



How to compile the device tree with the Developer Package

How to compile the device tree with the Developer Package



Contents



A quality version of this page, approved on 1 December 2020, was based off this revision.

Contents

1 Purpose	4
2 Rationale	5
3 Prerequisites	6
4 Preparing your environment	7
5 Updating the kernel device tree	8
5.1 Kernel : unpack and patch sources	8
5.2 Kernel : copy the DTS into the source code	8
5.3 Kernel : regenerate the kernel DTB	8
5.4 Kernel : copy the DTB into bootfs	9
6 Updating the u-boot device tree	10
6.1 U-boot : unpack and patch sources	10
6.2 U-boot : copy the DTS in the u-boot source code	10
6.3 U-boot : regenerate u-boot.stm32	10
6.4 U-boot : copy the u-boot into the target	11
7 Updating the TF-A device tree	12
7.1 TF-A : unpack and patch sources	12
7.2 TF-A : copy the DTS into the source code	12
7.3 TF-A : regenerate TF-A.stm32	12
7.4 TF-A : copy the DTB in target/bootfs	13
8 Update methods	13
8.1 Updating bootfs	14
8.2 Updating extlinux	15
8.2.1 extlinux basics	15
8.2.2 Updating extlinux	15



1 Purpose

This article explains how to update the [boot chain](#) (trusted mode) for a "custom" device tree. In particular, STM32CubeMX can generate a "custom" device tree.

This article describes how to update the device tree compiled (DTB) part of the boot binaries.



2 Rationale

There are various rationale for using a custom *device tree*, such as:

- the description of a new and private board
- the swapping of some internal peripherals from Cortex[®]-M side to Cortex-A side (or the opposite)



3 Prerequisites

i Information

Even if STMicroelectronics strongly recommends to use a Linux[®] environment, the steps described in this article can be executed in a [WSL2](#) (Windows Sub-system Linux 2) environment.

Compiling a new device tree means updating three software components belonging to the complete boot chain (trusted mode), Trusted Firmware A (TF-A), u-boot, and Linux kernel.

The material required to update the above software components is the following:

- **Starter package:**
 - the flashlayout as well as the images to flash, provided within the **en.FLASH-stm32mp1-openstlinux<YY>-<MM>-<DD>.tar.xz** file (download it from [st.com](#))
- **Developer package:**
 - the component sources and patches, provided within the **en.SOURCES-stm32mp1-openstlinux-<YY>-<MM>-<DD>.tar.xz** file (download it from [st.com](#))
 - the SDK toolchain, provided within the **en.SDK-x86_64-stm32mp1-openstlinux-<YY>-<MM>-<DD>.tar.xz** file (download it from [st.com](#))
- the **STM32CubeProgrammer**, which is the tool used to flash the images and binaries into the target.
- **Custom device tree sources:**
 - In the rest of this document, we assume that the custom device tree is generated by **STM32CubeMX** and stored in a *MyDeviceTree_fromCubeMX.tar.xz* tarball with following file tree:

```
MyDeviceTree_fromCubeMX
|-- kernel
|   |-- stm32mp157c-mydevicetree-mx.dts
|-- tf-a
|   |-- stm32mp15-mx.h
|   |-- stm32mp157c-mydevicetree-mx.dts
|-- u-boot
|   |-- stm32mp15-mx.h
|   |-- stm32mp157c-mydevicetree-mx-u-boot.dtsi
|   |-- stm32mp157c-mydevicetree-mx.dts
```

- Make sure the hardware configuration described in [PC_prerequisites#Linux PC has been executed](#) (even with a [WSL2](#) setup)



4 Preparing your environment

It is recommended to organize the numerous inputs described in #Prerequisites in your environment.

First create a dedicated *WORKDIR* under your *HOME* folder and copy there all the inputs listed in #Prerequisites:

```
PC $> cd $HOME
```

```
PC $> mkdir WORKDIR
```

```
PC $> cd WORKDIR
```

```
PC $> export WORKDIR="$PWD"
```

```
PC $> tar --strip-components=1 -xf <FLASH-st-image-weston-openstlinux-weston-stm32mp1.tar.xz> -C $WORKDIR/
```

```
PC $> tar --strip-components=1 -xf <SOURCES-st-image-weston-openstlinux-weston-stm32mp1.tar.xz> -C $WORKDIR/
```

```
PC $> tar --strip-components=1 -xf <SDK-st-image-weston-openstlinux-weston-stm32mp1.tar.xz> -C $WORKDIR/
```

```
PC $> tar xf <MyDeviceTree_fromCubeMX.tar.xz> -C $WORKDIR/
```

Then proceed with the [SDK installation](#).

