How to compile the device tree with the Distribution Package

Contents

Stable: 17.11.2020 - 16:26 / Revision: 09.06.2020 - 08:49

Stable: 17.11.2020 - 10:20 | Revision: 30.30.2020 - 08:40

A quality version of this page, approved on *17 November 2020*, was based off this revision.

Contents

1 Introduction

This article is intended for Yocto experts, or people who have some practical experience of the Yocto environmment.

This section describes the steps needed to create and configure a demo layer using DeviceTree files from the STM32CubeMX tool, and to add and configure a machine similar to those already supported by the OpenSTLinux Distribution Package (in particular the machine delivered inside the existing STM32MP BSP layer 'addons').

Reminder: this addon-layer is deployed under the following path in the delivery : **<path of OpenSTLinux distribution delivery> /layers/meta-st/meta-st-stm32mp-addons/**

2 Creating a new open embedded layer for your demo

You first need to create a new layer. See the latest How to create a new open embedded layer

After creation, you have under **<path of OpenSTLinux distribution delivery>/layers/meta-st/**:

```
$ tree meta-my-demo-layer
meta-my-demo-layer
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-example
    └── example
        └── example.bb

3 directories, 4 files
```

2.1 Update layer.conf file

Open the layer.conf file and add the lines below for the licenses, demo layer path, and dependency with the STM32MP BSP layer 'addons' :

```
EULA_FILE_ST_stm32mpmydemo = "${LAYERDIR}/conf/eula/${MACHINE}"
EULA_FILE_ST_MD5SUM_stm32mpmydemo = "8b505090fb679839cefbcc784afe8ce9"

#Inform bitbake for adding another location to search for licenses
LICENSE_PATH += "${LAYERDIR}/files/licenses"

# Set a variable to get the STM32MP MX BSP location
STM32MP_MY_DEMO_BASE = "${LAYERDIR}"

# This should only be incremented on significant changes that may
# cause compatibility issues with other layers
LAYERVERSION_meta-my-demo-layer = "1"

LAYERDEPENDS_meta-my-demo-layer = "stm-st-stm32mp-mx"

# OpenEmbedded compatibility information
# This should only be incremented on significant changes that will
# cause compatibility issues with other layers
LAYERVERSION_meta-my-demo-layer = "1"
LAYERSERIES_COMPAT_meta-my-demo-layer = "dunfell"
```

Information

LAYERSERIES_COMPAT must be aligned with the version of OpenEmbedded used.
Please refer to https://wiki.yoctoproject.org/wiki/Releases

2.2 Create the machine for your demo

- Copy the machine delivered inside the existing STM32MP BSP layer 'addons' into your demo layer

```
$ cp <path of OpenSTLinux distribution delivery>/layers/meta-st/meta-st-stm32mp-addons
/conf/machine/stm32mp1-mx.conf <path of OpenSTLinux distribution delivery>/layers/meta-st
/meta-my-demo-layer/conf/machine/stm32mp1-demo.conf
```

- Open stm32mp1-demo.conf and update the line below

```
#@NEEDED_BSPLAYERS: layers/meta-openembedded/meta-oe layers/meta-openembedded/meta-python
layers/meta-st/meta-st-stm32mp-addons
```

- Replace STM32MP_MX_BASE by STM32MP_MY_DEMO_BASE
- Add these lines after the series of includes:

```
# Define specific common machine name
MACHINEOVERRIDES .= ":stm32mpmydemo"
```

- Uncomment variables to configure your own Boot Mode Choice, Boot Device Choice, Board Type Choice, DeviceTree files and path

2.3 Associate EULA with the new demo machine

Copy the eula folder delivered inside the existing STM32MP BSP layer 'addons' into your demo layer

```
$ cp -rf <path of OpenSTLinux distribution delivery>/layers/meta-st/meta-st-stm32mp-addons
/conf/eula/ <path of OpenSTLinux distribution delivery>/layers/meta-st/meta-my-demo-layer
/conf/.
```

Then replace the existing symbolic link with the machine used for your demo:

```
$ rm stm32mp1-mx
$ ln -s ST_EULA_SLA stm32mp1-demo
```

2.4 Move DeviceTree files and project coming from STM32CubeMX tool

The principle is that the user generates devicetree files for the targeted demo from the STM32CubeMX tool.

