ZLS38100 SDK Build Guide On Raspberry Pi Platform

V1.0

Table of Contents

Contents

Document Owner	Shally Verma
Version	1.0
Date	June 2016

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
16 Dec 2016	1.1	Shally verma	Added Mixer Driver Build,Load and Test instructions

Abbreviations:

SDK	Software Development Kit
RPI	Raspberry Pi
VPROC	Voice Processor
HBI	Host Bus Interface
SSL	System Service Layer
Distro	Raspberry Pi bootable image. Also referred as “Raspberry Pi Distribution Package”. Current linux based distribution package is termed as Raspbian. Throughout this document terms Distro, Raspbian, Raspberry pi Distribution image will be used interchangeably to refer to linux based bootable image.

Table 1 Abbreviations

1. Purpose

This document describes build and load instructions of ZLS38100 SDK on Raspberry Pi platform.

The SDK support all Linux and non-Linux platforms. However, the build instructions provided in this document is primarily for Linux-based Raspberry Pi distribution packages and compatible for other Linux-based host platforms.

2. Disclaimer

Please note that ZLS38100 SDK has been tested with Raspberry Pi Distribution Image as mentioned in `RPI Development System` section.

Raspberry Pi project and platform is ever evolving with release of new distribution packages on linux and non-linux based operating system. If using any other operating system or latest image, user may need to port SDK as per system of that Raspberry Pi platform.

Generic SDK guideline to port ZLS38100 on host platform is given in ZLS38100_SDK_Porting_Guide.pdf.

3. References

[1] https://www.raspberrypi.org/documentation/

4. Assumptions

This document assumes that user is familiar to boot Raspberry pi using linux based distribution image and has up and running Raspberry Pi platform. Also, user either has UART or Display connected to Raspberry Pi and understand how to use them. If not, user can go to https://www.raspberrypi.org/downloads/raspbian/ and follow instructions there on how to boot raspberry pi.

If UART connection is desired, user can refer to this link http://elinux.org/RPi_Serial_Connection

5. RPI Development System

Raspberry Distro	`2015-05-05-raspbian-wheezy`
Raspberry Pi Board	`Raspberry Pi 1 Model B rev 2`
Toolchain	`gcc-linaro-arm-linux-gnueabihf-raspbian (for cross compilation on 32-bit system)`

Table 2 Development Platform

6. ZLS38100 SDK Release Package

ZLS38100 SDK Release Package consists of following main files.

Files	Description
Makefile.globals	System hardware and software configurations are defined here. Example, host endian can be big or little. Maximum number of vproc devices in a systems etc. User may omit the options not relevant to their system. User may port this file as per their development system.
config.mk	Converts the Makefile.globals variables to ‘C’ compiler options and add relevant include paths. User may port this file as per their development system
drivers/hbi	VPROC HBI Driver.

platform/raspberry/driver/	VPROC Sound and SPI/I2C Driver specific to development platform. Though SSL driver support for both SPI and I2C, however by default SDK build for SPI based implementation. Document **only** cover enabling device on i2s and spi interface of Raspbian.
Platform/raspberry/kernel/dts	Device Tree Overlay to register Microsemi VPROC device as sound and spi device.
Apps	User space sample applications
lnxdrivers	Linux kernel and user space driver for HBI

Table 3 Release Package

7. Building ZLS38100 SDK

7.1 Setting up File Sharing with Raspberry Pi

If you want to setup Raspberry Pi to be accessible from Windows PC then refer to Appendix A section on How to setup Pi for file sharing steps else skip to next section.

7.2 Setting up Raspberry as Build platform

To setup Raspberry Pi as Build platform for module building (if not already setup) then refer to Appendix B before proceeding to next section.

7.3 Install Device Tree Compiler

Required for building Device Tree Overlays

```
sudo apt-get install device-tree-compiler

OR

If you have setup Raspberry Pi as build platform then can also
find at /home/pi/linux/scripts/dtc/dtc
```

Reference https://www.raspberrypi.org/documentation/configuration/device-tree.md

7.4 Building SDK

Make sure to define relevant variables in Makefile.globals before compiling an SDK.

Important ones include:

`PLATFORM`	`raspberry. should always be set to raspberry`
`HOST_ENDIAN`	`little. If raspberry pi distro is compiled for little endian`
`HBI`	`I2C. if using i2c interface. If not mentioned, by default driver will be compiled for SPI`
`CHIP`	`Set to 38051 if building test_doa( direction-of-arrival test for 38051 devices).`
`KSRC`	`Path to kernel source directory. Required for native compilation`
`TOOLSPATH`	`PATH to toolchain. Leave it commented for native compilation`
`CROSS-COMPILER`	`Leave it blank for native compilation`
`ARCH`	`set to arm for raspberry pi`

7.4.1 Go to ZLS38100 Release Directory. This is considered as **ROOT** Directory. Any directory and file reference made using **`./ (please notice . before /)`** means that file or directory is accessed when present working directoy is ROOT.

7.4.2 Make sure variables in Makefile.globals are set as per table above and KSRC set to

```
/lib/modules/`uname -r`/build
```

7.4.3 Give command `make hbilnx`

This will build `/platform, /hbi and /hbilnx` components and save output `hbi.ko` in `/libs` directory

7.4.4 Give command `make apps`

This will build `hbi_test` by default

OR

```
make apps TEST_XXX=1
```

where `xxx= test options ex. TEST_DOA` to build direction-of-arrival test for ZL38051

Test binaries are built inside `./apps` directory

Refer to `ZLS38100_Apps.pdf` inside `/doc` folder for more information.

7.4.5 Individual component make rules are also supported. Some of them listed below:
7.4.5.1 `make platform` – builds complete `/platform` directory
7.4.5.2 `make ssl` – build only SSL driver as `ssl.o`
7.4.5.3 `make hbi` – build only hbi driver as `hbi.o`
7.4.5.4 `make hbilnx` – build kernel loadable module `hbi.ko`
7.4.5.5 make codec – build sound driver inside `/platform` directory

User can refer to master Makefile available at Root Directory for other supported target types.

