



How to build Linux kernel user space tools



Contents

1. How to build Linux kernel user space tools	3
2. STM32MP1 Developer Package	8



A quality version of this page, approved on 22 April 2020, was based off this revision.

Contents

1 Article purpose	4
2 Introduction	5
3 Installing the trace and debug tool on your target board	6
3.1 Using the STM32MPU Embedded Software distribution	6
3.1.1 Developer Package	6
3.1.2 Distribution Package	7
4 References	8



1 Article purpose

This article provides the basic information needed to build the user space tools available on the Linux[®] kernel.



2 Introduction

The Linux kernel provides some user-space tools that are available in the tools directory ^[1] of the source tree.

These tools are not compiled by default when compiling the Linux kernel for the target board. They can be compiled independently, depending on the user's needs.



3 Installing the trace and debug tool on your target board

3.1 Using the STM32MPU Embedded Software distribution

3.1.1 Developer Package

Prerequisites, please ensure:

- the SDK is installed
- the SDK is started up
- the Linux kernel is installed

The available user space tools can be listed by using the following commands in the Linux kernel source root path:

```
PC $> cd <Linux_kernel_source_path>
PC $> make tools/help O="<Linux_kernel_build_dir>" (optional)
Possible targets:

acpi           - ACPI tools
cgroup        - cgroup tools
cpupower      - a tool for all things x86 CPU power
firewire      - the userspace part of nosy, an IEEE-1394 traffic sniffer
freefall      - laptop accelerometer program for disk protection
gpio          - GPIO tools
hv            - tools used when in Hyper-V clients
iio           - IIO tools
kvm_stat      - top-like utility for displaying kvm statistics
leds          - LEDs tools
liblockdep    - user-space wrapper for kernel locking-validator
bpf           - misc BPF tools
perf          - Linux performance measurement and analysis tool
selftests     - various kernel selftests
spi           - spi tools
objtool       - an ELF object analysis tool
tmon          - thermal monitoring and tuning tool
turbostat     - Intel CPU idle stats and freq reporting tool
usb           - USB testing tools
virtio        - vhost test module
vm            - misc vm tools
wmi           - WMI interface examples
x86_energy_perf_policy - Intel energy policy tool
```

Note: some tools are made for specific platforms (ARM, x86, RISC, and so on), so cannot be used on STM32MPU systems

The following basic steps must be done :

- Compiling the application:
 - Refer to <Linux kernel installation directory>/README.HOW_TO.txt helper file to know how to compile (the latest version of this helper file is also available in GitHub: README.HOW_TO.txt).
 - Ensure at least that the kernel configuration file is generated (.config) (information available in README.HOW_TO file previously mentioned)
 - Compile the expected **tool (i.e. iio, spi...)**

```
PC $> cd <Linux_kernel_source_path>/tools
PC $> make <tool> [O=<Linux_kernel_build_dir>]
```



Note: The 'O' option can be used to specify the output directory

- Deploying the application on a board:
 - The binary is generated in the directory path `<Linux_kernel_build_dir>/<tool>`
 - Push it onto the board.

```
PC $> scp <tool_binary> root@<board_ip_address>: /<dest_path>
```

PS: please ensure that the `<dest_path>` is known in the `$PATH` to execute the tool binary from anywhere on the target board.

3.1.2 Distribution Package

There is currently no recipe to build the Linux kernel user space tools, so the [Developer package](#) has to be used.



4 References

- /tools

Linux[®] is a registered trademark of Linus Torvalds.

Central processing unit

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Industrial I/O Linux[®] subsystem

Executable and linkable file

Stable: 31.03.2021 - 16:33 / Revision: 31.03.2021 - 09:53

A quality version of this page, approved on 31 March 2021, was based off this revision.

This article describes how to get and use the **Developer Package** of the **STM32MPU Embedded Software** for any development platform of the **STM32MP1 family** (STM32MP15 boards), in order to modify some of its pieces of software, or to add applications on top of it.

It lists some **prerequisites** in terms of knowledges and development environment, and gives the **step-by-step** approach to download and install the STM32MPU Embedded Software components for this Package.

Finally, it proposes some guidelines to upgrade (add, remove, configure, improve...) any piece of software.

Contents

1 Developer Package content	10
2 Developer Package step-by-step overview	11
3 Checking the prerequisites	12
3.1 Knowledges	12
3.2 Development setup	12
4 Installing the Starter Package	15
5 Installing the components to develop software running on Arm Cortex-A (OpenSTLinux distribution) .	
16	
5.1 Installing the SDK	16
5.1.1 Starting up the SDK	18
5.2 Installing the Linux kernel	18
5.2.1 Downloading the Linux kernel	18
5.2.2 Building and deploying the Linux kernel for the first time	19
5.3 Installing the U-Boot	20
5.3.1 Downloading the U-Boot	20
5.3.2 Building and deploying the U-Boot for the first time	24
5.4 Installing the TF-A	24
5.4.1 Downloading the TF-A	24
5.4.2 Building and deploying the TF-A for the first time	28
5.5 Installing the OP-TEE	28
5.5.1 Downloading the OP-TEE	28



5.5.2 Building and deploying the OP-TEE for the first time	32
5.6 Installing the debug symbol files	32
5.6.1 Downloading the debug symbol files	32
5.6.2 Using the debug symbol files	35
6 Installing the components to develop software running on Arm Cortex-M4 (STM32Cube MPU Package)	36
6.1 Installing STM32CubeIDE	36
6.2 Installing the STM32Cube MPU Package	36
7 Developing software running on Arm Cortex-A7	39
7.1 Modifying the Linux kernel	39
7.2 Adding external out-of-tree Linux kernel modules	39
7.3 Adding Linux user space applications	40
7.4 Modifying the U-Boot	40
7.5 Modifying the TF-A	41
7.6 Modifying the OP-TEE	41
8 Developing software running on Arm Cortex-M4	42
8.1 How to create a Cube project from scratch or open/modify an existing one from STM32Cube MPU package	42
9 Fast links to essential commands	43
10 How to go further?	44

1 Developer Package content

If you are not yet familiar with the **STM32MPU Embedded Software** distribution and its **Packages**, please read the following articles:

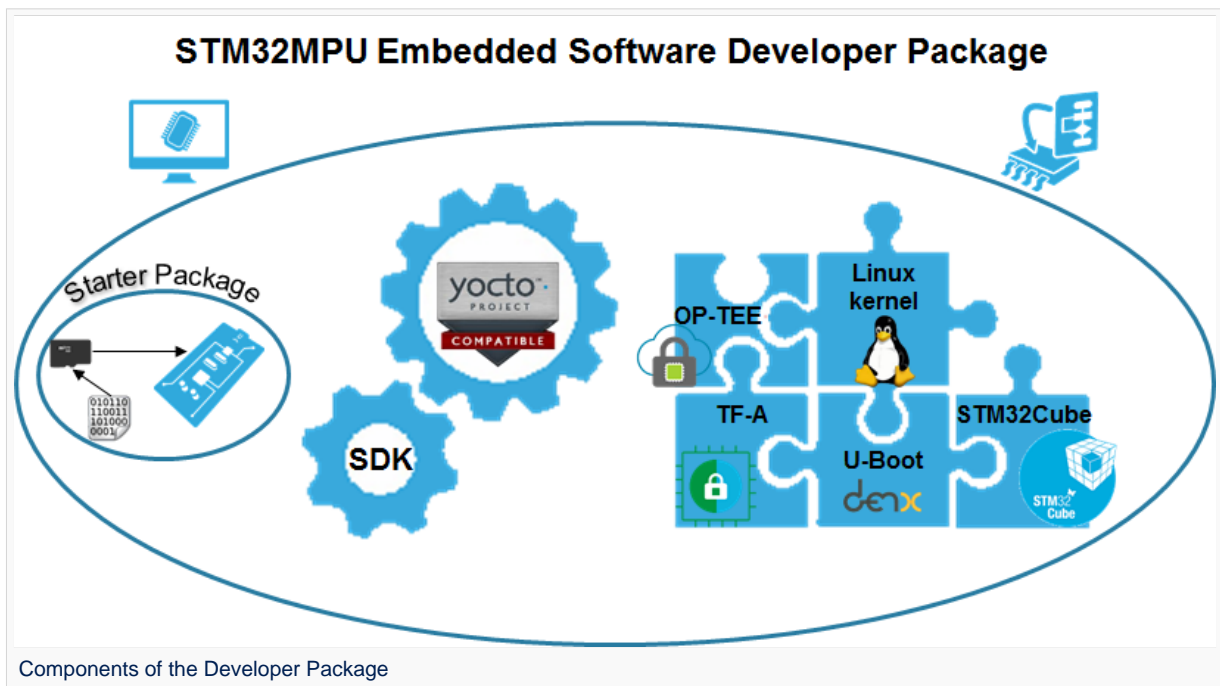
- Which STM32MPU Embedded Software Package better suits your needs (and especially the Developer Package chapter)
- STM32MPU Embedded Software distribution

If you are already familiar with the Developer Package for the STM32MPU Embedded Software distribution, the fast links to essential commands might interest you.

To sum up, this **Developer Package** provides:

- for the **OpenSTLinux distribution** (development on Arm®Cortex®-A processor):
 - the **software development kit** (SDK), based on Yocto SDK, for cross-development on an host PC
 - the following pieces of software in **source code**:
 - Linux® kernel
 - U-Boot
 - Trusted Firmware-A (TF-A)
 - optionally, Open source Trusted Execution Environment (OP-TEE)
 - the **debug symbol files** for Linux® kernel, U-Boot and TF-A
- for the **STM32Cube MPU Package** (development on Arm®Cortex®-M processor):
 - all pieces of software (BSP, HAL, middlewares and applications) in **source code**
 - the **integrated development environment (IDE)** (STM32CubeIDE)

Note that, the application frameworks for the OpenSTLinux distribution are not available as source code in this Package.





2 Developer Package step-by-step overview

The steps to get the STM32MPU Embedded Software Developer Package ready for your developments, are:

Checking the prerequisites

Installing the Starter Package for your board

Installing the components to develop software running on Arm[®]Cortex[®]-A (OpenSTLinux distribution)

Installing the SDK (**mandatory** for any development on Arm[®]Cortex[®]-A)

Installing the Linux kernel (**mandatory only** if you plan to modify the Linux kernel or to add external out-of-tree Linux kernel modules)

Installing the U-Boot (**mandatory only** if you plan to modify the U-Boot)

Installing the TF-A (**mandatory only** if you plan to modify the TF-A)

Installing the debug symbol files (**mandatory only** if you plan to debug Linux[®] kernel, U-Boot or TF-A with GDB)

Installing the components to develop software running Arm Cortex-M (STM32Cube MPU Package)

Installing STM32CubeIDE (**mandatory** for any development on Arm[®]Cortex[®]-M)

Installing the STM32Cube MPU Package (**mandatory only** if you plan to modify the Cube firmware)

Once these steps are achieved, you are able to:

- develop software running on Arm Cortex-A
 - Modifying the Linux kernel
 - Adding external out-of-tree Linux kernel modules
 - Adding Linux user space applications
 - Modifying the U-Boot
 - Modifying the TF-A
- develop software running on Arm Cortex-M4



3 Checking the prerequisites

3.1 Knowledges

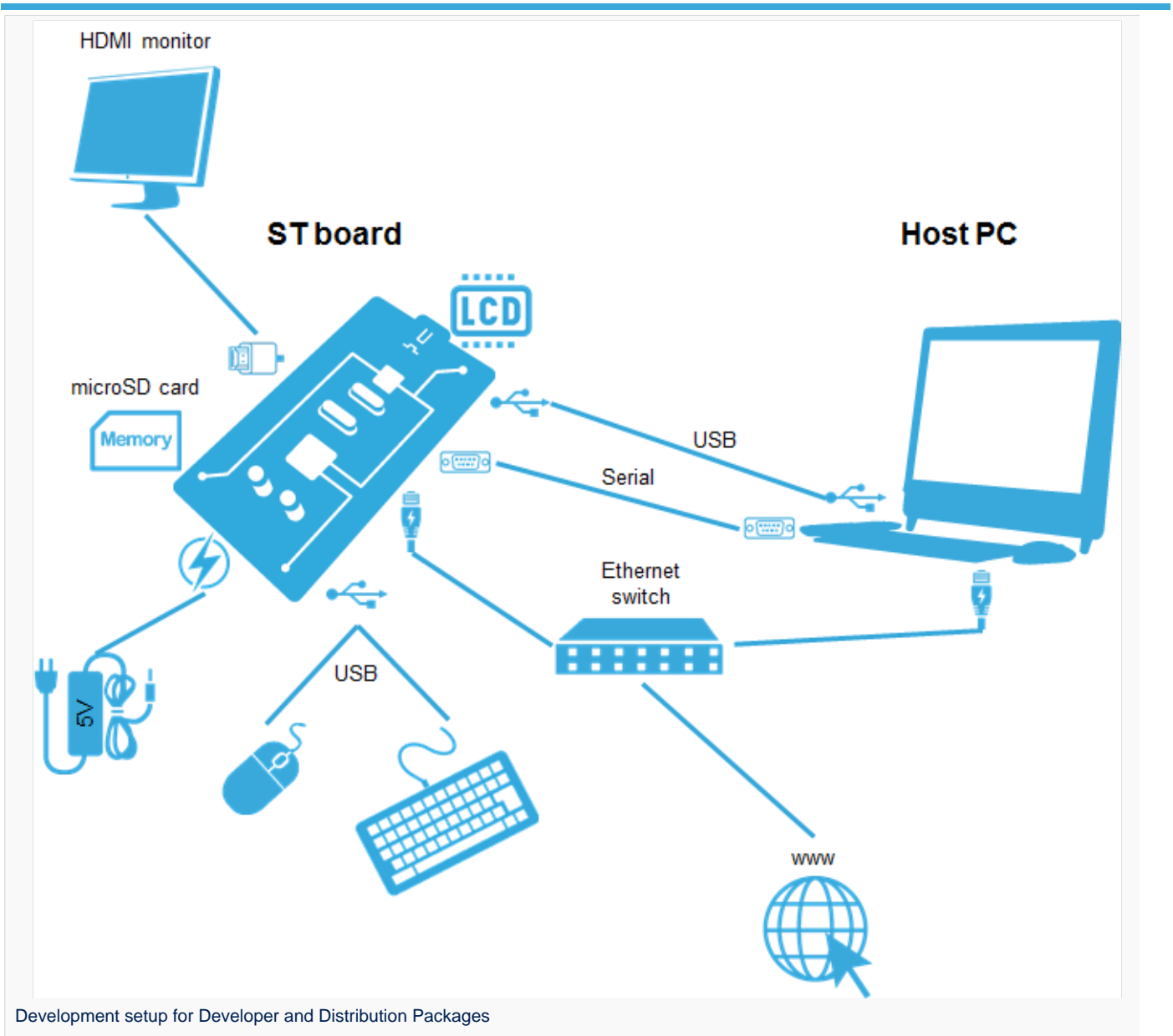
The STM32MP1 Developer Package aims at enriching a Linux-based software for the targeted product: basic knowledges about Linux are recommended to make the most of this Package.

