

meta-riscv status — current status and call for contributions

Yocto Project Workshop at Embedded Recipes 2026

Michael Opdenacker

Root Commit

May 29, 2026



© 2024-2026 Root Commit. Licensed under CC BY-SA 4.0.



Pierre Fichoux

- One of the pioneers of embedded Linux in France
- In the last years, trained many new Yocto users in French speaking countries
- Regular speaker at Embedded Linux Conference, FOSDEM and in French conferences
- Also big contributor to real-time Linux, in particular Xenomai
- I miss you ❤️

Embedded Linux consultant and trainer

- <https://rootcommit.com/about/michael-opdenacker/>
- Former founder of [Bootlin](#)
- New founder of [Root Commit](#)
- Free Software enthusiast and advocate (member of [April.org](#))
- Serial guest of the [I Use Yocto BTW](#) podcast

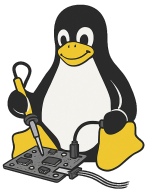


yoctopus



Consulting and engineering work

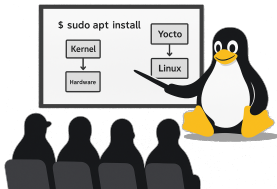
- Yocto Project
Project reviews, system implementation, new features
- Linux Kernel
Driver development, board bring-up, debugging
- Embedded Linux
Boot time, bug fixing, security and other optimizations



Training — <https://rootcommit.com/training>

- Yocto Project and OpenEmbedded — Free Materials!
- Linux kernel, board support, driver development
Free Materials after next course!
- Embedded Linux
- Linux Boot Time Reduction

What's special: focus on practical activities, interactivity and learning techniques.



- Support the growth of the RISC-V ecosystem
Yocto is a great enabler, especially for building the "boring" stuff (especially bootloader and firmware)
- Get more people to contribute to RISC-V and to `meta-riscv` in particular.
- Inspire contributors to other layers too.

DISCLAIMER: I'm not the maintainer

Special guests today:

- Yixun Lan, SpacemiT RISC-V SoC Linux kernel maintainer
- Drew Fustini, Tenstorrent and THead RISC-V SoC maintainer
- Khem Raj, maintainer of `meta-riscv` and many others!



Introduction

- ISA: *Instruction Set Architecture*
- Created by the University of California Berkeley, in a world dominated by proprietary ISAs with heavy royalties (ARM, x86)
- Exists in 32, 64 and 128 bit variants, from microcontrollers to powerful server hardware.
- Anyone can use and extend it to create their own SoCs and CPUs.
- This reduces costs and promotes reuse and collaboration
- Implementations can be proprietary (the majority) or open-source (minority so far)



See <https://en.wikipedia.org/wiki/RISC-V>

- Created in 2017 by Khem Raj, with the `qemuriscv64` machine
- 2018: Initial support for the *HiFive Unleashed development board* (`freedom-u540`) board
- 2019: `qemuriscv64` moved to OpenEmbedded Core
- 2021: Initial support for `beaglev-starlight-jh7100`
- 2022: New boards: *Nehza*, *VisionFive*, *Andes AE350*
- More and more boards
- 2025: RISC-V becomes a platinum member of Yocto. The platform is officially supported.
- 2025, 2026: a lot of activity on SpacemiT K1, K3 starting (first silicon with RVA23 profile)

```
$ git shortlog -sn
385 Khem Raj
163 Alistair Francis
59 Andreas Cord-Landwehr
38 Trevor Gamblin
30 Michael Opdenacker
25 Marcel Ziswiler
20 Alper Ak
15 Cezary Sobczak
14 Yu Chien Peter Lin
14 sungwon.pino
13 Mirza Krak
13 Pavel Zhukov
12 Koen Kooi
12 Maciej Pijanowski
11 Leonard Anderweit
8 Vlad Banea
7 Alexey Firago
7 Ricardo Salveti
6 GP Orcullo
6 Kanak Shilledar
5 Nick Owens
4 Moritz Fischer
4 RP38
4 Taras Kondratiuk
3 Andrei Lalaev
3 Sanjay Chitroda
3 Talel BELHAJSALEM
```

Full List of Supported Boards



OpenEmbedded Layer Index

Support agent

Login

meta-riscv



This is the general hardware specific BSP overlay for the RISC-V based devices.

