Freescale Semiconductor
User's Guide

Document Number: IMXLXYOCTOUG
Rev. 0, 12/2015

Freescale Yocto Project User's Guide

Contents

1 Overview

This document describes how to build an image for an i.MX Freescale board by using a Yocto Project build environment. It describes the Freescale release layer and Freescale-specific usage.

The Yocto Project is an open-source collaboration focused on embedded Linux® OS development. For more information regarding Yocto Project, see the Yocto Project page: www.yoctoproject.org/. There are several documents on the Yocto Project home page that describe in detail how to use the system. The basic Yocto Project, without the Freescale release layer, can be used by following the instructions in the *Yocto Project Quick Start* found at www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html.

The FSL Yocto Project Community BSP (found at freescale.github.io) is a development community outside Freescale providing support for i.MX boards in the Yocto Project environment. Freescale i.MX joined the Yocto Project community providing a release based on the Yocto Project framework. Information specific to FSL community BSP use can be found on the community web page. This document is an extension of the community BSP documentation.

Files used to build an image are stored in layers. Layers contain different types of customizations and come from different sources. Some of the files in a layer are called

recipes. Yocto Project recipes contain the mechanism to retrieve source code, build and package a component. The following lists show the layers used in this release.

Freescale release layer

- meta-fsl-bsp-release
    - meta-bsp - updates for meta-fsl-arm, poky, and meta-openembedded layers
    - meta-sdk - updates for meta-fsl-demos and distros

Yocto Project community layers

- meta-fsl-arm: provides support for the base and for Freescale ARM® reference boards.
- meta-fsl-arm-extra: provides support for 3rd party and partner boards.
- meta-fsl-demos: additional items to aid in development and exercise board capabilities.
- meta-fsl-community-base: often renamed to base. Provides base configuration for FSL Community BSP.
- meta-openembedded: Collection of layers for the OE-core universe. See layers.openembedded.org/.
- poky: basic Yocto Project items in Poky. See the Poky README for details.
- meta-browser: provides several browsers.
- meta-qt5: provides Qt5.

References to community layers in this document are for all the layers in Yocto Project except meta-fsl-bsp-release. Freescale i.MX boards are configured in the meta-fsl-bsp and meta-fsl-arm layers. This includes U-Boot, the Linux kernel, and reference board-specific details.

Freescale provides an additional layer called the Freescale BSP Release, named meta-fsl-bsp-release, to integrate a new Freescale release with the FSL Yocto Project Community BSP. The meta-fsl-bsp-release layer aims to release the updated and new Yocto Project recipes and machine configurations for new releases that are not yet available on the existing meta-fsl-arm and meta-fsl-demos layers in the Yocto Project. The contents of the Freescale BSP Release layer are recipes and machine configurations. In many test cases, other layers implement recipes or include files and the Freescale release layer provides updates to the recipes by either appending to a current recipe, or including a component and updating with patches or source locations. Most Freescale release layer recipes are very small because they use what the community has provided and update what is needed for each new package version that is unavailable in the other layers.

The Freescale BSP Release layer also provides image recipes that include all the components needed for a system image to boot, making it easier for the user. Components can be built individually or through an image recipe, which pulls in all the components required in an image into one build process.

Freescale kernel and U-Boot releases are accessed through Freescale public git servers. However, several components are released as packages on the Freescale mirror. The package-based recipes pull files from the Freescale mirror instead of a git location and generate the package needed.

All packages which are released as binary are built with hardware floating point enabled. Next release software floating point packages will not be provided. The package selection floating point configuration is determined by using the DEFAULTTUNE setting. (See the README file in meta-fsl-bsp-release/imx for instructions.)

Release L3.14.52_1.1.0_ga is released for Yocto Project 1.8 (Fido). The same recipes for Yocto Project 1.8 are going to be upstreamed and made available on Yocto Project release 2.0. The Yocto Project release cycle lasts roughly six months.

The recipes and patches in meta-fsl-bsp-release are upstreamed to the community layers. Once that is done for a particular component, the files in meta-fsl-bsp-release are no longer needed and the FSL Yocto Project Community BSP will provide support. The community supports Freescale reference boards, community boards, and third-party boards. A complete list can be found at freescale.github.io/doc/release-notes/1.8/index.html#document-bsp-scope. All board references in this document are related to the Freescale machine configuration files only.

1.1 End user license agreement

During the setup environment process of Freescale Yocto Project Community BSP, the Freescale i.MX End User License Agreement (EULA) is displayed. To continue to use the Freescale Proprietary software, users must agree to the conditions of this license. The agreement to the terms allows the Yocto Project build to untar packages from the Freescale mirror. Read this license agreement carefully during the setup process, because once accepted, all further work in the Freescale Yocto Project environment is tied to this accepted agreement.

1.2 References

This release includes the following references and additional information.

