



life.augmented

Distribution package classroom

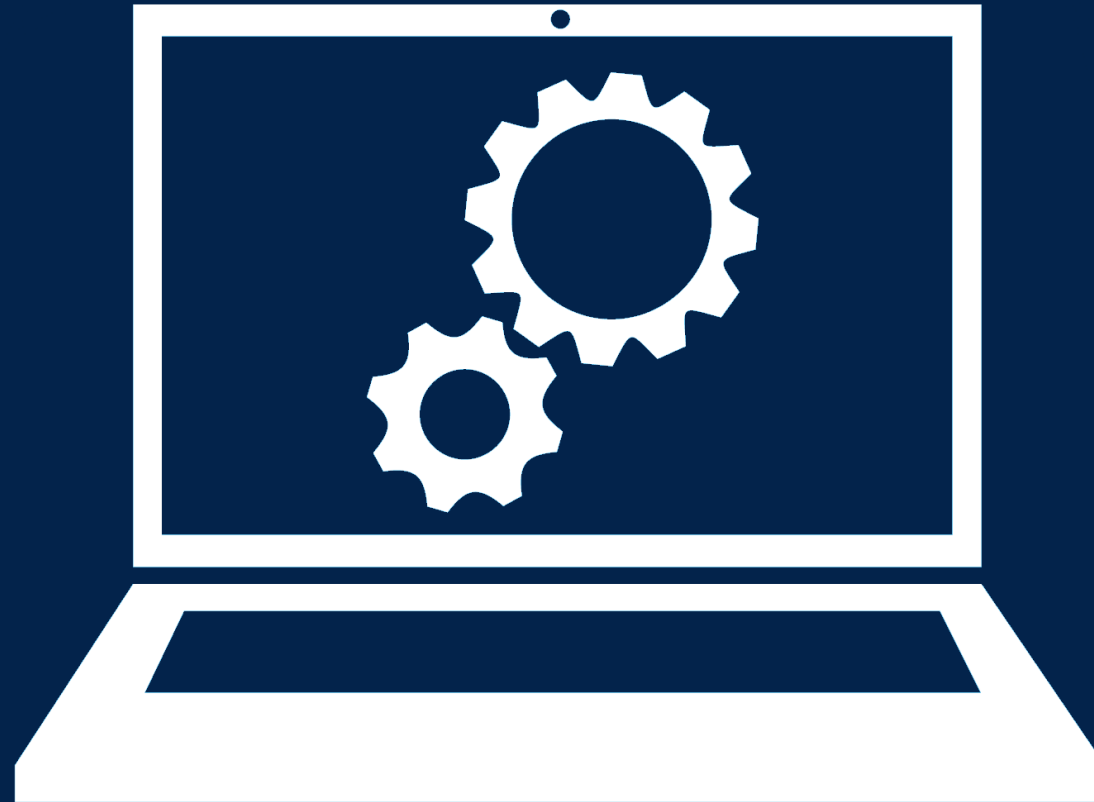
Cindy Ge

March 2021

Agenda

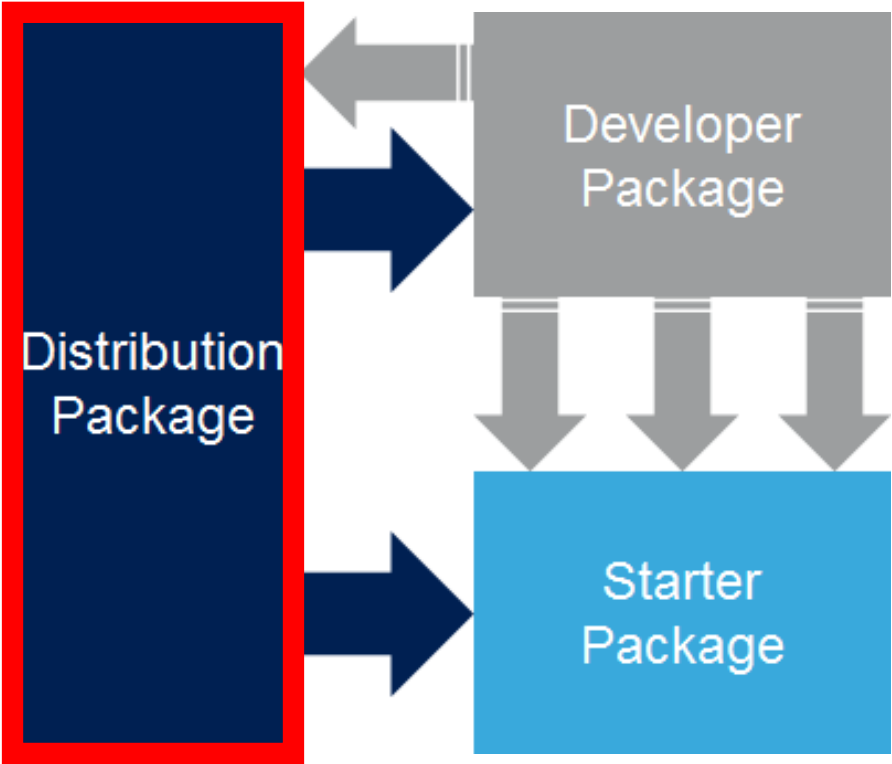
- 1 Distribution Package overview
- 2 Distribution Package Development
- 3 Ycoto Project
- 4 Reference

Distribution Package Overview



Distribution Package Overview

- The distribution package can generate the full starter package and the SDK needed in the developer package.
- The distribution package is based on Yocto/OpenEmbedded tools
- The main role of the distribution package is to manage the packages needed by the customer in the Linux environment
- Able to generate:
 - All the file system (bootfs, vendorfs, rootfs, userfs)



Distribution Package Overview

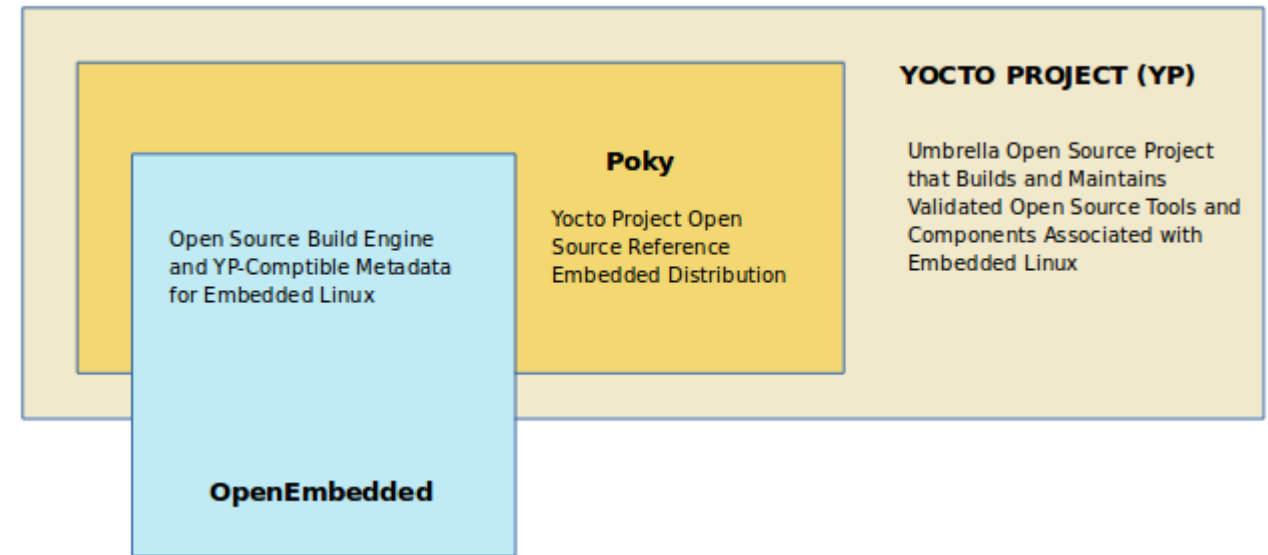
Distribution Package provides:

- a **build framework based on OpenEmbedded** (aka distribution builder)
- for the **OpenSTLinux distribution** (development on Arm Cortex-A processor), all pieces of software in **source code**: the BSP (Linux kernel, U-Boot, TF-A, optionally OP-TEE), and the application frameworks (Wayland-Weston, GStreamer, ALSA...)
- for the **STM32Cube MPU Package** (development on Arm Cortex-M processor), all pieces of software in **source code**: BSP, HAL, middlewares and applications
- a toolset to tune the system for your needs, and to handle the built image (e.g. STM32CubeProgrammer to install the built image on the board)

