Document Number: IMXLXYOCTOUG
Rev. L3.5.7_1.0.0-alpha, 07/2013

Freescale Yocto User's Guide

1 Overview

The Yocto Project is an open-source collaboration focused on embedded Linux developers. For more information regarding Yocto, please refer to the Yocto Project page: https://www.yoctoproject.org/

The Freescale Yocto Community BSP is a development community outside of Freescale providing support for i.MX boards on the Yocto Project environment. Freescale i.MX joined the Yocto community providing a release based on the Yocto framework. This document illustrates how to build an image for an i.MX Freescale board using a Yocto build environment.

For this release, Freescale provides an additional layer called the Freescale BSP Release known as meta-fsl-bsp-release to integrate a new Freescale release with the Freescale Yocto Community BSP. References to community are for all layers in Yocto except the meta-fsl-bsp-release. The meta-fsl-bsp-release layer is meant to release updated and new Yocto recipes and machine configurations for new releases which are not yet available on the existing meta-fsl-arm and meta-fsl-demos layers in the Yocto Project in addition to a stable release such as Yocto 1.4.1.

Release L3.5.7_1.0.0-alpha is released for Yocto 1.4.1 (dylan) which is the latest stable version. The same recipes for Yocto 1.4.1 are going to be upstreamed to be available on Yocto release 1.5. The Yocto Project release cycle lasts roughly 6 months.

For releases subsequent to 1.4.1, the meta-fsl-bsp-release layer is not required, the Freescale Yocto Community BSP provides support. For example, imx53qsb and imx28evk are some Freescale machines supported in the Freescale Yocto Community BSP. The community also has non Freescale machine configurations. All machine references in this document relate to Freescale machine configuration files.

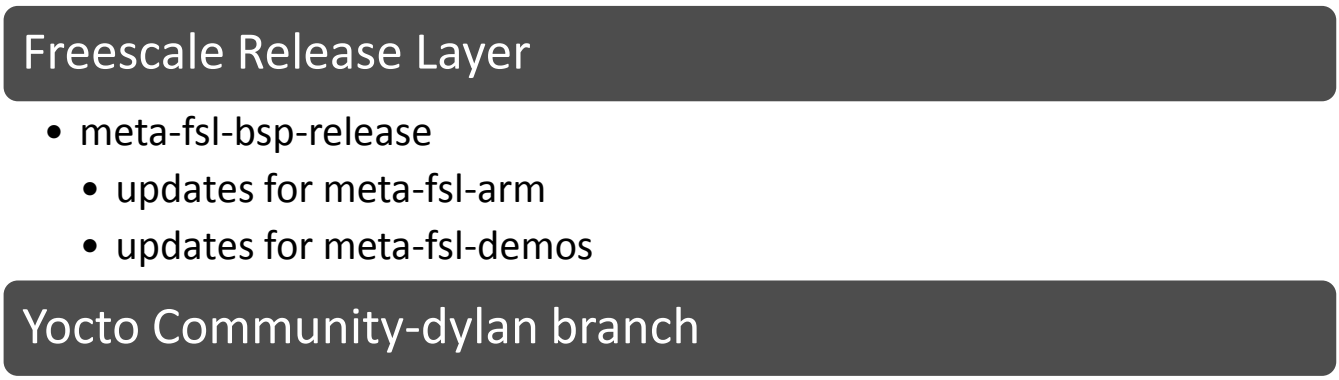
The contents of the Freescale BSP Release layer are recipes and machine configurations. Yocto recipes contain the mechanism to build and package a component. In many cases other layers have implemented recipes or include files and the Freescale release layer provides updates to the recipes by either 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 utilize what the community has provided and update what is needed for each new package version that is not available in the other layers.

Freescale i.MX boards are configured in the meta-fsl-arm layer. This includes U-Boot, kernel, and machine specific details.

Freescale kernel and U-Boot releases are available through Freescale public git servers. However, several components are released as packages on the Freescale mirror. The package based recipes pull from the Freescale mirror instead of a git location and build and package also. Starting with the L3.5.7_1.0.0-alpha release, packages which are released as binary are provided built as a hardware floating point. In a few cases, we have provided a software floating point version. The package selection is determined by using the DEFAULTTUNE setting.