Every rule is followed by clean with naming convention `target_clean`. Example, to clean `hbi.o` use command `make hbi_clean`. To clean `hbi.ko`, use `make hbilnx_clean`.

8. Loading ZLS38100 SDK on Pi

Raspberry Pi platform need Device Tree overlays to be loaded and enabled in Raspbian image.

Make sure your Raspberry Pi Distro using device tree. Some of the raspi-distro require manual enable of device tree so user may try following:

do `sudo raspi-config`.
Go to `Advanced Options -> Device Tree -> Yes`

ZLS38100 package comes with Device Tree Source file to register Microsemi VPROC device as i2s and spi device. These files are available at `/platform/raspberry/kernel/dts` and corresponding blob are present at `./libs also copied to /boot/overlays` directory.

In following section, first we will cover how to enable Device Tree Overlay followed by Load and Run instructions of Sound and HBI driver.

8.1 Enabling Device Tree Overlay

8.1.1 Open `/boot/config.txt`, add following:

```
sudo nano /boot/config.txt
```

```
dtparam=i2s=on
dtoverlay=microsemi-dac-overlay
dtoverlay = microsemi-spi-overlay
```

8.1.2 Reboot Raspberry Pi

8.1.3 Ensure Microsemi DAC device is detected by kernel by doing following checks. Please note these steps may vary a bit depending on your Raspberry Pi distro.

8.1.3.1 `ls -l /sys/bus/platform/devices/zl380-codec` should be present

8.1.3.2 `cat /sys/bus/platform/devices/sound/modalias` should display a string

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

8.1.3.3 `ls -l /sys/bus/spi/devices` spidev0.0 **should not** be present

8.1.3.4 do `lsmod`. Check for `spi_bcm2708 or spi_bcm2835 module (Raspbian image may have either of these two).` If your Raspbian image doesn’t have `spi_bcm2708`, you can use `spi_bcm2835` provided it support SPI bus VPROC device is connected to.

8.2 Loading HBI Driver

```
sudo insmod ./libs/lib/modules/`uname -r`/extra/hbi.ko
```

Successful load will result in `/dev/hbi` and `/proc/hbi` Directory.

8.3 Testing HBI Module

8.3.2 Sample Apps

Sample applications are available at `./apps` directory.
Please refer to `./docs/ZLS38100_Apps.pdf`

8.3.3 PROCFS Interface

PROCFS quick tool to test everything works on SPI

HBI module can be tested using HBI linux driver procfs interface (enabled/disabled based on HBI_ENABLE_PROCFS option). For details of supported PROC interfaces, please refer to `/docs/ZLS38100_HBI_Linux_Driver_Specification.docx`

Example proc command sequence to read bytes from register

8.3.3.1 Initialize driver

```
cat /proc/hbi/init_driver
```

8.3.3.2 Open device at bus 0, chip select 0

```
echo 0:0 > /proc/hbi/open_device
```

This will result in `/proc/hbi/dev_00/` directory

8.3.3.3 Read 10 bytes from reg 0x200

```
echo 200 10 > /proc/hbi/dev_00/read_reg
```

Dump the read bytes

```
cat /proc/hbi/dev_00/read_reg
```

8.3.3.4 Close device at bus 0

```
echo 0 > /proc/hbi/close_device
```

8.3.3.5 Terminate driver

```
cat /proc/hbi/term_driver
```

Example proc command to write registers

8.3.3.6 Initialise driver

```
cat /proc/hbi/init_driver
```

8.3.3.7 Open device at bus 0, chip select 0

```
echo 0:0 > /proc/hbi/open_device
```

8.3.3.8 Write register

```
echo 200 DEADBEEF > /proc/hbi/dev_00/write_reg
```

8.3.3.9 Read back written value

```
echo 200 8 > /proc/hbi/dev_00/read_reg
cat /proc/hbi/dev_00/read_reg
```

8.3.3.10 close device

```
echo 0 > /proc/hbi/close_device
```

8.3.3.11 terminate driver

```
cat /proc/hbi/term_driver
```

8.4 Loading Sound Driver on Raspberry Pi

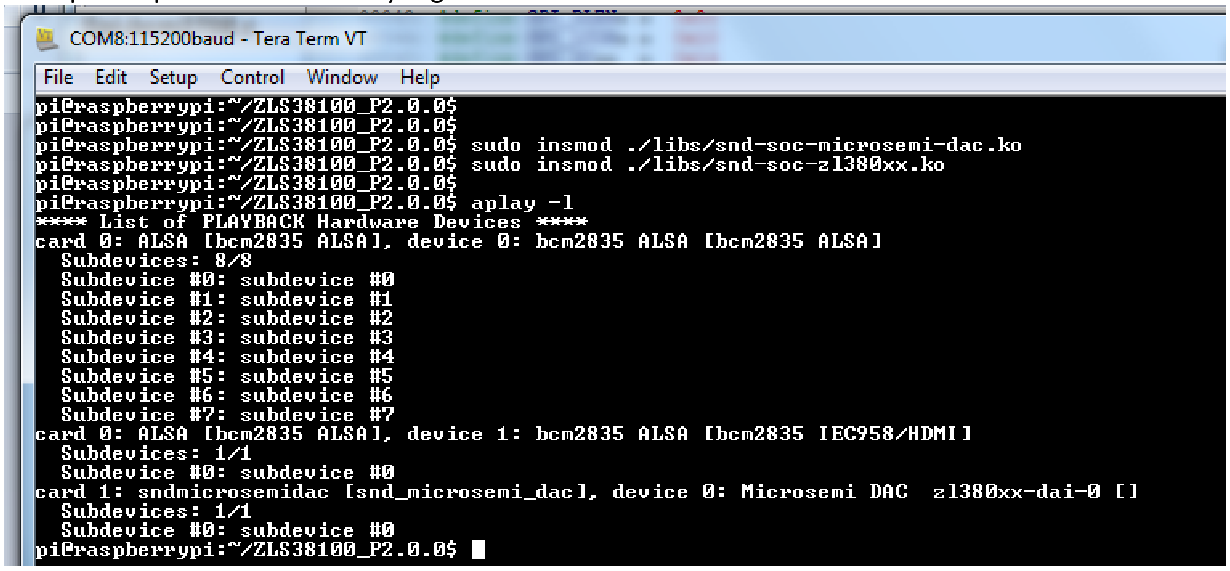
8.4.1 Do "`sudo insmod ./libs/snd-soc-zl380xx.ko`"

