Yocto Project Tools

Bryan Thomas

Field Applications Engineer

October 2018 | AMF-AUT-T3359

NXP | SECURE CONNECTIONS FOR A SMARTER WORLD

Agenda

- i.MX 8 Reference Boards
- Linux Introduction
- Yocto Introduction
- NXP Yocto BSP Release
- Host Machine Setup
- Yocto Setup
- Running a Yocto Build
- Using the Results
- Finding Help

Linux Introduction

Yocto Project: What is Linux?

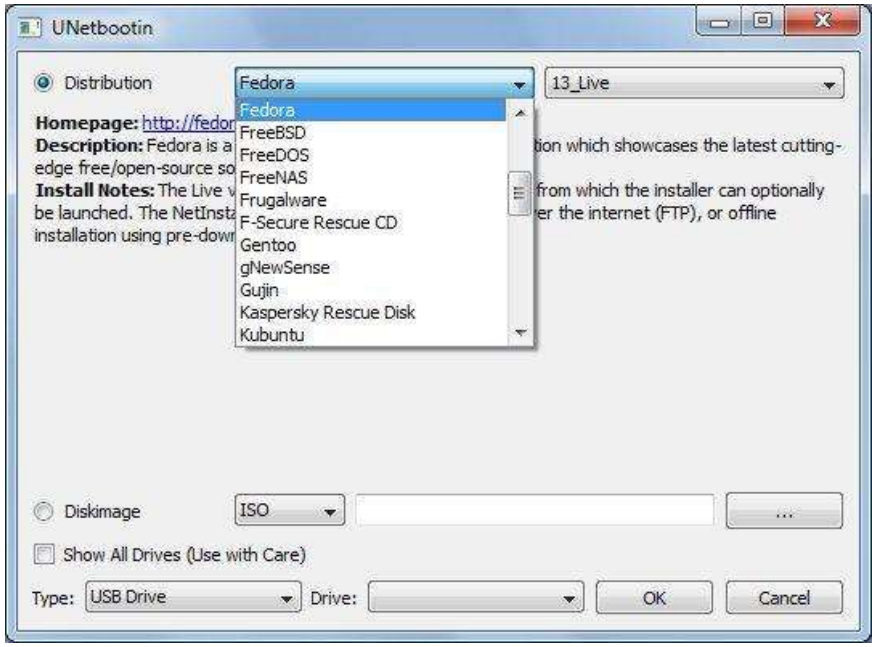
- Linux is an operating system
- GNU/Linux is a collection of programs and the Linux kernel that forms the Operating System
- Many of the components of are licensed under an open source license such as the General Public License (GPL)
- The open source nature of Linux allows you to view the source code of many of the components of the system

Yocto Project: Obtaining and Installing a Linux Distribution

- Select a distribution that others within your company are using
- If you are completely new to Linux then pick a distribution like Debian, Fedora, or Ubuntu
- Distributions are distributed as iso files that can be written to a CD/DVD or USB flash Drive

Yocto Project: Obtaining and Installing a Linux Distribution

- Use Unetbootin to flash an iso to a USB flash drive
- Download at: http://unetbootin.sourceforge.net/



Yocto Project: Obtaining and Installing a Linux Distribution

- Set the computer to boot from USB and insert the flash drive into one of the USB ports
- Some distributions have LIVE preview modes that run out of RAM and allow you to try out the distribution before installing
- Follow the installation guides available on the distribution's homepage to install
- 500 GB is a good size for an install hard disk

Yocto Project: The Terminal

- The terminal will be your new home!
- Cheat sheet (http://overapi.com/linux/)

