NORDICTECH WEBINARS

Adding custom board support in nRF Connect SDK

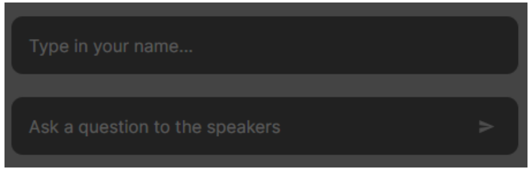
Today’s speaker

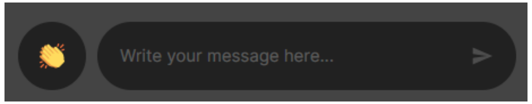
Ali Aljaani

Developer Marketing Manager

Practicalities

- Duration: 60 min presentation, 10 min Q&A
- Questions are encouraged!
  - Please type questions on the top of the right sidebar
  - All questions are anonymous
  - Try to keep them relevant to the topic
  - We will answer them toward the end
- The chat on the bottom of the right sidebar is not anonymous, and it should not be used for questions.
- Go to DevZone if you have more questions
- A recording of the webinar will be available together with the presentation at webinars.nordicsemi.com/on-demand





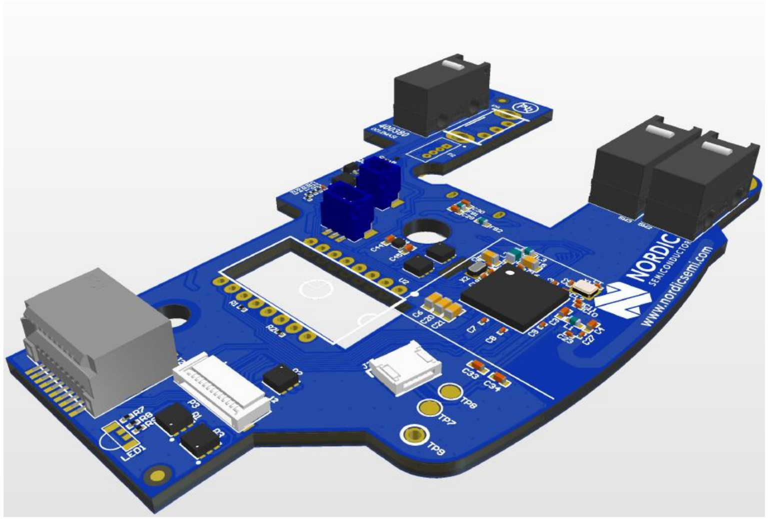
{DevZone

Agenda

- What is a ”board” in the context of nRF Connect SDK/Zephyr?
- Hardware support hierarchy in nRF Connect SDK/Zephyr.
- Mandatory, optional, and special use case board files.
- Special considerations for the nRF91 and nRF53 Series.
- Hands-on demo:
  - Translating Hardware schematics to devicetree syntax in nRF Connect for VS Code.
  - How to write your board Kconfig files for your hardware design.

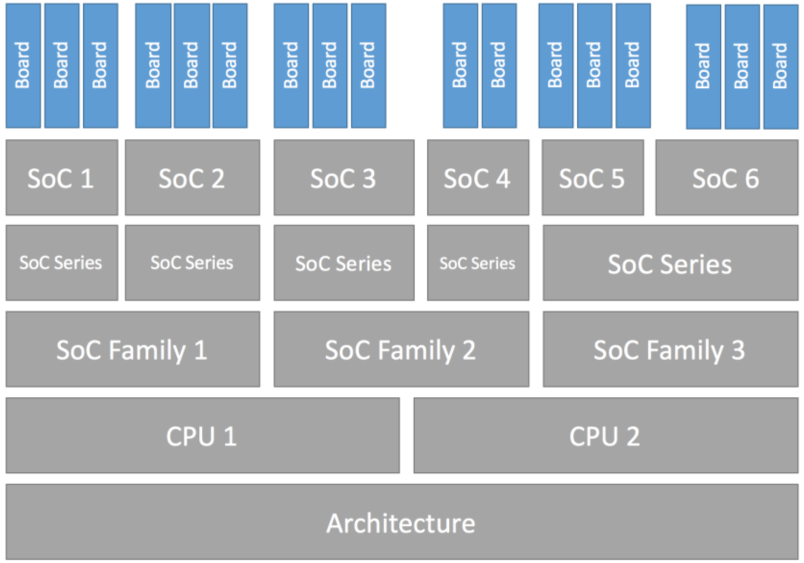
What is a "board" ?

- Your target hardware
  - Which SoC/SiP used
  - Configurations (HW/SW)
  - Components (sensors, connectors, etc.)
- A board is a directory that contains several files:
  - Devicetree files
  - Kconfig files
  - CMake/.c for special use cases
  - Optionally, documentation for your hardware
  - Optionally, YAML files