8.4.2 Do "`sudo insmod ./libs/snd-soc-microsemi-dac.ko`"

8.4.3 Now ensure Microsemi machine and codec driver is registered, you can perform following steps:

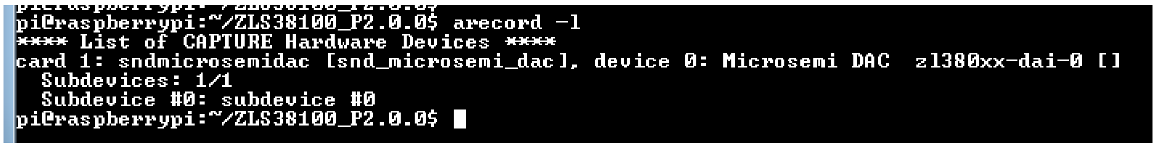
Run "aplay –l" .. it should show "sndmicrosemidac" as one of the registered card

Example snapshot of successfully registered card



COM8:115200baud - Tera Term VT

```
pi@raspberrypi:~/ZLS38100_P2.0.0$
pi@raspberrypi:~/ZLS38100_P2.0.0$
pi@raspberrypi:~/ZLS38100_P2.0.0$ sudo insmod ./libs/snd-soc-microsemi-dac.ko
pi@raspberrypi:~/ZLS38100_P2.0.0$ sudo insmod ./libs/snd-soc-zl380xx.ko
pi@raspberrypi:~/ZLS38100_P2.0.0$
pi@raspberrypi:~/ZLS38100_P2.0.0$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: sndmicrosemidac [snd_microsemi_dac], device 0: Microsemi DAC  zl380xx-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi:~/ZLS38100_P2.0.0$
```



```
pi@raspberrypi:~/ZLS38100_P2.0.0$ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: sndmicrosemidac [snd_microsemi_dac], device 0: Microsemi DAC  zl380xx-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi:~/ZLS38100_P2.0.0$
```

Now your device is ready for playback and capture

8.5 Testing Sound Driver

8.5.1 To record an audio file, you can run "arecord" app.

For example to record Signed 16 bit Little Endian Stereo, 16Khz wav file, from Microsemi VPROC device registered as card 1. Run

```
arecord -D "hw:1,0" -c 2 -f S16_LE -r 16000 test.wav
For list of other options, run "arecord -h"
```

8.5.2 To playback recorded file, run aplay app.

For example, to play recorded S16 Little Endian, 16Kz Stereo
aplay test.wav

```
aplay -D “hw:1,0” test.wav (if vproc device registered as card 1)
```

8.6 Building Mixer Driver

SDK now support another variant of codec driver which supports various mixer controls. In order to compile this no special command is required. Simply run make hbilnx and mixer driver will output as snd-soc-zl380xx-mixer.ko.

Note:
User cannot use both snd-soc-zl380xx and snd-soc-zl380xx-mixer.ko simultaneously, reason is both version registers themselves with same codec name and thus raspberry pi Microsemi dac machine driver will see two codecs registered with same name (See struct snd_soc_dai_link snd_microsemi_dac_dai[] { . codec_name=zl380-codec }). Thus, If user have snd_soc_zl380xx already installed, he should uninstall it using command ‘sudo rmmod snd_soc_zl380xx’ and then insmod snd-soc-zl380xx-mixer.ko

