<Product Area>

<Device>

Generic SDK Build on Raspberry Pi

Draft Version 1.0

June 2015

# Table of Contents

[1. Purpose 6](#_Toc427079103)

[2. Generic SDK Build System 6](#_Toc427079104)

[3. Building Generic SDK 6](#_Toc427079105)

[4. Building Microsemi ALSA Driver for Pi 7](#_Toc427079106)

[4.1 Running ALSA codec driver 7](#_Toc427079107)

[4.2 Testing ALSA codec driver 8](#_Toc427079108)

[5. Building SPI HBI Driver for Pi 9](#_Toc427079109)

# List of Tables

[Table 1 Abbreviations used in this document 5](#_Toc416449667)

# List of Figures

**No table of figures entries found.**

|  |  |
| --- | --- |
| **Document Owner** | Shally Verma |
| **Version** | 1.0 |
| **Date** | June 15 |

| Document Change History | | | |
| --- | --- | --- | --- |
| Date | Version | Changed by | Change Description |
| 15 June 2015 | 1.0 | Shally Verma | Re-formatted Generic SDK Build/Setup document on Pi as per Microsemi template.  Added history section to keep track of changes.  Document modified with latest build instructions.  Env.sh file is redundant and no need to source any more |
| 11 Aug 2015 | 1.0 | Shally Verma | Added SPI HBI driver build and run section for Pi |
|  |  |  |  |



# Purpose

This document described Generic Software Development Kit build and setup for Raspberry Pi Platform.

# Generic SDK Build System

Development source code for Generic SDK is available on svn at *http://aussvn01/svn/apps/vproc\_sdk/trunk*

There are two primary files inside the root directory of Generic SDK : Makefile and Makefile.globals.

All the environment variables and build option setting is done in Makefile.globals where as Makefile act as Master makefile which contains all rules for building different targets.

# Building Generic SDK

Microsemi VPROC Generic SDK can be both natively or cross compiled. Below section describes building steps for Generic SDK:

* 1. Go to trunk (the root folder)
  2. Makefile have provision to find current ROOTDIR and set TOOLSDIR accordingly. However if you want to point to Toolsdir at some other location then you can do that by settings TOOLSPATH environment variable in Makefile.globals.
  3. Makefile support different rules for different targets. Here’s list of all:
  4. make platform : it builds code inside platform directory which essentially means kernel and SSL and Sound driver. A successful build should result in object file and loadable module inside libs folder at root location (/trunk/libs)
  5. Once platform code is built, user can build other targets
  6. make hbi: builds the HBI driver. Make sure to build platform before.
  7. make hbilnx: build the linux version of HBI driver
  8. make codec: build the alsa driver inside platform directory to identify Microsemi as sound card to raspberry pi platform (Make sure to build kernel with Microsemi-dac-overlay.dts or you have device tree blob for Microsemi-dac and have it added in /boot/config.txt)
  9. make ssl. Build “System Service Layer” driver for Pi.
  10. Make all : build all of SDK component listed above. Apart from kernel, all other libraries and object code will be copied to “libs” folder.
  11. Every rule is followed by clean with naming convention target\_clean. Example, to clean HBI lib, hbi\_clean , for hbilnx hbilnx\_clean

\*hbi\_test is a simple test module for preliminary hbi code testing. Now user can use hbilnx driver module itself to test HBI function call.

# Building Microsemi ALSA Driver for Pi

Microsemi ALSA Driver for Pi is located at /trunk/platform/raspberry/driver/sound/lnxalsa and comply ASoC architecture (<http://www.alsa-project.org/main/index.php/ASoC>).

Code organization of Microsemi ALSA driver is something like this:



“soc” directory contains the ALSA driver code for codec (i.e. zl380xx) device and a machine driver to register Microsemi Vproc device as sound card to Pi.

Microsemi ALSA driver currently support device tree based build procedure only. Thus a device tree source file Microsemi-dac-overlay.dts is available which identifies Microsemi DAC as device to Pi. Current SDK doesn’t support auto building of Microsemi Device tree source file. Thus user manually has to copy Microsemi-dac-overlay.dts file into RPI kernel before building the kernel.

Here’re the build steps for building ALSA driver :

1. Copy microsemi-dac-overlay.dts from /trunk/platform/raspberry/kernel/dts/ into /trunk/platform/raspberry/kernel/linux/arch/arm/boot/dts
2. Modify makefile at /trunk/platform/raspberry/kernel/linux/arch/arm/boot/dts to add

dtb-$(RPI\_DT\_OVERLAYS) += microsemi-dac-overlay.dtb

1. Give command “make platform”. A successful build should result in microsemi-dac-overlay.dtb at /trunk/platform/raspberry/kernel/linux/arch/arm/boot/dts
2. Do “make codec”. This should result in snd-soc-microsemi-dac.ko and snd-soc-zl380xx.ko inside libs folder

## Running ALSA codec driver

Microsemi VPROC driver is tested keeping Raspberry Pi as Master and Voice Processor as Slave.

