Audio Training Exercises

Session Exercises

Clone the git repo for exercises.
$ git clone https://github.com/embitude/training/
PS. Just git pull, if repo has already been cloned

Session 1 Exercises

Exercise 1: Listing the sound card and devices

1. On the target board, execute the following command:
   $ modprobe snd-dummy
   This would load the dummy sound driver, which registers the sound card and corresponding devices
2. List the playback & capture devices
   $ aplay -l (This should list the playback devices)
   $ arecord -l (This would list the capture devices)

Steps for compiling the ALSA applications

The ALSA application compilation needs the cross alsa lib for compilation. This doesn't comes as a part of toolchain. So, the sdk generated which has the support for alsa lib is used to compile the applicatoin

1. Get into the Builds directory & untar the sdk
   $ cd training/Builds
   $ tar -xf arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.gz
2. Relocate the sdk
   $ cd arm-buildroot-linux-gnueabihf_sdk-buildroot
   $ ./relocate-sdk.sh
3. Next step is to source the sdk environment in shell which is used for application compilation & compile the application
   $ source training/Builds/arm-buildroot-linux-gnueabihf_sdk-buildroot/environment-setup
   $ cd training/Audio/Apps
   $ make

Exercise 2: Set Parameters

The objective here is to set the parameters & verify if it has been updated

1. Navigate to Audio/Apps directory
   $ cd training/Audio/Apps
2. Compile the application
   $ make
   This would generate the executable for all the applications. Let's test the set_params application
3. Transfer the application on the board and make sure that the snd-dummy is loaded
   $ scp set_params root@192.168.7.2:
   $ modprobe snd-dummy
4. Execute the application
   $ ./set_params hw:0,0

Exercise 3: Minimal Playback application

The objective here is to test minimal playback application on target board

1. Navigate to Audio/Apps directory
   $ cd training/Audio/Apps
2. Compile the application
   $ make
   This would generate the executable for all the applications. Let's test the playback_min application
3. Transfer the application on the board and make sure that the snd-dummy is loaded
   $ scp playback_min root@192.168.7.2:
   $ modprobe snd-dummy
4. Execute the application
   $ ./playback_min hw:0,0
   This should playback the audio to the dummy sound card

Exercise 4: Minimal Capture application

The objective here is to test minimal capture application on target board

1. Navigate to Audio/Apps directory
   $ cd training/Audio/Apps
2. Compile the application
   $ make
   This would generate the executable for all the applications. Let's test the cap_min application
3. Transfer the application on the board and make sure that the snd-dummy is loaded
   $ scp cap_min root@192.168.7.2:
   $ modprobe snd-dummy
4. Execute the application
   $ ./cap_min hw:0,0
   This should capture the audio from dummy sound card

Exercise 5: Playback Application with various transfer methods

The objective here is to demonstrate the various mechanisms to transfer the data to the driver

1. Navigate to Audio/Apps directory
   $ cd training/Audio/Apps
2. Compile the application
   $ make
   This would generate the executable for all the applications. Let's test the playback application
3. Transfer the application on the board and make sure that the snd-dummy is loaded
   $ scp playback root@192.168.7.2:
   $ modprobe snd-dummy
4. Execute the application
   $ ./playback < <audio file>
   This should playback the audio with selected transfer method

Session 2 Exercises (ALSA Drivers)

Exercise 1: Register the Platform Driver & Platform Device

The idea over here is to make sure that the platform driver and device is getting registered & the corresponding probe is getting invoked.

1. Navigate to Audio/Drivers/Alsa directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 1.1 to 1.5 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
   Verify that the probe gets invoked on insmod and remove gets invoked on rmmod.

Exercise 2: Register the Sound card

The objective here is to register the sound card with the alsa core

1. Navigate to Audio/Driver/Alsa directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 2.1 to 2.4 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
4. Verify that the sound card is listed
   $ cat /proc/asound/MySoundCard/id (Should display the id string for the sound card)
   $ cat /proc/asound/card0/id

Exercise 3: Register the PCM device and operations

The objective here is to register the pcm device for the card

1. Navigate to Audio/Driver directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 3.1 to 3.4 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
4. Verify that the sound card is listed
   $ cat /proc/asound/MySoundCard/id (Should display the id string for the sound card)
   $ cat /proc/asound/card0/id
5. Verify that the capture & playback devices are listed
   $ aplay -l (Should list the playback devices)
   $ arecord -l (Should list the capture devices)
6. Next verify if the device files are created
   $ ls /dev/snd/pcmC0D0[p/c]
   These are the devices files for the capture & playback
7. Next, try playback and observe the behaviour
   $ aplay <audio_file>

Exercise 4: Playback ops

The objective here is to observe the pcm ops getting invoked as a part of application invoking the playback

1. Navigate to Audio/Driver/Alsa directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 3.5 to 3.9 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
4. Verify that the capture & playback devices are listed
   $ aplay -l (Should list the playback devices)
5. Verify the playback
   aplay <audio file>
   Observe the calls such as open, close, hw_params, prepare, trigger and hw_free getting invoked

Exercise 5: Update the buffer positions

The objective here is to get the playback working & update the pointers accordingly

1. Navigate to Audio/Driver directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 4.1 to 4.18 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
4. Verify that the capture & playback devices are listed
   $ aplay -l (Should list the playback devices)
5. Verify the playback
   aplay <audio file>
   The playback should work without any issues and buffer positions should be updated accordingly

Exercise 6: Capture the dummy data

The objective here is to fill capture buffer with the pre-defined data

1. Navigate to Audio/Driver/Alsa directory
   $ cd training/Audio/Driver/Alsa
2. Complete the todos 5.1 to 5.3 in dummy.c & compile the driver
   $ make
   This would generate the kernel module with name dummy.ko.
3. Transfer dummy.ko to the board & load the same
   $ insmod dummy.ko
4. Verify that the capture & playback devices are listed
   $ aplay -l (Should list the playback devices)
5. Verify the capture
   arecord <audio file>

Transfer the audio file to the PC and observe the waveform