

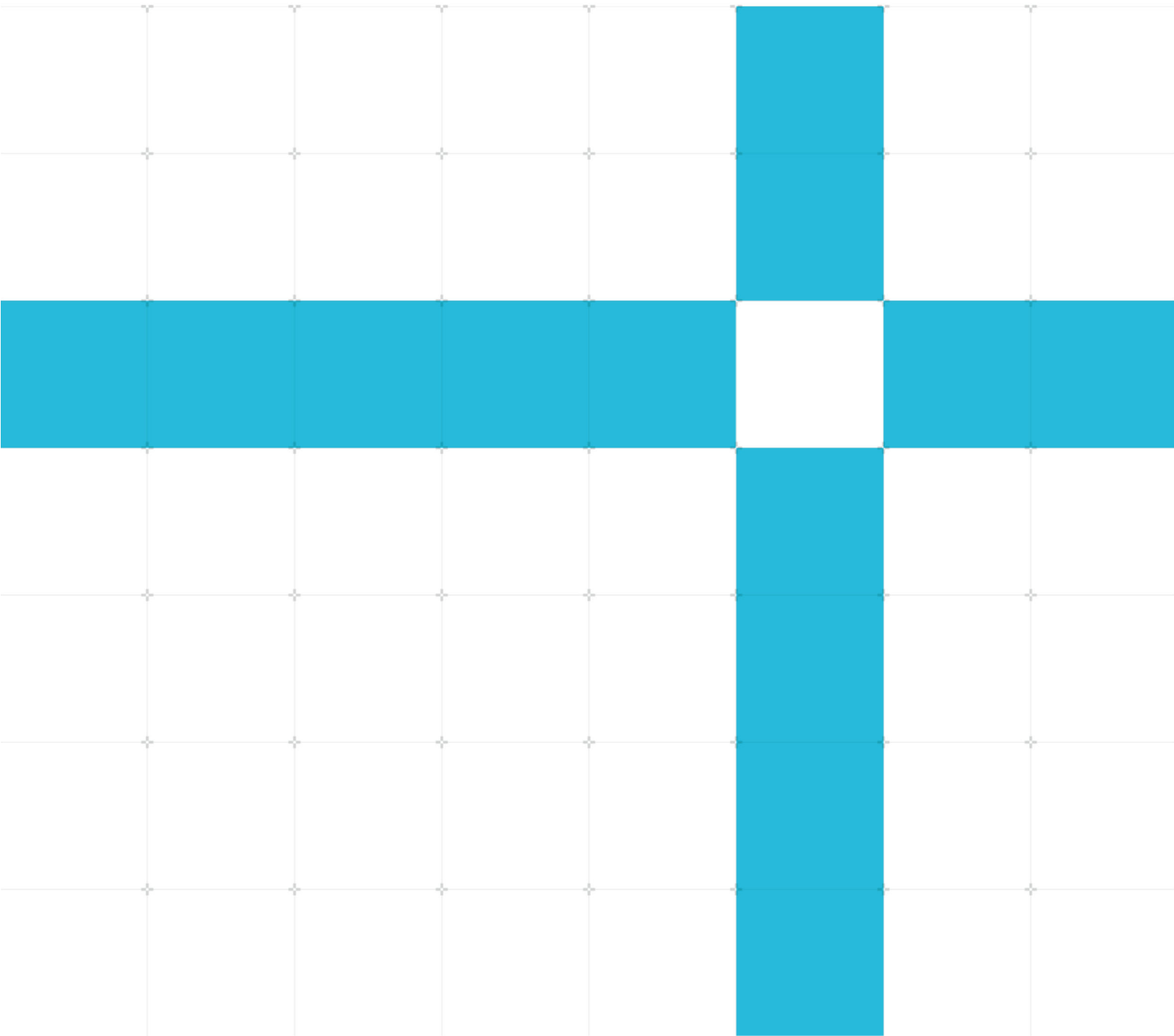


Building OpenWRT for Corstone-1000

Application Note

Non-confidential
Copyright © 2024 Arm Limited (or its affiliates).
All rights reserved.