The commands described in the rest of the document must be run in an SDK environment context: ([Starting_up_the_SDK](#)).



5 Updating the kernel device tree

Since 'extlinux.conf' explicitly points to the DTB, just update the kernel device tree by replacing the DTB file of the '/boot' partition. The path used must be something like '/boot/<devicetree>.dtb'.

The following chapters describe the procedure to generate and copy the new DTB into the target.

5.1 Kernel : unpack and patch sources

Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in *\$WORKDIR/sources/arm-ostl-linux-gnueabi/linux-stm32mp-**. Please notice a *grep "\$>" README.HOW_TO.txt* describes the few commands needed to build the artifact of kernel

Run the following command into a shell:

```
PC $> pushd $WORKDIR
PC $> mkdir -p kernel
PC $> tar xf sources/arm-ostl-linux-gnueabi/linux-stm32mp-*/linux-*.tar.xz -C kernel
PC $> mv kernel/linux-* kernel/kernel-sources/
PC $> pushd kernel/kernel-sources/
PC $> for p in $(ls -1 ../../sources/arm-ostl-linux-gnueabi/linux-stm32mp-*/*.patch); do patch -p1 < $p; done
PC $> popd
PC $> popd
```

5.2 Kernel : copy the DTS into the source code

```
PC $> pushd $WORKDIR
PC $> cp -r MyDeviceTree_fromCubeMX/kernel/* kernel/kernel-sources/arch/arm/boot/dts/
PC $> popd
```

5.3 Kernel : regenerate the kernel DTB

Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in *\$WORKDIR/sources/arm-ostl-linux-gnueabi/linux-stm32mp-**. Please notice a *grep "\$>" README.HOW_TO.txt* describes the few commands needed to build the artifact of kernel

```
PC $> pushd $WORKDIR/kernel/kernel-sources
PC $> make O="$PWD/../../build" multi_v7_defconfig
PC $> for f in `ls -1 ../../sources/arm-ostl-linux-gnueabi/linux-stm32mp-*/fragment*.config`; do scripts/kconfig/merge_config.
sh -m -r -O $PWD/../../build $PWD/../../build/.config $f; done
PC $> yes "" | make ARCH=arm oldconfig O="$PWD/../../build"
PC $> make stm32mp157c-mydevicetree-mx.dtb LOADADDR=0xC2000040 O="$PWD/../../build"
```




```
PC $> popd
```

```
PC $> ls -l $WORKDIR/kernel/build/arch/arm/boot/dts/stm32mp157c-mydevicetree-mx.dtb
```

5.4 Kernel : copy the DTB into bootfs

First of all #Updating bootfs with the new DTB so that it is taken into account at the next boot of the target.

Then, if needed, #Updating extlinux for the target according to this new DTB filename. This is only required if the filename of the generated DTB is different from the one used by extlinux to boot.



6 Updating the u-boot device tree

To update the u-boot device tree, replace the DTB part of the u-boot binary.

Adding a new device tree to the u-boot source code forces the Makefile to regenerate a new u-boot.stm32 containing the new DTS.

The following chapters describe the procedure to update the u-boot device tree.

6.1 U-boot : unpack and patch sources

i Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in `$WORKDIR/sources/arm-ostl-linux-gnueabi/u-boot-stm32mp-*`. Please notice a `grep "$>" README.HOW_TO.txt` describes the few commands needed to build the artifact of u-boot

```
PC $> pushd $WORKDIR
PC $> mkdir -p u-boot
PC $> tar xf sources/arm-ostl-linux-gnueabi/u-boot-stm32mp-*/v*.tar.gz -C u-boot
PC $> mv u-boot/u-boot* u-boot/u-boot-sources/
PC $> pushd u-boot/u-boot-sources
PC $> for p in ../../sources/arm-ostl-linux-gnueabi/u-boot-stm32mp-*/*.patch; do patch -p1 < $p; done
PC $> popd
```

6.2 U-boot : copy the DTS in the u-boot source code

```
PC $> pushd $WORKDIR
PC $> cp MyDeviceTree_fromCubeMX/u-boot/* u-boot/u-boot-sources/arch/arm/dts/
PC $> popd
```