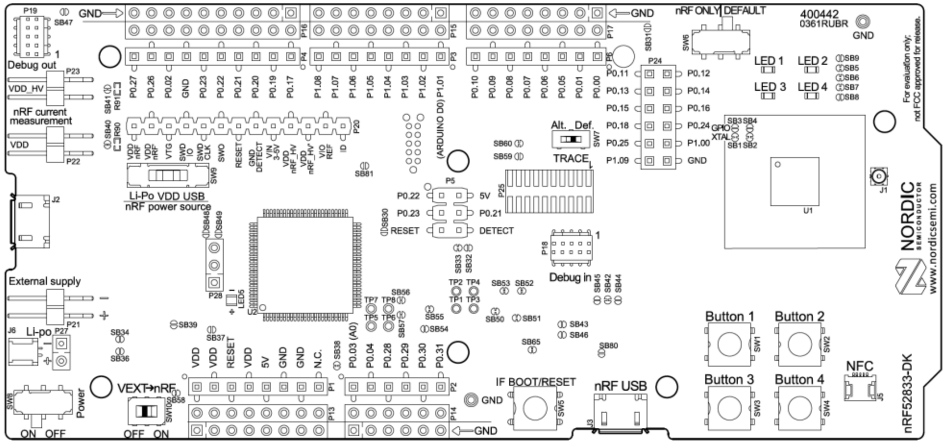


Hardware Support Hierarchy

- Board: a particular SoC/SiP instance and its peripherals in a concrete hardware specification
- SoC: the exact system on a chip the board's CPU is part of
- SoC Series: a smaller group of tightly related SoCs
- SoC Family: a wider group of SoCs with similar characteristics
- CPU Core: a particular CPU in an architecture
- Architecture: an instruction set architecture

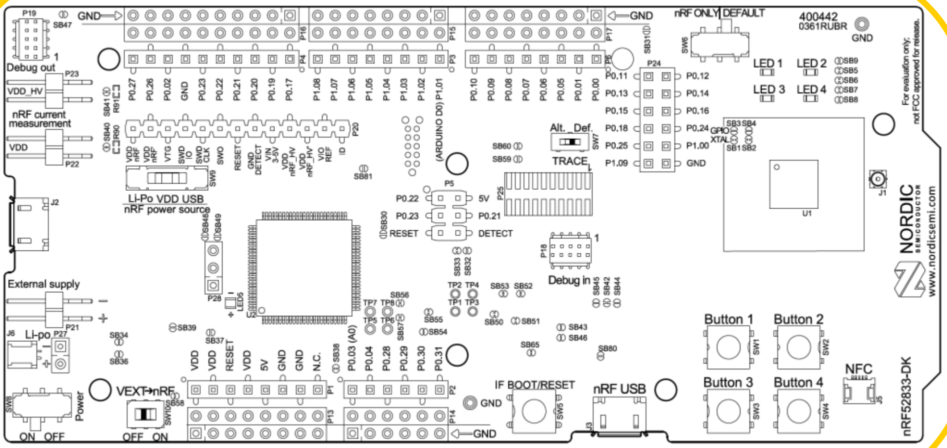


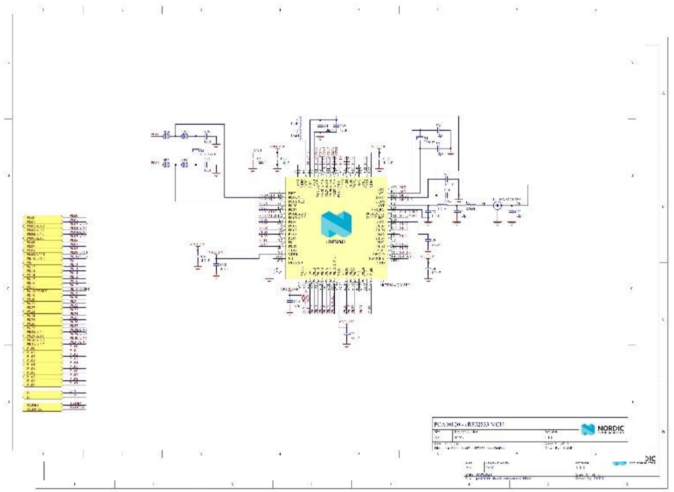
Hardware Support Hierarchy - Example



Board	SoC	SoC Series	SoC Family	CPU Core	Architecture
nrf52833dk_ nrf52833	nRF52833	nRF52	Nordic nRF	Arm Cortex-M4	Arm