Document Issue: 1
Document ID: 109672



Release information

Document history

Issue	Date	Confidentiality	Change
1	3/11/2024	Non-confidential	Initial release

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited (“Arm”). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of the document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm’s view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party’s products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or

indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
(PRE-1121-V1.0)

Feedback

Arm welcomes feedback on its products and documentation. To provide feedback on a product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on this document, fill the following survey:
<https://developer.arm.com/documentation-feedback-survey>.

Arm periodically provides updates and corrections to its technical content and Frequently Asked Questions (FAQs) on [Arm Developer](#).

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction.....	6
1.1. Document goals.....	6
1.2. Intended audience.....	6
1.3. Conventions.....	6
1.4. Useful resources	8
2. Corstone-1000	9
2.1. Implementations (FVP/MPS3).....	9
3. Overview of OpenWRT.....	10
4. Setting up the build environment.....	11
4.1. List of targets.....	11
4.2. Tools for building BL1.....	11
4.3. Tools for building U-Boot.....	11
4.4. Tools for building OpenWRT	12
5. Building.....	13
5.1. Building BL1.x	13
5.2. Building U-Boot	13
5.3. Building OpenWRT	13
5.3.1. Building the root file system.....	13
5.3.2. Building the kernel.....	15
6. Running OpenWRT on FVP.....	18
6.1. Installing the firmware	18
6.2. Installing OpenWRT.....	18
6.3. Running the Demo application.....	18
7. Running OpenWRT on MPS3.....	20
7.1. Installing the firmware	20
7.2. Installing OpenWRT.....	20
7.3. Running the Demo application.....	21

1. Introduction

1.1. Document goals

The goal of this document is to guide the reader through the build and installation process of OpenWrt on the Corstone-1000 platform, both on the Fixed Virtual Platform (FVP) software and on the MPS3 board hardware.

1.2. Intended audience

This application note is intended for people who are familiar with

- OpenWrt operating system
 - Have a general understanding of OpenWrt and some hands-on experience
 - A general understanding of network routers
 - Building OpenWrt on Linux using Buildroot
- Linux
 - C, C++ development using the **GNU Compiler Collection** toolchain
- the Corstone-1000 platform
 - Have access to the Yocto based build for Corstone-1000
 - Have a general understanding of Yocto

1.3. Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: <https://developer.arm.com/glossary>.





This document uses the following terms and abbreviations.



Terms and abbreviations

Term	Meaning
M0+	Cortex M0+ 32 bit CPU used as a general purpose CPU in microcontrollers, used as a secure enclave and startup control CPU for Corstone-1000 platform
A35	Cortex A35 64 bit CPU used as computation workload and communications processor for the Corstone-1000 platform
M3	Cortex M3 32 bit CPU used for running timing input-output critical tasks for the Corstone-1000 platform
Arm-none-eabi-gcc	A GNU GCC compiler chain compiled to produce cross compiled output for 32 bit ARM CPUs
Aarch64-linux-gnu-gcc	A GNU GCC compiler chain compiled to produce cross compiled output for a 64 bit ARM CPU (in this case the A35 CPU)

Typographical conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Citations.
bold	Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <code>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></code>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
 Caution	We recommend the following. If you do not follow these recommendations your system might not work.
 Warning	Your system requires the following. If you do not follow these requirements your system will not work.
 Danger	You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.
 Note	This information is important and needs your attention.

Convention	Use
 Tip	This information might help you perform a task in an easier, better, or faster way.
 Remember	This information reminds you of something important relating to the current content.

1.4. Useful resources

This document contains information that is specific to this product. See the following resources for other relevant information.

- Arm Non-Confidential documents are available on developer.arm.com/documentation. Each document link in the tables below provides direct access to the online version of the document.
- Arm Confidential documents are available to licensees only through the product package.