Having a look at the [STM32MPU Embedded Software architecture overview](#) is also highly recommended.

3.2 Development setup

The recommended setup for the development PC (host) is specified in the following article: [PC prerequisites](#).

Whatever the development platform (board) and development PC (host) used, the range of possible development setups is illustrated by the picture below.



The following components are **mandatory**:

- Host PC for cross-compilation and cross-debugging, installed as specified above
- Board assembled and configured as specified in the associated Starter Package article
- Mass storage device (for example, microSD card) to load and update the software images (binaries)

The following components are **optional**, but **recommended**:

- A serial link between the host PC (through [Terminal program](#)) and the board for traces (even early boot traces), and access to the board from the remote PC (command lines)
- An Ethernet link between the host PC and the board for cross-development and cross-debugging through a local network. This is an alternative or a complement to the serial (or USB) link
- A display connected to the board, depending on the technologies available on the board: DSI LCD display, HDMI monitor (or TV) and so on
- A mouse and a keyboard connected through USB ports



Additional optional components can be added by means of the connectivity capabilities of the board: cameras, displays, JTAG, sensors, actuators, and much more.



4 Installing the Starter Package

Before developing with the Developer Package, **it is essential to start up your board thanks to its Starter Package**. All articles relative to Starter Packages are found in [Category:Starter Package](#): find the one that corresponds to your board, and follow the installation instructions (if not yet done), before going further.

In brief, it means that:

- your board boots successfully
- the flashed image comes from the same release of the STM32MPU Embedded Software distribution than the components that will be downloaded in this article

Thanks to the Starter Package, **all Flash partitions are populated**.

Then, with the Developer Package, it is possible to modify or to upgrade the partitions independently one from the others.

For example, if you only want to modify the Linux kernel (part of *bootfs* partition), installing the SDK and the Linux kernel are enough; no need to install anything else.



5 Installing the components to develop software running on Arm Cortex-A (OpenSTLinux distribution)

5.1 Installing the SDK

Optional step: it is mandatory only if you want to modify or add software running on Arm Cortex-A (e.g. Linux kernel, Linux user space applications...).

The SDK for OpenSTLinux distribution provides a stand-alone cross-development toolchain and libraries tailored to the contents of the specific image flashed in the board. If you want to know more about this SDK, please read the [SDK for OpenSTLinux distribution](#) article.

- The STM32MP1 SDK is delivered through a tarball file named : `en.SDK-x86_64-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz`
- Download and install the STM32MP1 SDK.


The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the [software license agreement \(SLA\)](#). The detailed content licenses can be found [here](#).



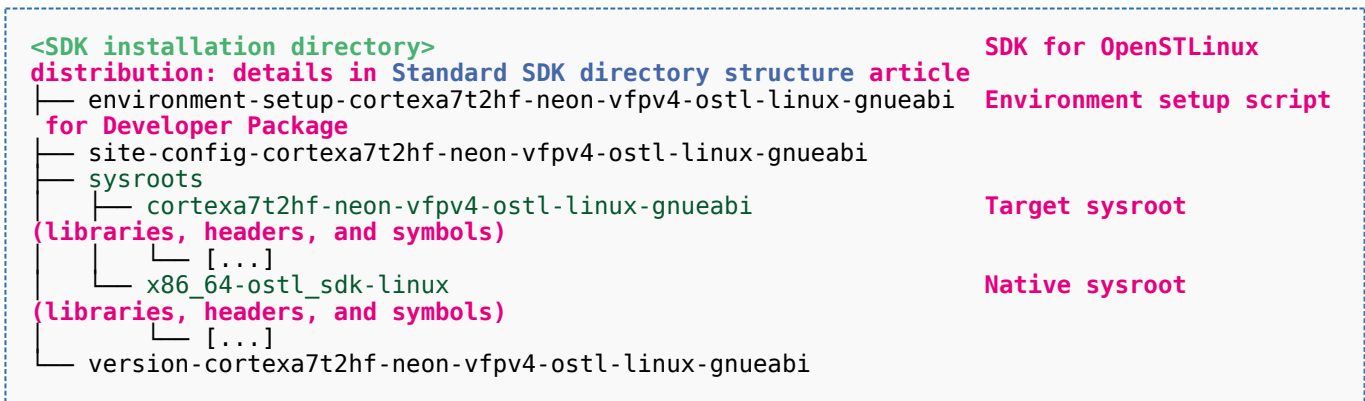
To download a package, it is recommended to be logged in to your "myst" account [1]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem. We apologize for this inconvenience

STM32MP1 Developer Package SDK - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on <i>my.st.com</i> before accessing the following link: <code>en.SDK-x86_64-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz</code>
	<ul style="list-style-type: none"> • Uncompress the tarball file to get the SDK installation script <pre>PC \$> tar xvf en.SDK-x86_64-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz</pre> <ul style="list-style-type: none"> • If needed, change the permissions on the SDK installation script so that it is executable <pre>PC \$> chmod +x stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sdk/st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-openstlinux-5.10-dunfell-mp1-21-03-31.sh</pre> <ul style="list-style-type: none"> • Run the SDK installation script <ul style="list-style-type: none"> • Use the <code>-d <SDK installation directory absolute path></code> option to specify the absolute path to the directory in which you want to install the SDK (<code><SDK installation directory></code>) • If you follow the proposition to organize the working directory, it means:



STM32MP1 Developer Package SDK - STM32MP15-Ecosystem-v3.0.0 release	
Installation	<p>PC \$> <code>./stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sdk/st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-openstlinux-5.10-dunfell-mp1-21-03-31.sh -d <working directory absolute path>/Developer-Package/SDK</code></p> <ul style="list-style-type: none"> A successful installation outputs the following log: <pre> ST OpenSTLinux - Weston - (A Yocto Project Based Distro) SDK installer version 3.1-openstlinux-5.10-dunfell-mp1-21-03-31 ===== ===== You are about to install the SDK to "<working directory absolute path>/Developer-Package/SDK". Proceed [Y/n]? Extracting SDK.....done Setting it up...done SDK has been successfully set up and is ready to be used. Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g. \$> . <working directory absolute path>/Developer-Package/SDK/env ironment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi </pre>
Release note	<p>Details about the content of the SDK are available in the associated STM32MP15 ecosystem release note.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

- The SDK is in the *<SDK installation directory>*:



 **Now that the SDK is installed, please do not move or rename the *<SDK installation directory>*.**



5.1.1 Starting up the SDK

The SDK environment setup script must be run once in each new working terminal in which you cross-compile:

```
PC $> source <SDK installation directory>/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-
linux-gnueabi
```

The following checkings allow to ensure that the environment is correctly setup:

- Check the target architecture

```
PC $> echo $ARCH
arm
```

- Check the toolchain binary prefix for the target tools

```
PC $> echo $CROSS_COMPILE
arm-ostl-linux-gnueabi-
```

- Check the C compiler version

```
PC $> $CC --version
arm-ostl-linux-gnueabi-gcc (GCC) <GCC version>
[...]
```

- Check that the SDK version is the expected one

```
PC $> echo $OECORE_SDK_VERSION
<expected SDK version>
```



If any of these commands fails or does not return the expected result, please try to reinstall the SDK.