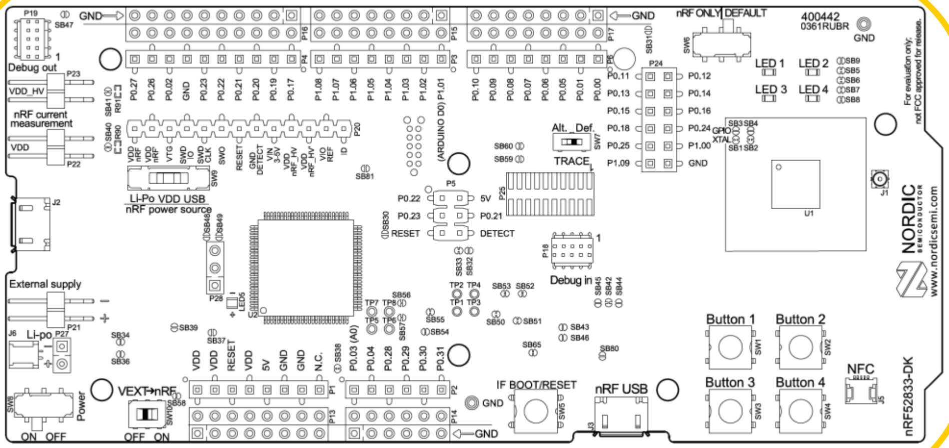
Hardware Support Hierarchy - Example

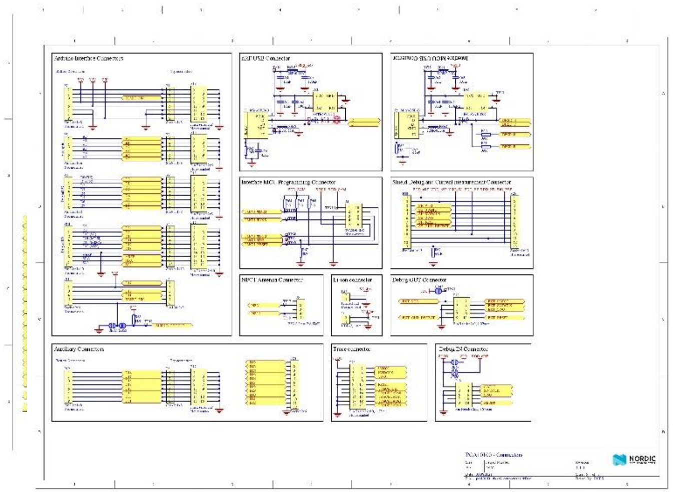




Board	SoC	SoC Series	SoC Family	CPU Core	Architecture
nrf52833dk_nrf52833	nRF52833	nRF52	Nordic nRF	Arm Cortex-M4	Arm

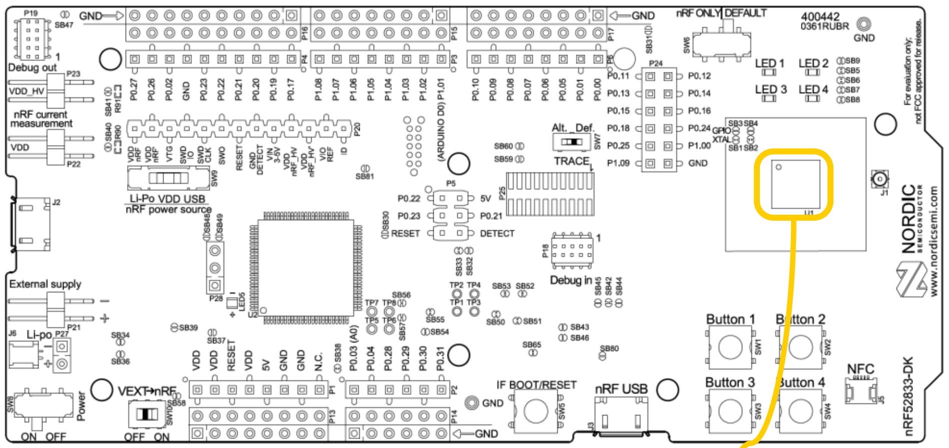
Hardware Support Hierarchy - Example

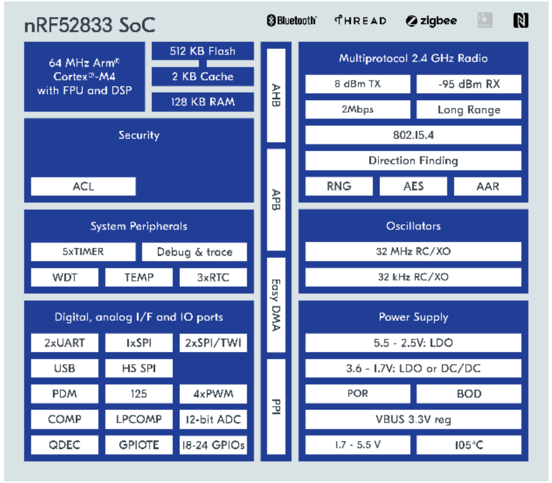




Board	SoC	SoC Series	SoC Family	CPU Core	Architecture
nrf52833dk_nrf52833	nRF52833	nRF52	Nordic nRF	Arm Cortex-M4	Arm

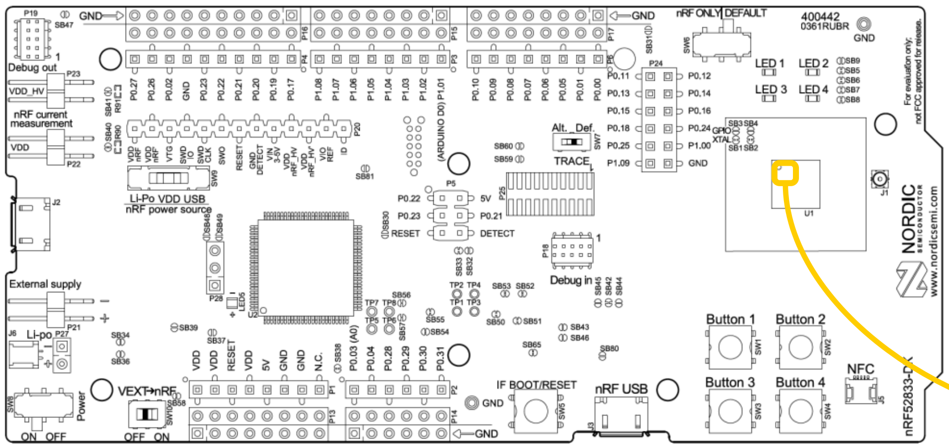
Hardware Support Hierarchy - Example

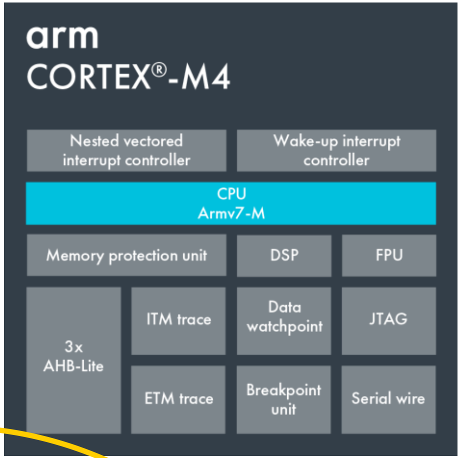




Board	SoC	SoC Series	SoC Family	CPU Core	Architecture
nrf52833dk_nrf52833	nRF52833	nRF52	Nordic nRF	Arm Cortex-M4	Arm

