

Embedded Linux System



OutLine

- ▶ CH1 Introduction to Embedded System p. 1
- ▶ CH2 Introduction Rockpi p. 18
- ▶ CH3 Embedded Linux Arch p. 30
- ▶ CH4 Basic Software And Tool p. 38
- ▶ CH5 Cross-compile Toolchain p. 120
- ▶ CH6 Introduction to Bootloader (u-boot) p. 142
- ▶ CH7 Embedded Linux Kernel p. 194
- ▶ CH8 RootFS p. 225
- ▶ CH9 Linux User Land p. 259
- ▶ CH10 Linux Device Driver p. 271

CH1 Introduction to Embedded System



- ▶ 嵌入式系統 定義
- ▶ 嵌入式系統 相關知識
 - ▶ SOC : ARM, MIPS, PowerPC ...
 - ▶ 週邊設備 : Flash, eMMC, I2C, I2S,
- ▶ 嵌入式作業系統 基本原理
- ▶ 開發版 NanoPi-M4 介紹
- ▶ 外掛模組介紹：
 - ▶ 溫溼度模組，亮度模組 ..
- ▶ 學習建立 嵌入系統開發環境

CH2 Introduction RockPi4B

- To know RockPi4B EVK



CH3 Embedded Linux Arch

- To know Embedded Linux System

CH4 Basic Software And Tool

- » GPL
- » Tool
 - Develop Tool
 - Video and Audio Tool
 - WiFi and Ethernet
 - Build Code Tool

CH5 Cross-compile Toolchain

- ▶ How to Cross-compile with Toolchain
- ▶ Create a Linux library to use

CH6 Introduction to Bootloader

- To know Bootloader
- How to U-boot
- Detail into U-boot



CH7 Embedded Linux Kernel

- » How to build Linux kernel
- » How to scale Linux kernel
- » To know Linux kernel something



CH8 RootFS

- To know RootFS in embedded system
- How to create a RootFS with Buildroot



CH9 Linux User Land

- To know Linux user land
- To know user land application
- How to control device on the Linux user land



CH10 Linux Device Driver

- » To know Linux device driver
- » How to build a Linux device driver
- » How to install/remove a device driver

Introduction to Embedded System



Embedded System

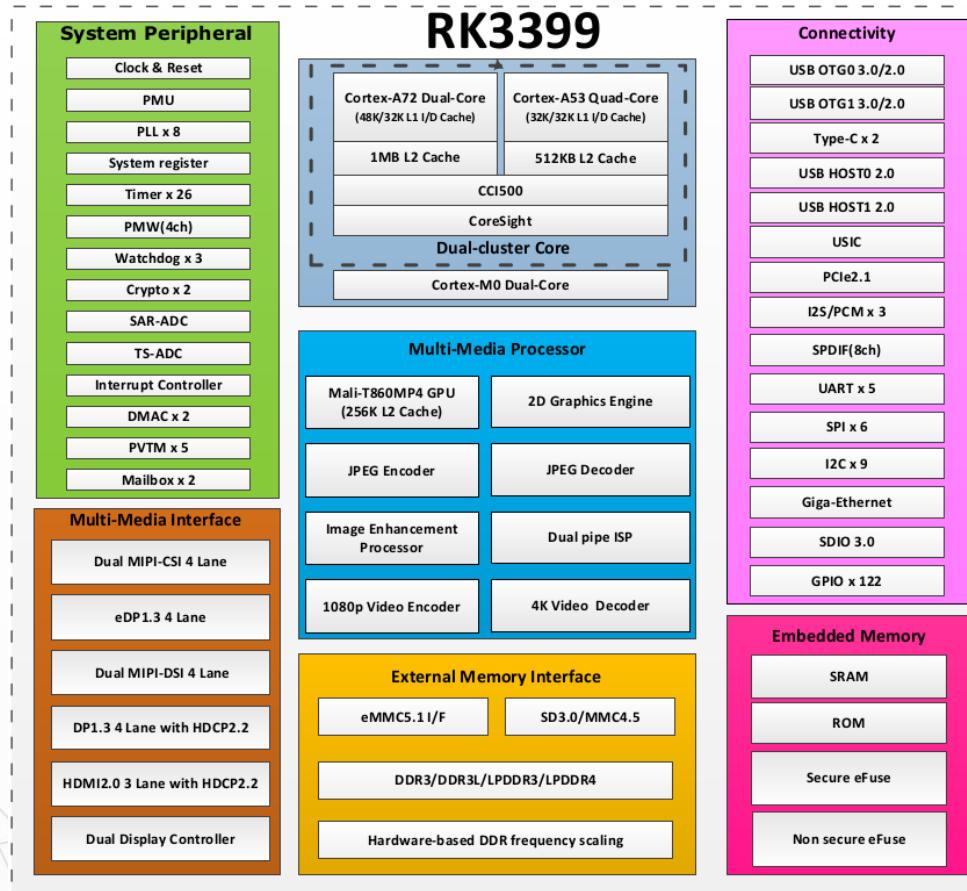
- An embedded system
 - combination of computer hardware and software
 - specifically designed for a particular function
- Applications
 - Mobile phone
 - Digital camera
 - Smart TV
 - Navigation system



Feature

- Designed to do some specific task
 - Low power
 - Small size
 - Special operating ranges
 - Low cost
- Install OS ?

SOC RK3399



http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension



Component of embedded system

- Processor
 - ARM, X86, MIPS
- RAM
 - 8MB ~ 2 GB
- Storagee
 - Nand, Nor flash
 - SD/MMC/eMMC
- System Bus
 - AMBA, AHB, APB, AXI ...



Component of embedded system

- Communication
 - I2C, I2S, USB, PCI/PCIe ...
- Media system
 - JPEG, H.264 ..
- System component
 - DMA, RTC ..



Embedded Linux ?

Embedded Linux is the usage of the
Linux kernel and various
open-source components in
embedded systems
(from Free Electrons)

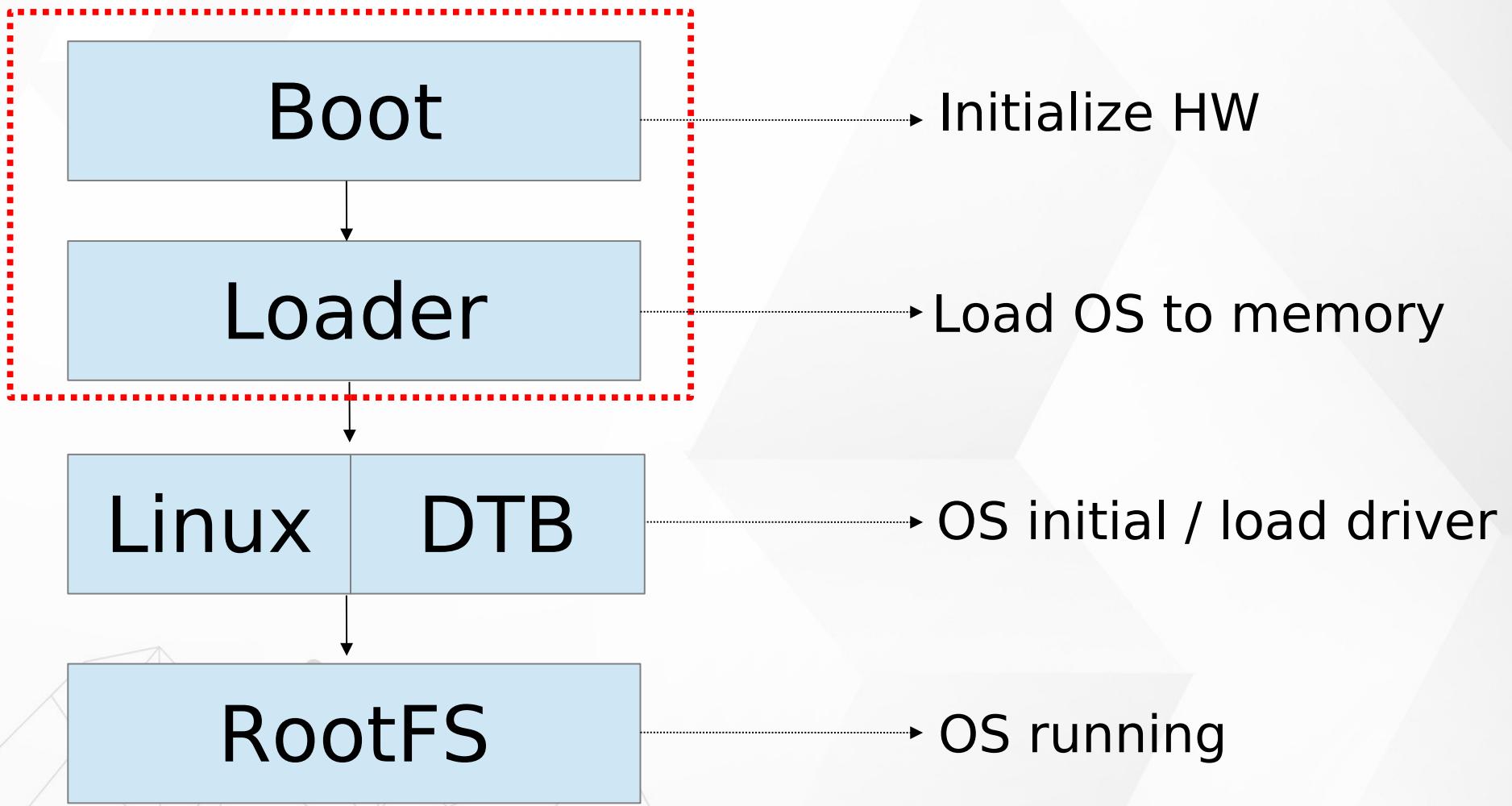


Advantages

- Re-use components
- Quickly design and develop complicated products
- No need to re-develop components
 - TCP/IP stack, USB stack, PCI stack ...
- Allow you modify components



Embedded Linux System Booting





Embedded Linux System Software components

- Cross-compilation toolchain
- Bootloader
- Linux Kernel, DeviceTree
- Rootfs
- C library
- Libraries and applications
- BSP (Board Support Package)

Develop Environment



Develop Environment

- Host PC
- Toolchain
- Target EVB (RockPi4)
- BSP

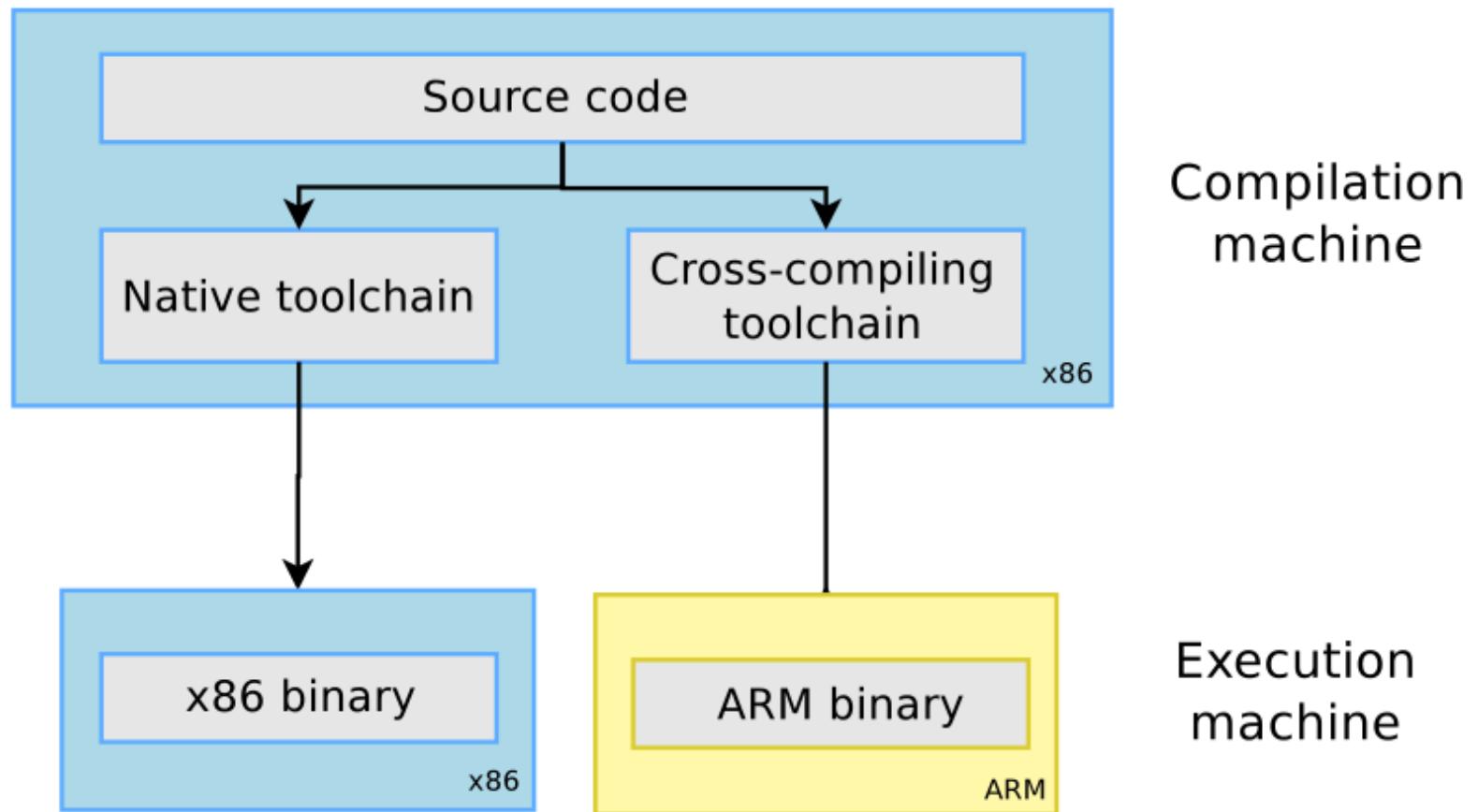


BSP

- Board Support Package
- From chip vendor
 - Distribution
 - Bootloader
 - Linux kernel
 - Device driver
 - Rootfs



Cross Compilation toolchain





Setup References - 1

Debian Image :

<https://github.com/radxa/rock-pi-images-released/releases>

Image Write Tool :

<https://www.balena.io/etcher/>

Install Image to SD Card :

<https://wiki.radxa.com/Rockpi4/install/microSD>

Setup References - 2



Debug Port :

<https://wiki.radxa.com/Rockpi4/dev/serial-console>

Linaro ToolChain :

https://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz

Install Package :

`sudo apt-get install libncurses5 libncurses5-dev build-essential libssl-dev mtools bc python dosfstools liblz4-tool`



Setup References - 3

- u-boot :** <https://wiki.radxa.com/Rockpi4/dev/u-boot>
- Linux Kernel :** <https://wiki.radxa.com/Rockpi4/dev/kernel-4.4>
- Debian :** <https://wiki.radxa.com/Rockpi4/dev/Debian>
- RockPi4 WiKi :** <https://wiki.radxa.com/Rockpi4>

Rock Pi



RockPi WiKi

» Rock Pi Wiki

» <https://wiki.radxa.com/Rockpi4>

» Rock Pi Hardware

» <https://wiki.radxa.com/Rockpi4/hardware/rockpi4>

» Rock Pi Image

» <https://wiki.radxa.com/Rockpi4/downloads>



Create Boot SD Card

▶ Rock Pi Image

▶ <https://wiki.radxa.com/Rockpi4/downloads>

▶ Write Image to SD Card

▶ https://wiki.radxa.com/Rockpi4/getting_started

▶ Etcher Tool

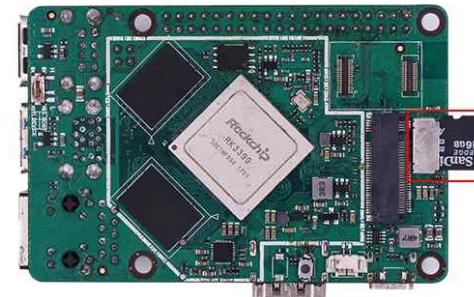
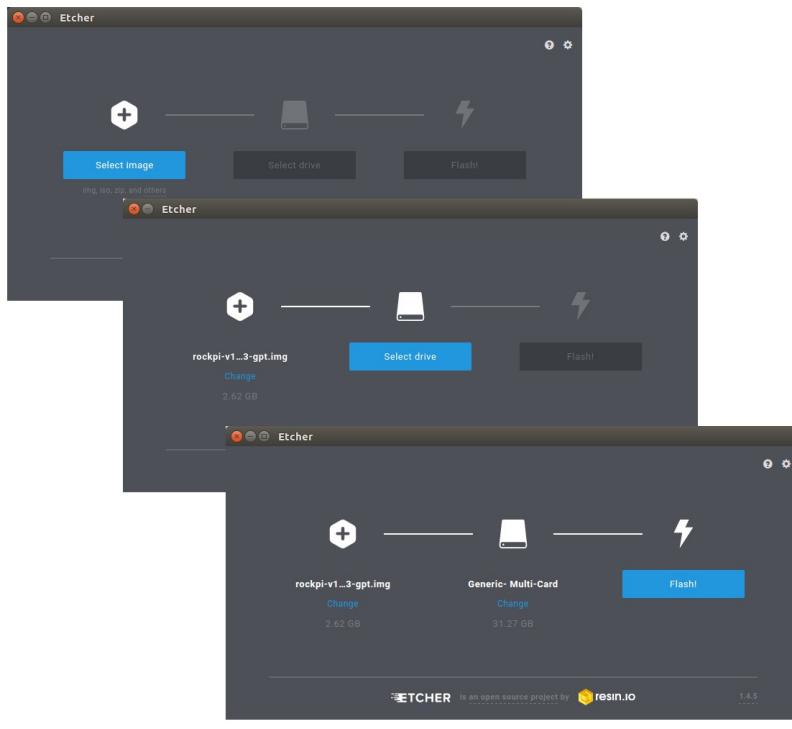
▶ [Linux -64 bit](#)

▶ [Linux-32 bit](#)

▶ [Windows](#)

Update Image

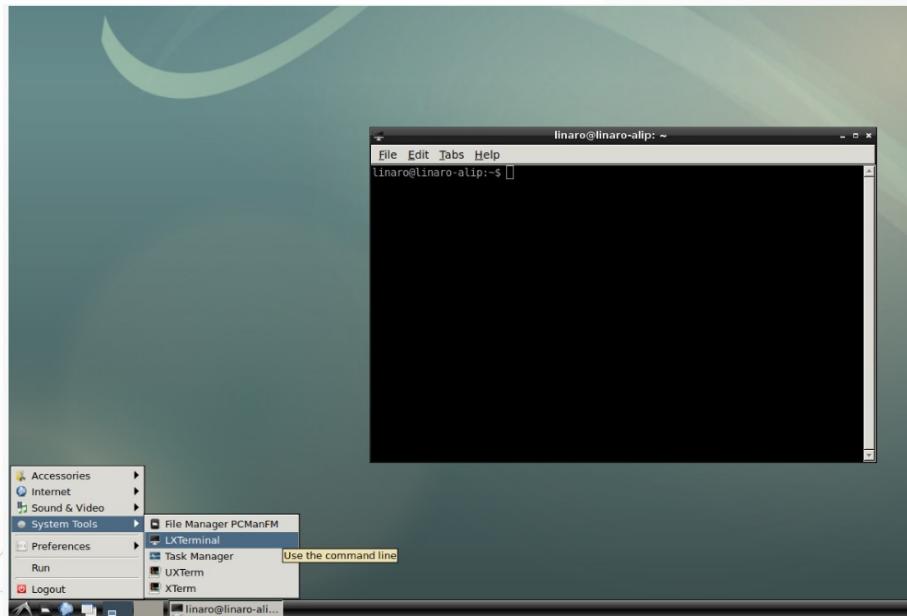
[CMD] ./etcher-etcher-electron-1.4.5



Debian

➤ Rock Pi Debian

➤ <https://wiki.radxa.com/Rockpi4/Debian>





RockPi Develop Data

» <https://wiki.radxa.com/Rockpi4>

 **Setup/Quick start**

- Getting started with your ROCK Pi 4, including what you need and how to get it booted.
- GPIO pinout
- Backup and Restore your SD card or eMMC module
- How to mount SSD with M2 extension board

 **Installation**

Installing an operating system on your ROCK PI 4, including microSD card, eMMC module, USB drive and M.2 NVME SSD,

- Install Rockchip Flashing tools
- Install image to eMMC from USB OTG Port
- Install on microSD card
- Install on eMMC module
- Install on SPI Flash
- Install on USB drive(wip)
- Install on M.2 NVME SSD

> More... [\[Expand\]](#)

 **Development**

Information about Linux and Android development, this is mostly for developers.

- USB Installation - How to use PC tools to install image on ROCK Pi 4.
- Serial Console - Serial console on GPIO header
- Build Debian - Build and generate Debian image
- Build vendor kernel(Rockchip 4.4) - Build vendor kernel for ROCK Pi 4
- Build Android (nougat) TV - Build Android for ROCK Pi 4
- Build Yocto - Build Yocto for ROCK Pi 4

> More... [\[Expand\]](#)

 **Hardware**

Technical specifications about the ROCK Pi 4 hardware, including WI-FI, display, camera, etc.

- Blog post from Radxa Team introducing the ROCK Pi hardware design
- ROCK Pi 4 - Introduction of the ROCK Pi 4 hardware
- Display
- Camera module
- Device Tree Overlays - Use other HAT

> More... [\[Expand\]](#)

 **Working With Linux**

Fundamental Linux usage for beginners and more advanced information for power users.

- Debian Desktop
- Ubuntu Server
- Linux system runs on M.2 NVME SSD
- Radxa APT
- Docker
- Samba

> More... [\[Expand\]](#)

 **Working With Android**

Fundamental Android usage for beginners and more advanced information for power users.

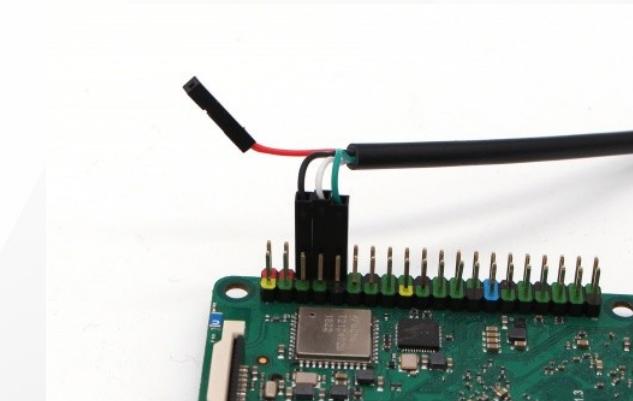
- Android7 Tablet(Support Raspberry Pi official 7" Display)
- Android7 TV
- Android9 Tablet
- Android9 TV
- Android9 Run on M.2 NVME SSD
- Android9 Mraa API
- Android10 Tablet
- Android11

Solve Google Play Device is not Play Protect certified issue [\[issue\]](#)

Debug Port

➤ Debug Port

ROCK Pi 4	USB to TTL cable
RX(pin 10)	TX
TX(pin 8)	RX
GND(pin 6)	GND



<https://wiki.radxa.com/Rockpi4/dev/serial-console>



Serial Console

➤ Serial setting on host PC

```
baud-rate: 1500000
data bit : 8
stop bit : 1
parity   : none
flow control: none
```

<https://wiki.radxa.com/Rockpi4/dev/serial-console>



Update

- » Linux Kernel
- » Linux DeviceTree
- » Linux Kernel Modules

<https://wiki.radxa.com/Rockpi4/dev/kernel-4.4>



Update Kernel/DTB - 1

» Build Linux kernel image

- » 1. Enter linux kernel source code
[CMD] cd rockpi-kernel
- » 2. Create a hide file
[CMD] touch .scmversion
- » 3. Build kernel Image
 - [CMD]** make rockchip_linux_defconfig
 - [CMD]** make -j8



Update Kernel/DTB - 2

▶ Package DEB file

- ▶ Enter linux kernel source code

[CMD] make bindeb-pkg -j8

▶ Transfer below to target board

- ▶ linux-firmware-image-4.4.154_4.4.154-4_all.deb

- ▶ linux-image-4.4.154_4.4.154-4_all.deb

- ▶ linux-libc-dev_4.4.154-4_all.deb

[CMD] scp \${FILE} rock@TARGET_IP:/home/rock/



Update Kernel/DTB - 3

➤ Below operator in Target Board

[CMD] sudo su

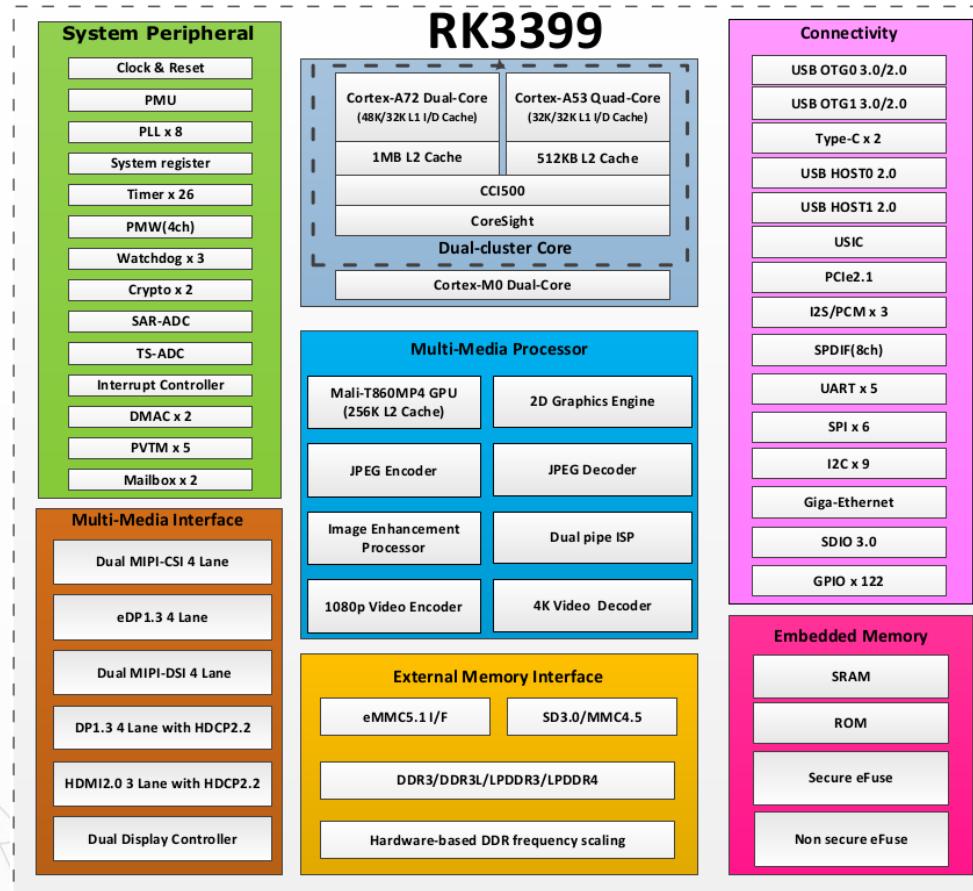
[CMD] dpkg -i linux-firmware-image-4.4.154_4.4.154-4_all.deb

[CMD] dpkg -i linux-image-4.4.154_4.4.154-4_all.deb

[CMD] dpkg -i linux-libc-dev_4.4.154-4_all.deb

Embedded Linux Arch

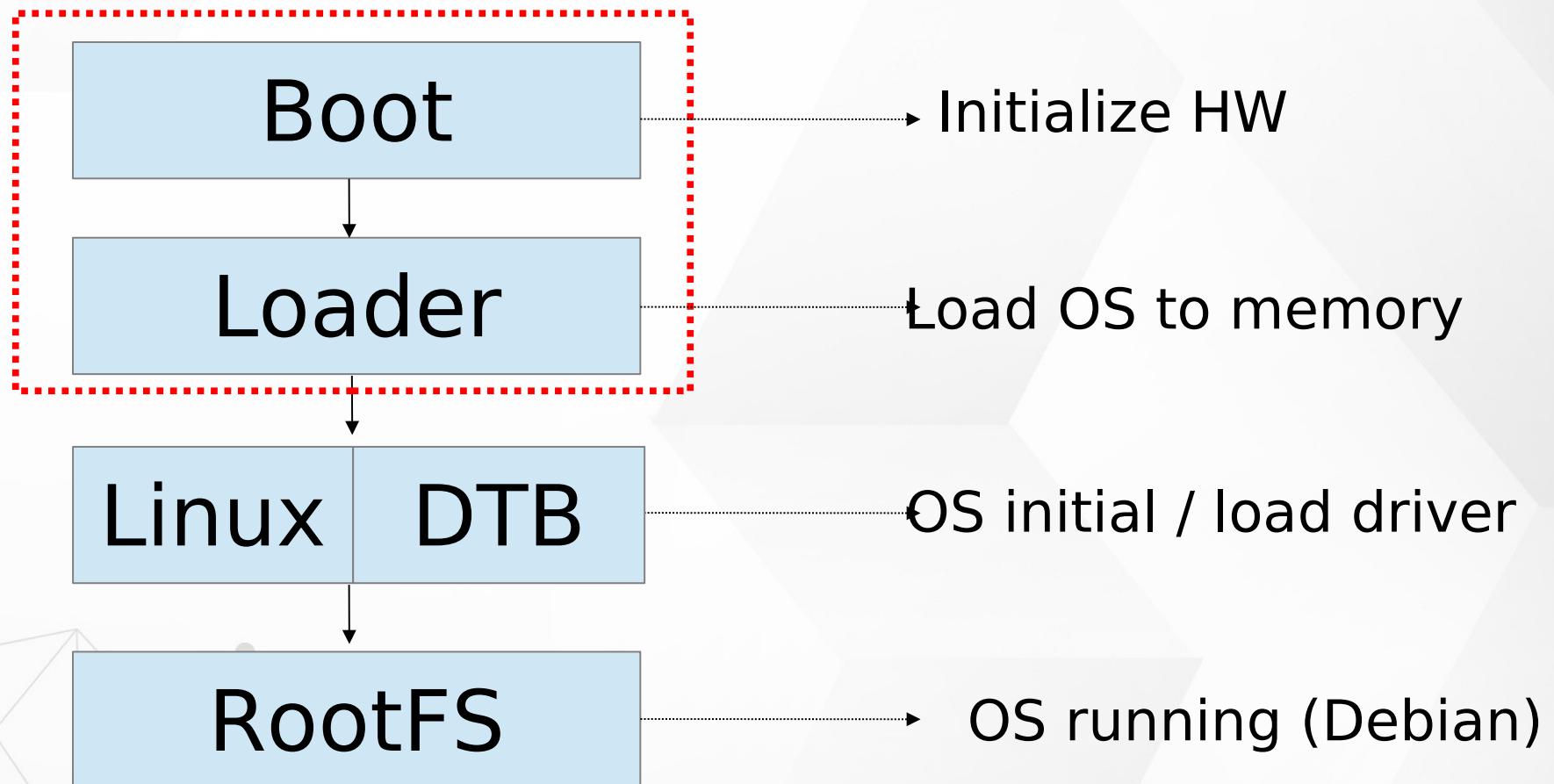
SOC RK3399



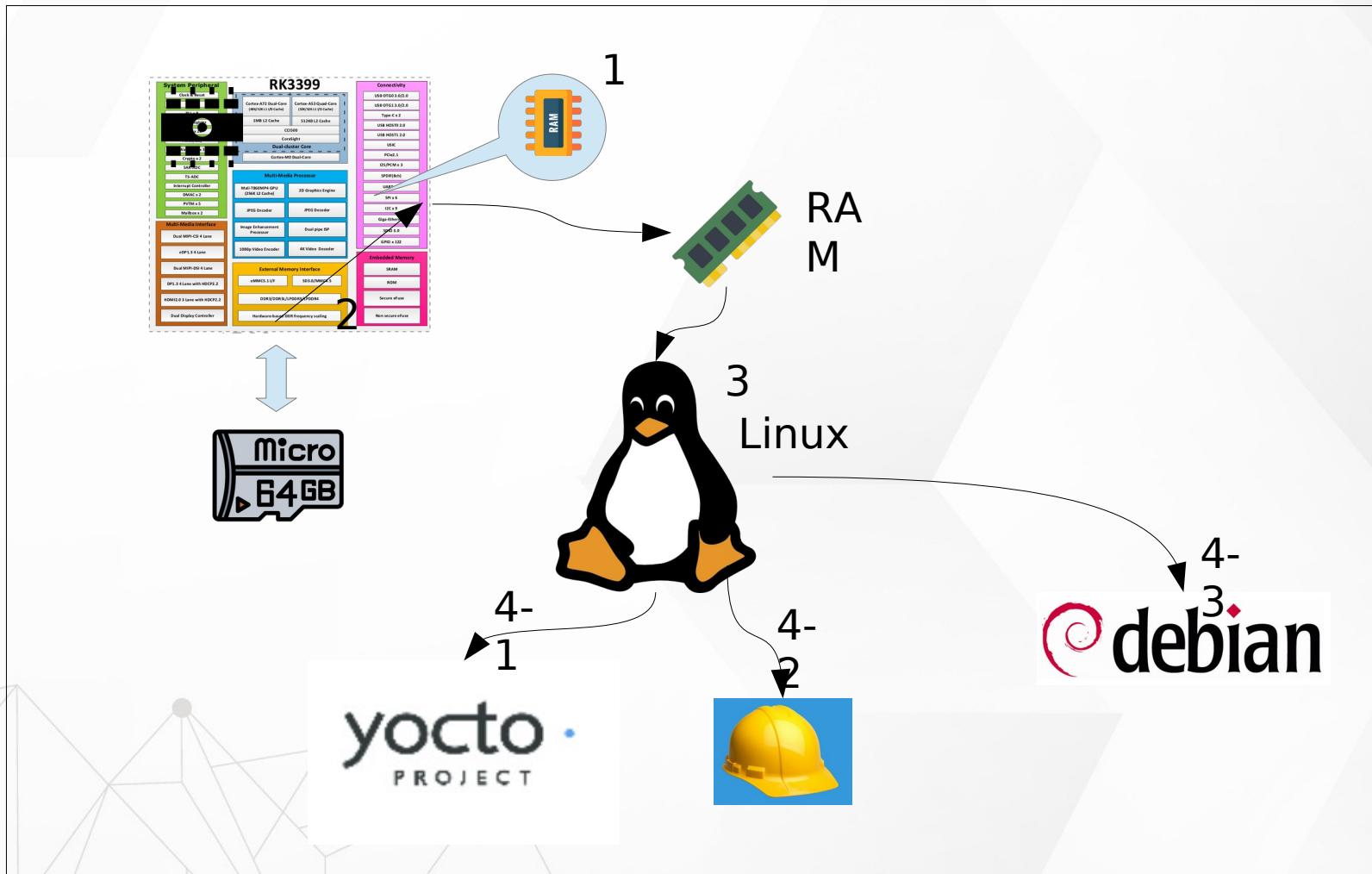
http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension



Embedded Linux System Booting



System Start Up





Boot

▶ Power On BootROM code (work in cache)

- Load BL1

▶ BL1 (work in cache - IDB_Loader)

- Initial simple exception vectors, PLL (clock)
- Initial Multi-CPU
- Load BL2

▶ BL2 (work in cache)

- Initial DDR memory
- Initial C environment (stack, heap,)
- Load BL31



Boot

▶ BL31 (work in DDR)

- Initial exception vectors
- Load BL32 (u-boot)

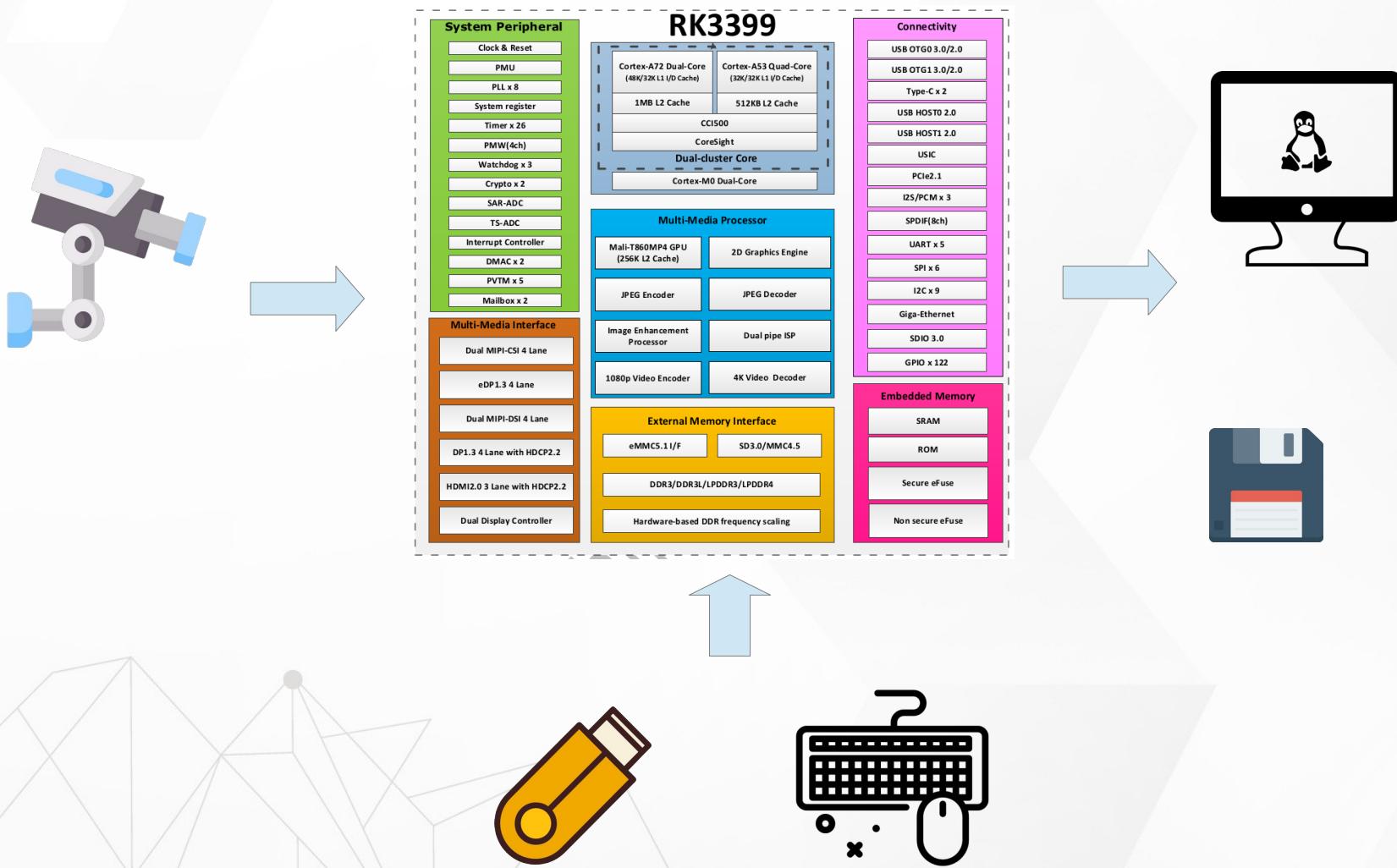
▶ BL32 U-boot

- Initial storage device
- Load Linux Kernel

▶ Kernel

- kernel/Documentation/arm64/booting.txt
- Load RootFS

Embedded Linux System





Embedded Linux System

User land

Video Play

Read Image

Reocrd Image

Tool

C Library

Qt

OpenCV

GLib

Virtual File System (VFS)

Kernel

Linux System Call

Linux Device Driver

Hardware

Disk

Image Sensor

Keyboard

mouse

Panel

CH3 Basic Software and Tool



Software and Tool

▶ Open Source License

▶ Develop Tool

- Geany, gedit, vim
- Git
- diff, patch

▶ Build Code Tool

- ARM toolchain
- make
- automake, autoconfig



Software and Tool

► Network

- WiFi, Ethernet, Net tool
- Bluetooth
- SSH, SSHFS
- NFS

► Media Software

- gstreamer
- ALSA Tool - aplay, arecord



Software and Tool

» Bus

- I2C – i2cset, i2cget, i2cdump
- USB – lsusb



Open Source License

► GNU General Public License

- 只要在一個軟件中使用 (" 使用 " 指類庫引用，修改後的代碼或者衍生代碼)
GPL 協議的產品，則該軟件產品必須也採用 GPL 協議，既必須也是開源和免費。
這就是所謂的 " 傳染性 "

► BSD License

- 基本上使用者可以 " 為所欲為 "，可以自由的使用，修改源代碼，
也可以將修改後的代碼作為開源或者專有軟件再發佈。

► LGPL

- LGPL 是 GPL 的一個為主要為類庫使用設計的開源協議。LGPL 允許商業軟件
通過類庫引用 (link) 方式使用 LGPL 類庫而不需要開源商業軟件的代碼。這使得採用
LGPL 協議的開源代碼可以被商業軟件作為類庫引用並發布和銷售。

Develop Tool



Ubuntu Package Management

- ▶ apt-get : command-line tool for handling packages
- ▶ apt-get --help
 - apt-get update
 - apt-get install \${PACKAGE_NAME}
 - apt-get remove \${PACKAGE_NAME}
 - apt-get autoremove
 - apt-get clean



Geany

► You can find a good edit for programing

→ Geany

<https://www.geany.org/>

\$ sudo apt-get install geany

→ Vim

\$ sudo apt-get install vim

→ gedit



Tracking code command

► Linux command

- Filter :
→ grep -r -n “function name”
- Fine special file include “String”
→ find -name “*.c” | xargs grep -n “String”



diff and patch

➤ diff - compare files line by line

- Create a patch file
 - `diff -Nuar file_a file_b > c.patch`
 - `-N`, treat absent files as empty
 - `-a, --text`
 - `-u`, output NUM (default 3) lines of unified context
 - `-r`, recursively compare any subdirectories found

➤ patch - apply a diff file to an original

- apply a patch file
 - `patch ./hello_1.c < ./tmp.patch`
- Reverse a patch file
 - `patch -R ./hello_1.c < tmp.patch`



Git

- ▶ <https://git-scm.com/book/zh-tw/v1/>
- ▶ 版本控制
- ▶ 程式回溯
- ▶ 管理多人共同開發



GitHub

► <https://github.com/>

The screenshot shows the GitHub sign-up interface. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for Explore, Features, Enterprise, Pricing, Sign up (in a green button), and Sign in. Below the navigation, a large white text area on a dark background features the headline "Where software is built". Below this, a paragraph explains GitHub's purpose: "Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free." It also mentions "Private plans start at \$7/mo.". To the right, there are three input fields: "Pick a username", "Your email", and "Create a password". A note below the password field specifies: "Use at least one lowercase letter, one numeral, and seven characters." A large green "Sign up for GitHub" button is centered below the password field. At the bottom, a small note states: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally."



Exercise

- ▶ 0. Create an empty Git repository (In local)
\$ git init
- ▶ 1. Clone code to local
\$ git clone https://github.com/xlloss/tiny4412-uboot.git
- ▶ 2. modify something
\$ gedit README
- ▶ 3. check source status
\$ git status
- ▶ 4. use "git add <file>..." to update what will be committed
\$ git add ./README
- ▶ 5. check status again
\$ git status



Exercise

- ▶ 6. commit code to local repository
 \$ git commit -a "test"
 Or \$ git commit
- ▶ 7. check log
 \$ git log
- ▶ 8. check how many branch in local repository
 \$ git branch
- ▶ 9. create new branch in local repository
 \$ git branch "new_branch_name"
 \$ git branch cadtc_uboot



Exercise

- ▶ 10. check out to new branch
 \$ git checkout "branch_name"
 \$ git checkout slash_uboot
- ▶ 11. check branch again
 \$ git branch
- ▶ 12 . push log branch to remote
 \$ git push origin slash-uboot
- ▶ 13. check remote branch status
 \$ git branch origin/ and push tab x2



Exercise

- ▶ reset your code, but modify code still live
 \$ git reset commit hash coed
- ▶ Hard reset your code, all modify code will discard
 \$ git reset - -hard hash coed
- ▶ Check log
 \$ git log
 \$ git show
- ▶ Download objects and refs from another repository
 \$ git fetch [--all]



BASIC Git Command

- » **init** Create an empty Git repository
- » **add** Add file contents to the index
- » **branch** List, create, or delete branches
- » **checkout** Checkout a branch or paths to the working tree
- » **clone** Clone a repository into a new directory
- » **commit** Record changes to the repository



BASIC Git Command

- » **diff** Show changes between commits, commit and working tree, etc
- » **rm** Remove files from the working tree and from the index
- » **pull** Fetch from and merge with another repository or a local branch
- » **push** Update remote refs along with associated objects
- » **reset** Reset current HEAD to the specified state
- » **cherry-pick** apply changes introduced by some existing commits

Build Code Tool

Makefile



Makefile

- Simplify compile command
- Automation compile, linker program source
- It can update source in accordance with the dependence



Compile a Hello_World

A

B

C

```
aarch64-linux-gnu-gcc -o helloworld ./hello.c
```

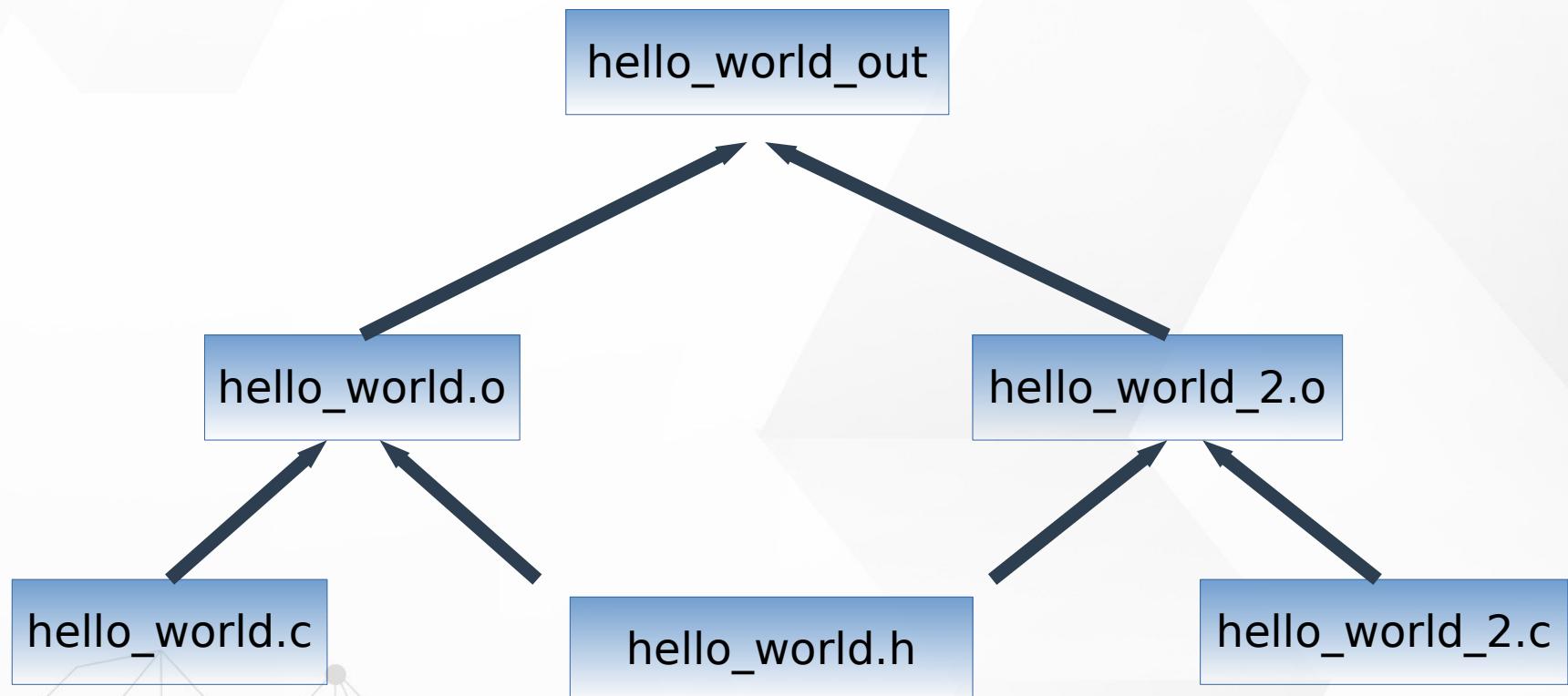
A : ARM C Compile

B : ARM C Compile Parameter
(Output name)

C : C source code



Another Sample





Compile Another Sample

- ▶ Step 1 : gcc -c hello_world_2.c
- ▶ Step 2 : gcc -c hello_world_2.c
- ▶ Step 3 : gcc -o hello_world hello_world.o hello_world_2.o



Another Sample - Makefile

```
CC=$(CROSS_COMPILE)gcc

all: hello_world

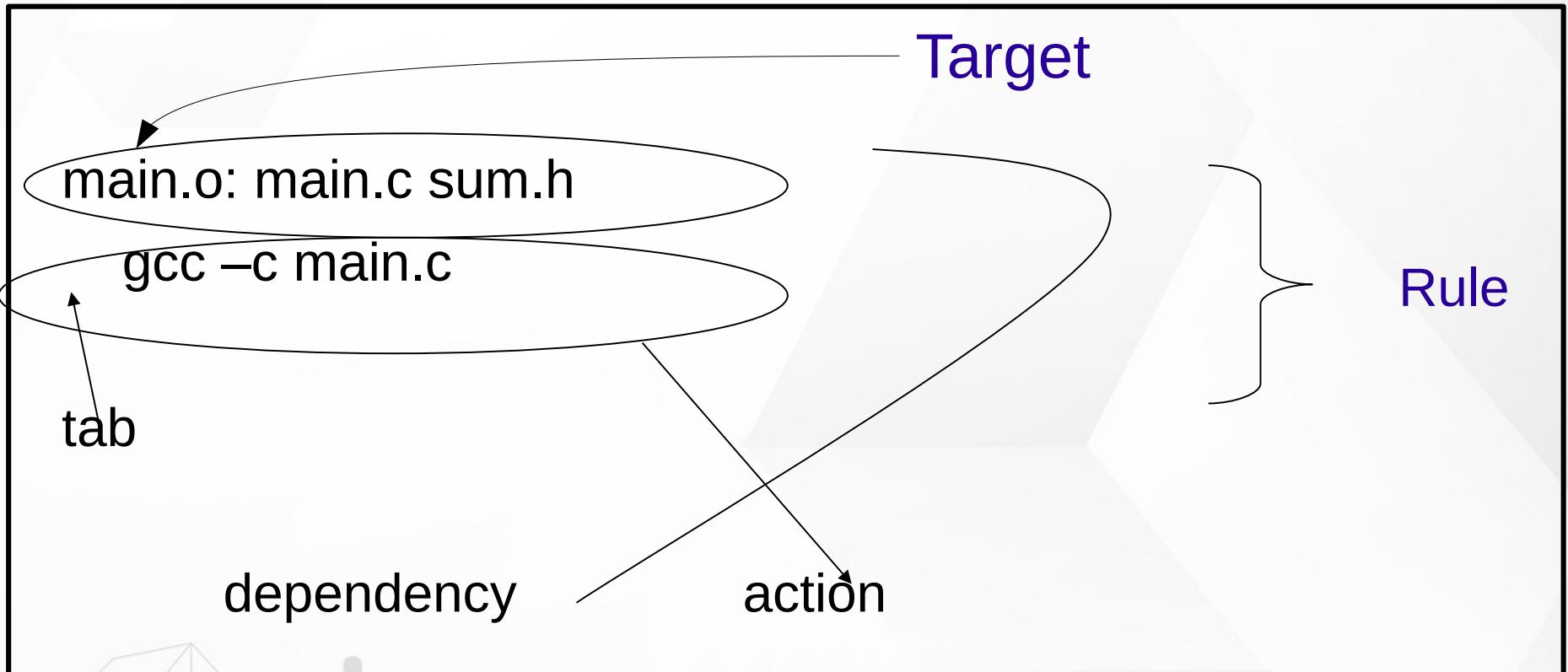
hello_world: hello_world.o hello_world_2.o
    $(CC) -o hello_world hello_world.o hello_world_2.o

hello_world.o: hello_world.c
    $(CC) -c hello_world.c

hello_world_2.o: hello_world_2.c
    $(CC) -c hello_world_2.c

clean:
    rm -r *.o
    rm hello_world
```

Rule Syntax



make 在編譯時，若發現 **target** 比較新，
也就是 **dependencies** 都比 **target** 舊，
那麼將不會重新建立 **target**，如此可以避免不必要的編譯動作

Rule Syntax

```
CC=$(CROSS_COMPILE)gcc  
  
CFLAGS=-WI,-Map,out.map -lpthread -lm  
  
all: hello_world  
  
hello_world: hello_world.o hello_world_2.o  
    $(CC) -o hello_world hello_world.o hello_world_2.o  
  
hello_world.o: hello_world.c  
    $(CC) $(CFLAGS) -c hello_world.c  
  
hello_world_2.o: hello_world_2.c  
    $(CC) $(CFLAGS) -c hello_world_2.c  
  
clean:  
    rm -r *.o  
    rm hello_world
```



Rule Syntax Sample 1

```
hello_world_2.o: hello_world_2.c  
$(CC) $(CFLAGS) -c hello_world_2.c
```

hello_world_2.o depend hello_world_2.c

If hello_world_2.c alive and **be update**,

it will do command \$(CC) \$(CFLAGS) -c hello_world_2.c),

then output hello_world_2.o object file



Rule Syntax Sample 2

```
hello_world_2.o: hello_world_2.c  
$(CC) $(CFLAGS) -c hello_world_2.c
```

hello_world.o depend hello_world.c

If hello_world.c alive and **be update**,
it will do command `$(CC) $(CFLAGS) -c hello_world.c`,
then output hello_world.o object file.



Rule Syntax Sample 2

```
hello_world_2.o: hello_world_2.c  
$(CC) $(CFLAGS) -c hello_world_2.c
```

hello_world depend hello_world.o and hello_world_2.o

If hello_world.o and hello_world_2.o alive and **be update**,

it will do command `$(CC) -o hello_world hello_world.o hello_world_2.o`,

then create hello_world execute file



hello_world_ex1

```
CC=$(CROSS_COMPILE)gcc

AA='1234' '5678'
AA := 'DDDD'

$(info AA=$(AA))

CFLAGS=-Wl,-Map,out.map -lpthread -lm

all: hello_world

hello_world: hello_world.o
→ $(CC) -o hello_world hello_world.o

hello_world.o: hello_world.c
→ $(CC) $(CFLAGS) -c hello_world.c

clean:
→ rm -r *.o
→ rm hello_world
```

□ : Tab



Assignment Operators

- = 定義一個 需做遞迴展開的 變數型態
- := 定義一個 立即運作的 變數型態
- += 將 指定值， 繼加在 原變數中
- ?= 如果之前 無任何設定該變數， 即現在設定，
否則 跳過設定（就是不做任何事）



Assignment Operators Sample 1

```
AA ='1234' '5678'  
BB = ${AA}  
AA = '789'  
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'  
BB='789' 'ABCDE'
```



Assignment Operators Sample 2

```
AA ='1234' '5678'  
BB := ${AA}  
AA = '789'  
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'  
BB='1234' '5678'
```



Assignment Operators Sample 3

```
AA ='1234' '5678'  
BB := ${AA}  
AA = '789'  
AA ?= 'ABCDE'
```

Output

```
AA='789'  
BB='1234' '5678'
```



The Automatic Variables

- ▶ \$@ The target filename.
- ▶ \$< The first prerequisite.
- ▶ \$^ The list of prerequisites, excluding duplicate elements.



The Automatic Variables

```
CC = gcc
CFLAGS = -Wall -g -std=c99
LDFLAGS = -lm

circle : circle.o circulararea.o
        $(CC) $(LDFLAGS) -o circle circle.o
circulararea.o

circle.o : circle.c
        $(CC) $(CFLAGS) -o circle.o -c circle.c

circulararea.o: circulararea.c
        $(CC) $(CFLAGS) -o circulararea.o -c
circulararea.c
```



The Automatic Variables

```
CC = gcc
CFLAGS = -Wall -g -std=c99
LDFLAGS = -lm

circle : circle.o circulararea.o
        $(CC) $(LDFLAGS) -o $@ $^

circle.o : circle.c
          $(CC) $(CFLAGS) -o $@ -c $<

circulararea.o: circulararea.c
                 $(CC) $(CFLAGS) -o $@ -c $<
```



Phony Targets

➤ .PHONY

- Any targets that are prerequisites of .PHONY are always treated as out of date.

```
#Naming our phony targets
.PHONY: clean install

#Removing the executable and the object files
clean:
    rm sample main.o example.o
    echo clean: make complete

#Installing the final product
install:
    cp sample /usr/local
    echo install: make complete
```



Command-Line Options

➤ -C dir, --directory= dir

- make changes the current working directory to dir before it does anything else. If the command line includes multiple -C options, each directory specified builds on the previous one

➤ -j [number] , --jobs[= number]

- Run multiple commands in parallel

Media Tool

Gstreamer



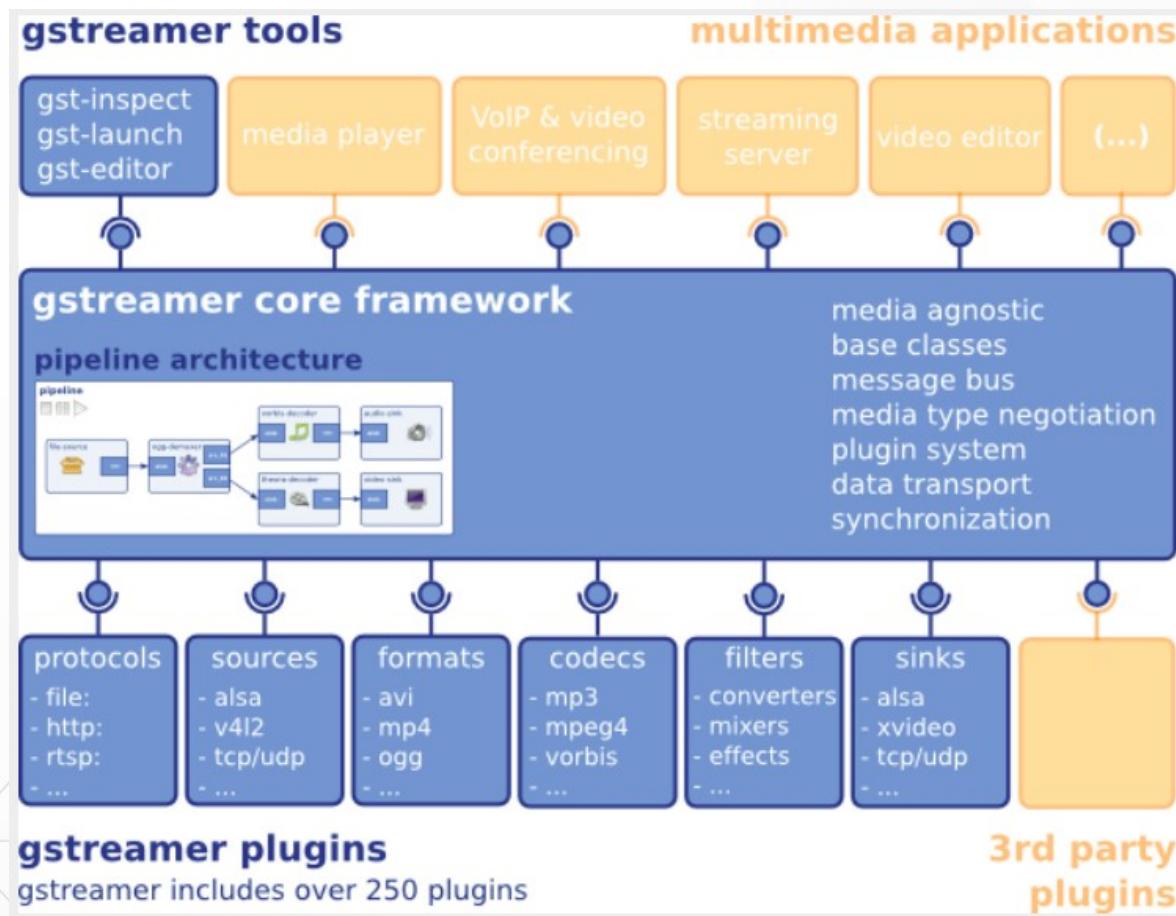


Open Source Multimedia Framework

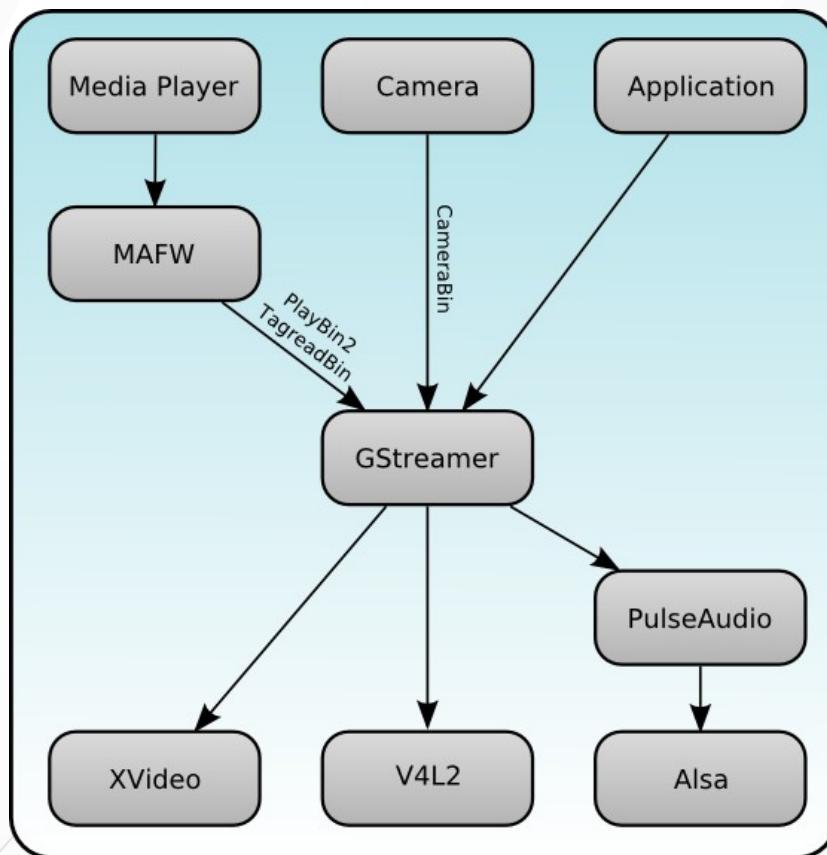
- ▶ <https://gstreamer.freedesktop.org/>
- ▶ Documentation
 - ▶ [Tutorials](#)



Block Diagram



Block Diagram



http://maemo.org/development/sdks/maemo_5_beta_docs/using_multimedia_components/

Overview

- GStreamer is a **framework** for creating streaming media applications
- The framework is based on **plugins** that will provide the various codec and other functionality