5.2 Installing the Linux kernel

Optional step: it is mandatory only if you want to modify the Linux kernel (configuration, device tree, driver...), or to add external out-of-tree Linux kernel modules.

Prerequisite: the SDK is installed.

5.2.1 Downloading the Linux kernel

- The STM32MP1 Linux kernel is delivered through a tarball file named **en.SOURCES-kernel-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz** for STM32MP157x-EV1  and STM32MP157x-DKx  boards.
- Download and install the STM32MP1 Linux kernel

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found [here](#).

To download a package, it is recommended to be logged in to your "myst" account [2].



If, trying to download, you encounter a “403 error”, you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem.

We apologize for this inconvenience

STM32MP1 Developer Package Linux kernel - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on to <i>my.st.com</i> before accessing the following link en.SOURCES-kernel-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz
Installation	<ul style="list-style-type: none"> Go to the host PC directory in which you want to install the Developer Package (<<i>Developer Package installation directory</i>>); if you follow the proposition to organize the working directory, this means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> Download the tarball file in this directory Uncompress the tarball file to get the Linux kernel (Linux kernel source code, ST patches, ST configuration fragments...): <pre>PC \$> tar xvf en.SOURCES-kernel-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz PC \$> cd stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.10.10-r0 PC \$> tar xvf linux-5.10.10.tar.xz</pre>
Release note	<p>Details of the content of the Linux kernel are available in the associated STM32MP15 OpenSTLinux release note.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

- The **Linux kernel installation directory** is in the <*Developer Package installation directory*>/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi directory, and is named *linux-stm32mp-<kernel version>*:

```

linux-stm32mp-5.10.10-r0
├── [*].patch                Linux kernel installation directory
├── [*].patch               ST patches to apply during the Linux kernel
├── [*].patch               preparation (see next chapter)
├── fragment-[*].config    ST configuration fragments to apply during the Linux
├── fragment-[*].config    kernel configuration (see next chapter)
├── linux-5.10.10          Linux kernel source code directory
├── linux-5.10.10.tar.xz   Tarball file of the Linux kernel source code
├── README.HOW_TO.txt     Helper file for Linux kernel management: reference
├── README.HOW_TO.txt     for Linux kernel build
└── series                 List of all ST patches to apply

```

5.2.2 Building and deploying the Linux kernel for the first time

It is mandatory to execute once the steps specified below before modifying the Linux kernel, or adding external out-of-tree Linux kernel modules.



The partitions related to the Linux kernel are:

- the *bootfs* partition that contains the Linux kernel U-Boot image (*ulmage*) and device tree
- the *rootfs* partition that contains the Linux kernel modules

The Linux kernel might be cross-compiled, either in the source code directory, or in a dedicated directory different from the source code directory.

This last method is recommended as it clearly separates the files generated by the cross-compilation from the source code files.



The [README_HOWTO.txt](#) helper file is **THE** reference for the Linux kernel build



The SDK must be started

Open the *<Linux kernel installation directory>/README.HOW_TO.txt* helper file, and execute its instructions to:

- setup a software configuration management (SCM) system (*git*) for the Linux kernel (optional but recommended)
- prepare the Linux kernel (applying the ST patches)
- configure the Linux kernel (applying the ST fragments)
- cross-compile the Linux kernel
- deploy the Linux kernel (i.e. update the software on board)

The Linux kernel is now installed: let's modify the Linux kernel, or add external out-of-tree Linux kernel modules.

5.3 Installing the U-Boot

Optional step: it is mandatory only if you want to modify the U-Boot.

Prerequisite: the SDK is installed.

5.3.1 Downloading the U-Boot

- The STM32MP1 U-Boot is delivered through a tarball file named **en.SOURCES-u-boot-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz** for STM32MP157x-EV1 and STM32MP157x-DKx boards.
- Download and install the STM32MP1 U-Boot

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.



To download a package, it is recommended to be logged in to your "myst" account [3]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem.

We apologize for this inconvenience



STM32MP1 Developer Package U-Boot - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on to <i>my.st.com</i> before accessing the following link en.SOURCES-u-boot-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz
Installation	<ul style="list-style-type: none"> Go to the host PC directory in which you want to install the Developer Package (<<i>Developer Package installation directory</i>>); if you follow the proposition to organize the working directory, this means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> <ul style="list-style-type: none"> Download the tarball file in this directory Uncompress the tarball file to get the U-Boot (U-Boot source code, ST patches and so on): <pre>PC \$> tar xvf en.SOURCES-u-boot-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz PC \$> cd stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi/u-boot-stm32mp-2020.10.r1-r0 PC \$> tar xvf u-boot-stm32mp-2020.10.r1-r0.tar.gz</pre>
Release note	<p>Details of the content of the U-Boot are available in the associated STM32MP15 OpenSTLinux release note.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

- In the <*Developer Package installation directory*>/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-312/sources/arm-ostl-linux-gnueabi directory

- The **U-Boot installation directory** is named *u-boot-stm32mp-<U-Boot version>*:

<pre>u-boot-stm32mp-2020.10.r1-r0 ├── [*].patch ├── preparation (see next chapter) ├── u-boot-stm32mp-2020.10.r1 ├── Makefile.sdk ├── README.HOW_TO.txt ├── nce for U-Boot build ├── series └── u-boot-stm32mp-2020.10.r1-r0.tar.gz</pre>	<p>U-Boot installation directory ST patches to apply during the U-Boot</p> <p>U-Boot source code directory Makefile for the U-Boot compilation Helper file for U-Boot management: refere</p> <p>List of all ST patches to apply Tarball file of the U-Boot source code</p>
--	--

- The **FIP artifacts directory** is named *FIP_artifacts*:

