Yocto with LSDK Components User Guide

Table of Contents

Introduction 3
Supported Boards 3
Download Yocto Layer 4
- Get the Yocto layers from repo manifest 4
  - Install the repo utility: 4
  - Download the Yocto layers 4
- Get Yocto layers from community repository 4

Build Image 6
Boot boards with Yocto image 7
- Prerequisites 7
- Boot with ramdisk rootfs image 7
- Booting with full rootfs from large storage device 9
- Secure boot 11

Prebuilt Toolchain 12
Known Issue 13
Reference 14

Introduction

Yocto with LSDK components provides recipes for the last Yocto release to use the latest and greatest components from LSDK as they get released. This eventually makes its way into the next community Yocto release at yoctoproject.org.

Supported Boards

The following table Yocto with LSDK components release supports the following QorIQ targets.

Target Type	**Board**	**LE**		**BE**	
		32b	**64b**	**32b**	**64b**
QorIQ LS Series Communications Processors	ls1012ardb	✓	✓	X	X
	ls1012afrwy	✓	✓	X	X
	ls1021atwr	✓	X	X	X
	ls1043ardb	✓	✓	X	✓
	ls1046ardb	✓	✓	X	✓
	ls1088ardb	X	✓	X	✓
	ls1088ardb-pb	X	✓	X	X
	ls2088ardb	X	✓	X	✓
QorIQ T Series Communications Processors	t1024rdb	X	X	✓	✓
	t1042d4rdb	X	X	✓	✓
	t2080rdb	X	X	✓	✓
	t4240rdb	X	X	✓	✓
QorIQ P Series Communications Processors	p1020rdb	X	X	✓	X
	p2020rdb	X	X	✓	X
	p2041rdb	X	X	✓	X
	p3041ds	X	X	✓	X
	p4080ds	X	X	✓	X
	p5040ds	X	X	✓	✓
QorIQ MPC Series Communications Processors	mpc8548cds	X	X	✓	X

Download Yocto Layer

To make sure the build host is prepared for Yocto running and build, please follow below guide to prepare the build environment.

https://www.yoctoproject.org/docs/2.6/brief-yoctoprojectqs/brief-yoctoprojectqs.html

Get the Yocto layers from repo manifest

The following is the step of how to use repo utility to download all Yocto layers according to the repo manifest.

Install the repo utility:

```
$: mkdir ~/bin

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

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

Download the Yocto layers

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

$: mkdir yocto-sdk

$: cd yocto-sdk

$: repo init -u
https://source.codeaurora.org/external/qoriq/qoriq-
components/yocto-sdk -b thud

$: repo sync --no-clone-bundle
```

Get Yocto layers from community repository

The following is the step of how to download all Yocto layers through git commands.

```
$: mkdir yocto-sdk

$: cd yocto-sdk

$: mkdir sources

$: cd sources

$: git clone git://git.yoctoproject.org/poky

$: cd  poky

$: git reset --hard eddff2b361928e88e3628ebc22a1a0ebb119e01b

$: cd ..

$: git clone git://git.openembedded.org/meta-openembedded

$: cd meta-openembedded
$: git reset --hard f1511d254632a34c1deb51f4bf8b8c21e7423f51

$: cd ..

$: git clone git://git.yoctoproject.org/meta-freescale

$: cd  meta-freescale

$: git reset --hard 9d1463c9af81edadf8c5a343a030555b6156f8cf

$: cd ..

$: git clone git://git.yoctoproject.org/meta-virtualization

$: cd meta-virtualization

$: git reset --hard f226bea1083b2bf7373ac7e4780459d454830cc4

$: cd ..

$: git clone git://git.yoctoproject.org/meta-cloud-services

$: cd meta-cloud-services

$: git reset --hard 5dec08769e026980e3b5d3217dbbe2ebd87bf33c

$: cd ..

$: git clone git://git.yoctoproject.org/meta-security

$: cd meta-security

$: git reset --hard dcb0395033e1b4a5d44d467d041114c5cb5e13eb

$: cd ..

$: git clone https://github.com/Freescale/meta-freescale-distro

$: cd meta-freescale-distro

$: git reset --hard 771fb6d4c0c9530083fcc8c5452270bb9b3915ba

$: cd ..

$: git clone https://source.codeaurora.org/external/qoriq/qoriq-components/meta-qoriq-demos

$: cd meta-qoriq-demos

$: git checkout -b thud origin/thud

$: cd ..

$: cp sources/meta-qoriq-demos/scripts/setup-env ./
```