Distribution Package Overview

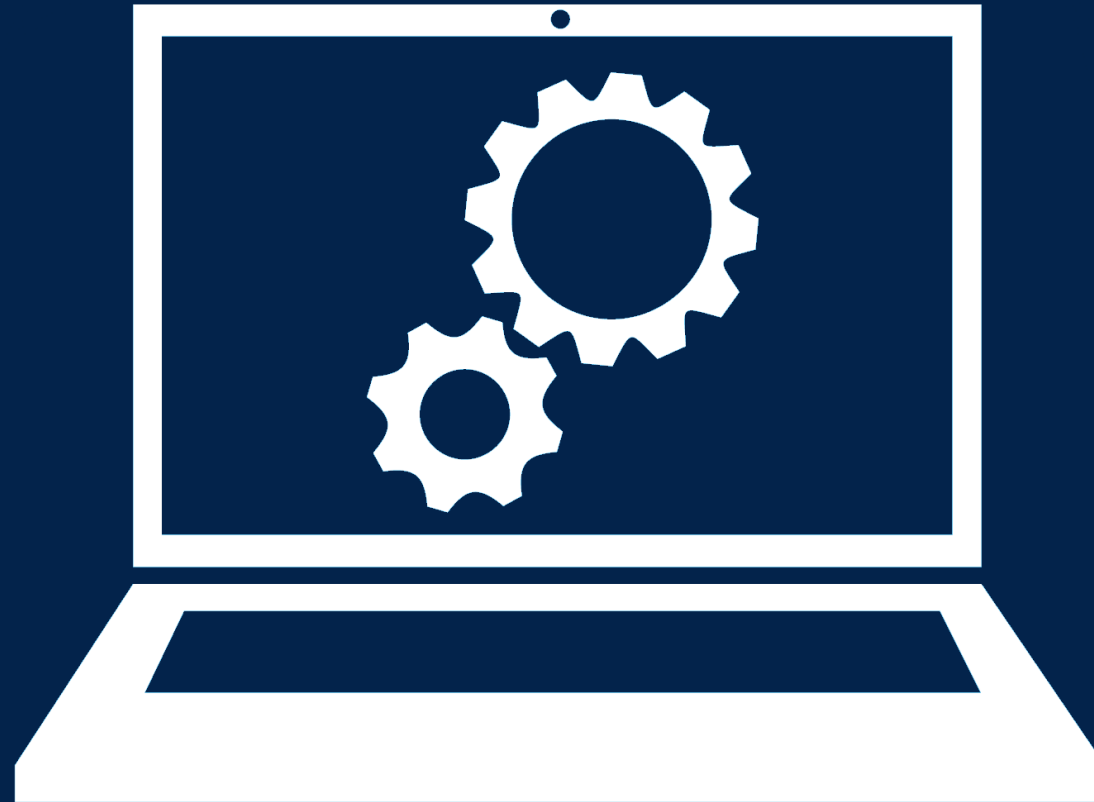
Concept of Yocto, Poky and OpenEmbedded:

- OpenEmbedded:
 - Build Framework for embedded linux
 - Maintained by the community
 - Source version of Poky
 - Setup mainly consolidated for ARM platforms
- Yocto
 - A project uses OpenEmbedded build system
- Poky
 - Poky is a reference system of the Yocto Project. Poky is used to validate the Yocto Project.
 - Poky is not a product level distro. Rather, it is a good starting point for customization.



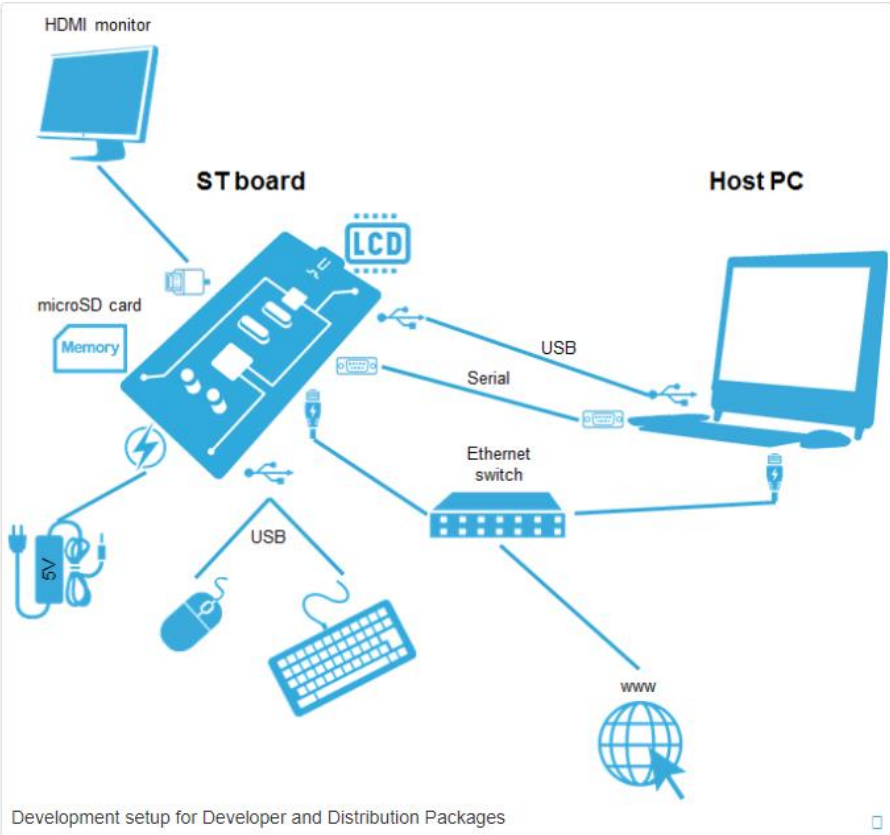
*Some projects will work on a Yocto base, some on a Poky base, but in the end, **everything is compatible***

Distribution Package Development



Development Setup

- Host PC for cross-compilation and cross-debugging.
- 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)
- 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)
- your board boots successfully

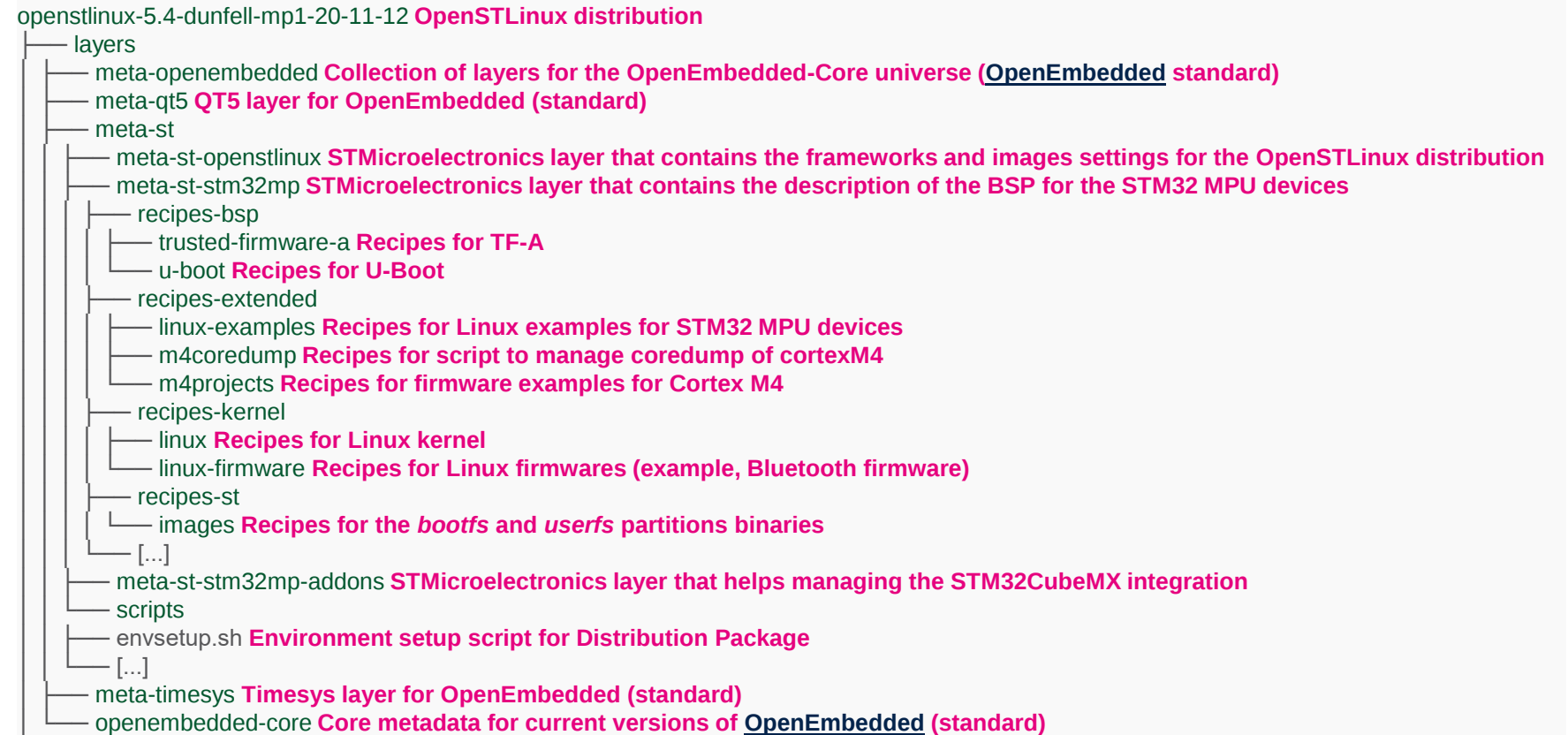


Installation Directory Overview

- Please follow the wiki page to do installation:

https://wiki.st.com/stm32mpu/wiki/STM32MP1_Distribution_page

- The installation path in Vmware: */work/distribution*



Build Distribution Package

- Initializing the OpenEmbedded build environment

PC \$> DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/envsetup.sh

Notes:

- The OpenEmbedded environment setup script must be run once in each new working terminal in which you use the BitBake or devtool tools (see later).
- openstlinux-weston (OpenSTLinux distribution featuring Weston/Wayland) and stm32mp1 (machine configuration for all STM32MP1 hardware configurations) are the default values for DISTRO and MACHINE

- Building the OpenSTLinux distribution

PC \$> bitbake st-image-Weston

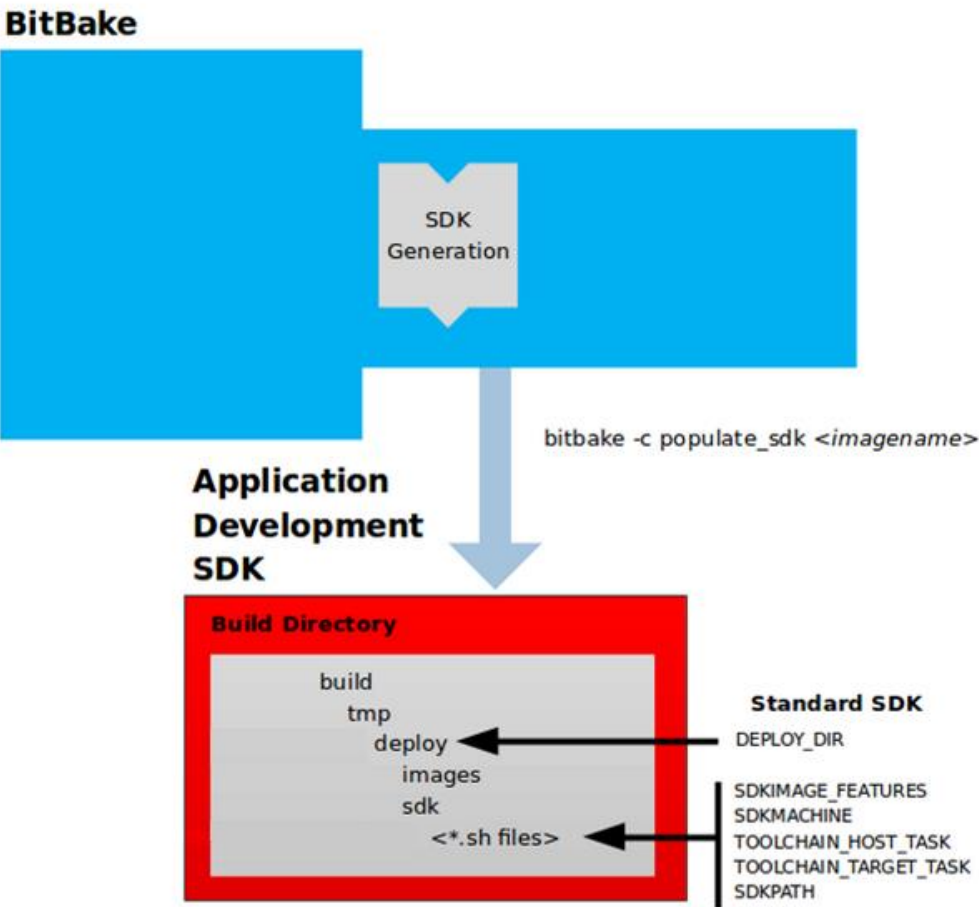
- Flashing the built image

The *build-<distro>-<machine>/tmp-glibc/deploy/images/stm32mp1* directory receives complete file system images.

SDK Generation

The *do_populate_sdk* task helps to create the standard SDK and handles two parts: a target part and a host part.

The target part is built for the target hardware and includes libraries and headers. The host part is the part of the SDK that runs on the host machine.



SDK Generation

- Check that the build environment script has been executed, and that the current directory is the build directory of the OpenSTLinux distribution (for example, *openstlinux-20-06-24/build-openstlinuxweston-stm32mp1*)

```
PC $> DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/envsetup.sh
```

- Generate the SDK installation files (including the installation script) for a standard SDK with the following command :

```
PC: $> bitbake -c populate_sdk <image>
```

Example:

```
PC $> bitbake -c populate_sdk st-image-weston
```

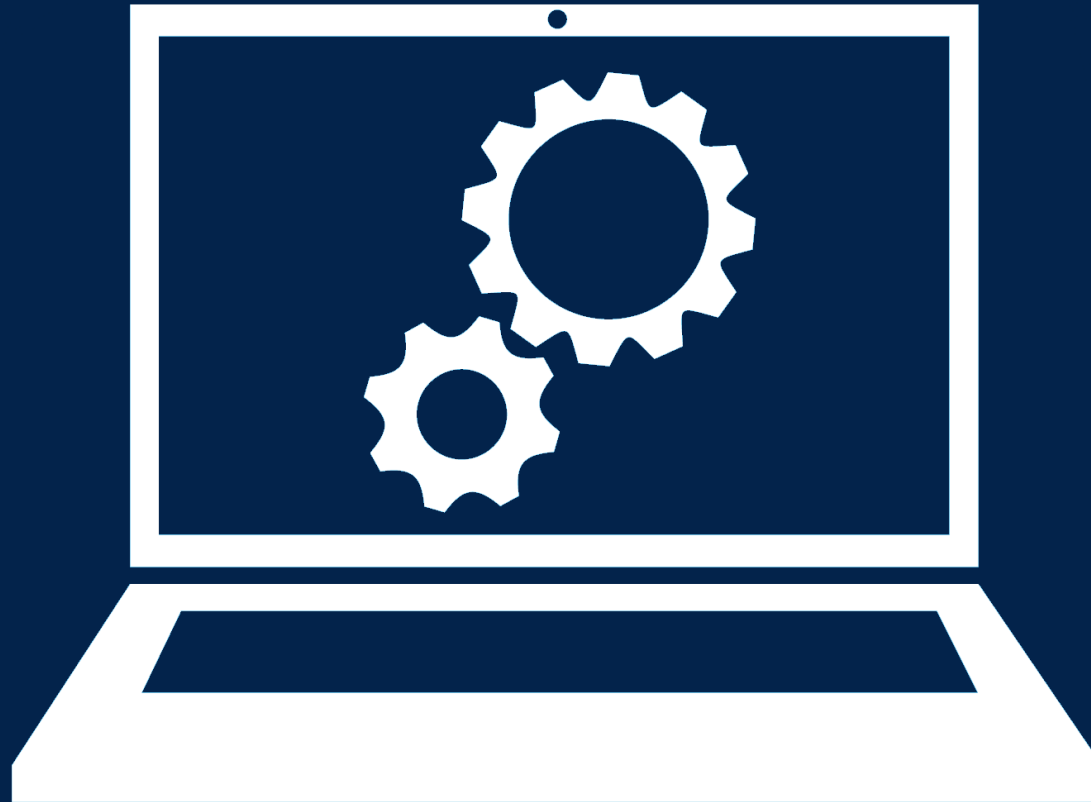
SDK Generation

The SDK installation files (*<image>-<distro>-<machine>-<host machine>-toolchain-<Yocto release>-snapshot.**) are written to the *deploy/sdk* directory inside the build directory.

- *<host machine>*: Host machine on which SDK is generated, *x86_64* (64-bit host machine; this is the only supported value).
- *<Yocto release>*: Release number of the Yocto Project; For example, *3.1* (aka *dunfell*).

```
PC $> ls tmp-glibc/deploy/sdk/  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.host.manifest  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.license  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot-license_content.html  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.sh  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.target.manifest  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.testdata.json
```

Yocto Project



Yocto Project: Overview (1/3)

What is the Yocto Project?

- An open-source collaboration project that helps developers create custom Linux-based systems that are designed for embedded products regardless of the product's hardware architecture.
- Yocto is more properly referred to as a meta-distribution. It is a collection of recipes, configuration values, and dependencies that are used to create a custom Linux runtime image tailored to your specific needs.