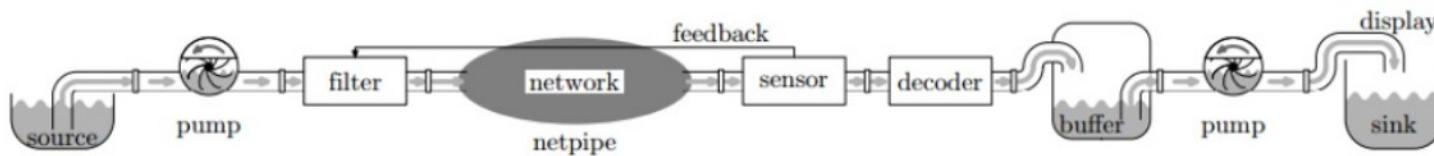
Overview



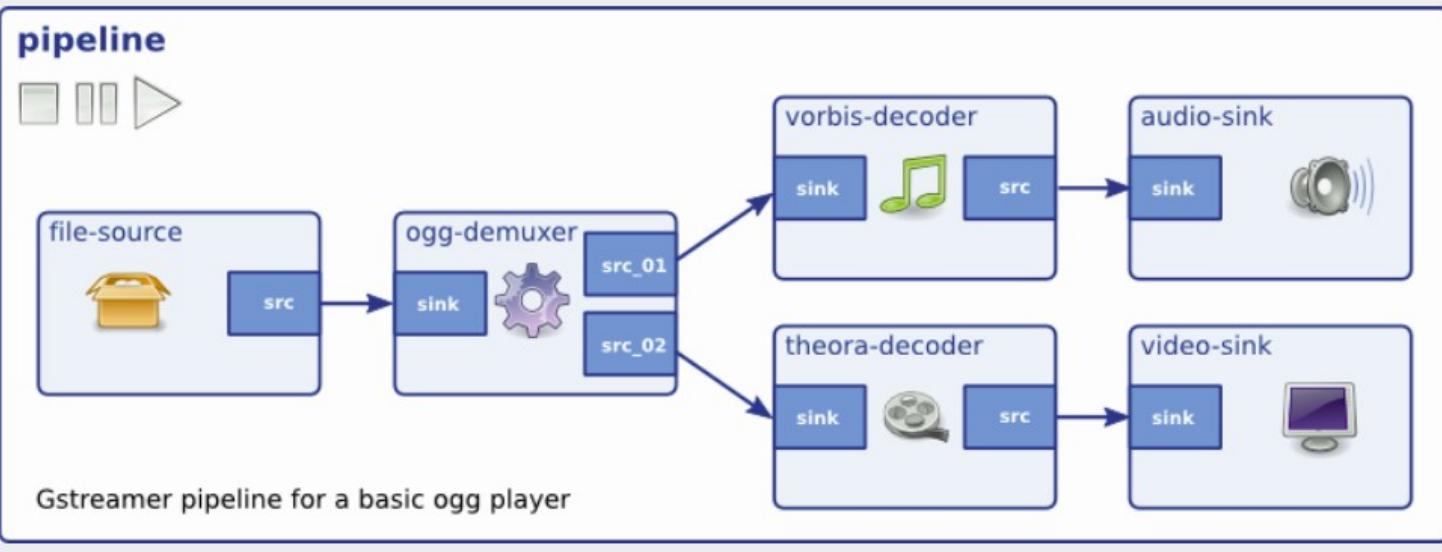
- ▶ **Gst-plugins-base:**
an essential exemplary set of elements
- ▶ **Gst-plugins-good:**
a set of good-quality plug-ins under LGPL
- ▶ **Gst-plugins-ugly:**
a set of good-quality plug-ins that might pose distribution problems
- ▶ **Gst-plugins-bad:**
a set of plug-ins that need more quality

Gstreamer Pipe

A streamer pipe



Gstreamer Pipe





Gstreamer - Tool

▶ Gst-inspect-1.0

▶ Print supported plug-in

▶ Gst-launch-1.0

▶ Gstreamer tester

▶ Gst-typefind-1.0

▶ Check file for gstreamer plug-in type

Gststreamer - Tool



» Gst-inspect-1.0

» Check what kind of videosink in Rockpi4b

» `gst-inspect-1.0 | grep -i videosink`

```
rock@rockpi4b:~$ gst-inspect-1.0 | grep -i videosink
debugutilsbad: fakevideosink: Fake Video Sink
debugutilsbad: fpsdisplaysink: Measure and show framerate on videosink
inter: intervideosink: Internal video sink
decklink: decklinkvideosink: Decklink Video Sink
autodetect: autovideosink: Auto video sink
rock@rockpi4b:~$
```

Gststreamer - Tool



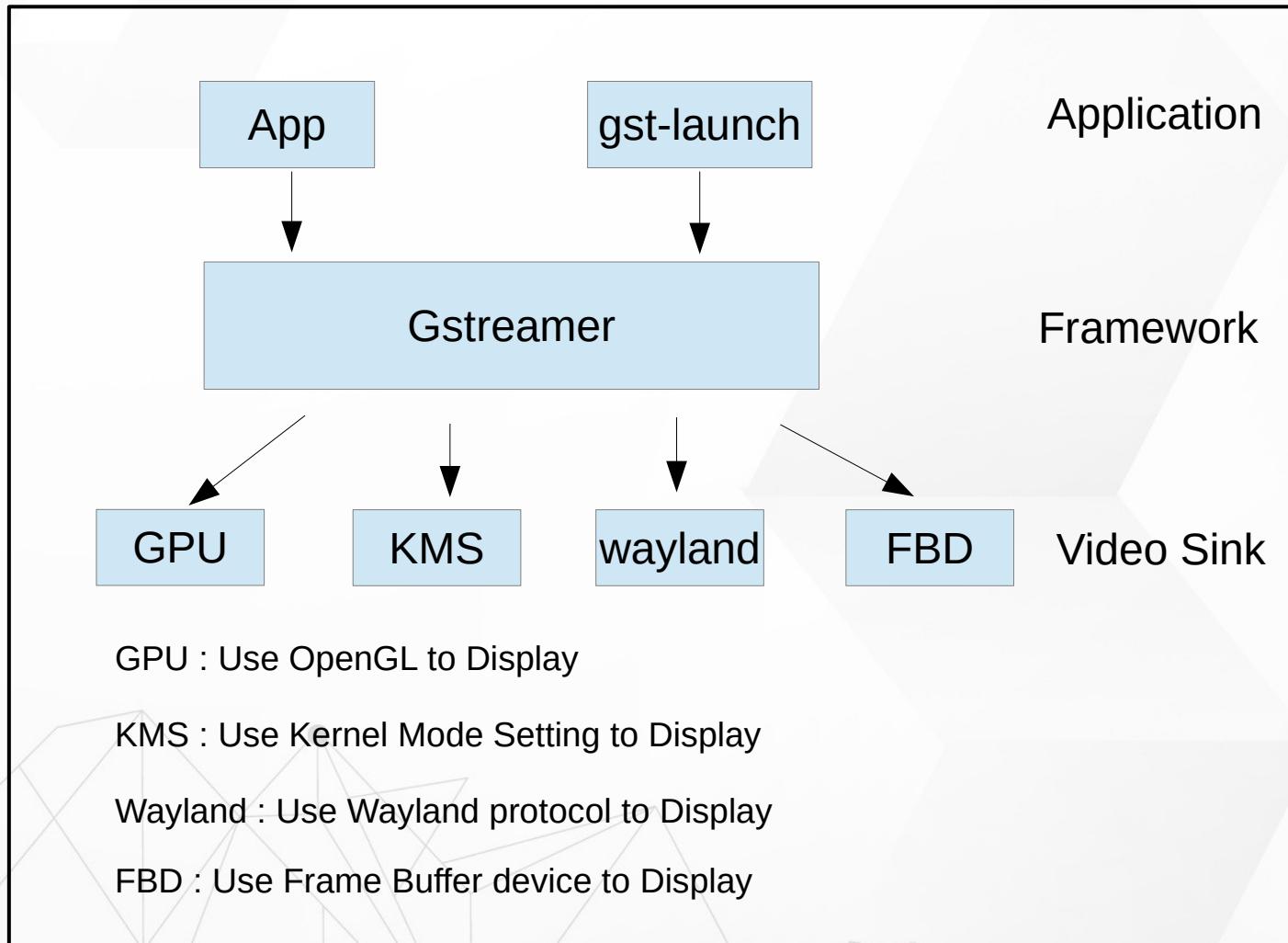
➤ Gst-typefind-1.0

➤ Check Serenity-DVD-320x240.m4v what kind
of file type in gstreamer

➤ gst-typefind-1.0 ./Serenity-DVD-320x240.m4v

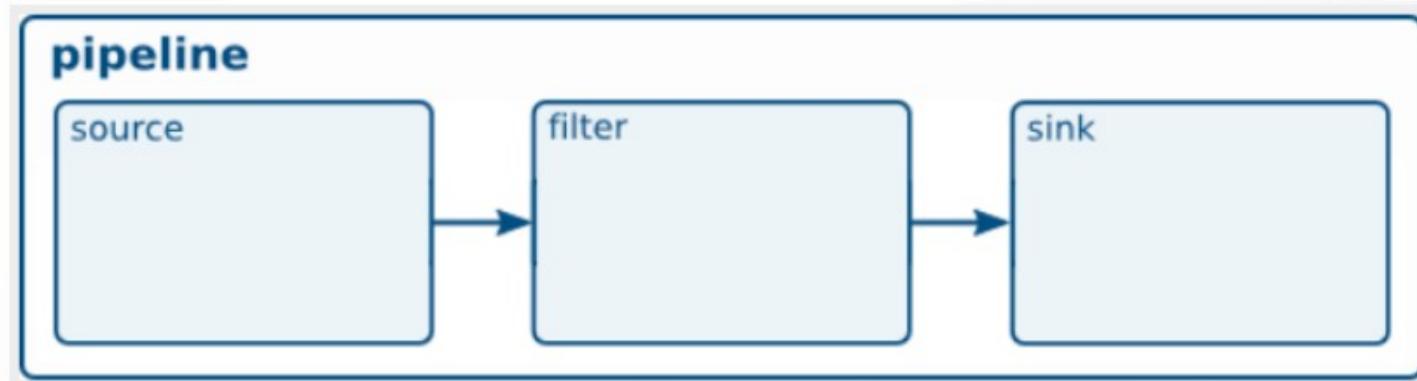
```
rock@rockpi4b:~$ gst-typefind-1.0 ./Serenity-DVD-320x240.m4v  
./Serenity-DVD-320x240.m4v - video/quicktime, variant=(string)iso
```

Gstreamer - Video





Play Video Test Stream

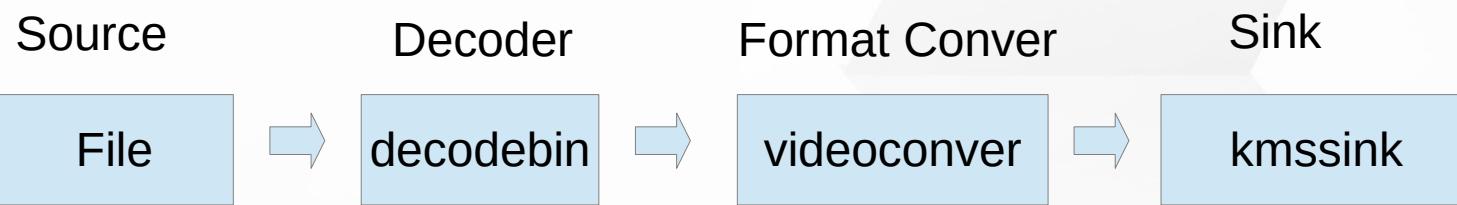


```
gst-launch-1.0 videotestsrc ! video/x-raw, width=1280, height=720 ! kmssink
```



Play a H.264 video

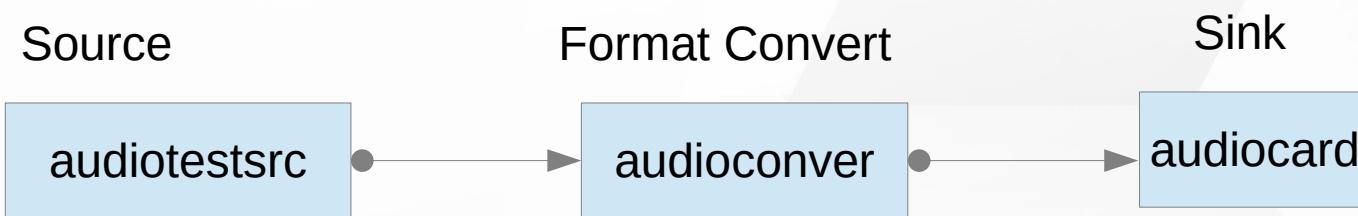
```
gst-launch-1.0 filesrc  
location=/oem/200frames_count.h264 !\n
```





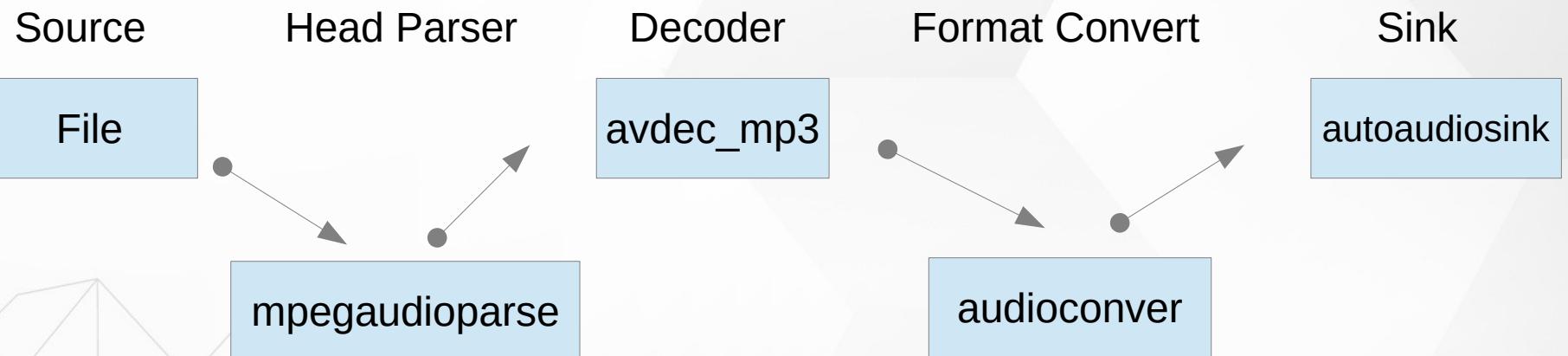
Play a Audio Test

```
gst-launch-1.0 audiotestsrc ! audioconvert ! alsasink device-name=rockchipes8316c
```



Play a MP3

```
gst-launch-1.0 filesrc location="/home/cadtc/audio/piano2-CoolEdit.mp3" ! \
mpegaudioparse ! \
mpg123audiodec ! \
audioconvert ! autoaudiosink
```





ALSA Tool

▶ ALSA Util

- **aplay**
 - Play a WAV file
- **arecord**
 - Record a sound
- **alsamixer**
 - A graph tool for adjusting audio gain
- **amixer**
 - A console tool for adjusting audio gain



ALSA Tool

▶ ALSA Utile

- **aplay**

- aplay -Dsysdefault:CARD=rockchipes8316c /usr/share/sounds/alsa/Front_Center.wav
- aplay -Dsysdefault:CARD=HDMICODEC /usr/share/sounds/alsa/Front_Center.wav

- **arecord**

- arecord -Dhw:0,0 -r 44100 -t wav -f CD -d 5 /tmp/test.wav

- **alsamixer**

- alasmixer

- **amixer**

- amixer scontrols | less
- amixer sget 'HP' 0%
- amixer sset 'HP' 0%

WiFi and Network



Basic Network Tool

- » ifconfig → Network setting check
- » ping → Network package check
- » iperf3 → perform network throughput tests
- » dhcpc → used for automatic retrieving of



WPA/WPA2

► iw → Finding the WiFi device name

- Scan SSID

► wpa_supplicant

- For connecting to a WPA/WPA2 network



WPA/WPA2 - Device

\$ iw dev

```
[root@rk3399:/]# iw dev  
phy#0  
      Interface wlan0  
            ifindex 3  
            wdev 0x1  
            addr cc:4b:73:92:50:6a  
            type managed  
            txpower 31.00 dBm
```

\$ ls /sys/class/net

```
[root@rk3399:/]# ls /sys/class/net/  
eth0  lo  wlan0  
[root@rk3399:/]# █
```



WPA/WPA2 - iw

\$ iw wlan0 scan

```
BSS 0c:9d:92:d9:e7:78 (on wlan0)
    TSF: 7656316992 usec (0d, 02:07:36)
    freq: 2462
    beacon interval: 100 TUs
    capability: ESS Privacy ShortPreamble ShortSlotTime RadioMeasure (0x1431)
    signal: -73.00 dBm
    last seen: 2 ms ago
    SSID: kevin asus
    Supported rates: 1.0* 2.0* 5.5* 11.0*
    DS Parameter set: channel 11
    ERP: Use_Protection
    Extended supported rates: 6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0
    RSN:
        * Version: 1
        * Group cipher: CCMP
        * Pairwise ciphers: CCMP
        * Authentication suites: PSK
        * Capabilities: 16-PTKSA-RC 1-GTKSA-RC (0x000c)
    HT capabilities:
        Capabilities: 0x12d
            RX LDPC
            HT20
            SM Power Save disabled
            RX HT20 SGI
            RX STBC 1-stream
            Max AMSDU length: 3839 bytes
            No DSSS/CCK HT40
            Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
            Minimum RX AMPDU time spacing: 2 usec (0x04)
            HT RX MCS rate indexes supported: 0-7
            HT TX MCS rate indexes are undefined
```



WPA/WPA2 – SSID and PASSWD

```
$ wpa_passphrase "SSID" > /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1

network={
    ssid="ssid"
    #psk="testtest"
    psk="password"
}
```



WPA/WPA2 - Connect

```
$ wpa_supplicant -B -D wext -i wlan0 -c /etc/wpa_supplicant.conf
```

```
[ 29.752634] CFG80211-ERROR) wl_escan_handler : escan is not ready ndev ffffffc0782d1000
[ 29.767372] wl_iw_set_essid: WLC_DISASSOC
[ 29.772806] Setting the D11auth 1
[ 29.788792] wl_iw_set_freq: chan=4
[ 29.794068] wl_iw_set_wap: WLC_REASSOC failed (-22).
[ 29.835315] Connecting with 62:07:b7:ed:02:4d channel (4) ssid "REASSO", len (6)
[ 29.835315]
[ 29.908754] wl_iw_event: Link UP with 62:07:b7:ed:02:4d
[ 29.914341] wl_bss_connect_done succeeded with 62:07:b7:ed:02:4d
[ 29.921748] wl_bss_connect_done succeeded with 62:07:b7:ed:02:4d
```



WPA/WPA2 - DHCP

```
$ udhcpc -i wlan0
```

```
[root@rk3399:/]# udhcpc -i wlan0
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 192.168.43.214
udhcpc: lease of 192.168.43.214 obtained, lease time 3599
deleting routers
adding dns 192.168.43.12
[root@rk3399:/]# █
```



WPA/WPA2 - IP

\$ ifconfig wlan0

```
[root@rk3399:/]# ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr CC:4B:73:92:50:6A
           inet  addr:192.168.43.214  Bcast:192.168.43.255  Mask:255.255.255.0
           inet6 addr: fe80::7e7:9ca:dc48:71ab/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:6 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:1312 (1.2 KiB)   TX bytes:4477 (4.3 KiB)

[root@rk3399:/]#
```



WPA/WPA2 - Ping

\$ Ping 8.8.8.8 (Google)

```
[root@rk3399:/]# ping -I wlan0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 192.168.43.214 wlan0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=49.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=37.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=35.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=63.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=29.5 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=40.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=117 time=45.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=117 time=53.0 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=117 time=35.6 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=117 time=33.8 ms
^C
--- 8.8.8.8 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10018ms
rtt min/avg/max/mdev = 29.590/42.629/63.804/9.521 ms
[root@rk3399:/]#
```

SSH



SSH

- ▶ Secure SHell protocol
- ▶ SSH Client
- ▶ SSH Server





SSH

▶ SSH Client

- `# sudo apt-get install ssh`
- <https://slashembeddedlinux.blogspot.com/p/tmp.html>

NFS



NFS

- ▶ Network File System
- ▶ NFS Client
- ▶ NFS Server



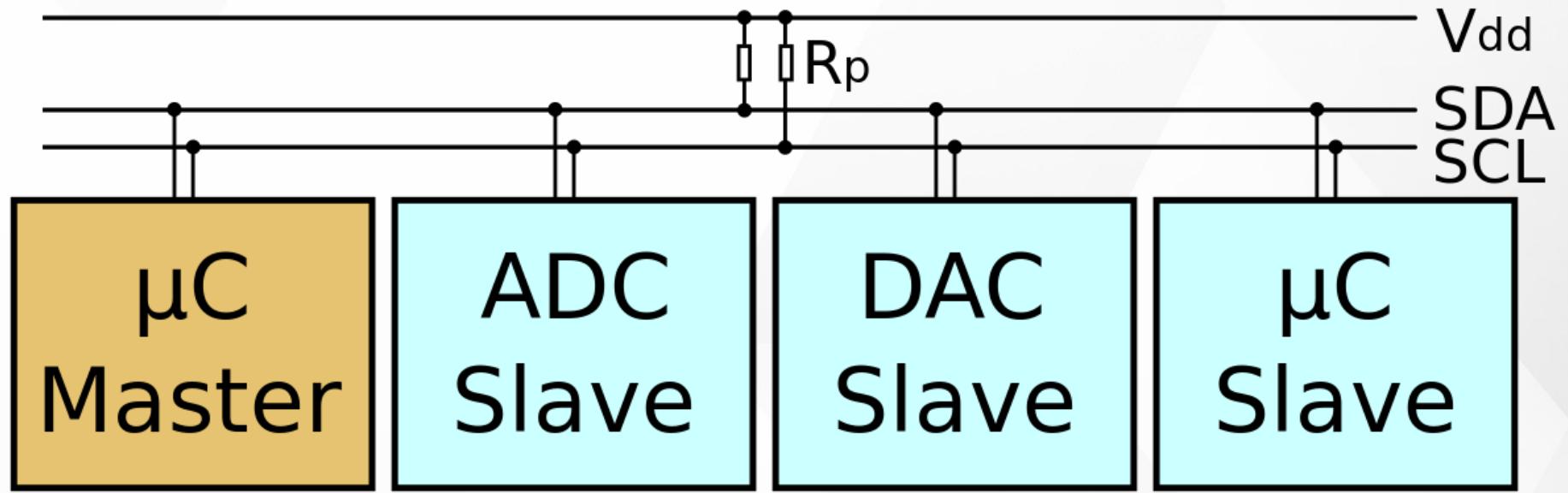


NFS

▶ <https://slashembeddedlinux.blogspot.com/p/tmp.html>

I2C Tool

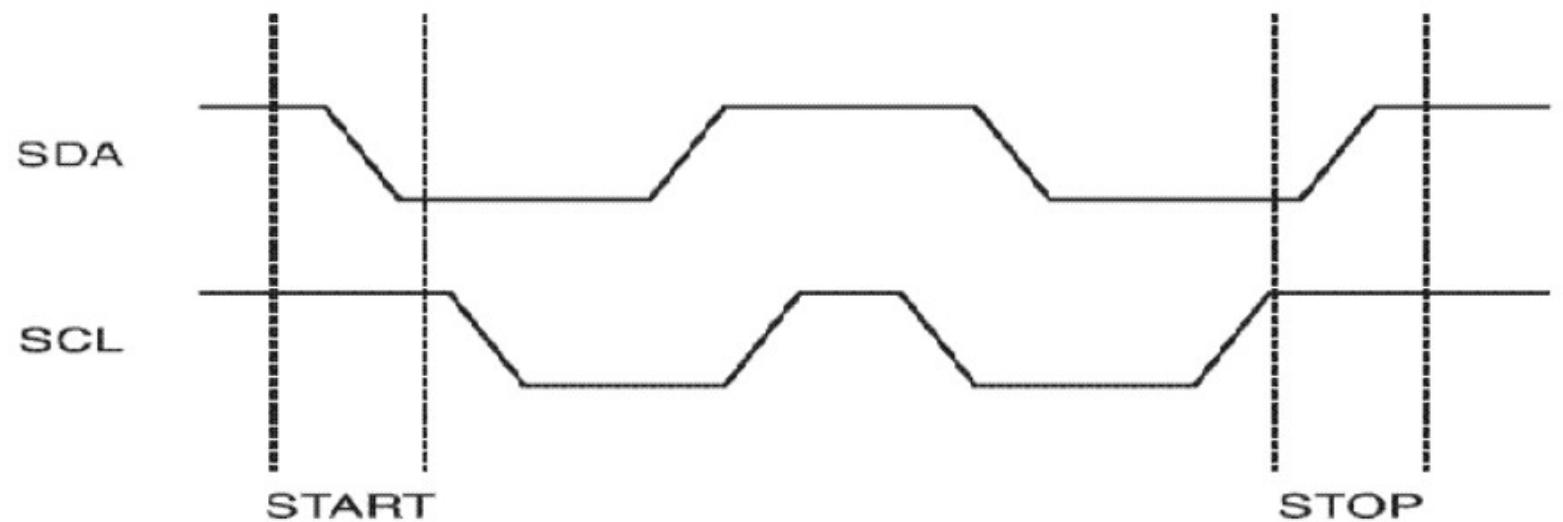
I2C Driver



I2C protocol

► Serial bus

- SDA data line
- SCL clock line



I2C protocol

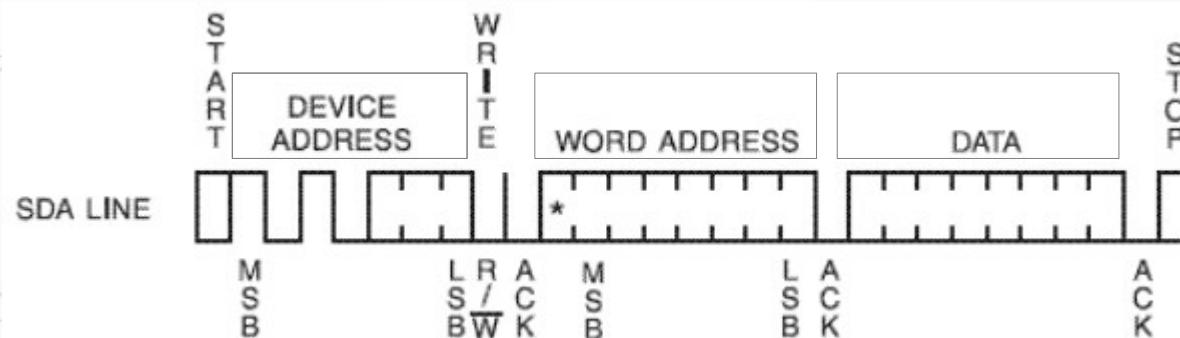
➤ Write

- byte write
- page write

➤ Device address

➤ Read/write bit : 0

➤ ACK



I2C protocol

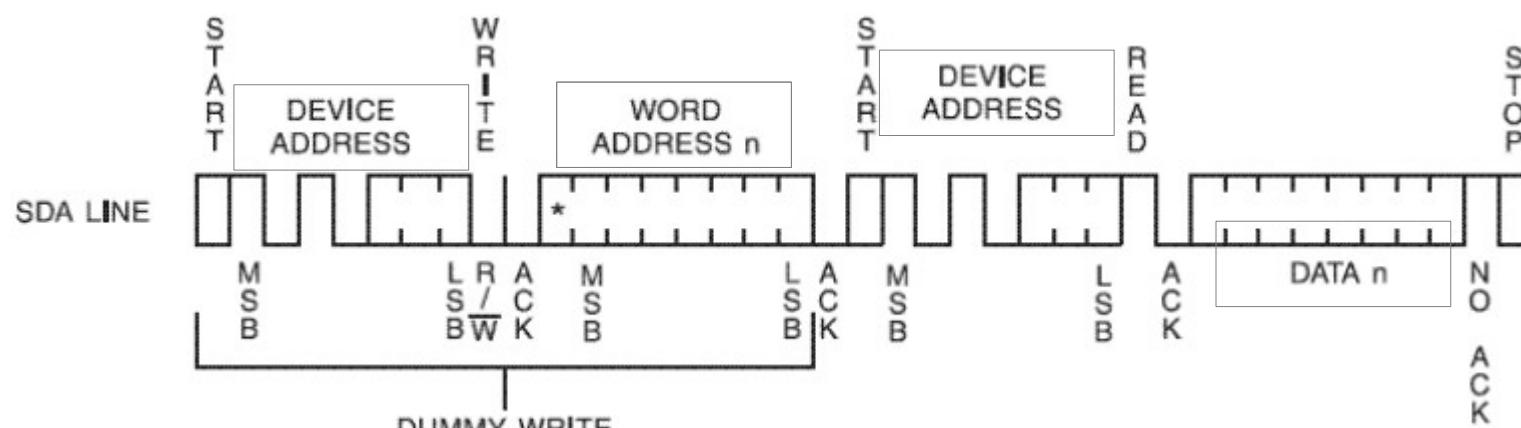
➤ Read

- byte read
- page read

➤ Device address

➤ Read/write bit : 1

➤ ACK





I2C Dev Interface

▶ i2c tool

- i2cset, i2cget, i2cdump
- i2cdetect -l

▶ /dev/i2c-x

- /dev/i2c-0, /dev/i2c-1, /dev/i2c-2 ...

▶ /sys/class/i2c-dev/

- i2c-0 i2c-1 i2c-2 i2c-3 i2c-7 i2c-8 ...