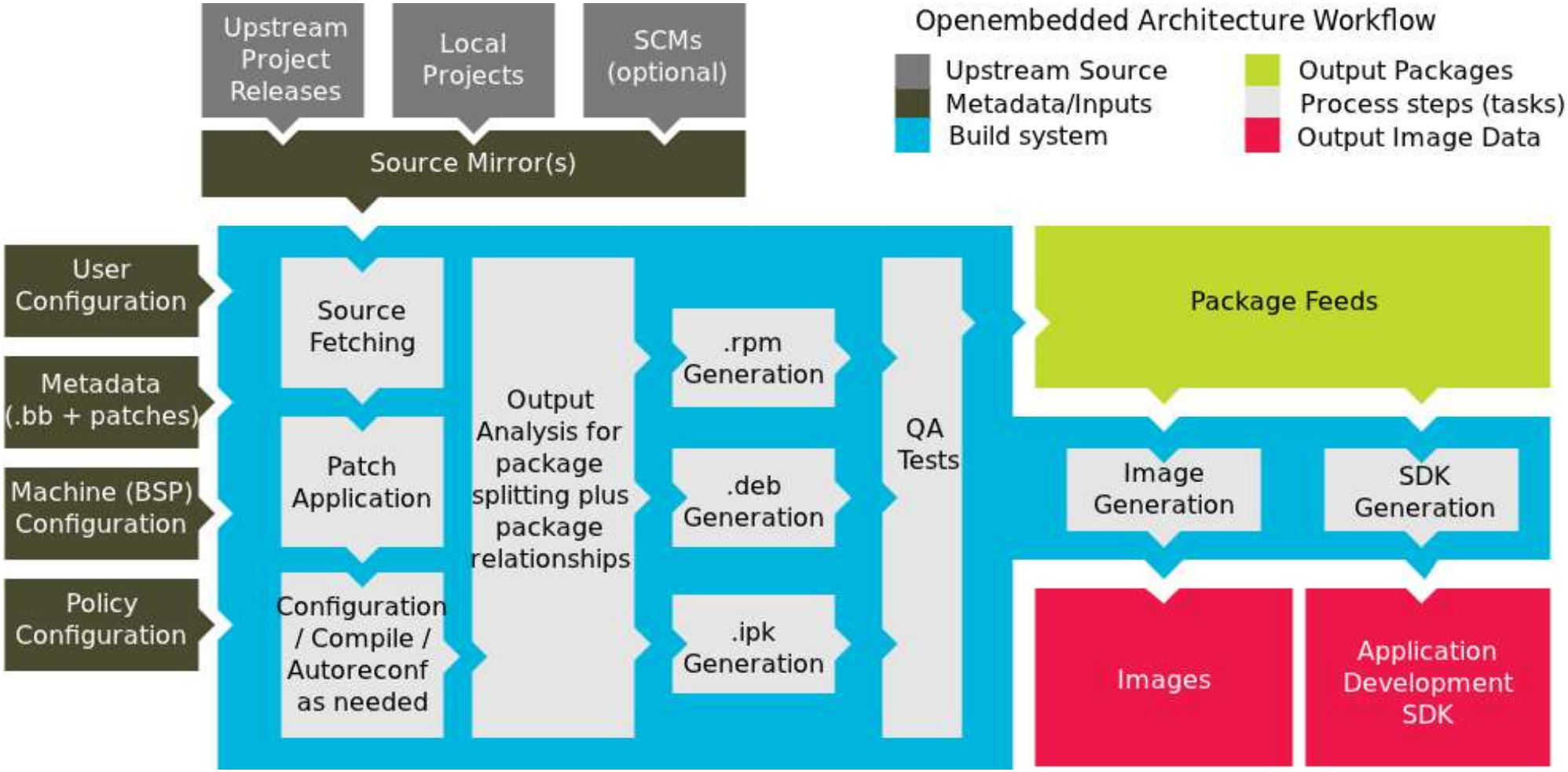
Yocto Introduction

Yocto Project: What is the Yocto Project?