Yocto Project: Overview (2/3)

Features

- Widely adopted across the industry
- Architecture agnostic, supports Intel, ARM, MIPS, AMD, PPC and other architectures
- Images and code transfer easily
- Flexibility
- Tools and SDK
- Mechanism rules over policy
- Uses a layer model (will introduce later)
- Supports partial builds

Yocto Project: Overview (3/3)

Released version

2.5 Sumo 2.6 Thud (OpenSTLinux1.2.0) 2.7 Warrior 3.0 Zeus 3.1 Dunfell

Development tools

- **CROPS**: Provides an easily managed, extensible environment that allows you to build binaries for a variety of architectures on Windows, Linux and Mac OS X hosts.
- **devtool**: As part of the extensible SDK (eSDK), help to optionally integrate what you build into an image built by the OpenEmbedded build system
- **Extensible Software Development Kit (eSDK)**: Provides a cross-development toolchain and libraries tailored to the contents of a specific image.

Yocto Project: Basic terms(1/2)

Some basic terms

- **Metadata:** A key element of the Yocto Project is the Metadata that is used to construct a Linux distribution and is contained in the files that the OpenEmbedded build system parses when building an image. Metadata includes recipes, configuration files, and other information that refers to the build instructions themselves, as well as the data used to control what things get built and the effects of the build.
- **Layer:** A collection of related recipes. Layers allow you to consolidate related metadata to customize your build. Layers also isolate information used when building for multiple architectures.
- **Container Layer:** Layers that hold other layers.
- **Classes:** class files (.bbclass) contain information that is useful to share between recipes files.
- **Machine:** machine metadata (configuration of a hardware)

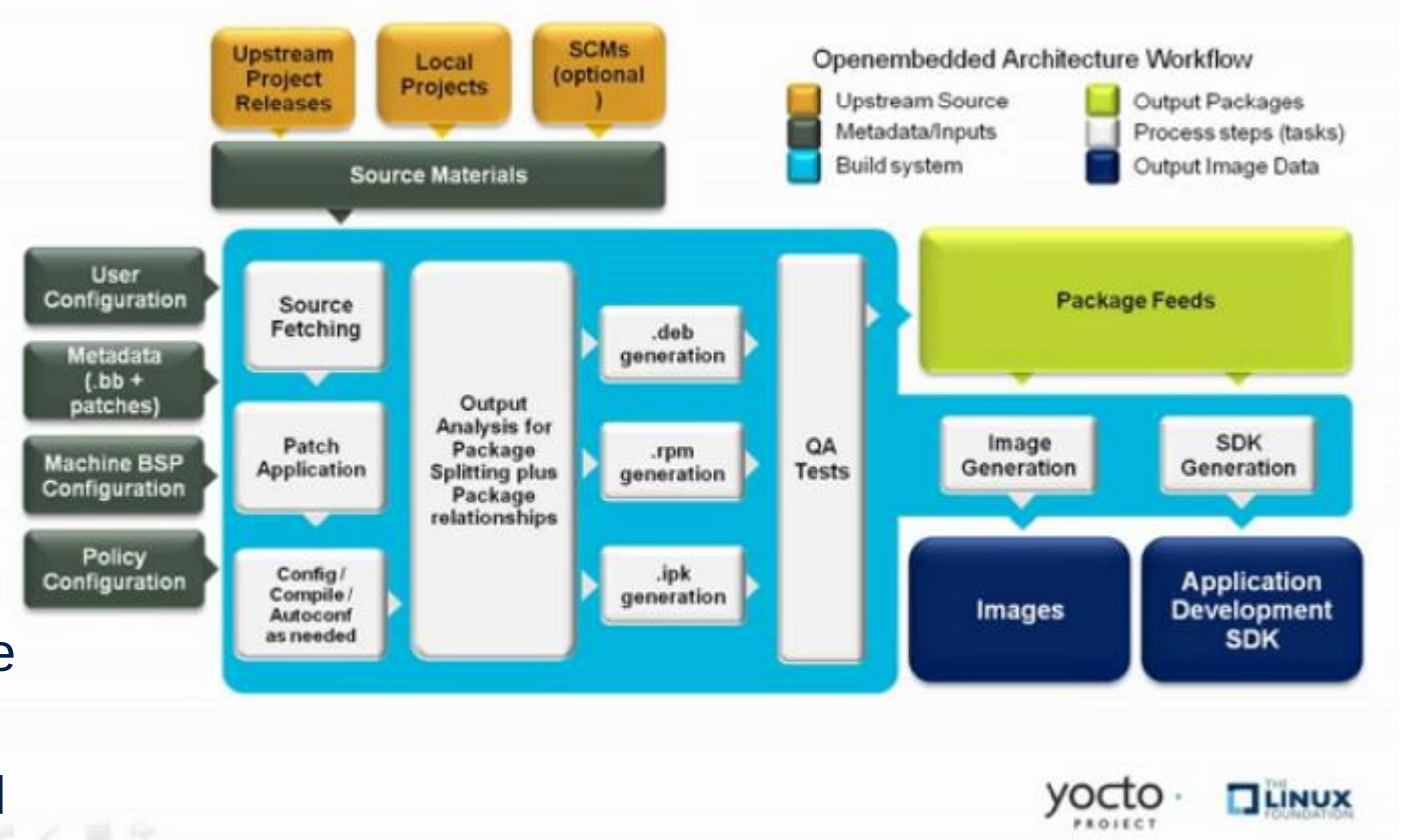
Yocto Project: Basic terms(2/2)

- **Configuration File:** Files (end with a .conf filename extension) that hold global definitions of variables, user-defined variables, and hardware configuration information. These files tell the OpenEmbedded build system what to build and what to put into the image to support a particular platform.
- **Recipe:** files that have the .bb suffix, a set of instructions for building packages. A recipe describes where you get source code, which patches to apply, how to configure the source, how to compile it and so on.
- **Append Files:** Files that append build information to a recipe file. The append file and corresponding recipe file must use the same root filename.(e.g. formfactor_0.0.bb and formfactor_0.0.bbappend).
- **Image:** set of software present on image ready to flash on board
- **BitBake:** The task executor and scheduler used by the OpenEmbedded build system to build images. For more information on BitBake, see the [BitBake User Manual](#) or command “bitbake -h”.

Yocto Project: Build system Workflow

OpenEmbedded Build system Workflow

- Developer specify architecture, policies, patches and configuration details.
- The build system fetches and download the source code
- Extracts the source code into a local work area where a patches are applied and common steps for configuring and compiling are run.
- Build system then install the software into a temporary staging area.
- Generates the file system image and SDK



Yocto Project: Bitbake Introduction

Concepts

BitBake is a program written in the Python language. At the highest level, BitBake interprets metadata, decides what tasks are required to run, and executes those tasks. Similar to GNU Make, BitBake controls how software is built. GNU Make achieves its control through "makefiles", while BitBake uses "recipes".

