Yocto 2.0 Pre-Built Image User's Guide

Rev 2.1 20180228

Contents

1. Environment Requirement....3
1.1 Supported hardware....3
1.2 Software version....3
1.3 Host setup....4
2. Get EDM Yocto BSP Source Code....4
3. Build Yocto Image....5
3.1 Configurations for setup script....5
3.2 Choosing Yocto target image....6
3.3 Build Yocto for TechNexion target platform....7
4. Image Deployment....10
4.1 Flash image into SD card....10
4.2 Flash image into eMMC....10
4.3 Flash image into NAND....12
5. Customize the image release....12
5.1 Change the default audio output....12
5.2 Change the image size....12
5.3 Change the kernel configuration....13
6. Create the toolchain for cross-compiling....13

1. Environment Requirement

1.1 Supported hardware

These are the systems covered in this guide:

System-on-Modules:

- EDM1-CF-IMX6
- EDM1-CF-IMX6P
- EDM1-CF-IMX6QP
- EDM1-CF-IMX6SX
- PICO-IMX6
- PICO-IMX6-POP
- PICO-IMX6UL-EMMC
- PICO-IMX6UL-NAND

Carrier Boards:

- EDM1-FAIRY
- EDM1-GOBLIN
- Toucan0700
- PICO-DWARF
- PICO-HOBBIT
- PICO-NYMPH

Box industrial PC:

- TEK3-IMX6
- TEP5-IMX6

1.2 Software version

name	version
u-boot	2015.04
linux kernel	4.1.15
Yocto	2.0 (jethro)

1.3 Host setup

The build process is tested under Ubuntu-14.04 64bit. So we recommend to set up ubuntu-14.04 environment for building yocto. In building yocto image process, it may take about 75GB hard disk space.

Install Yocto Project host packages:

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

EDM layers host packages for Ubuntu 14.04 host setup only:

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

Install 32-bit compatible libraries for ubuntu14.04 64-bit:

```
sudo apt-get install lib32stdc++6 lib32z1 lib32ncurses5 lib32bz2-1.0 gcc-multilib
```

2. Get EDM Yocto BSP Source Code

There are two ways that you can get EDM Yocto BSP source code.

1. From Technexion FTP:

ftp://ftp.technexion.net/development_resources/Freescale/yocto/

In Yocto section, download the source tarball. There are already pre-downloaded source packages in the “downloads” folder inside the source tarball.

2. From Technexion github:
https://github.com/TechNexion/edm-yocto-bsp

To get the BSP you need to have “repo” installed. Install the “repo” utility:

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

Download the BSP source:

```
PATH=${PATH}:~/bin
mkdir edm_yocto
cd edm_yocto
repo init -u https://github.com/TechNexion/edm-yocto-bsp.git -b jethro_4.1.15-1.1.0_GA
repo sync
```

To speed up the download process, you can add “-j8” after “repo sync”, e.g. “repo sync -j8”.

3. Build Yocto Image

There are various imx6 SOMs and baseboards in the TechNexion product list. To simplify the build configuration for different combination of SOM and baseboard, we create “baseboard” option to select different baseboard for SOM.

3.1 Configurations for setup script

“MACHINE” is the target of build. It usually corresponds to the name of SOM or SBC.

For more information, please check the file under “sources/meta-edm-bsp-release/conf/machine”.

“DISPLAY” is the disply type. This option only works on i.mx6 (i.mx6 Solo/Dual Lite/Dual/Quad) SOMs and doesn’t work on i.mx6ul, i.mx6sx and i.mx7.

“BASEBOARD” is the baseboard type. This option only works on i.mx6 (i.mx6 Solo/Dual Lite/Dual/Quad) SOMs and doesn’t work on i.mx6ul and i.mx6sx.

“-b” specify the build directory.

“-e” sets the graphical back end for frame buffer and direct fb images. X11 is default if no backend is set.

paramerter	Available options	Description
MACHINE	edm1-cf-imx6	Compatible with TechNexion EDM1-CF-IMX6/EDM1-CF-IMX6P/EDM1-CF-IMX6QP (i.MX6 Solo/Dual Lite/Dual/Quad/Quad Plus)
	edm1-cf-imx6-no-console	The same as “edm1-cf-imx6”, but disable the debug console output(This option is used for TC0700)
	pico-imx6	Compatible with TechNexion PICO-IMX6/PICO-IMX6-POP (i.MX6 Solo/Dual Lite/Dual/Quad/Quad pop)
	tek-imx6	Compatible with TechNexion TEK3-IMX6/TEK5-IMX6 (i.MX6 Solo/Dual Lite/Dual/Quad/Quad Plus)
	edm1-cf-imx6sx	Compatible with TechNexion EDM1-CF-IMX6SX with EDM1-GOBLIN baseboard (two LAN port)
	pico-imx6ul-emmc	Compatible with TechNexion PICO-IMX6UL-EMMC with PICO-HOBBIT baseboard
	pico-imx6ul-nand	Compatible with TechNexion PICO-IMX6UL-NAND with PICO-HOBBIT baseboard
	tek3-imx6ul	Compatible with TechNexion TEK3-IMX6UL
	tep3-imx6ul	Compatible with TechNexion TEP3-IMX6UL
DISPLAY	lvds7	7 inch 1024x600 LVDS panel
	hdmi720p	1280x720 HDMI
	hdmi1080p	1920x1080 HDMI