Freescale also provides image recipes which include all components needed for a system image to boot. Components can be built individually or built through an image recipe which pulls in all the components required in an image into one build process.

Below is a diagram of the Freescale Release Layer.



Freescale Yocto Layer Diagram

1.1 End User License Agreement

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

2. Features Summary

The key features of the Freescale Yocto Release layers include the following:

- Linux Kernel Recipe
  - The kernel recipe resides in recipes-kernel folder and integrates a Freescale kernel from the source download in the Freescale git server.
  - Note that L3.5.7_1.0.0-alpha is the first linux kernel that Freescale has released only for the Yocto Project. Previous BSP releases based on Linux version 3.0.35 are released with ltib and the community implementation (from meta-fsl-arm) is recommended.
  - Freescale L3.5.7_1.0.0-alpha is the first Freescale kernel release to support a device tree. This change does add device tree settings in the i.MX 6 machine configuration files.
- U-Boot Recipe
  - The U-Boot recipe resides in recipes-bsp folder and integrates a Freescale uboot-imx.git from source download from the Freecale git server.
  - Certain i.MX boards use different U-Boot versions.
  - Freescale release L3.5.7_1.0.0-alpha for i.MX 6 devices uses an updated v2013.04 Freescale version. This version has not been updated for other i.MX Freescale hardware.
  - Releases based on Linux version 2.6.35 for imx5qsb uses v2009.08 from meta-fsl-arm
  - The Freescale Yocto Community BSP uses u-boot-fslc v2013.04 from mainline.
  - Freescale release L3.5.7_1.0.0-alpha requires using the Freescale v2013-04 U-Boot release.
  - The Freescale Yocto Community BSP updates U-Boot versions frequently and so 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 Freescale graphics package release. For i.MX 6 boards, the gpu-viv-bin-mx6q recipes package the graphic components for each backend – X11, frame buffer (fb) and Direct Frame Buffer (directfb).
    - Xorg-driver integrates our xserver-xorg.
    - For i.MX5, the amd-gpu-bin provides packages for X11 and frame buffer backends.
- i.MX Package recipes
  - imx-lib, imx-test, 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, gstreamer that pull from the Freescale mirror to build and package into image recipes.
  - Some recipes are provided for restricted codecs for which packages are not on the Freescale mirror. These packages are available separately.
- 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. Some releases will not provide any updates unless needed.
- Demo recipes
  - Demonstration recipes reside in the meta-fsl-demos directory. This layer contains image recipes and recipes for customization (such as touch calibration) or recipes for demonstration applications such as glcubes-demo on imx53qsb.

3. Yocto Builds

3. 1 Setup for Builds

A Freescale Yocto Community BSP build requires some packages installed for the build that are documented under the Yocto Project.

Please go to Yocto Project Quick Start and check what packages must be installed for your build machine.

For example, when building on a machine running Ubuntu. The recommended minimum Ubuntu version is 12.04 or later:

```
$ sudo apt-get install sed wget cvs subversion git-core coreutils \
unzip texi2html texinfo libsdl1.2-dev docbook-utils gawk \
python-pysqlite2 diffstat help2man make gcc build-essential \
g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev \
mercurial autoconf automake groff curl

For 3.5.7-1.0.0 – kernel device tree

$ sudo apt-get install lzop

For 3.5.7-1.0.0 – imx-test

$ sudo apt-get install asciidoc
```

3.2 Setup Repo Utility

Install the `repo` utility using the steps below

Create a bin folder in home directory

```
$ mkdir ~/bin (this step may not be needed if the bin folder already exists)

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

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

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

3.3 Retrieve Yocto Layers

Download the Freescale Yocto Community BSP layers using the below steps. This downloads the Yocto layers into “source” directory. To complete this step, use the Freescale Yocto Community BSP setup process. The Freescale i.MX source code is retrieved during the fetch process of the each component’s build process.

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
$ repo init -u http://github.com/Freescale/fsl-community-bsp-platform -b dylan
$ repo sync
```

Once this has completed, the source code is checked out at fsl-community-bsp/sources.

Please do repo sync periodically to update to the latest code.

```
$ MACHINE=<machine file> source ./setup-environment build directory
```

Also EULA is required to be accepted for the first time, after that **ACCEPT_FSL_EULA** is set to 1 and not prompt EULA.