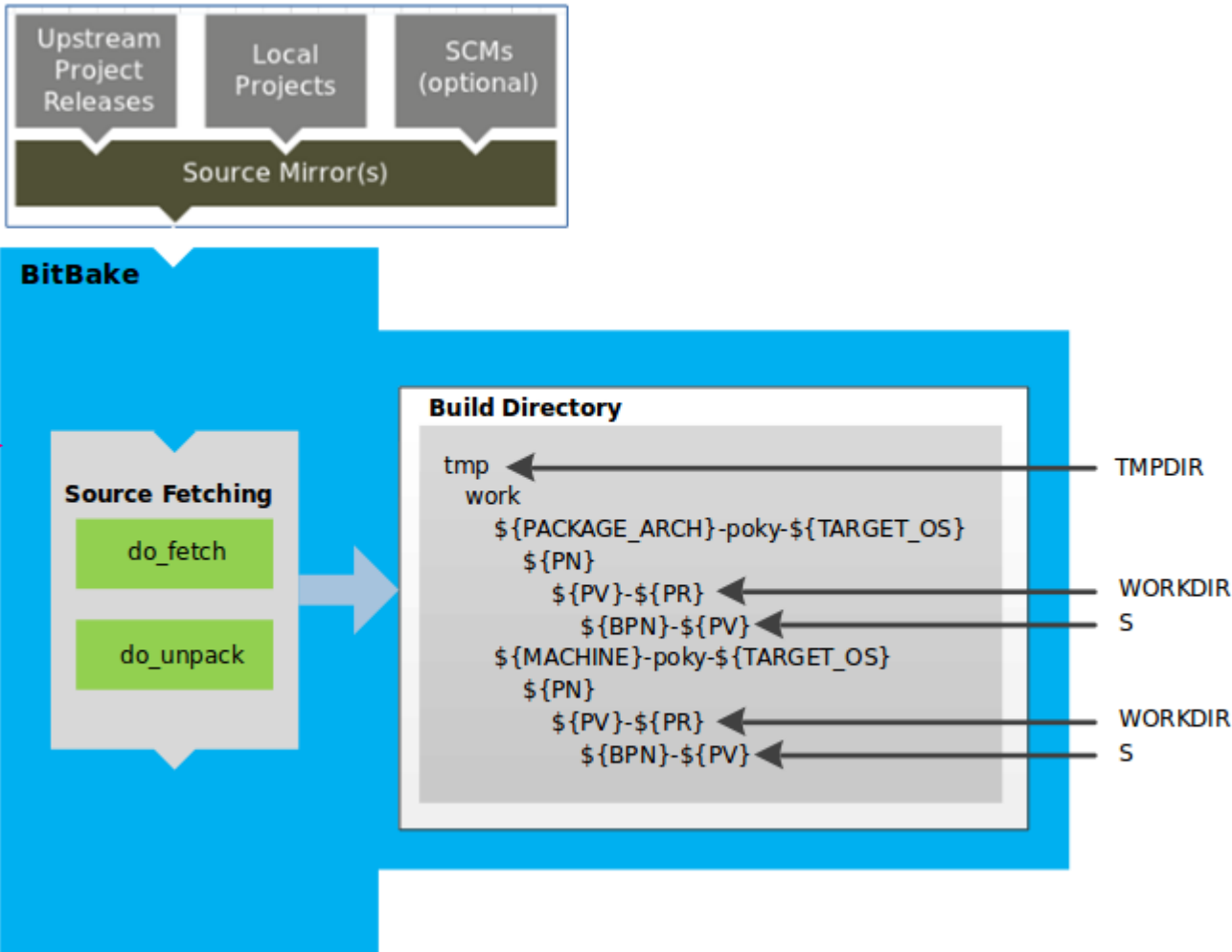
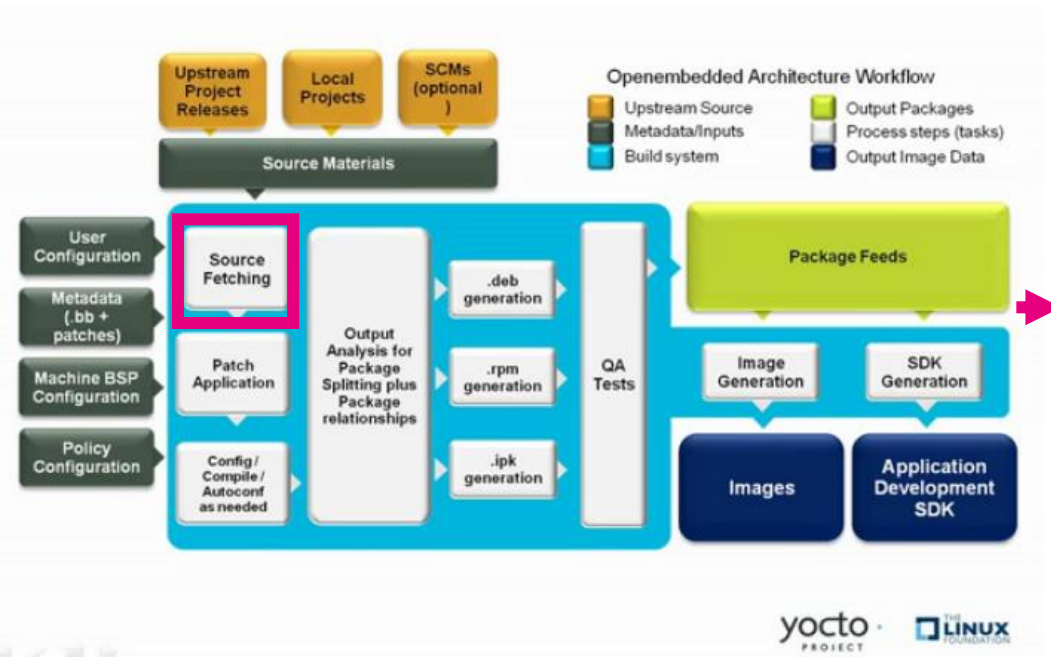
Obtaining BitBake

- Cloning BitBake: `$ git clone git://git.openembedded.org/bitbake`
- Using the BitBake that Comes With Your Build Checkout
- Taking a snapshot of BitBake

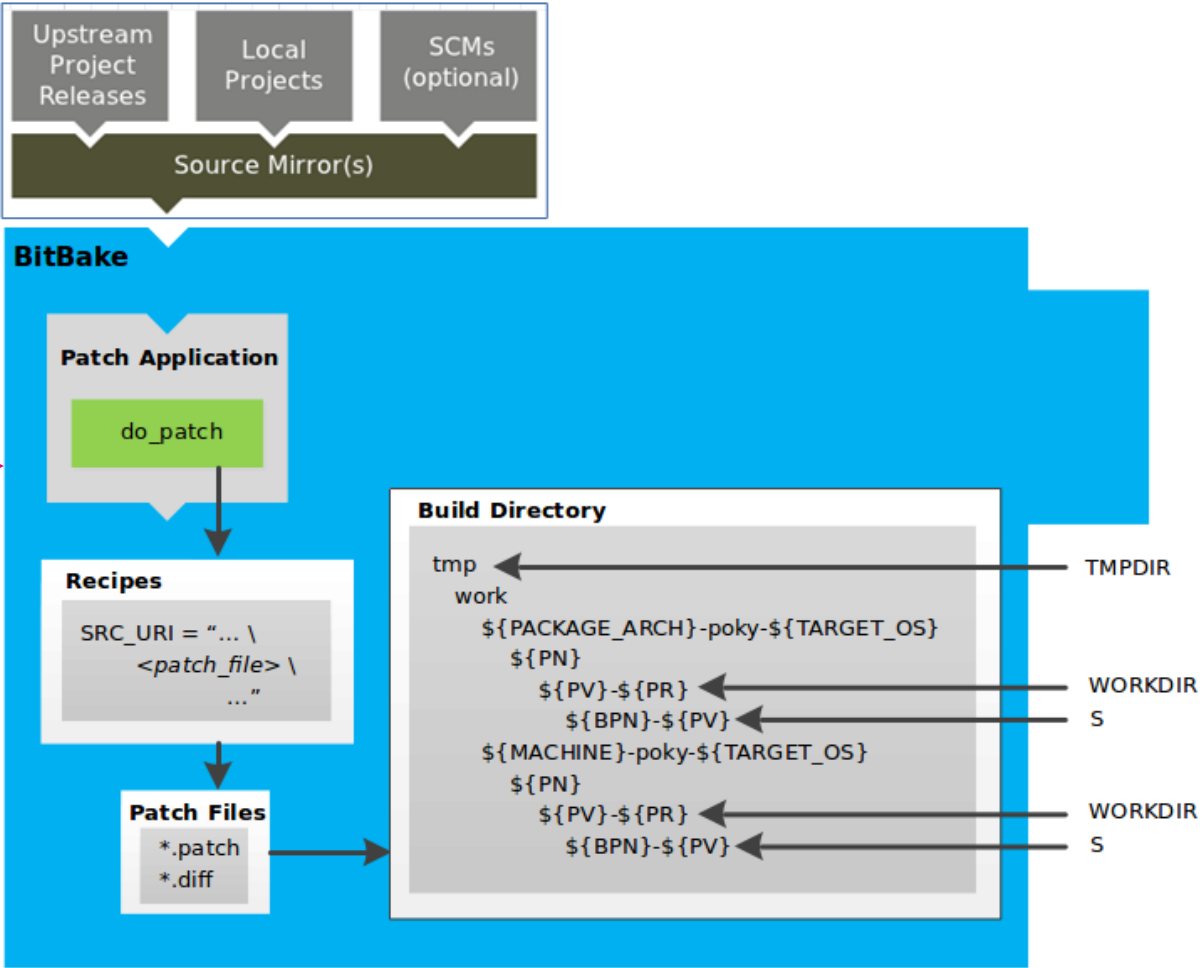
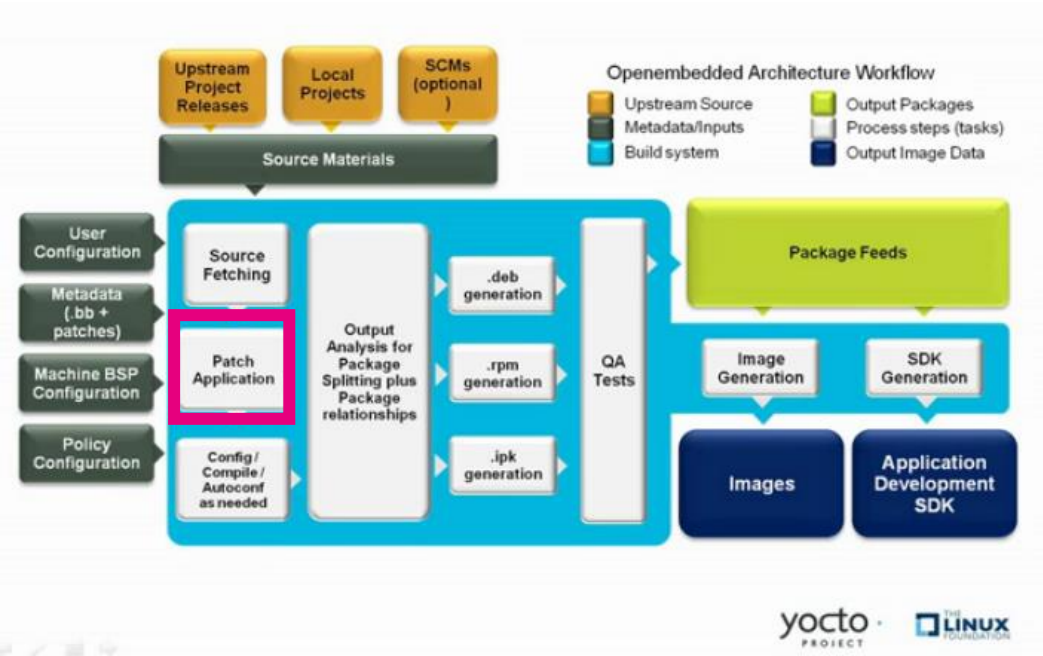
Bitbake command usage and syntax

