Board Porting Guide

To add Zephyr support for a new board, you at least need a *board directory* with various files in it. Files in the board directory inherit support for at least one SoC and all of its features. Therefore, Zephyr must support your SoC as well.

Transition to the current hardware model

Shortly after Zephyr 3.6.0 was released, a new hardware model was introduced to Zephyr. This new model overhauls the way both SoCs and boards are named and defined, and adds support for features that had been identified as important over the years. Among them:

- Support for multi-core, multi-arch AMP (Asymmetrical Multi Processing) SoCs
- Support for multi-SoC boards
- Support for reusing the SoC and board Kconfig trees outside of the Zephyr build system
- Support for advanced use cases with Sysbuild (System build)
- Removal of all existing arbitrary and inconsistent uses of Kconfig and folder names

All the documentation in this page refers to the current hardware model. Please refer to the documentation in Zephyr v3.6.0 (or earlier) for information on the previous, now obsolete, hardware model.

More information about the rationale, development and concepts behind the new model can be found in the original issue, the original Pull Request and, for a complete set of changes introduced, the hardware model v2 commit.

Some non-critical features, enhancements and improvements of the new hardware model are still in development. Check the hardware model v2 enhancements issue for a complete list.

The transition from the previous hardware model to the current one (commonly referred to as “hardware model v2”) requires modifications to all existing board and SoC definitions. A decision was made not to provide direct backwards compatibility for the previous model, which leaves users transitioning from a previous version of Zephyr to one including the new model (v3.7.0 and onwards) with two options if they have an out-of-tree board (or SoC):

1. Convert the out-of-tree board to the current hardware model (recommended)
2. Take the SoC definition from Zephyr v3.6.0 and copy it to your downstream repository (ensuring that the build system can find it via a zephyr module or `SOC_ROOT`). This will allow your board, defined in the previous hardware model, to continue to work

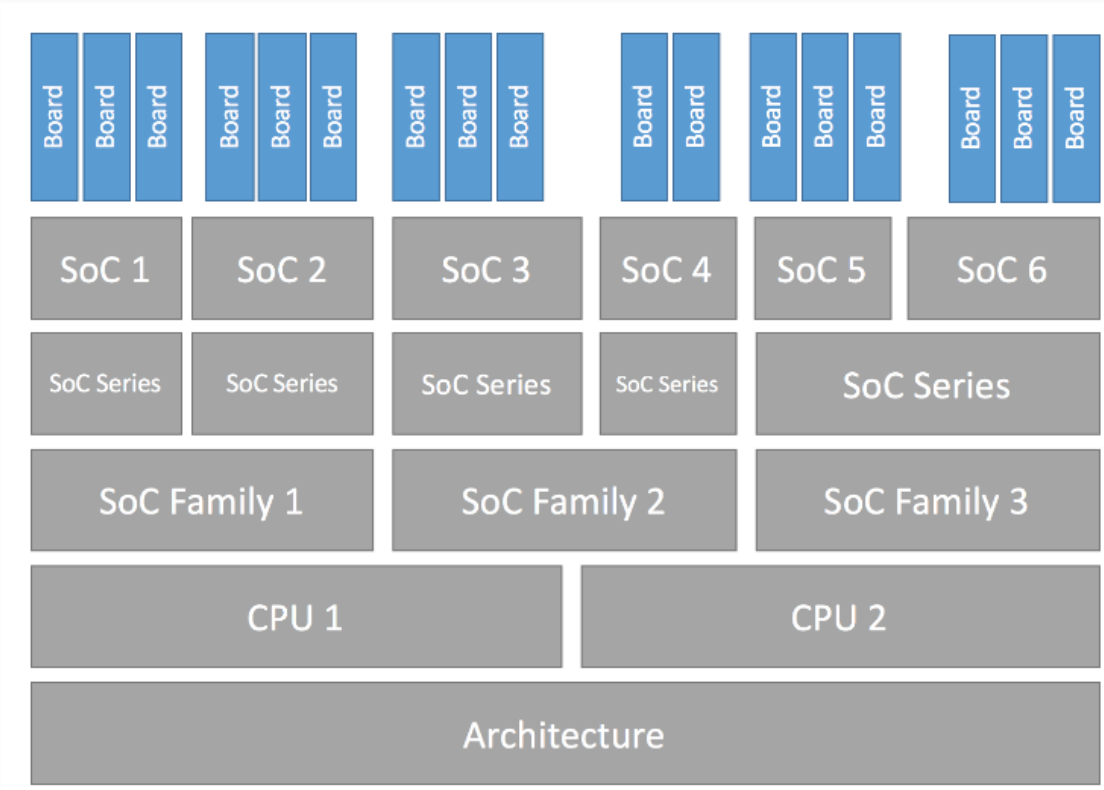
When converting your board from the previous to the current hardware model, we recommend first reading through this page to understand the model in detail. You can then use the example-application conversion Pull Request as an example on how to port a simple board. Additionally, a conversion script is available and works reliably in many cases (though multi-core SoCs may not be handled entirely). Finally, the hardware model v2 commit contains the full conversion of all existing boards from the old to the current model, so you can use it as a complete conversion reference.