8.7 Testing Mixer Driver

8.7.1 Record and playback

```
Follow usual aplay and arecord commands to playback and recording
apps
```

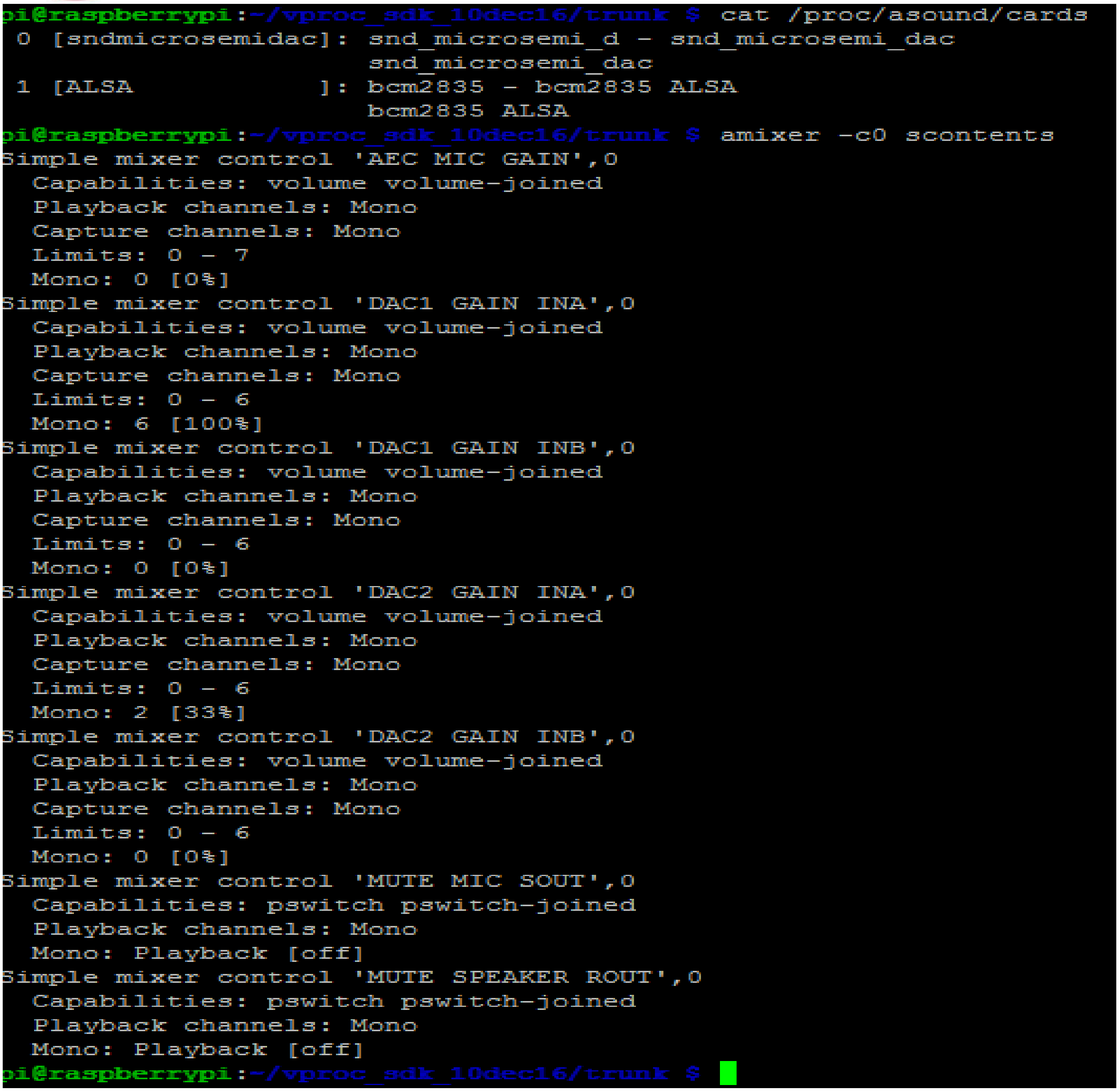
8.7.2 Testing Mixer Controls

Mixer controls can be tested using standard alsa `amixer` app commands.

Example , to see supported mixers controls

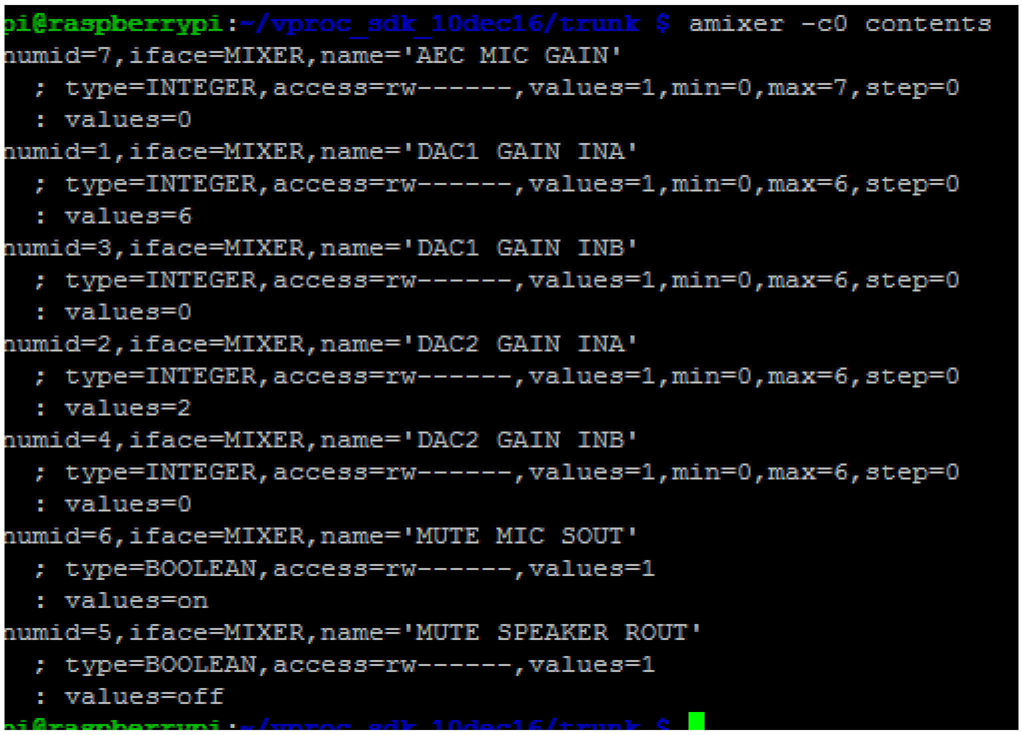
```
amixer -c<card_num> scontents , and
amixer -c<card_num> contents
```