- *i.MX Linux® Release Notes* (IMXLXRN) - Provides the release information.
- *i.MX Linux® User's Guide* (IMXLUG) - Contains the information on installing U-Boot and Linux OS and using i.MX-specific features.
- *Freescale Yocto Project User's Guide* (IMXLXYOCTOUG) - Contains the instructions for setting up and building Linux OS in the Yocto Project.
- *i.MX Linux® Reference Manual* (IMXLXRM) - Contains the information on Linux drivers for i.MX.
- *i.MX 6 Graphics User's Guide* (IMX6GRAPHICUG) - Describes the graphics used.
- *i.MX BSP Porting Guide* (IMXXBSPPG) - Contains the instructions on porting the BSP to a new board.
- *i.MX VPU Application Programming Interface Linux® Reference Manual* (IMXVPUAPI) - Provides the reference information on the VPU API.

The quick start guides contain basic information on the board and setting it up. They are on the Freescale website.

- SABRE Platform Quick Start Guide (IMX6QSDPQSG)
- SABRE Board Quick Start Guide (IMX6QSDBQSG)
- SABRE Automotive Infotainment Quick Start Guide (IMX6SABREINFOQSG)
- i.MX 6SoloLite Evaluation Kit Quick Start Guide (IMX6SLEVKQSG)

Documentation is available online at freescale.com.

- i.MX 6 information is at freescale.com/iMX6series
- i.MX 6 SABRE information is at freescale.com/imxSABRE
- i.MX 6SoloLite EVK information is at freescale.com/6SLEVK
- i.MX 7Dual information is at freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX7D
- i.MX 6UltraLite information is at freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX6UL.

2 Features

Freescale Yocto Project Release layers have the following features:

- Linux kernel recipe
  - The kernel recipe resides in the recipes-kernel folder and integrates a Freescale kernel from the source downloaded from the Freescale git server. This is done automatically by the recipes in the project.
  - L3.14.52_1.1.0_ga is a Linux kernel that Freescale has released only for the Yocto Project.
  - Freescale L3.14.52_1.1.0_ga supports using device trees. Device tree settings are found in the i.MX machine configuration files.
- U-Boot recipe
  - The U-Boot recipe resides in the recipes-bsp folder and integrates a Freescale uboot-imx.git from the source downloaded from the Freescale git server.
  - Certain i.MX boards use different U-Boot versions.

    - Freescale release L3.14.52_1.1.0_ga for the i.MX 6 and i.MX 7 devices uses an updated v2015.04 Freescale version. This version has not been updated for other i.MX Freescale hardware.
    - The Freescale Yocto Project Community BSP uses u-boot-fslc from the mainline, but this is only supported by the U-Boot community and does not work for the L3.14.52 kernel.
    - The Freescale Yocto Project Community BSP updates U-Boot versions frequently, so the information above might change as new U-Boot versions are integrated to meta-fsl-arm layers and updates from Freescale u-boot-imx releases are integrated into the mainline.
- Graphics recipes
    - Graphics recipes reside in recipes-graphics.
    - Graphics recipes integrate the Freescale graphics package release. For the i.MX 6 boards that have a GPU, the imx-gpu-viv recipes package the graphic components for each DISTRO – X11, frame buffer (FB), Direct Frame Buffer (directFB), Wayland backend, and Weston compositor (Weston).
    - Xorg-driver integrates the xserver-xorg.
- i.MX package recipes

  imx-lib, imx-test, and firmware-imx reside in recipes-bsp and pull from the Freescale mirror to build and package into image recipes.

- Multimedia recipes
    - Multimedia recipes reside in recipes-multimedia.
    - Recipes include libfslcodec, libfslparser, libvpuwrap, and gstreamer that pull from the Freescale mirror to build and package into image recipes.
    - Some recipes are provided for codecs that are restricted. Packages for these are not on the Freescale mirror. These packages are available separately. Contact your Freescale Marketing representative to acquire these.
- Core recipes

  Some recipes for rules, such as udev, provide updated i.MX rules to be deployed in the system. These recipes are usually updates of policy recipes and are used for customization only. Releases only provide updates if needed.

- Demo recipes

  Demonstration recipes reside in the meta-sdk directory. This layer contains image recipes and recipes for customization, such as touch calibration, or recipes for demonstration applications.

3 Host Setup

To get the Yocto Project expected behavior in a Linux Host Machine, the packages and utilities described below must be installed. An important consideration is the hard disk space required in the host machine. For example, when building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB for the X11 backend. It is recommended that at least 120 GB is provided, which is enough to compile all backends together.

The recommended minimum Ubuntu version is 14.04 but builds for dizzy works on 12.04 or later. Earlier versions may cause the Yocto Project build setup to fail, because it requires python versions only available starting wtih Ubuntu 12.04. See The Yocto Project reference manual for more information.