- Open-source collaborative project focused on embedded Linux development
- Currently provides a build system that is referred to as OpenEmbedded build system in the Yocto Project Documentation
- Helps developers create custom Linux-based systems for embedded products

- Source: http://www.yoctoproject.org/docs/1.6.1/yocto-project-qs/yocto-project-qs.html

Yocto Project: Yocto Project Development Environment



Yocto Project: Yocto Project Components

Poky

- Poky is a *reference system* of the Yocto Project - a collection of Yocto Project tools and metadata that serves as a set of working examples. To use the Yocto Project tools, you can download Poky and use it to bootstrap your own distribution.
- Poky is the platform-independent, cross-compiling integration layer that utilizes OpenEmbedded Core. It provides the mechanism to build and combine thousands of distributed open source projects together to form a fully customizable, complete, coherent Linux software stack.
- Poky's objective is to provide all the features and functionalities an embedded developer needs from one solution.
  - Source: https://www.yoctoproject.org/tools-resources/projects/poky

Yocto Project: Yocto Project Components

- BitBake is a build engine that follows recipes in a specific format in order to perform sets of tasks. BitBake is a core component of the Yocto Project.

Yocto Project : Yocto Project Documentation

- The Yocto Project documentation is available on the Yocto Project website: (https://www.yoctoproject.org/documentation)

- Recommended Reading:

1. Yocto Project Quick Start (Link)
2. BitBake User Manual (Link)
3. Yocto Project Reference Manual (Link)

NXP Yocto Release

Yocto Project : NXP Yocto BSP Release

The NXP Yocto BSP Release is an extension of OpenEmbedded and Poky that supports NXP reference boards

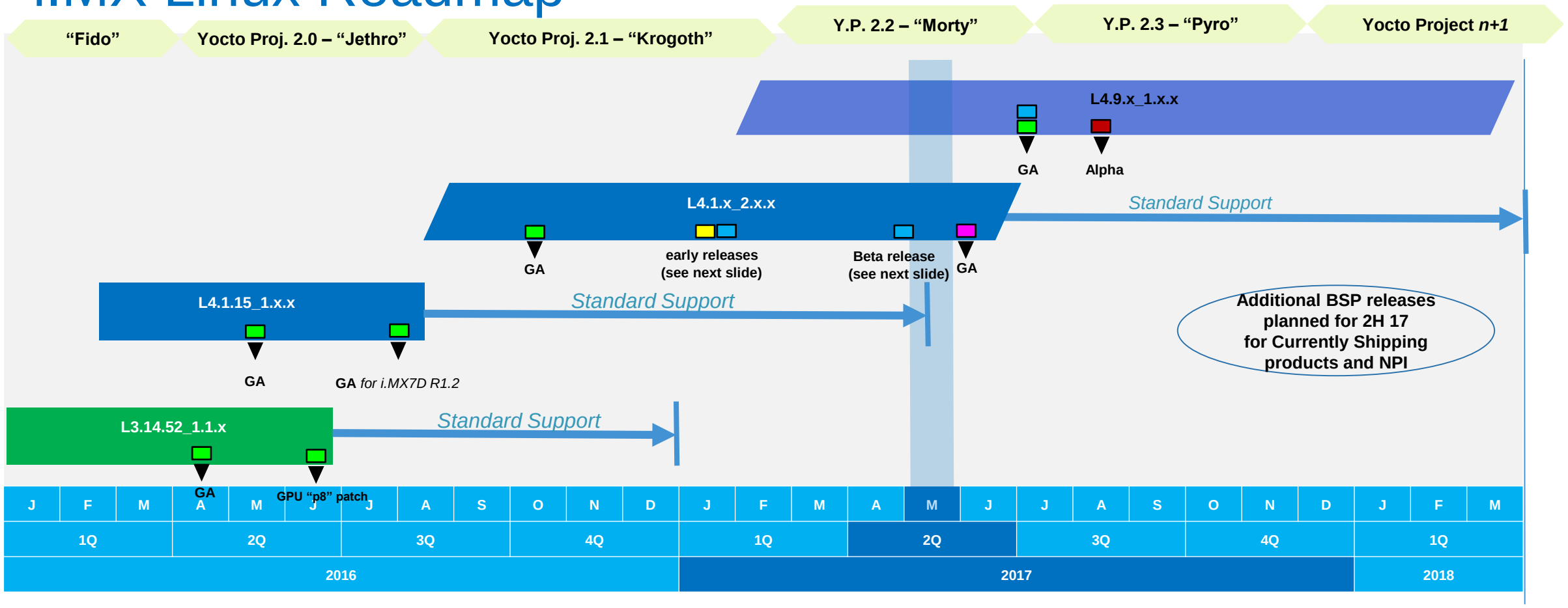
- NXP i.MX6Q SABRE Smart Device
- NXP i.MX6Q SABRE Auto
- NXP i.MX6DL SABRE Smart Device
- NXP i.MX6DL SABRE Auto
- NXP i.MX6SOLO SABRE Smart Device
- NXP i.MX6SOLO SABRE Auto
- NXP i.MX6 Solo Lite EVK
- NXP i.MX8 Family