Example snapshot of `amixer -c<card_num> scontents` for a system where `sndmicrosemidac` registered as `card 0`



```
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $ cat /proc/asound/cards
 0 [sndmicrosemidac]: snd_microsemi_d - snd_microsemi_dac
                      snd_microsemi_dac
 1 [ALSA           ]: bcm2835 - bcm2835 ALSA
                      bcm2835 ALSA
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $ amixer -c0 scontents
Simple mixer control 'AEC MIC GAIN',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 7
  Mono: 0 [0%]
Simple mixer control 'DAC1 GAIN INA',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 6
  Mono: 6 [100%]
Simple mixer control 'DAC1 GAIN INB',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 6
  Mono: 0 [0%]
Simple mixer control 'DAC2 GAIN INA',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 6
  Mono: 2 [33%]
Simple mixer control 'DAC2 GAIN INB',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 6
  Mono: 0 [0%]
Simple mixer control 'MUTE MIC SOUT',0
  Capabilities: pswitch pswitch-joined
  Playback channels: Mono
  Mono: Playback [off]
Simple mixer control 'MUTE SPEAKER ROUT',0
  Capabilities: pswitch pswitch-joined
  Playback channels: Mono
  Mono: Playback [off]
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $
```

```
amixer -c<card_num> contents
```



```
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $ amixer -c0 contents
numid=7,iface=MIXER,name='AEC MIC GAIN'
  ; type=INTEGER,access=rw------,values=1,min=0,max=7,step=0
  : values=0
numid=1,iface=MIXER,name='DAC1 GAIN INA'
  ; type=INTEGER,access=rw------,values=1,min=0,max=6,step=0
  : values=6
numid=3,iface=MIXER,name='DAC1 GAIN INB'
  ; type=INTEGER,access=rw------,values=1,min=0,max=6,step=0
  : values=0
numid=2,iface=MIXER,name='DAC2 GAIN INA'
  ; type=INTEGER,access=rw------,values=1,min=0,max=6,step=0
  : values=2
numid=4,iface=MIXER,name='DAC2 GAIN INB'
  ; type=INTEGER,access=rw------,values=1,min=0,max=6,step=0
  : values=0
numid=6,iface=MIXER,name='MUTE MIC SOUT'
  ; type=BOOLEAN,access=rw------,values=1
  : values=on
numid=5,iface=MIXER,name='MUTE SPEAKER ROUT'
  ; type=BOOLEAN,access=rw------,values=1
  : values=off
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $
```

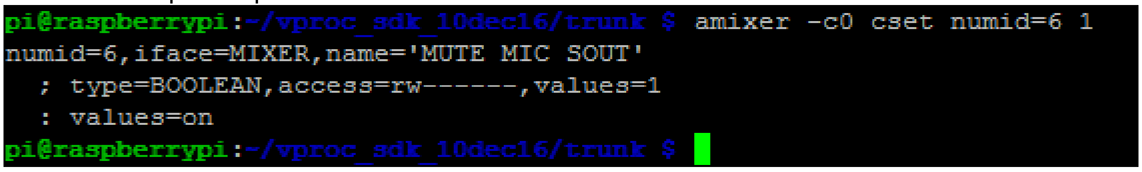
`amixer -c0 contents` command gives you `numid` specific to each control. User can use this `numid to get` and set values of specific controls using following commands:
For writing: `amixer -c<card_num> cset numid=<controd id> <val>`
For reading: `amixer -c<card_num> cget numid=<controd id>`

Example to read mixer control value with id 6 from card 0 `amixer` command would be:

`amixer -c0 cget numid:1`

To write mixer control value 1 with id 6 from card 0 `amixer` command would be, `amixer -c0 cset 1`

Example snapshot of `cset` command



```
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $ amixer -c0 cset numid=6 1
numid=6,iface=MIXER,name='MUTE MIC SOUT'
  ; type=BOOLEAN,access=rw------,values=1
  : values=on
pi@raspberrypi:~/vproc_sdk_10dec16/trunk $
```

Appendix

A. Set Samba File Sharing

This section covers how to setup samba server and share on Raspberry so that user can access Raspberry Pi home drive from windows PC. Though these steps should be able to work for all linux based distro package however certain settings in samba configuration file `smb.conf` may differ a bit per each Raspberry linux distro package.

When you log in to Pi and open a terminal window or you boot to command line (example UART console) instead of the graphical user interface, you start in your home folder `/home/pi`, assuming your username is `pi`.

This is where the user's own files are kept. The content of the user's desktop is in directory here called `Desktop` along with other files and folder.

This section will make `/home/pi` as samba share so that user can access it from windows PC and transfer content to and from it.

Make sure that you have Ethernet cable connected to Raspberry Pi and that it has valid IP Address before continuing to next step.

a) Install samba server on to raspberry pi
```
sudo apt-get install samba samba-common-bin
```
If you get an error, run `sudo apt-get update` and retry.

b) Open Samba configuration file
```
sudo nano /etc/samba/smb.conf
```

c) Search for the section marked `####Authentication###` and change text from
```
#security=user
```
to
```
security=user
```

d) Search for section [`homes`] and set `read only = no`

e) Make `/home/pi` as samba accessible directory. Add following at the end of file:
```
[public]
  comment = Public Storage
  path = /home/pi
  valid users = @users
  force group = users
  create mask = 0660
  directory mask = 0771
  read only = no
```

f) Save and Close `smb.conf`

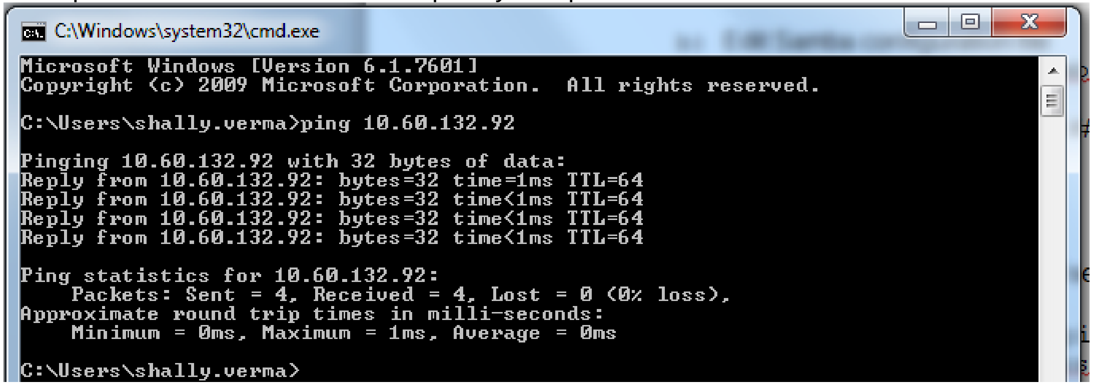
g) If not set, set `/home/pi` owner and mode as:
```
sudo chown -R root:users /home/pi
sudo chmod -R ug=rwx,o=rx /home/pi
```

h) By default user `pi` is defined. To allow user named "`pi`" an access to samba share, give following command and enter your password twice
```
sudo smbpasswd -a pi
```
for demonstration purpose, we use samba password `raspberry` here.

