Adding Support for New Boards

OpenEmbedded Workshop 2026

Michael Opdenacker

Root Commit

Feb. 2, 2025

Michael Opdenacker

root commit

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)

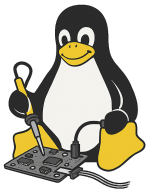
April
promouvoir et défendre le logiciel libre

What I Do For a Living — What's Funding This Work

root commit

Consulting and engineering work

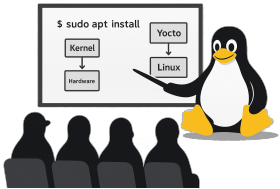
- Yocto Project
  Project reviews, system implementation, new features
- Linux Kernel
  Driver development, board bring-up, debugging
- Embedded Linux
  Boot time, bug fixing, security and other optimizations



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.



Introduction

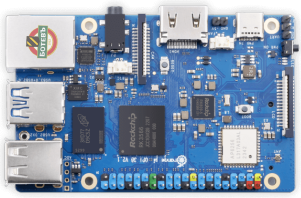
Motivations

root commit

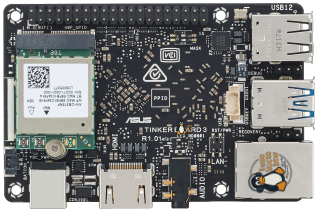
- You need a new MACHINE to use your favorite build system
- What you build is reproducible... Our build system makes it easy for others to build on your work.
- This helps people contribute to mainline U-Boot and Linux support, without having to worry about the SoC specific booting details.
- This presentation can help people making custom vendor trees, but such trees are often left to bitrot after their initial release.
- Goal to support contributors to community layers, which are much more likely and able to offer support and updates.
- Take over the world and make it a better place!

Contributed MACHINEs

- `meta-rockchip`
  - `orangepi-3b` (commit)
  - `tinker-board-3` and `tinker-board-3s` from Asus (waiting for Linux 6.19, patch)
- `meta-riscv`
  - `orangepi-rv2-mainline` (commit)
  - `bananapi-f3` (commit)



OrangePi 3B

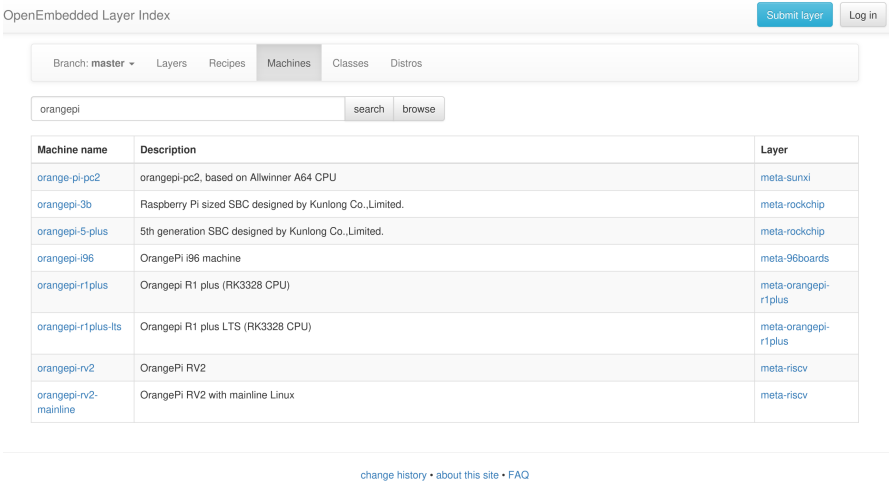


Asus Tinker Board 3S

First Thing To Do

root commit

Check `https://layers.openembedded.org` for already supported machines



OpenEmbedded Layer Index

Submit layer | Log in

Branch: **master** | Layers | Recipes | Machines | Classes | Distros

orangepi — search | browse

Machine name	Description	Layer
orange-pi-pc2	orangepi-pc2, based on Allwinner A64 CPU	meta-sunxi
orangepi-3b	Raspberry Pi sized SBC designed by Kunlong Co.,Limited.	meta-rockchip
orangepi-5-plus	5th generation SBC designed by Kunlong Co.,Limited.	meta-rockchip
orangepi-i96	OrangePi i96 machine	meta-96boards
orangepi-r1plus	Orangepi R1 plus (RK3328 CPU)	meta-orangepi-r1plus
orangepi-r1plus-lts	Orangepi R1 plus LTS (RK3328 CPU)	meta-orangepi-r1plus
orangepi-rv2	OrangePi RV2	meta-riscv
orangepi-rv2-mainline	OrangePi RV2 with mainline Linux	meta-riscv