More information can be found at: <https://riscv.org/> (Official Site)

[Setup information](#)

Git repository

• <https://github.com/riscv/meta-riscv> [View history](#)

Last commit: 1 week, 8 days ago (master branch)

Maintainer

• Khem Raj [Avatar](#)

Dependencies

The meta-riscv layer depends upon:

- [openembedded-core](#)

[Recipes](#) [Machines](#) [Appends](#) [Classes](#) [Updates](#)

meta-riscv machines (21)

ae350-ax45mp	AE350 Platform (AX45MP)
bananapi-f3	Banana Pi F3
baremetal-riscv32	a generic riscv32
baremetal-riscv32nf	a generic riscv32 without FP
baremetal-riscv64	a generic riscv64
beaglev-ahead	Beagle-V Ahead board
beaglev-starlight-jh7100	beta BeagleV Starlight board
eswin-ebc77	ESWIN Computing EBC77 with vendor Linux kernel
eswin-ebc77-mainline	ESWIN Computing EBC77 with mainline Linux kernel
freedom-u540	HiFive Unleashed development board
mangopi-mq-pro	MangoPi MQ Pro
milky-duo	Milk-V Duo
milky-megrez	Milk-V Megrez
muse-pi-pro	Muse Pi Pro with mainline Linux
nezha-allwinner-d1	Nezha Allwinner-d1 board
orangepi-r2s	OrangePi R2S
orangepi-rv2	OrangePi RV2
orangepi-rv2-mainline	OrangePi RV2 with mainline Linux
star64	Star64 board
visionfive	VisionFive board
visionfive2	VisionFive 2 board

ae350-ax45mp	AE350 Platform (AX45MP)
bananapi-f3	Banana Pi F3
baremetal-riscv32	a generic riscv32
baremetal-riscv32nf	a generic riscv32 without FP
baremetal-riscv64	a generic riscv64
beaglev-ahead	Beagle-V Ahead board
beaglev-starlight-jh7100	beta BeagleV Starlight board
eswin-ebc77	ESWIN Computing EBC77 with vendor Linux kernel
eswin-ebc77-mainline	ESWIN Computing EBC77 with mainline Linux kernel
freedom-u540	HiFive Unleashed development board
mangopi-mq-pro	MangoPi MQ Pro
milky-duo	Milk-V Duo
milky-megrez	Milk-V Megrez
muse-pi-pro	Muse Pi Pro with mainline Linux
nezha-allwinner-d1	Nezha Allwinner-d1 board
orangepi-r2s	OrangePi R2S
orangepi-rv2	OrangePi RV2
orangepi-rv2-mainline	OrangePi RV2 with mainline Linux
star64	Star64 board
visionfive	VisionFive board
visionfive2	VisionFive 2 board

Challenges and Lessons Learned

- Support for different SoCs at the same time
- No mailing list, just GitHub issues and pull requests 🤔
- Open to accepting recipes for downstream firmware, bootloader and kernel, before upstream is ready
- Some boards tend to go unmaintained
- Need for more code consolidation!
- Not sure it's used in production projects.
LTS branches are not maintained.
Hopefully *Wrynose* will be the first one.