Build Image

The build steps are common for all platforms, the document takes ls1046ardb for an example.

```
$ cd yocto-sdk

$: . ./setup-env -m ls1046ardb

$: bitbake fsl-image-networking

$: bitbake fsl-image-networking-full
```

Note:

1. The images will be available in yocto-sdk/build_ls1046ardb/tmp/deploy/images/ls1088ardb/ folder.
2. The following kernel config options are used for enabling the upstream version of DPAA driver for PowerPC targets:

   CONFIG_FSL_DPAA, CONFIG_FSL_FMAN and CONFIG_FSL_DPAA_ETH need to be enabled in kernel menuconfig as below:

```
Device Drivers  --->

   SOC (System On Chip) specific Drivers  --->

       [*] Freescale DPAA 1.x support  --->

   Device Drivers  --->

      [*] Network device support  --->

      [*]  Ethernet driver support  --->

            <*>     FMan support

            <*>     DPAA Ethernet  ----
```

Boot boards with Yocto image

• Prerequisites

- The tftp server is setup for image download
- A serial cable is connected from your PC to UART1
- The ethernet cable is connected to the first ethernet port on the board.

• Boot with ramdisk rootfs image

- Power up or reset the board and press a key on the terminal when prompted to get to the U-Boot command line
- Set up the environment in U-Boot

```
=> setenv ipaddr <board_ipaddr>

=> setenv serverip <tftp_serverip>
```

Board	Bootargs
ls1021atwr	`=> setenv bootargs root=/dev/ram0 rw console=ttyS0,115200 ramdisk_size=0x1000000`
ls1012a	`=> setenv bootargs root=/dev/ram0 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500 ramdisk_size=0x10000000`
ls1043a	`=> setenv bootargs root=/dev/ram0 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500 ramdisk_size=0x10000000`
ls1046a	`=> setenv bootargs root=/dev/ram0 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500 ramdisk_size=0x10000000`
ls1088ardb	`=> setenv bootargs root=/dev/ram0 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500 ramdisk_size=0x2000000 default_hugepagesz=2m hugepagesz=2m hugepages=512`
ls2088ardb	`=> setenv bootargs root=/dev/ram0 rw console=ttyS1,115200 earlycon=uart8250,mmio,0x21c0600 ramdisk_size=0x2000000 default_hugepagesz=1024m hugepagesz=1024m hugepages=8`
mpc8548cds	`=> setenv bootargs root=/dev/ram rw console=ttyS1,115200 ramdisk_size=1000000 log_buf_len=128K`
t1024rdb	`=> setenv bootargs root=/dev/ram rw console=ttyS1,115200 ramdisk_size=1000000 log_buf_len=128K`

Other PPC targets	`=> setenv bootargs root=/dev/ram rw console=ttyS0,115200 ramdisk_size=1000000 log_buf_len=128K`

- The ls1088ardb and ls2088ardb need the below commands to enable DPAA2 ethernet in Linux

Board	Commands

ls1088ardb	`=> sf probe 0:0`
	`=> sf read 0x80000000 0xA00000 0x100000`
	`=> sf read 0x80100000 0xE00000 0x100000`
	`=> fsl_mc start mc 0x80000000 0x80100000`
	`=> sf read 0x80200000 0xd00000 0x100000`
	`=> fsl_mc lazyapply dpl 0x80200000`

ls2088ardb	`=> fsl_mc start mc 0x580a00000 0x580e00000`
	`=> fsl_mc lazyapply dpl 0x580d00000`