i) Start samba service
```
sudo /etc/init.d/samba start
```

j) Note Raspberry Pi IP Address by running command `ifconfig`

k) Ping address from windows to ensure you have connectivity between Pi and Windows PC.
Example screenshot below with Raspberry Pi <ipaddress> = 10.60.132.92



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\shally.verma>ping 10.60.132.92

Pinging 10.60.132.92 with 32 bytes of data:
Reply from 10.60.132.92: bytes=32 time=1ms TTL=64
Reply from 10.60.132.92: bytes=32 time<1ms TTL=64
Reply from 10.60.132.92: bytes=32 time<1ms TTL=64
Reply from 10.60.132.92: bytes=32 time<1ms TTL=64

Ping statistics for 10.60.132.92:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\shally.verma>
```

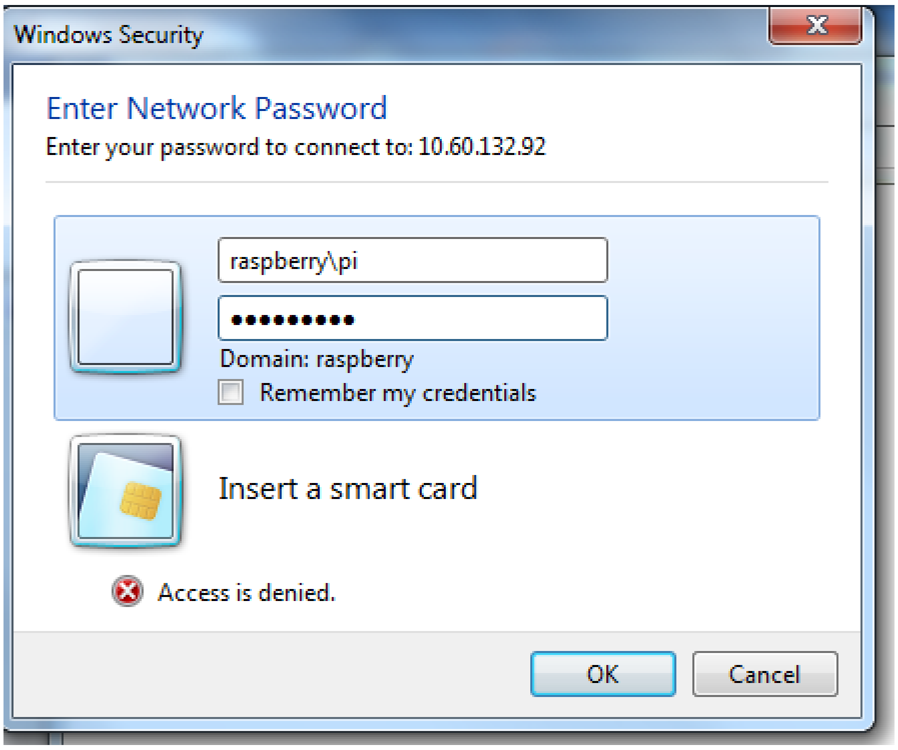
l) Open windows start menu. Give \\<ipaddr>\pi in its search box and press enter or Map Network Drive option on Windows PC

m) It will ask for user name and password as shown in snapshot below
Login with following info:

username : `raspberry\pi`
password : <`password you entered in` step e)>

For our demonstration case
password: raspberry

Example snapshot



Enter Network Password
Enter your password to connect to: 10.60.132.92

raspberry\pi
••••••••••
Domain: raspberry
Remember my credentials

Access is denied.

After this you can copy your SDK package on Raspberry `/home/pi` directory from your windows PC.

Still getting trouble, refer http://elinux.org/R-Pi_NAS

B. Setting up Raspberry for Native Compilation

This step is required for Native compilation and one time activity.
Please note whenever you start with fresh Raspberry Pi and setup activity it may take almost half-a-day to set it up. So be prepared for it.

Raspberry Pi distribution image doesn't come with linux headers which are prerequisite to module building. Thus we need to download kernel tar ball manually which distribution image is based on.

There are two ways to find out and retrieve kernel version.

1) You can refer to raspberry.org for build instructions https://www.raspberrypi.org/documentation/linux/kernel/building.md.

Please note instruction at this page will download latest raspbian kernel so you will need to check compatibility of distro image and kernel downloaded.

2) Another we recommend to use `rpi-source` script. It is available with ZLS38100 SDK at `./platform/raspberry/tools` Directory or can also be downloaded from internet following instructions given in following sections.

`rpi-source` installs the kernel source used to build the kernel in the Raspian image. The script uses sudo internally when self-updating and when making the links*/lib/modules/$(uname -r)/{build, source}*

Please note these steps are to be performed on raspberry pi only

a) Install rpi-source
`rpi-source` is available in ZLS38100 package under ./`platform/raspberry/tools` directory. Alternatively user can get from internet by running following on raspberry pi terminal or UART window:

```
sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source -O /usr/bin/rpi-source && sudo chmod +x /usr/bin/rpi-source
```

b) Run rpi-source

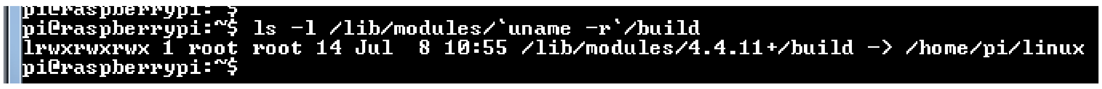
```
rpi-source
```

If run successfully, then `rpi-source` will do everything for you. It will download, build and setup required symbolic links.

If you get error, then continue to next sections "Handling Errors" before proceeding here else continue.

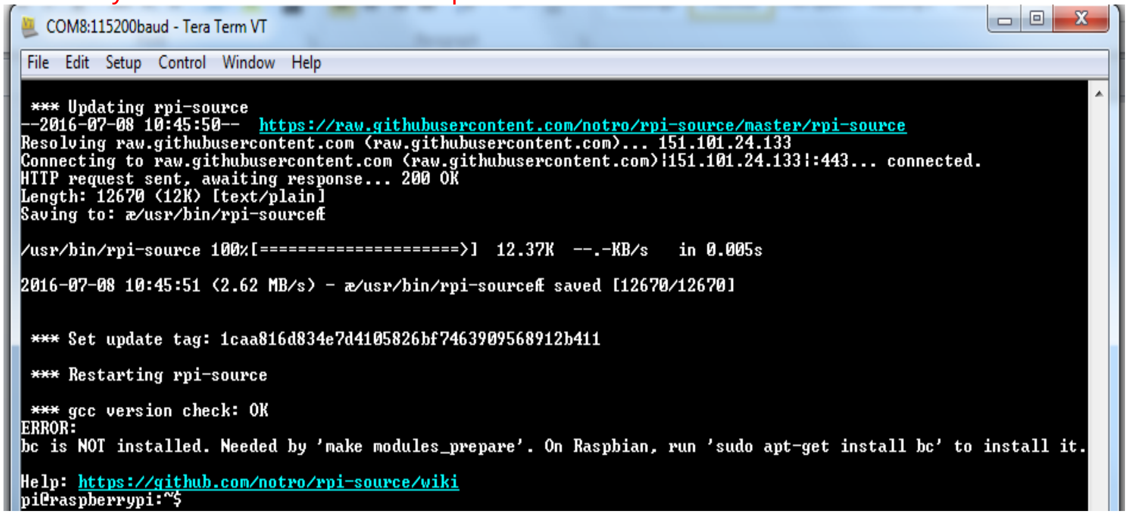
After successful execution, user should see `/home/pi/linux` Directory and `kernel tar ball` in `/home/pi` Directory and `/lib/modules/`uname -r`/build` pointing to `/home/pi/linux`.

Example snapshot of successful linking:



```
pi@raspberrypi:~$
pi@raspberrypi:~$ ls -l /lib/modules/`uname -r`/build
lrwxrwxrwx 1 root root 14 Jul  8 10:55 /lib/modules/4.4.11+/build -> /home/pi/linux
pi@raspberrypi:~$
```

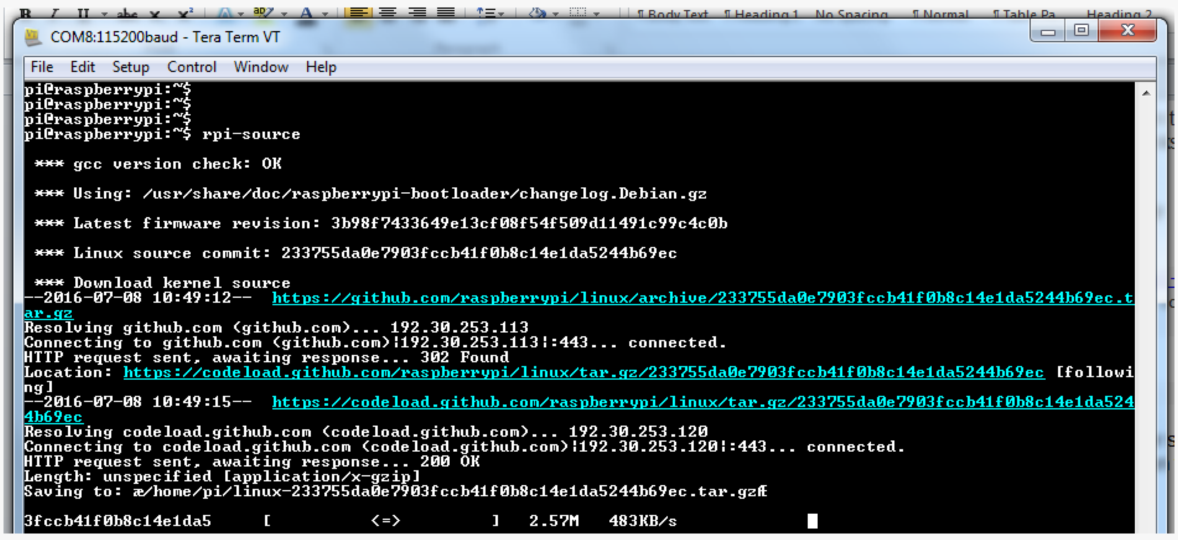
c) Handling Errors
If your error is as shown in snapshot


COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
 *** Updating rpi-source
--2016-07-08 10:45:50--  https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.24.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.24.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12670 (12K) [text/plain]
Saving to: æ/usr/bin/rpi-sourceÆ

/usr/bin/rpi-source 100%[===================>]  12.37K  --.-KB/s   in 0.005s

2016-07-08 10:45:51 (2.62 MB/s) - æ/usr/bin/rpi-sourceÆ saved [12670/12670]


 *** Set update tag: 1caa816d834e7d4105826bf7463909568912b411

 *** Restarting rpi-source

 *** gcc version check: OK
ERROR:
bc is NOT installed. Needed by 'make modules_prepare'. On Raspbian, run 'sudo apt-get install bc' to install it.

Help: https://github.com/notro/rpi-source/wiki
pi@raspberrypi:~$
```