Hardware Support Hierarchy - Example

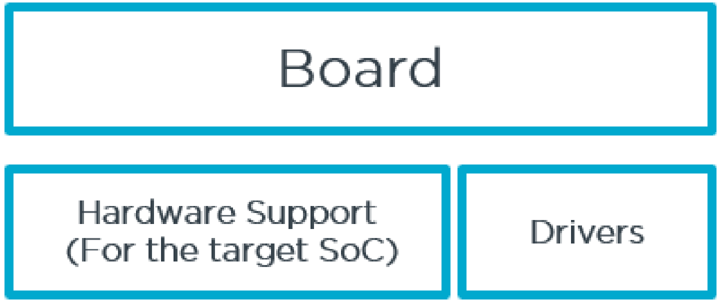


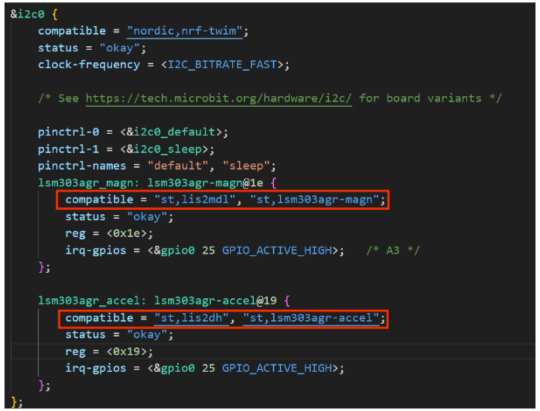


Board	SoC	SoC Series	SoC Family	CPU Core	Architecture
nrf52833dk_nrf52833	nRF52833	nRF52	Nordic nRF	Arm Cortex-M4	Arm

Drivers

- Drivers are needed for some external components connected to the SoC's/SiP's peripherals (Ex: sensors)
- Drivers reside outside the board directory
- Zephyr and nRF Connect SDK comes with a rich set of drivers
- Drivers are specified through the `compatible` property in devicetree
- Examples of boards with sensors
  - micro:bit v2, Thingy:53 , Thingy:91





```
&i2c0 {
    compatible = "nordic,nrf-twim";
    status = "okay";
    clock-frequency = <I2C_BITRATE_FAST>;

    /* See https://tech.microbit.org/hardware/i2c/ for board variants */

    pinctrl-0 = <&i2c0_default>;
    pinctrl-1 = <&i2c0_sleep>;
    pinctrl-names = "default", "sleep";
    lsm303agr_magn: lsm303agr-magn@1e {
        compatible = "st,lis2mdl", "st,lsm303agr-magn";
        status = "okay";
        reg = <0x1e>;
        irq-gpios = <&gpio0 25 GPIO_ACTIVE_HIGH>;   /* A3 */
    };

    lsm303agr_accel: lsm303agr-accel@19 {
        compatible = "st,lis2dh", "st,lsm303agr-accel";
        status = "okay";
        reg = <0x19>;
        irq-gpios = <&gpio0 25 GPIO_ACTIVE_HIGH>;
    };
};
```

Board files

Adding custom board support in nRF Connect SDK

Naming convention

- Name your board a unique name
  - Run `west boards` in CLI for all available boards
- It is recommended to include the target SoC in the board name
- Specify target, If the board has multiple targets
- Your board will have:
  - Human-readable name "DevAcademy nRF52833"
  - Board ID "`devacademy_nrf52833`" (Build target)
  - Kconfig board symbol "`BOARD_DEVACADEMY_NRF52833`"

Where to define your custom board

- In a dedicated directory (out-of-tree board)
  - Suitable for close-source designs
  - Demoed in this webinar
- In a “boards” folder in your application directory (out-of-tree board)
  - Suitable for prototyping/debugging
- Upstream Zephyr (in-tree board)
  - Suitable for a development kit, a prototyping platform, or a reference design (Public)
  - Need to provide documentation
  - Zephyr maintainer(s) need to approve your board PR

Point the build system to an out-of-tree board

- The build system goes to specific folders to look for boards:
  - <nRF Connect SDK Path>/zephyr/boards/arm/
  - <nRF Connect SDK Path>/nrf/boards/arm/
- For out-of-tree boards, you need to update `BOARD_ROOT`
  - From within nRF Connect for VS Code
    - `File->Preferences-> Settings-> Extensions-> nRF Connect-> Board Roots`
  - (Or) Pass the directory location to `west`
    - `west build -b <board name> -- -DBOARD_ROOT=<path to boards>`
  - (Or) Inside `CMakeLists.txt`
    - `list(APPEND BOARD_ROOT ${CMAKE_CURRENT_SOURCE_DIR}/<extra-board-root>)`