Arm products	Document ID	Confidentiality
Arm MPS3 FPGA Prototyping Board Getting Started Guide	107789	Non-Confidential
Arm® Corstone™-1000 Technical Overview	102360	Non-Confidential
Fast Models Fixed Virtual Platforms (FVP) Reference Guide	100966	Non-Confidential

2. Corstone-1000

The Corstone-1000 includes the Corstone SSE-710 subsystem, which is a flexible subsystem designed to provide a secure solution for rich Internet of Things (IoT) applications based on Arm® supported Cortex®-A processors.

The Corstone-1000 platform is:

- Secure from the ground up, easily certifiable with multiple security standards, with remote management and over the air upgrade. It includes a separate Cortex-M0+ processing hardware node to provide a secure enclave.
- Provides real-time sensor support, by a separate real-time subsystem, that is free from the constraints imposed by non-RTOS operating systems. One or more separate Cortex-M3 processing nodes are included for this purpose.
- Provides extended processing capabilities to be able to run CPU and memory intensive endpoint sensor data processing, run AI workloads, image processing tasks and a standard network software stack. One or more Cortex-A32/35/53 cores are present for this purpose.

2.1. Implementations (FVP/MPS3)

The two development platforms for the Corstone-1000 platform are:

- The FVP software fast model, that runs on Windows and Linux.
- The FPGA based development platform, which is tested on and runs on the Arm MPS3 development board.

Significant differences exist between them, for example, the FVP software implementation simulates different network interfaces, while the MPS3 board has physical LAN 911x based interfaces present.

3. Overview of OpenWRT

Due to its historic roots as an operating system oriented at network devices, OpenWRT runs very well on:

- Resource constrained systems, provides useful functionality with less than 100MB of RAM and 100Mhz of CPU speed.
- Devices where physical access is an issue. As opposed to desktop or server systems the following operations are usually performed remotely, so they must be designed to fail safe:
 - Configuration
 - Repair
 - Diagnosis
 - Update
- Devices that are part of management clusters. All the configuration data sources are geared towards key-value pair configurations, meaning that it integrates naturally with industrial scale SNMP configuration tools.

4. Setting up the build environment

4.1. List of targets

For both the FVP and the MPS3 board implementations, the following items need to be compiled and packaged:

- Startup code and secure enclave implementation for the `boot cpu/secure enclave`.
- Startup code for the 64-bit CPU.
- An OpenWRT image, the subject of this application note.

The image for the 32-bit M0+ CPU is loaded as a separate image by both the FVP and the MPS3 implementations.

The images for the 64-bit startup, and 64-bit operating system are combined into one virtual disk image conforming to the GPT partition standard.

4.2. Tools for building BL1

The first phase bootloader is built for an ARM32 CPU, which is an M0+ CPU which performs the system initialization at startup and provides a secure enclave to the system while in operation.

For this an `arm-none-eabi` compile and build chain is the natural candidate. Although `arm-linux-gnueabi` will work, if the linker build scripts are set up to avoid loading the `crt0` and `gnu libc` defaults that would rely on calls to a Linux kernel.

The build procedure in this document uses Yocto to build the `arm-none-eabi-gcc` build chain. This toolchain is set up to link with the `mcuboot` and `arm-libc` libraries.

4.3. Tools for building U-Boot

U-Boot runs on the 64-bit CPU, it has to be built by an `aarch64-none` or an `aarch64-linux-gcc` toolchain.

The instructions in this document rely on the Yocto build system to compile an `aarch64-linux-gnu` build chain. Yocto makes sure that no `libc` libraries are used. These libraries would rely on the presence of a Linux environment. The Linux environment is not available while the U-Boot bootloader executes.

U-Boot comes with its own build and packaging system, these instructions rely on Yocto build system to configure and use the U-Boot build and packaging system.

4.4. Tools for building OpenWRT

OpenWrt has a build system based on buildroot. It has very few external dependencies, relies on a locally built `gcc cross-compiler` (aarch64-gcc).

On a standard Linux distribution, it is recommended that you use the system installed `aarch64-gcc` compiler, as it saves a lot of compilation time and disk space with no downsides. It is the same compiler as the one OpenWrt would generate. Versions are not an issue as gcc is a very mature product.