recipes-bsp/u-boot/u-boot_%.bbappend

```
_DEPS = ""
_DEPS:riscv32 = "opensbi:do_deploy"
_DEPS:riscv64 = "opensbi:do_deploy"
_DEPS:milkv-duo = ""

do_compile[depends] += "${_DEPS}"

do_compile:prepend:ae350-ax45mp() {
    export OPENSBI=${DEPLOY_DIR_IMAGE}/fw_dynamic.bin
}

do_compile:prepend:freedom-u540() {
    export OPENSBI=${DEPLOY_DIR_IMAGE}/fw_dynamic.bin
}

do_compile:prepend:k1() {
    export OPENSBI=${DEPLOY_DIR_IMAGE}/fw_dynamic.bin
}

do_compile:prepend:visionfive2() {
    export OPENSBI=${DEPLOY_DIR_IMAGE}/fw_dynamic.bin
}
```

Nice code-consolidation opportunities

- Good solution to replace multiple MACHINE overrides
- Part of MACHINEOVERRIDES
- Can also use it in COMPATIBLE_MACHINE:

```
recipes-bsp/u-boot/u-boot-spl-k1.bb
```

```
...  
COMPATIBLE_MACHINE = "(k1)"
```

```
conf/machine/include/k1.inc
```

```
#@TYPE: SOC  
#@NAME: SpacemiT K1  
#@SOC: SpacemiT K1  
#@DESCRIPTION: SOC configuration for the SpacemiT K1 SoC  
  
require conf/machine/include/riscv/tune-riscv.inc  
require conf/machine/include/soc-family.inc  
  
SOC_FAMILY = "k1"  
  
PREFERRED_PROVIDER_virtual/kernel ?= "linux-mainline-k1"  
PREFERRED_PROVIDER_virtual/bootloader ?= "u-boot"  
  
KERNEL_DANGLING_FEATURES_WARN_ONLY = "1"  
...
```

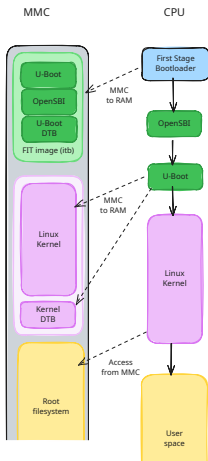

- The only thing the `deploy` class does is take care of staging the files from `DEPLOYDIR` to `DEPLOY_DIR_IMAGE`.
-  Don't install files directly to `DEPLOY_DIR_IMAGE`. This messes up with the shared state cache and causes unpredictable build failures.
- You still have to define the `deploy()` function and declare it through `addtask` too:

```
recipes-bsp/u-boot/boot-bundle.bb
```

```
addtask deploy after do_install before do_build
```

```
do_deploy() {  
    install -Dm 0644 ${B}/boot-bundle.itb ${DEPLOYDIR}/boot-bundle.itb  
}
```

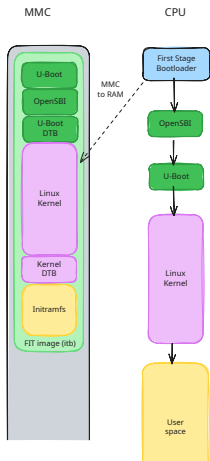
Booting without MMC support



In early stages, you may not have access to MMC from Linux and U-Boot (currently the case for SpacemiT K1 based boards)

- If the first stage bootloader (FSBL) supports MMC, you can load everything to RAM from there!
- Just put more things into the FIT image loaded by the FSBL.

Iker Pedrosa's work is adding MMC/SD support to SpacemiT K1 boards (Linux 7.2).



Typically to replace the original kernel by a mainline or recompiled one

- Dump the initial FIT image:

```
dtc beaglev_fire.itb > beaglev_fire.its
```

- Modify the .its file with a text editor:

Replace "data" binary sections with:

```
data = /incbin/("Image.gz");
```

```
...
```

```
data = /incbin/("mpfs-beaglev-fire.dtb");
```

Also remove the hash values (they are optional)