We are using i2s-a connector port P4 on ZLE3800 rev101.0004 (an 38051 Engineering Version)

* + 1. Boot the Pi
    2. Make sure pi using device tree. To check, do sudo raspi-config,

Go to option 8, Advanced Options -> A5 Device Tree -> Yes

* + 1. Copy the microsemi-dac-overlay.dtb file into Pi kernel into /boot/overlays
    2. Open /boot/config.txt, add following lines

dtparam=i2s=on

dtoverlay=microsemi-dac-overlay, and

**reboot** the board.

* + 1. Ensure Microsemi DAC device is detected by kernel by checking /sys/bus/platform/devices/. zl380-codec device should be visible there and on doing

cat /sys/bus/platform/devices/sound/modalias , it should display a string

“of:NsoundT<NULL>Cmicrosemi,microsemi-dacpi@raspberrypi:”

* + 1. Copy snd-soc-microsemi-dac.ko and snd-soc-zl380xx.ko from libs directory to pi home directory (or any drive which is shareable)
    2. Do “sudo insmod snd-soc-zl380xx.ko”
    3. Do “sudo insmod snd-soc-microsemi-dac.ko”
    4. Now ensure Microsemi machine and codec driver is registered, you can perform following steps:
       1. Run “aplay –l” .. it should show “sndmicrosemidac” as one of the registered card
       2. /sys/bus/platform/drivers/ should show zl380-codec/ and snd-microsemi-dac/ as one of the directory
       3. Now your device is ready for playback and capture

## Testing ALSA codec driver

* + 1. Load the desired configuration record via either HBI (I2C/SPI) or UART on to device. Issue a soft reset for settings to take effect
    2. To record an audio file, you can run “arecord” app.

For example to record Signed 16 bit Little Endian Stereo, 16Khz wav file, run

arecord –c 2 –f S16\_LE –r 16000 test.wav

For list of other options, run “arecord –h”

* + 1. To playback recorded file, run aplay app.

For example, to play recorded S16 Little Endian, 16Kz Stereo

aplay test.wav, or

aplay –D hw:1,0 test.wav (if vproc device registered as card 1)

# Building SPI HBI Driver for Pi

The device tree of Debian wheezy distribution of Raspberry Pi registers a virtual device “spidev” on both SPI chip select 0 and 1. This means user can connect any device physically and do basic read, write through any userspace application.

So, this gives us two options to use Microsemi VPROC device over SPI

1. via SPIDEV interface ( as /dev/spidev0.0 and /dev/spdev0.1)

Useful if user just needs to do read/write to device with user space driver for pi. This will not let user to do other operations like boot from host, flash erase etc.

\*Current Gen SDK system do not support this. Any user wish will need to write their own application in userspace to access device via spidev.

1. via Microsemi HBI driver interface (as /dev/hbi)

This would give full feature set of device. Register read/write, boot from host, configuration record loading etc.

However in order to use hbi device driver, user need to apply microsemi-spi-overlay.dtb on RPI distribution to disable spidev device registration. Please refer to following section on “how to” enable Microsemi SPI HBI driver on Pi.

## Building SPI HBI Driver

By default, Gen SDK compiles for SPI. To build for i2c, user specifically need to mention HBI=i2c in make command

* + 1. Goto trunk
    2. If this is fresh checkout and kernel is not built yet, then do “make platform”. This will untar linux tar file and build it.
    3. Copy platform/raspberry/kernel/dts/microsemi-spi-overlay.dts into platform/raspberry/kernel/linux/arch/arm/boot/dts
    4. Modify platform/raspberry/kernel/linux/arch/arm/boot/dts/Makefile to add

dtb-$(RPI\_DT\_OVERLAYS) += microsemi-spi-overlay.dtb

* + 1. Build any target with HBI=SPI as make command line input.

Example, if building hbilnx then

make hbilnx HBI=SPI Or

make hbilnx

## Enabling an overlay into Pi

* + 1. Boot Pi
    2. Copy the built microsemi-spi-overlay.dtb into /boot/overlays
    3. Modify /boot/config.txt with this info

dtoverlay = microsemi-spi-overlay

* + 1. reboot pi
    2. check devices inside /sys/bus/spi/devices spidev0.0 **should not** be present
    3. do lsmod, check for spi\_bcm2708 module, it should be present
    4. sudo insmod hbi.ko
    5. open device using echo 0:0 > /proc/hbi/open\_device. We use SPI bus 0, chip select 0 on Pi to connect vproc devices
    6. Successful open should create /proc/hbi/dev\_00

## Testing HBI Driver

HBI driver can be tested using HBI linux driver procfs interface (enabled/disabled based on HBI\_ENABLE\_PROCFS option). For details of these interface, please refer to HBI\_Linux\_Driver\_Specification.docx

# Troubleshooting

1. Audio not coming out of speaker on “aplay”
2. This may be because of loose i2s connections between pi and vproc device Or clock not coming properly to vproc as Microsemi vproc device configured as i2s slave.