The instructions contained in this document work with the default settings, letting the OpenWrt build system generate a local toolchain. They are not the most efficient solution, but this minimizes both the length of this document and external requirements.

5. Building

5.1. Building BL1.x

The first stage bootloader is built and packaged separately by the Yocto based installation, following the instructions on <https://corstone1000.docs.arm.com/en/latest/user-guide.html#building-the-software-stack>.

5.2. Building U-Boot

The U-Boot bootloader stages are also built by the Yocto based build system by following the instructions on: <https://corstone1000.docs.arm.com/en/latest/user-guide.html#building-the-software-stack>. It is packaged into a disk image with the poky-linux based embedded Linux, which will have to be swapped out for the OpenWrt Linux distribution later.

5.3. Building OpenWRT

The OpenWrt build system can optionally build:

- The build tools for the build.
- The root file system that contains the user space utilities and configuration files.
- A kernel and the kernel modules and the configuration for it.
- Any combination of these.

In this document the build tools are generated by the OpenWrt build system, the user space utilities are built by the OpenWrt build system, using the build tools that were just generated. The kernel is built separately.

5.3.1. Building the root file system

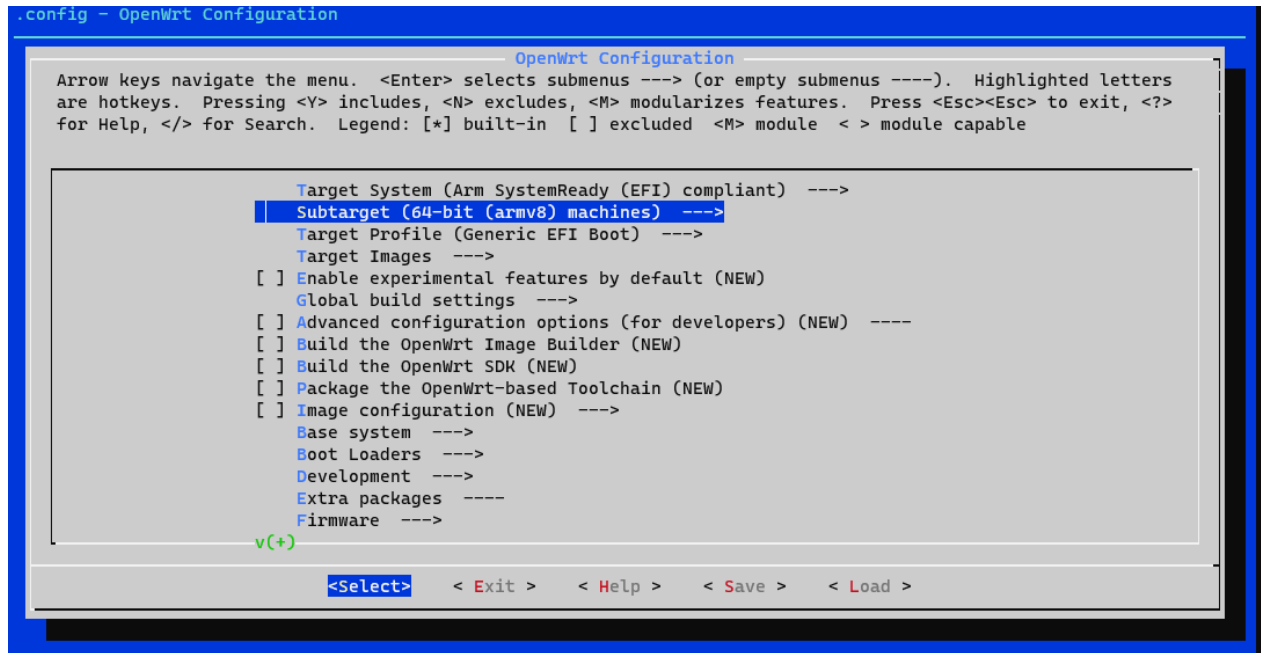
1. Get the OpenWRT sources:

```
git clone https://github.com/openwrt/openwrt.git
```