- Copy your kernel and DTB to the current directory
- Update the fit Image:

```
mkimage -f beaglev_fire.its beaglev_fire.itb
```



recipes-bsp/u-boot/boot-bundle.bb

 (excerpt)

```
DEPENDS:k1 = "u-boot-tools-native"
inherit deploy
SRC_URI = "file://boot-bundle.its"
S = "${UNPACKDIR}"

do_install[depends] += "virtual/kernel:do_deploy virtual/bootloader:do_deploy opensbi:do_deploy"

B = "${WORKDIR}/build"

do_install:k1() {
    install -d ${B}
    cd ${B}
    DTBFILE=$(basename ${KERNEL_DEVICETREE})
    install -m 0644 ${DEPLOY_DIR_IMAGE}/${KERNEL_IMAGETYPE} ${B}/
    install -m 0644 ${DEPLOY_DIR_IMAGE}/${DTBFILE} ${B}/
    install -m 0644 ${DEPLOY_DIR_IMAGE}/u-boot-nodtb.bin ${B}/
    install -m 0644 ${DEPLOY_DIR_IMAGE}/u-boot.dtb ${B}/
    install -m 0644 ${DEPLOY_DIR_IMAGE}/fw_dynamic.bin ${B}/
    sed "s/KERNEL_DEVICETREE/${DTBFILE}/" ${S}/boot-bundle.its > boot-bundle.its
    mkimage -f boot-bundle.its boot-bundle.itb
}

addtask deploy after do_install before do_build

do_deploy() {
    install -Dm 0644 ${B}/boot-bundle.itb ${DEPLOYDIR}/boot-bundle.itb
}
```

Any better way
to do this?

- I used to create my config fragments "very manually" 🤖
- A very convenient way exists:
 - ① Modify and save your existing configuration:
`bitbake -c menuconfig u-boot`
 - ② Compute the difference with the previous one:
`bitbake -c diffconfig u-boot`
This automatically generates the corresponding configuration fragment (check the output of the command)
- Big thanks to Navin Reddy for sharing this tip with me!

- When a layer carries Kas files
- Clone this layer, and use the Kas definitions inside:
`kas build /path/to/layer/kas/board.yml`
- However, neither `board.yml` nor its includes need to specify the path to the layer itself.
- Very convenient to test your changes in the layer without having to modify the Yaml files to point to a local directory.

```
header:
  version: 20
  includes:
    - include/local.yml
    - include/oe.yml

machine: qemuriscv64
distro: poky-altcfg
target: core-image-full-cmdline

repos:
  meta-riscv:
```

Instead of:

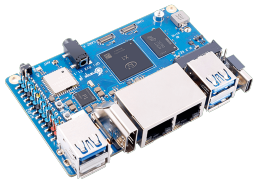
```
header:
  version: 20
  includes:
    - include/local.yml
    - include/oe.yml

machine: qemuriscv64
distro: poky-altcfg
target: core-image-full-cmdline

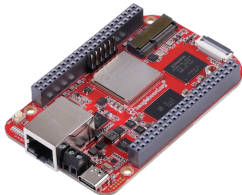
repos:
  meta-riscv:
    path: /home/dev/meta-riscv
```

Opportunities to contribute

Today's Attractive Boards



Orange Pi RV2
(SpacemiT K1, currently 80 EUR for 4 GB version)



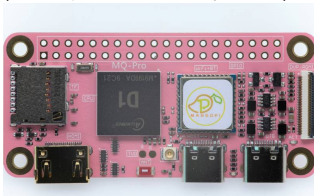
BeagleV-Fire
(Microchip PolarFire, currently 85 EUR)



MilkV Duo-S
(Sophgo SG2000, currently 9 EUR)



Orange Pi R2S
(SpacemiT K1, currently 65 EUR for 2 GB version)



MangoPi MQ Pro
(Allwinner D1, currently 25 EUR)



Sipeed / BananaPi K3 Com260
(SpacemiT K3, currently 280 EUR for 8 GB version)



The gentleman on the right is excited about the new architecture 😊

Mainline Linux kernel support (most important)

- Without mainline Linux kernel support, products only have a "demo" value: no updates, no security
- Mainline Linux also defines the reference Device Tree Sources (DTS) for your board.
- Yocto contribution work will be much easier too!
- Subscribe to the `linux-riscv` mailing list and check its archives on <https://lore.kernel.org/linux-riscv/>
- Use the `b4` tool to manage your changes and submissions: <https://b4.docs.kernel.org/en/latest/>
- Easier to submit new code right after an `rc1` release has been made: you don't have to manage dependencies with multiple maintainer trees.