3.1 Host packages

A Yocto Project build requires that some packages be installed for the build that are documented under the Yocto Project. You can go to Yocto Project Quick Start and check for the packages that must be installed for your build machine.

Essential Yocto Project host packages are:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
  build-essential chrpath  socat  libsdl1.2-dev
```

i.MX layers host packages for a Ubuntu 12.04 or 14.04 host setup are:

```
$ sudo apt-get install libsdl1.2-dev xterm  sed cvs subversion coreutils texi2html \
docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils \
libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc
```

i.MX layers host packages for a Ubuntu 12.04 host setup only are:

```
$ sudo apt-get install uboot-mkimage
```

i.MX layers host packages for a Ubuntu 14.04 host setup only are:

```
$ sudo apt-get install u-boot-tools
```

The configuration tool uses the default version of grep that is on your build machine. If there is a different version of grep in your path, it may cause builds to fail. One workaround is to rename the special version to something not containing "grep".

3.2 Setting up the repo utility

Repo is a tool built on top of Git that makes it easier to manage projects that contain multiple repositories, which do not need to be on the same server. Repo complements very well the layered nature of the Yocto Project, making it easier for users to add their own layers to the BSP.

To install the “repo” utility, perform these steps:

1. Create a bin folder in the home directory.

   ```
   $ mkdir ~/bin (this step may not be needed if the bin folder already exists)
   $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo  > ~/bin/repo
   $ chmod a+x ~/bin/repo
   ```

2. Add the following line to the `.bashrc` file to ensure that the `~/bin` folder is in your PATH variable.

   ```
   export PATH=~/bin:$PATH
   ```

4 Yocto Project Setup

First make sure that git is setup properly with the commands below.

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --list
```

The Freescale Yocto Project BSP Release directory contains a "sources" directory, which contains the recipes used to build, one or more build directories, and a set of scripts used to set up the environment.

The recipes used to build the project come from both the community and Freescale. The Yocto Project layers are downloaded to the `sources` directory. This sets up the recipes that are used to build the project.

The following example shows how to download the Freescale Yocto Project Community BSP recipe layers. For this example, a directory called `fsl-release-bsp` is created for the project. Any name can be used instead of this.

```
$ mkdir fsl-release-bsp
$ cd fsl-release-bsp
$ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.14.52-1.1.0_ga
```

```
$ repo sync
```

When this process is completed, the source code is checked out into the directory `fsl-release-bsp/sources`.

You can perform repo synchronization, with the command `repo sync`, periodically to update to the latest code.

If errors occur during repo initialization, try deleting the .repo directory and running the repo initialization command again.

5 Image Build

This section provides the detailed information along with the process for building an image.

5.1 Build configurations

Freescale has provided a script, `fsl-setup-release.sh`, that simplifies the setup for Freescale machines. To use the script, the name of the specific machine to be built for needs to be specified as well as the graphical backend desired. The script sets up a directory and the configuration files for the specified machine and backend.

In the `meta-fsl-bsp-release` layer, Freescale provides new or updated machine configurations that overlay the `meta-fsl-arm` machine configurations. These files are copied into the `meta-fsl-arm/conf/machine` directory by the `fsl-setup-release.sh script`. The following are Freescale machine configuration files that can be selected. Check either the release notes or the machine directory for the latest additions.

- imx6qpsabreauto
- imx6qpsabresd
- imx6ulevk
- imx6dlsabreauto
- imx6dlsabresd
- imx6qsabreauto
- imx6qsabresd
- imx6slevk
- imx6solosabreauto
- imx6solosabresd
- imx6sxsabresd
- imx6sxsabreauto
- imx7dsabresd

Each build must configure a distro configuration. The distro sets up the build environment. Each graphical backend Frame Buffer, Wayland, Xwayland and X11 each have a distro configuration. If no DISTRO file is specified then the x11 distro is setup as default. In past releases the fsl-setup-release script had a -e parameter for setting the backend. If this is used then the corresponding distro file is specified. Distro configurations are saved in the local.conf in the DISTRO setting and will show when the bitbake is running. In past releases we used the poky distro and customized versions and providers in our layer.conf but a custom distro is a better solution. If the poky distro is used, the build configuration might pick up community kernel, uboot and gstreamer solutions and not the supported i.MX components.

Here are the list of DISTRO configurations. Note that DirectFB is no longer supported.

- fsl-imx-x11 - Only X11 graphics
- fsl-imx-wayland - Wayland weston graphics
- fsl-imx-xwayland - Wayland graphics and X11. X11 applications using EGL not supported
- fsl-imx-fb - Frame Buffer graphics - no X11 or Wayland

Users are welcome to create their own custom distro file based on one of these to customize their environment without having to update the local.conf.