Board files

- Board files classifications
  - Mandatory
- Types
  - Devicetree
  - Kconfig

```
boards/<ARCH>/devacademy_nrf52833
├── Kconfig.board
├── Kconfig.defconfig
├── devacademy_nrf52833_defconfig
├── devacademy_nrf52833.dts
└── devacademy_nrf52833-pinctrl.dtsi
```

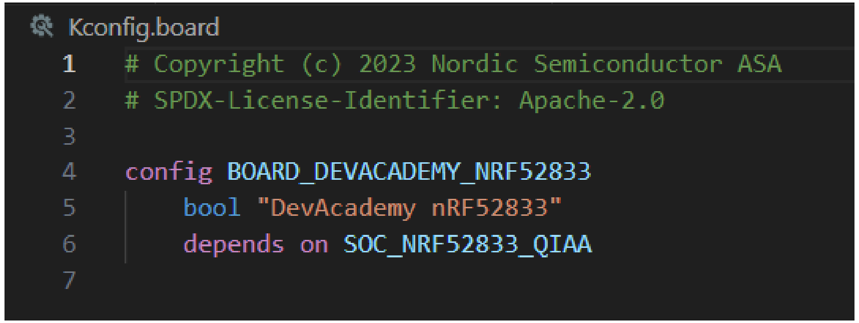
Board files

- Board files classifications
  - Mandatory
  - Optional and special use case
- Types
  - Devicetree
  - Kconfig
  - Cmake
  - C files
  - YAML
  - Documentation

```
boards/<ARCH>/devacademy_nrf52833
├── Kconfig.board
├── Kconfig.defconfig
├── devacademy_nrf52833_defconfig
├── devacademy_nrf52833.dts
├── devacademy_nrf52833-pinctrl.dtsi
├── board.cmake # Used for flash and debug
├── CMakeLists.txt  # Needed in special cases
├── c_files.c  # Needed in special cases
├── doc # Optional
│   ├── devacademy_nrf52833.png
│   └── index.rst
├── Kconfig # Optional to create a board Kconfig options menu
├── devacademy_nrf52833.yaml # Optional for Test Runner (Twister)
├── devacademy_nrf52833_<revision>.conf     # Needed to support multiple hardware revisions
├── devacademy_nrf52833_<revision>.overlay  # Needed to support multiple hardware revisions
└── revision.cmake # Needed to support multiple hardware revisions
└── dts # Optional
    └── bindings
```

Mandatory Kconfig files

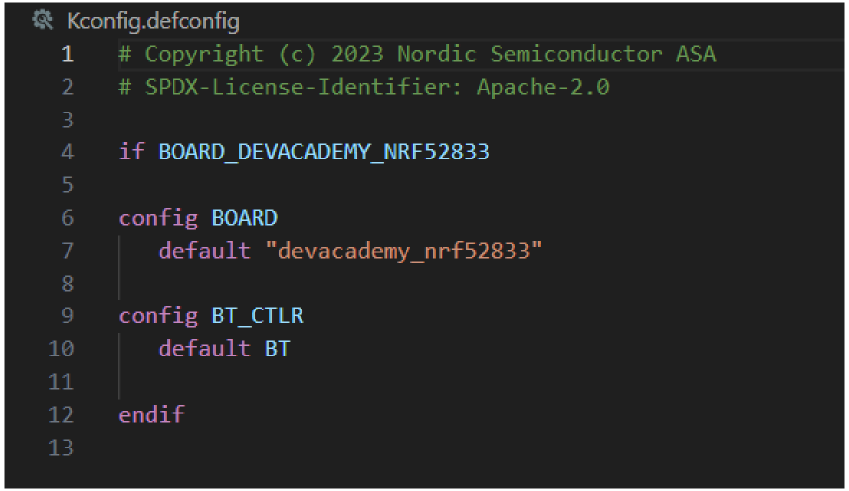
- `Kconfig.board`
  - Makes a Boolean Kconfig Symbol for your board `BOARD_DEVACADEMY_NRF52833`
  - Specifies a dependency to the SoC hardware support layer(through the `depends on` Keyword)


Kconfig.board

```
# Copyright (c) 2023 Nordic Semiconductor ASA
# SPDX-License-Identifier: Apache-2.0

config BOARD_DEVACADEMY_NRF52833
    bool "DevAcademy nRF52833"
    depends on SOC_NRF52833_QIAA

```

Mandatory Kconfig files

- `Kconfig.defconfig`
  - Board-specific default values for Kconfig options
  - Placed inside `IF /ENDIF` pair
  - Sets the Kconfig `BOARD` symbol to your board ID