- Download Images and bootup

Board	Commands

ls1021atwr	`=> tftp 82000000 uImage-ls1021atwr.bin`
	`=> tftp 88000000 fsl-image-networking-ls1021atwr.ext2.gz.u-boot`
	`=> tftp 8f000000 uImage-ls1021a-twr.dtb`
	`=> bootm 82000000 88000000 8f000000`

mpc8548cds	`=> tftpboot  0x01000000 uImage-mpc8548cds.bin`
	`=> tftpboot  0x03000000 fsl-image-networking-mpc8548cds.ext2.gz.u-boot`
	`=> tftpboot  0x02000000 uImage-`

	mpc8548cds_32b.dtb => bootm 0x01000000 0x03000000 0x02000000
p1020rdb p2020rdb p2041rdb p3041ds p4080ds p5040ds	=> tftpboot 0x01000000 uImage-<board>.bin => tftpboot 0x02000000 fsl-image-networking-<board>.ext2.gz.u-boot => tftpboot 0x00c00000 uImage-<board>.dtb => bootm 0x01000000 0x04000000 0x02000000
t1024rdb t1042d4rdb t2080rdb t4240rdb	=> tftpboot 0x01000000 uImage-<board>.bin => tftpboot 0x05000000 fsl-image-networking-<board>.ext2.gz.u-boot => tftpboot 0x02000000 uImage-<board>.dtb => bootm 0x01000000 0x05000000 0x02000000
ls1012afrwy-32b ls1012afrwy ls1012ardb-32b ls1012ardb ls1021atwr ls1043ardb-32b ls1043ardb-be ls1043ardb ls1046ardb-32b ls1046ardb-be ls1046ardb ls1088ardb-be ls1088ardb ls1088ardb-pb ls2080ardb ls2088ardb-be ls2088ardb	=> tftp a0000000 fitImage-fsl-image-networking-<board>.bin => bootm a0000000

- Booting with full rootfs from large storage device
    - Prepare the media (SATA/SD/USB) and format it to ext2 format, mount the ext2 partition and extract a full rootfs into this partition, unmount this partition.
    - Power up or reset the board and press a key on the terminal when prompted and get to the U-Boot command line.
    - Set up the environment in u-boot:

```
=> setenv ipaddr <board_ipaddr>

=> setenv serverip <tftp_serverip>
```

Board	Commands
ls1021atwr	=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500

ls1012a series ls1043a series ls1046a series	`=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500`
ls1088ardb	`=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS0,115200 earlycon=uart8250,mmio,0x21c0500 default_hugepagesz=2m hugepagesz=2m hugepages=512`
ls2088ardb	`=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS1,115200 earlycon=uart8250,mmio,0x21c0600 default_hugepagesz=1024m hugepagesz=1024m hugepages=8`
mpc8548cds	`=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS1,115200 log_buf_len=128K`
P and T series	`=> setenv bootargs root=/dev/sda* rootdelay=5 rw console=ttyS0,115200 log_buf_len=128K`

- For ls1088ardb and ls2088ardb, please run below commands to enable DPAA2 ethernet in Linux:

Board	**Commands**

ls1088ardb	`=> sf probe 0:0`
	`=> sf read 0x80000000 0xA00000 0x100000`
	`=> sf read 0x80100000 0xE00000 0x100000`
	`=> fsl_mc start mc 0x80000000 0x80100000`
	`=> sf read 0x80200000 0xd00000 0x100000`
	`=> fsl_mc lazyapply dpl 0x80200000`

ls2088ardb	`=> fsl_mc start mc 0x580a00000 0x580e00000`
	`=> fsl_mc lazyapply dpl 0x580d00000`