<pre>FIP_artifacts ├── arm-trusted-firmware │ ├── bl32 │ │ ├── stm32mp157a-dk1-bl32.dtb │ │ ├── stm32mp157a-ev1-bl32.dtb │ │ ├── stm32mp157c-dk2-bl32.dtb │ │ ├── stm32mp157c-ed1-bl32.dtb │ │ └── stm32mp157c-ev1-bl32.dtb ├── kits ├── boards ├── kits ├── boards</pre>	<p>Device tree for TF-A → STM32MP15 Discovery</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p> <p>Device tree for TF-A → STM32MP15 Discovery</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p>
---	---



boards			
kits		stm32mp157d-dk1-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
boards		stm32mp157d-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
kits		stm32mp157f-dk2-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
boards		stm32mp157f-ed1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards		stm32mp157f-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
		tf-a-bl32-stm32mp15.bin	Binary file for bl32 stage
		fwconfig	
		stm32mp157a-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits		stm32mp157a-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits		stm32mp157a-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157a-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		stm32mp157c-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits		stm32mp157c-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits		stm32mp157c-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157c-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		stm32mp157c-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157c-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		stm32mp157d-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits		stm32mp157d-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits		stm32mp157d-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157d-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		stm32mp157f-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits		stm32mp157f-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits		stm32mp157f-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157f-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		stm32mp157f-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards		stm32mp157f-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards			
		optee	
		tee-header_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits		tee-header_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards		tee-header_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits		tee-header_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards		tee-header_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards		tee-header_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits			



tee-header_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-header_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-header_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-header_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
u-boot	
u-boot-nodtb-stm32mp15.bin	
u-boot-stm32mp157a-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157a-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157c-dk2-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157c-ed1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157c-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157d-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157d-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157f-dk2-trusted.dtb	Device tree for U-Boot → STM32MP15



Discovery kits

├── u-boot-stm32mp157f-ed1-trusted.dtb

Device tree for U-Boot → STM32MP15

Evaluation boards

├── u-boot-stm32mp157f-ev1-trusted.dtb

Device tree for U-Boot → STM32MP15

Evaluation boards

5.3.2 Building and deploying the U-Boot for the first time

It is mandatory to execute once the steps specified below before modifying the U-Boot.

As explained in the [boot chain overview](#), the trusted boot chain is the default solution delivered by STMicroelectronics.

Within this scope, the partition related to the U-Boot is the *ssbl* one that contains the U-Boot and its device tree blob.



The [README_HOWTO.txt](#) helper file is **THE** reference for the U-Boot build



The SDK must be started

Open the *<U-Boot installation directory>/README.HOW_TO.txt* helper file, and execute its instructions to:

setup a software configuration management (SCM) system (*git*) for the U-Boot (optional but recommended)

prepare the U-Boot (applying the ST patches)

cross-compile the U-Boot

deploy the U-Boot (i.e. update the software on board)

The U-Boot is now installed: let's modify the U-Boot.

5.4 Installing the TF-A

Optional step: it is mandatory only if you want to modify the TF-A.

Prerequisite: the SDK is installed.

5.4.1 Downloading the TF-A

- The STM32MP1 TF-A is delivered through a tarball file named **en.SOURCES-tf-a-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz** for STM32MP157x-EV1 and STM32MP157x-DKx boards.


- Download and install the STM32MP1 TF-A

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found [here](#).



To download a package, it is recommended to be logged in to your "myst" account [4]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem. We apologize for this inconvenience



STM32MP1 Developer Package TF-A - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on my.st.com before accessing the following link: en.SOURCES-tf-a-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz
Installation	<ul style="list-style-type: none"> Go to the host PC directory in which you want to install the Developer Package (<Developer Package installation directory>); if you follow the proposition to organize the working directory, it means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> <ul style="list-style-type: none"> Download the tarball file in this directory Uncompress the tarball file to get the TF-A (TF-A source code, ST patches...): <pre>PC \$> tar xvf en.SOURCES-tf-a-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz PC \$> cd stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi/ PC \$> tar xvf tf-a-stm32mp-2.4.r1-r0.tar.gz</pre>
Release note	<p>Details about the content of the TF-A are available in the associated STM32MP15 OpenSTLinux release note.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

• In the <Developer Package installation directory>/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi directory,

- The **TF-A installation directory** is named *tf-a-stm32mp-<TF-A version>*:

<pre>tf-a-stm32mp-2.4.r1-r0 ├── [*].patch ├── tf-a-stm32mp-2.4.r1 ├── Makefile.sdk ├── README.HOW_T0.txt ├── -A build │ ├── series │ └── tf-a-stm32mp-2.4.r1-r0.tar.gz</pre>	<p>TF-A installation directory ST patches to apply during the TF-A preparation</p> <p>TF-A source code directory Makefile for the TF-A compilation Helper file for TF-A management: reference for TF</p> <p>List of all ST patches to apply Tarball file of the TF-A source code</p>
--	---

- The **FIP artifacts directory** is named *FIP_artifacts*:

<pre>FIP_artifacts ├── arm-trusted-firmware │ ├── bl32 │ │ ├── stm32mp157a-dk1-bl32.dtb │ │ ├── stm32mp157a-ev1-bl32.dtb │ │ ├── stm32mp157c-dk2-bl32.dtb │ │ ├── stm32mp157c-ed1-bl32.dtb │ │ └── stm32mp157c-ev1-bl32.dtb ├── kits ├── boards ├── kits ├── boards</pre>	<p>Device tree for TF-A → STM32MP15 Discovery</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p> <p>Device tree for TF-A → STM32MP15 Discovery</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p>
---	---



boards	stm32mp157d-dk1-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
kits	stm32mp157d-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards	stm32mp157f-dk2-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
kits	stm32mp157f-ed1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards	stm32mp157f-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards	tf-a-bl32-stm32mp15.bin	Binary file for bl32 stage
	fwconfig	
	stm32mp157a-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157a-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157a-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157a-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157c-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157c-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157c-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157c-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157c-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157c-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157d-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157d-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157d-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157d-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157f-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157f-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157f-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157f-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157f-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157f-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards		
	optee	
	tee-header_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits		



tee-header_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-header_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-header_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-header_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pageable_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pageable_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	
tee-pager_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
tee-pager_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	
u-boot	
u-boot-nodtb-stm32mp15.bin	
u-boot-stm32mp157a-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157a-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157c-dk2-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157c-ed1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157c-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	
u-boot-stm32mp157d-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	
u-boot-stm32mp157d-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	



Discovery kits	u-boot-stm32mp157f-dk2-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	u-boot-stm32mp157f-ed1-trusted.dtb	Device tree for U-Boot → STM32MP15
	u-boot-stm32mp157f-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15

5.4.2 Building and deploying the TF-A for the first time

It is mandatory to execute once the steps specified below before modifying the TF-A.

As explained in the [boot chain overview](#), the trusted boot chain is the default solution delivered by STMicroelectronics.

Within this scope, the partition related to the TF-A is the *fsbl* one.



The [README_HOWTO.txt](#) helper file is **THE** reference for the TF-A build



The SDK must be started

Open the *<TF-A installation directory>/README.HOW_TO.txt* helper file, and execute its instructions to:

- setup a software configuration management (SCM) system (*git*) for the TF-A (optional but recommended)
- prepare the TF-A (applying the ST patches)
- cross-compile the TF-A
- deploy the TF-A (i.e. update the software on board)

The TF-A is now installed: let's modify the TF-A.

5.5 Installing the OP-TEE

Optional step: it is mandatory only if you want to modify the OP-TEE.

Prerequisite: the SDK is installed.

5.5.1 Downloading the OP-TEE

- The STM32MP1 OP-TEE is delivered through a tarball file named **en.SOURCES-optee-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz** for STM32MP157x-EV1 and STM32MP157x-DKx boards.
- Download and install the STM32MP1 OP-TEE

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the *software license agreement (SLA)*. The detailed content licenses can be found [here](#).