I2C Dev Interface

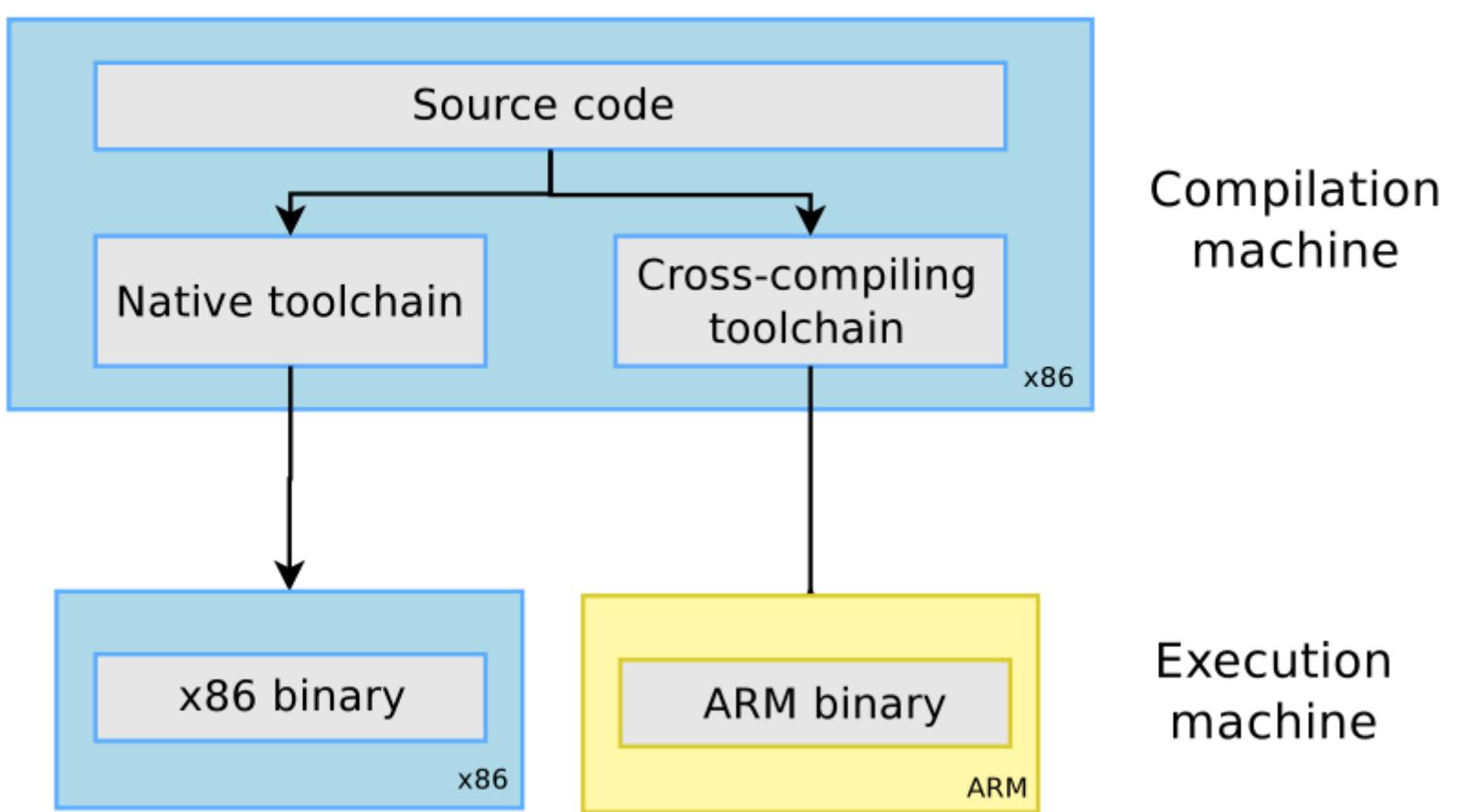
➤ Documentation/i2c/dev-interface

➤ i2c-tools

- i2cdump
- i2cdetect
- i2cget
- i2cset

CH4 Cross Compilation Toolchain

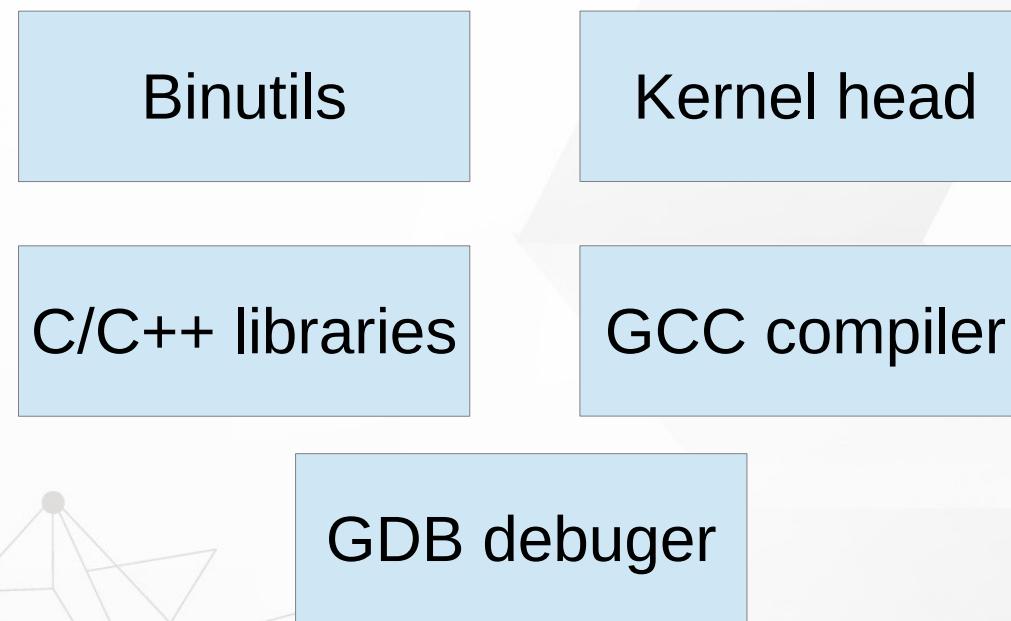
Cross Compilation Tool-chain





GCC Components

- » The GNU C Compiler
- » The GNU Compiler Collection





Binutils

► Binutils

- **as** : the assembler, that generates binary code from assembler source code
- **ld** : the linker
- **ar, ranlib** : to generate .a archives, used for libraries
- **objdump, readelf, size, nm, strings** : to inspect binaries
- **strip** : to strip useless parts of binaries in order to reduce their size



Kernel head

- The C library and compiled programs needs to interact with the kernel
- Compiling the C library requires kernel headers, and many applications also require them
- The kernel to user space ABI is backward compatible



GCC

- ▶ GCC originally stood for the "GNU C Compiler."
- ▶ GNU Compiler Collection
 - C, C++, Ada, Objective-C, Fortran, JAVA ...
- ▶ <http://gcc.gnu.org/>



GCC flag

▶ arm-linux-gnueabihf-gcc –help

- -c : Compile and assemble, but do not link
- -o <file> : Place the output into <file>
- -shared : Create a shared library
- -g : add debug information
- -O : sets the compiler's optimization level
- -Wall : enables all compiler's warning messages
- -D : defines a macro to be used by the preprocessor
- -I : adds include directory of header files
- -L,-l :
 - -L looks in directory for library files
 - -l links with a library file



C library

- ▶ The C library is an essential component of a Linux system
- ▶ Several C libraries are available:
 - **glibc, uClibc, eglIBC, dietlibc, newlib**
- ▶ The choice of the C library must be made at the time of the cross-compiling toolchain generation, as the GCC compiler is compiled against a specific C library.



sysroot

- ▶ The sysroot is the logical root directory for headers and libraries
- ▶ GCC look for head and LD look for library
- ▶ We can assign sysroot locate avoid toolchain change locate
 - --with-sysroot=<locate>



Floating point support

- ▶ For processors having a **floating point unit**, the toolchain should generate hard float code, in order to use the floating point instructions directly
- ▶ For processors without a floating point unit
 - Generate hard float code and rely on the kernel to emulate the floating point instructions
 - Generate soft float code, so that instead of generating floating point instructions, calls to a user space library are generated



Floating point support

<https://www.linaro.org/downloads/>

Latest Linux Targeted Binary Toolchain Releases	
arm-linux-gnueabihf	32-bit Armv7 Cortex-A, hard-float, little-endian
armv8l-linux-gnueabihf	32-bit Armv8 Cortex-A, hard-float, little-endian
aarch64-linux-gnu	64-bit Armv8 Cortex-A, little-endian



Obtain a Toolchain

➤ Building a cross-compiling toolchain by ourself

- Crosstool-NG
- <http://crosstool-ng.org/#introduction>

➤ Pre-build toolchain

- Linaro - <https://www.linaro.org/downloads/>
- By Linux distribution -
 - `sudo apt-get install gcc-arm-linux-gnueabi`
- BSP
- CodeSourcery



Installing and using Toolchain

➤ Add the path to toolchain binaries in your PATH: export

- [CMD] `PATH=${TOOLCHAIN_PATH}/bin/:$PATH`

➤ Compile your applications

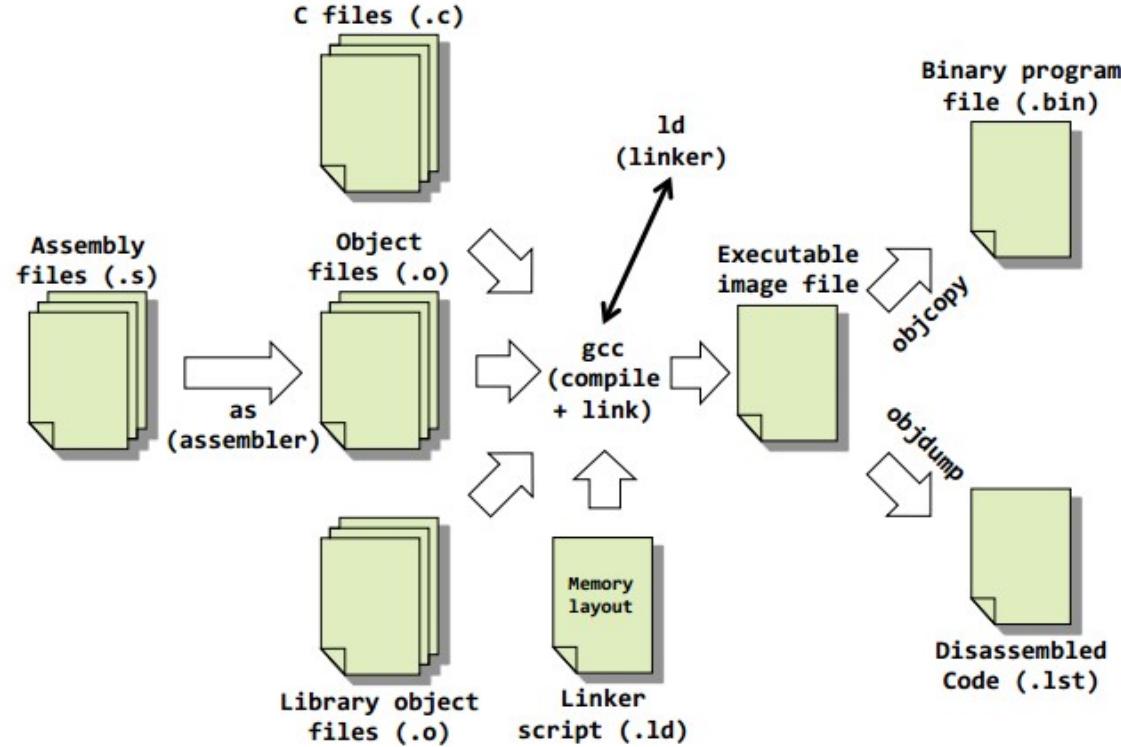
- [CMD] `${PREFIX}-gcc -o testme testme.c`

➤ PREFIX

- depends on the toolchain configuration

Compile, Assembler, Linker

Software Development Tools Overview





Tools Descriptions

» C/C++ compiler

- produces ARM machine code object modules

» Assembler

- Translates Assembly Language Source Files Into Machine Language Object modules

» Linker

- Combines object files into a single executable object module

Create Linux Library



Linux Library

► Static Libraries

- statically aware

► Dynamically Linked "Shared Object" Libraries

- Dynamically linked at run time



Static Libraries

➤ static_lib_name.a

➤ Create static library with **ar**

- **ar --help**

- **ar -cvq libctest.a test1.o test2.o**

➤ Compile

- **gcc -o test main.c libctest.a**

- **gcc -o test main.c -L/path/to/library-directory -lctest**



Dynamically Linked "Shared Object" Libraries

- » Dynamic_lib_name.so
- » Create share library
 - gcc -fPIC -shared -o libctest.so.1 test1.o test2.o
 - ln -s libctest.so.1 libctest.so.0.1
 - ln -s libctest.so.0.1 libctest.so
- » gcc -o test main.c -L/library_PATH/ -lctest
- » export LD_LIBRARY_PATH=LIB_PATH:\$LD_LIBRARY_PATH
- » ./test



Dynamically Linked "Shared Object" Libraries

- ldconfig
- configure dynamic linker run-time bindings
- /etc/ld.so.conf
 - 1. \$ vim /etc/ld.so.conf
 - and add LIB in path /usr/local
 - 2. #ldconfig /usr/local/
 - /etc/ld.so.cache



What and Need soname ?

Real-name	libctest.so.1.0
Soname	libctest.so.1
Linkname	libctest.so

→ libctest.so.1.0
→ libctest.so.1

Modify

Real-name	libctest.so.1.1
Soname	libctest.so.1
Linkname	libctest.so

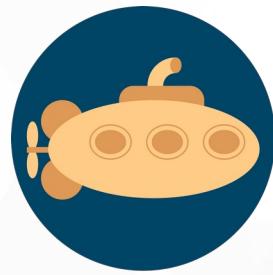
→ libctest.so.1.1
→ libctest.so.1

Real-name	libctest.so.1.5
Soname	libctest.so.1
Linkname	libctest.so

→ libctest.so.1.5
→ libctest.so.1

main.c no need to re-compile

```
gcc -o test main.c -L/library_PATH/ -lctest
```



U-Boot

U-boot



Bootloader

- » What is bootloader
- » Boot : short bootstrap

- Initialize basic of SOC (CPU, RAM, CLK)
- BL1, BL2

- » Loader

- Load OS to RAM(DDR) form storage
- u-boot



Different Embedded Linux bootloader

» U-boot

» UEFI

» Redboot

» Stubby (Linaro) ...

» Anyway, they are same target

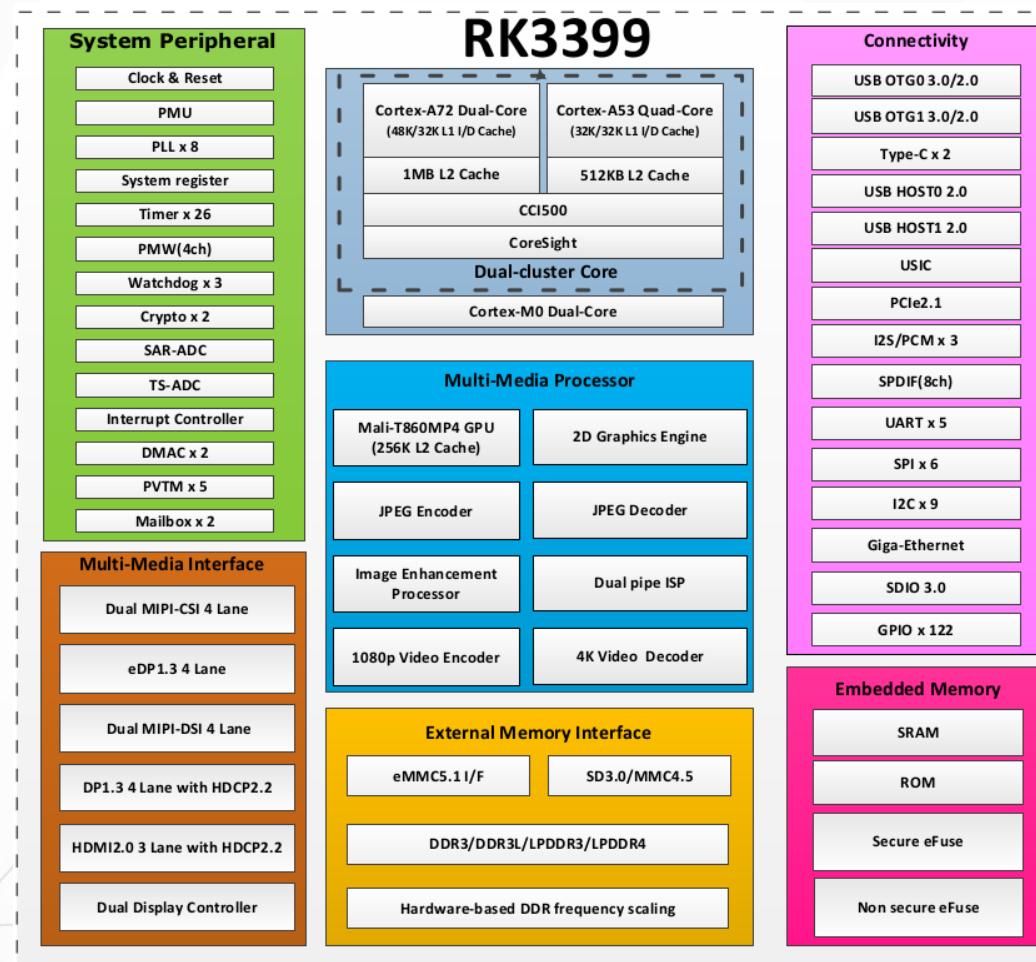
- Load and boot OS to RAM from storage



Concepts of the Boot Loader

- ▶ Boot Loader is varied from CPU to CPU, from board to board.
- ▶ All the system software and data are stored in some kind of nonvolatile memory.
- ▶ Operation Mode of Boot Loader
 - **Boot : Initialize basic of SOC**
 - **Load : load OS to RAM then execute**

RK3399 SOC



http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension



Embedded Linux System Booting

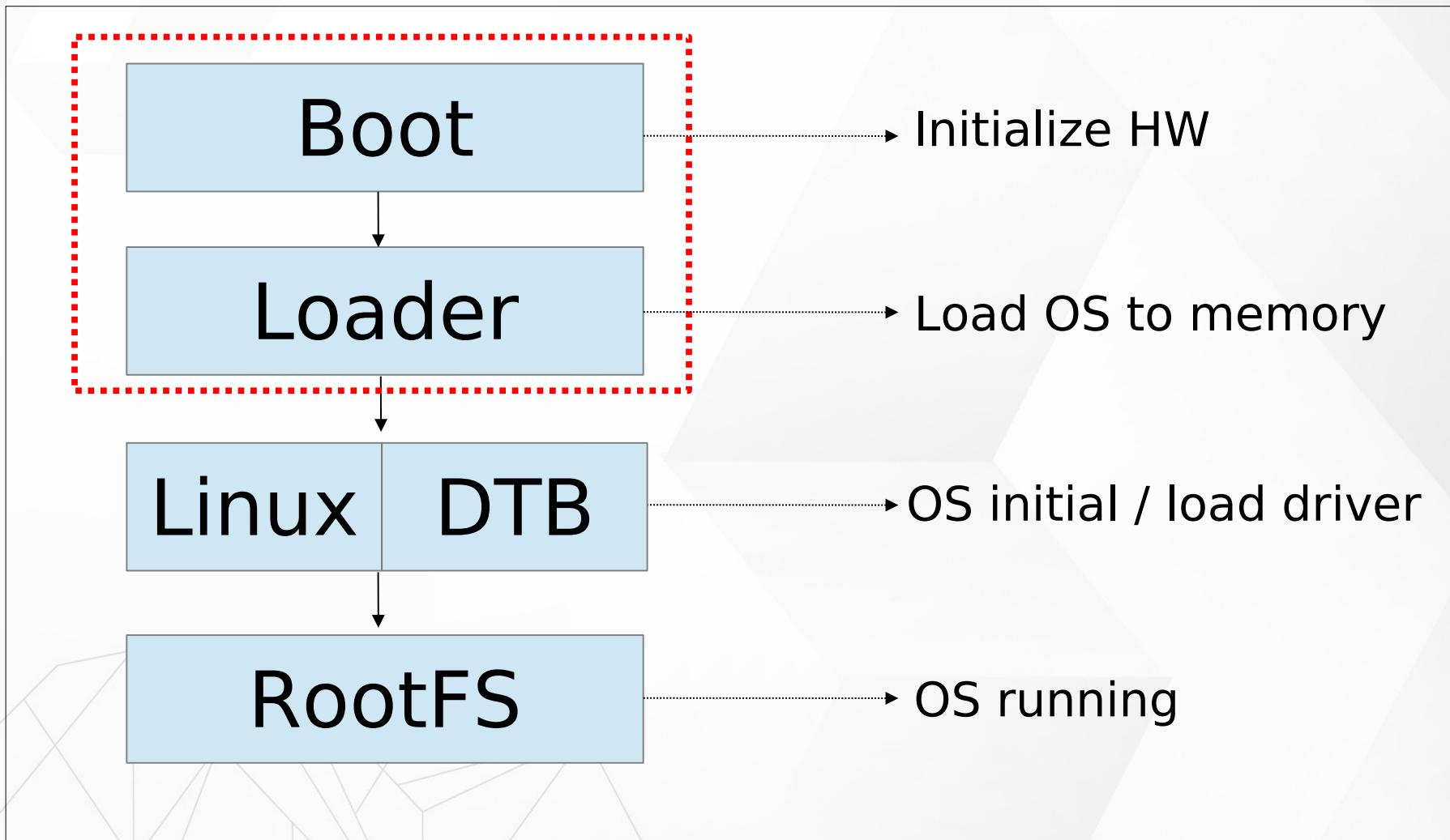




Image Partition

partmap.txt – Image layout in SD card

Loader	idbloader.img	0x8000,0x280000
u-boot	Uboot.img	0x800000,0x0400000
Trust	Trust.img	0xC00000,0x0400000
Linux kernel	Linux kernel	SD Card Part List 4
RootFS	rootfs.img	SD Card Part List 5



Boot

▶ Power On BootROM code (work in cache)

- Load BL1

▶ BL1 (work in cache)

- Initial simple exception vectors, PLL (clock)
- Initial Multi-CPU
- Load BL2

▶ BL2 (work in cache)

- Initial DDR memory
- Initial C environment (stack, heap,)
- Load BL31



Boot

▶ BL31 (work in DDR)

- Initial exception vectors
- Load BL32 (u-boot)

▶ BL32 U-boot

- Initial storage device
- Load Linux Kernel

▶ Kernel

- kernel/Documentation/arm64/booting.txt
- Mount RootFS

Introduce U-boot



U-boot

- » Das U-Boot -- the Universal Boot Loader
- » <http://www.denx.de/wiki/U-Boot>
- » GitHub for u-boot
- » Open Source follow GPL
- » Supply many CPU
 - PPC, ARM, x86, MIPS, AVR32 ...
- » Supply basic periphery devices
 - UART, Flash, SD/MMC



u-boot directory structure

- » Arch → Many types CPU : Arm, mips, i386 ...
- » Board → Many types develop board : Samsung, ti, davinci ...
- » Tools → Make Image (u-boot, linux) or S-Record image tool
- » Drivers → Some HW control code
- » Fs → Supply file system : fat, jffs2, ext2, cramfs
- » Lib → General public library : CRC32/16, bzlib, ldiv ..
- » Disk → Supply disk driver and partition handling



u-boot directory structure

- » Common → Major command and relation environment setting source code
- » Api → Implement unrelated hardware code
- » Example → Standalone application



u-boot directory structure about rk3399

► **arch/arm/cpu/armv8/**

- ARMv8 relate

► **arch/arm/cpu/armv8/rk33xx/**

- rk3399 related
- Clock, i2c, irom, mmc, emmc ...

► **board/rockchip/rk33xx/**

- Board level related
- Peripheral initial



Rk3399 Configure File

» Common

- u-boot command related

» include/configs/

- config_distro_bootcmd.h
- evb_rk3399.h
- rk3399_common.h



How to Build u-boot

» Easy Build :

[CMD] cd ./rockchip-bsp

[CMD] ./build/mk-uboot.sh rockpi4b

» Clear :

[CMD] make distclean

» Configure :

[CMD] make rock-pi-4b-rk3399_defconfig

» Build :

[CMD] make -j4

» Create Image :

[CMD] cd rockchip-bsp/rkbin/tool/

[CMD] loaderimage --pack --uboot ../../u-boot/u-boot-dtb.bin /uboot.img



U-boot Configure

▶ Change u-boot about

- Command
 - [cmd] help
- Boot parameter
 - Kernel load address
- Function
 - Enable LED



U-boot Configure

► Edit Configure

► Method 1

- menuconfig
 - [cmd] make menuconfig

► Method 2

- Edit configure file
 - config_distro_bootcmd.h
 - evb_rk3399.h
 - rk3399_common.h

U-boot Common Function



Operating U-boot

- ▶ Understand and use command
- ▶ Understand and modify parameters



Help

» \$ help

→ help mm

» \$?

=> help mm

mm - memory modify (auto-incrementing address)

Usage:

mm [.b, .w, .l, .q] address

=>



Help

» help

→ print command description/usage

» \$ help

→ help mm

» \$?

=> help mm

mm - memory modify (auto-incrementing address)

Usage:

mm [.b, .w, .l, .q] address

=>



md

» md

- memory display
- md [.b, .w, .l, .q] address [# of objects]

```
=> md 0x02080000
02080000: 4e04e260 e461206c 0808a115 2a666646    `...Nl a.....Fff*
02080010: 88689ca1 224002e2 2e000a62 20a26262    ...h....@"b...bb.
02080020: a8207386 60a626e4 00016006 62a40642    .s ...&..`..`..B..b
02080030: e000a239 62476067 a3284802 24e66242    9...g`Gb.H(.Bb.$
02080040: 22300806 32a0c270 41620081 62042664    ..0"p..2..bAd&.b
```



mw

➤ mw

- memory write
- mw [.b, .w, .l, .q] address value [count]

```
=> mw.l 0x02080000 0x12345678 1
=> md.l 0x02080000
02080000: 12345678 e461206c 0808a115 2a666646      xv4.1 a.....Fff*
02080010: 88689ca1 224002e2 2e000a62 20a26262      ..h...@"b...bb.
02080020: a8207386 60a626e4 00016006 62a40642      .s ...&..`...B..b
02080030: e000a239 62476067 a3284802 24e66242      9...g`Gb.H(.Bb.$
02080040: 22300806 32a0c270 41620081 62042664      ..0"p...2..bAd&.b
```



mmc

➤ mmc list

→ lists available devices

```
=> mmc list
mmc@fe310000: 2
mmc@fe320000: 1 (SD)
sdhci_@fe330000: 0
```



mmc

▶ mmc info

→ display info of the current MMC device

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> mmcinfo
Device: mmc@fe320000
Manufacturer ID: 3
OEM: 5344
Name: SU04G
Bus Speed: 50000000
Mode: SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 3.7 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
```



mmc

▶ mmc part

→ lists available partition on current mmc device

```
=> mmc part

Partition Map for MMC device 1  --  Partition Type: DOS

Part      Start Sector      Num Sectors      UUID          Type
 1_        196608            7547904        97ddff01-01      83
```



mmc

▶ mmc dev

- show or set current mmc device [partition]
- mmc dev [dev] [part]

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> mmcinfo
Device: mmc@fe320000
Manufacturer ID: 3
OEM: 5344
Name: SU04G
Bus Speed: 50000000
Mode: SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 3.7 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
```



FAT List

» fatls

- list files in a directory
- fatls <interface> [<dev[:part]>] [directory]
 - list files from 'dev' on 'interface' in a 'directory'

```
=> fatls mmc 1:4
155639 config-4.4.154-113-rockchip-gdb9dfc2cdd25
20371464 vmlinuz-4.4.154-113-rockchip-gdb9dfc2cdd25
extlinux/
dtbs/
overlays/
0371464 vmlinuz-4.4.154
1968 hw_intf.conf
4786387 System.map-4.4.154-00039-g00fccd3
156441 config-4.4.154
4786387 System.map-4.4.154
5038020 initrd.img-4.4.154
```



FAT Load

► fatload

- fatload - load binary file from a FAT filesystem
- fatload <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]
load binary file 'filename' from 'dev' on 'interface'
to address 'addr' from fat filesystem

```
=> fatload mmc 1:4 ${kernel_addr_r} vmlinuz-4.4.154  
reading vmlinuz-4.4.154  
20371464 bytes read in 858 ms (22.6 MiB/s)
```

```
=> fatload mmc 1:4 ${fdt_addr_r} dtbs/4.4.154/rockchip/rockpi-4b-linux.dtb  
reading dtbs/4.4.154/rockchip/rockpi-4b-linux.dtb  
94603 bytes read in 15 ms (6 MiB/s)
```

```
=> fatload mmc 1:4 ${ramdisk_addr_r} initrd.img-4.4.154  
reading initrd.img-4.4.154  
5038020 bytes read in 215 ms (22.3 MiB/s)
```



printenv

➤ printenv

- print environment variables
- printenv name

```
=> printenv
altbootcmd=setenv boot_syslinux_conf extlinux/extlinux-rollback.conf;run distro_bootcmd
arch=arm
baudrate=1500000
board=eVB_rK3399
board_name=eVB_rK3399
boot_a_script=load ${devtype} ${devnum}:${distro_bootpart} ${scriptaddr} ${prefix}${script}; so
boot_efi_binary=load ${devtype} ${devnum}:${distro_bootpart} ${kernel_addr_r} efi/boot/bootaa64
ernel_addr_r} ${fdt_addr_r};else booteFI ${kernel_addr_r} ${fdtcontroladdr};fi
boot eFI bootmar;if fdt addr ${fdt_addr_r}: then booteFI bootmar ${fdt_addr_r}:else booteFI boo

=> printenv loadimage
loadimage=ext4load mmc 1:1 ${kernel_addr_r} boot/Image
```



setenv

- » saveenv
- » setenv

- set environment variables
- setenv name value

```
=> setenv test 12345
=> printenv test
test=12345
=> setenv test
=> printenv test
## Error: "test" not defined
```



Simple Script

LED Blank

Script

```
1=> while true; do; gpio toggle 125; sleep 1; done
2gpio: pin 125 (gpio 125) value is 0
3gpio: pin 125 (gpio 125) value is 1
4gpio: pin 125 (gpio 125) value is 0
5gpio: pin 125 (gpio 125) value is 1
6gpio: pin 125 (gpio 125) value is 0
7gpio: pin 125 (gpio 125) value is 1
```



run

» run

- run commands in an environment variable
- run var [...]

```
=> run bootcmd
switch to partitions #0, OK
mmc1 is current device
Scanning mmc 1:4...
Found /extlinux/extlinux.conf
pxefile_addr_str = 0x00500000
bootfile = /extlinux/extlinux.conf
Retrieving file: /extlinux/extlinux.conf
reading /extlinux/extlinux.conf
1646 bytes read in 5 ms (321.3 KiB/s)
select kernel
1:   kernel-4.4.154
2:   kernel-4.4.154-999-rockchip-gcfa47f25e
3:   kernel-4.4.154-113-rockchip-gdb9dfc2cdd25
4:   kernel-4.4.154-00039-g00fccd3
5:   kernel-4.4.154
Enter choice: Retrieving file: /hw_intfc.conf
reading /hw_intfc.conf
1968 bytes read in 4 ms (480.5 KiB/s)
```

command



booti

▶ booti

- booti - boot Linux kernel 'Image' format from memory
- booti [addr [initrd[:size]] [fdt]]