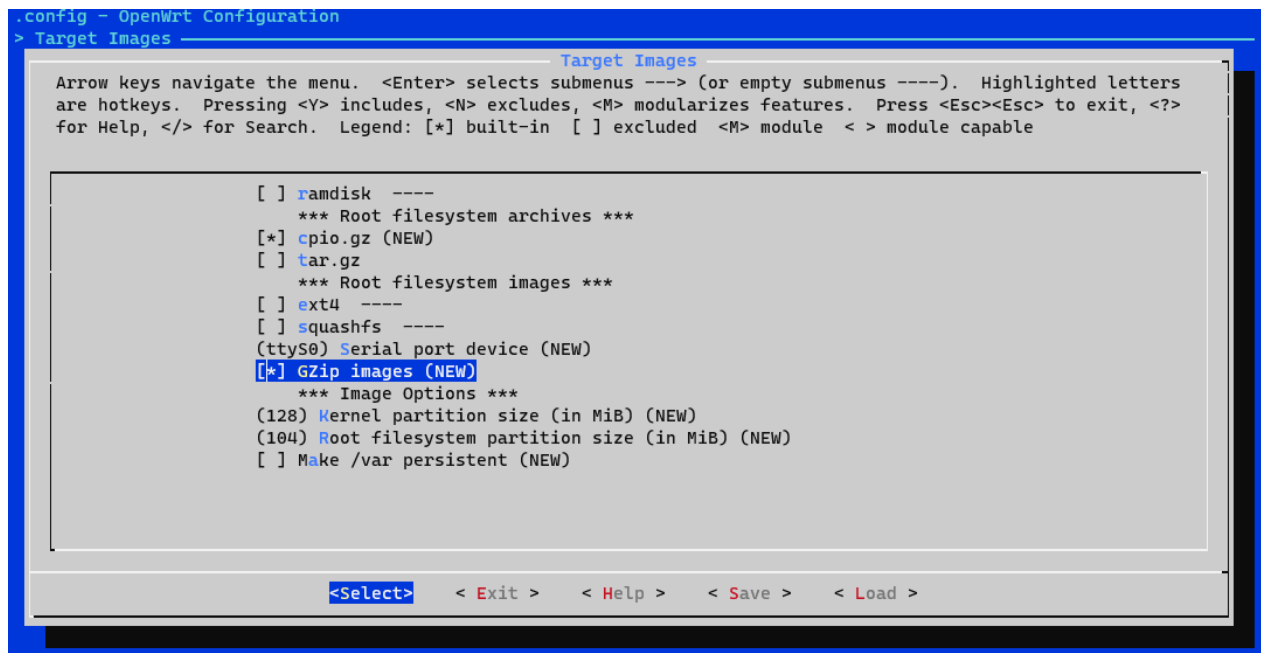
2. In the openwrt directory, either use the referenced .config file or manually enter:

```
make menuconfig
```