To download a package, it is recommended to be logged in to your "myst" account [5]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem.



We apologize for this inconvenience

STM32MP1 Developer Package OP-TEE - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on <i>my.st.com</i> before accessing the following link: en.SOURCES-optee-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz
Installation	<ul style="list-style-type: none"> Go to the host PC directory in which you want to install the Developer Package (<<i>Developer Package installation directory</i>>); if you follow the proposition to organize the working directory, it means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> <ul style="list-style-type: none"> Download the tarball file in this directory Uncompress the tarball file to get the OP-TEE (OP-TEE source code, ST patches...): <pre>PC \$> tar xvf en.SOURCES-optee-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz PC \$> cd stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi/ PC \$> tar xvf optee-os-stm32mp-3.12.0.r1-r0.tar.gz</pre>
Release note	<p>Details about the content of the OP-TEE are available in the associated STM32MP15 OpenSTLinux release note.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

- In the <*Developer Package installation directory*>/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi directory,

- The **OP-TEE installation directory** is named *optee-os-stm32mp-<OP-TEE version>*:

<pre>optee-os-stm32mp-3.12.0.r1-r0 ├── [*].patch ├── preparation (see next chapter) ├── optee-os-stm32mp-3.12.0.r1 ├── Makefile.sdk ├── optee-os-stm32mp-3.12.0.r1-r0.tar.gz ├── README.HOW_TO.txt └── nce for OP-TEE build series</pre>	<p>OP-TEE installation directory ST patches to apply during the OP-TEE</p> <p>OP-TEE source code directory Makefile for the OP-TEE compilation Tarball file of the OP-TEE source code Helper file for OP-TEE management: refere</p> <p>List of all ST patches to apply</p>
--	---

- The **FIP artifacts directory** is named *FIP_artifacts*:

<pre>FIP_artifacts ├── arm-trusted-firmware │ ├── bl32 │ │ ├── stm32mp157a-dk1-bl32.dtb │ │ ├── stm32mp157a-ev1-bl32.dtb │ │ └── stm32mp157c-dk2-bl32.dtb ├── kits ├── boards └── kits</pre>	<p>Device tree for TF-A → STM32MP15 Discovery</p> <p>Device tree for TF-A → STM32MP15 Evaluation</p> <p>Device tree for TF-A → STM32MP15 Discovery</p>
--	---



boards	stm32mp157c-ed1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards	stm32mp157c-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
kits	stm32mp157d-dk1-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
boards	stm32mp157d-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
kits	stm32mp157f-dk2-bl32.dtb	Device tree for TF-A → STM32MP15 Discovery
boards	stm32mp157f-ed1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
boards	stm32mp157f-ev1-bl32.dtb	Device tree for TF-A → STM32MP15 Evaluation
	tf-a-bl32-stm32mp15.bin	Binary file for bl32 stage
	fwconfig	
	stm32mp157a-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157a-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157a-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157a-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157c-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157c-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157c-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157c-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157c-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157c-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157d-dk1-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157d-dk1-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157d-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157d-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157f-dk2-fw-config-optee.dtb	Device tree for FW config
→ STM32MP15 Discovery kits	stm32mp157f-dk2-fw-config-trusted.dtb	Device tree for FW config →
STM32MP15 Discovery kits	stm32mp157f-ed1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157f-ed1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	stm32mp157f-ev1-fw-config-optee.dtb	Device tree for FW config
→ Evaluation boards	stm32mp157f-ev1-fw-config-trusted.dtb	Device tree for FW config →
Evaluation boards	tee-header_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15



Evaluation boards	tee-header_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-header_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-header_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pageable_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pageable_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pageable_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pageable_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pageable_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157a-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pager_v2-stm32mp157a-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157c-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pager_v2-stm32mp157c-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157c-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157d-dk1.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pager_v2-stm32mp157d-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157f-dk2.bin	Binary file for OP-TEE OS → STM32MP15
Discovery kits	tee-pager_v2-stm32mp157f-ed1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	tee-pager_v2-stm32mp157f-ev1.bin	Binary file for OP-TEE OS → STM32MP15
Evaluation boards	u-boot	
	u-boot-nodtb-stm32mp157.bin	
	u-boot-stm32mp157a-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	u-boot-stm32mp157a-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	u-boot-stm32mp157c-dk2-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	u-boot-stm32mp157c-ed1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	u-boot-stm32mp157c-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15
Evaluation boards	u-boot-stm32mp157d-dk1-trusted.dtb	Device tree for U-Boot → STM32MP15
Discovery kits	u-boot-stm32mp157d-ev1-trusted.dtb	Device tree for U-Boot → STM32MP15



Evaluation boards

├── u-boot-stm32mp157f-dk2-trusted.dtb

Device tree for U-Boot → STM32MP15

Discovery kits

├── u-boot-stm32mp157f-ed1-trusted.dtb

Device tree for U-Boot → STM32MP15

Evaluation boards

├── u-boot-stm32mp157f-ev1-trusted.dtb

Device tree for U-Boot → STM32MP15

Evaluation boards

5.5.2 Building and deploying the OP-TEE for the first time

It is mandatory to execute once the steps specified below before modifying the OP-TEE.

As explained in the [boot chain overview](#), the trusted boot chain is the default solution delivered by STMicroelectronics. Within this scope, the partition related to the OP-TEE is the *fsbl* one.



The `README_HOWTO.txt` helper file is **THE** reference for the OP-TEE build



The SDK must be started

Open the `<OP-TEE installation directory>/README.HOW_TO.txt` helper file, and execute its instructions to:

setup a software configuration management (SCM) system (*git*) for the OP-TEE (optional but recommended)

prepare the OP-TEE (applying the ST patches)

cross-compile the OP-TEE

deploy the OP-TEE (i.e. update the software on board)

The OP-TEE is now installed: let's modify the OP-TEE.

5.6 Installing the debug symbol files

Optional step: it is mandatory only if you want to debug Linux[®] kernel, U-Boot or TF-A with GDB.

5.6.1 Downloading the debug symbol files


- The STM32MP1 debug symbol files is delivered through a tarball file named `en.DEBUG-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz` for STM32MP157x-EV1 and STM32MP157x-DKx boards.
- Download and install the STM32MP1 debug symbol files

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found [here](#).



To download a package, it is recommended to be logged in to your "myst" account [6]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem. We apologize for this inconvenience



STM32MP1 Developer Package debug symbol files - STM32MP15-Ecosystem-v3.0.0 release	
Download	You need to be logged on to <i>my.st.com</i> before accessing the following link en.DEBUG-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz
Installation	<ul style="list-style-type: none"> Go to the host PC directory in which you want to install the Developer Package (<Developer Package installation directory>); if you follow the proposition to organize the working directory, this means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> <ul style="list-style-type: none"> Download the tarball file in this directory Uncompress the tarball file to get the debug symbol files (for Linux kernel, U-Boot, TF-A and OP-TEE OS): <pre>PC \$> tar xvf en.DEBUG-stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31.tar.xz</pre>
Release note	 If you are interested in older releases, please have a look into the section Archives .