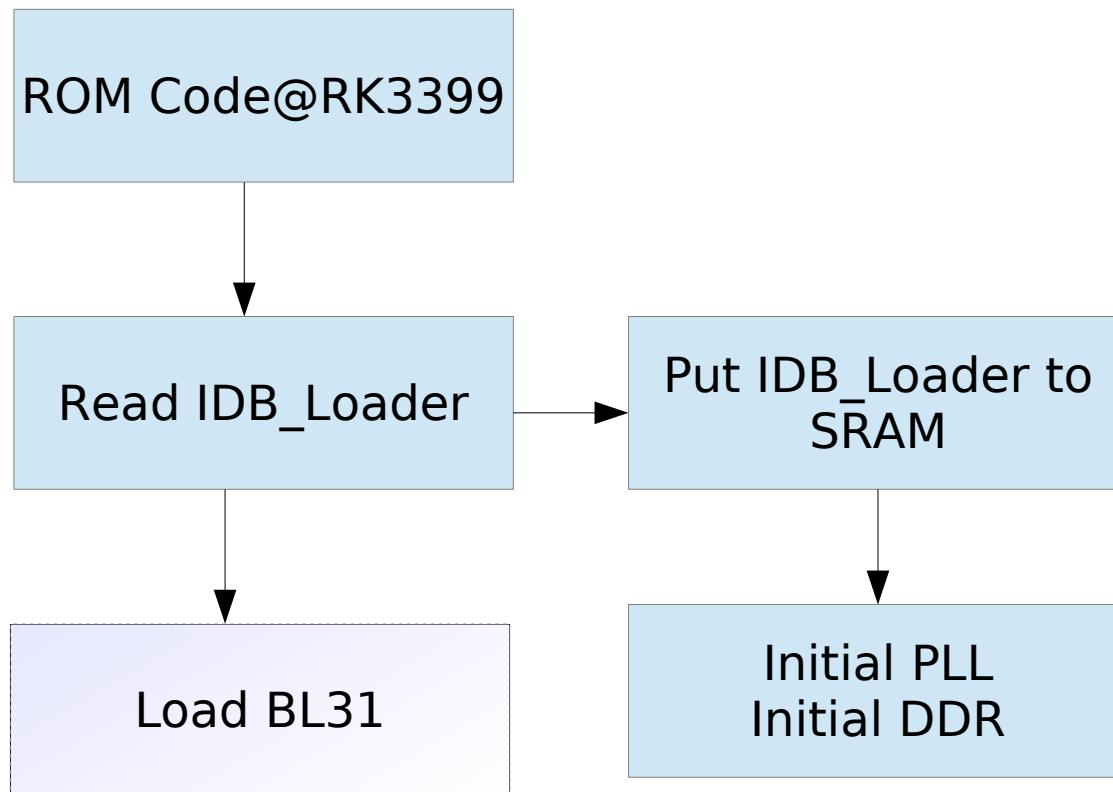
```
=> booti ${kernel_addr_r} ${ramdisk_addr_r}:5038020 ${fdt_addr_r}
## Flattened Device Tree blob at 01f00000
Booting using the fdt blob at 0x1f00000
Loading Ramdisk to 76db8000, end 7bdf0020 ... OK
Loading Device Tree to 0000000076d9d000, end 0000000076db718a ... OK
Adding bank: start=0x00200000, size=0x7fe00000

Starting kernel ...

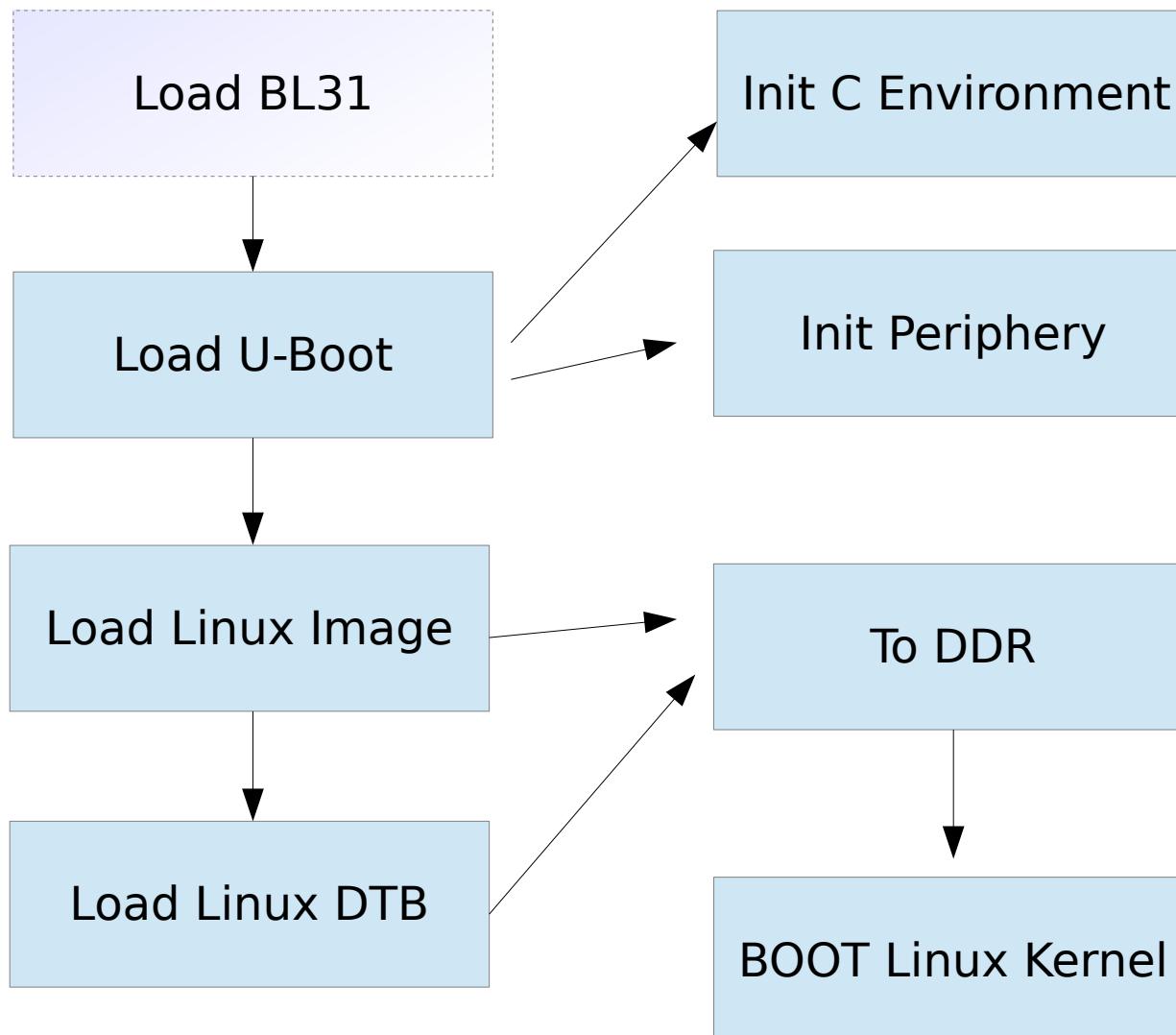
[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 4.4.154 (slash@slash-ThinkPad-E14-Gen-2) (gcc version 7.3.1 20180425 [linaro-7.3-2018.05
revision d29120a424ecfbc167ef90065c0eeb7f91977701] (Linaro GCC 7.3-2018.05) ) #4 SMP Sun Jan 23 15:55:32 CST 2022
```

Boot Linux kernel

System Start Up



System Start Up





Boot Linux Kernel Command

▶ bootcmd

- Power On will **auto run content of bootcmd**

```
[CMD] printenv bootcmd
```

```
bootcmd=run distro_bootcmd;boot_android ${devtype} ${devnum};bootrkp;
```



Boot OS (Linux) Command

▶ booti

→ boot Linux **kernel 64 bit standard kernel image**

▶ boota

→ boot **android style kernel image**

▶ Boot command

→ booti [Kernel image addr] [RAM disk adr] [DTB addr]

→ \$ booti **kernel_addr ramdisk_addr DTB_addr**



Linux Kernel Command (bootargs)

» bootargs

- \$ printenv bootargs (Linux kernel command)

```
earlyprintk console=ttyFIQ0,1500000n8 rw init=/sbin/init  
rootfstype=ext4 rootwait root=UUID=a54358ce-55c2-4f4f-bf6d-  
7106997cfb8f console=ttyS2,1500000n8
```

- earlycon : early console device
- console : Linux console device
- root : RootFS device

» Enter Linux kernel to check

- \$ cat /proc/cmd



How to jump to kernel

▶ Use boot command

→ cmd/cmd_booti.c

▶ Jump to Linux kernel

→ arch/arm/lib/bootm.c

→ boot_os_fn *bootm_os_get_boot_func(int os)

→ int do_bootm_linux(int flag, int argc, char *const argv[],
bootm_headers_t *images)

→ void (*kernel_entry)(void *fdt_addr,
void *res0,
void *res1,
void *res2);



Linux Enter Point

arch/arm/lib/bootm.c

```
static void boot_jump_linux(bootm_headers_t *images, int flag)
{
#ifndef CONFIG_ARM64
    void (*kernel_entry)(void *fdt_addr, void *res0, void *res1,
                         void *res2);
    int fake = (flag & BOOTM_STATE_OS_FAKE_GO);

    kernel_entry = (void (*)(void *fdt_addr, void *res0, void *res1,
                           void *res2))images->ep;

```

```
→ if (IMAGE_ENABLE_OF_LIBFDT && images->ft_len)
→ → r2 = (unsigned long)images->ft_addr;
→ else
→ → r2 = gd->bd->bi_boot_params;
```

```
→ unsigned long machid = gd->bd->bi_arch_number;
→ char *s;
→ void (*kernel_entry)(int zero, int arch, uint params);
→ unsigned long r2;
→ int fake = (flag & BOOTM_STATE_OS_FAKE_GO);

→ kernel_entry = (void (*)(int, int, uint))images->ep;
```

Assign DTB

Start To Kernel

Add Function to Board Setting File



evb-rk3399.c

► board/rockchip/evb_rk3399/evb-rk3399.c

► Add function to here

```
diff --git a/board/rockchip/evb_rk3399/evb-rk3399.c b/board/rockchip/evb_rk3399/evb-rk3399.c
index 8e3fce4aa4..e5b4041288 100644
--- a/board/rockchip/evb_rk3399/evb-rk3399.c
+++ b/board/rockchip/evb_rk3399/evb-rk3399.c
@@ -30,6 +30,11 @@ int rk_board_init(void)
        struct udevice *pinctrl, *regulator;
        int ret;

+       printf("%s\n", __func__);
+
+       /* user2 led power-off */
+       run_command("gpio clear 125", 0);
+
+       /*
+        * The PWM does not have dedicated interrupt number in dts and can
+        * not get periph_id by pinctrl framework, so let's init them here.
```

New a Command



Add Feature

- » Add command → cmd/
- » Add driver → driver/
- » Add application → example/



Add Command

- » How to create a command ?
- » Directory
 - cmd/ → booti.c , mmc.c, mem.c ...
- » U_BOOT_CMD(name,maxargs,rep,cmd,usage,help)
 - include/command.h



How to Command

cmd/cmd_version.c

```
static int do_version(struct cmd_tbl *cmdtp, int flag, int argc,
                      char *const argv[])
{
    char buf[DISPLAY_OPTIONS_BANNER_LENGTH];

    printf("hellow_wold\n");

    printf(display_options_get_banner(false, buf, sizeof(buf)));

    return 0;
}

U_BOOT_CMD(
    version,      1,      1,  do_version,
    "print monitor, compiler and linker version",
    ""
);
```



Include/command.h

```
#define U_BOOT_CMD(_name, _maxargs, _rep, _cmd, _usage, _help)
```



Hello World Command

```
Command line interface
Use the menu. <Enter> selects submenus ---> (or empty submenus ----). | 
| excludes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit
in [ ] excluded <M> module < > module capable

[*] Support U-Boot commands
[*] Command Hello World Sample
  -- Use hush shell
  (=) Shell prompt
    Autoboot options --->
    *** FASTBOOT ***
[*] Fastboot support --->
  *** Commands ***
  Info commands --->
  Boot commands --->
  Environment commands --->
  Memory commands --->
  Compression commands --->
  Device access commands --->
  Shell scripting commands --->
  Network commands --->
[ ] Enable memtester for ddr
  Misc commands --->
+(+)
```



Hello World Command

cmd/Makefile

```
obj-$(CONFIG_CMD_USB_MASS_STORAGE) += usb_mass_storage.o
obj-$(CONFIG_CMD_USB_SDP) += usb_gadget_sdp.o
obj-$(CONFIG_CMD_THOR_DOWNLOAD) += thordown.o
obj-$(CONFIG_CMD_XIMG) += ximg.o
obj-$(CONFIG_CMD_YAFFS2) += yaffs2.o
obj-$(CONFIG_CMD_SPL) += spl.o
obj-$(CONFIG_CMD_ZIP) += zip.o
obj-$(CONFIG_CMD_ZFS) += zfs.o

obj-$(CONFIG_CMD_DFU) += dfu.o
obj-$(CONFIG_CMD_GPT) += gpt.o
obj-$(CONFIG_CMD_ETHSW) += ethsw.o
obj-$(CONFIG_CMD_HELLOWORLD) += helloworld.o

# Power
obj-$(CONFIG_CMD_PMIC) += pmic.o
obj-$(CONFIG_CMD_REGULATOR) += regulator.o

obj-$(CONFIG_CMD_BLOB) += blob.o
endif # !CONFIG_SPL_BUILD
```



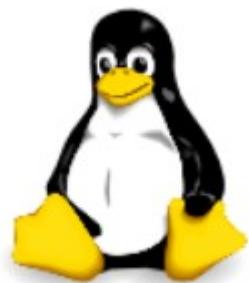
Hello World Command

cmd/Kconfig

```
menu "Command line interface"

config CMDLINE
    bool "Support U-Boot commands"
    default y
    help
        Enable U-Boot's command-line functions. This provides a means
        to enter commands into U-Boot for a wide variety of purposes. It
        also allows scripts (containing commands) to be executed.
        Various commands and command categorys can be individually enabled.
        Depending on the number of commands enabled, this can add
        substantially to the size of U-Boot.

config CMD_HELLOWORLD
    bool "Command Hello World Sample"
    depends on CMDLINE
    help
        This option enables the Hello World as command line
        Sample.
```



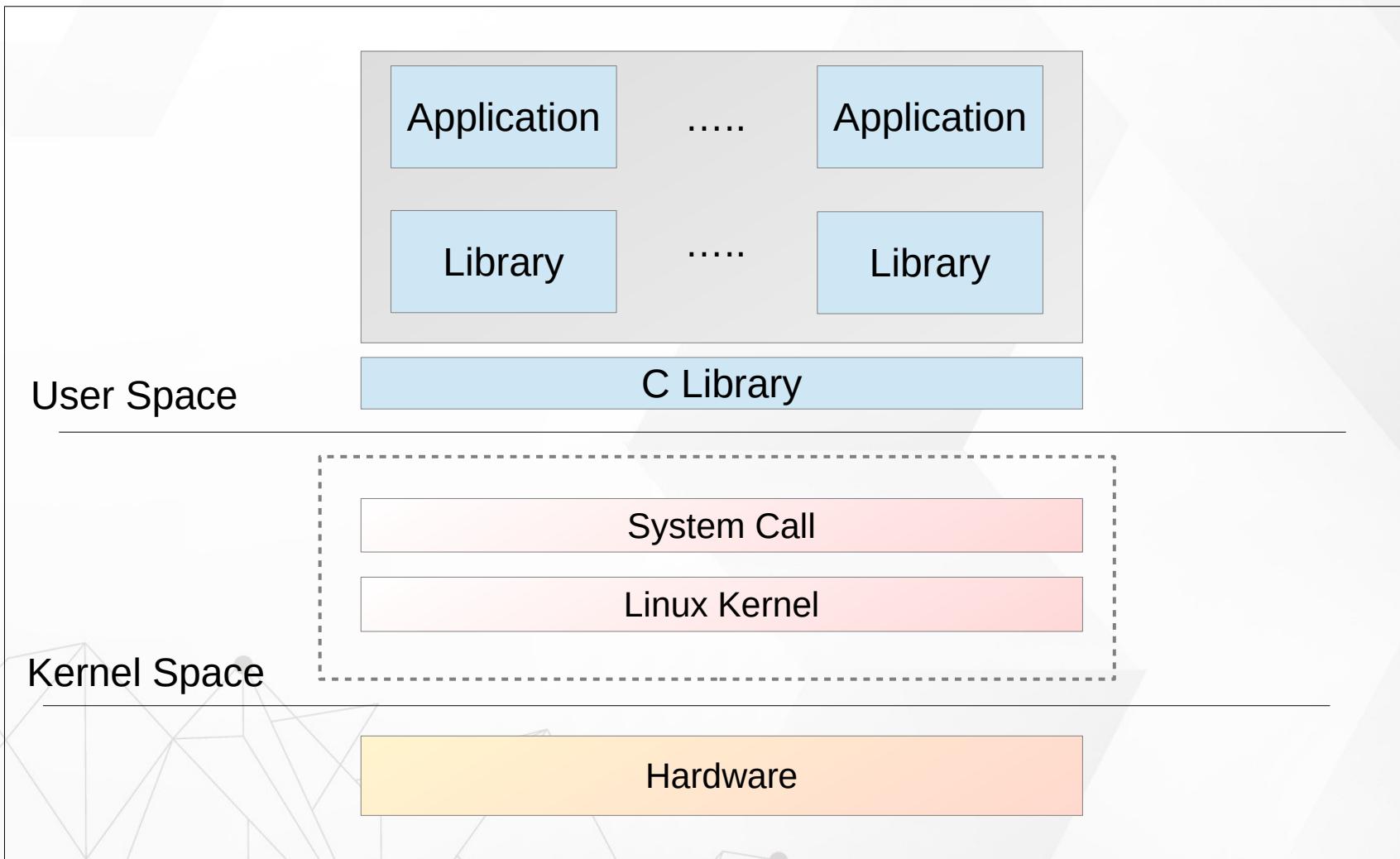
Linux Kernel



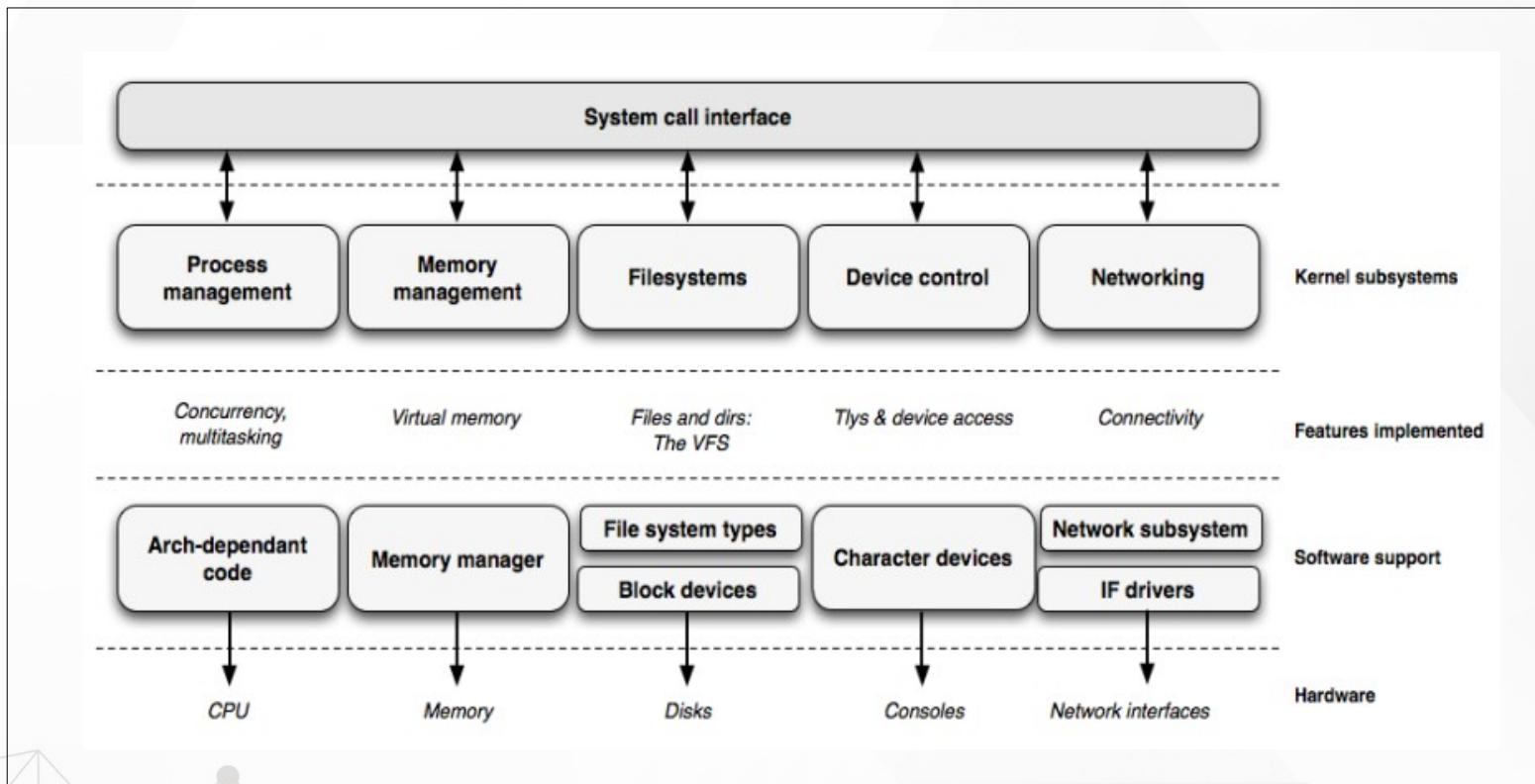
Linux kernel key features

- ▶ Portability and hardware support
- ▶ Scalability
- ▶ Compliance to standards and interoperability
- ▶ Exhaustive networking support
- ▶ Stability and reliability
- ▶ Modularity
- ▶ Easy to program.

Linux Kernel in the System



Linux Kernel





Kernel Source

- » <https://www.kernel.org/>
- » Many chip vendors
- » kernel sub-communities
 - Architecture communities
 - ARM, MIPS, PowerPC ...
 - device drivers communities
 - I2C, SPI, USB, PCI, network ...



Programming language

- Implemented in C like all Unix systems
- A little Assembly is used too
- **No C++ used**
- **No C library**
- **No floating point** computation
- Kernel code has to supply its own library implementations
 - X : printf(), memset(), malloc(),...
 - O: printk(), memset(), kmalloc()



Linux Sources Important Folder

▶ Kernel Image

- arch/<ARCH>/boot/
- Arch/arm64/boot

▶ DTS

- arch/<ARCH>/boot/dts
- Arch/arm64/boot/dts/rockchip/

▶ driver/

▶ Documentation/

Kernel Basic Command



Basic Build Command

» \$ make

- Build all

» \$ make dtbs

- Build Device-tree only

» \$ make modules

- Build kernel modules only

» \$ make modules_install

- Install all modules to folder



Basic Build Command

- ▶ [CMD] make modules_install
 - Install all modules to INSTALL_MOD_PATH
- ▶ [CMD] make mrproper
 - Remove all generated files (.config)
- ▶ [CMD] make clean
- ▶ [CMD] make distclean
 - Remove editor backup and patch reject files

Make Help

[CMD] make help

```
acs5k_defconfig      - Build for acs5k
acs5k_tiny_defconfig - Build for acs5k_tiny
afeb9260_defconfig   - Build for afeb9260
ag5evm_defconfig    - Build for ag5evm
am200epdkit_defconfig - Build for am200epdkit
ap4evb_defconfig    - Build for ap4evb
armadillo800eva_defconfig - Build for armadillo800eva
assabet_defconfig    - Build other generic targets:
at91_dt_defconfig    - Bui all          - Build all targets marked with [*]
at91rm9200_defconfig - Bui * vmlinux     - Build the bare kernel
at91sam9260_defconfig - Bui * modules    - Build all modules
at91sam9261_defconfig - Bui modules_install - Install all modules to INSTALL_MOD_PATH (default: /)
at91sam9263_defconfig - Bui firmware_install - Install all firmware to INSTALL_FW_PATH
(atdefault: $(INSTALL_MOD_PATH)/lib/firmware)
at91sam9g20_defconfig - Bui dir/         - Build all files in dir and below
at91sam9g45_defconfig - Bui dir/file.[o|S] - Build specified target only
at91sam9rl_defconfig - Bui dir/file.lst  - Build specified mixed source/assembly target only
(atrequires a recent binutils and recent build (System.map))
at91x40_defconfig    - Bui dir/file.ko    - Build module including final link
badge4_defconfig     - Bui modules_prepare - Set up for building external modules
bcmring_defconfig    - Bui tags/TAGS      - Generate tags file for editors
bonito_defconfig     - Bui cscope        - Generate cscope index
cam60_defconfig      - Bui gtags         - Generate GNU GLOBAL index
cerfcube_defconfig   - Bui kernelrelease - Output the release version string
cm_x2xx_defconfig   - Bui kernelversion  - Output the version stored in Makefile
cm_x300_defconfig   - Bui headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
(cndefault: /home/xlloss/work/tiny-4412/build/linux_3.5.0_tiny4412/usr)
colibri_pxa270_defconfig - Bui ...
colibri_pxa300_defconfig - Bui ...
collie_defconfig     - Build for collie
corgi_defconfig      - Build for corgi
cpu9260_defconfig   - Build for cpu9260
cpu9n20_defconfig   - Build for cpu9n20
```

How to Build Linux Kernel



Specifying Cross-compilation

- ▶ **make ARCH=arm64 CROSS_COMPILE=arm-linux- ...**
 - [CDM] export ARCH=arm64
 - [CMD] export CROSS_COMPILE=aarch64-linux-gnu-

- ▶ Add above setting to script
 - [CMD] source \$PATH/set_toolchain.sh



set_toolchain.sh

```
export PATH=/home/cadtc/host_share_folder/toolchain/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/  
# Toolchain path add to environment variable  
  
export ARCH=arm64  
# Set SOC architecture type  
  
export CROSS_COMPILE=aarch64-linux-gnu-  
#Set compile prefix name  
  
export KERNELDIR=/home/cadtc/rockchip-bsp/kernel  
#Set Linux kernel source path
```



Predefined Configuration Files

» Default configuration

- arch/<arch>/configs/

» **make nanopi4_linux_defconfig**

» To create your own default configuration file

- **make savedefconfig**, to create a minimal configuration file
- **mv defconfig arch/arm64/configs/myown_defconfig**



Kernel Compilation

➤ Build kernel Image → **make -j4**

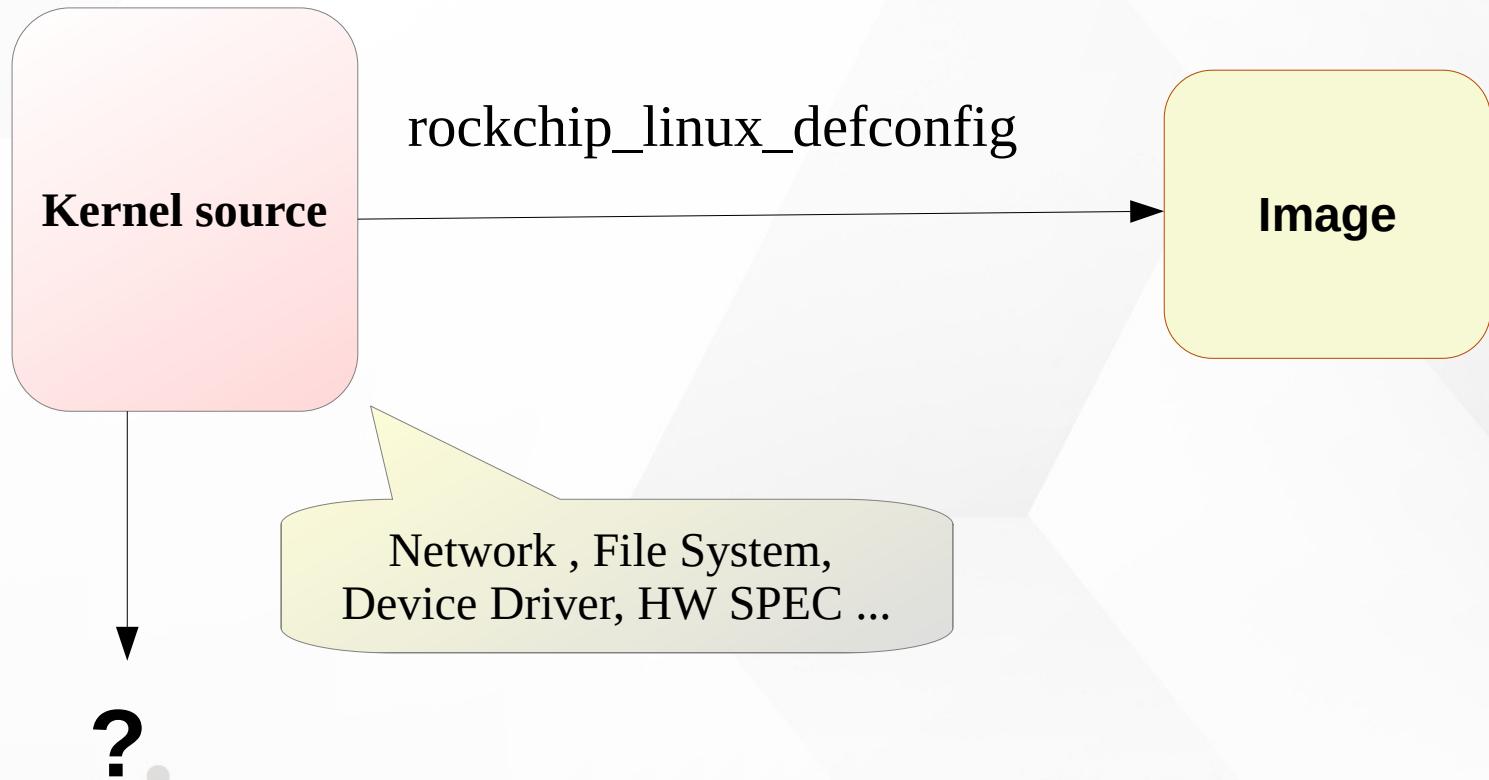
- To run multiple jobs in parallel if you have multiple CPU cores

➤ Generates Image

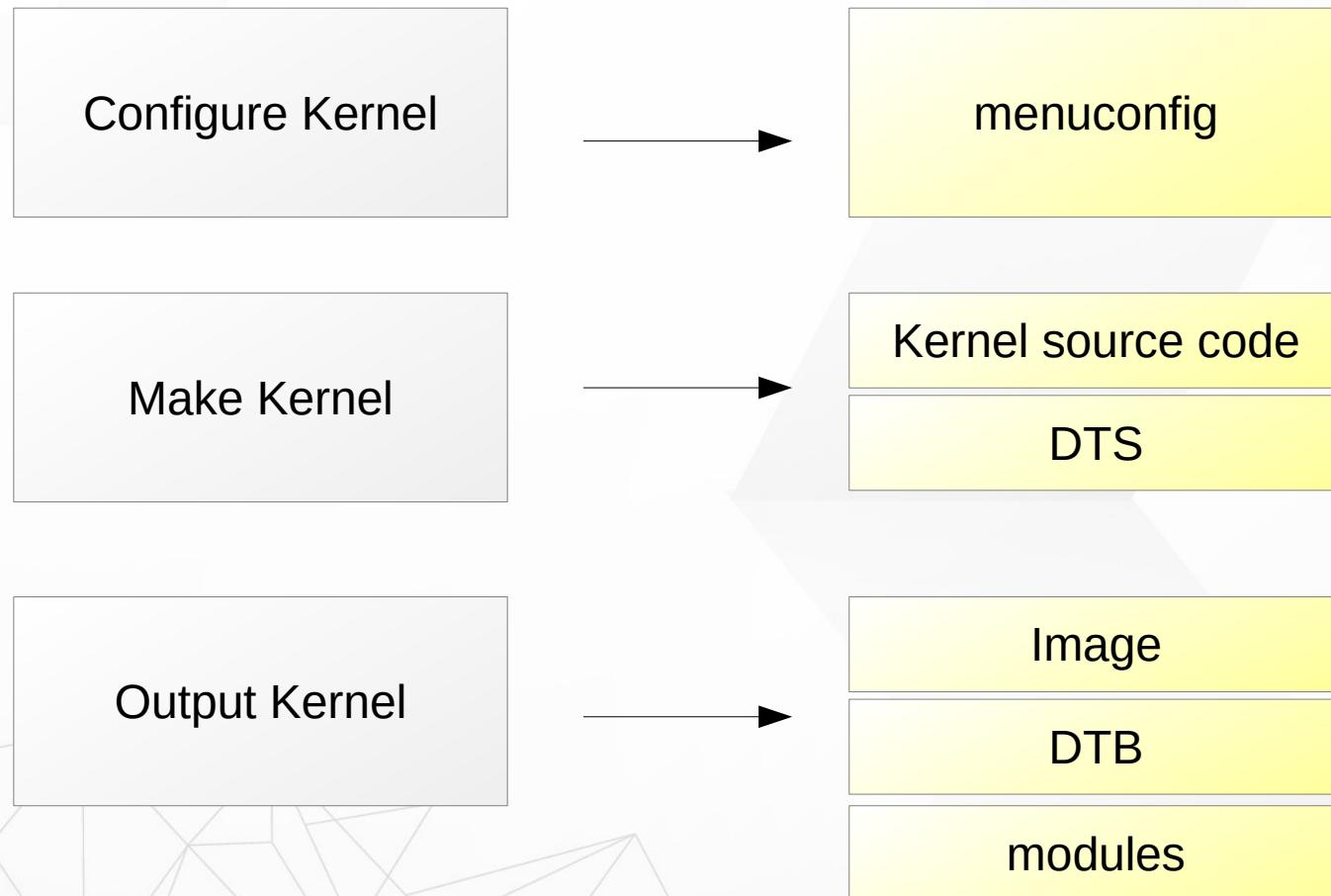
- arch/arm64/boot/Image
 - **Image** for ARM64,
- arch/arm64/boot/dts/rockchip/
 - DTB : rk3399-nanopi4-rev01.dtb
 - DTB : rk3399-nanopi4-rev21.dtb



Kernel Configuration



Kernel Configuration



How to Select Feature in Kernel



Kernel Configuration

- ▶ The kernel configuration and build system is based on multiple **Makefiles**
- ▶ The configuration is stored in the **.config** file at the root of kernel sources
- ▶ As options have dependencies, typically never edited by hand, but through graphical or text interfaces
 - [CMD] make menuconfig → Text
 - [CMD] make xconfig → graphical

Kernel Configuration

```
4096 Apr 20 11:56 .
4096 Mar 31 08:42 ..
4096 Mar 31 08:41 android
4096 Mar 31 08:41 arch
419 Mar 31 08:41 backported-features
4096 Mar 31 08:41 block
459 Mar 31 08:41 build.config.cuttlefish.aarch64
457 Mar 31 08:41 build.config.cuttlefish.x86_64
296 Mar 31 08:41 build.config.goldfish.arm
303 Mar 31 08:41 build.config.goldfish.arm64
277 Mar 31 08:41 build.config.goldfish.mips
279 Mar 31 08:41 build.config.goldfish.mips64
298 Mar 31 08:41 build.config.goldfish.x86
303 Mar 31 08:41 build.config.goldfish.x86_64
4096 Mar 31 08:41 certs
9 Mar 31 08:41 .checkpatch.conf
154048 Apr 20 11:56 .config
```

.config

```
cuttlefish_defconfig
defconfig
lsk_defconfig
nanopi4_linux_defconfig
px30_linux_defconfig
px30_linux_robot_defconfig
ranchu64_defconfig
rk1808_linux_defconfig
rk1808_x4_linux_defconfig
rk3308_linux_defconfig
rk3326_linux_defconfig
rk3326_linux_robot_defconfig
rk3399pro_npu_defconfig
rk3399pro_npu_PCIE_defconfig
rockchip_cros_defconfig
rockchip_defconfig
rockchip_linux_defconfig
```

→ \${KERNEL}/arch/arm64/configs/
rockchip_linux_defconfig



Kernel or Module?