change history • about this site • FAQ

Priority to Mainline Linux

root commit

- Only mainline Linux and derivatives get updates
- U-Boot also depends on Linux for describing the hardware (Device Tree)
- Therefore, it's acceptable to rely **initially** on vendor U-Boot and firmware
- That depends on layer policy though:
  - `meta-rockchip` doesn't want to introduce new kernel recipes. New boards should be supported by a kernel covered by `linux-yocto`. A recipe based on `meta-linux-mainline` wasn't accepted.
  - `meta-riscv` is more flexible and accepts machines with vendor trees for firmware, U-Boot and Linux, considering it's a first step. Price to pay: many custom and harder to maintain recipes.

What to Do with Vendor Layers and Trees

root commit

- See whether they are regularly updated. If that's the case, you may decide to keep them in your project (example: Toradex layers)
- Otherwise, or if you want to contribute to a community BSP layer, it's easier to start from scratch, just using vendor definitions as a source of information.
- In most cases, relying on your own trees and community layers will turn out to be much cheaper in terms of maintenance efforts.

`MACHINE` Definitions

Create a new MACHINE definition

root commit

conf/machine/orangepi-3b.conf 🔗 (meta-rockchip)

```
#@TYPE: Machine
#@NAME: Orange Pi 3B
#@DESCRIPTION: Raspberry Pi sized SBC designed by Kunlong Co.,Limited.
#http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-3B.html

require conf/machine/include/rk3566.inc

KERNEL_DEVICETREE = "rockchip/rk3566-orangepi-3b-v2.1.dtb rockchip/rk3566-orangepi-3b-v1.1.dtb"
MACHINE_EXTRA_RRECOMMENDS += "kernel-modules"

UBOOT_MACHINE = "orangepi-3b-rk3566_defconfig"
```

MACHINE Definition Notes

root commit

- Remember that MACHINE refers to a .conf file
  with configuration (global) variables that all recipes will see.
- Here are the main variables to define here:
  - KERNEL_DEVICETREE: list of kernel device trees
    (the bootloader should find which one to load).
  - UBOOT_MACHINE: U-Boot configuration file
  - PREFERRED_PROVIDER for virtual/kernel amd virtual/bootloader
    if not using the default recipes for kernel and bootloader.
  - WKS_FILE: specification of flash or block storage image.
    Can be shared with boards with the same SoC.

SOC_FAMILY

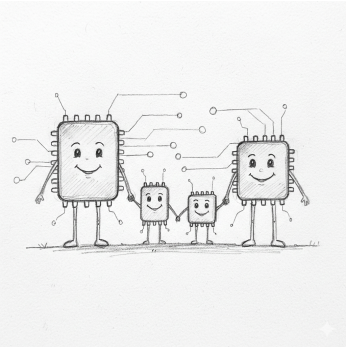
- Part of MACHINEOVERRIDES
- Useful to share settings between machines with the same SoC

meta-riscv 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"
```



SOC_FAMILY in MACHINEOVERRIDES

root commit

recipes-bsp/opensbi/opensbi_%.bbappend

```
# Master branch on Dec 12, 2025, includes SpacemiT K1 fixes
SRCREV:k1 = "51fe6a8bc958166ff79805cf69bafe5e297776f4"
```

recipes-kernel/linux/linux-mainline-k1.bb

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

What Not To Put in a MACHINE Definition

root commit

Where to go for...

BSP Layers

- Machine definitions
- Kernel recipes and config
- Bootloader recipes
- Firmware recipes
- Custom Hardware utilities
- Instruction set definitions
- Machine features (supported HW drivers)
- Machine specific bbappends

Images

- List of packages
- Package groups
- Image size and free space
- Partitioning scheme
- Image features (root login...)

Distro Layers

- Package policies
- Toolchain
- C standard library
- Init manager
- Distro features (ipv6, sound server...)

Suggestion: create https://OpenLayerMap.org

WKS_FILE Definition

root commit

wic/orangepi-rv2.wks

