FTF FREESCALE TECHNOLOGY FORUM 2014

The Yocto Project and Linux® Software Development for i.MX Application Processors

FTF-SDS-F0106

Lauren Post | Yocto i.MX Team Lead

APR.2014

freescale™

External Use

Agenda

- Yocto Project Introduction
- Yocto Project Freescale i.MX Releases
- Yocto Project Setup and Build
- Yocto Project Debugging Tips
- Freescale Manufacturing Tool

Yocto Project Introduction: High Level Overview

- What is the Yocto Project?

  “A Linux Foundation workgroup whose goal is to produce tools and processes that enable the creation of Linux distributions for embedded software independent of architecture”

Freescale Yocto Project

- Linux distributed build system
- Replacement for LTIB
- Higher integration with community
- Flexible release targets

Yocto Project Introduction: Releases

Yocto Project has releases on 6 month cycles approximately every October and April.

Upcoming Releases: Yocto Project 1.6 in April 2014

Past releases:

- Dora Yocto Project 1.5 in October 2013
- Dylan Yocto Project 1.4 in April 2013
- Danny Yocto Project 1.3 in October 2012
- Denzil Yocto Project 1.2 in April 2012

Yocto Project Introduction: Building Blocks

Poky - Open source platform build tool. At the core of Poky is the bitbake task executor together with various types of configuration files. Poky contains common components and toolchain.

Bitbake - Parses metadata, generating a list of tasks from it and then executing them.