Starting from *u-boot* 2019.04 version the device tree to be compiled must be explicitly added to the *dts Makefile*, `u-boot/u-boot-sources/arch/arm/dts/Makefile`:

```
dtb-y += stm32mp157c-mydevicetree-mx.dtb
targets += $(dtb-y)
```

6.3 U-boot : regenerate u-boot.stm32

i Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in `$WORKDIR/sources/arm-ostl-linux-gnueabi/u-boot-stm32mp-*`. Please notice a `grep "$>" README.HOW_TO.txt` describes the few commands needed to build the artifact of u-boot

```
PC $> pushd $WORKDIR/u-boot/u-boot-sources
PC $> make stm32mp15_<config>_defconfig
```



```
<config> : could be trusted or basic according the boot type
PC $> make DEVICE_TREE=<device tree> all
<device tree> : is the device tree just copied, i.e.: stm32mp157c-mydevicetree-mx
PC $> popd
PC $> ls -l $WORKDIR/u-boot/u-boot-sources/u-boot.stm32
```

6.4 U-boot : copy the u-boot into the target

- Because of 'extlinux' and before flashing the new u-boot.stm32, make sure #Updating extlinux is compliant with the 'compatible' value in the DTS file.
- Then flash the u-boot.stm32 into the 'ssbl' partition of the target using STM32CubeProgrammer.



7 Updating the TF-A device tree

To update the TF-A device tree, replace the DTB part of the TF-A binary.

The TF-A binary allocates a 'fixed' area for the DTB, just after the 'mkimage' headers. If the DTB is smaller than the reserved area, the remaining memory is padded with zero.

Below the procedure to generate TF-A with a new DTB and then flash it on the target:

7.1 TF-A : unpack and patch sources

i Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in `$WORKDIR/sources/arm-ostl-linux-gnueabi/tf-a-stm32mp-*`. Please notice a `grep "$>" README.HOW_TO.txt` describes the few commands needed to build the artifact of tf-a

```
PC $> pushd $WORKDIR
PC $> mkdir -p tf-a
PC $> tar xf sources/arm-ostl-linux-gnueabi/tf-a-stm32mp-[0-9]*/*.tar.gz -C tf-a
PC $> mv tf-a/tf-a-* tf-a/tf-a-sources
PC $> pushd tf-a/tf-a-sources
PC $> for p in ../../sources/arm-ostl-linux-gnueabi/tf-a-stm32mp-[0-9]*/*.patch; do patch -p1 < $p; done
PC $> popd
PC $> popd
```

7.2 TF-A : copy the DTS into the source code

```
PC $> pushd $WORKDIR
PC $> cp -r MyDeviceTree_fromCubeMX/tf-a/* tf-a/tf-a-sources/fdts/
PC $> popd
```

7.3 TF-A : regenerate TF-A.stm32

i Information

The procedure below is an extract of the README.HOW_TO.txt file which is available in `$WORKDIR/sources/arm-ostl-linux-gnueabi/tf-a-stm32mp-*`. Please notice a `grep "$>" README.HOW_TO.txt` describes the few commands needed to build the artifact of tf-a

```
PC $> pushd $WORKDIR/tf-a/tf-a-sources
PC $> make -f ../../sources/arm-ostl-linux-gnueabi/tf-a-stm32mp-[0-9]*-r0/Makefile.sdk TFA_DEVICETREE=<device tree>
TF_A_CONFIG=<config> all
    <config> : could be trusted or serialboot according the boot type
    <device tree> : is the device tree just copied, i.e.: stm32mp157c-mydevicetree-mx
```



```
PC $> popd
```

```
PC $> ls -l $WORKDIR/tf-a/build/<config>/tf-a-<device tree>-<config>.stm32
```

7.4 TF-A : copy the DTB in target/bootfs

Then flash the tf-a-*.stm32 into the 'fsbl1' and 'fsbl2' partitions of the target with STM32CubeProgrammer

Warning

'fsbl1' and 'fsbl2' are two redondant partitions and so, they should have same content

8 Update methods



8.1 Updating bootfs

There are two methods to update bootfs

- On an up and running target

```
PC $> scp stm32mp157c-mydevicetree-mx.dtb root@<Target_IP>:/boot/
```