Mainline bootloader

- Linux has priority, but important too
- Will use the Linux DTS, and can reuse Linux driver code
- Support more boards in U-Boot!
- Port k1 MMC support to U-Boot

Other projects worth supporting:

- Firmware: OpenSBI
- First stage bootloader: DRAM training

Contribution ideas

- Find new boards and make them work.
- Review and test contributions from others.
Great way to learn and get new ideas.
- Just build images and try to use your board.
You will find contribution ideas.
- Create a new semiconductor company making
(cheap) DRAM 😊

Where help is needed

- Update and refactor existing recipes
- Replace vendor support code by mainline alternatives whenever possible
- Helping with CI testing
(work with Trevor Gamblin)
- Maintaining a first LTS branch (Wrynose)

Here's how we flash our K1 eMMC or SD images

```
$ sudo bmaptool copy \  
  build/tmp/deploy/images/bananapi-f3/core-image-full-cmdline-bananapi-f3.rootfs.wic.gz \  
  /dev/sdx  
$ sudo dd if=build/tmp/deploy/images/bananapi-f3/bootinfo_sd.bin of=/dev/sdx
```

How to add a small binary at the beginning of the Wic image?

- Tried many options, none worked at offset 0.
- Really need to write a custom Wic plugin like the code flashing MBR on x86?
- Are there Wic experts in the room?

- Add a new machine:
`conf/machine/newboard.conf`
- Adapt recipes when needed: firmware, bootloader, kernel
- Add a Kas file for building an image:
`kas/newboard.yml`
- Test your changes
- Document how to use your board:
`docs/newboard.md`
- Add the new board to `README.md`
- Create a new branch:
`git checkout -b add-newboard`
- Commit your changes:
`git add -A .`
`git commit -as`
- Generate patches for your changes:
`git format-patch master`
- Use `patchtest` to run verification tests:
`patchtest --patch *.patch`
- Submit your pull request on GitHub
- Our hooks will run further checks too (yocto-check-layer)

- Adding Support for New Boards — OpenEmbedded Workshop 2026
<https://rootcommit.com/pub/conferences/2026/oe-workshop/yocto-new-boards/>
- Orange Pi RISC-V Support — FOSDEM 2026
<https://rootcommit.com/pub/conferences/2026/fosdem/orange-pi-riscv-support/>
- Root Commit freely available training materials
<https://rootcommit.com/yocto/>

- A pure community project. Not driven by RISC-V.org or RISE (Linux Foundation).
- Small community with still a lot of room for contribution!
- Your patches don't have to be perfect. Let us help you help us.
- Even complicated code can be accepted, and even recipes building vendor BSP code (firmware, bootloader and even kernel), as long as it's a step in the right direction.



- A pure community project. Not driven by RISC-V.org or RISE (Linux Foundation).
- Small community with still a lot of room for contribution!
- Your patches don't have to be perfect. Let us help you help us.
- Even complicated code can be accepted, and even recipes building vendor BSP code (firmware, bootloader and even kernel), as long as it's a step in the right direction.



Questions? Comments?

- mo@rootcommit.com
-  <https://fosstodon.org/@MichaelOpdenacker>
- XMPP: omichael@conversations.im
- Signal: rootcommit.01
- Slides available under the CC-BY-SA 4.0 license
<https://rootcommit.com/pub/conferences/2026/yocto-workshop-nice/meta-riscv-status/>
- Sources ($\text{\LaTeX}$):
<https://gitlab.com/rootcommit/meta-riscv-status/>