Metadata - .bb files are usually referred to as 'recipes‘

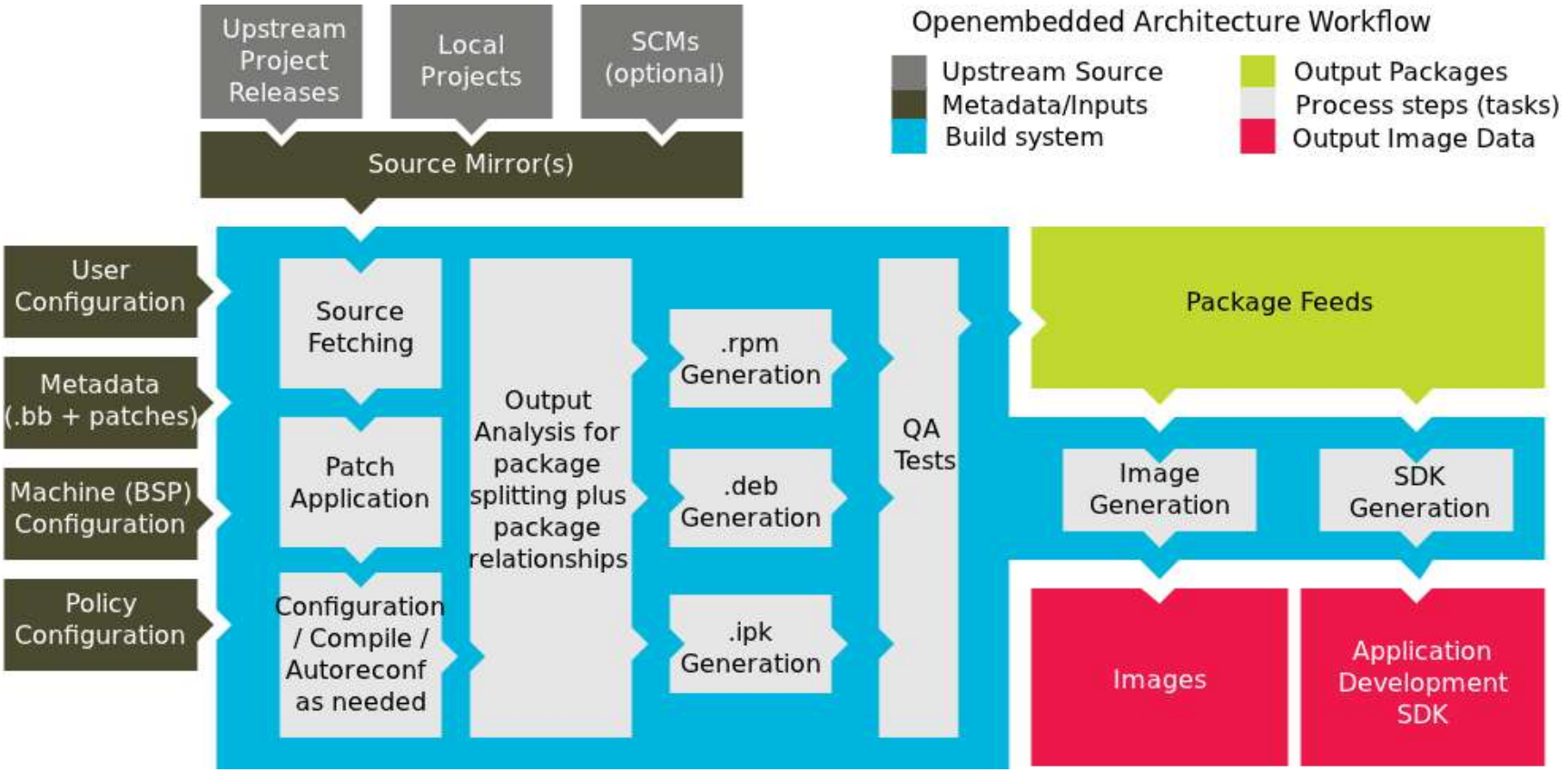
OpenEmbedded-Core - metadata repository

Class - (.bbclass) Contain information which is useful to share between metadata files

Configuration - (.conf) files define various configuration variables which govern what Poky does.

Layers - Sets of common recipes such as meta-fsl-arm

Yocto Project Introduction: Development Environment

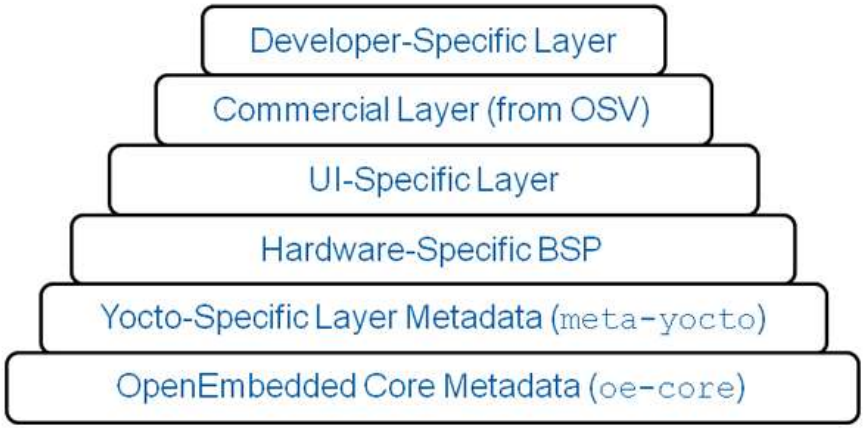


Source: http://www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html

Yocto Project Introduction: Layers of recipes

The OpenEmbedded build system supports organizing Metadata into multiple layers. Layers allow you to isolate different types of customizations from each other. You might find it tempting to keep everything in one layer when working on a single project. However, the more modular you organize your Metadata, the easier it is to cope with future changes.

Source: http://www.yoctoproject.org/docs/1.4/dev-manual/dev-manual.html#understanding-and-creating-layers



Source: https://www.yoctoproject.org/tools-resources/projects/openembedded-core

Let's check out: http://layers.openembedded.org/layerindex/

Yocto Project Introduction: The Freescale i.MX Community BSP

- *"The Freescale Yocto Community BSP is a development community outside of Freescale providing support for i.MX boards on the Yocto Project environment."*

- *There are several developers working on the Freescale Yocto Community BSP, its maintainer is Otavio Salvador from O.S. Systems*

  (*http://ossystems.com.br/* or *https://www.slideshare.net/secret/cmdMnute8kNFUK*)

- *Mail list for the project: https://lists.yoctoproject.org/listinfo/meta-freescale*

Yocto Project Freescale i.MX Releases

Yocto Project Freescale i.MX Releases: History

Release	Description
L3.10.17-1.0.0_ga on dora	Kernel 3.10.17 GA release with p13 Vivante graphics and manufacturing image
L3.10.17-1.0.0_beta on dora	Kernel 3.10.17 with p13 Vivante graphics and manufacturing image with crypto support
L3.10.9-1.0.0_alpha on dora	Kernel 3.10.9 with p12 Vivante graphics and weston/wayland support
L3.5.7-1.0.0_alpha2 on dylan	Kernel 3.5.7 alpha release with hardware floating point using p12 Vivante graphics fixes from 3.0.35-4.1.0 final ltib release
L3.5.7-1.0.0_alpha on dylan	Kernel 3.5.7 alpha release with hardware floating point using p12 Vivante graphics and 2013.04 uboot