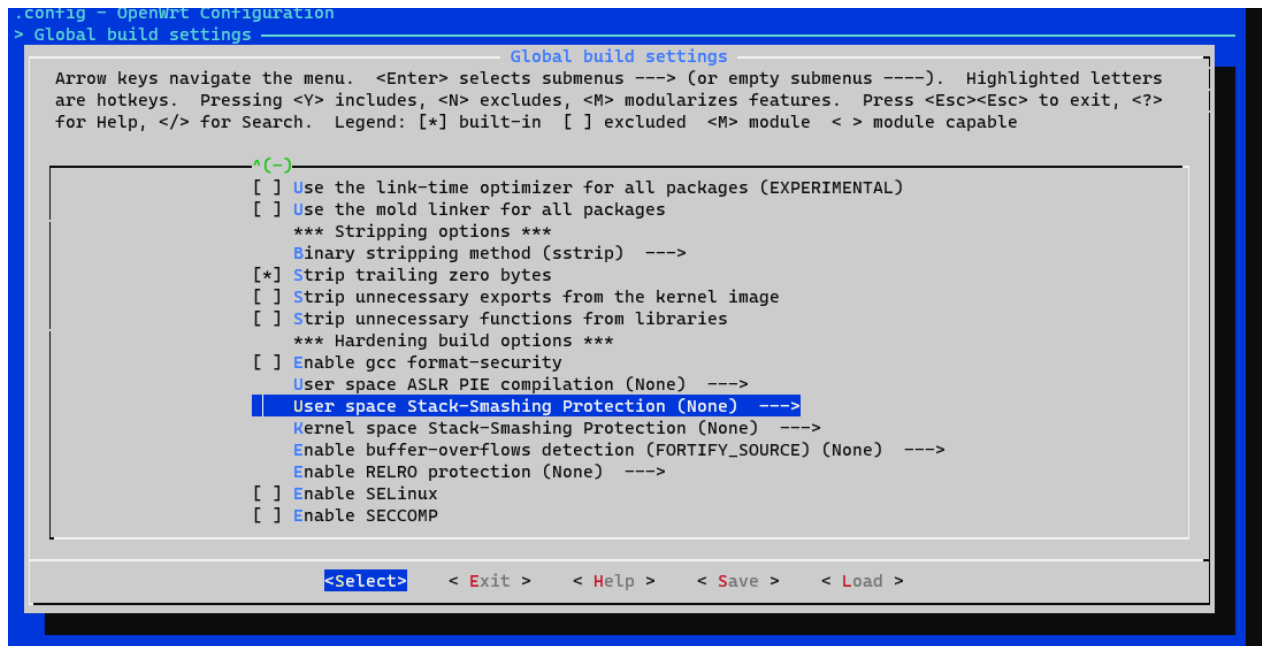
3. Verify that the target system is ARM SystemReady (EFI) compliant, the subtarget 64 bit (armv8).



4. Build only the filesystem and the user space utilities



5. Exclude the hardening options for now, some of them are not compatible with the TF-A – OP-TEE hypervisor because they try to gain access to the run levels that TF-A already uses:



5.3.2. Building the kernel

The kernel sources can be obtained from www.kernel.org. In the kernel source directory.

```
make ARCH=arm64 defconfig
make ARCH=arm64 menuconfig
```

This command sets up a kernel that will be compiled for an arm64 system.



If you ever run `make menuconfig` without the `ARCH=arm64` bit, run `make distclean` and start over, the `.config` file will be permanently messed up, it will have Intel related config parameters in there, messing up the build.

1. In `menuconfig`, make sure you have enabled:
 - o ARM AMBA PL011 serial
 - o PL011 serial console
 - o `initramfs/initrd` support
2. Ensure that the `initramfs` source file points to the `hello_root.cpio.gz` that you have created. It will be packaged with the kernel image. The boot options must be `root=/dev/ram0` for the kernel to prevent from mounting a more permanent root.
3. For the MPS3 platform, it is important to include drivers for the LAN911x network chip and the NXP1760 USB hub. These are not part of the Corstone-1000 platform, but are physically present on the MPS3 board:

```

.config - Linux/arm64 6.7.4 Kernel Configuration
> Device Drivers > Network device support > Ethernet driver support
                                     Ethernet driver support
Arrow keys navigate the menu. <Enter> selects submenus ----> (or empty submenus ----). Highlighted letters
are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
[ ] Renesas devices
[ ] Rocker devices
[ ] Samsung Ethernet devices
[ ] SEEQ devices
[ ] Solarflare devices
[*] SMC (SMSC)/Western Digital devices
[ ] SMC 91C9x/91C1xxx support
[*] SMC LAN911x/LAN921x families embedded ethernet support
[ ] Socionext ethernet drivers
[ ] STMicroelectronics devices
[ ] Synopsys devices
[ ] Vertexcom devices
[ ] VIA devices
[ ] Wangxun devices
[ ] WIZnet devices
[ ] Xilinx devices

<Select> < Exit > < Help > < Save > < Load >