Hardware support hierarchy

Zephyr’s hardware support is based on a series of hierarchical abstractions. Primarily, each board has one or more SoC. Each SoC can be optionally classed into an SoC

series, which in turn may optionally belong to an SoC family. Each SoC has one or more CPU cluster, each containing one or more CPU core of a particular architecture.

You can visualize the hierarchy in the diagram below:

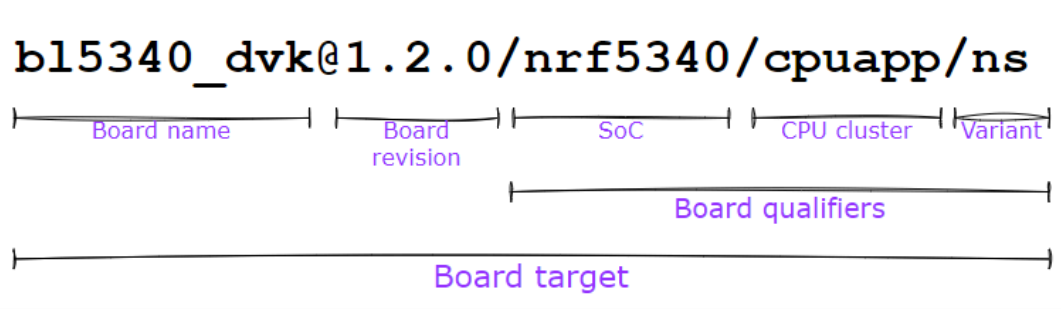


Hardware support Hierarchy

Board terminology

The previous section introduced the hierarchical manner in which Zephyr classifies and implements hardware support. This section focuses on the terminology used around hardware support, and in particular when defining and working with boards and SoCs.

The overall set of terms used around the concept of board in Zephyr is depicted in the image below, which uses the Ezurio BL5340 DVK board as reference.



Board terminology diagram

board name	board qualifiers	SoC	SoC Series	SoC family	CPU core	architecture
nrf52dk	nrf52832	nRF52832	nRF52	Nordic nRF	Arm Cortex-M4	ARMv7-M
frdm_k64f	mk64f12	MK64F12	Kinetis K6x	NXP Kinetis	Arm Cortex-M4	ARMv7-M
rv32m1_vega	openisa_rv32m1/ri5cy	RV32M1	(Not used)	(Not used)	RI5CY	RISC-V RV32
nrf5340dk	nrf5340/cpuapp	nRF5340	nRF53	Nordic nRF	Arm Cortex-M33	ARMv8-M
	nrf5340/cpunet	nRF5340	nRF53	Nordic nRF	Arm Cortex-M33	ARMv8-M
mimx8mp_evk	mimx8ml8/a53	i.MX8M Plus	i.MX8M	NXP i.MX	Arm Cortex-A53	ARMv8-A
	mimx8ml8/m7	i.MX8M Plus	i.MX8M	NXP i.MX	Arm Cortex-M7	ARMv7-M

The diagram shows the different terms that are used to describe boards:

- The board name: `bl5340_dvk`
- The optional board revision: `1.2.0`
- The board qualifiers, that optionally describe the SoC, CPU cluster and variant: `nrf5340/cpuapp/ns`

- The board target, which uniquely identifies a combination of the above and can be used to specify the hardware to build for when using the tooling provided by Zephyr: `bl5340_dvk@1.2.0/nrf5340/cpuapp/ns`

Formally this can also be seen as *`board name`*`[@`*`revision`*`][/`*`board qualifiers`*`]`, which can be extended to *`board name`*`[@`*`revision`*`][/`*`SoC`*`[/`*`CPU cluster`*`][/`*`variant`*`]]`.

If a board contains only one single-core SoC, then the SoC can be omitted from the board target. This implies that if the board does not define any board qualifiers, the board name can be used as a board target. Conversely, if board qualifiers are part of the board definition, then the SoC can be omitted by leaving it out but including the corresponding forward-slashes: `//`.

Continuing with the example above, The board Ezurio BL5340 DVK is a single SoC board where the SoC defines two CPU clusters: `cpuapp` and `cpunet`. One of the CPU clusters, `cpuapp`, additionally defines a non-secure board variant, `ns`.

The board qualifiers `nrf5340/cpuapp/ns` can be read as:

- `nrf5340`: The SoC, which is a Nordic nRF5340 dual-core SoC
- `cpuapp`: The CPU cluster `cpuapp`, which consists of a single Cortex-M33 CPU core. The number of cores in a CPU cluster cannot be determined from the board qualifiers.
- `ns`: a variant, in this case `ns` is a common variant name is Zephyr denoting a non-secure build for boards supporting Trusted Firmware-M.

Not all SoCs define CPU clusters or variants. For example a simple board like the Thingy:52 contains a single SoC with no CPU clusters and no variants. For `thingy52` the board target `thingy52/nrf52832` can be read as:

- `thingy52`: board name.
- `nrf52832`: The board qualifiers, in this case identical to the SoC, which is a Nordic nRF52832.

