Linux drivers and kernel building with MERUS™ audio amp HAT ZW

Drivers and kernel building – KIT_40W_AMP_HAT_ZW

About this document

Scope and purpose

This application note describes one of the methods of software driver installation in Raspbian for the MERUS™ audio amp HAT Zero W, which includes the MA12070P proprietary multi-level amplifier. The installation process is carried out as a kernel building in an Ubuntu host machine. This allows the user not only to make use of all of the driver features for the MA12070P device, but also to maintain compatibility with the last Raspbian kernel version and all its loadable kernel modules. This method may also be applied to any kernel customization required by the user prior to compiling and building, and used to add any additional features required on the MA12070P Linux codec driver too. Additionally, a brief review of the driver features is made, and AirPlay set-up for Raspberry Pi is also explained as a tool for quick wireless audio streaming testing.

Intended audience

Audio DIY community, audio amplifier design engineers.

Table of contents

About this document **1**
Table of contents **1**
1 Introduction **2**
2 Kernel building process with MERUS™ audio Linux drivers **3**
2.1 Tools and requirements 3
2.2 Compile, build and install process 3
2.2.1 Download the Infineon MERUS™ audio driver source files 3
2.2.2 Set-up of the build environment 4
2.2.3 Get the Raspbian kernel source 4
2.2.4 Verification of the SD card format 4
2.2.5 Makefile 5
2.3 Configuring the device tree overlay 5
2.4 Configuring the wireless network and SSH 5
3 Exploring Raspberry Pi and MERUS™ audio amp HAT ZW **7**
3.1 First log in to the Raspberry Pi with SSH 7
3.2 MERUS™ audio amp alsamixer 8
4 Music streaming with AirPlay and MERUS™ audio amp HAT **10**
4.1 Installation process for shairport-sync client 10
Revision history **11**

1 Introduction

There are several methods to install drivers or loadable kernel modules in a Linux-based OS. This depends on many factors, such as the target machine, local or cross-compilation building method, and the kernel source, among others. Specifically in the case of Raspbian, which is the OS supported on Raspberry Pi boards, the sound cards' drivers are already built in to the current kernel version of Raspbian. Therefore, there is no need to build and compile the kernel for the built-in modules as described later. These built-in modules are available and ready to use after the standard Raspbian installation process. This last process consists of burning a pre-compiled image file on to a Raspberry Pi SD card. However, at the time of writing this application note the MERUS™ Audio Linux drivers are not included yet in the last Raspbian kernel source and currently the available method to install them consists of manually compiling and building their source files with the Raspbian kernel. After this process has been completed the output files need to be installed on an SD card.

The complete process of compiling, building and installing the drivers is handled by a makefile script, so there is minimal command typing by the user. Another advantage of this method is that it allows the user to add any additional features to the MERUS™ audio Linux drivers for the required application. This is especially aimed at users and makers interested in configuring and testing other power management modes and audio performance parameters in the MA12070P proprietary multi-level amplifier. This method is also useful when a custom kernel with MERUS™ audio Linux drivers is needed.

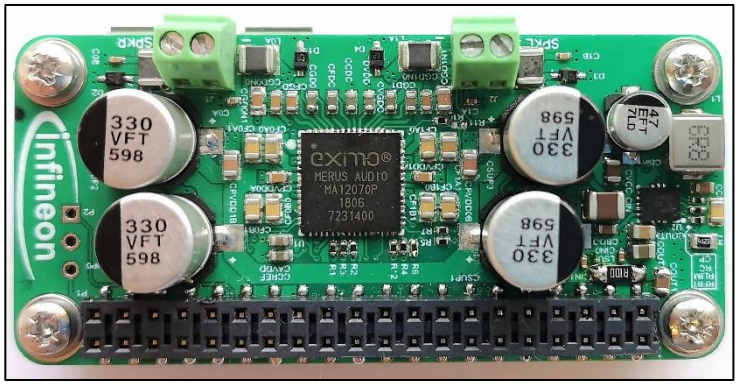


Figure 1 MERUS™ audio amp HAT Zero W board

2 Kernel building process with MERUS™ audio Linux drivers

This section describes how to compile, build and install an image for Raspbian with built-in Linux drivers for the MERUS™ audio amp HAT board. The process consists of acquiring the Raspbian kernel source, then later configuring it with the drivers and cross-compiling them in a local Ubuntu machine. The final build and image is transferred to an SD card that will finally be used with the Raspberry Pi Zero. The entire process is run from a script included in the drivers' download folder so there is minimal manual code to be run by the user outside of the installation script.