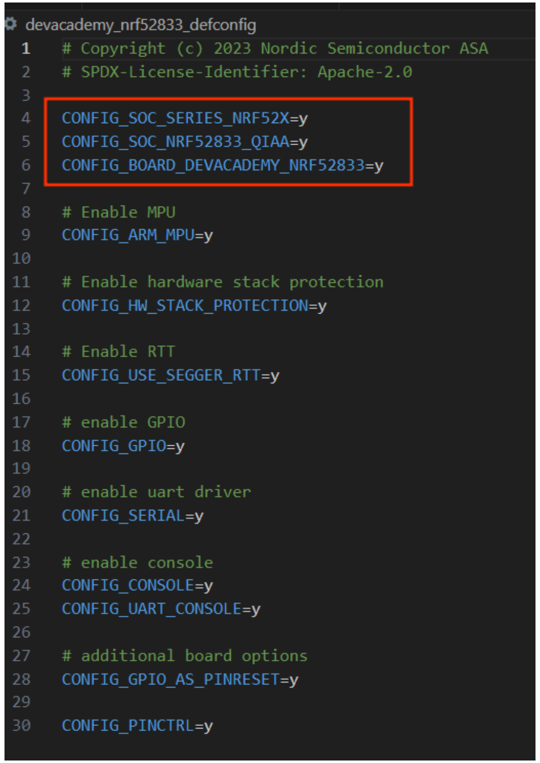


Kconfig.defconfig

```
# Copyright (c) 2023 Nordic Semiconductor ASA
# SPDX-License-Identifier: Apache-2.0

if BOARD_DEVACADEMY_NRF52833

config BOARD
    default "devacademy_nrf52833"

config BT_CTLR
    default BT

endif

```

Mandatory Kconfig files

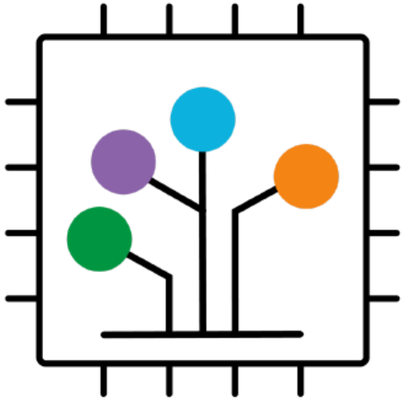
- `devacademy_nrf52833_defconfig`
  - Kconfig fragment merged as-is into the final build of any application built for the specified board
  - Must enable
    - The SoC hardware support layer
    - The SoC Series hardware support layer
    - The Board's Kconfig symbol
  - GPIO/Serial are enabled for convenience
  - Boards must enable the bare minimum
    - It's the application configuration `prj.conf` responsibility to configure what is needed


devacademy_nrf52833_defconfig

```
# Copyright (c) 2023 Nordic Semiconductor ASA
# SPDX-License-Identifier: Apache-2.0

CONFIG_SOC_SERIES_NRF52X=y
CONFIG_SOC_NRF52833_QIAA=y
CONFIG_BOARD_DEVACADEMY_NRF52833=y

# Enable MPU
CONFIG_ARM_MPU=y

# Enable hardware stack protection
CONFIG_HW_STACK_PROTECTION=y

# Enable RTT
CONFIG_USE_SEGGER_RTT=y

# enable GPIO
CONFIG_GPIO=y

# enable uart driver
CONFIG_SERIAL=y

# enable console
CONFIG_CONSOLE=y
CONFIG_UART_CONSOLE=y

# additional board options
CONFIG_GPIO_AS_PINRESET=y

CONFIG_PINCTRL=y
```

Devicetree files

- `devacademy_nrf52833.dts`
  - Board-level devicetree file
  - Describes your board hardware schematic using devicetree syntax.
  - Must include the SoC specific variant devicetree file
- You can structure it into multiple files
- `devacademy_nrf52833-pinctrl.dtsi`
  - Pin-mapping for peripherals
- We will cover this in-depth in the demo

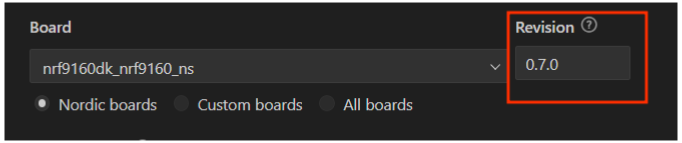


Optional and special use case files

- `board.cmake`: Used for Flash and debug support, if the board has flash or debug support.
- `CMakeLists.txt`: Add source files to be executed Pre- or Post- kernel. This is in case your hardware uses some muxes or needs to be configured in a particular way. (EX: nRF52840 Dongle)
- `doc/index.rst, doc/devacademy_nrf52833.png`: Documentation and a picture of your board. You only need this if you're contributing your board to Zephyr.
- `Kconfig` : Give us the flexibility of creating a board Kconfig menu
- `devacademy_nrf52833.yaml`: A YAML file with miscellaneous metadata used by the Test Runner (Twister)