PC: `$> bitbake -h`

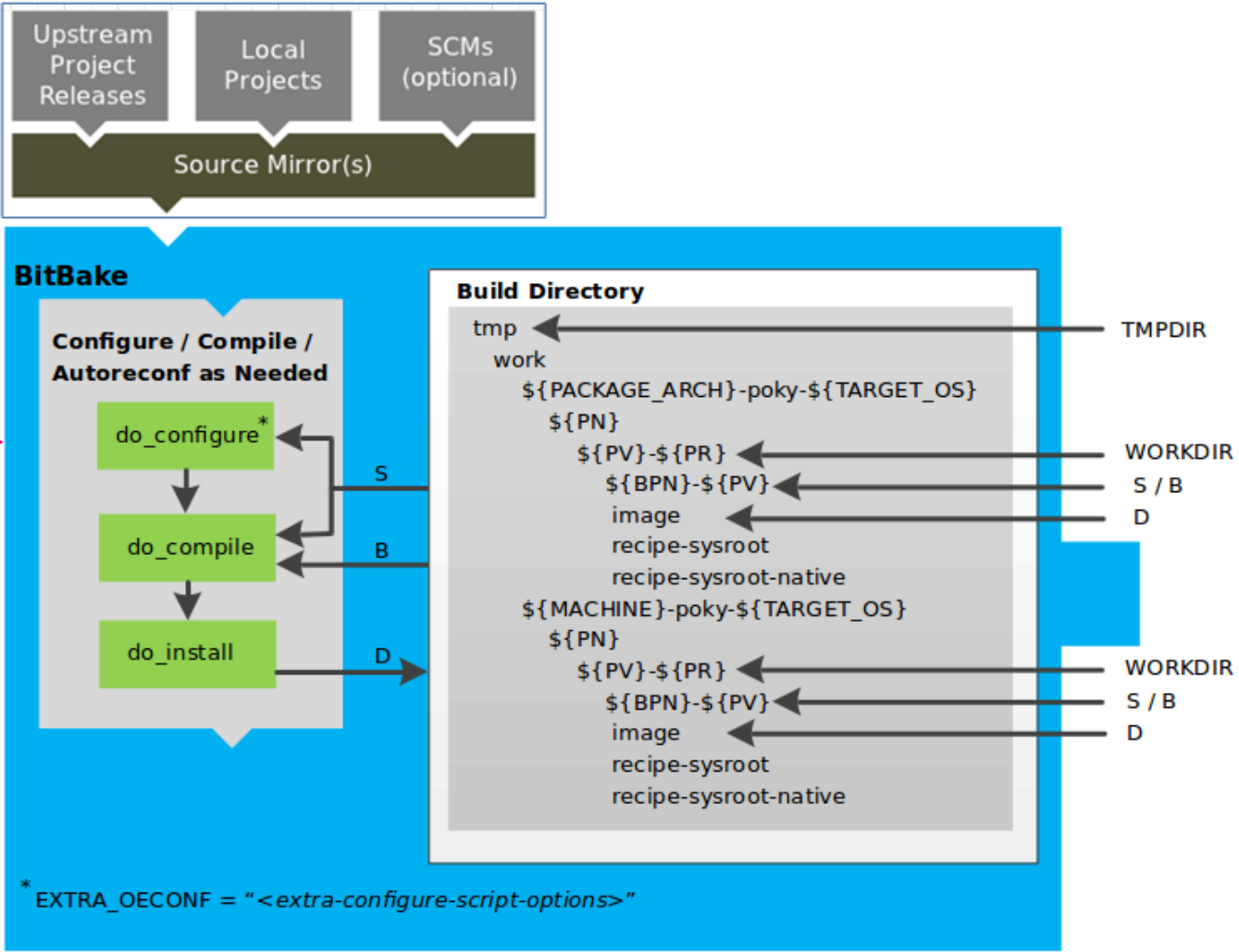
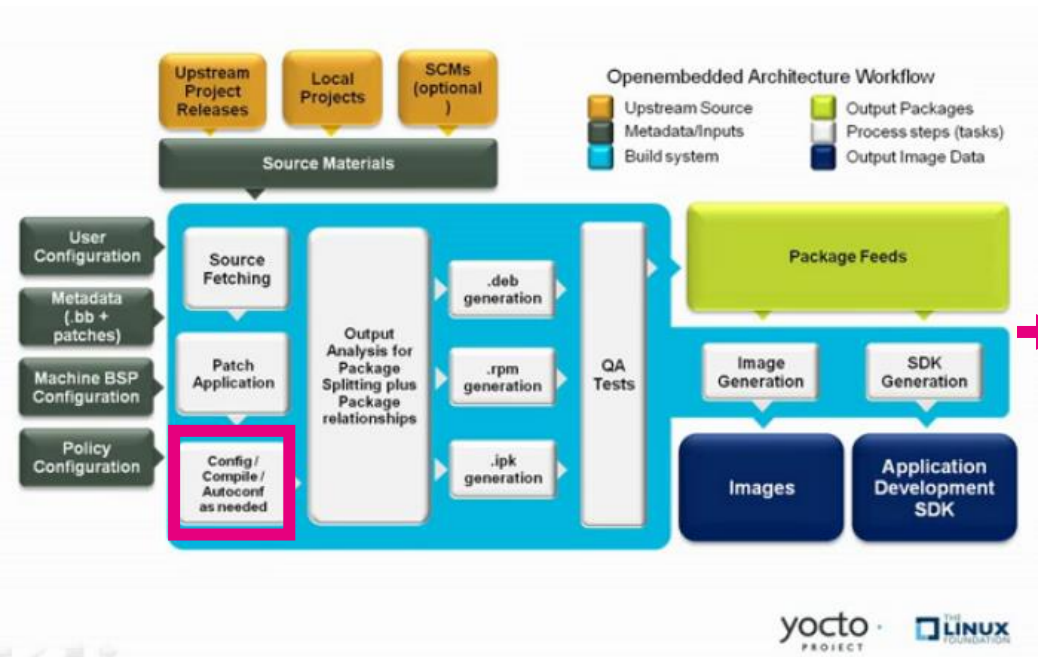
Yocto Project: Bitbake Workflow (1/5)