Yocto Project: Obtaining the NXP Yocto BSP Release

The NXP Yocto BSP Release is available from our website under Software & Tools Section (Link)

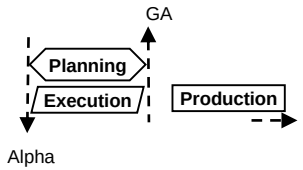
- Contains pre-built images for NXP reference boards
- Contains a README file that explains the procedure to download the BSP using REPO
- Currently based on Yocto version 2.1 (Krogoth)
- Next release based on Yocto version 2.2 (Morty)

i.MX Linux Roadmap



Consolidated GA – Support for 1 year
Alpha, Beta & Single-GA – Availability until next release

Release Schedule



Linux Kernel

- 4.9.x
- 4.1.x
- 3.14.x

Supported Platforms

- i.MX8 (x64)
- Shipping (currently shipping i.MX6 and i.MX7 devices)
- i.MX 6SLL
- i.MX 7ULP

Host Machine Setup

Yocto Project: A Suitable Host Machine

- Depending on the number of processors and cores, the amount of RAM, the speed of your Internet connection and other factors, the build process could take several hours the first time you run it. Subsequent builds run much faster since parts of the build are cached.
- Multiple build directories can consume large amounts of hard drive space
- Virtual Machines are not recommended for daily development

Yocto Project: Supported Linux Distributions

Each Yocto release adds additional supported distributions. Other distributions can also work but are not specifically tested by the Yocto Project. These are supported by Rocko:

- Ubuntu 14.04 (LTS)
- Ubuntu 14.10
- Ubuntu 15.04
- Ubuntu 15.10
- Ubuntu 16.04
- Fedora release 22
- Fedora release 23
- Fedora release 24
- CentOS release 7.x
- Debian GNU/Linux 8.x (Jessie)
- Debian GNU/Linux 9.x (Stretch)
- openSUSE 13.2
- openSUSE 42.1
- Git 1.8.3.1 or greater, tar 1.27 or greater, Python 3.4.0 or greater

Yocto Project: Required Host Packages

The list of packages you need on the host development system can be large when covering all build scenarios using the Yocto Project

Debian based distributions:

Essential packages

```
$ sduo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
    build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \
    xz-utils debianutils iputils-ping
```

Graphical extras (host)

```
$ sudo apt-get install libsdl1.2-dev xterm
```

Documentation packages

```
$ sudo apt-get install make xsltproc docbook-utils fop dblatex xmlto
```

OpenEmbedded Self-Test:

```
$ sudo apt-get install python-git
```

Yocto Project: Required Host Packages

Fedora based distributions:

Essential packages

```
$ sudo dnf install gawk make wget tar bzip2 gzip python3 unzip perl patch \
     diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath \
     ccache perl-Data-Dumper perl-Text-ParseWords perl-Thread-Queue perl-bignum socat \
     python3-pexpect findutils which file cpio python python3-pip xz
```

Graphical extras (host)

```
$ sudo dnf install SDL-devel xterm
```

Documentation packages

```
$ sudo dnf install make docbook-style-dsssl docbook-style-xsl docbook-dtds docbook-utils fop
libxslt dblatex xmlto
```

OpenEmbedded Self-Test

```
$ sudo yum install python3-GitPython
```

Yocto Project: Repo

What is Repo?