Board revisons

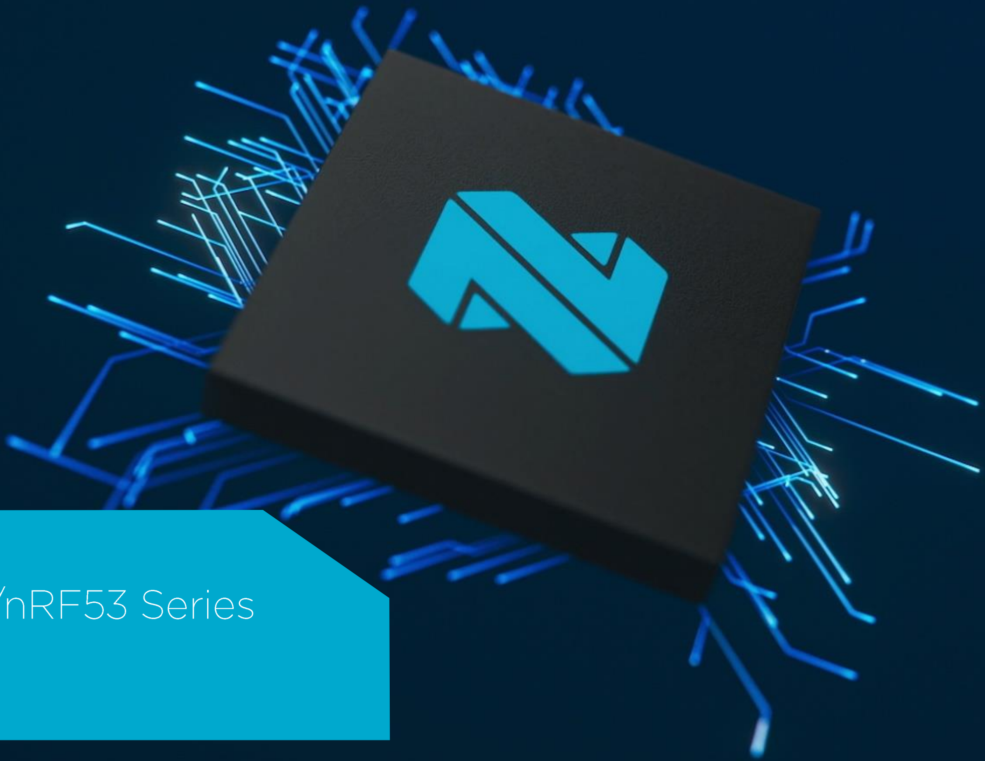
- In case you have a new hardware revision
  - Add the changes as Kconfig fragment and dts overlay
- `devacademy_nrf52833_<revision>.conf`
  - Will be merged into the board Kconfig files.
- `devacademy_nrf52833_<revision>.overlay`
  - Will overlay the board level devicetree .
- revision.cmake
  - Controls how the build system matches the `<board>@<revision>`
    - › `west build nrf9160dk_nrf9160_ns@0.7.0`
- Example: nRF9160 DK
  - 0.7.0 and 0.14.0 revisions


Board: nrf9160dk_nrf9160_ns
Revision: 0.7.0
[● Nordic boards] Custom boards All boards

Available boards definitions

- nRF Connect SDK comes with **many** boards definitions
  - From Zephyr RTOS (in sdk-zephyr repository)
    - Available locally in <nRF Connect SDK Path>/zephyr/boards/arm/
    - Includes Nordic and non-Nordic-based boards
  - From the SDK itself (in sdk-nrf repository)
    - Available locally in <nRF Connect SDK Path>/nrf/boards/arm/
    - Includes only Nordic-based boards
- Mainly for development kits, prototyping platforms, reference designs
- Can be used as a sample/starting point for creating your own custom board

Special considerations for the nRF91/nRF53 Series

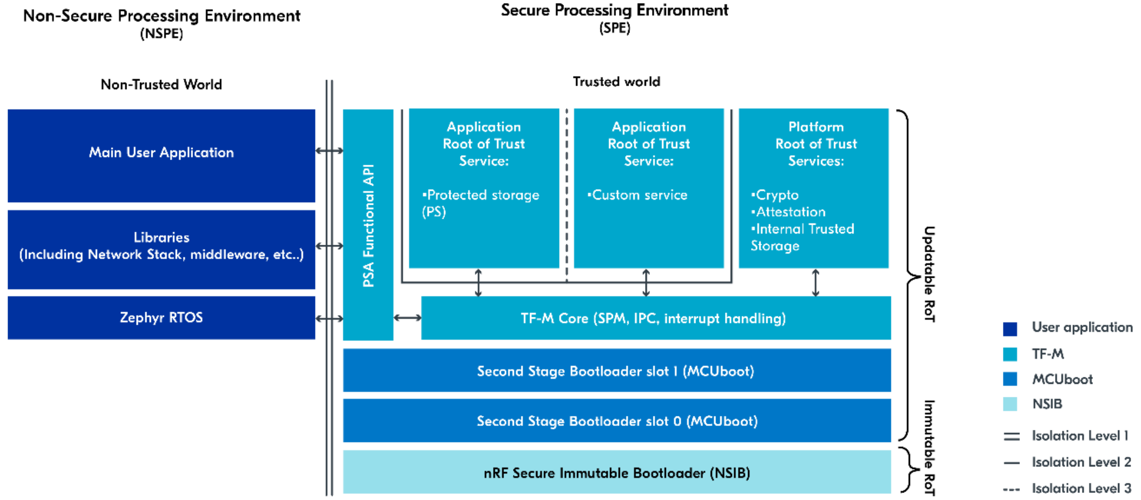


Multi-Core and Trusted Firmware-M (TF-M)

Series	CPU	Architecture	Number of User Programmable Cores	TrustZone Hardware
nRF52	Cortex-M4	Armv7-M	1	No
nRF53	Cortex-M33	Armv8-M	2	Yes
nRF91	Cortex-M33	Armv8-M	1	Yes