The syntax for the fsl-setup-release script is shown below.

```
$ DISTRO=<distro name> MACHINE=<machine name> source fsl-setup-release.sh -b <build dir>
```

DISTRO=<distro configuration name> is the distro which configures the build environment stored in meta-fsl-bsp-release/imx/meta-sdk/conf/distro.

MACHINE=<machine configuration name> is the machine name which points to the configuration file in conf/machine in meta-fsl-arm and meta-fsl-bsp-release.

-b <build dir> specifies the name of the build directory that the script creates.

When the script is run, it asks the user to accept the EULA. Once the EULA is accepted, the acceptance is logged and the EULA acceptance query is no longer displayed.

After the script runs, the prompt is in the newly created directory, specified with the -b option. A conf is created containing the files bblayers.conf and local.conf.

The <build dir>/conf/bblayers.conf file contains all the metalayers used in the Freescale Yocto Project release.

The local.conf file contains the machine and distro specifications:

```
MACHINE ??= 'imx6qsabresd'
DISTRO ?= 'fsl-imx-x11'
ACCEPT_FSL_EULA = "1"
```

The MACHINE configuration can be changed by editing this file, if necessary.

ACCEPT_FSL_EULA in the local.conf file indicates that you have accepted the conditions of the EULA.

In the meta-fsl-bsp-release layer, consolidated machine configurations (imx_6qdlsolo.conf, imx6ul7d.conf and imx6sx_all.conf) are provided for i.MX 6 and i.MX 7 machines. Freescale uses these to build a common image with all the device trees in one image for testing. Do not use these machines for anything other than testing.

5.2 Choosing a Freescale Yocto project image

The Yocto Project provides some images which are available on different layers. Poky provides some images, meta-fsl-arm and meta-fsl-demos provide others, and additional image recipes are provided in the meta-fsl-bsp-release layer. The following table lists various key images, their contents, and the layers that provide the image recipes.

Table 1. Freescale Yocto project images

Image name	Target	Provided by layer
core-image-minimal	A small image that only allows a device to boot.	poky
core-image-base	A console-only image that fully supports the target device hardware.	poky
core-image-sato	An image with Sato, a mobile environment and visual style for mobile devices. The image supports X11 with a Sato theme and uses Pimlico applications. It contains a terminal, an editor and a file manager.	poky
fsl-image-machine-test	An FSL Community i.MX core image with console environment - no GUI interface	meta-fsl-demos
fsl-image-gui	Builds a Freescale image with a GUI without any Qt content.	meta-fsl-bsp-release/imx/meta-sdk
fsl-image-qt5	Builds an opensource Qt 5 image. These images are only supported for i.MX SoC with hardware graphics. They are not supported on the i.MX 6UltraLite and i.MX 7Dual.	meta-fsl-bsp-release/imx/meta-sdk

5.3 Building an image

The Yocto Project build uses the bitbake command. For example, `bitbake <component>` builds the named component. Each component build has multiple tasks, such as fetching, configuration, compilation, packaging, and deploying to the target rootfs. The bitbake image build gathers all the components required by the image and build in order of the dependency per task. The first build is the toolchain along with the tools required for the components to build.

The following command is an example on how to build an image:

```
$ bitbake fsl-image-gui
```

5.4 Bitbake options

The bitbake command used to build an image is `bitbake <image name>`. Additional parameters can be used for specific activities described below. Bitbake provides various useful options for developing a single component. To run with a bitbake parameter, the command looks like this:

```
bitbake  <parameter> <component>
```

`<component>` is a desired build package.

The following table provides some bitbake options.

Table 2. Bitbake options

Bitbake paramater	Description
-c fetch	Fetches if the downloads state is not marked as done.
-c cleanall	Cleans the entire component build directory. All the changes in the build directory is lost. The rootfs and state of the component are also cleared. The component is also removed from the download directory.
-c deploy	Deploys an image or component to the rootfs.
-k	Continues building components even if a build break occurs.
-c compile -f	It is not recommended that the source code under the tmp directory is changed directly, but if it is, the Yocto Project might not rebuild it unless this option is used. Use this option to force a recompile after the image is deployed.
-g	Lists a dependency tree for an image or component.
-DDD	Turns on debug 3 levels deep. Each D adds another level of debug.

5.5 U-Boot configuration

U-Boot configurations are defined in the main machine configuration file. The configuration is specified by using the UBOOT_CONFIG settings. This requires setting UBOOT_CONFIG in `local.conf`. Otherwise, the U-Boot build uses SD boot by default.

These can be built separately by using the following commands (change MACHINE to the correct target).