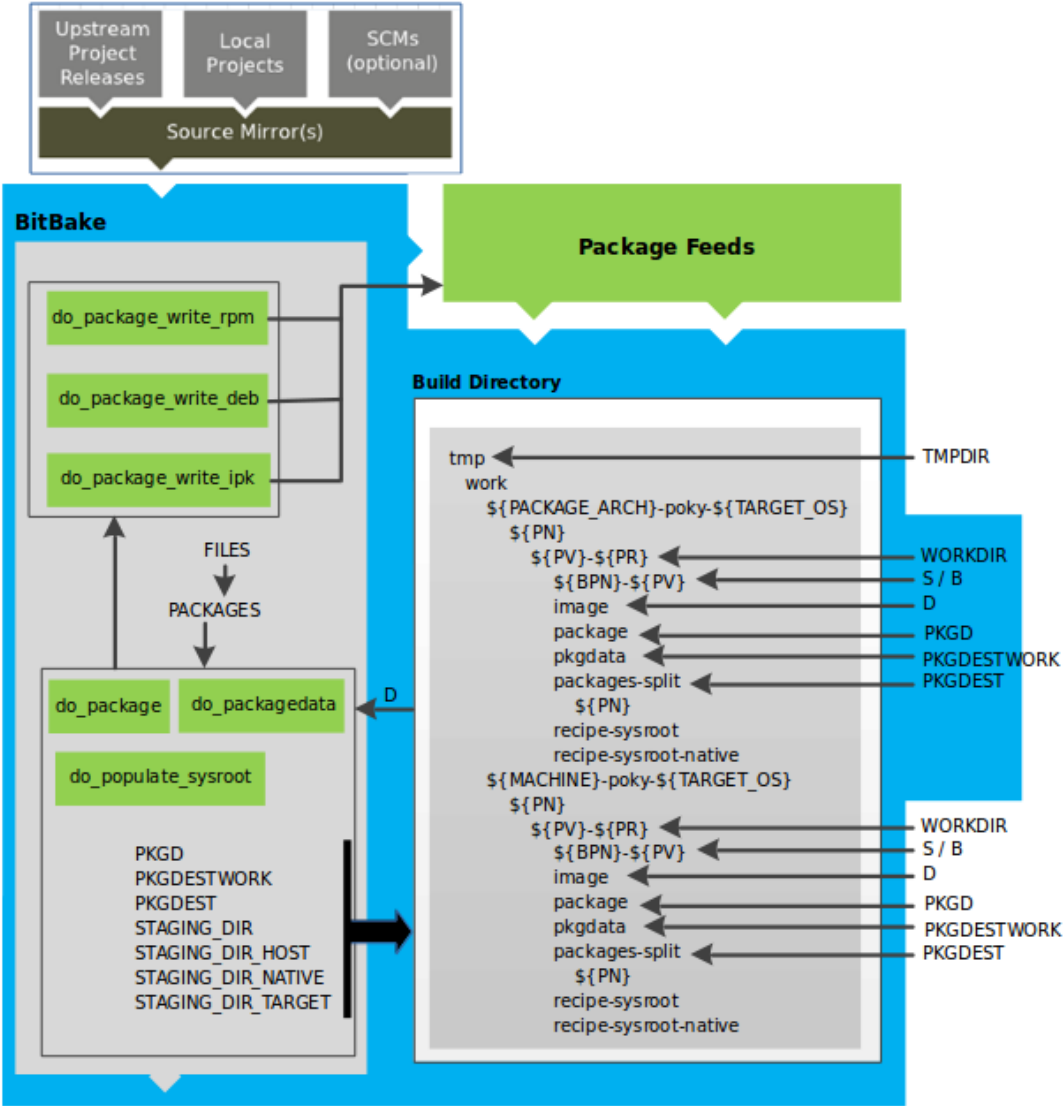
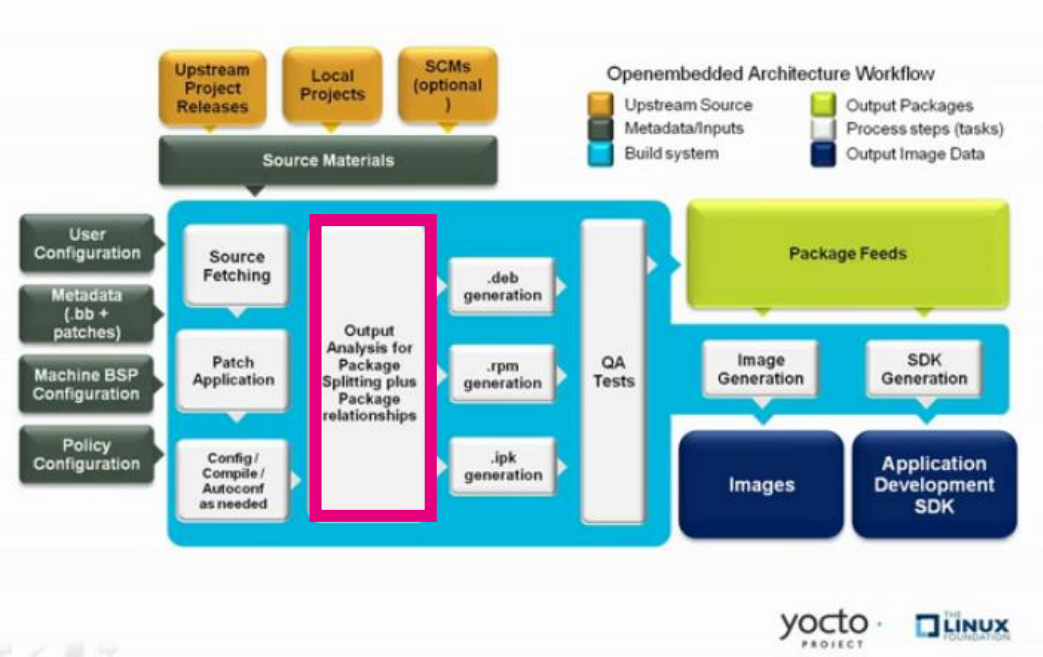
Yocto Project: Bitbake Workflow (2/5)



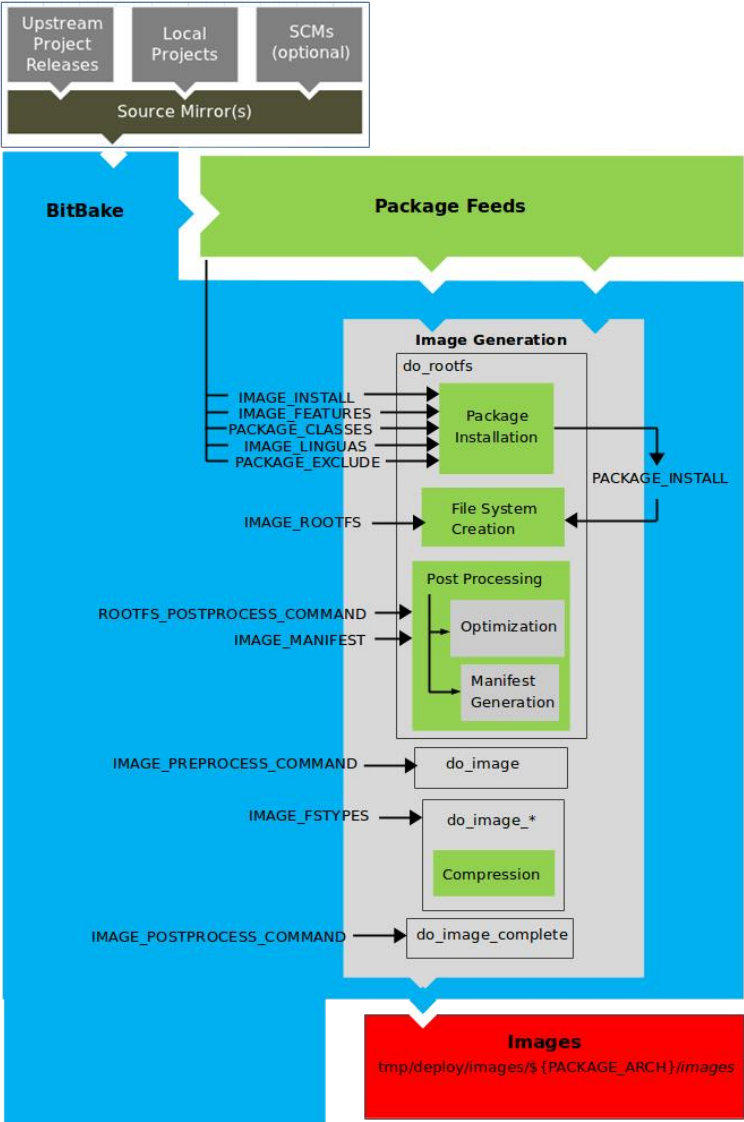
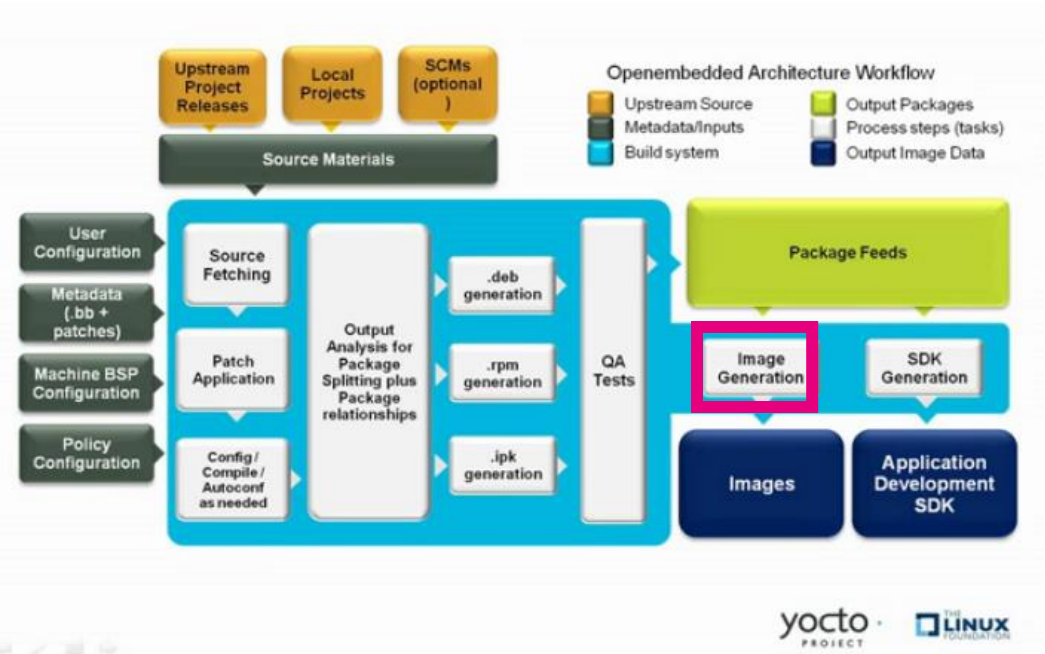
Yocto Project: Bitbake Workflow (3/5)