Yocto Project Freescale i.MX Releases: Distribution

meta-fsl-bsp-release layer

- Distributes changes on top of community layers
  - meta-fsl-arm
  - meta-fsl-demos
  - poky

Freescale i.MX Yocto Project mirror

- Stores Freescale packages
- Git.freescale.com
- Kernel and u-boot-imx releases
- Moderated download for packages with license restrictions
- After each release the changes are upstreamed into the layers for the next Yocto release to provide

Yocto Project Freescale i.MX Releases: Contents

Contents	Description
recipes-kernel	Kernel and crypto recipes
recipes-bsp	U-boot, imx-test, imx-lib, imx-vpu, imx-kobs, imx-uuc, firmware,
recipes-graphics	gpu-viv-bin-mx6q, gpu-viv-g2d, xorg-driver
recipes-multimedia	libfslcodec, libfslparser, libfslvpuwrap, gstreamer, alsa
recipes-core	busybox, eglibc, udev
recipes-devtools	gcc, mtd
recipes-connectivity	linuxptp, obexftp, openobex, openssl

Yocto Project Freescale i.MX Releases: 2014 Roadmap

- 2014 activities
- 3.10.17-1.0.0 GA
- Kernel upgrades
- Uboot upgrades
- i.MX6SX
- QT5
- Gstreamer 1.0
- IOTG – Internet of Things Gateway
- Yocto Project Releases 1.6 and beyond

Yocto Project Build

Yocto Project Build: Steps 1 and 2

- **Step 1 - Install required host packages**
  - Check references on previous pages for the required host software.

- **Step 2 – Install Repo**

```
$ mkdir ~/bin
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ PATH=$PATH:~/bin
$ chmod a+x ~/bin/repo
```

Yocto Project Build:
Step 3 – Initialize Repo and Start Download

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
$ repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b dora
$ cp fsl-yocto-release-manifest.xml .repo
$ repo sync
```

Try this for repo problems

$ git clone https://gerrit.googlesource.com/git-repo

$ cd git-repo/

$ git checkout v1.12.4

$ cp ../git-repo/repo ~/bin/repo

To download again remove the .repo directory in fsl-community-bsp

Yocto Project Build: Step 3: After BSP Download

- Layers:
  - poky: base build system
  - meta-openembedded: extra packages and features
  - meta-fsl-arm: support for Freescale's processors and board
  - meta-fsl-arm-extra: support for boards using Freescale's
  - meta-fsl-demos: demo images and recipes
  - meta-fsl-bsp-release – Freescale’s release layer of changes

Yocto Project Build: Manifest and Layer Integration

- Manifest describes each layer to download and where to download

- Which branch/revision of each layer
  - Can be specific to commits on each layer (used for Freescale releases)
  - The manifest released by Freescale can be used to reproduce the same images tested by the Freescale test team

Layers are integrated in the **bblayer.conf**. Just because repo downloads does not mean it is part of the build until the layer is in **bblayer.conf**.

Yocto Project Build: Community Manifest

```
<remote fetch="git://git.yoctoproject.org" name="yocto"/>
<remote fetch="git://github.com/Freescale" name="freescale"/>
<remote fetch="git://git.openembedded.org" name="oe"/>

<project remote="yocto" revision="dora" name="poky" path="sources/poky"/>
<project remote="yocto" revision="dora" name="meta-fsl-arm" path="sources/meta-fsl-arm"/>
<project remote="oe" revision="dora" name="meta-openembedded" path="sources/meta-openembedded"/>

<project remote="freescale" revision="dora" name="fsl-community-bsp-base" path="sources/base">
<copyfile dest="README" src="README"/>
<copyfile dest="setup-environment" src="setup-environment"/>
</project>

<project remote="freescale" revision="dora" name="meta-fsl-arm-extra" path="sources/meta-fsl-arm-extra"/>
<project remote="freescale" revision="dora" name="meta-fsl-demos" path="sources/meta-fsl-demos"/>
```

Yocto Project Build: i.MX Release Manifest