Make sure your SoC is supported

Start by making sure your SoC is supported by Zephyr. If it is, it's time to Create your board directory. If you don't know, try:

- checking Supported Boards for names that look relevant, and reading individual board documentation to find out for sure.
- asking your SoC vendor

If you need to add a SoC, CPU cluster, or even architecture support, this is the wrong page, but here is some general advice.

Architecture

See Architecture Porting Guide.

CPU Core

CPU core support files go in `core` subdirectories under arch, e.g. arch/x86/core.

See Install a Toolchain for information about toolchains (compiler, linker, etc.) supported by Zephyr. If you need to support a new toolchain, Build and Configuration Systems is a

good place to start learning about the build system. Please reach out to the community if you are looking for advice or want to collaborate on toolchain support.

SoC

Zephyr SoC support files are in architecture-specific subdirectories of soc. They are generally grouped by SoC family.

When adding a new SoC family or series for a vendor that already has SoC support within Zephyr, please try to extract common functionality into shared files to avoid duplication. If there is no support for your vendor yet, you can add it in a new directory `zephyr/soc/<VENDOR>/<YOUR-SOC>`; please use self-explanatory directory names.

Create your board directory

Once you've found an existing board that uses your SoC, you can usually start by copy/pasting its board directory and changing its contents for your hardware.

You need to give your board a unique name. Run `west boards` for a list of names that are already taken, and pick something new. Let's say your board is called `plank` (please don't actually use that name).

Start by creating the board directory `zephyr/boards/<VENDOR>/plank`, where `<VENDOR>` is your vendor subdirectory. (You don't have to put your board directory in the zephyr repository, but it's the easiest way to get started. See Custom Board, Devicetree and SOC Definitions for documentation on moving your board directory to a separate repository once it's working.)

Note

A `<VENDOR>` subdirectory is mandatory if contributing your board to Zephyr, but if your board is placed in a local repo, then any folder structure under `<your-repo>/boards` is permitted. If the vendor is defined in the list in dts/bindings/vendor-prefixes.txt then you must use that vendor prefix as `<VENDOR>`. `others` may be used as vendor prefix if the vendor is not defined.

Note

The board directory name does not need to match the name of the board. Multiple boards can even defined be in one directory.

Your board directory should look like this:

```
boards/<VENDOR>/plank
├── board.yml
├── board.cmake
├── CMakeLists.txt
├── doc
│   ├── plank.png
│   └── index.rst
├── Kconfig.plank
├── Kconfig.defconfig
├── plank_defconfig
```

```
├── plank_<qualifiers>_defconfig
├── plank.dts
├── plank_<qualifiers>.dts
└── plank.yaml
```

Replace `plank` with your board's name, of course.

The mandatory files are:

1. `board.yml`: a YAML file describing the high-level meta data of the boards such as the boards names, their SoCs, and variants. CPU clusters for multi-core SoCs are not described in this file as they are inherited from the SoC's YAML description.
2. `plank.dts` or `plank_<qualifiers>.dts`: a hardware description in devicetree format. This declares your SoC, connectors, and any other hardware components such as LEDs, buttons, sensors, or communication peripherals (USB, BLE controller, etc).
3. `Kconfig.plank`: the base software configuration for selecting SoC and other board and SoC related settings. Kconfig settings outside of the board and SoC tree must not be selected. To select general Zephyr Kconfig settings the `Kconfig` file must be used.

The optional files are:

- `Kconfig`, `Kconfig.defconfig` software configuration in Configuration System (Kconfig) formats. This provides default settings for software features and peripheral drivers.

- `plank_defconfig` and `plank_<qualifiers>_defconfig`: software configuration in Kconfig `.conf` format.
- `board.cmake`: used for Flash and debug support
- `CMakeLists.txt`: if you need to add additional source files to your build.
- `doc/index.rst`, `doc/plank.png`: documentation for and a picture of your board. You only need this if you're Contributing your board to Zephyr.
- `plank.yaml`: a YAML file with miscellaneous metadata used by the Test Runner (Twister).

Board qualifiers of the form `<soc>/<cpucluster>/<variant>` are normalized so that `/` is replaced with `_` when used for filenames, for example: `soc1/foo` becomes `soc1_foo` when used in filenames.

Write your board YAML

The board YAML file describes the board at a high level. This includes the SoC, board variants, and board revisions.

Detailed configurations, such as hardware description and configuration are done in devicetree and Kconfig.

The skeleton of the board YAML file is:

```
board:
  name: <board-name>
  vendor: <board-vendor>
  revision:
    format: <major.minor.patch|letter|number|custom>
    default: <default-revision-value>
    exact: <true|false>
    revisions:
    - name: <revA>
    - name: <revB>
      ...
```

```
  socs:
  - name: <soc-1>
    variants:
    - name: <variant-1>
    - name: <variant-2>
      variants:
      - name: <sub-variant-2-1>
        ...
  - name: <soc-2>
    ...
```

It is possible to have multiple boards located in the board folder. If multiple boards are placed in the same board folder, then the file `board.yml` must describe those in a list as:

```
boards:
- name: <board-name-1>
  vendor: <board-vendor>
  ...
- name: <board-name-2>
  vendor: <board-vendor>
  ...
...
```