```
# Orange Pi RV2 SD Card Layout

part spl --offset 256K --fixed-size 256K --fstype=none --source rawcopy --sourceparams="file=u-boot-spl.bin"
part uboot --offset 1M  --fixed-size 3M   --fstype=none --source rawcopy --sourceparams="file=u-boot-opensbi.itb"
part /boot --ondisk mmcblk0 --fstype=vfat --label boot --offset 4M --size 128M --source bootimg-partition --use-uuid --active --
align 4096
part / --ondisk mmcblk0 --fstype=ext4 --label root --source rootfs --use-uuid --align 4096
```

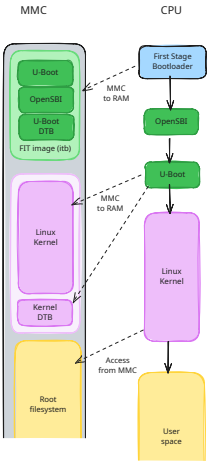
IMAGE_BOOT_FILES specifies the contents of the FAT partition described by `--source bootimg-partition`.

conf/machine/orangepi-rv2.conf

```
IMAGE_BOOT_FILES += " \
 ${UBOOT_ENV}.${UBOOT_ENV_SUFFIX} \
 x1_orangepi-rv2.dtb \
 ${KERNEL_IMAGETYPE} \
 initramfs.img"
```

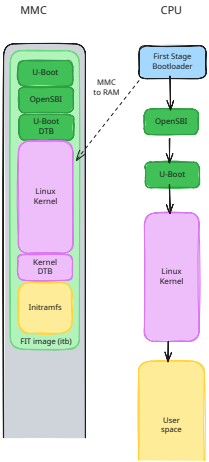
Implementation Details

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.



How to "Decapsulate" a FIT Image

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
  ```

The boot-bundle recipe in meta-riscv

root commit

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
}
```

Our build system makes it easy!

- In your `machine.conf` file, you can set the `INITRAMFS_IMAGE` variable:

  ```
  INITRAMFS_IMAGE = "core-image-minimal-initramfs"
  ```

  This will automatically get this additional image to be built.

- If you're using U-Boot, you can add this to your kernel recipe, or to a custom one:

  ```
  DEPENDS:append = " u-boot-tools-native"

  do_deploy:append() {
          cd ${DEPLOY_DIR_IMAGE}
          mkimage -A riscv -O linux -T ramdisk -n "Initial Ram Disk" \
                  -d ${INITRAMFS_IMAGE}-${MACHINE}.cpio.gz initramfs.img
  }
  ```

- Then pass the address where the Initramfs was loaded to U-Boot when starting Linux:

  ```
  booti <kernel_addr> <initramfs_addr> <dtb_addr>
  ```

Limitations of the "Boot Bundle" Approach

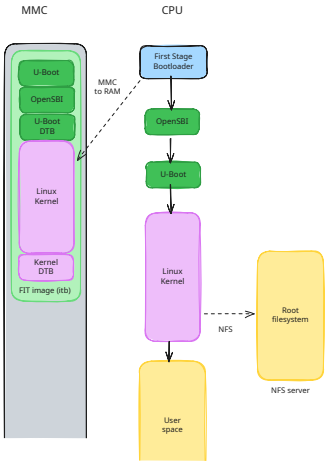
root commit

- Though attractive, don't expect to load the whole system using this method
- For example, on OrangePi RV2,
  this stopped working with a FIT image bigger than 64 MB.

Variant: booting from RAM + NFS

On SpacemiT K1 (Orange Pi RV2, BananaPi F3...), we are lucky that the mainline Linux kernel already supports networking.

- You can boot your kernel on a directory shared over the network through NFS (Network File System)
- NFS is great when you don't have support for storage yet
- It's better than initramfs, as it can be as big as necessary.
- See https://github.com/riscv/meta-riscv/blob/master/docs/orangepi-rv2-mainline.md#set-up-nfs for setup details.



U-Boot Tips

root commit

- Even if U-Boot doesn't have a configuration for your board yet:
  - It may provide a generic configuration for your SoC (Asus Tinker Board 3 example):

    ```
    UBOOT_MACHINE = "generic-rk3568_defconfig"
    ```
  - You may also be lucky and be able to build U-Boot with a configuration for a similar board (Asus Tinker Board 3 alternative):

    ```
    UBOOT_MACHINE = "soquartz-model-a-rk3566_defconfig"
    ```
- U-Boot should have logic to detect which board variant you have, and pass the right DTB to the kernel (Orange Pi 3B example):