```
<remote fetch="git://git.yoctoproject.org" name="yocto" />
<remote fetch="git://github.com/Freescale" name="freescale" />
<remote fetch="git://git.openembedded.org" name="oe" />
<remote fetch="git://git.freescale.com/imx" name="fsl-release" />
<remote fetch="git://github.com/OSSystems" name="OSSystems"/>
<project remote="yocto" revision="75bed4086eb83f1d24c31392f3dd54aa5c3679b1" name="poky"
   path="sources/poky" />
<project remote="yocto" revision="5b3b6618426ad06e4fb6a3a77c4a53dfb47f0556" name="meta-fsl-arm"
   path="sources/meta-fsl-arm" />
 <project remote="oe" revision="513e7ca20ddd0a5c3b649bf292a67c3e0473d3a8" name="meta-
   openembedded" path="sources/meta-openembedded" />
<project remote="freescale" revision="11ac45d22963a3f047a85c96009c36cb6e156d0a" name="meta-fsl-
   arm-extra" path="sources/meta-fsl-arm-extra" />
<project remote="freescale" revision="cd6275042cdd2d87490521f6cbeb65972ed37a66" name="meta-fsl-
   demos" path="sources/meta-fsl-demos" />
<project remote="OSSystems" revision="fc3969f63bda343c38c40a23f746c560c4735f3e" name="meta-
   browser" path="sources/meta-browser" />
<project remote="fsl-release" name="meta-fsl-bsp-release" path="sources/meta-fsl-bsp-release"
   revision="dora_3.10.17-1.0.0_beta">
    <copyfile src="imx/tools/fsl-setup-release.sh" dest="fsl-setup-release.sh" />
</project>
```

Yocto Project Build: Machine configuration files

- Imx6qsabresd.conf Example

- #@TYPE: Machine
- #@NAME: Freescale i.MX6Q SABRE Smart Device
- #@SOC: i.MX6Q
- #@DESCRIPTION: Machine configuration for Freescale i.MX6Q SABRE Smart Device

- require conf/machine/include/imx6sabresd-common.inc

- SOC_FAMILY = "mx6:mx6q"

- KERNEL_DEVICETREE = "imx6q-sabresd.dtb imx6q-sabresd-ldo.dtb “
- KERNEL_DEVICETREE += “imx6q-sabresd-hdcp.dtb"

- UBOOT_CONFIG ??= "sd"
- UBOOT_CONFIG[sd] = "mx6qsabresd_config,sdcard"
- UBOOT_CONFIG[sata] = "mx6qsabresd_sata_config"

Yocto Project Build:
Step 4 - Configure the build environment

- What are the supported boards?

i.MX Reference Boards	
i.MX6Q	imx6qsabresd imx6qsabreauto
i.MX6DualLite	imx6dlsabresd imx6dlsabreauto
i.MX6Solo	imx6solosabresd imx6solosabreauto
i.MX6SoloLite	imx6slevk
i.MX6SX	imx6sx
i.MX53	imx53qsb
i.MX28	Imx28evk

```
$ MACHINE=imx6qsabresd source setup-environment build
```

This is your build environment, you can set a different name to reflect the configuration

Yocto Project Build: License Agreement

- Freescale releases packages on external mirror
- Proprietary packages are packaged as self-extracting binaries with our LAOPT27 EULA
- fsl-eula-unpack class will unpack based on acceptance of EULA
- Microsoft, AACPlus, AC3 not on mirror – only on extranet with moderated downloads (AACPlus is on freescale.com)
- Set-up environment will show EULA and record acceptance in local.conf

- **Do not set up i.MX Yocto Project unless you agree to the License terms**

Yocto Project Build: Step 6 – Build an Image

```
$ bitbake fsl-image-test
```

What are some of the images available?

Targt image	Description
core-image-minimal	Kernel image command prompt
core-image-sato	Kernel build with GUI
fsl-image-test	core-image-base plus Freescale test appls and multimedia
fsl-image-gui	X11 image with QT
fsl-image-x11	Freescale image x11 build with QT
fsl-image-fb	Freescale image Frame buffer
fsl-image-dfb	Freescale image Direct FB
fsl-image-weston	Freescale image Weston compositor using Wayland graphics

Yocto Project Build: Hook in the i.MX release

- Hook in the meta-fsl-bsp-release layer using fsl-setup-release

Graphics Backend	Command
X11	MACHINE=imx6qsabresd source setup-environment build-x11 cd .. source setup fsl-setup-release.sh –b build-x11 bitbake fsl-image-x11
Frame Buffer	MACHINE=imx6qsabresd source setup-environment build-fb cd .. source setup fsl-setup-release.sh –b build-fb **–e fb** bitbake fsl-image-fb
DirectFB	MACHINE=imx6qsabresd source setup-environment build-dfb cd .. source setup fsl-setup-release.sh –b build-dfb **–e dfb** bitbake fsl-image-dfb
Wayland with Weston compositor	MACHINE=imx6qsabresd source setup-environment build-wayland cd .. source setup fsl-setup-release.sh –b build-wayland **–e wayland** bitbake fsl-image-weston

Yocto Project Build: Step 7 - Deploy image on the target

- Build image resides in <build>/tmp/deploy/images/<machine>
- Contents are sdcard, kernel, uboot, rootfs