```

```

> Device Drivers > USB support
                                     USB support
Arrow keys navigate the menu. <Enter> selects submenus ----> (or empty submenus ----). Highlighted letters
are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
*** USB Imaging devices ***
[ ] USB Mustek MDC800 Digital Camera support
[ ] USB/IP support
*** USB dual-mode controller drivers ***
[ ] Cadence USB Support
[ ] Inventra Highspeed Dual Role Controller
[ ] DesignWare USB3 DRD Core Support
[ ] DesignWare USB2 DRD Core Support
[ ] ChipIdea Highspeed Dual Role Controller
[*] NXP ISP 1760/1761/1763 support
    ISP1760 Mode Selection (Dual Role mode) ---->
*** USB port drivers ***
[*] USB Serial Converter support ---->
*** USB Miscellaneous drivers ***
[ ] EMI 6|2m USB Audio interface support
[ ] EMI 2|6 USB Audio interface support
v(+)

<Select> < Exit > < Help > < Save > < Load >

```

4. Verify that the root filesystem of the kernel should be the file generated in [_Building_the_root](#) Error! Reference source not found. Error! Reference source not found. Error! Reference source not found.:


```
.config - Linux/arm64 6.5.10 Kernel Configuration
> General setup

General setup
Arrow keys navigate the menu. <Enter> selects submenus ---- (or empty submenus ----). Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
[ ] Printk indexing debugfs interface
Scheduler features ----
[ ] Memory placement aware NUMA scheduler
[ ] Control Group support ----
[ ] Namespaces support ----
[ ] Checkpoint/restore support
[ ] Automatic process group scheduling
[ ] Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
(/home/peter/openwrt/bin/targets/arm64/armv8/openwrt-arm64-armv8-rootfs.cpio.gz) Initram
(1000) User ID to map to 0 (user root)
(1000) Group ID to map to 0 (group root)
[*] Support initial ramdisk/ramfs compressed using gzip
[ ] Support initial ramdisk/ramfs compressed using bzip2
[ ] Support initial ramdisk/ramfs compressed using LZMA
v(+)

<Select> < Exit > < Help > < Save > < Load >
```

5. Compile the kernel:

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
```

6. Running OpenWRT on FVP

6.1. Installing the firmware

Yocto puts the generated firmware in the `build/tmp/deploy/images/corstone1000-fvp/` directory.

To run the unmodified firmware, enter:

```
./meta-arm/scripts/runfvp --terminals=xterm ./build/tmp/deploy/images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf
```

6.2. Installing OpenWRT

The newly generated OpenWrt image is installed by mounting the disk image file on a loopback device and changing the poky Linux installation to the OpenWrt build.

1. The `.wic` file produced by Yocto is a 32MB disk image.

Mount it as a block device by entering:

```
losetup -fP ./build/tmp/deploy/images/corstone-1000-fvp/corstone1000-image-corstone1000-fvp-20231102122806.rootfs.wic
```

2. The 9th partition contains the Linux image. Overwrite it with the following shell commands:

```
cat Image.gz > /dev/loop0p9  
losetup -d /dev/loop0
```

3. To release the block device. Load the binary onto the MPS3 board, and the newly built kernel will boot up and run the init program from the newly built root file system.

6.3. Running the Demo application

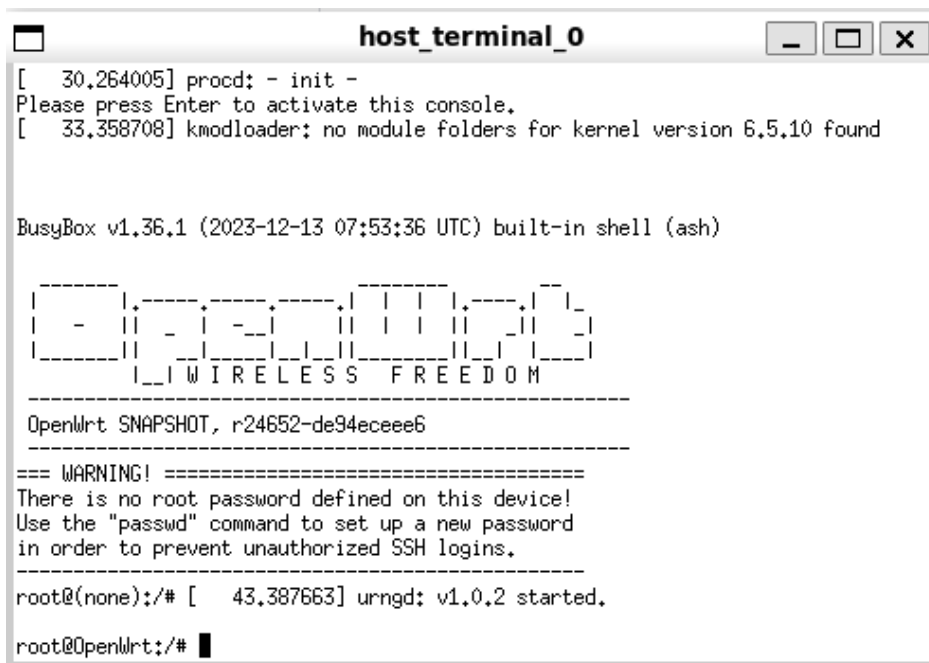
From the build directory, enter:

```
./meta-arm/scripts/runfvp --terminals=xterm ./build/tmp/deploy/images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf
```

The bootloader loads and starts the OpenWrt operating system. The system console is now connected to the FVP telnet session.



OpenWrt requires you to press Enter to activate a console.



7. Running OpenWRT on MPS3

7.1. Installing the firmware

When loaded with the AN550 or AN552 image, the MPS3 board emulates the Corstone-1000 IP. It loads software images according to the images.txt file on the SD card that is in the first SD card slot of the MPS3 board.

Load BI1.bin and the cs1000.bin on the SD card, reference them in the IMAGES.TXT file.

7.2. Installing OpenWRT

1. Install the newly generated OpenWrt image by mounting the disk image file on a loopback device and changing the poky Linux installation to the OpenWrt build.

The .wic file produced by Yocto is a 32MB disk image. Mount it as a block device by

```
losetup -fP cs1000.bin
```

2. Overwrite the 9th partition as it contains the Linux image.

```
cat Image.gz > /dev/loop0p9
losetup -d /dev/loop0
```

3. To release the block device, load this binary onto the MPS3 board, and your newly built kernel will boot up and run OpenWrt, providing a login prompt on the primary serial console.

The images.txt file must reference the newly created images, copied onto the SD card:

```
;*****
;      Preload port mapping                      *
;*****
; PORT 0 & ADDRESS: 0x00_0000_0000 QSPI Flash (XNVM) (32MB)
; PORT 0 & ADDRESS: 0x00_8000_0000 OCVm (DDR4 2GB)
; PORT 1      Secure Enclave (M0+) ROM (64KB)
; PORT 2      External System 0 (M3) Code RAM (256KB)
; PORT 3      Secure Enclave OTP memory (8KB)
; PORT 4      CVM (4MB)
;*****

[IMAGES]
TOTALIMAGES: 3      ;Number of Images (Max: 32)

IMAGE0PORT: 1
IMAGE0ADDRESS: 0x00_0000_0000
```

```

IMAGE0UPDATE: RAM
IMAGE0FILE: \SOFTWARE\bl1.bin

IMAGE1PORT: 0
IMAGE1ADDRESS: 0x00_0000_0000
IMAGE1UPDATE: AUTOQSPI
IMAGE1FILE: \SOFTWARE\cs1000.bin

IMAGE2PORT: 2
IMAGE2ADDRESS: 0x00_0000_0000
IMAGE2UPDATE: RAM
IMAGE2FILE: \SOFTWARE\es0.bi

```

7.3. Running the Demo application

Powering up the MPS3 board with this SD card setup results in the image being loaded into the system memory and OpenWrt starting up with the console on `ttyS0`. The console is connected to a USB-serial port.

Root login is possible without a password.



OpenWrt requires you to press Enter to activate a console.

[illegible]