Warning

Most of the time, generated devicetree files - mainly user sections - must be reworked by the end user for compilation and functional purposes. Each demo is delivered with an application note that describe changes applied on STM32CubeMX devicetree files

.

These files are then moved into the “mx” folder created into your demo layer : **<path of OpenSTLinux distribution delivery> /layers/meta-st/meta-my-demo-layer/mx/**

Sub-folders are created and populated with the generated devicetree files:

```
mx/${CUBEMX_PROJECT}/kernel
mx/${CUBEMX_PROJECT}/u-boot
mx/${CUBEMX_PROJECT}/tf-a
mx/${CUBEMX_PROJECT}/optee-os
```

With CUBEMX_PROJECT that is equal to the value defined inside the machine used for the demo.

2.5 Update the README file

Please update the README file with the information needed for building and executing the demo.

2.6 Clean up useless content

You can delete the content of the recipes-example folder created by the create-layer command.

After making all of the updates, your demo layer should be similar to:

```
$ tree meta-my-demo-layer
meta-my-demo-layer
├── conf
│   └── eula
│       └── ST_EULA_SLA
│       └── stm32mp1-demo -> ST_EULA_SLA
│       └──  ...
│   └── layer.conf
│   └── machine
│       └── stm32mp1-demo.conf
├── COPYING.MIT
├── mx
│   └── STM32MP157C-EV1
│        └── my-demo
│            └── DeviceTree
│                └── my-demo
│                    └── kernel
│                        └── stm32mp157c-my-demo.dts
│                    └── tf-a
│                        └── stm32mp157c-my-demo.dts
│                        └── stm32mp15-mx.h
│                    └── u-boot
│                        └── stm32mp157c-my-demo.dts
│                        └── stm32mp157c-my-demo-u-boot.dtsi
│                        └── stm32mp15-mx.h
│                    └── optee-os
│                        └── stm32mp157c-my-demo.dts
└── README
```

3 Adding specific recipes and content necessary for your demo

Examples of further add-on components:

- Recipes for installing distro-specific configuration files
- Any image recipes specific to user distribution
- A *psplash append file* for a branded splash screen
- Any other append files to make custom changes

Some other added components (*bb) are more specific: images, system services, and so on (a non-exhaustive list is shown below):

- Recipes-core for *psplash screen*, *systemd services*
- Recipes-samples for example images

...

Board support package
Stable: 19.10.2020 - 14:17 / Revision: 19.10.2020 - 14:16

A quality version of this page, approved on *19 October 2020*, was based off this revision.

This article describes how to create and add a new layer to your STM32MPU Embedded Software Distribution Package for any development platform of the STM32MP1 family (for example an STM32MP157 Evaluation board). This can be done in order to modify some of its installed software, or to add new applications. It first explains the OpenEmbedded principle for customization by distribution, and then gives a step-by-step approach to creating a new layer and adding it it to your distribution. Finally, it gives guidelines on how to upgrade (add, remove, configure, improve...) any piece of installed software.

Contents

1 Why create a layer?

The OpenEmbedded distribution comes with a set of layers that provide the different pieces of software used to build images.