```
$ cd tmp/deploy/images
$ cat proc/partitions
$ sudo dd if=fsl-image-test-imx6qsabresd.sdcard of=<sd card device> bs=1M && sync
```

Yocto Project Build: Build and Deploy Uboot

```
$ bitbake u-boot-imx –c deploy
```

- Look in machine configuration for uboot options
- Default is SD boot, so no change needed for SD boot
- To change, add line to <build>/conf/local.conf
  - UBOOT_CONFIG=“<boot_config”>

Uboot type	Local.conf
SD boot	UBOOT_CONFIG = “sd”
EIM NOR	UBOOT_CONFIG = “eimnor”
SPI NOR	UBOOT_CONFIG = “spinor”
SATA	UBOOT_CONFIG = “sata”
NAND	UBOOT_CONFIG = “nand”

Yocto Project Build: Build and Deploy Kernel

```
$ bitbake linux-imx
```

- Freescale i.MX kernel recipes point defconfig to location in git tree
- Community kernel recipes use static defconfig
- Custom defconfig can be specified in local.conf

recipes-kernel/linux/linux-imx_3.10.17/mx6/defconfig

Uboot type	Local.conf
Default config	FSL_KERNEL_DEFCONFIG = "imx_v7_defconfig"
Manufacturing image	FSL_KERNEL_DEFCONFIG = "imx_v7_mfg_defconfig"

Yocto Project Build: Adding new device trees

1. Create device tree in kernel source arch/arm/boot/dts
2. Update machine configuration files with new name
   - Note that device tree source is extension dts
   - Binary device tree is extension dtb
     - KERNEL_DEVICETREE = “<new device tree.dtb>“

3. Build kernel
   - bitbake –c compile –f linux-imx
   - bitbake –c deploy –f linux-imx
   - New device tree should be in tmp/deply/images/<machine>

4. Update uboot –
   - Change the fdt_file parameter to the new device tree.dtb

Yocto Project Build: Building QT5 demo

QT5 on dora is provided as demo only.
The following are directions to setup, build and execute for cinematic experience.

- Run qt5 setup from release layer

  source sources/meta-fsl-bsp-release/imx/meta-fsl-qt5/tools/fsl-qt5-setup-demo.sh

- Build X11, FB or Wayland QT5 image – on

  bitbake fsl-image-x11-qt5
  bitbake fsl-image-fb-qt5
  bitbake fsl-image-wayland-qt5

  - Start cinematic experience for each backend
  - Frame buffer

  Qt5_CinematicExperience -platform eglfs -plugin evdevtouch:/dev/input/event0

  - Wayland

  Qt5_CinematicExperience -platform wayland-egl -plugin evdevtouch:/dev/input/event0 –fullscreen

  - X11

  Qt5_CinematicExperience -platform xcb -plugin evdevtouch:/dev/input/event0

Yocto Project Debugging Tips

Yocto Project Debugging Tips: Bitbake commands

bitbake command	Details
bitbake –c cleanall <component>	Cleans downloads and all cache entries for component. Cleans out all entries in working directory!
bitbake –c cleansstate <component>	Cleans cache but does not remove the download entry. Cleans out all entries in working directory!
bitbake –c compile –f <component>	Forces a recompile of a component. Use after making changes in working directory.
bitbake –c configure –f <component>	Forces a reconfigure – useful in kernel if defconfig is changed
bitabke –c fetch <component>	Fetches a component – useful for debugging fetch errors
btibake <component> -v	Verbose output on command line (same as what is in the log)
bitbake <component> -k	Continue past build breaks until a dependency requires stopping
bitbake <component> -c deploy –f	Deploys a component to the rootfs with force – sometimes Yocto thinks a component is already deployed so this forces it
bitbake <component> -g	Lists a dependency tree for component
bitbake <image> -c populate_sdk	Generates an SDK script to reproduce the rootfs and build environement for non-Yocto Project build systems

Yocto Project Debugging Tips: Preferred Providers

- Multiple recipes can provide same component
- Use preferred providers in local or layer.conf.
- Bitbake will fail asking which provider if it is not clear