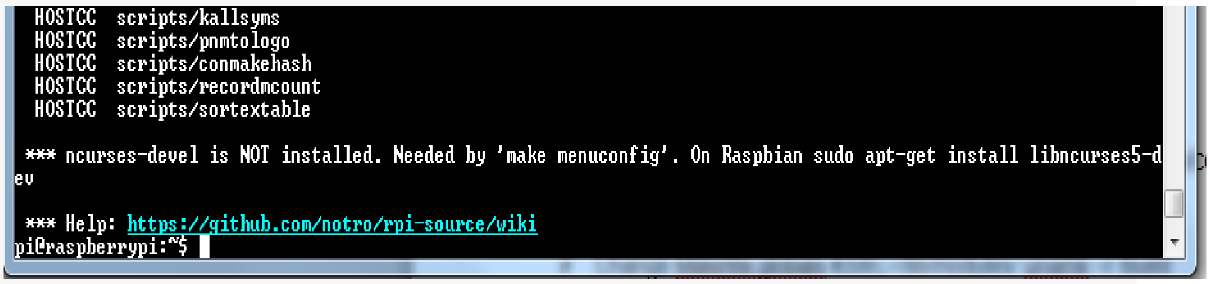
Then run `sudo apt-get install bc` as advised.

Re-run `rpi-source`, a successful run should have snapshot as


COM8:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ rpi-source

 *** gcc version check: OK

 *** Using: /usr/share/doc/raspberrypi-bootloader/changelog.Debian.gz

 *** Latest firmware revision: 3b98f7433649e13cf08f54f509d11491c99c4c0b

 *** Linux source commit: 233755da0e7903fccb41f0b8c14e1da5244b69ec

 *** Download kernel source
--2016-07-08 10:49:12--  https://github.com/raspberrypi/linux/archive/233755da0e7903fccb41f0b8c14e1da5244b69ec.t
ar.gz
Resolving github.com (github.com)... 192.30.253.113
Connecting to github.com (github.com)|192.30.253.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/raspberrypi/linux/tar.gz/233755da0e7903fccb41f0b8c14e1da5244b69ec [followi
ng]
--2016-07-08 10:49:15--  https://codeload.github.com/raspberrypi/linux/tar.gz/233755da0e7903fccb41f0b8c14e1da524
4b69ec
Resolving codeload.github.com (codeload.github.com)... 192.30.253.120
Connecting to codeload.github.com (codeload.github.com)|192.30.253.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: æ/home/pi/linux-233755da0e7903fccb41f0b8c14e1da5244b69ec.tar.gzÆ

3fccb41f0b8c14e1da5     [          <=>          ]   2.57M   483KB/s
```

At the end of run, you may get message as in snapshot:



```
  HOSTCC  scripts/kallsyms
  HOSTCC  scripts/pnmtologo
  HOSTCC  scripts/conmakehash
  HOSTCC  scripts/recordmcount
  HOSTCC  scripts/sortextable

 *** ncurses-devel is NOT installed. Needed by 'make menuconfig'. On Raspbian sudo apt-get install libncurses5-d
ev

 *** Help: https://github.com/notro/rpi-source/wiki
pi@raspberrypi:~$
```

Install `sudo apt-get install libncurses5-dev` as advised.

d) If you get an error message as shown below:

```
ERROR:
gcc version check: mismatch between gcc (4.6.3) and /proc/version (4.8.3)
Skip this check with --skip-gcc
```

Then major.minor version of gcc installed in raspberry distribution package is different from the one used to build distribution package. You can cross-check difference of gcc version by doing following:

Below are the example run to check the GCC version used to build kernel. Version output may vary depending upon raspbian image used.

```
$ cat /proc/version
Linux version 3.18.11+ (dc4@dc4-XPS13-9333) (gcc version 4.8.3 20140303
(prerelease) (crosstool-NG linaro-1.13.1+bzr2650 - Linaro GCC 2014.03) ) #781
PREEMPT Tue Apr 21 18:02:18 BST 2015
```

Version installed:

```
$ gcc --version | grep gcc
gcc (Debian 4.6.3-14+rpi1) 4.6.3
```

If `cat /proc/version` mention GCC version 4.7, refer to section “Install gcc 4.7”.
If `cat /proc/version` mention GCC version 4.8, refer to section “Install gcc 4.8”

e) Install gcc 4.7

```
 $ sudo apt-get install gcc-4.7 g++-4.7
```

f) Install gcc 4.8

1. Open file
```
sudo nano /etc/apt/sources.list.d/jessie.list
```
2. add line:
```
deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib
non-free rpi
```
3. fetch package lists:
```
sudo apt-get update
```
4. Install
```
sudo apt-get install gcc-4.8 g++-4.8
```

g) Setup GCC versions

1. For GCC 4.7
```
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.6
60 --slave /usr/bin/g++ g++ /usr/bin/g++-4.6
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.7
40 --slave /usr/bin/g++ g++ /usr/bin/g++-4.7
```

2. For GCC 4.8
```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.6
20
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8
50
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.6
20
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.8
50
```

h) Choose GCC Version
Please note below it can show gcc 4.7 if GCC 4.7 installed

```
$ sudo update-alternatives --config gcc
There are 2 choices for the alternative gcc (providing /usr/bin/gcc).

Selection    Path              Priority   Status
------------------------------------------------------------
* 0            /usr/bin/gcc-4.8   50        auto mode
  1            /usr/bin/gcc-4.6   20        manual mode
  2            /usr/bin/gcc-4.8   50        manual mode

Press enter to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/bin/gcc-4.8 to provide /usr/bin/gcc (gcc) in
manual mode
```

i) Check current GCC version
Example run of gcc –version command on system with GCC 4.8 installed.
```
$ gcc --version | grep gcc
gcc (Raspbian 4.8.4-1) 4.8.4
```

j) Now re-run your `rpi-source`

k) If you get an error
Make prepare_modules showing an error
```
#error Your compiler is too buggy; it is known to miscompile kernels
```

This means correct GCC version is not installed. Follow instructions to install correct GCC version as detailed above.

If you run into any issue not covered here, please refer to link https://github.com/notro/rpi-source/wiki.