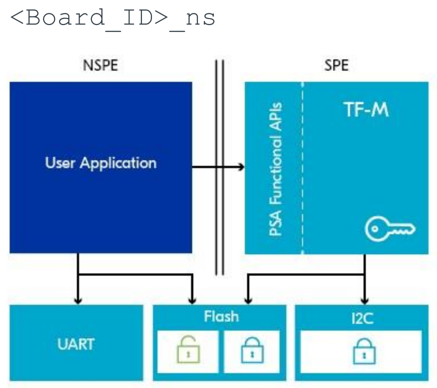
TF-M support

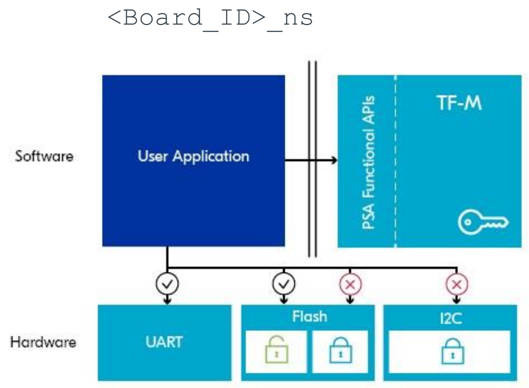
- TF-M enforces “Security by Separation”
- Creates two worlds: Secure and Non-secure Processing Environment
- This creates multi-target boards
  - Two build targets:
    - `<Board_ID>_ns`
    - `<Board_ID>`


Non-Secure Processing Environment (NSPE)
Secure Processing Environment (SPE)
Non-Trusted World
Trusted world
Main User Application
Libraries (Including Network Stack, middleware, etc..)
Zephyr RTOS
PSA Functional API
Application Root of Trust Service: •Protected storage (PS)
Application Root of Trust Service: •Custom service
Platform Root of Trust Services: •Crypto •Attestation •Internal Trusted Storage
TF-M Core (SPM, IPC, interrupt handling)
Second Stage Bootloader slot 1 (MCUboot)
Second Stage Bootloader slot 0 (MCUboot)
nRF Secure Immutable Bootloader (NSIB)
Updatable RoT
Immutable RoT
User application
TF-M
MCUboot
NSIB
Isolation Level 1
Isolation Level 2
Isolation Level 3

Option 1 (With TF-M)

- Option 1: Enforce security by separation by utilizing TF-M
- TF-M runs in the Secure Processing Environment
- Application runs in the **Non-Secure Processing Environment**
- Application build target:
  `<Board_ID>_ns`

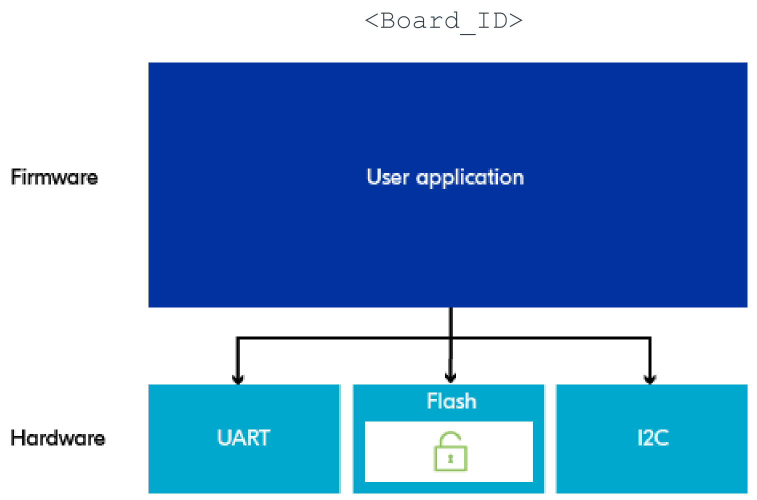




Option 2 (Without TF-M)

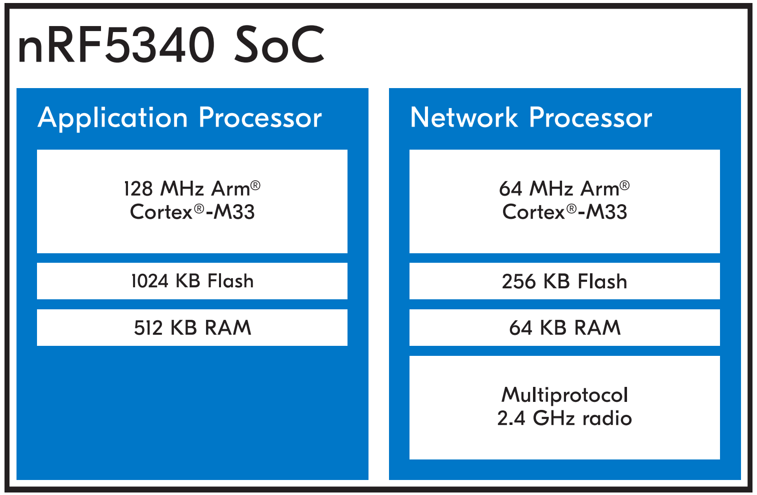
- Option 2: Do not enforce security by separation by having the application run as a single image with full access privileges
- The application is built as a single image without security by separation
- Application build target:

  `<Board_ID>`



Multi-core device

- The nRF5340 SoC has two fully programmable cores
- Application core can run with/without TF-M
  - `<BoardID>_cpuapp` for build targets with TF-M disabled.
  - `<BoardID>_cpuapp_ns` for build targets that have TF-M enabled
- Network core
  - `<BoardID>_cpunet`



Demo placeholder

Extra resources

- Nordic Developers Academy
  - nRF Connect SDK Fundamentals
    - Great start if you are new to nRF Connect SDK or Zephyr
  - Bluetooth Low Energy Fundamentals
  - Cellular IoT Fundamentals
  - More courses are coming very soon
    - Register to get news about new courses!
- Zephyr Board Porting Guide
- Devicetree specifications
- Kconfig – Tips and Best Practices
- Custom Kconfig Preprocessor Functions
  - Useful to get devicetree information into Kconfig files

Q&A