- Directly into 'bootfs' image

You do not need to have a target up and running. Only the "st-image-bootfs-openstlinux-weston-stm32mp1.ext4" file is required. To modify an 'ext4' file, a loopback mount, available within any Linux Distribution (even through WSL2), is required:

```
PC $> mkdir -p $WORKING/bootfs
```

```
PC $> mount -o loop <st-image-bootfs-openstlinux-weston-stm32mp1.ext4> $WORKING/bootfs
```

```
##Then copy the new dtb file at the root of $WORKING/bootfs
```

```
PC $> umount $WORKING/bootfs
```

```
PC $> sync
```

```
PC $> dd if=<st-image-bootfs-openstlinux-weston-stm32mp1.ext4> of=/dev/mmcblk0p4 conv=fdatasync
```

Information

The '/dev/mmcblk0p4' is in case of the sdcard is inserted in dedicated drive of the PC, using an USB sdcard reader will probably create /dev/sdb4 entry.



8.2 Updating extlinux

8.2.1 extlinux basics

extlinux describes how u-boot boots. Updating **extlinux** consists in updating the extlinux.conf:

- In case of an DK-2 board booting from the sdcard. A **stm32mp157c-dk2_extlinux.conf** file is located in `/boot/mmc0_extlinux/`,
- otherwise if there is no specific extlinux.conf for your board then the **extlinux.conf** is taking into account.

extlinux.conf is the description of a boot menu with one or several entries; 'DEFAULT' selects the default entry.

Below an example of **extlinux.conf**:

```

menu title Select the boot mode
MENU BACKGROUND ../splash.bmp
TIMEOUT 5
DEFAULT stm32mp157c-mydevicetree-mx
LABEL stm32mp157c-dk2-sdcard
    KERNEL /uImage
    FDTDIR /
    APPEND root=PARTUUID=e91c4e10-16e6-4c0e-bd0e-77becf4a3582 rootwait rw
console=ttySTM0,115200
LABEL stm32mp157c-dk2-a7-examples-sdcard
    KERNEL /uImage
    FDT /stm32mp157c-dk2-a7-examples.dtb
    APPEND root=PARTUUID=e91c4e10-16e6-4c0e-bd0e-77becf4a3582 rootwait rw
console=ttySTM0,115200
LABEL stm32mp157c-dk2-m4-examples-sdcard
    KERNEL /uImage
    FDT /stm32mp157c-dk2-m4-examples.dtb
    APPEND root=PARTUUID=e91c4e10-16e6-4c0e-bd0e-77becf4a3582 rootwait rw
console=ttySTM0,115200
LABEL stm32mp157c-mydevicetree-mx
    KERNEL /uImage
    FDT /stm32mp157c-mydevicetree-mx.dtb
    APPEND root=PARTUUID=e91c4e10-16e6-4c0e-bd0e-77becf4a3582 rootwait rw
console=ttySTM0,115200

```

Please update/add the highlighted lines according to what have been compiled in chapter 5, 6 and/or 7:

- **DEFAULT**: This is the default 'LABEL' to boot
- **LABEL**: The entry 'LABEL' is the value of 'compatible' of the DTS file compiled with u-boot.
The 'compatible' value is at head of the DTS file and looks like : "st,stm32mp157c-mydevicetree-mx"
- **FDT**: The path from /boot of the kernel DTB to use

8.2.2 Updating extlinux

Updating 'extlinux' consists in modifying the extlinux.conf. There are two ways to do this:

- On an up and running target

Open an ssh connection to the target or use a direct connection with a tty terminal. Then use an vi editor to modify the extlinux.conf file.

Do not forget to synchronize the file system before rebooting the target:

Board \$> sync

- Into 'bootfs' image directly



You do not need to have a target up and running. Only the "st-image-bootfs-openstlinux-weston-stm32mp1.ext4" file is required. To modify an 'ext4' file, a loopback mount tool, available in any Linux Distribution (even through [WSL2](#)), is needed:

```
PC $> mkdir -p $WORKING/bootfs
```

```
PC $> mount -o loop <st-image-bootfs-openstlinux-weston-stm32mp1.ext4> $WORKING/bootfs
```

```
##Then edit the extlinux.conf file (for WSL2 use a 'Linux' type editor; vi, ...)
```

```
##Once extlinux.conf up-to-date, umount loopback and flash the bootfs into sdcard with STM32CubeProgrammer
```

Device Tree Binary (or Blob)

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Trusted Firmware for Arm Cortex-A

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

Device Tree Source (in software context) or Digital Temperature Sensor (in peripheral context)