MACHINE=<machine configuration file> is the machine file in meta-fsl-arm/conf/machine. The setup script checks for a valid machine. Without setting the MACHINE, the setup script assumes imx6qsabresd as the default. IMX machine files are provided in meta-fsl-arm/conf/machine and meta-fsl-bsp-release/imx/meta-fsl-arm/conf/machine. Also the MACHINE can be changed in local.conf.

Part of a local.conf created from the setup-environment script is show below.

```
MACHINE ??= 'imx6qsabresd'
DISTRO ?= 'poky'

ACCEPT_FSL_EULA = "1"
```

3.4 Retrieve Freescale Yocto Release Layer

The Freescale Yocto Release layer resides in the meta-fsl-bsp-release git. The community setup-environment does not set up this layer. To retrieve the Freescale Yocto Release layer, the following setup is required in the directory that the setup-environment script was run. Note that this step is only required for new releases such as L3.5.7_1.0.0-alpha.

This setup requires access to the fsl-yocto-release manifest for the new release to be built. This manifest is the community manifest with the Freescale release layer included. Below are the instructions for updating the manifest and downloading the Freescale release layer. The <location> of the manifest is where the manifest is downloaded from the Freescale Yocto release package.

```
$ cd fsl-community-bsp
$ cp .repo/manifest.xml .repo/manifest.orig
$ rm .repo/manifest.xml
$ cp <location>/fsl-yocto-release-manifest.xml .repo/manifest.xml
$ repo sync
$ source fsl-setup-release.sh <optional parameters>
```

This setup script has some optional parameters listed below:

- Setting the build directory if directory name build was not used in setup-environment.
  - -b <build dir >
- Setting the graphical back end for frame buffer and direct fb images. Do not set for X11.
  - -e fb
  - -e dfb

This script integrates the Freescale Yocto Release Layer into the Yocto Build by inserting the layer into the <build dir>/conf/bblayers.conf file.

Executing repo sync after the manifest is copied to download the meta-fsl-bsp-release layer contents into the source directory and retrieve the fsl-setup-release.sh script.

In some releases, new machine configuration files are provided in the meta-fsl-bsp-release layer and the community setup does not find machines while executing the setup-environment script. In these cases you must copy the new machine files into the meta-fsl-arm layer/conf/machine directory and rerun setup-environment with a new build directory.

3.5 Local Configuration Setup

In the build directory, the Yocto community setup script creates a conf/local.conf. This file contains variables that are tuned to your build environment. The setup script looks at the machine's capabilities and sets variables in local.conf based on these capabilities. Variables can be set in the local.conf that override variables set through the machine configuration files.

3.6 Local Configuration Tuning

A Yocto 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 use a shared sstate cache and downloads directory. These can be set at any location in the local.conf

```
DL_DIR="/opt/freescale/yocto/imx/download"

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

These directories should have appropriate permissions. Note the shared sstate helps when multiple build directories are set – each uses a shared cache minimizing build time. A shared download directory minimizes fetch time. Without these settings, Yocto 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. To avoid fetching, touch the .done file for the package name.

3.7 Choose a Freescale Yocto Image

Yocto provides some images which are available on different layers. Poky provides some images, meta-fsl-arm provides others, and new image recipes are provided in the meta-fsl-bsp-release layer. New releases starting in 3.5.7 provide image recipes for each graphics back end such as fsl-image-x11, fsl-image-fb, and fsl-image-dfb for X11 and frame buffer and DirectFB respectively. Each image packages a variety of applications. Below is a table of various images and their contents and what layer provides the image recipe.

Image Name	Target	Provided by Layer
core-image-minimal	A small image just capable of allowing a device to boot.	poky
core-image-base	A console-only image that fully supports the target device hardware.	poky
core-image-sato	Image with sato, a mobile environment and visual style for mobile devices. The image supports X11 with a Sato theme, Pimlico applications and contains terminal, editor and file manager.	poky
fsl-image-test	Builds contents core-image-base plus Freescale test applications and multimedia components.	meta-fsl-arm
fsl-image-x11	Builds contents of core-image-sato with Freescale test applications and multimedia with hardware accelerated X11	meta-fsl-bsp-release/imx/meta-fsl-demos starting with 3.5.7 release layer
fsl-image-fb	Builds a hardware accelerated Frame buffer image with graphics. Requires specific parameters on fsl-setup-release.sh -e fb	meta-fsl-bsp-release/imx/meta-fsl-demos starting with 3.5.7 release layer
fsl-image-dfb	Builds a hardware accelerated direct frame buffer image with graphics. Requires specific parameters on fsl-setup-release.sh -e dfb	meta-fsl-bsp-release/imx/meta-fsl-demos starting with 3.5.7 release layer

Note that the wayland graphics back end is not provided as an image. Instead, for i.MX 6 Wayland binaries are provided in the GPU package. A Wayland image recipe will be supported in future Freescale releases.

3.8 Build an image

Yocto builds using the bitbake command. For example bitbake <component> builds the component selected. 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 dependency per task. The first build is the toolchain and tools required for the components to build.

To build an image:

```
$ bitbake <image name>
```

3.9 Bitbake Options

Bitbake provides various options that are useful when developing on a single component. Below are some options that are listed. To run with a bitbake parameter.

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

Bitbake paramater	Description
-c fetch	Will fetch if downloads is not marked as done.
-c cleanall	Will clean entire component build directory. All changes in the build directory will be lost. The component's rootfs and state is also cleared
-c deploy	Will deploy an image or component to the rootfs
-k	Continue building components even if a build break occurs
-c compile -f	If source is changed, Yocto might not rebuild unless using this option if the state is marked as already deployed. Use this option for forcing a recompile after image has been deployed.
-g	List a dependency tree for an image or component

3.10 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 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 override by Cross Compile tools in order to get the package built with Yocto.

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 updated source. For example, a bbappend recipe to include a new patch should have the following contents.

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

PRINC := "${@int(PRINC) + 1}"

SRC_URI += file://<patch name>.patch
```

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

Tip: If a bbappend recipe is not picked up then check the fetch log. Sometimes a git version of the recipe is being used over the version of the bbappend files.

4. Deploy an Image

Once a build is complete, the image created resides in <build directory>/tmp/deploy/images. An image is specific to the machine set in the setup-environment. Each image build creates a uboot, kernel and image type based on the IMAGE_FSTYPES defined in the machine configuration file. Most machine configurations provide an SD card image, ext3 and tar.bz2. The ext3 is the root file system.

4.1 Flash an SD Card image

An SD card image provides the full system to boot with uboot and kernel. To flash an SD card image run the following command.

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

or use .gz file directly

```
$ gunzip –c <image name>.sdcard.gz | sudo dd of=/dev/sd<partition> bs=4M
```

4.2 Flash rootfs only

To flash rootfs, run the following commands.

```
$ sudo mount /dev/sd<partition> /mnt/card
$ sudo tar jxf <image name>.tar.bz2 -c /mnt/card
```

4.3 U-Boot EIM-NOR

Release L3.5.7_1.0.0-alpha provides machine configuration files for EIM-NOR for i.MX 6 Sabre-AI (Auto Infotainment) machine configurations. These can be built separately using the following (change the machine to the correct target)

```
$ MACHINE=imx6dlsabreauto-eim-nor bitbake -c deploy u-boot-imx
```

Refer to the Freescale U-Boot User's Guide for more information on EIM-NOR.

4.4 U-Boot SPI-NOR

Release L3.5.7_1.0.0-alpha provides machine configuration files for SPI-NOR for i.MX 6 Sabre-AI (Auto Infotainment) machine configurations. These can be built separately using the following (change the machine to the correct target)

```
$ MACHINE=imx6qsabreauto-spi-nor bitbake -c deploy u-boot-imx
```

Refer to the Freescale U-Boot User's Guide for more information on SPI-NOR.

4.5 U-Boot NAND

Release L3.5.7_1.0.0-alpha provides machine configuration files for NAND for i.MX 6 Sabre-AI (Auto Infotainment) machine configurations. These can be built separately using the following (change the machine to the correct target)

```
$ MACHINE=imx6solosabreauto-nand  bitbake -c deploy u-boot-imx
```

Refer to the Freescale U-Boot User's Guide for more information on NAND.

4.6 U-Boot SATA

Release L3.5.7_1.0.0-alpha provides machine configuration files for SATA for i.MX 6 Sabre-SDB machine configuration. This can be built separately using the following (change the machine to the correct target)

```
$ MACHINE=imx6dlsabresd-sata bitbake -c deploy u-boot-imx
```

Refer to the Freescale U-Boot User's Guide for details for more information on SATA.