	lcd	5 inch/7 inch 800x480 TTL parallel LCD panel
	lvds7_hdmi720p	Dual display output to both 7 inch LVDS and HDMI
	custom	Reserved for custom panel
BASEBOARD	fairy, tc0700	Compatible with TechNexion EDM1-CF-IMX6/EDM1-CF-IMX6P/EDM1-CF-IMX6QP (i.MX6 Solo/Dual Lite/Dual/Quad/Quad Plus)
	dwarf, hobbit, nymph	Compatible with TechNexion PICO-IMX6/PICO-IMX6-POP (i.MX6 Solo/Dual Lite/Dual/Quad/Quad pop)
-b	<build dir>	Assign the name of build directory
-e	fb	Frame Buffer graphics
	x11	Only X11 graphics
	wayland	Wayland weston graphics. It doesn't support i.mx6ul.

3.2 Choosing Yocto target image

The following bitbake target images are available:

Image name	Target
core-image-minimal	A small image that only allows a device to boot.
core-image-base	A console-only image that fully supports the target device hardware.
core-image-sato	An image with Sato, a mobile environment and visual style for mobile devices. The image supports X11 with a Sato theme, Pimlico applications. It contains a terminal, an editor and a file manager.
fsl-image-machine-test	An FSL Community i.MX core image with console environment - no GUI interface
fsl-image-gui	Builds a Freescale image with a GUI without any QT content. This image recipe works on all backends for X11, DirectFB, Frame Buffer and Wayland
fsl-image-qt5	Builds a QT5 image for X11, Frame Buffer and Wayland backends

3.3 Build Yocto for TechNexion target platform

For EDM1-CF-IMX6/EDM1-CF-IMX6P/EDM1-CF-IMX6QP :

FAIRY baseboard, QT5 with X11 image for HDMI output:

```
MACHINE=edm1-cf-imx6 BASEBOARD=fairy source edm-setup-release.sh -b build-x11-fairy -e x11

bitbake fsl-image-qt5
```

FAIRY baseboard, QT5 with X11 image for 7 inch LVDS panel:

```
DISPLAY=lvds7 MACHINE=edm1-cf-imx6 BASEBOARD=fairy source edm-setup-release.sh \
-b build-x11-fairy  -e x11

bitbake fsl-image-qt5
```

FAIRY baseboard, QT5 with X11 image for 7 inch/5 inch TTL-LCD panel:

```
DISPLAY=lcd MACHINE=edm1-cf-imx6 BASEBOARD=fairy source edm-setup-release.sh \
-b build-x11-fairy  -e x11

bitbake fsl-image-qt5
```

Toucan0700, QT5 with X11 image for 7 inch LVDS panel and disabled debug console:

```
DISPLAY=lvds7 MACHINE=edm1-cf-imx6-no-console BASEBOARD=tc0700 \
source edm-setup-release.sh -b build-x11-toucan -e x11

bitbake fsl-image-qt5
```

For PICO-IMX6/PICO-IMX6-POP :

DWARF baseboard, QT5 with X11 image for HDMI output:

```
MACHINE=pico-imx6 BASEBOARD=dwarf source edm-setup-release.sh -b build-x11-pico -e x11

bitbake fsl-image-qt5
```

DWARF baseboard, QT5 with X11 image for 7 inch LVDS panel:

```
DISPLAY=lvds7 MACHINE=pico-imx6 BASEBOARD=dwarf source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

DWARF baseboard, QT5 with X11 image for 7 inch/5 inch TTL-LCD panel:

```
DISPLAY=lcd MACHINE=pico-imx6 BASEBOARD=dwarf source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

HOBBIT baseboard, QT5 with X11 image for 7 inch LVDS panel:

```
DISPLAY=lvds7 MACHINE=pico-imx6 BASEBOARD=hobbit source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

HOBBIT baseboard, QT5 with X11 image for 7 inch/5 inch TTL-LCD panel:

```
DISPLAY=lcd MACHINE=pico-imx6 BASEBOARD=hobbit source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

NYMPH baseboard, QT5 with X11 image for HDMI output:

```
MACHINE=pico-imx6 BASEBOARD=nymph source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

NYMPH baseboard, QT5 with X11 image for 7 inch LVDS panel:

```
DISPLAY=lvds7 MACHINE=pico-imx6 BASEBOARD=nymph source edm-setup-release.sh \
-b build-x11-pico -e x11

bitbake fsl-image-qt5
```

For TEK3-IMX6 :

TEK3-IMX6 BOX PC, QT5 with X11 image for HDMI output:

```
MACHINE=tek-imx6 source edm-setup-release.sh -b build-x11-tek -e x11

bitbake fsl-image-qt5
```

For TEP5-IMX6 :

TEP5-IMX6 BOX PC, QT5 with X11 image for 10 inch LVDS output:

```
DISPLAY=lvds10 MACHINE=tek-imx6 source edm-setup-release.sh -b build-x11-tek -e x11

bitbake fsl-image-qt5
```

For EDM1-CF-IMX6SX :

Goblin baseboard, X11 image for 7 inch/5 inch TTL-LCD panel:

```
MACHINE=edm1-cf-imx6sx source edm-setup-release.sh -b build-x11-goblin -e x11

bitbake fsl-image-qt5
```

For PICO-IMX6UL-EMMC :

HOBBIT baseboard, X11 image for 7 inch/5 inch TTL-LCD panel:

```
MACHINE=pico-imx6ul-emmc source edm-setup-release.sh -b build-x11-pico-imx6ul -e x11

bitbake fsl-image-gui
```

HOBBIT baseboard, QT5 with FB image for 7 inch/5 inch TTL-LCD panel:

```
MACHINE=pico-imx6ul-emmc source edm-setup-release.sh -b build-fb-pico-imx6ul -e fb

bitbake fsl-image-qt5
```

For PICO-IMX6UL-NAND :

HOBBIT baseboard, X11 image for 7 inch/5 inch TTL-LCD panel:

```
MACHINE=pico-imx6ul-nand source edm-setup-release.sh -b build-x11-pico-imx6ul-nand -e x11

bitbake fsl-image-gui
```

HOBBIT baseboard, QT5 with FB image for 7 inch/5 inch TTL-LCD panel:

```
MACHINE=pico-imx6ul-nand source edm-setup-release.sh -b build-fb-pico-imx6ul-nand -e fb

bitbake fsl-image-qt5
```

Add Chromium into target image :

To enable chromium requires the following steps before “bitbake”:

vim conf/local.conf

```
CORE_IMAGE_EXTRA_INSTALL += "chromium libexif"
LICENSE_FLAGS_WHITELIST="commercial"
```

Every time after changing the display settings, it requires to clean the target build first:

```
bitbake -c clean fsl-image-qt5
```

When you issue the “bitbake” command, you need to make sure the present directory is “build” directory.

If the build process hangs on fetching some packages, please terminate the existing build process then restart it.

4. Image Deployment

When build completes, the generated release image is under **“${BUILD-TYPE}/tmp/deploy/images/${MACHINE}”:**

```
fsl-image-qt5-edm1-cf-imx6-20161208145633.rootfs.ext4
fsl-image-qt5-edm1-cf-imx6-20161208145633.rootfs.manifest
fsl-image-qt5-edm1-cf-imx6-20161208145633.rootfs.sdcard
fsl-image-qt5-edm1-cf-imx6-20161208145633.rootfs.tar.bz2
```

“fsl-image-qt5-edm1-cf-imx6-20161208145633.rootfs.sdcard” is the target image. Just flash this image into your target board to deploy yocto.

4.1 Flash image into SD card

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:

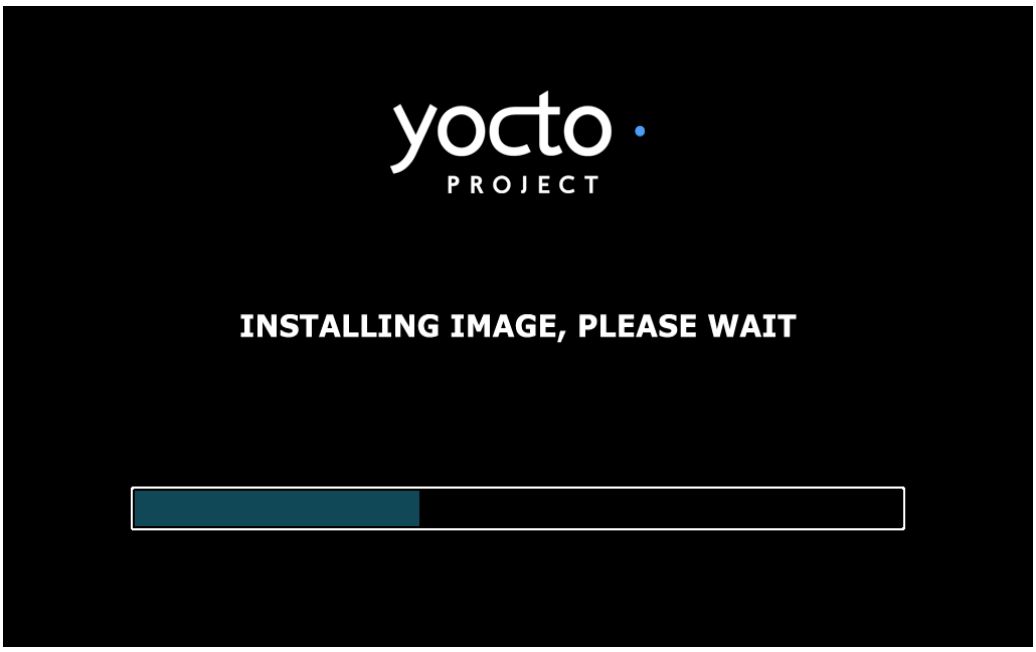
```
sudo dd if=<image name>.sdcard  of=/dev/${sd_partition} bs=1M && sync
```

4.2 Flash image into eMMC

There are two ways to flash image into eMMC.

1. Use installer card to automatically install image into eMMC:

The behavior is like the pre-built image. Set up the hardware boot mode, then insert the SD card. The installing process will automatically start. Please follow the document “**General_Installer_User_Guide.pdf**” in section “**5.3 Automatic mode**”. This method is useful when you need to deploy for mass production.



2. Use generic installer card to boot into USB OTG storge mode:

Please follow the document "**General_Installer_User_Guide.pdf**" in section "**5.2 Storage mode**".

This method is convenient when you are in developing stage. This mode can let you manipulate eMMC as USB storage.

4.3 Flash image into NAND

Please download generic installer image from TechNexion FTP:

ftp://ftp.technexion.net/development_resources/development_tools/installer/

Flash generic installer image into SD card.

Then are four required files for installation.

1. u-boot.imx
2. zImage
3. *.dtb
4. ubi.img