U-Boot type	Build setup	Build command
U-Boot EIM-NOR	`$ echo "UBOOT_CONFIG = \"eimnor\"" >> conf/local.conf`	`$ MACHINE=imx6dlsabreauto bitbake -c deploy u-boot-imx`
U-Boot SPI-NOR	`$ echo "UBOOT_CONFIG = \"spinor\"" >> conf/local.conf`	`$ MACHINE=imx6qsabreauto bitbake -c deploy u-boot-imx`
U-Boot NAND	`$ echo "UBOOT_CONFIG = \"nand\"" >> conf/local.conf`	`$ MACHINE=imx6solosabreauto bitbake -c deploy u-boot-imx`
U-Boot SATA	`$ echo "UBOOT_CONFIG = \"sata\"" >> conf/local.conf`	`$ MACHINE=imx6qsabresd bitbake -c deploy u-boot-imx`
U-Boot ARM® Cortex®-M4 core	`$ echo "UBOOT_CONFIG = \"m4fastup\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabresd bitbake -c deploy u-boot-imx`
U-Boot QSPI1	`$ echo "UBOOT_CONFIG = \"qspi1\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabreauto bitbake -c deploy u-boot-imx`
U-Boot QSPI2	`$ echo "UBOOT_CONFIG = \"qspi2\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabresd bitbake -c deploy u-boot-imx`
U-Boot EMMC	`$ echo "UBOOT_CONFIG = \"emmc\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabresd bitbake -c deploy u-boot-imx`
U-Boot m4fastup	`$ echo "UBOOT_CONFIG = \"m4fastup\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabresd bitbake -c deploy u-boot-imx`
U-Boot epdc	`$ echo "UBOOT_CONFIG = \"epdc\"" >> conf/local.conf`	`$ MACHINE=imx6sxsabresd bitbake -c deploy u-boot-imx`

5.6 Build scenarios

The following are build setup scenarios for various configurations.

Set up the manifest and populate the Yocto Project layer sources with these commands:

```
$ mkdir fsl-release-bsp
$ cd fsl-release-bsp
$ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.14.52-1.1.0_ga
$ repo sync
```

The sections following give some specific examples. Replace the machine names and the backends specified to customize the commands.

5.6.1 X-11 image on i.MX 6Quad SABRE-SD

```
$ DISTRO=fsl-imx-x11 MACHINE=imx6qsabresd source fsl-setup-release.sh -b build-x11
$ bitbake fsl-image-gui
```

This builds an X11 image without Qt 5. To build with Qt 5, build fsl-image-qt5.

5.6.2 Frame Buffer image on i.MX 6QuadPlus SABRE-AI

```
$ DISTRO=fsl-imx-fb MACHINE=imx6qpsabreauto source fsl-setup-release.sh –b build-fb
$ bitbake fsl-image-qt5
```

This builds Qt 5 on a frame buffer backend. To build without Qt 5, use image recipe fsl-image-gui.

5.6.3 Xwayland image on i.MX 6SoloX SABRE-SD

```
$ DISTRO=fsl-imx-xwayland MACHINE=imx6sxsabresd source fsl-setup-release.sh –b build-xwayland
$ bitbake fsl-image-gui
```

5.6.4 Wayland image on i.MX 6SoloX SABRE-SD

```
$ DISTRO=fsl-imx-wayland MACHINE=imx6sxsabresd source fsl-setup-release.sh -b build-wayland
$ bitbake fsl-image-qt5
```

This builds a Qt 5 Weston Wayland image. To build without Qt 5, build fsl-image-gui. To build xwayland use distro fsl-imx-xwayland.

5.6.5 Restarting a build environment

If a new terminal window is opened or the machine is rebooted after a build directory is set up, the setup environment script should be used to set up the environment variables and run a build again. The full `fsl-setup-release.sh` is not needed.

```
$ source setup-environment <build-dir>
```

5.6.6 Chromium Browser on X11, XWayland and Wayland

The Yocto Project community has chromium-imx recipes to enable hardware acceleration for X11, XWayland and Wayland version Chromium Browser for i.MX SoC with GPU hardware. Note that the VPU patches created by the community have problems but community GPU patches work. This section describes how to integrate Chromium into your rootfs and enable hardware accelerated rendering of WebGL. The Chromium browser requires additional layers added in the `fsl-release-setup.sh` script.

In `local.conf`, you can perform the following operations:

- Add Chromium into your image.

  ```
  CORE_IMAGE_EXTRA_INSTALL += "chromium libexif"
  ```
- Add the commercial white list into `local.conf`.

  ```
  LICENSE_FLAGS_WHITELIST="commercial"
  ```

This allows proprietary code to be built into your image. Additional license obligations will need to be met for these additions. Make sure you know what they are and are in compliance.

5.6.7 Qt 5 and QtWebEngine browsers

Qt 5 has both a commercial and an open source license. When building in Yocto Project the open source license is the default. Make sure to understand the differences between these licenses and choose appropriately. Once custom Qt 5 development has started on the open source license it can't be used with the commercial license. Work with a legal representative to understand the differences between these licenses.