- Repo is a tool that Google built on top of Git
- Repo helps manage many Git repositories
- Repo is not meant to replace Git
- The repo command is an executable Python script that you can put anywhere in your path

- Repo repository and cheat sheet: Link

Yocto Project: Repo

Installing Repo

1. Add a bin directory to your home directory:

```
$ mkdir ~/bin
```

2. Download the script file to the bin directory:

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

3. Add execute permissions:

```
$ chmod a+x ~/bin/repo
```

Source (Link)

Yocto Project: Repo

Adding ~/bin directory to your path
* Debian with BASH, other distributions will vary slightly
Temporary:

```
$ export PATH=/home/YOURUSERNAME/bin:$PATH
$ source .bashrc
```

Permanent:

```
      $ vim .profile
+     if [ -d "$HOME/bin" ] ; then
+     PATH="$HOME/bin:$PATH"
+     fi
* NOTE: most distros include ~/bin in your default path, the
directory just doesn’t exist!
```

Yocto Setup

Yocto Project: Repo Initialization

- We will use repo to pull our Yocto sources from multiple Git trees based on the manifest for the particular branch we are using. In this case we will use the current GA release based on the previous Yocto Project release
- Git.freescale.com (Link)

```
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-rocko -m
  imx-4.9.88-2.2.0-8qxp_beta2.xml

-u is the manifest url of our new code repo at codeaurora
-m is the manifest of the yocto build that you will be using
-b is branch or revision in this case Rocko 2.4
```

- If we look in the directory we only see that a hidden directory has been created called .repo. This directory contains information that tells repo what Git repositories and commits to pull into our directory. Repo doesn't pull the source until we run the sync command.

Yocto Project: Repo Initialization

```
 b35938@ontario ~/projects/fsl/yocto $ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.14.28-1.0.1_patch
Get https://gerrit.googlesource.com/git-repo/clone.bundle
Get git://git.freescale.com/imx/fsl-arm-yocto-bsp.git
remote: Counting objects: 171, done.
remote: Compressing objects: 100% (170/170), done.
remote: Total 171 (delta 59), reused 0 (delta 0)
Receiving objects: 100% (171/171), 23.81 KiB | 0 bytes/s, done.
Resolving deltas: 100% (59/59), done.
From git://git.freescale.com/imx/fsl-arm-yocto-bsp
 * [new branch]      imx-3.10.17-1.0.0_ga -> origin/imx-3.10.17-1.0.0_ga
 * [new branch]      imx-3.10.17-1.0.1_ga -> origin/imx-3.10.17-1.0.1_ga
 * [new branch]      imx-3.10.17-1.0.2_ga -> origin/imx-3.10.17-1.0.2_ga
 * [new branch]      imx-3.10.53-1.1.0_ga -> origin/imx-3.10.53-1.1.0_ga
 * [new branch]      imx-3.10.53-1.1.1_patch -> origin/imx-3.10.53-1.1.1_patch
 * [new branch]      imx-3.10.53-1.1.2_patch -> origin/imx-3.10.53-1.1.2_patch
 * [new branch]      imx-3.14.28-1.0.0_ga -> origin/imx-3.14.28-1.0.0_ga
 * [new branch]      imx-3.14.28-1.0.1_patch -> origin/imx-3.14.28-1.0.1_patch
 * [new branch]      imx-3.14.28-7D_alpha -> origin/imx-3.14.28-7D_alpha
 * [new branch]      imx-3.14.38-6QP_beta -> origin/imx-3.14.38-6QP_beta
 * [new branch]      imx-3.14.38-6QP_ga -> origin/imx-3.14.38-6QP_ga
 * [new branch]      imx-3.14.38-6UL7D_beta -> origin/imx-3.14.38-6UL7D_beta
 * [new branch]      imx-3.14.38-6UL_ga -> origin/imx-3.14.38-6UL_ga

Your identity is: Bryan Thomas <bryan@freescale.com>
If you want to change this, please re-run 'repo init' with --config-name
```

Yocto Project: Repo Initialization

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>

  <default sync-j="2"/>

  <remote fetch="git://git.freescale.com/imx" name="fsl-release" />
  <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://github.com/OSSystems" name="OSSystems"/>
  <remote fetch="git://github.com/meta-qt5"  name="QT5"/>