- Community has kernel and uboot
- i.MX provides kernel and uboot
- Specify which one

- Examples to provide imx as provider
  - PREFERRED_PROVIDER_u-boot_mx6 = "u-boot-imx“
  - PREFERRED_PROVIDER_virtual/kernel_mx6 = “linux-imx"

Yocto Project Debugging Tips: Preferred Versions

- Multiple versions of components in different layers
- Build will pick up git or highest version unless stated in local.conf.
- Used to designate specific versions of components to build
- **Examples:**
  - PREFERRED_VERSION_u-boot-imx_mx6 = "2013-04"
  - PREFERRED_VERSION_linux-imx_mx6 = "3.10.17-1.0.0"
  - PREFERRED_VERSION_imx-lib_mx6 = "3.10.17-1.0.0"
  - PREFERRED_VERSION_imx-test_mx6 = "3.10.17-1.0.0“

Yocto Project Debugging Tips: SoC extensions

- To make target specific changes to:
  - System on chip (SoC) family like i.MX 6 Series, i.MX6Q or i.MX6SL
  - Machine-like i.MX6Q SABRE SD or i.MX 6SLEVK
  - Recipes can limit builds to a specific SoC family, SoC or boards
  - Within recipes, settings or tasks can be limited by SoC family, SoC or boards.

- SOC_FAMILY = "mx6:mx6sl"

- Examples between SoC families
  - PREFERRED_PROVIDER_u-boot_mx6 = "u-boot-imx"
  - PREFERRED_PROVIDER_u-boot_mx5 = "u-boot-fslc“

- Examples between boards, sololite has no vpu
  - DEPENDS_mx6q = "virtual/kernel imx-lib imx-vpu"
  - DEPENDS_mx6sl = "virtual/kernel imx-lib“

Yocto Project Debugging Tips: Recipe Contents

Syntax	Description
LICENSE	State the license of the component
LIC_FILES_CHKSUM	md5sum to verify the license file for component, md5sum COPYING
DESCRIPTION	Description of the component being built
inherit	Inherits classes for additional features
include recipes-<>/<comp>/<comp>.inc	Include .inc files (can exist in other layers) to minimize duplication in recipe. If not found build might continue
Require recipes-<>/<comp>/<comp>.inc	Recipe requires this include. If not found build will stop
DEPENDS = <component>	Forces the component to be built before this component for this recipe is built
S = “$WORKDIR}/<dir>	Change when path name of expanded source package differs from component

Yocto Project Debugging Tips: Recipe Contents

Note any of these can be specific to SoC by adding _<soc>

Syntax	Description
SRC_URI	Specifies git or package download location. Can add patches using file://<patch name>. For git source must specify branch name
SRC_URI[md5sum] = “<md5sumhash>” SRC_URI[shamd5sum] = “<shamd5sumhash>”	Used to verify if the package matches what recipe expects md5sum <package> generates <md5sumhash> sha256sum <package> generates <sha256sum>
SRC_REV	For git source, specifies which commit to use SRC_REV=“$AUTOREV” means tip of branch
COMPATIBLE_MACHINE=“<soc>”	SOC is defined in the machine configuration and limits recipe to specific soc or machine versions.
FILES_${PN} += "/<dir>" FILES_${PN}-dbg += "/<dir>/.debug"	Components must specify if they build binaries to put on rootfs. If missing, build will warn about files built but not installed

Yocto Project Debugging Tips: Makefiles

Bitbake knows how to build most makefiles.

- Sometimes makefiles set user environments that conflict with Yocto Project build environment setting $CC. Use recipes to override conflicts in do_compile or do_configure
- Some components use auto tools, so include this line in recipes for those components.
  - inherit autotools pkgconfig
- Some components require additional flags, use EXTRA_OECONF in recipe to set.
- Check sources/poky/meta/conf/bitbake.conf for common variables to use in recipes.

Yocto Project Debugging Tips: Add packages

Three options:

1. local.conf – Add CORE_IMAGE_EXTRA_INSTALL

   CORE_IMAGE_EXTRA_INSTALL += “ <pkg1> <pkg2> “

2. Add package to the image recipes under IMAGE_INSTALL