- The kernel image is a single file, resulting from the linking of all object files that correspond to features enabled in the configuration
- Some features (device drivers, file-system, etc.) can however be compiled as modules

Menuconfig



menuconfig

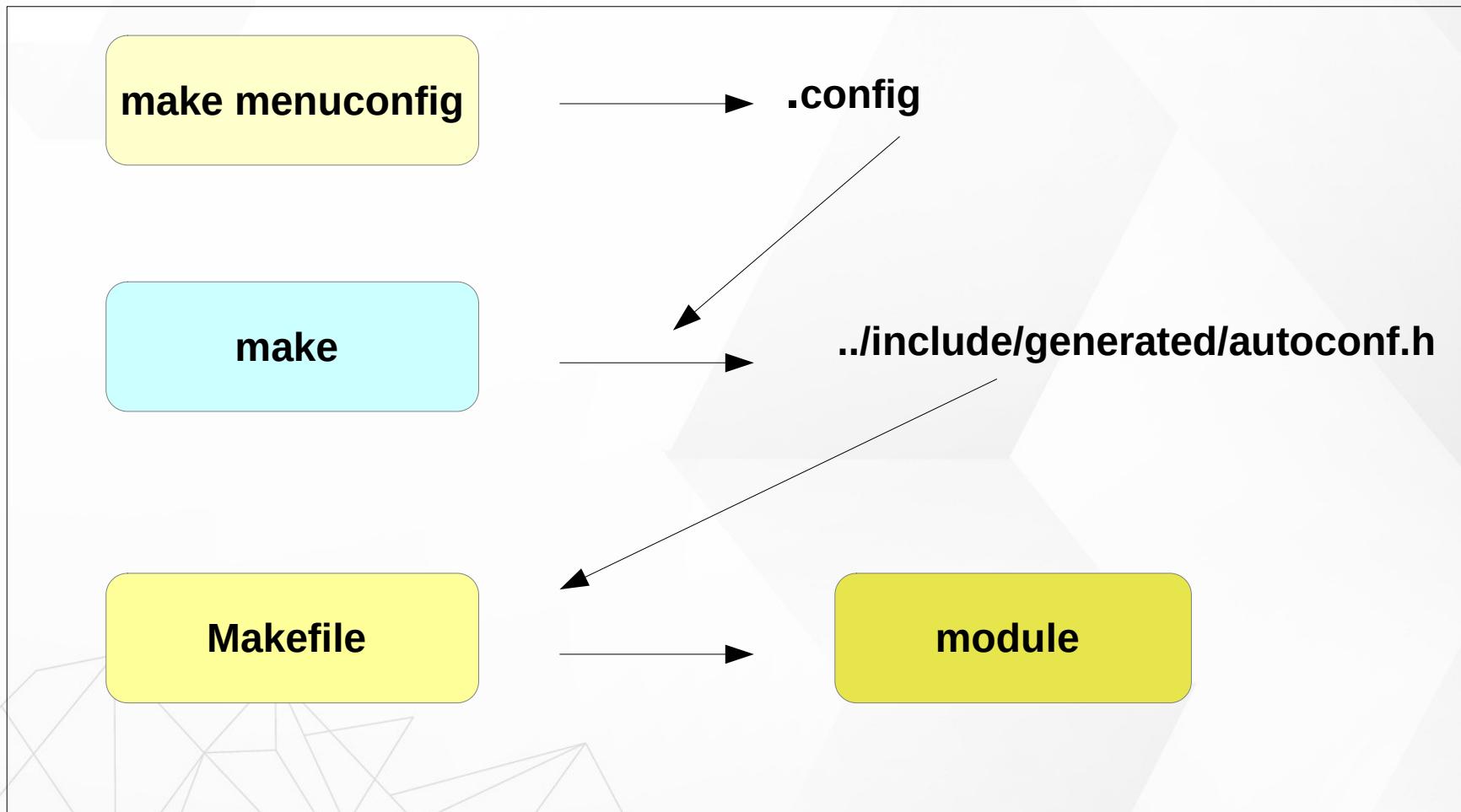
```
Linux/arm64 4.4.179 Kernel Configuration
menus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pres
</> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

[ ] General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    Platform selection --->
    Bus support --->
    Kernel Features --->
    Boot options --->
    Userspace binary formats --->
    Power management options --->
    CPU Power Management --->
[*] Networking support --->
    Device Drivers --->
    Firmware Drivers --->
[ ] ACPI (Advanced Configuration and Power Interface) Support ----
    File systems --->
[ ] Virtualization ----
    Kernel hacking --->
    Security options --->
-- Cryptographic API --->
    Library routines --->
```

Kernel Configuration Options

```
[ ] Enable AHB driver for NVIDIA Tegra SoCs
  Generic Driver Options --->
    Bus devices --->
      {M} Connector - unified userspace <-> kernelspace linker ----
        <-> Memory Technology Device (MTD) support ----
        -*- Device Tree and Open Firmware support --->
        <-> Parallel port support ----
      [*] Block devices --->
        <-> NVM Express block device
        Misc devices --->
          SCSI device support --->
        <-> Serial ATA and Parallel ATA drivers (libata) --->
      [*] Multiple devices driver support (RAID and LVM) --->
        <-> Generic Target Core Mod (TCM) and ConfigFS Infrastructure ----
      [ ] Fusion MPT device support ----
        IEEE 1394 (FireWire) support --->
      [*] Network device support --->
      [ ] Open-Channel SSD target support ----
        Input device support --->
        Character devices --->
        █ I2C support --->
      [*] SPI support --->
        <-> SPMI support ----
        <-> HSI support ----
        PPS support --->
        PTP clock support --->
        Pin controllers --->
        -*- GPIO Support --->
      <M> Dallas's 1-wire support --->
        -*- Power supply class support --->
      [*] Adaptive Voltage Scaling class support ---->
        <-> Hardware Monitoring support --->
        <-> Generic Thermal sysfs driver --->
      [*] Watchdog Timer Support --->
```

Configuration



Corresponding .config File Excerpt

```
#  
# I2C system bus drivers (mostly embedded / system-on-chip)  
#  
# CONFIG_I2C_CADENCE is not set  
# CONFIG_I2C_CBUS_GPIO is not set  
# CONFIG_I2C_DESIGNWARE_PLATFORM is not set  
# CONFIG_I2C_DESIGNWARE_PCI is not set  
# CONFIG_I2C_EMEV2 is not set  
CONFIG_I2C_GPIO=m  
# CONFIG_I2C_NOMADIK is not set  
# CONFIG_I2C_OCORES is not set  
# CONFIG_I2C_PCA_PLATFORM is not set  
# CONFIG_I2C_PXA_PCI is not set  
CONFIG_I2C_RK3X=y  
# CONFIG_I2C_SIMTEC is not set  
# CONFIG_I2C_XILINX is not set
```

.config

`$(KERNEL_PATH)/drivers/i2c/buses/
Makefile`

```
obj-$(CONFIG_I2C_PXA_PCI)          += i2c-pxa-pci.o  
obj-$(CONFIG_I2C_QUP)              += i2c-qup.o  
obj-$(CONFIG_I2C_RIIC)             += i2c-riic.o  
obj-$(CONFIG_I2C_RK3X)              += i2c-rk3x.o  
obj-$(CONFIG_I2C_S3C2410)           += i2c-s3c2410.o  
obj-$(CONFIG_I2C_SH7760)             += i2c-sh7760.o  
obj-$(CONFIG_I2C_SH_MOBILE)          += i2c-sh_mobile.o  
obj-$(CONFIG_I2C_SIMTEC)            += i2c-simtec.o  
obj-$(CONFIG_I2C_SIRF)              += i2c-sirf.o
```

Linux Kernel Booting



Linux Kernel Booting

► Bootloader run kernel with parameters

- $x0 = \text{DTB}$ (Device Tree Blob)
- $r1 = \text{NULL}$
- $r2 = \text{NULL}$

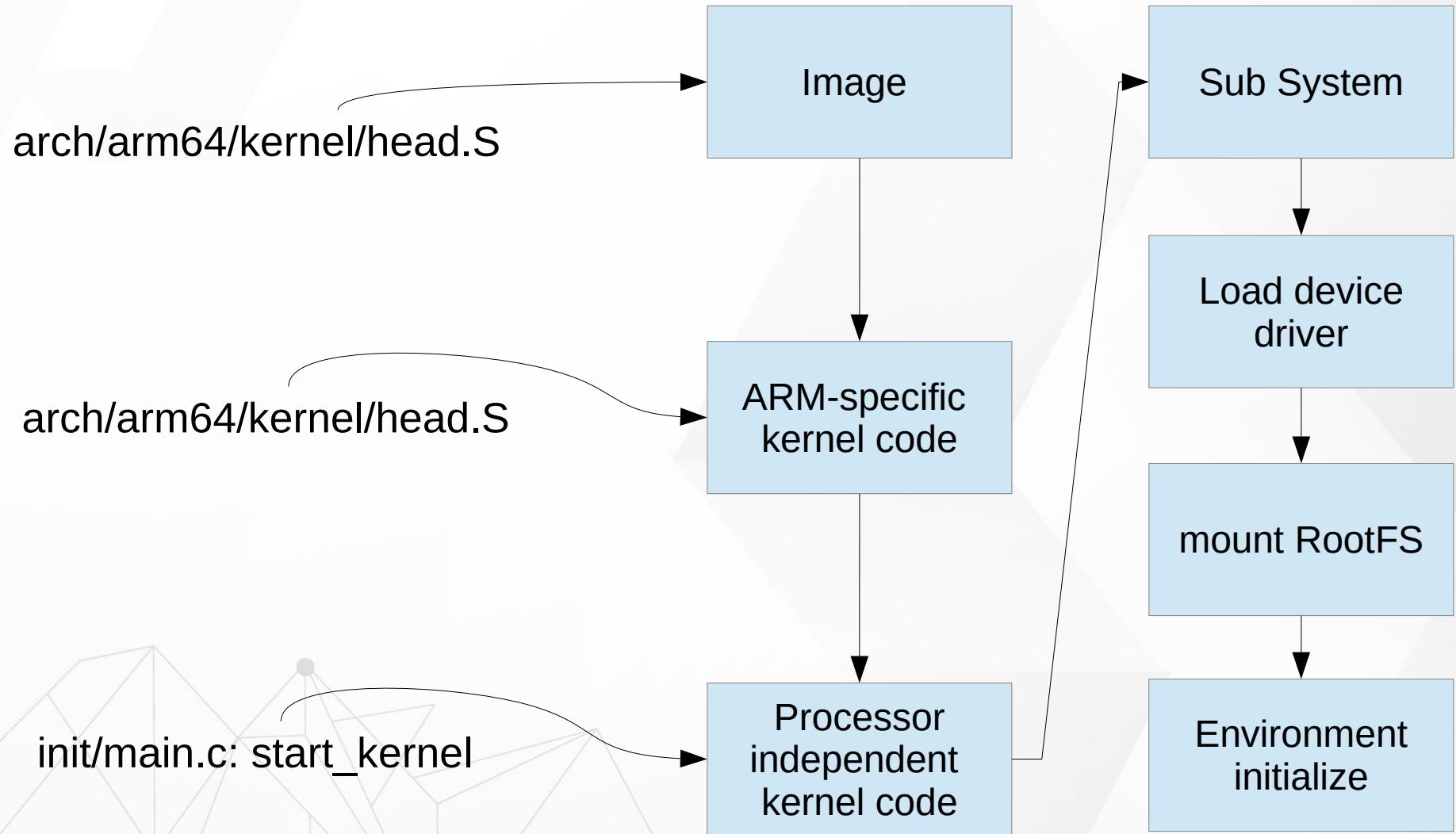


Kernel Startup Entry Point

arch/arm64/kernel/head.S

```
ENTRY(stext)
    bl→preserve_boot_args
    bl→el2_setup → → → // Drop to EL1, w20=cpu_boot_mode
    adrp → x24, __PHYS_OFFSET
    and x23, x24, MIN_KIMG_ALIGN - 1 → // KASLR offset, defaults to 0
    bl→set_cpu_boot_mode_flag
    bl→__create_page_tables → → → // x25=TTBR0, x26=TTBR1
    /*
     * The following calls CPU setup code, see arch/arm64/mm/proc.S for
     * details.
     * On return, the CPU will be ready for the MMU to be turned on and
     * the TCR will have been set.
     */
    bl→__cpu_setup → → → // initialise processor
    adr_l → x27, __primary_switch → → → // address to jump to after
    → → → → → → // MMU has been enabled
    b → __enable_mmu
ENDPROC(stext)
```

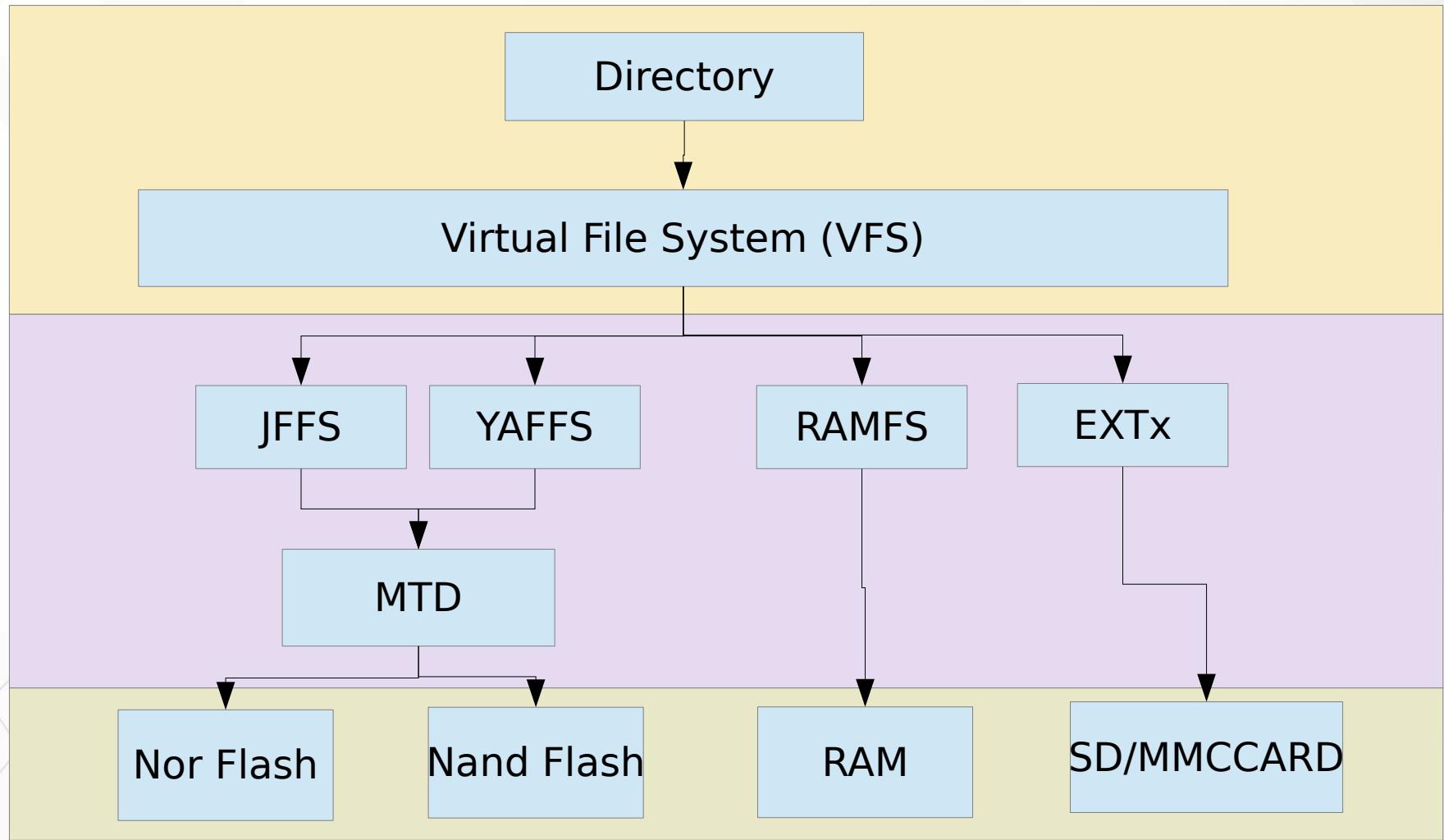
Linux Kernel Booting



`init/main.c: start_kernel`

CH 7 Linux Root File System

File System in Linux





File System

- ▶ A file system defines how files are **named, stored, and retrieved** from a storage device
- ▶ For desktop users
 - FAT32, NFS, EXT3, EXT4
- ▶ For embedded system
 - Cramfs
 - JFFS2
 - Squashfs
 - YAFFS2
 - EXT2, EXT3



File System and Kernel

► [CMD] make menuconfig

- File systems

```
[ ] Second extended fs support (NEW)
[ ] The Extended 3 (ext3) filesystem (NEW)
[ ] The Extended 4 (ext4) filesystem (NEW)
[ ] Reiserfs support (NEW)
[ ] JFS filesystem support (NEW)
[ ] XFS filesystem support (NEW)
[ ] GFS2 file system support (NEW)
[ ] Btrfs filesystem support (NEW)
[ ] NILFS2 file system support (NEW)
[ ] F2FS filesystem support (NEW)
[ ] Direct Access (DAX) support (NEW)
[ ] FS Encryption (Per-file encryption) (NEW)
[*] Dnotify support (NEW)
[*] Inotify support for userspace (NEW)
[ ] Filesystem wide access notification (NEW)
[ ] Quota support (NEW)
[ ] Kernel automounter version 4 support (also supports v3) (NEW)
[ ] FUSE (Filesystem in Userspace) support (NEW)
```



Mount a File System Driver

- ▶ Make sure which File-System be supported
 - [CMD] cat /proc/filesystems

```
rock@rockpi4b:~$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev ramfs
nodev bdev
nodev proc
nodev cpuset
nodev cgroup
nodev cgroup2
nodev tmpfs
nodev devtmpfs
nodev configfs
nodev debugfs
nodev tracefs
nodev securityfs
nodev sockfs
nodev pipefs
nodev rpc_pipefs
nodev devpts
ext3
ext2
ext4
squashfs
```



Mount a File System Driver

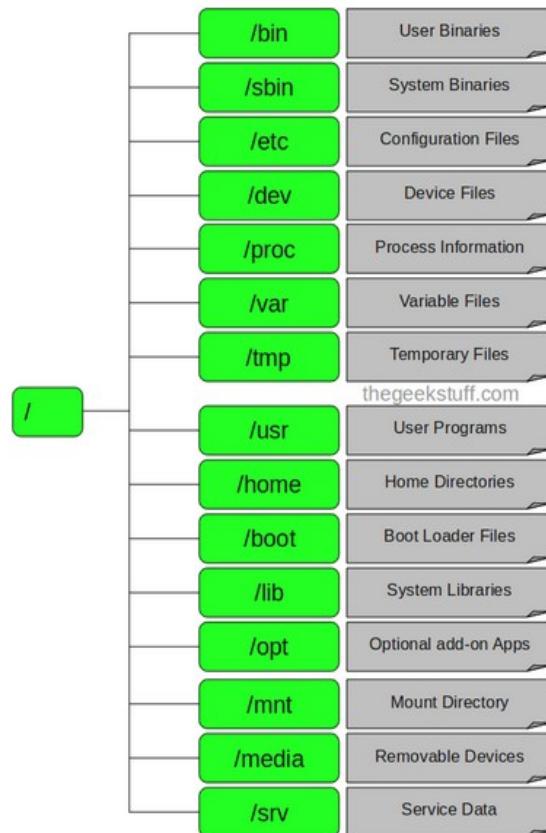
► Make sure which File-System be supported

- `mount -t ${FILE_SYS_TYPE} ${DISK} ${MOUNT_FOLDER}`
- [CMD] `sudo dd if=/dev/zero of=/home/rock/vdidk.img bs=1M count=5 status=progress`
- [CMD] `sudo mkfs.vfat ./vdidk.img`
- **[CMD] `sudo mount -t vfat ./vdidk.img /tmp/vfat_folder/`**
- [CMD] `lsblk -f`

Root File System



Root File System Structure





Root

- ▶ Every single file and directory starts from the root directory
- ▶ Only root user has write privilege under this directory
- ▶ Please note that /root is root user's home directory, which is not same as /



/bin – User Binaries

- ▶ Contains binary executables.
- ▶ Common linux commands you need to use in single-user modes are located under this directory.
- ▶ Commands used by all the users of the system are located here.
- ▶ For example: ps, ls, ping, grep, cp.



/sbin – System Binaries

- » Just like /bin, /sbin also contains binary executables.
- » But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- » For example: iptables, reboot, fdisk, ifconfig, swapon



/etc – Configuration Files

- ▶ Contains configuration files required by all programs.
- ▶ This also contains startup and shutdown shell scripts used to start/stop individual programs.
- ▶ For example: /etc/resolv.conf, /etc/logrotate.conf



/dev – Device Files

- ▶ Contains device files.
- ▶ These include terminal devices, usb, or any device attached to the system.
- ▶ For example: /dev/tty1, /dev/usbmon0



/proc – Process Information

- ▶ Contains information about system process.
- ▶ This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- ▶ This is a virtual filesystem with text information about system resources. For example: /proc/uptime



/var – Variable Files

- » var stands for variable files.
- » Content of the files that are expected to grow can be found under this directory.
- » This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);



/tmp – Temporary Files

- » Directory that contains temporary files created by system and users.
- » Files under this directory are deleted when system is rebooted.



/usr – User Programs

- ▶ Contains binaries, libraries, documentation, and source-code for second level programs.
- ▶ **/usr/bin** contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- ▶ **/usr/sbin** contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- ▶ **/usr/lib** contains libraries for /usr/bin and /usr/sbin
- ▶ **/usr/local** contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2



/home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita



/boot – Boot Loader Files

- ▶ Contains boot loader related files.
- ▶ Kernel initrd, vmlinu, grub files are located under /boot



/lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either `ld*` or `lib*.so.*`



/opt – Optional add-on Applications

- » opt stands for optional.
- » Contains add-on applications from individual vendors.
- » add-on applications should be installed under either /opt/ or /opt/ sub-directory.



/mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

Linux System Initial Program



System V

- » Unix System Five (V)
- » AT&T developed
- » The first initial process is → init
 - PID=1
- » SystemV handles startup processes through shell scripts in /etc/init*
 - /etc/inittab
 - /etc/init.d/ and /etc/init.d/rcS



System V

➤ /etc/init.d/S*

- initial system script

➤ Start a Service

- [CMD] etc/init.d/S50sshd start
- [CMD] etc/init.d/S50sshd stop

```
[root@rk3399:/etc/init.d]# ls
S01logging      S22hdmion      S50link_iq          S80dnsmasq
S10init         S30dbus        S50sshd            S99input-event-daemon
S10udev         S40network     S50telnet           rcK
S20urandom      S41dhpcd       S50usbdevice        rcs
S21mountall.sh  S50launcher_  S66load_wifi_modules
```



System D

- » System Daemon
- » /sbin/init -> /lib/systemd/systemd
 - PID=1
- » SystemD is the new system that many distros are moving to
- » SystemD handles startup processes through .service files



System D Configure File

► The unit configuration files are loaded from a set of paths

- "/lib/systemd/system":
 - OS default configuration files
- "/etc/systemd/system":
 - system administrator configuration files
 - override the OS default
- "/run/systemd/system":
 - un-time generated configuration files
 - override the installed configuration files



Service Control

▶ `systemctl ${CTL} ${SERVICE}`

- [CMD] `systemctl enable ssh`
- [CMD] `systemctl status ssh`
- [CMD] `systemctl start ssh`
- [CMD] `systemctl stop ssh`

Linux Distribution

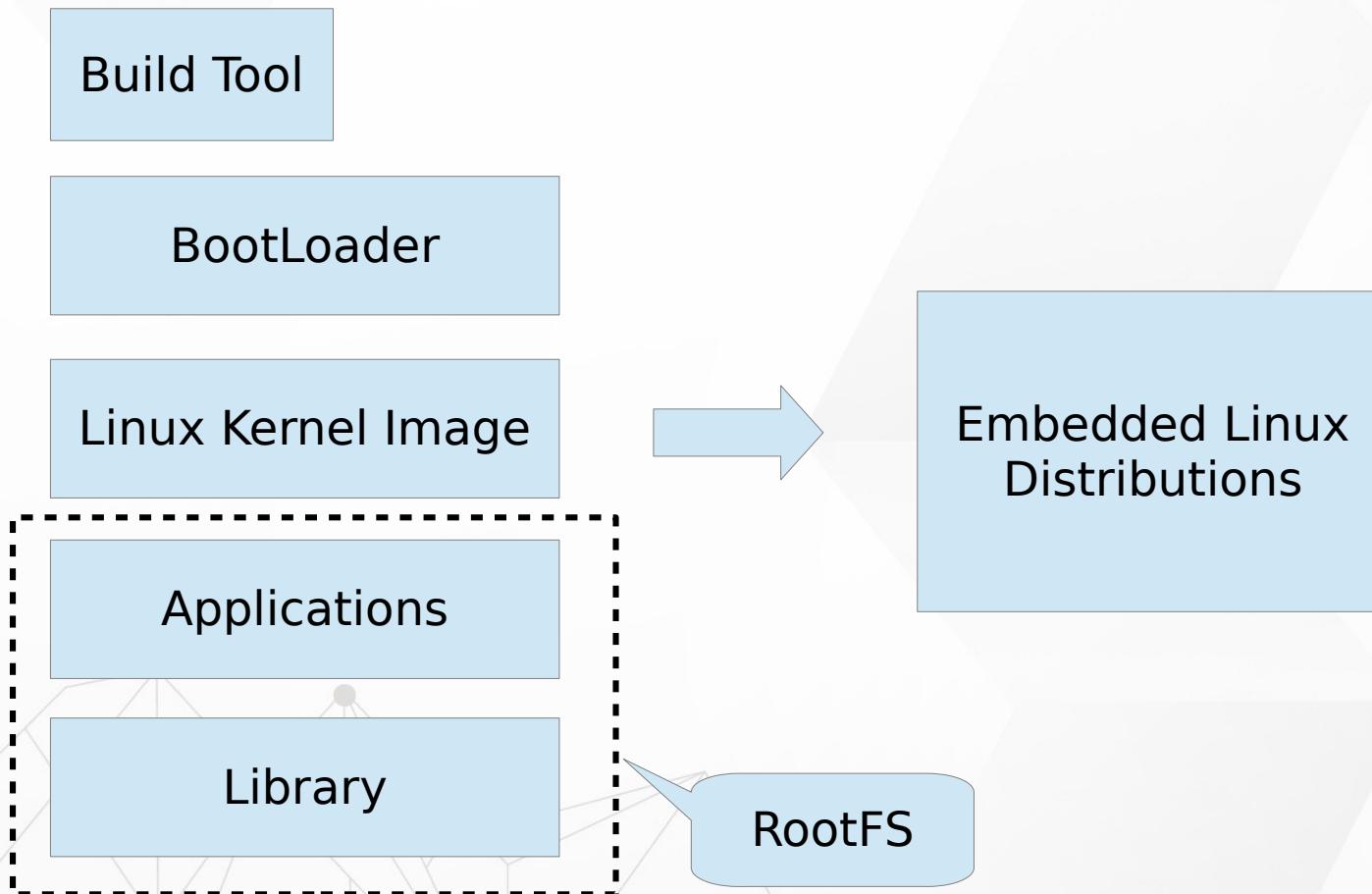


Linux Distribution

- ▶ Boot-loader
- ▶ Linux kernel
- ▶ RootFS
- ▶ Application
- ▶ Library
- ▶ Linux driver modules



Build Distribution by Tool





System integration

<https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf>

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well-defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

Debian



RockPi4B Debian

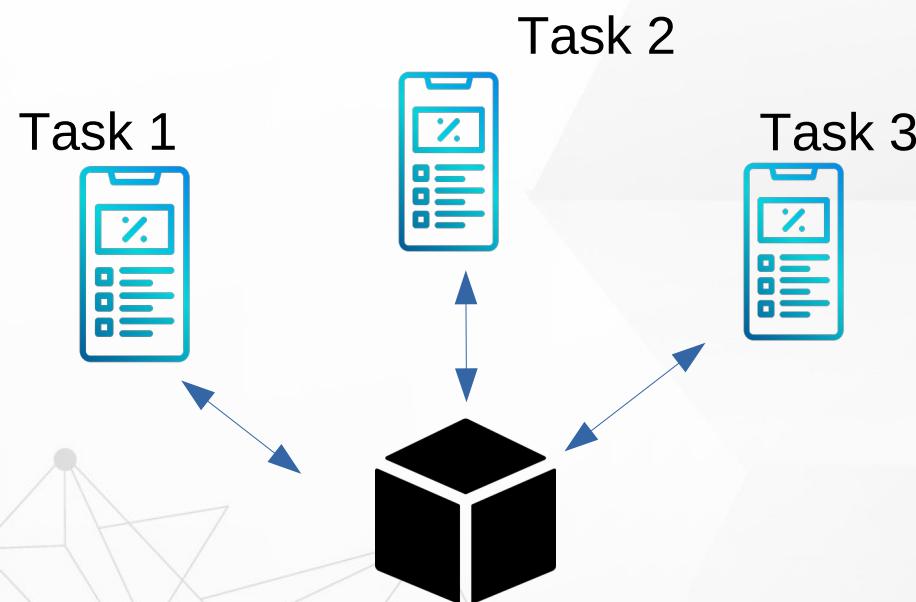
- ▶ <https://wiki.radxa.com/Rockpi4/dev/Debian>
- ▶ <https://github.com/radxa/rk-rootfs-build/tree/rockchip-debian>