There are three Qt 5 browsers available. QtWebEngine browsers can be found in `/usr/share/qt5/examples/webenginewidgets/browser`, `/usr/share/qt5/examples/webenginewidgets/fancybrowser` and `/usr/share/qt5/examples/webengine/quicknanobrowser`.

To run any of these, after booting up Linux OS on your device, tell Qt 5 which graphics to use by setting the environment variable below. See Section "Qt 5" in the *i.MX Linux® User's Guide* (IMXLUG) for the information on the graphics for different graphical backends.

```
$export QT_QPA_PLATFORM=$Graphics
```

All three browsers can be run by going to the directory above and running the executable found there. Touchscreen can be enabled by adding the parameters `-plugin evdevtouch:/dev/input/event0` to the executable. The DISPLAY variable may need to be set in the environment before beginning:

```
export DISPLAY=:0.0
```

The command line might look like one of these:

```
./browser -plugin evdevtouch:/dev/input/event0
./fancybrowser -plugin evdevtouch:/dev/input/event0
./quicknanobrowser -plugin evdevtouch:/dev/input/event0
```

6 Image Deployment

After a build is complete, the created image resides in `<build directory>/tmp/deploy/images`. An image is, for the most part, specific to the machine set in the environment setup. Each image build creates a U-Boot, a kernel, and an image type based on the IMAGE_FSTYPES defined in the machine configuration file. Most machine configurations provide an SD card image (.sdcard), an ext3 and tar.bz2. The ext3 is the root file system only. The .sdcard image contains U-Boot, the kernel and the rootfs completely set up for use on an SD card.

6.1 Flashing an SD card image

An SD card image provides the full system to boot with U-Boot and kernel. To flash an SD card image, run the following command:

```
$ sudo dd if=<image name>.sdcard  of=/dev/sd<partition> bs=1M && sync
```

For more information on flashing, see Section "Preparing an SD/MMC Card to Boot" in the *i.MX Linux® User's Guide* (IMXLUG).

6.2 Manufacturing Tool, MFGTool

One way to place an image on a device is to use the MFGTool. The recipes used to build a manufacturing tool image are `linux-imx-mfgtool` and `u-boot-mfgtool`.

To build a manufacturing image do the following -

```
 $ bitbake fsl-image-mfgtool-initramfs
```

A manufacturing tool kernel is built using the imx_v7_mfg_defconfig while the default kernel is built by using the imx_v7_defconfig. This is handled automatically by the MFGTool recipes listed above.

For more details on how to use the manufacturing tool, see Section "Serial download mode for the Manufacturing Tool" in the *i.MX Linux® User's Guide* (IMXLUG).

7 Creating a Custom DISTRO

A custom DISTRO can configure a custom build environment. The DISTRO files released fsl-imx-x11, fsl-imx-wayland, fsl-imx-xwayland, and fsl-imx-fb all show configurations for specific graphical backends. DISTROs can also be used to configure other parameters such as kernel, uboot, and gstreamer. The Freescale i.MX DISTRO files are set to create a custom build environment required for testing our i.MX Linux OS BSP releases.

It is recommended for each customer to create their own distro file and use that for setting providers, versions, and custom configurations for their build environment. To create a DISTRO can be done by copying an existing distro file, or including one like poky.conf and adding additional changes, or including one of the Freescale i.MX DISTROs and using that as a starting point.

8 Creating a Custom Board Configuration

Vendors who are developing reference boards may want to add their board to the FSL Community BSP. Having the new machine supported by the FSL Community BSP makes it easy to share source code with the community, and allows for feedback from the community.

The Yocto Project makes it fairly easy to create and share a BSP for a new Freescale based board. The upstreaming process should start when a Linux OS kernel and a bootloader are working and tested for that machine. It is highly important to have a stable Linux kernel and bootloader (for example, U-Boot) to be pointed to in the machine configuration file, to be the default one used for that machine.

Another important step is to determinate a maintainer for the new machine. The maintainer is the one responsible for keeping the set of main packages working for that board. The machine maintainer should keep the kernel and bootloader updated, and the user-space packages tested for that machine. For more information on the machine maintainer role see freescale.github.io/doc/release-notes/1.8/index.html#document-machines-maintainers.

The steps needed are listed below.

1. Customize the kernel config files as needed. The kernel config file is location in arch/arm/configs and the vendoer kernel recipe should customize a version loaded through the kernel recipe.

2. Customize U-Boot as needed. See the *i.MX BSP Porting Guide* (IMXBSPPG) for details on this.

3. Assign someone to be the maintainer of the board. This person makes sure that files are updated as needed so the build always works. For more information see freescale.github.io/doc/release-notes/1.8/index.html#document-machines-maintainers.