  ```
  KERNEL_DEVICETREE = "rockchip/rk3566-orangepi-3b-v2.1.dtb rockchip/rk3566-orangepi-3b-v1.1.dtb"
  ```
- Many boards now ship with different RAM sizes.
  U-Boot should also detect that and update the DTB.

Kernel Recipes Choices

- `linux-yocto`
  - Easiest recipe to use. Supports configuration fragments.
  - But only supports stable kernel, and sometimes with a delay (no `-rc` ones)
  - Works with boards well supported by the mainline Linux kernel
- `linux-mainline`
  - Supports all Linux mainline stable and `-rc` kernels.
  - Inherits the `kernel-yocto` class, and therefore offers the features of `linux-yocto`
  - But requires the `meta-linux-mainline` layer
- Custom kernel recipe
  - Very easy to implement with the basic `kernel` class
  - Can use mainline or vendor kernel

The maintainer of `meta-linux-mainline` is British and is here in Brussels and even in this room 😉. Image source: Wikimedia Commons

Create a Kas File for Your Board

root commit

- Several layers are doing this.
- This makes it easier for people to test support for your board. One file can be sufficient.
- Example:
  `kas build --update /path/to/meta-riscv/kas/orangepi-rv2-mainline.yml`
- Not sure we can do that in one command yet with `bitbake-setup`, which I'm using too 😉.

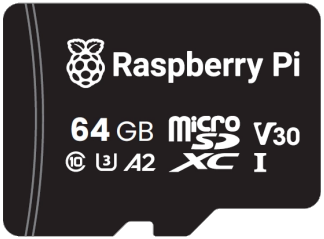
kas/orangepi-rv2-mainline.yml

```yaml
header:
  version: 20
  includes:
    - base-riscv.yml
machine: orangepi-rv2-mainline
```

Additional Tips

root commit

- Deploy all modules to the filesystem:
  `MACHINE_EXTRA_RRECOMMENDS += "kernel-modules"`
- Build an image with *systemd*. It will automatically load all relevant modules:
  `INIT_MANAGER = "systemd"`
- If you're frequently rebuilding your kernel as expected, try to use `ccache` to speed-up compiling: https://docs.yoctoproject.org/ref-manual/classes.html#ccache.
- Reuse as much existing code as possible, this reduces maintenance costs.
  Use machine overrides for board or SoC specific settings.
- Use `bmaptool` to flash your images, so much faster than `dd`
- Invest in a fast SD card



Raspberry Pi official SD cards are fast, reliable and cheap (currently 15 EUR for 64 GB at https://kubii.com.)

Checklist Before Sharing

root commit

- Don't forget to add the new `MACHINE` to the layer `README` file.
- Check your changes with `patchtest`
  https://docs.yoctoproject.org/contributor-guide/submit-changes.html#validating-patches-with-patchtest
- Check the resulting layer with `yocto-check-layer` (optional but nice):
  https://docs.yoctoproject.org/dev-manual/layers.html#yocto-check-layer-script

Belgian policeman from Gaston Lagaffe (Franquin)

Wrapping Up

What To Remember

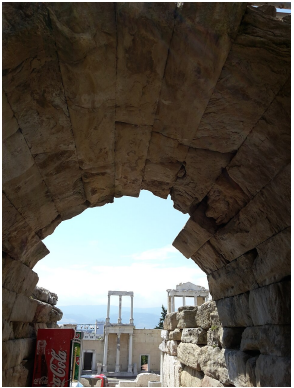
root commit

- Very few board vendors are good at working with the community
- Standard OE / Yocto support for a board helps to make the hardware accessible to a wider public (vendor scripts and layers can be so crap... who would want touch those?)
- Mainline Linux support should be the first priority, but as an OE / Yocto contributor, you can abstract away SoC specific boot details, for the benefit of other contributors.
- Adding support for a new board in Linux, U-Boot and Yocto can be easier that you can expect, and anyway instructive, rewarding and even addictive.
- You are never alone (you'll be surprise how helpful the community can be), you can also learn from the way other (often similar) boards are supported.
- `meta-rockchip` (https://git.yoctoproject.org/meta-rockchip) is a clean community layer worth imitating.

Contributor Tips

root commit

- Be patient, value the reviews you get as opportunities to grow, and patiently improve your work until all the comments are addressed.
- Adapt to the way the project is maintained and how contributions are shared (mailing lists or pull/merge requests through a web interface).
- If you're stuck with a problem, move to something else, sleep on it, and try again the next morning.
- Also dedicate some time to review contributions from others. Reviews from people are as valuable as code contributions, if not even more so.



Be patient, you will reach the end of the tunnel
Image source: Wikimedia Commons

Thank you

root commit

Thanks to Leon and Frank for their contributions to this talk 😉

Questions? Comments?

- mo@rootcommit.com
- https://fosstodon.org/@MichaelOpdenacker
- XMPP: omichael@conversations.im
- Signal: rootcommit.01

- Sources (LaTeX):
  https://gitlab.com/rootcommit/<title>/

I ❤ Minneapolis