How to compile the device tree with the Developer Package

How to compile the device tree with the Developer Package

Stable: 22.04.2020 - 13:13 / Revision: 20.04.2020 - 15:54

Contents

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

> 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

The procedure below is equivalent to **chapter 3 of the README_HOWTO.txt** file available in *$WORKDIR/sources/arm-openstlinux_weston-linux-gnueabi/linux-stm32mp-**

Run the following command into a shell:

PC $> pushd $WORKDIR
PC $> mkdir -p kernel
PC $> tar xf sources/arm-openstlinux_weston-linux-gnueabi/linux-stm32mp-4.19-r0/linux-4.19.*.tar.xz -C kernel
PC $> mv kernel/linux-* kernel/kernel-sources/
PC $> pushd kernel/kernel-sources/
PC $> for p in $(ls -1 ../../sources/arm-openstlinux_weston-linux-gnueabi/linux-stm32mp-4.19-r0/*.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

The procedure below is equivalent to **chapter 5 of the** README_HOWTO.txt file available in $WORKDIR/sources/arm-openstlinux_weston-linux-gnueabi/linux-stm32-*

PC $> pushd $WORKDIR/kernel/kernel-sources
**PC $>** make O="$PWD/../build" multi_v7_defconfig
PC $> for f in `ls -1 ../../sources/arm-openstlinux_weston-linux-gnueabi/linux-stm32mp-4.19-r0/fragment*.config`; do scripts/kconfig/merge_config.sh -m -r -O $PWD/../build $PWD/../build/.config $f; done
**PC $>** make oldconfig O="$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 it 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

The procedure below is equivalent to **chapter 3 of the** README_HOWTO.txt file available in $WORKDIR/sources/arm-openstlinux_weston-linux-gnueabi//u-boot-stm32mp-*

PC $> pushd $WORKDIR
PC $> mkdir -p u-boot
PC $> tar xf sources/arm-openstlinux_weston-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-openstlinux_weston-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

The procedure below is equivalent to **chapter 5.1 of the** README_HOWTO.txt file available in $WORKDIR/sources/arm-openstlinux_weston-linux-gnueabi//u-boot-stm32mp-*

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

The procedure below is equivalent to **chapter 3 of the** README_HOWTO.txt file available in $WORKDIR/sources/arm-openstlinux_weston-linux-gnueabi/tf-a-stm32mp-*

PC $> pushd $WORKDIR
PC $> mkdir -p tf-a
PC $> tar xf sources/arm-openstlinux_weston-linux-gnueabi/tf-a-stm32mp-*/v*.tar.gz -C tf-a
PC $> mv tf-a/arm-trusted-firmware-* tf-a/tf-a-sources
PC $> pushd tf-a/tf-a-sources
PC $> for p in ../../sources/arm-openstlinux_weston-linux-gnueabi/tf-a-stm32mp-*/*.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

The procedure below is equivalent to **chapter 5 of the** README_HOWTO.txt file available in $WORKDIR/st-image-weston-openstlinux-weston-stm32mp1/sources/arm-openstlinux_weston-linux-gnueabi/tf-a-stm32mp-*

PC $> pushd $WORKDIR/tf-a/tf-a-sources

PC $> make -f ../../sources/arm-openstlinux_weston-linux-gnueabi/tf-a-stm32mp-2.0-r0/Makefile.sdk TFA_DEVICETREE=*<device tree>* TF_A_CONFIG=*<config>* all

<config> : could be ***trusted*** or *basic* 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

'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, avaible 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/mmblk0p4** conv=fdatasync

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. The **extlinux.conf** file is located in */boot/mmc0_stm32mp157c-dk2_extlinux/*,
- otherwise if *mmc0_<something>_extlinux* directory is not available, **extlinux.conf** is located in */boot/extlinux/*.

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
        FDT /stm32mp157c-dk2.dtb
        APPEND root=/dev/mmcblk0p6 rootwait rw console=ttySTM0,115200
LABEL stm32mp157c-dk2-a7-examples-sdcard
        KERNEL /uImage
        FDT /stm32mp157c-dk2-a7-examples.dtb
        APPEND root=/dev/mmcblk0p6 rootwait rw console=ttySTM0,115200
LABEL stm32mp157c-dk2-m4-examples-sdcard
        KERNEL /uImage
        FDT /stm32mp157c-dk2-m4-examples.dtb
        APPEND root=/dev/mmcblk0p6 rootwait rw console=ttySTM0,115200
LABEL stm32mp157c-mydevicetree-mx
       KERNEL /uImage
       FDT /stm32mp157c-mydevicetree-mx.dtb
       APPEND root=/dev/mmcblk0p6 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, avaible 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)

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)