2.1 Tools and requirements

In order to build and use the sources successfully, the following tools, sources and files will be needed:

- Ubuntu 16 or a later Linux machine (native or virtual – must be able to read SD cards with or without an adapter)
- Driver folder (downloaded from the Infineon site, explained below)
- Raspbian kernel (downloaded from GitHub, explained below)
- SD card (Raspbian formatted with boot and rootfs partitions)
- 5 V to 2.5 A DC power supply

2.2 Compile, build and install process

This subsection describes the step-by-step installation process of the MERUS™ audio Linux drivers on to a Raspbian SD card.

2.2.1 Download the Infineon MERUS™ audio driver source files

Download or clone the MERUS™ audio Linux drivers from https://github.com/Infineon/merus-audio-amp-hat-zw and save it into the home folder (thismachine//<home>/<user>). The driver version for the Raspberry Pi Zero W will be in the folder named "merus_linux_audio_driver".

The driver folder tree has the following structure:

merus_linux_audio_driver

-Linux

------ arch/arm/boot/dts/overlays/merus-amp-overlay.dts (device tree overlay)

------ sound/soc/bcm/merus-amp.c (sound card driver)

------ sound/soc/codecs/ma120x0.c (codec driver)

-Makefile (makefile that will compile, build and install the kernel)

-README

-Source_code_patch (kernel source patch instructions and description)

2.2.2 Set-up of the build environment

Before starting the building process it is necessary to install the tools packages on the host machine with the following steps:

- Install the following tools and packages by typing:

```
sudo apt-get install git bison flex libssl-dev gcc-arm-linux-gnueabihf make
```

- Install the Raspberry Pi toolchain with the following command:

```
git clone https://github.com/raspberrypi/tools ~/tools
```

- Update the path variable in the Raspberry Pi tools with the following command for 64-bit systems:

```
echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin >> ~/.bashrc
source ~/.bashrc
```

2.2.3 Get the Raspbian kernel source

- Clone the last stable version of the Raspbian kernel source in your home path by typing:

```
cd ~/
git clone --depth=1 https://github.com/raspberrypi/linux
```

This will also include all the built-in and loadable modules and drivers currently supported on Raspbian.

2.2.4 Verification of the SD card format

In order to install Raspbian with the MERUS™ audio Linux drivers it is necessary to have a properly formatted Raspberry Pi SD card. It should have a partition named “boot” and a partition named “rootfs”. Usually the SD card is already supplied in this format with any Raspberry Pi model, or after another Raspbian installation process. If the user only has an empty SD card, a Raspbian installation should be made following the instructions at https://www.raspberrypi.org/documentation/installation/installing-images/.

Once the format has been verified, insert the SD into the host computer.

2.2.5 Makefile

Once the tools and environment have been set, the kernel and driver installation script can be run from the project makefile. The makefile will first patch all of the Raspbian kernel source files, then it will be compiled and finally the output files and boot image will be installed on the SD card.

To start the process, type the command:

```
cd ~/merus_audio_amp_hat_zw/merus_linux_audio_driver
Make all
```

The whole process can take up to 30 minutes depending on the host computer capabilities. After the compiling process the script will ask for the user's password (after approximately 20 minutes). Then it will take a few seconds to finish the installation process.

2.3 Configuring the device tree overlay

Before unmounting the SD card, modify the "config.txt" file located at the boot partition.

Replace the line:

```
dtparam=audio=on
```

with:

```
dtparam=audio=off
```

This will turn off the built-in sound card of the Raspberry Pi.

