

# Lab 4D. Linux boot-up on XUP board and writing Device Drivers for Custom-IP

## Objective

In this lab, you will learn to interface the Audio ports on the XUP board. You will write a device driver for the Custom IP `ir_detect` (created in part C of this lab) as well as the Audio Codec and run it from the Linux OS.

## Procedure

The following steps will guide you through the steps for playing music on the audio (Line Out) port on the XUP board:

1. Create a directory called 'lab4d', where all the XPS generated files for part D of this lab will be stored. Start a new project on XPS. On the 'Create New XPS Project using BSB Wizard' window, select 'lab4d' as the path, and specify 'C:/EDK\_lib' as the User Repository (Figure 1).

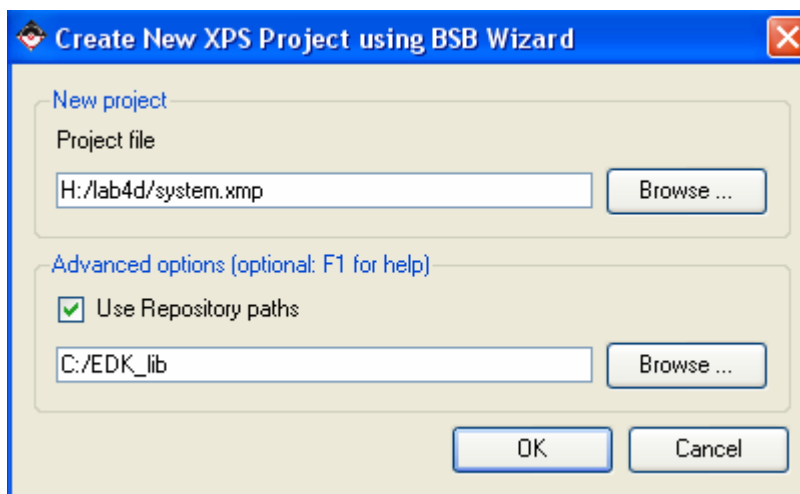


Figure 1.: Snap shot of 'Create New XPS Project using BSB Wizard' with the Repository path

By default, BSB only shows some of the peripherals on the XUP board. By including the EDK\_lib repository, we can access all the peripherals including the Audio Codec.

2. Create a BSB with:
  - 300MHz PLB (processor) clock, and 100MHz OPB (peripheral) clock.
  - RS232\_Uart\_1, choose OPB UARTLITE. Keep the other default settings. (deselect onewire\_0)
  - SysACE\_CompactFlash, choose OPB SYSACE.
  - DDR\_512MB\_64MX64\_rank2\_row13\_col10\_cl2\_5, choose PLB DDR.Check the 'Use interrupt' checkbox for each pcore above for Linux purposes.

- AC97 Audio Codec.

Choose 128 KB for the Memory size of your BRAM.

3. Click on 'Generate Bitstream' to build your Hardware. This will take several minutes. In the mean time, proceed to step 4.
4. Download the file 'my\_music.h' from the course website. The file contains the digitized data for a .wav file. In the Application tab, add a new source file (audio\_test.c) to write the Software to play the data in my\_music.h. For this, include the files 'xparameters.h' and 'xac97\_1.h'. The baseaddress for the Audio Codec is specified in xparameters.h, all the Audio Codec related functions can be found in xac97\_1.h.
  - Use the function HardReset to reset the Audio Codec.
  - Read the data from my\_music.h and write to the Fifo using the function 'PlayAudio'.After the bitstream generation (Step 3 above) is complete, Build your Software.
5. Your hardware (.bit) as well as your software (.elf) is ready to be downloaded on the XUP board. Combine them into a .ace file (similar to part A of this lab).
6. Put the system.ace file in the Compact Flash card provided to you, using the Compact Flash reader.
7. Open hyperterminal. Insert the Compact Flash card in the Compact Flash slot on the XUP board and the Speaker in the Line Out port. Turn on the board. You would see any software 'printf's on the hyperterminal, and hear music on the Speakers.
8. Now, write a driver in Linux to run the same program that you ran on Windows above. In /software/modules/audio on the LINUX-449 machine, you will find some music\*.h files which contains some sample music data that you can play. Copy this audio directory to the modules directory in your home. You will also need to copy the following files from your lab4d directory on Windows:
  - ppc\_0/include/xac97\_1.h
  - ppc\_0/libsrc/ac97\_v2\_00\_a/src/xac97\_1.c
  - audio\_test.c : you can copy the code from the main function of your audio\_test.c file to your driver, as appropriate.

### **Deliverables – Part D**

1. Demonstrate to the TA your working audio\_test on Windows. **[2 points]**
2. Write a simple audio driver on linux (Step 8 in procedure - equivalent to the multiply driver in Part B) using only the module\_init and module\_cleanup functions. Demonstrate the music played when you insmod your audio driver. Also, submit your driver code. **[3 points]**

3. Now, add the ir\_detect Custom IP to your system using the 'Create and Import Hardware Wizard' (similar to Part C). Write a device driver containing register\_chardev, etc (equivalent to the multiplier driver in Part B) for both the Audio Codec and ir\_detect devices. Create a list of the music files available in the /software/modules/audio directory. Print out this list when init\_module is called.

In devtest.c, write a function to first play the first music file in your list. When '1' is pressed on the remote, play the music file which is before the last music file played. When '2' is pressed, play the music file after the last music file played. E.g. if you have three files, A, B and C, and B was the last music file played, then when '1' is pressed A should be played. If instead of '1', '2' was pressed, C should be the file played.

Demonstrate your working driver and submit the code for driver as well as devtest.c **[5 points]**

In both deliverables 2 and 3 above, the report carries half the points.