CH9 Linux Device Driver

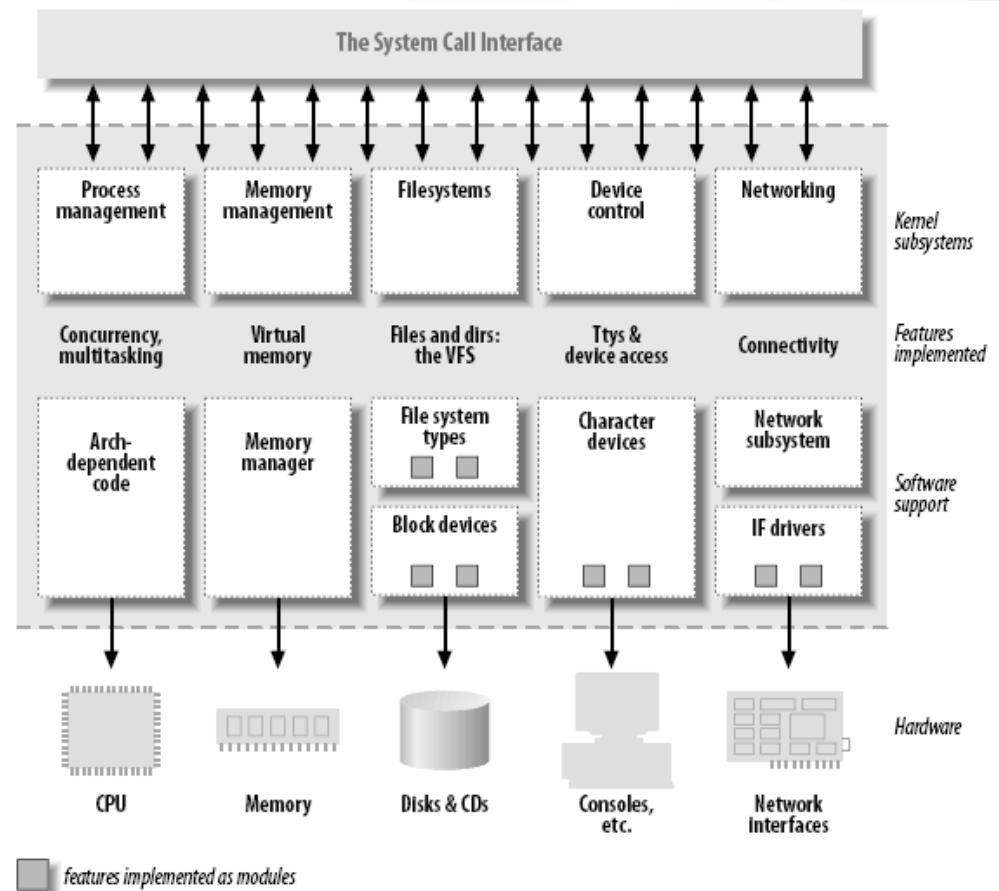
Introduction

▶ Device drivers

- Black boxes to hide details of hardware devices
- Use standardized calls



Kernel Modularization





Example

- ▶ [CMD] make
- ▶ [CMD] sudo insmod simple.ko
- ▶ [CMD] dmesg | tail
- ▶ [CMD] lsmod | grep simple
- ▶ [CMD] sudo rmmod simple



Classes of Devices Driver

▶ Char module

- simple
- access stream of bytes

▶ Block module

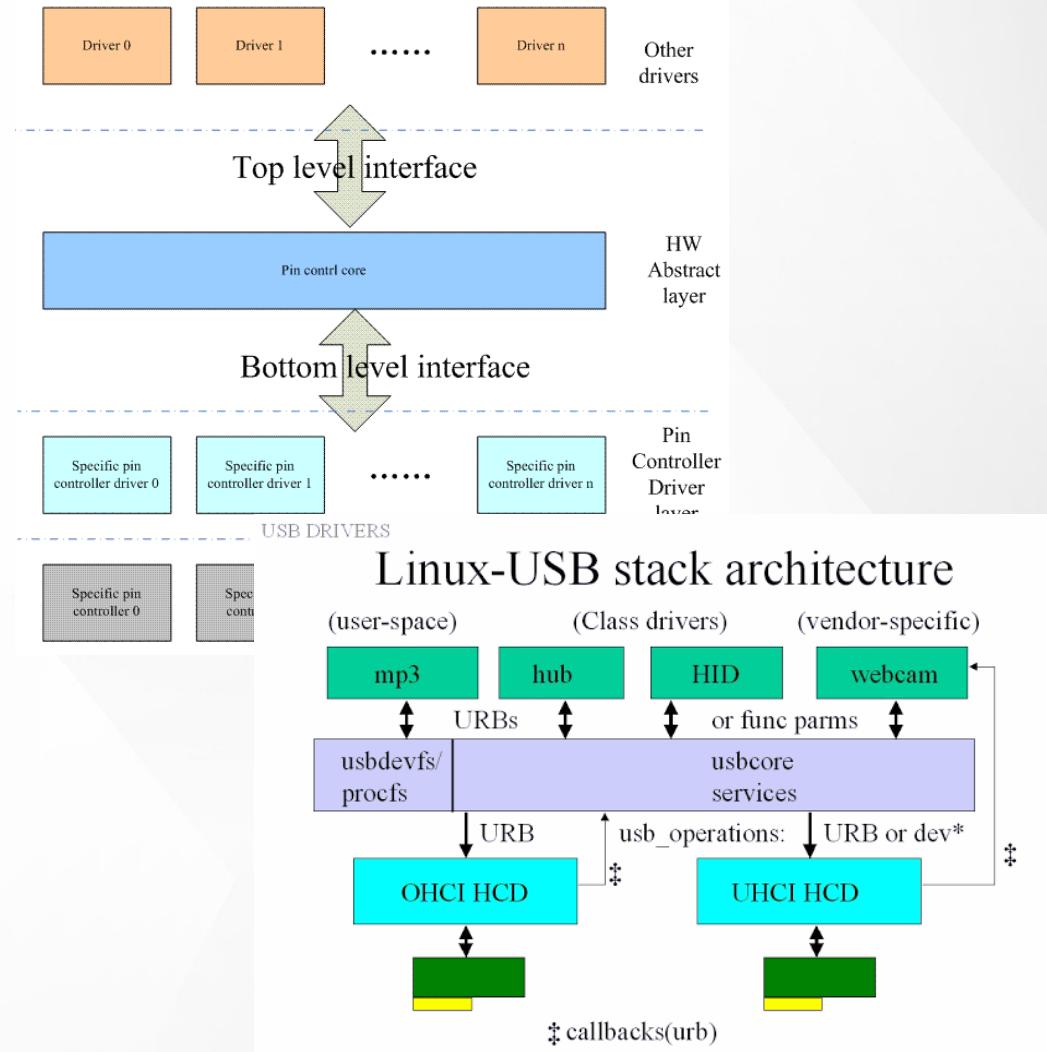
- block and char devices differ only in the way data is managed internally by the kernel

▶ Network module

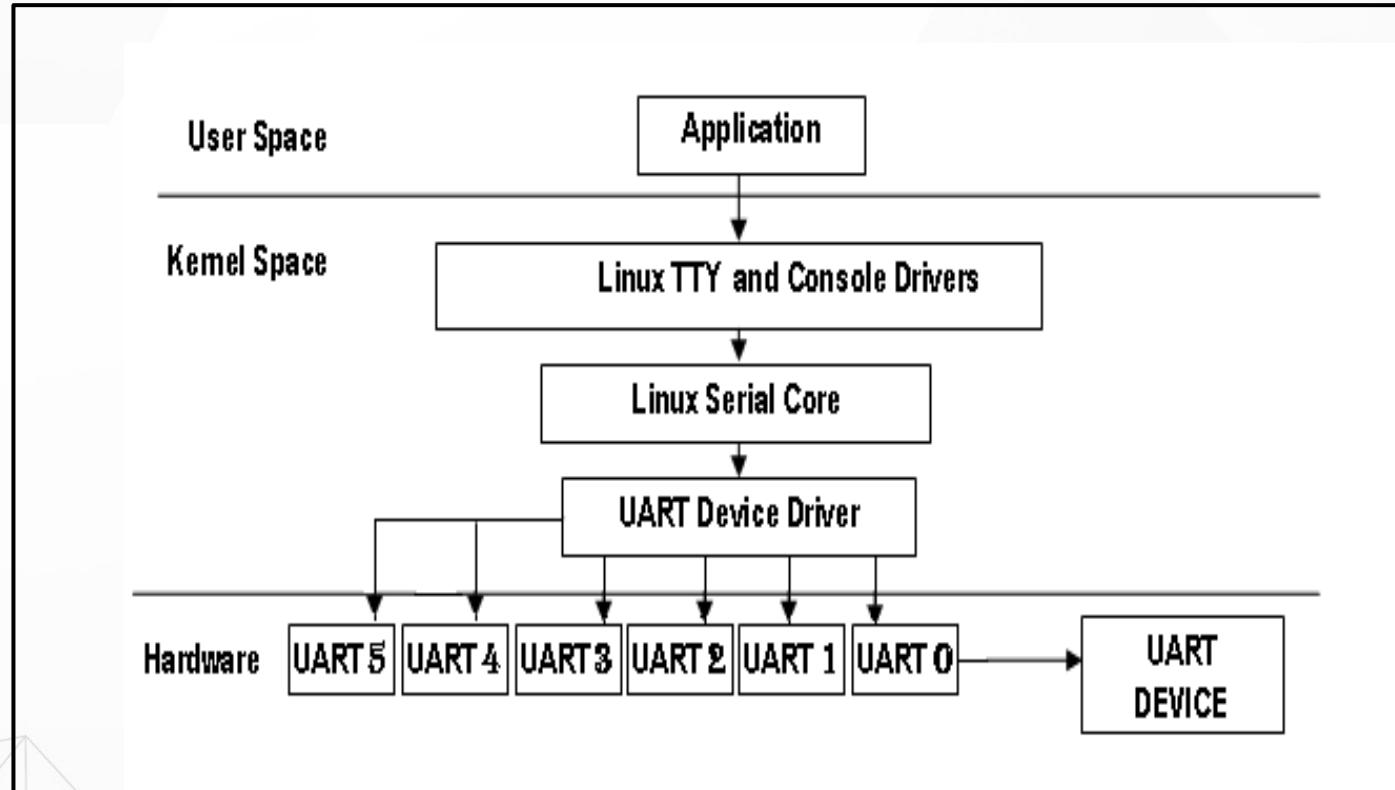
- Manage network data packets

Subsystem

- » DRM Subsystem
- » GPIO Subsystem
- » I2C Subsystem
- » SPI Subsystem
- » MTD Subsystem



TTY Sub-system





Where are Modules in Kernel

➤ \${KERNEL}/drivers

- \${KERNEL}/drivers/chars
- \${KERNEL}/drivers/i2c
- \${KERNEL}/drivers/gpio
- Module aliases for module loading utilities.

➤ Kernel build configure

- \${KERNEL}/.config

➤ Kconfig

- \${KERNEL}/drivers/chars/Kconfig

➤ [CMD] make menuconfig



Build Modules

▶ Build modules

→ [CMD] make modules

▶ Add install patch

→ [CMD] export **INSTALL_MOD_PATH**=./modules

▶ Install module to **INSTALL_MOD_PATH**

→ [CMD] make modules_install

→ Installs all modules in /lib/modules/<version>



Module Deploy

» modules_install

- `modules.alias` : Module aliases for module loading utilities.
- `modules.dep` : Module dependencies
- `modules.symbols` : Tells which module a given symbol



Install Module

▶ Install module

- \$ modprobe \${module_name}
- \$ insmod \${module_name}

▶ Remove module

- \$ modprobe -r \${module_name}
- \$ rmmod



modprobe depmod

► modprobe

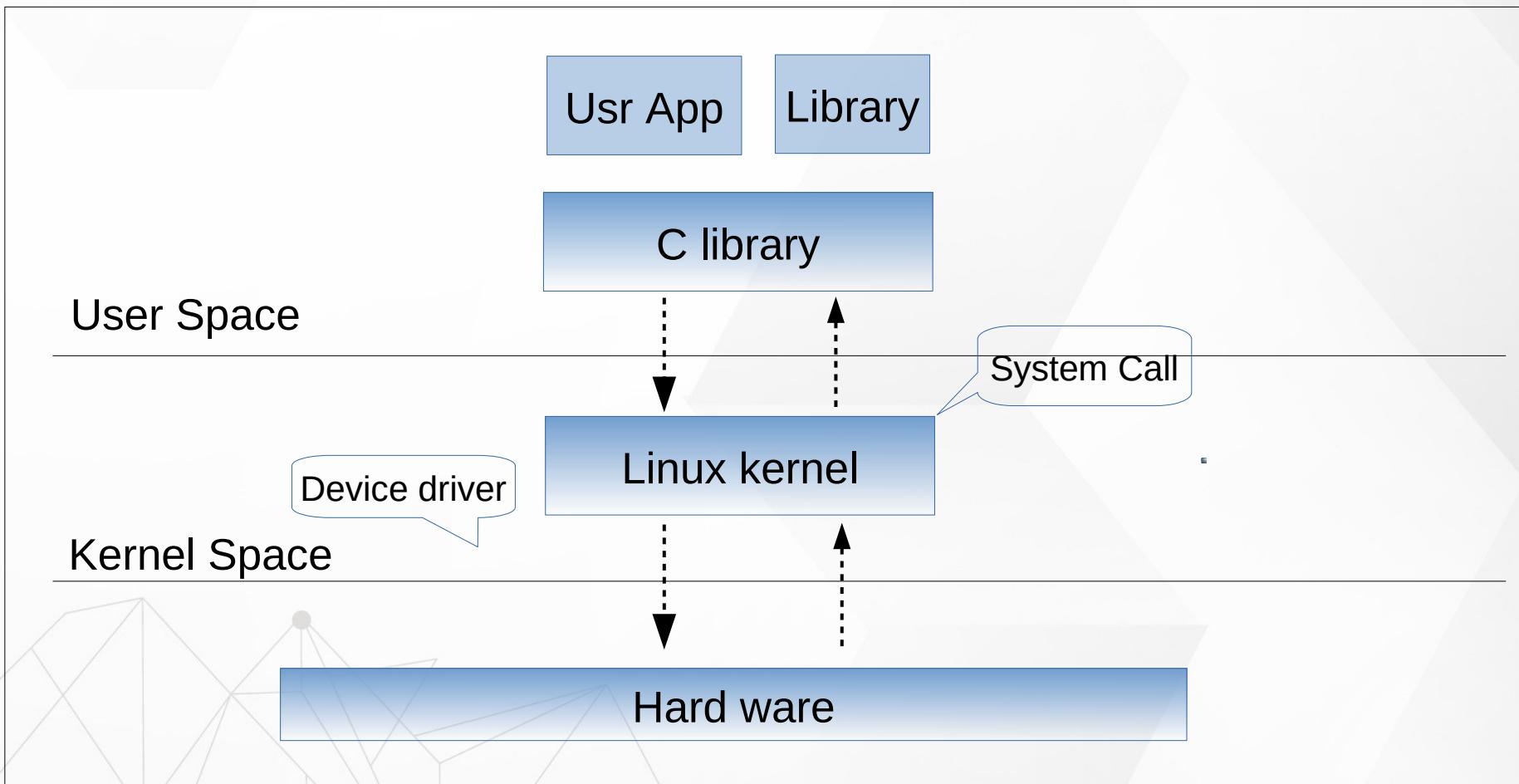
→ `/lib/modules/`uname -r``

► Depmod

→ creates a list of module dependencies `/lib/modules/version`

CH9 Linux User Land

Linux kernel





Sys Filesystem

- Allows kernel code to export information to user processes
- SysFS is an in-memory filesystem
- It provides two components
 - A kernel programming interface for exporting these items via sysfs
 - User interface to view and manipulate these items that maps back to the kernel objects which they represent



Sys File System

```
# tree -L 1 /sys/
```

```
/sys/
├── block
├── bus
├── class
├── dev
├── devices
├── firmware
├── fs
├── hypervisor
├── kernel
├── module
└── power
```

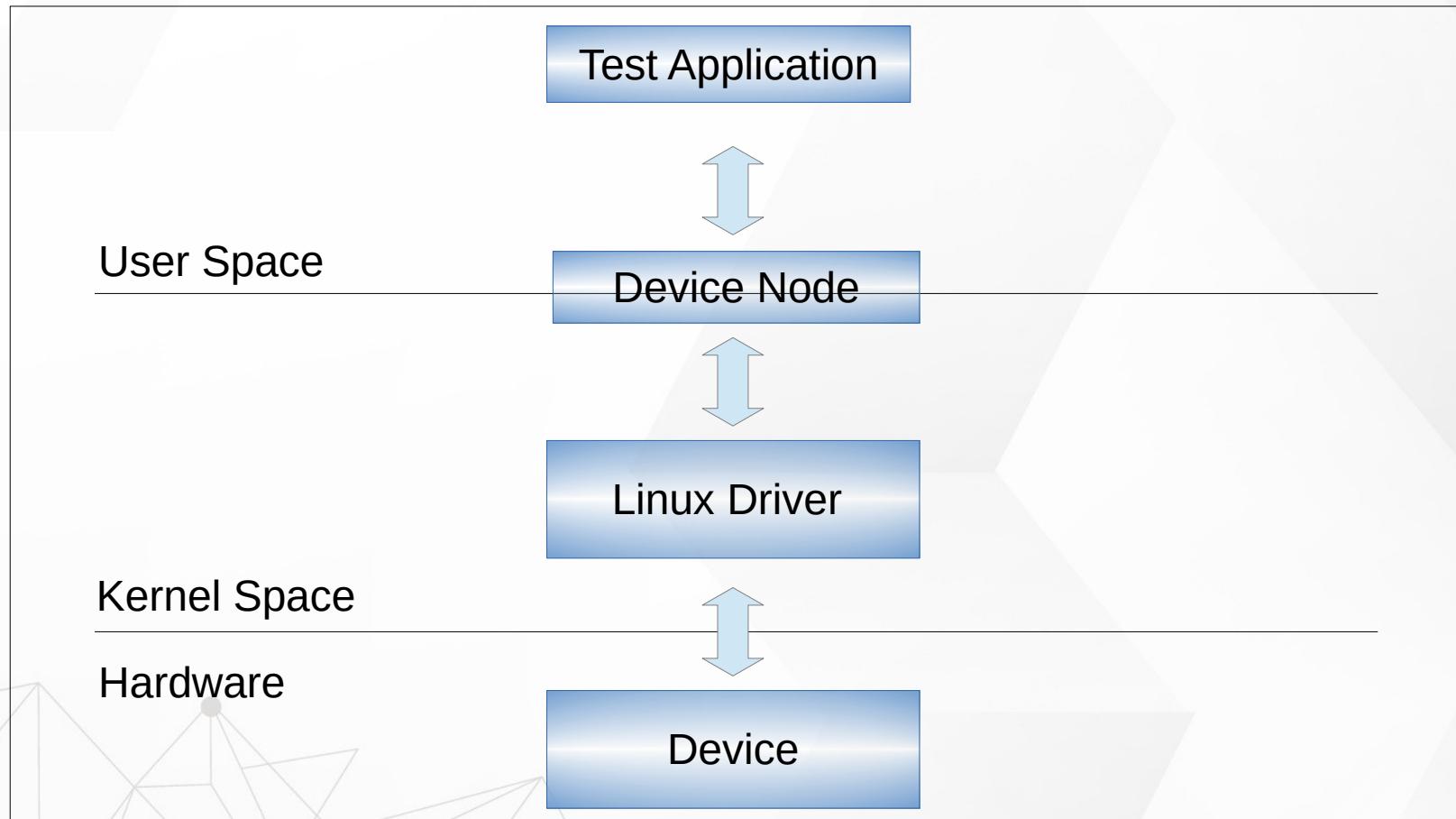
```
# tree -L 1 /sys/class/i2c-dev/i2c-0/
```

```
/sys/class/i2c-dev/i2c-0/
├── dev
├── device  -> ../../i2c-0
├── name
├── power
└── subsystem  -> ../../../../../../class/i2c-dev
    └── uevent
```

```
# tree -L 1 /sys/class/i2c-dev
```

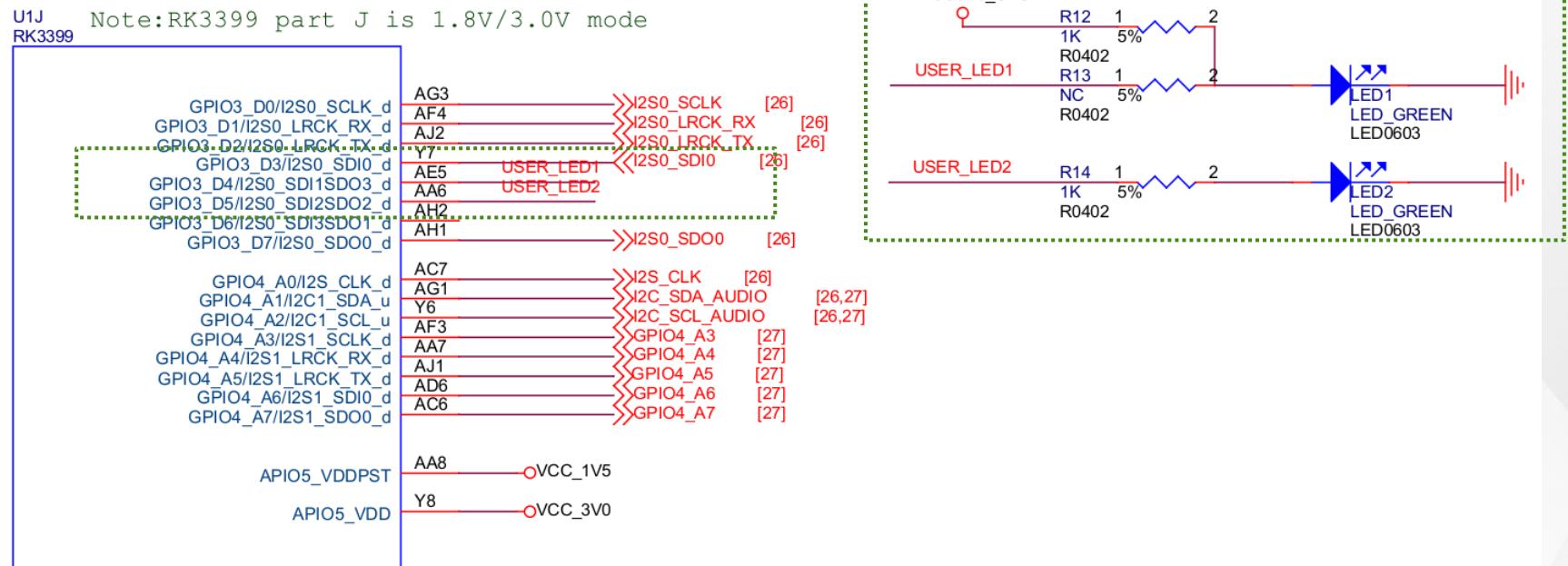
```
/sys/class/i2c-dev/
├── i2c-0  -> ../../devices/pci0000:00/0000:00:02.0/i2c-0/i2c-dev/i2c-0
├── i2c-1  -> ../../devices/pci0000:00/0000:00:02.0/i2c-1/i2c-dev/i2c-1
├── i2c-2  -> ../../devices/pci0000:00/0000:00:02.0/i2c-2/i2c-dev/i2c-2
├── i2c-3  -> ../../devices/pci0000:00/0000:00:02.0/i2c-3/i2c-dev/i2c-3
├── i2c-4  -> ../../devices/pci0000:00/0000:00:02.0/i2c-4/i2c-dev/i2c-4
├── i2c-5  -> ../../devices/pci0000:00/0000:00:02.0/i2c-5/i2c-dev/i2c-5
├── i2c-6  -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-1/i2c-6/i2c-dev/i2c-6
├── i2c-7  -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-2/i2c-7/i2c-dev/i2c-7
└── i2c-8  -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-3/i2c-8/i2c-dev/i2c-8
```

User land and Driver



LED Drivers

LED Schematic



GPIO3_D4/I2S0_SDI1SDO3_d → LED1

GPIO3_D5/I2S0_SDI2SDO2_d → LED2



LED Subsystem

▶ Control LED convenient with SysFS

▶ For example

- echo 1 > /sys/class/leds/user-led2/shot

▶ Switch different LED trigger type in SysFS

▶ For example

- echo "gpio" > /sys/class/leds/user-led2/trigger
- echo "1" > /sys/class/leds/user-led2/brightness
- echo "0" > /sys/class/leds/user-led2/brightness

LED SysFS

```
root@rockpi4b:/sys/class/leds/user-led2# ls -l  
  
brightness  
device -> ../../../../../gpio-leds  
max_brightness  
power  
subsystem -> ../../../../../../class/leds  
trigger  
uevent
```

Check trigger type

```
root@rockpi4b:/sys/class/leds/user-led2# cat trigger  
none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock  
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock  
kbd-shiftllock kbd-shiftrlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1  
timer oneshot heartbeat backlight [gpio] cpu0 cpu1 cpu2 cpu3  
cpu4 cpu5
```

Switch trigger type

```
root@rockpi4b:/sys/class/leds/user-led2# echo heartbeat > trigger
```

```
root@rockpi4b:/sys/class/leds/user-led2# cat trigger  
none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock  
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock  
kbd-shiftllock kbd-shiftrlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1  
timer oneshot [heartbeat] backlight gpio cpu0 cpu1 cpu2 cpu3  
cpu4 cpu5
```

9

GPIO Control



Driver LED in User Space

» Paths in Sysfs

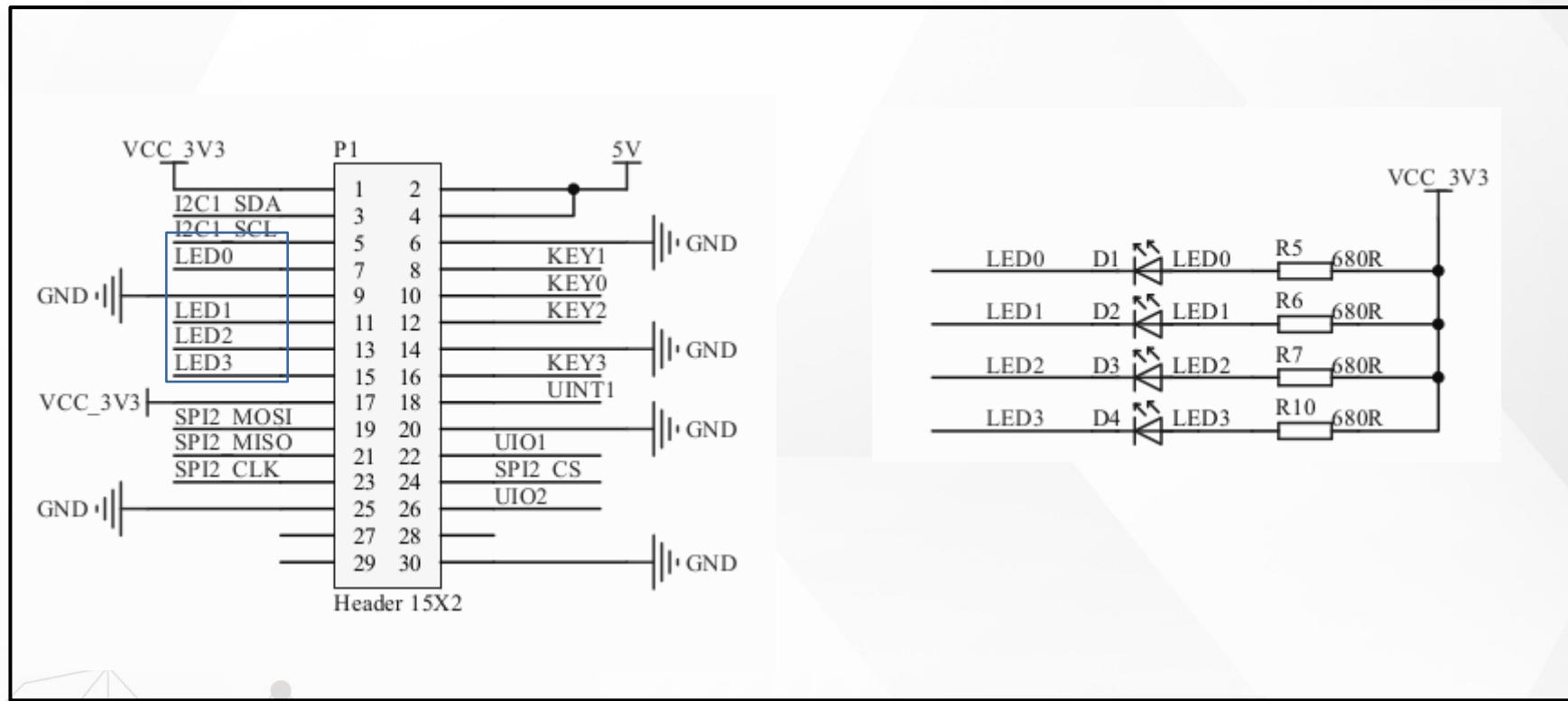
» /sys/class/gpio:

- Control interfaces used to get userspace control over GPIOs;
- GPIOs themselves
- GPIO controllers("gpio_chip" instances)

» /sys/class/gpio/

- "export" : ask the kernel to export GPIO to userspace by writing
 - "echo 19 > export"
 - create a "gpio19" node in /sys/class/gpio
- "unexport" : Reverses the effect of exporting to userspace
 - "echo 19 > unexport"
 - remove "gpio19" node from /sys/class/gpio

Cadtc Ext Board LED



RockPi4B HEAD

Rock Pi 4 A/B/C general purpose input-output (GPIO) connector

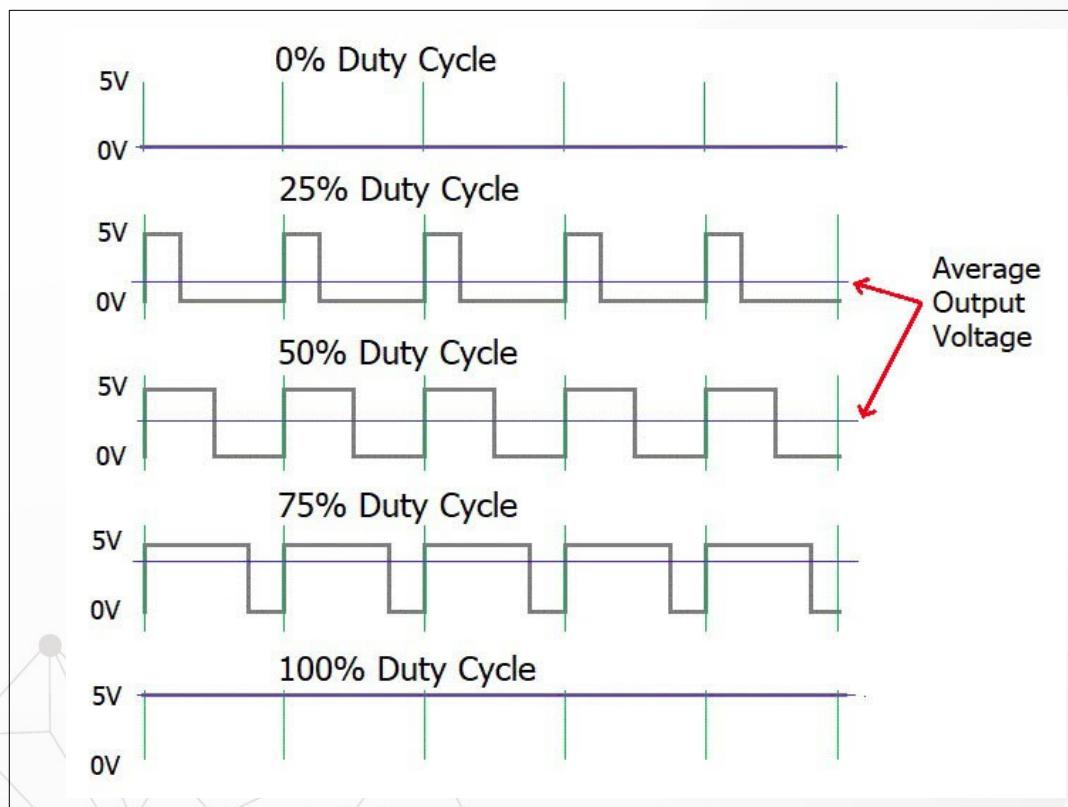
ROCK Pi 4 has a 40-pin expansion header. Each pin is distinguished by color.