Next, add the following line below the previous modification in order to load the MERUS™ audio driver during boot-up, and save the file.

```
dtoverlay=merus-amp
```

2.4 Configuring the wireless network and SSH

In order to stream audio via a wireless network it is necessary to configure the file named "wpa_supplicant" located in the rootfs partition in the etc/wpa_supplicant directory. Open the file with a text editor and replace the text inside with:

```
Country=US # Enter here the 2-digit country code
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
network={
ssid="YOUR_NETWORK_NAME"
```

```
        psk="YOUR_PASSWORD"
        key_mgmt=WPA-PSK
}
```

Next, to enable SSH log-in through your host computer create a file called “ssh” inside the boot partition. The file doesn’t need any content, and must not have any extension such as .txt.

Once this file has been saved, the SD card can be unmounted and then inserted into the Raspberry Pi Zero W.

3 Exploring Raspberry Pi and MERUS™ audio amp HAT ZW

The following section describes the user space applications of the MERUS™ audio amp HAT board including the alsamixer utilities and AirPlay configuration for wireless music streaming.

3.1 First log in to the Raspberry Pi with SSH

In order to log in with SSH to the Raspberry Pi it is first necessary to obtain its IP address. This can be done by typing the following command into the host computer terminal:

```
ping raspberrypi.local
```

Note that if the host name of the Raspberry Pi was set before it should be replaced by “raspberrypi” in the previous command.

The following will appear in the terminal output, where the user will be able the read the IP address:

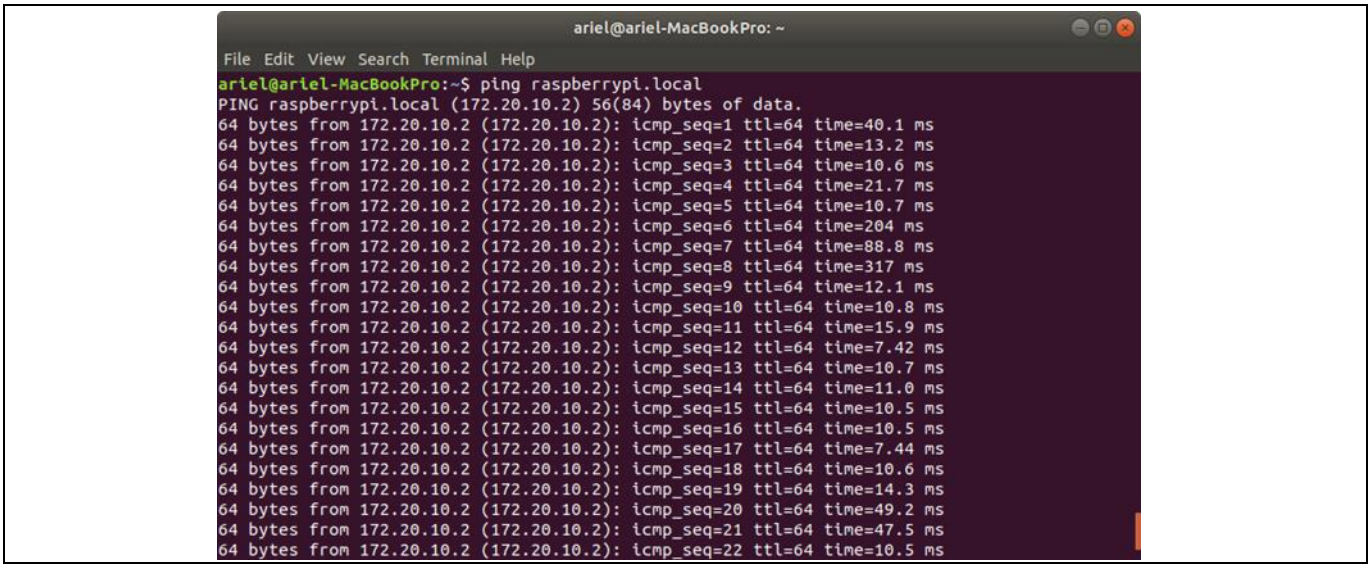


Figure 2 Ping command output

Next, in order to log in remotely to the Raspberry Pi, type the following (including your IP address):

```
ssh pi@<ipadress>
```

Finally, type:

```
aplay -l
```

to check that the drivers' sound card and codec were loaded. If the process was successful the terminal should show the following output prompt:

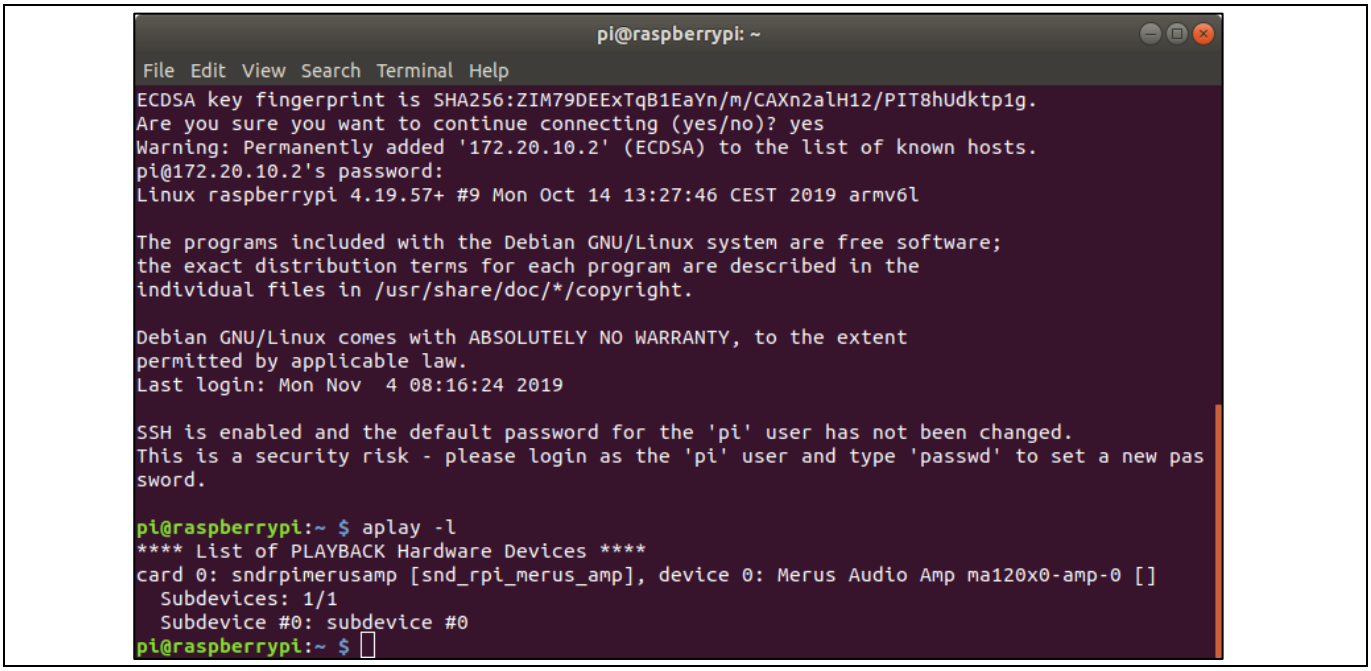


Figure 3 Sound card and codec verification with aplay -l command output

This shows that the default soundcard is "snd_rpi_merus_amp" with "ma120x0-amp-0" codec referring to Infineon's MA120x0P class D amplifier series.

3.2 MERUS™ audio amp alsamixer

The MERUS™ audio MA120x0P codec driver has its own alsamixer where the main control parameters of the MA12070P can be modified. By typing

```
alsamixer
```

the following mixer will be shown:

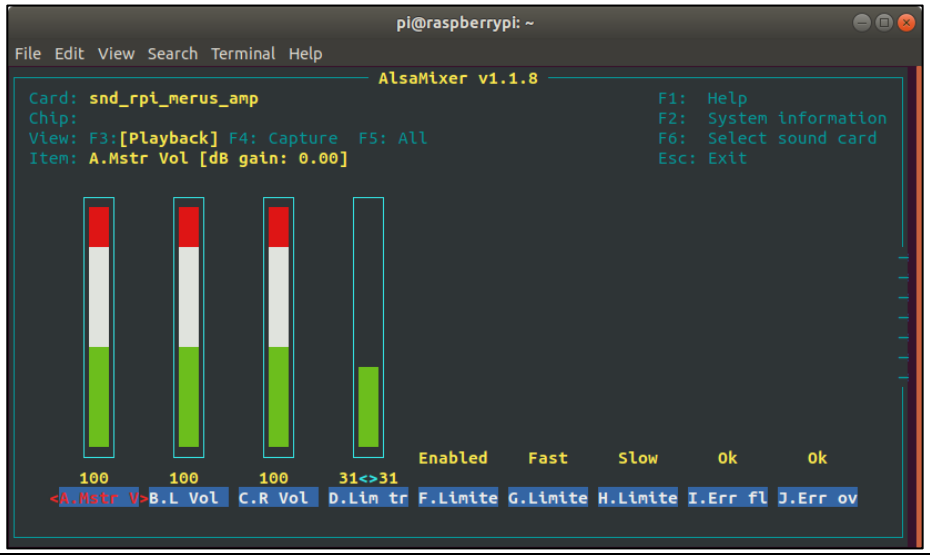


Figure 4 MERUS™ audio amp sound card driver mixer (first page)

By scrolling to the right, the last controls will be shown, as in the following picture:

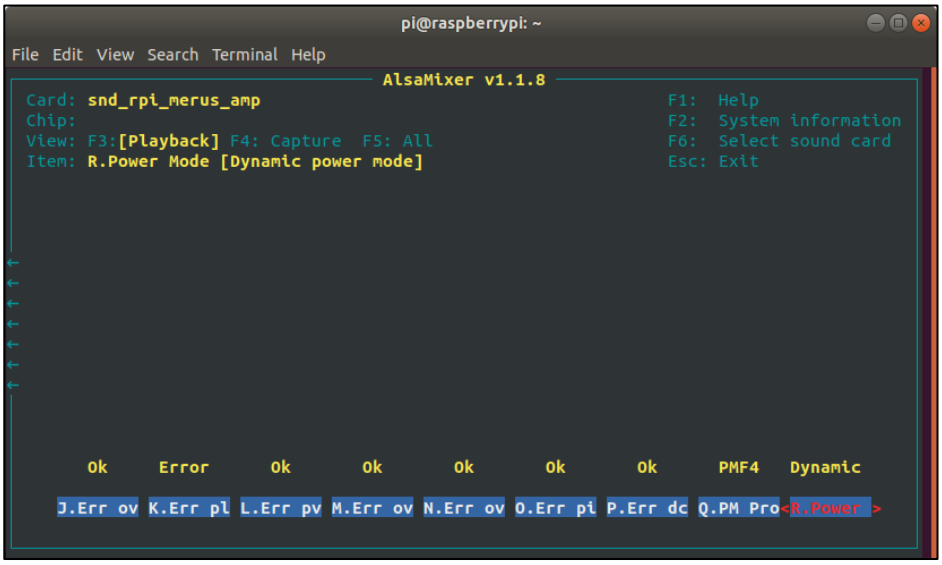


Figure 5 MERUS™ audio amp sound card driver mixer (second page)

It should be noted that all the mixer controls are ordered from left to right and from “A” to “R”. It is recommended to use the item description located at the top-left corner to have a more detailed description of each control and its current status. Increasing master volume and limiter thresholds controls from their default values should be done carefully. Exceeding the recommended settings may damage the devices permanently. For detailed information about volume and limiter settings please refer to the user manual of the board.

4 Music streaming with AirPlay and MERUS™ audio amp HAT

The following section describes how to install and make use of AirPlay with the MERUS™ audio amp HAT and a Raspberry Pi Zero W. AirPlay support in the Raspberry Pi is provided by the shairport-sync client.

4.1 Installation process for shairport-sync client

First, log in to the Raspberry Pi and install the necessary tools and packages by typing:

```
sudo apt-get install autoconf automake avahi-daemon build-essential git
libasound2-dev libavahi-client-dev libconfig-dev libdaemon-dev libpopt-dev
libssl-dev libtool xmltoman
```

Next, install the shairport-sync with the following command:

```
Sudo apt-get install shairpoirt-sync
```

For more information on how to configure additional features and support with shairport-sync, visit:

https://github.com/mikebrady/shairport-sync

Revision history

Document version	Date of release	Description of changes
1.0		Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2019-12-02

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2019 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN_1911_PL88_1912_131733

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.