4. Set up the Yocto Project build as described in the Yocto Project community instructions, simplified below. Use the community master branch.
   a. Download the needed host package, depending on your host Linux OS distribution, from www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html.

   b. Download repo with the command:

      ```
      $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
      ```

   c. Create a directory to keep everything in. Any name will work. This document is using `fsl-community-bsp`.

```
$ mkdir fsl-community-bsp
```

d. `$ cd fsl-community-bsp`

e. Initialize the repo with the master branch of the repository.

```
$ repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b master
```

f. Get the recipes that will be used to build.

```
$ repo sync
```

g. Set up the environment with:

```
$ source setup-environment build
```

5. Choose a similar machine file in fsl-community-bsp/sources/meta-fsl-arm-extra/conf/machine and copy it, using a name indicative of your board. Edit the new board file with the information about your board. Change the name and description at least. Add MACHINE_FEATURE. See www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#ref-features-machine.

6. Test your changes with the latest community master branch, making sure everything works well. Use at least `core-image-minimal`.

```
$ bitbake core-image-minimal
```

7. Prepare the patches. Follow the style guide at www.openembedded.org/wiki/Styleguide and git.yoctoproject.org/cgit/cgit.cgi/meta-fsl-arm/tree/README in the section entitled *Contributing*.

8. Upstream into meta-fsl-extra. To upstream, join send the patches to meta-freescale@yoctoproject.org.

Appendix A Frequently Asked Questions

A.1 Quick Start

This section summarizes how to set up the Yocto Project on a Linux machine and build an image. Detailed explanations of what this means is in the sections above.

Install the `repo` utility:

To get the BSP you need to have `repo` installed. This only needs to be done once.

```
$: mkdir ~/bin
$: curl http://commondatastorage.googleapis.com/git-repo-downloads/repo  > ~/bin/repo
$: chmod a+x ~/bin/repo
$: PATH=${PATH}:~/bin
```

Download the BSP Yocto Project Environment.

Use the correct name for the release desired in the -b option for `repo init`. This needs to be done once for each release and sets the distribution for the directory created in the first step. `repo sync` can be run to update the recipes under `sources` to the latest.

```
$: mkdir fsl-arm-yocto-bsp
$: cd fsl-arm-yocto-bsp
$: repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.14.52-1.1.0_ga
```

```
$: repo sync
```

Setup for Specific Backends

Setup for X11

```
$: DISTRO=fsl-imx-X11 MACHINE=<machine name> source fsl-setup-release.sh -b build-x11
```

Setup for FB

```
$: DISTRO=fsl-imx-fb MACHINE=<machine name> source fsl-setup-release.sh -b build-fb
```

Setup for Wayland

```
$: DISTRO=fsl-imx-wayland MACHINE=<machine name> source fsl-setup-release.sh -b build-wayland
```

Setup for XWayland

```
$: DISTRO=fsl-imx-xwayland MACHINE=<machine name> source fsl-setup-release.sh -b build-
xwayland
```

Build For All Backends

Build without Qt

```
$: bitbake fsl-image-gui
```

Build with Qt 5

```
$: bitbake fsl-image-qt5
```

A.2 Local configuration tuning

A Yocto Project build can take considerable build resources both in time and disk usage, especially when building in multiple build directories. There are methods to optimize this, for example, use a shared sstate cache (caches the state of the build) and downloads directory (holds the downloaded packages). These can be set to be at any location in the `local.conf` file by adding statements such as these:

```
DL_DIR="/opt/freescale/yocto/imx/download"
SSTATE_DIR="/opt/freescale/yocto/imx/sstate-cache"
```

The directories need to already exist and have appropriate permissions. The shared sstate helps when multiple build directories are set, each of which uses a shared cache to minimize the build time. A shared download directory minimizes the fetch time. Without these settings, Yocto Project defaults to the build directory for the sstate cache and downloads.

Every package downloaded in the DL_DIR directory is marked with a `<package name>.done`. If your network has a problem fetching a package, you can manually copy the backup version of package to the DL_DIR directory and create a <package_name>.touch file with the touch command. Then run the bitbake command: `bitbake <component>`.

For more information, see the Yocto Project Reference Manual.

A.3 Recipes

Each component is built by using a recipe. For new components, a recipe must be created to point to the source (SRC_URI) and specify patches, if applicable. The Yocto Project environment builds from a makefile in the location specified by the SRC_URI in the recipe. When a build is established from auto tools, a recipe should inherit autotools and pkgconfig. Makefiles must allow CC to be overridden by Cross Compile tools to get the package built with Yocto Project.