GPIO number	Function2	Function1	GPIO	Pin#	Pin#	GPIO	Function1	Function2	GPIO number
		+3.3V		1	2		+5.0V		
71		I2C7_SDA	GPIO2_A7	3	4		+5.0V		
72		I2C7_SCL	GPIO2_B0	5	6		GND		
75		SPI2_CLK	GPIO2_B3	7	8	GPIO4_C4	UART2_TXD		148
		GND		9	10	GPIO4_C3	UART2_RXD		147
146		PWM0	GPIO4_C2	11	12	GPIO4_A3	I2S1_SCLK		131
150		PWM1	GPIO4_C6	13	14		GND		
149		SPDIF_TX	GPIO4_C5	15	16	GPIO4_D2			154
		+3.3V		17	18	GPIO4_D4			156
40	UART4_TXD	SPI1_TXD	GPIO1_B0	19	20		GND		
39	UART4_RXD	SPI1_RXD	GPIO1_A7	21	22	GPIO4_D5			157
41		SPI1_CLK	GPIO1_B1	23	24	GPIO1_B2	SPI1_CSn		42
		GND		25	26		ADC_IN0		
64		I2C2_SDA	GPIO2_A0	27	28	GPIO2_A1	I2C2_CLK		65
74	I2C6_SCL	SPI2_TXD	GPIO2_B2	29	30		GND		
73	I2C6_SDA	SPI2_RXD	GPIO2_B1	31	32	GPIO3_C0	SPDIF_TX	UART3_CTSn	112
76		SPI2_CSn	GPIO2_B4	33	34		GND		
133		I2S1_LRCK_TX	GPIO4_A5	35	36	GPIO4_A4	I2S1_LRCK_RX		132
158			GPIO4_D6	37	38	GPIO4_A6	I2S1_SD		134
		GND		39	40	GPIO4_A7	I2S1_SDO		135

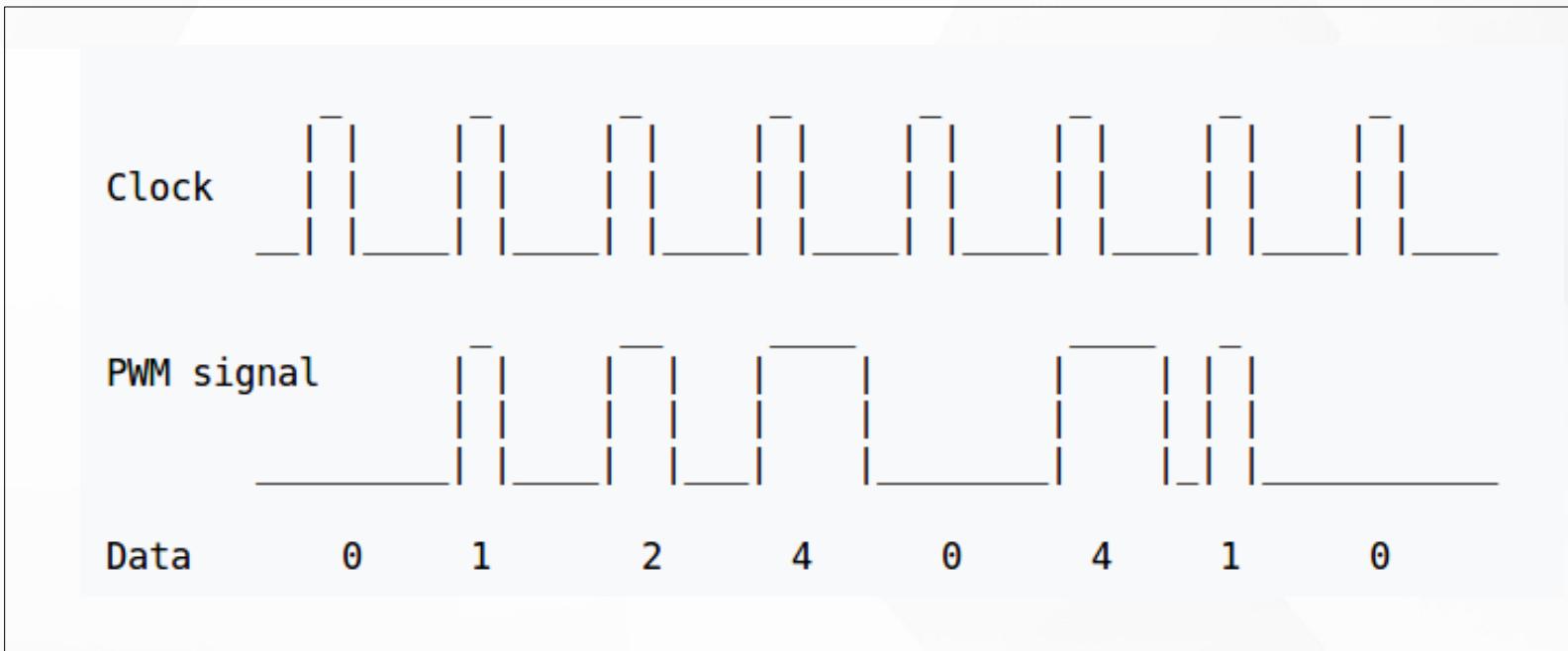
PWM Sub System

PWM

▶ PWM : Pulse Width Modulation



PWM



https://en.wikipedia.org/wiki/Pulse-width_modulation



PWM Parameter in Linux

▶ Period

- The total period of the PWM signal
- Value is in nanoseconds
- sum of the active and inactive time of the PWM

▶ duty_cycle

- The active time of the PWM signal
- Value is in nanoseconds
- must be less than the period.



PWM Parameter in Linux

► Polarity

- The polarity of the PWM signal

► Enable

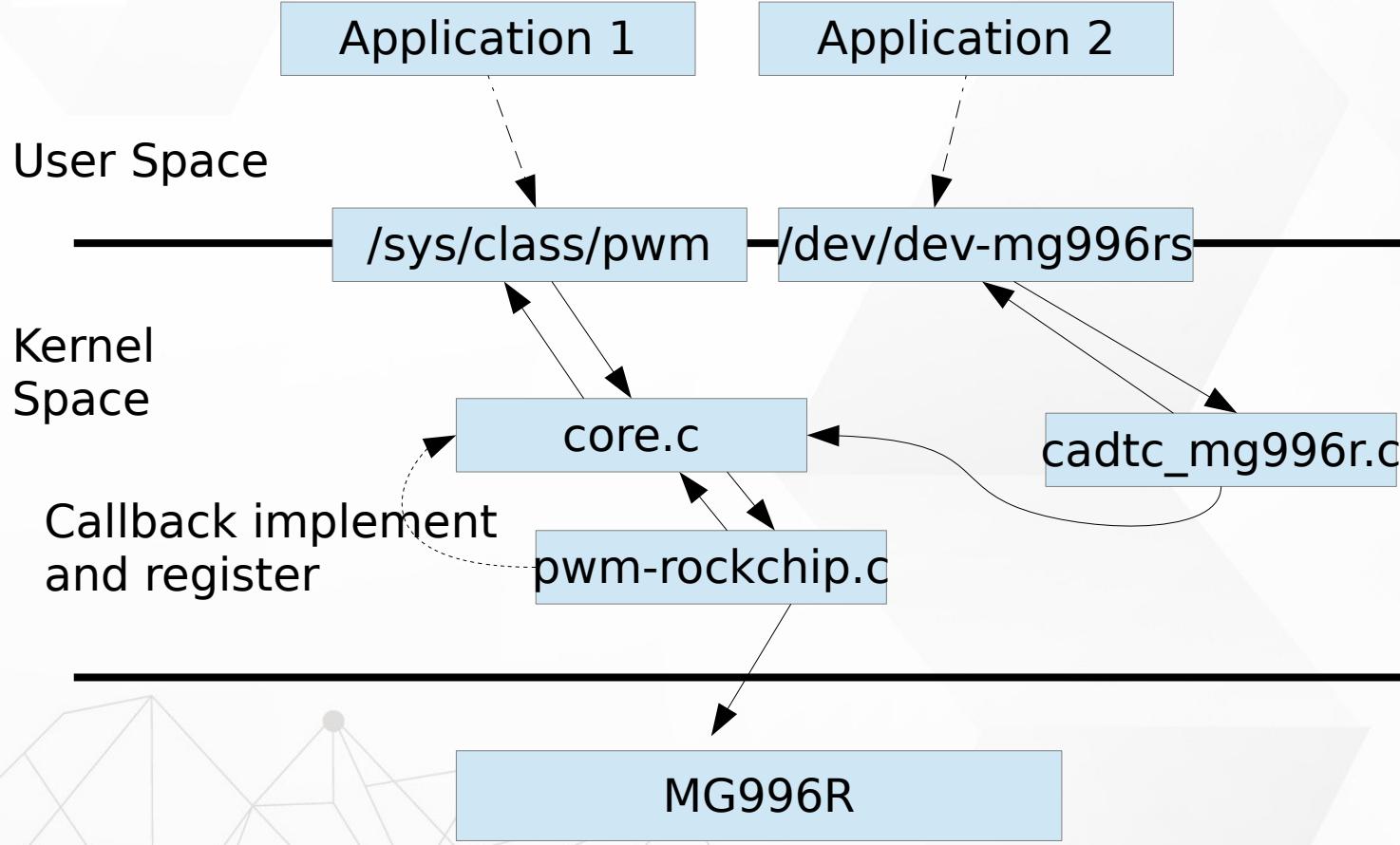


PWM Driver

- \$(KERNEL_SRC)/Documentation/pwm.txt
- Platform Driver

- drivers/pwm/
- drivers/pwm/core.c
- drivers/pwm/pwm-rockchip.c

PWM Subsystem



RockPi4B HEAD

Rock Pi 4 A/B/C general purpose input-output (GPIO) connector

ROCK Pi 4 has a 40-pin expansion header. Each pin is distinguished by color.

GPIO number	Function2	Function1	GPIO	Pin#	Pin#	GPIO	Function1	Function2	GPIO number
		+3.3V		1	2		+5.0V		
71		I2C7_SDA	GPIO2_A7	3	4		+5.0V		
72		I2C7_SCL	GPIO2_B0	5	6		GND		
75		SPI2_CLK	GPIO2_B3	7	8	GPIO4_C4	UART2_TXD		148
		GND		9	10	GPIO4_C3	UART2_RXD		147
146		PWM0	GPIO4_C2	11	12	GPIO4_A3	I2S1_SCLK		131
150		PWM1	GPIO4_C6	13	14		GND		
149		SPDIF_TX	GPIO4_C5	15	16	GPIO4_D2			154
		+3.3V		17	18	GPIO4_D4			156
40	UART4_TXD	SPI1_TXD	GPIO1_B0	19	20		GND		
39	UART4_RXD	SPI1_RXD	GPIO1_A7	21	22	GPIO4_D5			157
41		SPI1_CLK	GPIO1_B1	23	24	GPIO1_B2	SPI1_CSn		42
		GND		25	26		ADC_IN0		
64		I2C2_SDA	GPIO2_A0	27	28	GPIO2_A1	I2C2_CLK		65
74	I2C6_SCL	SPI2_TXD	GPIO2_B2	29	30		GND		
73	I2C6_SDA	SPI2_RXD	GPIO2_B1	31	32	GPIO3_C0	SPDIF_TX	UART3_CTSn	112
76		SPI2_CSn	GPIO2_B4	33	34		GND		
133		I2S1_LRCK_TX	GPIO4_A5	35	36	GPIO4_A4	I2S1_LRCK_RX		132
158			GPIO4_D6	37	38	GPIO4_A6	I2S1_SD		134
		GND		39	40	GPIO4_A7	I2S1_SDO		135



PWM SYSFS

```
/sys/class/pwm/pwmchip0
```

```
device  export  npwm    power    subsystem uevent  unexport
```

```
echo 0 > export
```

```
capture  enable   polarity  uevent  duty_cycle  period    power
```

```
echo "2000000" > period      //20ms, 50 Hz
```

```
echo "200000" > duty_cycle  //2ms
```

```
echo 1 > enable            //Enable
```

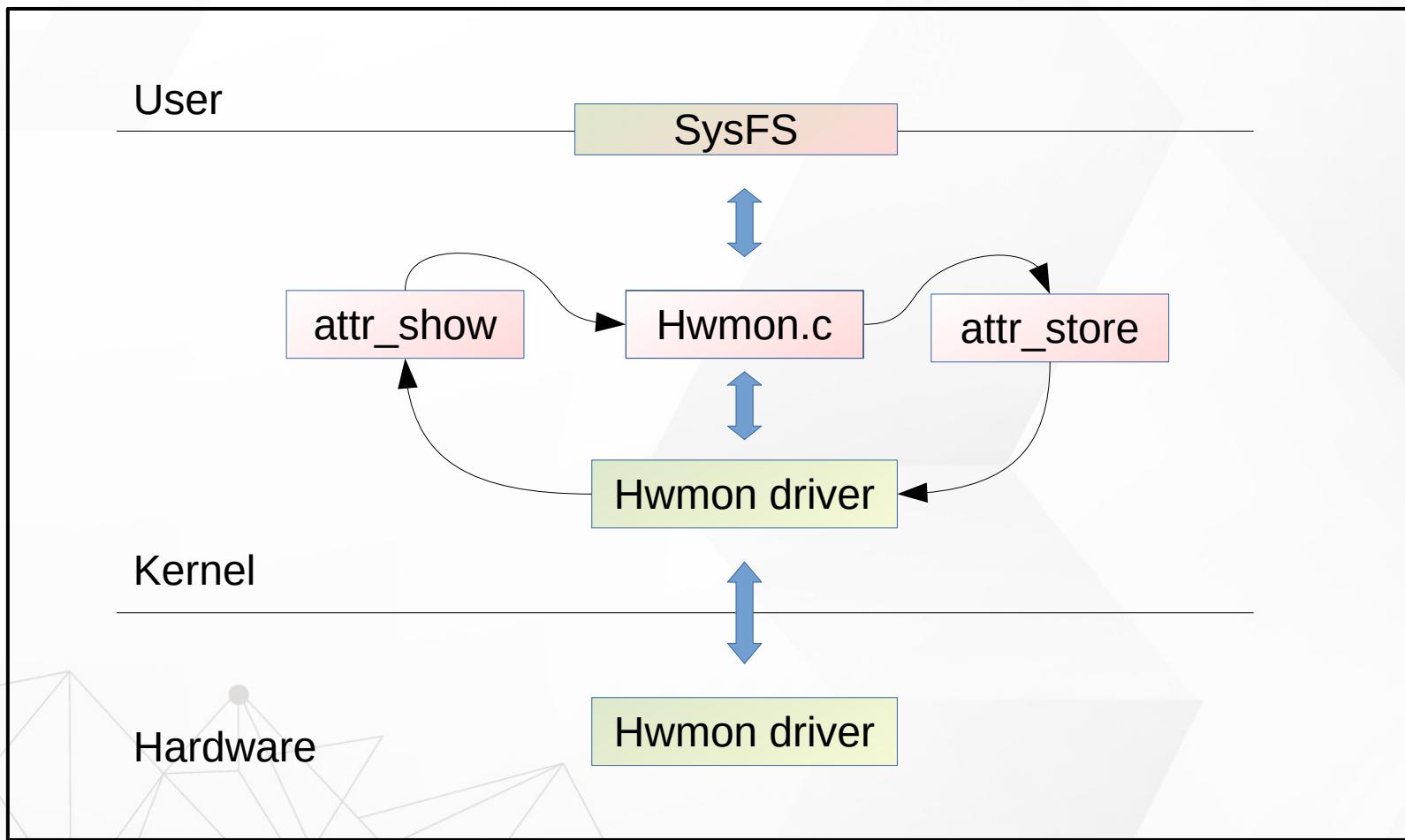


PWM DoReMi

	Frequency (Hz)
C4	261.63
C4#	277.18
D4	293.66
D4#	311.13
E4	329.63
F4	349.23
F4#	369.99
G4	392.00
G4#	415.30
A4	440.00
A4#	466.16
B4	493.88
C5	523.25

Hwmon Subsystem

Hwmon Subsystem

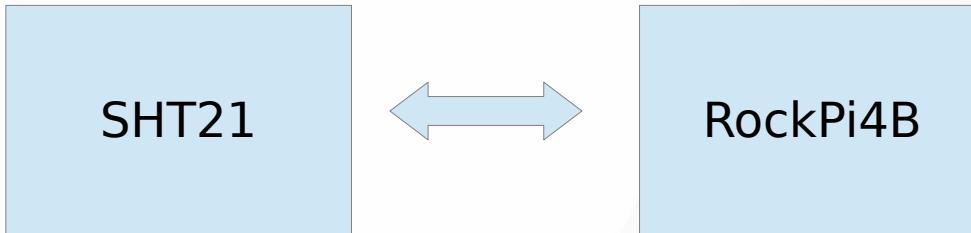




SHT21

- ▶ Simple interface
- ▶ Bus interface
 - I2C, GPIO, SPI
- ▶ Sensors
 - Temperature
 - Voltage
 - Humidity
 - Fan speed
 - PWM control

SHT21



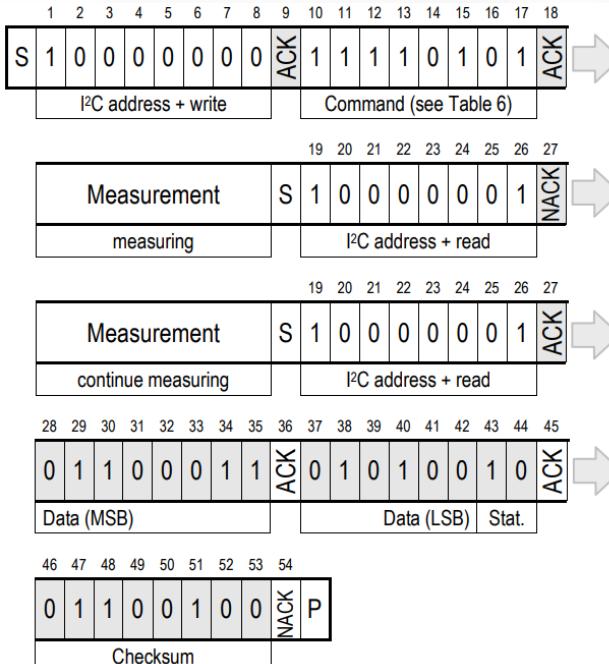
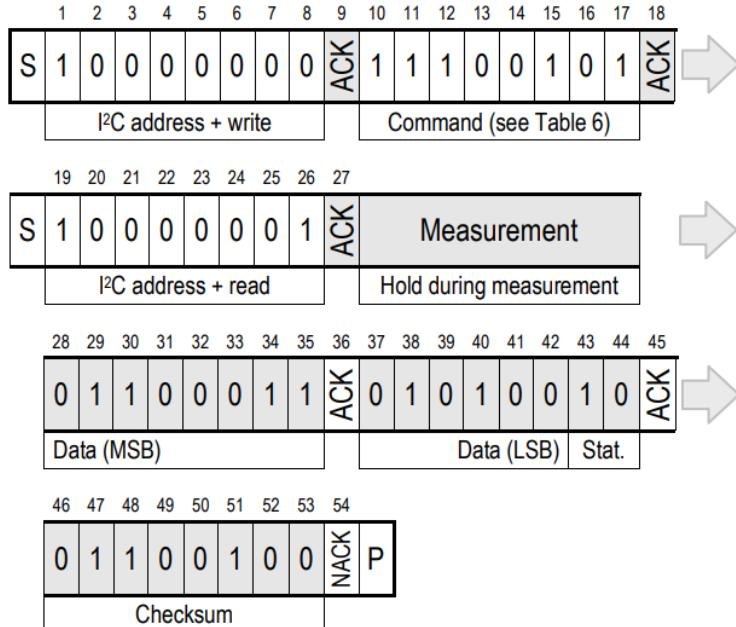
Pin	Name	Comment	
1	SDA	Serial Data, bidirectional	4
2	VSS	Ground	5
5	VDD	Supply Voltage	6
6	SCL	Serial Clock, bidirectional	3
3,4	NC	Not Connected	2
			1

A small gray rectangle represents the physical pin layout of the SHT21 sensor. The pins are numbered 1 through 6 around the perimeter. Pin 1 is at the top, pin 2 is at the bottom, pin 3 is at the right, pin 4 is at the top-left, pin 5 is at the bottom-left, and pin 6 is at the right.

Command	Comment	Code
Trigger T measurement	hold master	1110'0011
Trigger RH measurement	hold master	1110'0101
Trigger T measurement	no hold master	1111'0011
Trigger RH measurement	no hold master	1111'0101
Write user register		1110'0110
Read user register		1110'0111
Soft reset		1111'1110

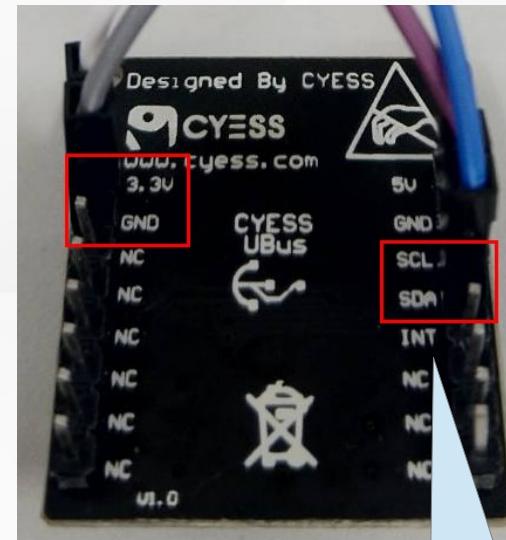
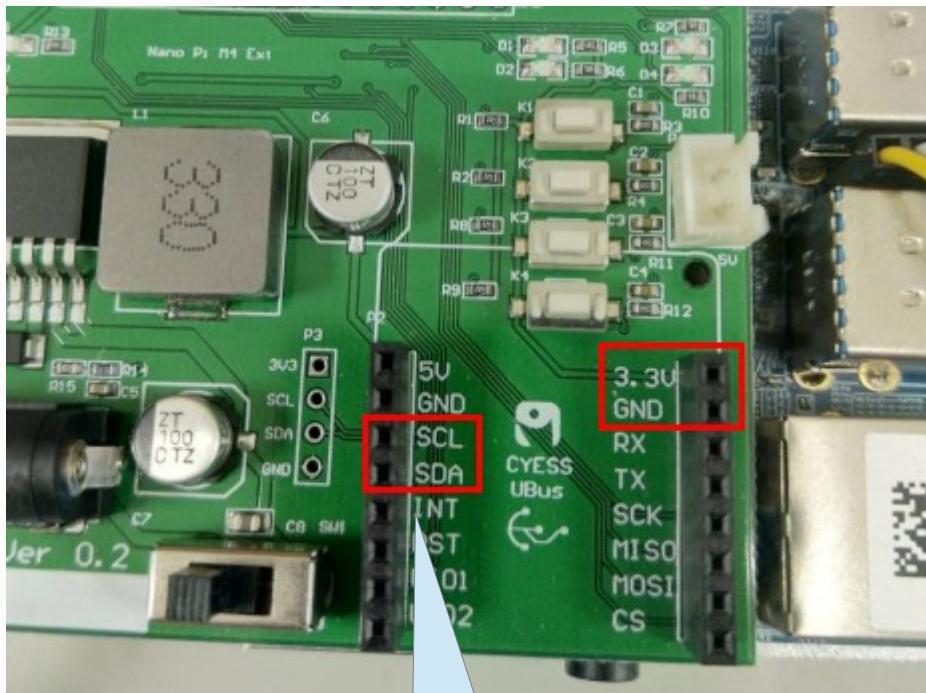
SHT21

Hold master communication sequence No Hold master communication sequence





SHT21





SHT21

Hwmon Sysfs

```
# ls /sys/class/hwmon/hwmon0
device          name    subsystem   uevent
humidity1_input power   temp1_input
```

temperature

```
# cat /sys/class/hwmon/hwmon0/temp1_input
32279
```

humidity

```
# cat /sys/class/hwmon/hwmon0/humidity1_input
34512
```

IIO Subsystem



IIO Introduction

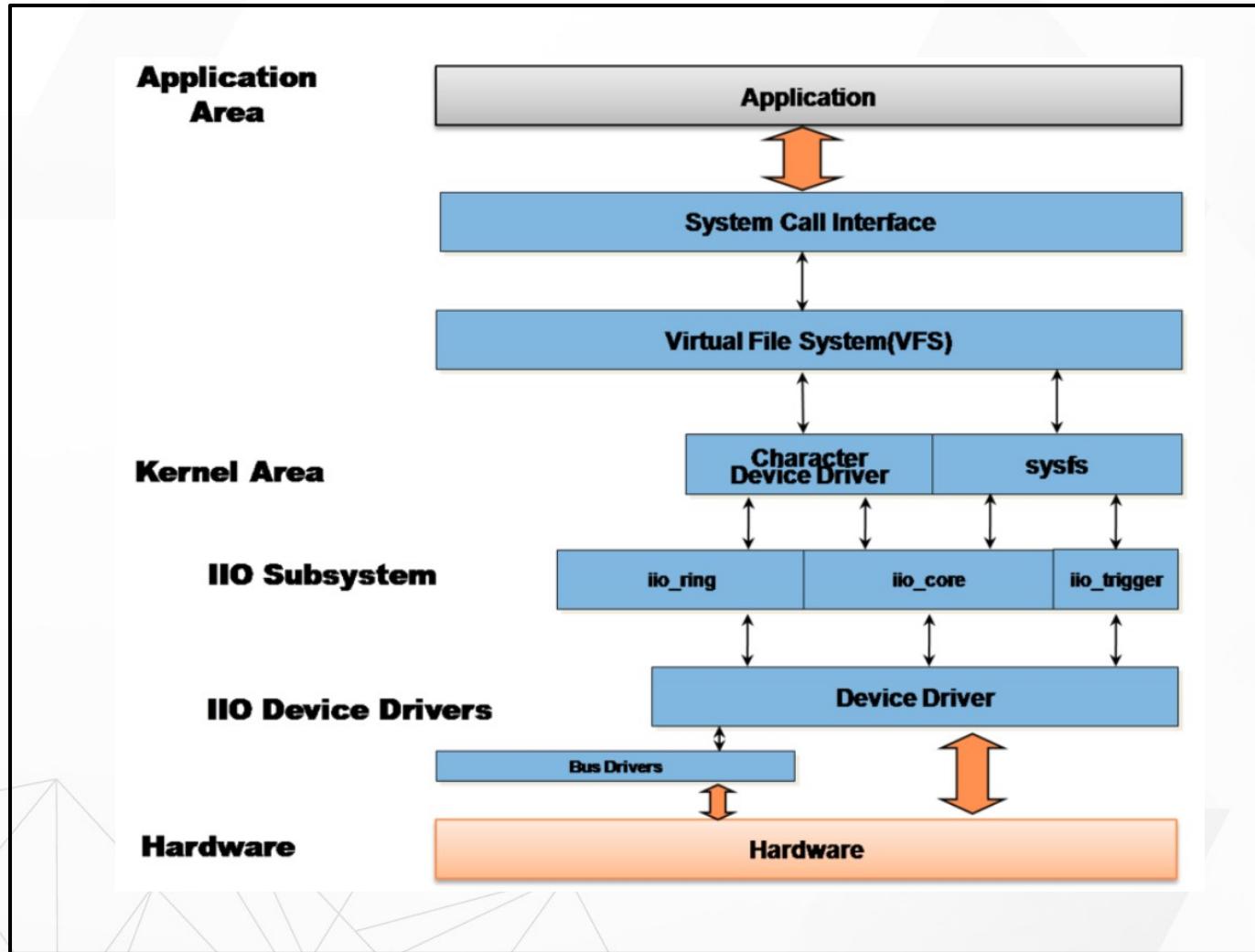
- ▶ IIO - The Industrial I/O
 - ▶ support for devices that in some sense
 - analog to digital (ADC)
 - digital to analog converters (DAC)
- ▶ Devices that fall into this category are
 - ADCs
 - Accelerometers
 - Gyros
 - DAC
 - Pressure Sensors



IIO Introduction

- Fill the gap between the somewhat similar hwmon and input subsystems
- Hwmon is very much directed at low sample rate sensors used in applications
 - fan speed control
 - temperature measurement.
- Input is, as it's name suggests focused on human interaction input devices

IIO Introduction





IIO Interface

➤ There are 2 ways for a user space application to interact with an IIO driver

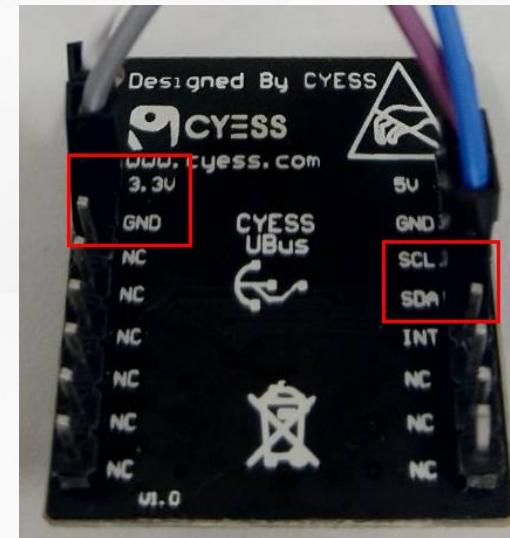
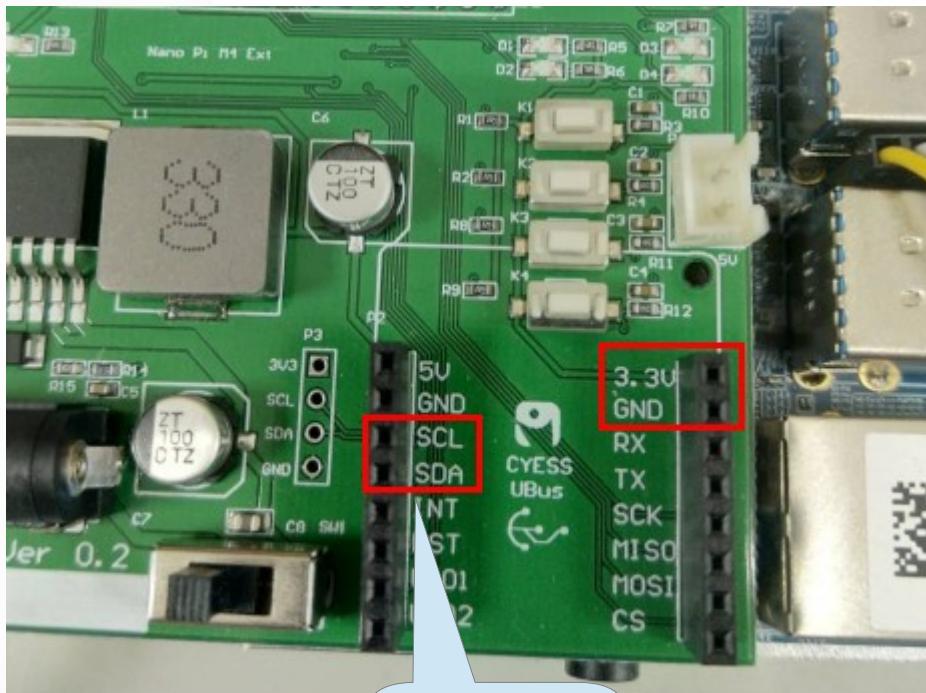
➤ **/sys/bus/iio/iio:deviceX/**

- data channels

➤ **/dev/iio:deviceX**

- buffered data transfer
- events information

ISL29023



I2C