PortAudio Setup and Configuration Guide for BeagleBone

Team: STB
Author: Brandon Ong

Introduction:

This guide will walk through how to configure the external library, PortAudio, for cross-compiling, as well as how to integrate real-time duplex audio streaming/processing into BeagleBone projects (using C and CMake).

Table of Contents:

1. Requirements

- Followed Dr. Brian's audio guide
  - Loaded audio device tree overlay in `/boot/uEnv.txt` on BBG
  - Installed sound tools/libraries for normal audio playback (alsa-utils, usbutils, i2c-tools) on BBG
  - Installed libasound2 on BBG
  - Installed libasound2-dev and libasound2-dev:armhf on host
- Have an input device connected (e.g. microphone/audio interface)
- Have an output device connected (e.g. headphones, speakers)
- Have VSCode installed for cross-compiling (for CMake)

2. Configuration for Cross-compiling to the BeagleBone

To have the project run a duplex audio stream, we will first need to download PortAudio into any folder.

1. Go to the official PortAudio website and download the latest stable release and open the terminal into where it was downloaded.
2. Extract the file.

```
(host)$ tar xzvf pa_stable_v[version number]_[date of release].tgz
```

3. Go into the directory.

```
(host)$ cd portaudio
```

4. Now, we want to configure the external library on the host so that it works on the BBG.

```
(host)$ ./configure –host=arm-linux-gnueabihf –build=x86_64-linux-gnu
```

A successful configuration should result in the following:

```
Configuration summary:

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

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

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

5. Cross-compile the library file using CMake.

```
(host)$ code .
Select kit > GCC arm-linux-gnueabihf
Click the build button at the bottom of VSCode.
```

6. Copy libportaudio.a and portaudio.h into your corresponding project folders.

```
(host)$ cp /build/libportaudio.a ~/cmpt433/work/myApps/[projectfolder]
(host)$ cp /include/portaudio.h ~/cmpt433/work/myApps/[projectfolder]/app/include
```

7. Troubleshooting
   - If the configuration command on step 4 does not work, it may be better to download and configure PortAudio on the BBG, following the method in the other student guide instead[1].
     a. Extract the PortAudio tgz file into the shared folder.
     b. ssh into the BBG and cd into the extracted folder.
     c. Configure and make the library files.
        ```
        (BBG)$ ./configure && make
        ```
     d. Now, we can go back to the host and do step 6.

[1] Note that the method used in the other student guide builds and compiles both files all on the BBG as opposed to building and compiling on the host for the BBG in this guide, and thus would need internet access and make already installed on the BBG.

3. CMake setup

We now want to integrate the external library into the project.

1. Assuming libportaudio.a is in the base folder and portaudio.h is in the , we can add the following to `/app/CMakeLists.txt`:

```
find_package(ALSA REQUIRED)
target_link_libraries([project-exe] LINK_PRIVATE asound)
target_link_libraries([project-exe] LINK_PRIVATE
${CMAKE_SOURCE_DIRECTORY}/libportaudio.a)
```

2. If you do not want to use CMake, you may run something similar to the following to cross-compile:

```
(host)$ arm-linux-gnueabihf-gcc [inputfile].c libportaudio.a -lm -lasound -pthread -o
[outputfile]
```

4. Sample C code setup

The following is an example setup configuration to start a real-time duplex stream for your program. (Error handling omitted, modified from [PortAudio's example file pa_fuzz.c](pa_fuzz.c)).

1. First, the stream needs to be initialized:

```
PaStreamParameters inputParameters, outputParameters;
PaStream *stream;
Pa_Initialize();
```

2. Set the input parameters:

```
inputParameters.device = Pa_GetDefaultInputDevice();
inputParameters.channelCount = 2;/* stereo input */
inputParameters.sampleFormat = PA_SAMPLE_TYPE;
inputParameters.suggestedLatency = Pa_GetDeviceInfo( inputParameters.device
)->defaultLowInputLatency;
inputParameters.hostApiSpecificStreamInfo = NULL;
```

3. Set the output parameters:

```
outputParameters.device = Pa_GetDefaultOutputDevice();
outputParameters.channelCount = 2;/* stereo output */
outputParameters.sampleFormat = PA_SAMPLE_TYPE;
outputParameters.suggestedLatency = Pa_GetDeviceInfo( outputParameters.device
)->defaultLowOutputLatency;
outputParameters.hostApiSpecificStreamInfo = NULL;
```

4. Open and start the stream. There are two ways to open the stream and deal with its input/output buffers: through callbacks or blocking I/O. This guide details the callback method only.

```
Pa_OpenStream(&stream, &inputParameters, &outputParameters, SAMPLE_RATE,
FRAMES_PER_BUFFER, 0, fuzzCallback, NULL);
Pa_StartStream(stream);
```

   a. Looking at pa_fuzz.c, we can see that the header for the callback is as follows:

```
static int fuzzCallback( const void *inputBuffer, void *outputBuffer,
                         unsigned long framesPerBuffer,
                         const PaStreamCallbackTimeInfo* timeInfo,
                         PaStreamCallbackFlags statusFlags,
                         void *userData );
```

Each of the three highlighted parameters are important for these duplex streams. For example, to get normal playback, all that needs to be done is to make each output frame in the outputBuffer equal to its input frame from its inputBuffer.

5. When done, close the stream and terminate it to shutdown gracefully.

```
Pa_CloseStream(stream);
Pa_Terminate();
```

6. Troubleshooting
   - There is a high chance you may see these errors such as:

```
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition
'cards.AudioCape_Rev_B.pcm.front.0:CARD=0'
ALSA lib conf.c:4745:(_snd_config_evaluate) function snd_func_refer returned error: No
such file or directory
ALSA lib conf.c:5233:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM front
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
```

     - To fix this, navigate to `/usr/share/alsa/alsa.conf` and comment out lines such as:

```
pcm.front cards.pcm.front
pcm.rear cards.pcm.rear
```

The explanation for this error and its solution can be found in a previous student guide under the troubleshooting section.

- If you encounter any one of these errors, there is a good chance that PortAudio is looking at the wrong device for either your input or output devices.

```
An error occurred while using the portaudio stream
Error number: -9998
Error message: Invalid number of channels
An error occurred while using the portaudio stream
Error number: -9988
Error message: Invalid stream pointer
```

    - Use `aplay -l` or `arecord -l` and take note of which devices are the proper input/output indexes that should be used.
    - There are two options:
        - The first is to change the input/output devices in the C code to the proper device index, notably:

```
inputParameters.device = [index of proper sound card];
```

and

```
outputParameters.device = [index of proper sound card];
```

        - Or, go into `/etc/asound.conf` and add/change:

```
defaults.pcm.card [index of proper sound card]
defaults.ctl.card [index of proper sound card]
```

- If no errors show up and there is still no playback, use `alsamixer` to check audio levels.
    - Use F3 to check the playback volume.
    - Use F4 to check the capture device's gain levels.

5. Sources/References

Previous student PortAudio guide from 2019
Dr. Brian's Audio Guide