```
$ sudo mount /dev/${sd_partition_1} /mnt/temp
$ cd build-x11-pico-imx6ul-nand/tmp/deploy/images/pico-imx6ul-nand
$ sudo cp u-boot.imx /mnt/temp/image/
$ sudo cp zImage /mnt/temp/image/
$ sudo cp zImage-imx6ul-pico-nand_hobbit.dtb /mnt/temp/image/
$ sudo cp fsl-image-gui-pico-imx6ul-nand.ubi /mnt/temp/image/ubi.img
$ sudo umount /mnt/temp
```

Please refer to "**General_Installer_User_Guide_for_NAND.pdf**" to complete the installation process.

5. Customize the image release

5.1 Change the default audio output

The default audio output for target image is SGTL5000. You can change it to HDMI audio or SPDIF.

vim sources/meta-edm-bsp-release/recipes-multimedia/pulseaudio/pulseaudio/default.pa

```
#set-default-sink alsa_output.platform-sound-hdmi.analog-stereo
set-default-sink alsa_output.platform-sound.analog-stereo
```

5.2 Change the image size

You can specify the image size in local.conf to enlarge the usable spece.

$ vim conf/local.conf

```
MACHINE ??= 'edm1-cf-imx6'
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"
PACKAGECONFIG_append_pn-qemu-native = " sdl"
PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
ASSUME_PROVIDED += "libsdl-native"
CONF_VERSION = "1"

BB_NUMBER_THREADS = '8'
PARALLEL_MAKE = '-j 8'

DL_DIR ?= "${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"

DISPLAY_TYPE = "hdmi720p"

IMAGE_ROOTFS_SIZE = "3000000"
```

5.3 Change the kernel configuration

```
bitbake -c menuconfig virtual/kernel

cp tmp/work/edm1_cf_imx6-poky-linux-gnueabi/linux-tn-imx/4.1.15-r0/build/.config ../sources/meta-edm-bsp-release/recipes-kernel/linux/linux-tn-imx-4.1.15/defconfig

bitbake -c cleansstate virtual/kernel

bitbake fsl-image-qt5
```

6. Create the toolchain for cross-compiling

Bitbake a poky toolchain:

```
bitbake meta-toolchain
```

Install the toolchain in host PC:

Run the installation script located in “build-x11/tmp/deploy/sdk”

For IMX6 (based on ARM Cortex-A9):

```
sh fsl-imx-x11-glibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-toolchain-4.1.15-1.2.0.sh
```

For IMX6UL (based on ARM Cortex-A7):

```
sh fsl-imx-x11-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-4.1.15-1.2.0.sh
```

Compile the C file:
For IMX6:

```
source /opt/fsl-imx-x11/4.1.15-1.2.0/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi

$CC hello_arm_world.c
```

For IMX6UL:

```
source /opt/fsl-imx-x11/4.1.15-1.2.0/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi

$CC hello_arm_world.c
```