Some components have recipes but need additional patches or updates. This can be accomplished by using a bbappend recipe. This appends to an existing recipe details about the updated source. For example, a bbappend recipe to include a new patch should have the following contents:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"
SRC_URI += file://<patch name>.patch
```

FILESEXTRAPATHS_prepend tells Yocto Project to look in the directory listed to find the patch listed in SRC_URI.

Tip: If a bbappend recipe is not picked up, view the fetch log file (log.do_fetch) under the work folder to check whether the related patches are included or not. Sometimes a git version of the recipe is being used instead of the version in the bbappend files.

A.4 How to select additional packages

Additional packages can be added to images as long as there is a recipe provided for that package. A searchable list of recipes provided by the community can be found at layers.openembedded.org/. You can search to see if an application already has a Yocto Project recipe and find where to download it from.

A.4.1 Updating an image

An image is a set of packages and the environment configuration.

An image file (such as fsl-image-gui.bb) defines the packages that go inside the file system. Root file systems, kernels, modules, and the U-Boot binary are available in `build/tmp/deploy/images/`.

Note: You can build packages without including it in an image, but you must rebuild the image if you want the package installed automatically on a rootfs.

A.4.2 Package group

A package group is a set of packages that can be included on any image.

A package group can contain a set of packages. For example, a multimedia task could determine, according to the machine, whether the VPU package is built or not, so the selection of multimedia packages may be automated for every board supported by the BSP, and only the multimedia package is included in the image.

Additional packages can be installed by adding the following line in `<build dir>/local.conf`.

```
CORE_IMAGE_EXTRA_INSTALL += "<package_ name1 package_name2>"
```

There are many package groups. Look for them in subdirectories named "packagegroup" or "packagegroups".

A.4.3 Preferred version

The preferred version is used to specify the preferred version of a recipe to use for a specific component. Sometimes a component might have multiple recipes in different layers and a preferred version points to a specific version to use.

In the meta-fsl-bsp-release layer, in `layer.conf`, preferred versions are set for all the recipes to provide a static system for a production environment. These preferred version settings are used for formal Freescale releases but are not essential for future development.

Preferred versions also help when previous versions may cause confusion about which recipe should be used. For example, previous recipes for imx-test and imx-lib used a year-month versioning which has changed to <kernel-version> versioning. Without a preferred version, an older version might be picked up. Recipes that have _git versions are usually picked over other recipes, unless a preferred version is set. To set a preferred version, put the following in `local.conf`.

```
PREFERRED_VERSION_<component>_<soc family> = "<version>"
```

For example, imx-lib would be:

```
PREFERRED_VERSION_imx-lib_mx6 = "5.2"
```

See the Yocto Project manuals for more information on using preferred versions.

A.4.4 Preferred provider

The preferred provider is used to specify the preferred provider for a specific component. A component can have multiple providers. For example, the Linux kernel can be provided by Freescale or by `kernel.org` and preferred provider states the provider to use.

For example, U-Boot is provided by both the community via denx.de and Freescale. The community provider is specified by u-boot-fslc. The Freescale provider is specified by u-boot-imx. To state a preferred provider, put the following in `local.conf`:

```
PREFERRED_PROVIDER_<component>_<soc family> = "<provider>"
PREFERRED_PROVIDER_u-boot_mx6 = "u-boot-imx"
```

A.4.5 SoC family

The SoC family documents a class of changes that apply to a specific set of system chips. In each machine configuration file, the machine is listed with a specific SoC family. For example, i.MX 6DualLite Sabre-SD is listed under the i.MX 6 and i.MX 6DualLite SoC families. i.MX 6Solo Sabre-auto is listed under the i.MX 6 and i.MX 6Solo SoC families. Some changes can be targeted to a specific SoC family in `local.conf` to override a change in a machine configuration file. The following is an example of a change to an mx6dlsabresd kernel setting.

```
KERNEL_DEVICETREE_mx6dl = "imx6dl-sabresd.dts"
```

SoC families are useful when making a change that is specific only for a class of hardware. For example, i.MX 28 EVK does not have a Video Processing Unit (VPU), so all the settings for VPU should use i.MX 5 or i.MX 6 to be specific to the right class of chips.

A.4.6 Bitbake logs

Bitbake logs the build and package processes in the temp directory in `tmp/work/<architecture>/<component>/temp`.

If a component fails to fetch a package, the log showing the errors is in the file log.do_fetch.

If a component fails to compile, the log showing the errors is in the file log.do_compile.

Sometimes a component does not deploy as expected. Check the directories under the build component directory (tmp/work/<architecture>/<component>). Check the package, packages-split, sysroot-destdir directories to see if the files were placed there (where they are staged prior to being copied to the deploy directory).

Appendix B References

- For details on boot switches, see Section "How to Boot the i.MX Boards" in the *i.MX Linux® User's Guide* (IMXLUG).
- For how to download images using U-Boot, see Section "Downloading Images Using U-Boot" in the *i.MX Linux® User's Guide* (IMXLUG).
- For how to set up an SD/MMC card, see Section "Preparing an SD/MMC Card to Boot" in the *i.MX Linux® User's Guide* (IMXLUG).

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMXLXYOCTOUG
Rev. L3.14.52_1.1.0-ga
12/2015