- The debug symbol files are in the <Developer Package installation directory>/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/images/stm32mp1 directory:

```

stm32mp1
├── arm-trusted-firmware
│   └── debug
│       ├── tf-a-bl2-emmc.elf
│       │   Debug symbol file for TF-A → TF-A for emmc boot stage
│       ├── tf-a-bl2-nand.elf
│       │   Debug symbol file for TF-A → TF-A for nand boot stage
│       ├── tf-a-bl2-nor.elf
│       │   Debug symbol file for TF-A → TF-A for nor boot stage
│       ├── tf-a-bl2-sdcard.elf
│       │   Debug symbol file for TF-A → TF-A for SDcard boot stage
│       ├── tf-a-bl2-uart.elf
│       │   Debug symbol file for TF-A → TF-A for UART
│       │   downloading boot stage
│       └── tf-a-bl2-usb.elf
│           Debug symbol file for TF-A → TF-A for USB downloading
│           boot stage
├── kernel
│   └── vmlinux
│       Debug symbol file for Linux kernel
├── optee
│   └── debug
│       └── tee-stm32mp157a-dk1.elf
│           Debug symbol file for OP-TEE OS → STM32MP15 Discovery
│           kits

```



```

|           | tee-stm32mp157a-ev1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
|           | tee-stm32mp157c-dk2.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Discovery
kits
|           | tee-stm32mp157c-ed1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
|           | tee-stm32mp157c-ev1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
|           | tee-stm32mp157d-dk1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Discovery
kits
|           | tee-stm32mp157d-ev1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
|           | tee-stm32mp157f-dk2.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Discovery
kits
|           | tee-stm32mp157f-ed1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
|           | tee-stm32mp157f-ev1.elf           De
bug symbol file for OP-TEE OS → STM32MP15 Evaluation
boards
├─ u-boot
│   └─ debug
elf       | u-boot-stm32mp157a-dk1-trusted.
Debug symbol file for U-Boot → STM32MP15
Discovery kits
elf       | u-boot-stm32mp157a-ev1-trusted.
Debug symbol file for U-Boot → STM32MP15
Evaluation boards
elf       | u-boot-stm32mp157c-dk2-trusted.
Debug symbol file for U-Boot → STM32MP15
Discovery kits
elf       | u-boot-stm32mp157c-ed1-trusted.
Debug symbol file for U-Boot → STM32MP15
Evaluation boards
elf       | u-boot-stm32mp157c-ev1-trusted.
Debug symbol file for U-Boot → STM32MP15
Evaluation boards
elf       | u-boot-stm32mp157d-dk1-trusted.
Debug symbol file for U-Boot → STM32MP15
Discovery kits
elf       | u-boot-stm32mp157d-ev1-trusted.
Debug symbol file for U-Boot → STM32MP15
Evaluation boards
elf       | u-boot-stm32mp157f-dk2-trusted.
Debug symbol file for U-Boot → STM32MP15
Discovery kits
elf       | u-boot-stm32mp157f-ed1-trusted.

```



```
elf          Debug symbol file for U-Boot → STM32MP15  
Evaluation boards  
└─ u-boot-stm32mp157f-ev1-trusted.  
elf          Debug symbol file for U-Boot → STM32MP15  
Evaluation boards
```

5.6.2 Using the debug symbol files

These files are used to debug the Linux[®] kernel, U-Boot or TF-A with GDB. Especially, the [Debug OpenSTLinux BSP components](#) chapter explains how to load the debug symbol files in GDB.



6 Installing the components to develop software running on Arm Cortex-M4 (STM32Cube MPU Package)

6.1 Installing STM32CubeIDE

Optional step: it is needed if you want to modify or add software running on Arm Cortex-M.

The table below explains how to download and install STM32CubeIDE which addresses STM32 MCU, and also provides support for Cortex-M inside STM32 MPU.

STM32 MPU support inside STM32CubeIDE is available on Linux[®] and Windows[®] host PCs, but it is **NOT** on macOS[®].

	STM32CubeIDE for Linux [®] host PC	STM32CubeIDE for Windows [®] host PC
Download	Version 1.6.1 <ul style="list-style-type: none"> Download the preferred all-in-one Linux installer from my.st.com <ul style="list-style-type: none"> <i>Generic Linux Installer - STM32CubeIDE-Lnx</i> <i>RPM Linux Installer - STM32CubeIDE-RPM</i> <i>Debian Linux Installer - STM32CubeIDE-DEB</i> 	Version 1.6.1 <ul style="list-style-type: none"> Download the all-in-one Windows installer from my.st.com <ul style="list-style-type: none"> <i>Windows Installer - STM32CubeIDE-Win</i>
Installation guide	<ul style="list-style-type: none"> Refer to <i>STM32CubeIDE installation guide (UM2563)</i> available on my.st.com. 	
User manual	<ul style="list-style-type: none"> When the installation is completed, see additional information about STM32CubeIDE in my.st.com: <ul style="list-style-type: none"> <i>STM32CubeIDE quick start guide (UM2553)</i> <i>Getting started with projects based on the STM32MP1 Series in STM32CubeIDE (AN5360)</i> 	
Detailed release note	<ul style="list-style-type: none"> Details about the content of this tool version are available in the <i>STM32CubeIDE release v1.6.1</i> release note from my.st.com 	

Minor releases may be available from the update site. Check chapter 10 in (UM2609) for more information on how to update STM32CubeIDE.

6.2 Installing the STM32Cube MPU Package

Optional step: it is mandatory only if you want to modify the STM32Cube MPU Package.

Prerequisite: the STM32CubeIDE is installed.

- The STM32CubeMP1 Package is delivered through an archive file named **en.STM32Cube_FW_MP1_V1.4.0.zip**.



- Download and install the STM32CubeMP1 Package

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found [here](#).



To download a package, it is recommended to be logged in to your "myst" account [7]. If, trying to download, you encounter a "403 error", you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem.

We apologize for this inconvenience

STM32MP1 Developer Package STM32CubeMP1 Package - v3.0.0 release	
Download	You need to be logged on my.st.com before accessing the following link: en.STM32Cube_FW_MP1_V1.4.0.zip
Installation	<ul style="list-style-type: none"> • Go to the host PC directory in which you want to install the Developer Package (<Developer Package installation directory>); if you follow the proposition to organize the working directory, it means: <pre>PC \$> cd <working directory path>/Developer-Package</pre> <ul style="list-style-type: none"> • Download the archive file in this directory • Uncompress the archive file to get the STM32CubeMP1 Package: <pre>PC \$> unzip en.STM32Cube_FW_MP1_V1.4.0.zip</pre>
Release note	<p>Details about the content of the STM32CubeMP1 Package are available in the STM32Cube_FW_MP1_V1.4.0/Release_Notes.html file.</p> <p> If you are interested in older releases, please have a look into the section Archives.</p>

- The STM32CubeMP1 Package installation directory is in the <Developer Package installation directory> directory, and is named `STM32Cube_FW_MP1_V1.4.0`:

```

STM32Cube_FW_MP1_V1.4.0
├── P1 Package content article
│   ├── Drivers
│   │   ├── BSP
│   │   │   └── [...]
│   │   ├── CMSIS
│   │   │   └── [...]
│   │   └── STM32MP1xx_HAL_Driver
│   │       └── [...]
│   └── devices
│       ├── htmresc
│       │   └── [...]
│       ├── License.md
│       ├── Middlewares
│       │   └── [...]
│       └── package.xml

```

STM32CubeMP1 Package: details in STM32CubeM

BSP drivers for the supported STM32MP1 boards

HAL drivers for the supported STM32MP1



Projects	
STM32CubeProjectsList.html	List of examples and applications for
STM32CubeMP1 Package	
STM32MP157C-DK2	Set of examples and applications →
STM32MP15 Discovery kits	
[...]	
STM32MP157C-EV1	Set of examples and applications →
STM32MP15 Evaluation boards	
[...]	
Readme.md	
Release_Notes.html	Release note for STM32CubeMP1 Package
Utilities	
[...]	