  <project remote="yocto" revision="f20e4c0cf6ddb29a1aad6e7b095e1472e81d330c" name="poky" path="sources/poky"/>
  <project remote="yocto" revision="b74e5e690d8d4e149ea9de3f0fcca37bad93935f" name="meta-fsl-arm" path="sources/meta-fsl-arm"/>

  <project remote="oe" revision="7bbacd0023fa1111da94ba0b2aafd7d872301ffe" name="meta-openembedded" path="sources/meta-openembedded"/>

  <project remote="freescale" revision="89c605386ec81d64b38562acbc66942964fad971" 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="b32528c10caac5e85f2b5efe0e5b95322dd68ace" name="meta-fsl-arm-extra" path="sources/meta-fsl-arm-extra"/>
  <project remote="freescale" revision="48cb0bcdd226d2e7eee1fdc222713e1dff93342c" name="meta-fsl-demos" path="sources/meta-fsl-demos"/>

  <project remote="OSSystems" revision="63963cc56c8d0291779693e62b66cb16e5c86883" name="meta-browser" path="sources/meta-browser" />
  <project remote="QT5" revision="41c5daa84af4466bfc9aa61f6f772c68470a628b" name="meta-qt5" path="sources/meta-qt5" />

  <project remote="fsl-release" name="meta-fsl-bsp-release" path="sources/meta-fsl-bsp-release" revision="dizzy_3.14.28-1.0.1_patch" >
     <copyfile src="imx/tools/fsl-setup-release.sh" dest="fsl-setup-release.sh"/>
  </project>

</manifest>
```

Yocto Project: Repo Sync

We run repo sync and this goes to each git repository that is defined in the manifest.xml file and clones the project that is specified.

```
user@machine:~/projects/fsl/yocto$ repo sync
Fetching project meta-fsl-bsp-release
Fetching project fsl-community-bsp-base
remote: Counting objects: 194, done.
remote: Total 194 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (194/194), 34.57 KiB | 0 bytes/s, done.
Resolving deltas: 100% (114/114), done.
From git://github.com/Freescale/fsl-community-bsp-base
 * [new branch]      daisy      -> freescale/daisy
*** BRANCHES AND TAGS OMMITED TO SAVE SPACE!!!!  ***
 * [new tag]         1.6        -> 1.6
Fetching projects:  12% (1/8)  Fetching project poky
Fetching projects:  25% (2/8)  Fetching project meta-browser
..
Fetching projects: 100% (9/9), done.
```

Some output omitted for clarity

Yocto Project: Repo Sync

After the sync command completes the source directory will contain a setup script and a sources directory.

```
├── fsl-setup-release.sh
├── README
├── setup-environment
└── sources

1 directory, 3 files
```

Yocto Project: Sourcing a Build

- Now you need to pick a machine type that you want to set up to build. These machines are available in the specific <layer>/conf/machine/*.conf files
- You can find available machines with the following command in your yocto top directory

```
$ ls sources/meta-fsl-*/conf/machine/*.conf
sources/meta-fsl-arm/conf/machine/imx23evk.conf
sources/meta-fsl-arm/conf/machine/imx28evk.conf
sources/meta-fsl-arm/conf/machine/imx31pdk.conf
sources/meta-fsl-arm/conf/machine/imx35pdk.conf
sources/meta-fsl-arm/conf/machine/imx51evk.conf
sources/meta-fsl-arm/conf/machine/imx53ard.conf
sources/meta-fsl-arm/conf/machine/imx53qsb.conf
sources/meta-fsl-arm/conf/machine/imx6dlsabreauto.conf
sources/meta-fsl-arm/conf/machine/imx6dlsabresd.conf
sources/meta-fsl-arm/conf/machine/imx6qsabreauto.conf
sources/meta-fsl-arm/conf/machine/imx6qsabresd.conf
sources/meta-fsl-arm/conf/machine/imx6slevk.conf
sources/meta-fsl-arm/conf/machine/imx6solosabreauto.conf
sources/meta-fsl-arm/conf/machine/imx6solosabresd.conf
sources/meta-fsl-arm/conf/machine/twr-vf65gs10.conf
```

New Machines!

imx8qxpmek

imx8qmmek

Yocto Project: Sourcing a Build

- The machine name tells setup script which board you will be using for development