Checking the layers included in your distribution is easy with the **bitbake-layers** command tool:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- List your layers and priority:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
...
```

When you want to integrate or modify a distribution , the **OpenEmbedded principle is to append modifications/addons in a dedicated place** outside of the already available distribution layers.
This place is a new layer that you add to the distribution layers and feed with your modifications or add-ons. Hence the first thing to do when you want to integrate your modifications into a distribution is to create a new layer.

2 How to proceed

2.1 Tools

The openembedded-core layer provides a utility tool called **bitbake-layers**, which performs actions associated with layers. It allows you to perform actions including the following:

- see the list of actual layers taken into account in your configuration
- add existing layers to your configuration
- create a new layer

For an exhaustive list of options and subcommands:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Display bitbake-layers help:

```
$ bitbake-layers --help
NOTE: Starting bitbake server...
usage: bitbake-layers [-d] [-q] [-F] [--color COLOR] [-h] <subcommand> ...

BitBake layers utility

optional arguments:
  -d, --debug           Enable debug output
  -q, --quiet           Print only errors
  -F, --force           Force add without recipe parse verification
  --color COLOR         Colorize output (where COLOR is auto, always, never)
  -h, --help            Show this help message and exit

subcommands:
  <subcommand>
    show-layers         Show current configured layers
    show-overlayed      List overlayed recipes (where the same recipe exists
                        in another layer)
    show-recipes        List available recipes, showing the layer by which they
                        are provided
    show-appends        List bbappend files and recipe files to which they apply
    show-cross-depends  Show dependencies between recipes that cross layer
                        boundaries
    add-layer           Add a layer to bblayers.conf
    remove-layer        Remove a layer from bblayers.conf
    flatten             flatten layer configuration into a separate output
                        directory
    layerindex-fetch    Fetches a layer from a layer index along with its
                        dependent layers, and adds them to conf/bblayers.conf
    layerindex-show-depends
                        Find layer dependencies from layer index.
    create-layer        Create a basic layer

Use bitbake-layers <subcommand> --help to get help on a specific command
```

2.2 Creating a new layer

2.2.1 Available layers from configuration

You may first check the list of available layers from your configuration and their priority:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Then get the list of layers for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
```

Information

priority**: indicates the order followed by bitbake to apply rules from layers. When two layers have a '.bbappend' file based on the same recipe, the rules defined by the layer having the lowest priority are applied first, and then the rules defined by the layer having the highest priority are applied. This allows a layer with higher priority than the other layers to override any rules defined by the other layers.**

For more information about priority see the documentation: Priority doc

2.2.2 Create the new layer

To create a new layer you need to specify:

- the location of the new layer
- the name of the new layer
- (optionally) the priority of the new layer.

As an example, let's create the new layer *meta-my-custo-layer* on *meta-st* directory with priority set to *7*:

```
$ bitbake-layers create-layer --priority 7 ../layers/meta-st/meta-my-custo-layer
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer'
```

```
$ cd ../layers/meta-st
$ tree meta-my-custo-layer
meta-my-custo-layer
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-example
    └── example
        └── example.bb

3 directories, 4 files
```

2.2.3 Recommended actions on a new layer

- Update the README file
- Update the priority in the **conf/layer.conf** file (if it's not the expected one)
- Apply your own modifications and/or addons for the distribition

Information

Note: you can refer to OpenEmbedded - devtool to learn how to add new applications, modify existing applications and more.

2.3 Add the new layer to your configuration

2.3.1 Generic case

You may first check the list of available layers from your configuration and their priority:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Then get the list of layers for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
```

- You can then add the new layer *meta-my-custo-layer* to your build configuration:

Information

Make sure that you run the command from your build directory

```
$ bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer/
```

- Then check that it has been properly enabled for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
```

```
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
meta-my-custo-layer   /local/openstlinux-18-01-23/layers/meta-st/meta-my-custo-layer  7
```

2.3.2 With STMicroelectronics Distribution Package

Pre-requisite:

- Your custom layers are integrated in git.

To apply the new layer on top of fresh Distribution Package delivery:

- Download your layer using the *git clone* command under the Distribution Package delivery folder tree
- Source your build environment:

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Add your layer *meta-my-custo-layer* to your build configuration

```
bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer/
```

Initial ramdisk (https://en.wikipedia.org/wiki/Initial_ramdisk)