How to create your own machine

How to create your own machine

Stable: 21.02.2020 - 09:42 / Revision: 11.02.2020 - 08:41

Contents

1 Introduction

For your own needs, you can add in the Yocto project a new machine reflecting your own board and your own features.
This article is reserved to Yocto experts or at least people who have already practiced with the Yocto environmment.

This section describes how to add and configure a machine that is similar to those that the OpenSTLinux Distribution Package already supports.
This customer machine is associated to STM32CubeMX tool which provides DeviceTree files per component (tf-a, u-boot and kernel).

We suppose here that all the material described below is done inside an existing STM32MP BSP layer 'addons'.
For reminder this addons layer is deployed here in our delivery : **<path of STM32MP1_Distribution_Package>/layers/meta-st/meta-st-stm32mp-addons/**

2 Generate device tree

The principle is that the user generates device tree files from the STM32CubeMX tool.
With the STM32CubeMX tool, the user browses inside the OpenSTLinux Distribution Package file system until “mx” folder located into STM32MP BSP layer addons : **<path of STM32MP1_Distribution_Package>/layers/meta-st/meta-st-stm32mp-addons/mx/**

- 3 sub-folders are created and populated with generated device tree files :

- <ProjectName>/kernel
- <ProjectName>/u-boot
- <ProjectName>/tf-a

Where <ProjectName> is the “STM32CubeMX user project name”

Each directory of <ProjectName> contains device tree files associated to the component directory, here: kernel, u-boot and tf-a (Trusted Firmware-A).

3 Create a customer machine

Create a machine based on the one provided by ST and align some environment variables with the content of mx /<ProjectName> sub-folders

3.1 Create the new machine

```
  $> cd <path of STM32MP1_Distribution_Package>/layers/meta-st/meta-st-stm32mp-addons
/conf/machine
  $> cp stm32mp1-mx.conf stm32mp1-<ProjectName>.conf
```

3.2 Edit the new machine file: stm32mp1-<ProjectName>.conf

In the customer machine file, move to *User customizing sections* paragraph to configure your machine:

- **Boot Scheme** (default configuration is *trusted*)

To select your boot scheme configuration(s), comment and uncomment the *BOOTSCHEME_LABELS* lines.

- **Boot Device Choice** (default configuration is *sdcard*)

To select your boot device configuration(s), comment and uncomment the *FLASHLAYOUT_CONFIG_LABELS* lines.

- **Board Type Choice** (default configuration is *stm32mp157c-ev1*)

To select your original device tree configuration, comment and uncomment the *CUBEMX_DT_FILE_BASE* lines.

- **CubeMX Project config** (default configuration is empty)

You have to uncomment and configure the following variables to set your CubeMX project:
- *CUBEMX_DTB* name of CubeMX generated device tree files, without file extension (e.g. stm32mp157c-<ProjectName>-mx)
- *CUBEMX_PROJECT* path of CubeMX generated device tree files inside *mx* folder (e.g. <ProjectName>)

3.3 Create symbolic link for EULA with new machine created

To support GPU and third party content, you need to accept the EULA. So a symbolic link must be created with the EULA existing file and the new machine :

```
  $> cd <path of STM32MP1_Distribution_Package>/layers/meta-st/meta-st-stm32mp-addons
/conf/eula
  $> ln -s ST_EULA_SLA stm32mp1-<ProjectName>
```

4 Miscellaneous

4.1 MACHINEOVERRIDES regular expression

In each customer machine we retrieve this line:

```
MACHINEOVERRIDES .= ":stm32mpcommonmx"
```

A regular expression that resolves to one or more target machines with which a recipe is compatible.

The impact in the build environment is that recipes can now do things conditionally only on the builds for all mx specific machines: stm32mpcommonmx
It is very useful in some existing recipes to keep the compatibility for all stm32mp1-<ProjectName>.conf machines.
For instance use it like this (as it is done in the recipe <path of OpenSTLinux distribution delivery>/layers/meta-st/meta-st-stm32mp-addons/recipes-bsp/alsa/alsa-state-stm32mp1.bbappend):

```
SRC_URI_append_stm32mpcommonmx
```

The append will be applied for all machines *(all stm32mp1-<ProjectName>.conf)* which include **MACHINEOVERRIDES .= ":stm32mpcommonmx"**

So it avoids to create an append file by customer machine.

4.2 STM32CubeMX class

A dedicated class is provided here :

```
<STM32MP BSP layer addons>/classes/cubemx-stm32mp.bbclass
```

The main goal of this class is to manage any change done by the customer in sub folders mx/<ProjectName>/...

So if a device tree file is updated for one or more of components, this change will be taken into account automatically during the next compilation done in the Distribution Package.

5 Compile your image with the yocto build process

```
 $> cd <path of yocto delivery>
 (directory which contains meta-st, openembedded-core, meta-openembedded)

 $> MACHINE=stm32mp1-<ProjectName> DISTRO=openstlinux-weston source layers/meta-st
/script/envsetup.sh
Accept the term of EULA (if you agree with)

 $> bitbake st-image-weston
The generated images are available on build-openstlinuxweston-stm32mp1-<ProjectName>
/tmp-glibc/deploy/images/stm32mp1-<ProjectName>
```

Board support package

Device Tree

Device Tree Binary (or Blob)

Graphics Processing Units