```
$ DISTRO=<distro-name> MACHINE=<machine-name> source
fsl-setup-release.sh -b <build-directory>  -e
<backend fb, dfb, wayland, x11>
```

- Example using NXP i.MX8QXP with xwayland

```
$ DISTRO=fsl-imx-xwayland MACHINE=imx8qxpmek source \
fsl-setup-release.sh -b build-xwayland
```

Yocto Project: Sourcing a Build

```
user@machine:~/projects/fsl/yocto$ MACHINE=imx6qsabreauto DISTRO=fsl-imx-x11 source ./fsl-setup-release.sh -b bld-x11
 Build directory is  build-imx6qsabreauto
 Using FB backend with FB DIST_FEATURES to override poky X11 DIST FEATURES
Configuring for imx6qsabreauto
*** EULA is Displayed ***
EULA has been accepted.
Welcome to NXP Community BSP
The Yocto Project has extensive documentation about OE including a
reference manual which can be found at:
    http://yoctoproject.org/documentation
For more information about OpenEmbedded see their website:
    http://www.openembedded.org/
You can now run 'bitbake <target>'
Common targets are:
    core-image-minimal
    meta-toolchain
    meta-toolchain-sdk
    adt-installer
    meta-ide-support
Your build environemnt has been configured with:

    MACHINE=imx6qsabreauto
    SDKMACHINE=i686
    DISTRO=poky
    EULA=1
```

Yocto Project: Sourcing a Build

The setup script will create the conf directory and copy some files that are needed to create the build. When the script is completed you will be placed into the build directory that you specified. We will use bitbake to spawn off the build from inside this directory

```
user@machine:~/projects/fsl/yocto/build-imx6qsabreauto$ tree

└── conf
    ├── bblayers.conf
    ├── bblayers.conf.org
    ├── local.conf
    ├── local.conf.org
    └── local.conf.sample
```

Yocto Project: Optimizing Host Machine

<build>/conf/local.conf has a few options that you can change in order to speed up building on your host machine

```
MACHINE ??= 'imx6qsabreauto'
DISTRO ?= 'poky'
PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES = "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K"
CONF_VERSION = "1"

BB_NUMBER_THREADS = ‘8’          <- increase to reflect the number of cores
PARALLEL_MAKE = '-j 8’   <- increase to reflect the number of cores

DL_DIR ?= "${BSPDIR}/downloads/”  <- make a common directory if you have several builds
SSTATE_DIR ?= "/share/sstate-cache“ <- make a common directory to speed builds

ACCEPT_FSL_EULA = ””
```

Running a Yocto Build

Yocto Project: Using BitBake

Now we use bitbake to spawn off the build

```
build-imx6qsabreauto$ bitbake <image-name>
```

<image-name> are images that are provided from the release. We can find them with find . –name fsl-image* from the yocto top directory

```
b35938@b35938-13:~/projects/fsl/yocto$ find . -name fsl-image*
./sources/meta-fsl-arm/recipes-fsl/images/fsl-image-mfgtool-initramfs.bb
./sources/meta-fsl-demos/recipes-fsl/images/fsl-image-multimedia.bb
./sources/meta-fsl-demos/recipes-fsl/images/fsl-image-machine-test.bb
./sources/meta-fsl-demos/recipes-fsl/images/fsl-image-multimedia-full.bb
./sources/meta-fsl-bsp-release/imx/meta-fsl-demos/recipes-fsl/images/fsl-image-qt5.bb
./sources/meta-fsl-bsp-release/imx/meta-fsl-demos/recipes-fsl/images/fsl-image-gui.bb
```

Yocto Project: Using BitBake

Example command

build-imx6qsabreauto$ bitbake fsl-image-gui

b35938@b35938-13:~/projects/fsl/yocto/build-imx6qsabreauto$ bitbake fsl-image-gui

Parsing recipes: 100% |##| Time: 00:00:03

Parsing of 1605 .bb files complete (0 cached,1605 parsed). 2004 targets, 160 skipped, 0 masked, 0 errors.

NOTE: Resolving any missing task queue dependencies

NOTE: multiple providers are available for runtime libgl-mesa-dev (mesa, mesa-gl)

NOTE: consider defining a PREFERRED_PROVIDER entry to match libgl-mesa-dev

NOTE: multiple providers are available for jpeg (jpeg, libjpeg-turbo)

NOTE: consider defining a PREFERRED_PROVIDER entry to match jpeg

Yocto Project: Using BitBake

Output

