working for you, we have attached our modified version of alsa.conf which worked for our project.