- `Download Image and bootup`

Board	Commands

ls1021atwr	`=> tftp 82000000 uImage-ls1021atwr.bin`
	`=> tftp 8f000000 uImage-ls1021a-twr.dtb`
	`=> bootm 82000000 - 8f000000`

mpc8548cds	`=> tftpboot  0x01000000 uImage-mpc8548cds.bin`
	`=> tftpboot  0x02000000 uImage-mpc8548cds_32b.dtb`
	`=> bootm 0x01000000 - 0x02000000`

p2020rdb	`=> tftpboot  0x01000000 uImage-p2020rdb.bin`
	`=> tftpboot  0x00c00000 uImage-p2020rdb-pc_32b.dtb`
	`=> bootm 0x01000000 - 0x00c00000`

p1020rdb
p2041rdb
p3041ds
p4080ds
p5040ds
t1024rdb
t1042d4rdb
t2080rdb
t4240rdb
	`=> tftpboot  0x01000000 uImage-<board>.bin`
	`=> tftpboot  0x02000000 uImage-<board>.dtb`
	`=> bootm 0x01000000 - 0x02000000`

ls1012afrwy-32b
ls1012afrwy
ls1012ardb-32b
ls1012ardb
ls1021atwr
ls1043ardb-32b
ls1043ardb-be
ls1043ardb
ls1046ardb-32b
ls1046ardb-be
ls1046ardb
ls1088ardb-be
ls1088ardb
ls1088ardb-pb
ls2080ardb
ls2088ardb-be
ls2088ardb
	`=> tftp a0000000 fitImage-<board>.bin`
	`=> bootm a0000000:kernel@1 - a0000000:<fdt name>`
	
	`NOTE: <fdt name> can be got by the following command:`
	`      $: grep fdt@ fitImage-its-<board>.its`

• Secure boot

To enable the secure boot on QorIQ platforms, please refer to the "6.1 Secure boot" section of the following LSDK documentation.

https://www.nxp.com/support/developer-resources/run-time-software/linux-software-and-development-tools/layerscape-software-development-kit:LAYERSCAPE-SDK?tab=Documentation_Tab

Prebuilt Toolchain

The prebuilt toolchain for support targets are available in NXP official image mirror.

ARM32: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-cortexa7hf-neon-toolchain-2.6.sh

ARM64: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-aarch64-toolchain-2.6.sh

ARM64-BE: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-aarch64_be-toolchain-2.6.sh

PPCE500V2: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce500v2-toolchain-2.6.sh

PPCE500MC: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce500mc-toolchain-2.6.sh

PPCE5500: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce5500-toolchain-2.6.sh

PPCE5500-64B: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppc64e5500-toolchain-2.6.sh

PPCE6500: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce6500-toolchain-2.6.sh

PPCE6500-64B: https://www.nxp.com/lgfiles/sdk/lsdk1809-yocto26/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppc64e6500-toolchain-2.6.sh

Known Issue

ID	Description	Disposition	Opened In	Workarounds
QLINUX -10188	PM sleep test failed on T1042D4RDB	Open	LSDK-1809	

QLINUX -9692	T2 optical_xfi port not Works	Open	LSDK-1809	update t2080rdb.dts
				ethernet@f0000 {
				- phy-handle = <&xg_cs4315_phy1>;
				+ phy-handle = <&xg_cs4315_phy2>;
				phy-connection-type = "xgmii";
				};
				ethernet@f2000 {
				- phy-handle = <&xg_cs4315_phy2>;
				+ phy-handle = <&xg_cs4315_phy1>;
				phy-connection-type = "xgmii";
				};

QLINUX -10670	Guest rootfs boot failed on Text2.gz rootfs 2080RDB and T4240RDB with	Open	LSDK-1809	
QSDK -5118	Openssl offload : asymmetric_ciphers_rsa_caam_jr failed to work	Open	LSDK-1809	

Reference

- NXP LSDK official website: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/qoriq-layerscape-arm-processors/layerscape-software-development-kit-v18.09:LAYERSCAPE-SDK
- NXP LSDK github portal: https://lsdk.github.io/
- Yocto Open Source User Guide: https://www.yoctoproject.org/docs/