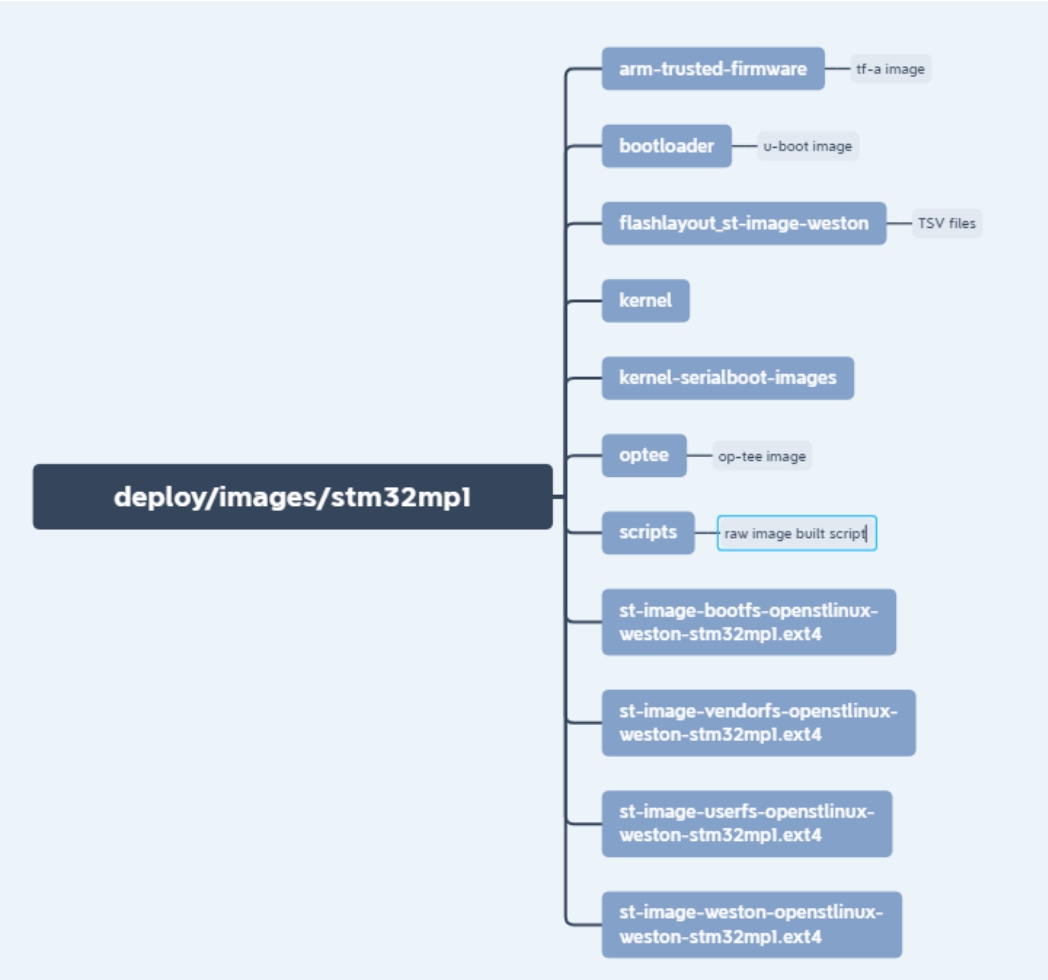
Yocto Project: Bitbake Workflow (4/5)



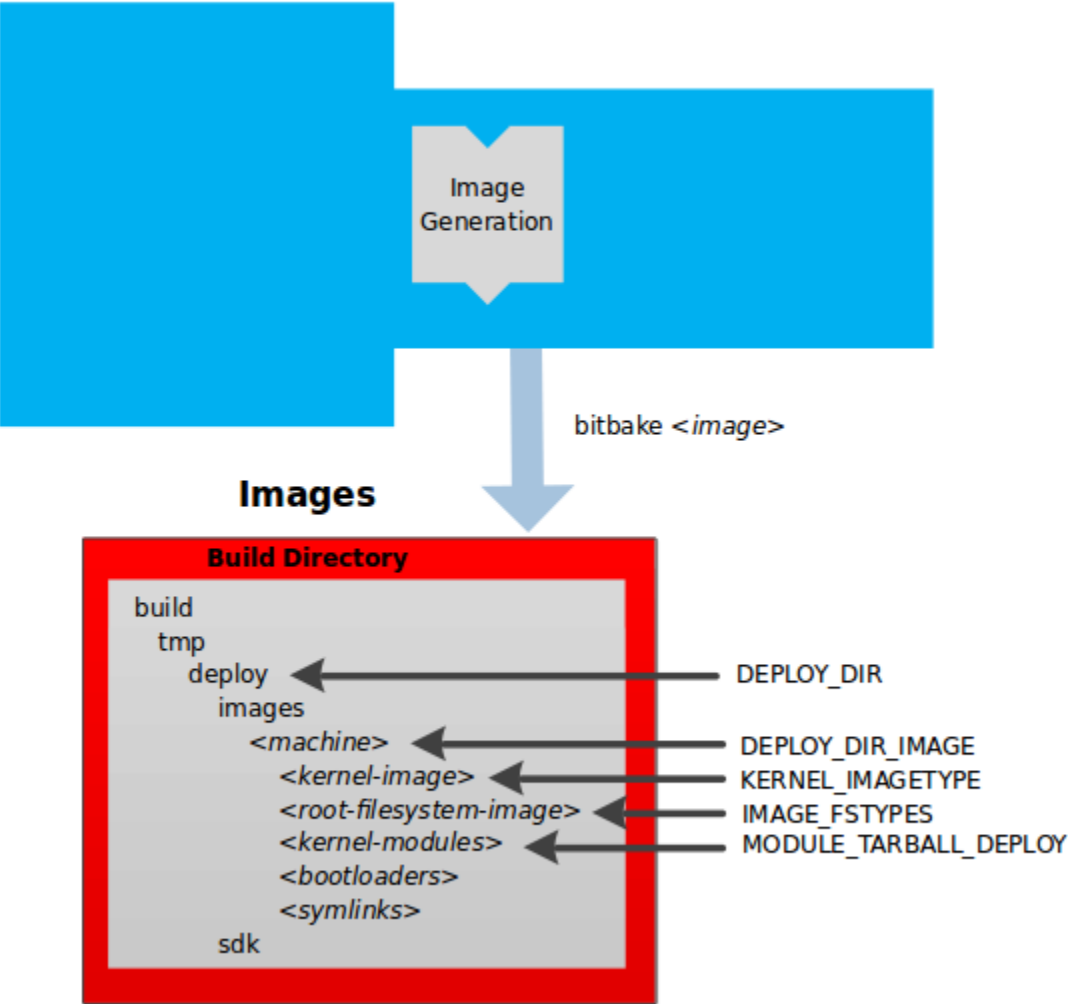
Yocto Project: Bitbake Workflow(5/5)



Yocto Project: Image



BitBake



Yocto Project: devtool

- The devtool command-line tool provides a number of features that help you build, test, and package software. This command is available alongside the *bitbake* command. Additionally, the devtool command is a key part of the extensible SDK.
- Getting help using command “*devtool -h*”.
- devtool allows to:
 - Add a new recipe to workspace layer
 - Modify an existing recipe
 - Build recipe and Image
 - Deploy/remov software on the target machine
 - update existing bbappend file

References

Yocto reference manual:

<https://www.yoctoproject.org/docs/3.1.3/ref-manual/ref-manual.html#normal-recipe-build-tasks>

Yocto mega-manual:

<https://www.yoctoproject.org/docs/3.1.3/mega-manual/mega-manual.html>

Bitbake user manual:

<https://www.yoctoproject.org/docs/3.1.3/bitbake-user-manual/bitbake-user-manual.html>

Distribution Package wiki:

https://wiki.st.com/stm32mpu/wiki/STM32MP1_Distribution_page

Thank you