The STM32Cube MPU Package is now installed: let's develop software running on Arm Cortex-M4.



7 Developing software running on Arm Cortex-A7

7.1 Modifying the Linux kernel

Prerequisites:

- the SDK is installed
- the SDK is started up
- the Linux kernel is installed

The *<Linux kernel installation directory>/README.HOW_TO.txt* helper file gives the commands to:

configure the Linux kernel

cross-compile the Linux kernel

deploy the Linux kernel (that is, update the software on board)

You can refer to the following simple examples:

- Modification of the kernel configuration
- Modification of the device tree
- Modification of a built-in device driver
- Modification of an external in-tree module

7.2 Adding external out-of-tree Linux kernel modules

Prerequisites:

- the SDK is installed
- the SDK is started up
- the Linux kernel is installed

Most device drivers (or modules) in the Linux kernel can be compiled either into the kernel itself (built-in, or internal module) or as Loadable Kernel Modules (LKMs, or external modules) that need to be placed in the root file system under the `/lib/modules` directory. An external module can be in-tree (in the kernel tree structure), or out-of-tree (outside the kernel tree structure).

External Linux kernel modules are compiled taking reference to a Linux kernel source tree and a Linux kernel configuration file (`.config`).

Thus, a makefile for an external Linux kernel module points to the Linux kernel directory that contains the source code and the configuration file, with the `"-C <Linux kernel path>"` option.

This makefile also points to the directory that contains the source file(s) of the Linux kernel module to compile, with the `"M=<Linux kernel module path>"` option.

A generic makefile for an external out-of-tree Linux kernel module looks like the following:

```
# Makefile for external out-of-tree Linux kernel module

# Object file(s) to be built
obj-m := <module source file(s)>.o

# Path to the directory that contains the Linux kernel source code
# and the configuration file (.config)
KERNEL_DIR ?= <Linux kernel path>

# Path to the directory that contains the generated objects
```



```

DESTDIR ?= <Linux kernel installation directory>

# Path to the directory that contains the source file(s) to compile
PWD := $(shell pwd)

default:
    $(MAKE) -C $(KERNEL_DIR) M=$(PWD) modules

install:
    $(MAKE) -C $(KERNEL_DIR) M=$(PWD) INSTALL_MOD_PATH=$(DESTDIR) modules_install

clean:
    $(MAKE) -C $(KERNEL_DIR) M=$(PWD) clean

```

Such module is then cross-compiled with the following commands:

```

$ make clean
$ make
$ make install

```

You can refer to the following simple example:

- Addition of an external out-of-tree module

7.3 Adding Linux user space applications

Prerequisites:

- the SDK is installed
- the SDK is started up

Once a suitable cross-toolchain (OpenSTLinux SDK) is installed, it is easy to develop a project outside of the OpenEmbedded build system.

There are different ways to use the SDK toolchain directly, among which Makefile and Autotools.

Whatever the method, it relies on:

- the sysroot that is associated with the cross-toolchain, and that contains the header files and libraries needed for generating binaries (see [target sysroot](#))
- the environment variables created by the SDK environment setup script (see [SDK startup](#))

You can refer to the following simple example:

- Addition of a "hello world" user space application

7.4 Modifying the U-Boot

Prerequisites:

- the SDK is installed
- the SDK is started up
- the U-Boot is installed

The [<U-Boot installation directory>/README.HOW_TO.txt](#) helper file gives the commands to:

cross-compile the U-Boot

deploy the U-Boot (that is, update the software on board)



You can refer to the following simple example:

- Modification of the U-Boot

7.5 Modifying the TF-A

Prerequisites:

- the SDK is installed
- the SDK is started up
- the TF-A is installed

The *<TF-A installation directory>/README.HOW_TO.txt* helper file gives the commands to:

cross-compile the TF-A

deploy the TF-A (that is, update the software on board)

You can refer to the following simple example:

- Modification of the TF-A

7.6 Modifying the OP-TEE

Prerequisites:

- the SDK is installed
- the SDK is started up
- the OP-TEE is installed

The *<OP-TEE installation directory>/README.HOW_TO.txt* helper file gives the commands to:

cross-compile the OP-TEE

deploy the OP-TEE (that is, update the software on board)



8 Developing software running on Arm Cortex-M4

8.1 How to create a Cube project from scratch or open/modify an existing one from STM32Cube MPU package

Please refer to [STM32CubeMP1 Package](#) article.



9 Fast links to essential commands

If you are already familiar with the Developer Package for the STM32MPU Embedded Software distribution, fast links to the essential commands are listed below.



With the links below, you will be redirected to other articles; use the *back* button of your browser to come back to these fast links

Link to the command
Starter Packages
Essential commands of the STM32MP15 Evaluation board Starter Package
Essential commands of the STM32MP15 Discovery kit Starter Package
SDK
Download and install the latest SDK
Start the SDK
Linux kernel
Download and install the latest Linux kernel
Helper file for the Linux kernel build, and update on board
U-Boot
Download and install the latest U-Boot
Helper file for the U-Boot build, and update on board
TF-A
Download and install the latest TF-A
Helper file for the TF-A build, and update on board
OP-TEE
Download and install the latest OP-TEE
Helper file for the OP-TEE build, and update on board
Linux user space
Simple user space application
STM32Cube MPU Package
Download and install the latest STM32CubeMP1 Package
Create or modify a Cube project



10 How to go further?

Now that your developments are ready, you might want to switch to the STM32MP1 Distribution Package, in order to create your own distribution and to generate your own SDK and image.

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex®

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

Linux® is a registered trademark of Linus Torvalds.

Das U-Boot -- the Universal Boot Loader (see [U-Boot_overview](#))

Trusted Firmware for Arm® Cortex®-A

Open Portable Trusted Execution Environment

Microprocessor Unit

Board support package

Hardware Abstraction Layer

(Software)Integrated development/design/debugging environment

GNU dedugger, a portable debugger that runs on many Unix-like systems

Display Serial Interface (MIPI® Alliance standard)

High-Definition Multimedia Interface (HDMI standard)

debug and test protocol, named from the Joint Test Action Group that developed it

The Linux Foundation® and Yocto Project® are registered trademarks of the Linux Foundation. Linux® is a registered trademark of Linus Torvalds

Firmware Image Package is a packaging format used by TF-A

Operating System

Universal Asynchronous Receiver/Transmitter

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

Cortex Microcontroller Software Interface Standard