```
Build Configuration:
BB_VERSION        = "1.24.0"
BUILD_SYS         = "x86_64-linux"
NATIVELSBSTRING   = "Debian-8.2"
TARGET_SYS        = "arm-poky-linux-gnueabi"
MACHINE           = "imx6qsabreauto"
DISTRO            = "poky"
DISTRO_VERSION    = "1.7"
TUNE_FEATURES     = "arm armv7a vfp neon callconvention-hard cortexa9"
TARGET_FPU        = "vfp-neon"
meta
meta-yocto        = "(nobranch):f20e4c0cf6ddb29a1aad6e7b095e1472e81d330c"
meta-oe
meta-multimedia   = "(nobranch):7bbacd0023fa1111da94ba0b2aafd7d872301ffe"
meta-fsl-arm      = "(nobranch):b74e5e690d8d4e149ea9de3f0fcca37bad93935f"
meta-fsl-arm-extra = "(nobranch):b32528c10caac5e85f2b5efe0e5b95322dd68ace"
meta-fsl-demos    = "(nobranch):48cb0bcdd226d2e7eee1fdc222713e1dff93342c"
meta-fsl-arm
meta-fsl-demos    = "(nobranch):126bd42a7390fe0e0deca937a40fb526dea82c8c"
meta-browser      = "(nobranch):63963cc56c8d0291779693e62b66cb16e5c86883"
meta-gnome
meta-networking
meta-python
meta-ruby
meta-filesystems  = "(nobranch):7bbacd0023fa1111da94ba0b2aafd7d872301ffe"
meta-qt5          = "(nobranch):41c5daa84af4466bfc9aa61f6f772c68470a628b"
meta-fsl-qt5
meta-fsl-bluez    = "(nobranch):126bd42a7390fe0e0deca937a40fb526dea82c8c"
```

Yocto Project: First Build Time

- The first build can take up to several hours depending on the speed of your machine
- Subsequent builds will take much less time because the built packages are cached to improved build times
- Only modified or added recipes will be built and deployed in the subsequent builds

Using the Results

Yocto Project: Location of the Build Output

- The build output is inside the ../tmp/deploy/images directory in the build directory you created:

```
user@machine:~/projects/fsl/yocto/build-
imx6qsabreauto/tmp/deploy/images/imx6qsabreauto$ ls

*** More file are in the directory but here is a sample ***
fsl-image-gui-imx6qsabreauto-20150821000528.rootfs.tar.bz2
fsl-image-gui-imx6qsabreauto-20150821000528.rootfs.sdcard
uImage
u-boot.imx
fsl-image-gui-imx6qsabreauto.sdcard
```

Yocto Project: Flashing SD Card Image

- Inside the images directory there is a .sdcar image file that is a complete SD card image that can be flashed and used on the reference board.
- Use dmesg or lsblk to find the name of your disk (sdX)

```
$ sudo dd if=fsl-image-fb-imx6qsabreauto.sdcard of=/dev/sdX bs=1M
516+0 records in
516+0 records out
541065216 bytes (541 MB) copied, 45.932 s, 11.8 MB/s
```

- Card can now be used to boot the reference platform

Yocto Project: Finding Help

- Community.NXP.com
- Meta-NXP mailing list
- Yocto Project Documentation
- Presenter Contact: Bryan Thomas - bryan.thomas@nxp.com

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com