3. Create a package group for a set of multiple packages and add package group to the image recipe

Yocto Project Debugging Tips: Debugging

- Yocto has multiple tasks that it runs.

- Each task has a log output in the component working directory
  - log.do_<task> shows output of task
  - run.do_<task> shows content of task

- Example: Kernel working directory
  - <build-dir>/tmp/work/imx6qsabresd-poky-linux-gnueabi/linux-imx/3.10.17-r0/temp
  - Kernel dependent components will go into machine working directories.
  - Non-Kernel dependent components will go the cortexa9 working directories.
  - Sometimes when bitbake is cancelled it continues with python tasks – kill manually otherwise might complain about bitbake task running.

Yocto Project Debugging Tips: Toolchains

- Yocto Project updates the toolchains on each release.
- Can switch to last one by setting GCC_VERSION
- Try setting GCC_VERSION in local.conf to previous version
  - GCC_VERSION = “4.7”
- Hardware Floating Point is Default
  - DEFAULTTUNE_mx6 ?= "cortexa9hf-neon"
- To change to software floating point set in local.conf
  - DEFAULTTUNE_mx6 = "cortexa9-neon“
- Any changes to toolchain require a clean build.
- Only hardware floating point binaries get full test before release.

Yocto Project Debugging Tips: Graphics

- Four graphics backends supported – X11, FB, DFB and Wayland
- Do not mix builds. Set up each in separate build directories.
- Each backend is setup with a different set of DISTRO_FEATURES.
- Each of these backends conflict with each other.
- Many recipes check DISTRO_FEATURES to determine how to build, so it must be set in the local.conf.
- Use the fsl-setup-release.sh to set the local.conf with the correct DISTRO_FEATURES.

Yocto Project Debugging Tips: Reducing Build Time

- In local.conf
- DL_DIR = “<directory path>”
- Multiple builds will share common downloads, reducing disk space and improving speed

- SSTATE_DIR = “<directory path>”
- Multiple builds will share state of previous builds to reduce build time. For example, will not rebuild toolchain.

- Keep track of the tmp/deploy/images contents.
- Multiple builds will generate new images and increase disk space usage.

Yocto Project Debugging Tips: Assistance

Community Email

meta-freescale@yoctoproject.org

https://lists.yoctoproject.org/listinfo/meta-freescale

i.MX Community

https://community.freescale.com/community/imx/content

Click on Yocto Project tab

Freescale Manufacturing Tool

Freescale Manufacturing Tool: Details

- Firmware of MfgTools needs 3 parts from Yocto Project build
  - initramfs.tar.gz – includes components mtd-utils, sfdisk, kobs-ng, tar, mkfs.vfat and mkfs.ext3
  - Uboot –auto detects boots from storage media or usb serial recovery
  - Kernel – Must use the imx_v7_mfg_defconfig to build

  - Freescale i.MX provides an image to build the initramfs.
    - bitbake fsl-image-manufacturing

Installation:

- Copy 3 files to mfgtools\Profiles\Linux\OS Firmware\firmware
- Change mfgtools's ucl.xml to update file name

Freescale Manufacturing Tool: Kernel Configuration

Manufacturing Install defconfig key section

```
CONFIG_USB_GADGET=y
#  CONFIG_USB_ZERO is not set
# CONFIG_USB_AUDIO is not set
# CONFIG_USB_ETH is not set
# CONFIG_USB_G_NCM is not set
# CONFIG_USB_GADGETFS is not set
# CONFIG_USB_FUNCTIONFS is not set
CONFIG_USB_MASS_STORAGE=y
CONFIG_FSL_UTP=y
# CONFIG_USB_G_SERIAL is not set
# CONFIG_USB_MIDI_GADGET is not set
# CONFIG_USB_G_PRINTER is not set
# CONFIG_USB_CDC_COMPOSITE is not set
# CONFIG_USB_G_ACM_MS is not set
# CONFIG_USB_G_MULTI is not set
# CONFIG_USB_G_HID is not set
# CONFIG_USB_G_DBGP is not set
# CONFIG_USB_G_WEBCAM is not set
```

www.Freescale.com