5 How to Select Additional Packages

Additional packages can be added to images as long as there is a recipe provided for that package.

5.1 Updating an image

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

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

Important 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.

5.2 Package Group

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

A package group can contain a procedure of compilation and installation for a set of packages. For example, a multimedia task could determine, according to the machine, if 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 task is included on the images.

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

```
CORE_IMAGE_EXTRA_INSTALL += “<package_ name1 package_name2>”
```

5.3 Preferred Version

The preferred version is used to document to Yocto the preferred 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 versioning can 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 since 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 the local.conf

```
PREFERRED_VERSION_<component>_<soc family> = “<version>”
```

For example gpu-viv-bin-mx6q would be

```
PREFERRED_VERSION_gpu-viv-bin-mx6q_mx6 = “3.5.7-1.0.0”
```

5.4 Preferred Provider

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

For example, U-Boot is provided by both the community and Freescale. A community provider is u-boot-fslc. A Freescale provider is 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”
```

5.5 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, imx6dlsabresd is listed under mx6 and mx6dl SoC families. Imx6solosabreauto is listed under mx6 and mx6solo SoC families. Some changes can be targeted to a specific SoC family in local.conf to override a change in a machine configuration file. For example below is an example of a change to an mx6dlsabresd kernel setting.

```
KERNEL_DEVICETREE_mx6dl = "${S}/arch/arm/boot/dts/imx6dl-sabresd.dts"
```

SoC families are useful when making a change that is specific only for a class of hardware. For example imx28evk does not have a Video Processing Unit (VPU) so all settings for VPU should use mx5 or mx6 to be specific to the right class of chips.

5.6 Bitbake Logs

Bitbake logs the build and package process in the temp directory in tmp/work/<toolchain>/<component>/temp.

If a component is not fetching a patch, look at the log.do_fetch.

If a component is not compiling, look at log.do_compile.

Sometimes a component does not deploy as expected. Check the package directory under the build component directory.

6 Build Scenarios

The following are build setup scenarios for various configurations.

Download the Freescale Yocto release which contains the fsl-yocto-manifest.xml. Note these two sections are only required for L3.5.7_1.0.0-alpha release.

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
$ repo init -u http://github.com/Freescale/fsl-community-bsp-platform -b dylan
$ cp .repo/manifest.xml .repo/manifest.orig
$ rm .repo/manifest.xml
$ cp <location>/fsl-yocto-release-manifest.xml .repo/manifest.xml
$ repo sync
$ cp -r sources/meta-fsl-bsp-release/imx/meta-fsl-arm/conf/machine sources/meta-
fsl-arm/conf
```

This step copies the new machine configuration files into the meta-fsl-arm so that community setup can find new machine files.

```
$ cp -r sources/meta-fsl-bsp-release/imx/meta-fsl-arm/conf/machine sources/meta-
fsl-arm/conf
```

6.1 X-11 image on i.MX 6Quad Sabre-SD

```
$ MACHINE=imx6qsabresd source setup-environment build-x11
$ cd ..
$ source fsl-setup-release.sh –b build-x11
$ bitbake fsl-image-x11
```

Note: If bitbake shows an error not finding fsl-image-x11, this means that the fsl-setup-release.sh was not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto layer system.

6.2 FB image on i.MX 6Quad Sabre-AI

```
$ MACHINE=imx6qsabreauto source setup-environment build-fb
$ cd ..
$ source fsl-setup-release.sh –b build-fb –e fb
$ bitbake fsl-image-fb
```

Note: If bitbake shows an error not finding fsl-image-x11, this means that the fsl-setup-release.sh was not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto layer system.

6.3 DFB image on i.MX 6Solo Sabre-SD

```
$ MACHINE=imx6solosabresd source setup-environment build-dfb
$ cd ..
$ source fsl-setup-release.sh –b build-dfb –e dfb
$ bitbake fsl-image-dfb
```

Note: If bitbake shows an error not finding fsl-image-x11, this means that the fsl-setup-release.sh was not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto layer system.

6.4 Restarting a build environment after reboot

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

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

Document Number: IMX6LXYOCTOUG
Rev. L3.5.7_1.0.0-alpha
07/2013