Document Number: LSDK
Rev. YP 2.7– LSDK 1906

Yocto with LSDK Components User Guide

Table of Contents

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	X
	ls1012afrwy	X	✔	X	X
	ls1021atwr	✔	X	X	X
	ls1043ardb	X	✔	X	X
	ls1046ardb	X	✔	X	X
	ls1088ardb-pb	X	✔	X	X
	lx2160ardb	X	✔	X	X
	ls2088ardb	X	✔	X	X
QorIQ T Series Communications Processors	t1024rdb	X	X	✔	✔
	t2080rdb	X	X	X	✔
	t4240rdb	X	X	X	✔
	t1042d4rdb	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/2.7/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 warrior

    $: 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 0e392026ffefee098a890c39bc3ca1f697bacb52

  $: cd ..

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

  $: cd meta-openembedded
  ```

```
$: git reset --hard 0477c76116cd1dc479d0df0e9721cbbd729ac4d2
$: cd ..
$: git clone git://git.yoctoproject.org/meta-freescale
$: cd  meta-freescale
$: git reset --hard 3cf8e849e501c1243c0710889e97d14477688784
$: cd ..
$: git clone git://git.yoctoproject.org/meta-virtualization
$: cd meta-virtualization
$: git reset --hard abcdb5841dfffc1ff4410f6c149dc304d4ac5e42
$: cd ..
$: git clone git://git.yoctoproject.org/meta-cloud-services
$: cd meta-cloud-services
$: git reset --hard 215b97571f52d43915eda1fdc5cd80719687abcd
$: cd ..
$: git clone git://git.yoctoproject.org/meta-security
$: cd meta-security
$: git reset --hard 5959e4f4bf6120edf82c71e6c7a0b6118f275419
$: cd ..
$: git clone https://github.com/Freescale/meta-freescale-distro
$: cd meta-freescale-distro
$: git reset --hard 8fbf269e32650cd9392603c1c81d8ace4111f89b
$: cd ..
$: git clone https://source.codeaurora.org/external/qoriq/qoriq-components/meta-qoriq-demos
$: cd meta-qoriq-demos
$: git reset --hard dd64b165170782d745d7b8297c699086af6d5b88
$: 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
$: bitbake atf
```

Note:

1. The images will be available in yocto-sdk/build_ls1046ardb/tmp/deploy/images/ls1046ardb/ folder.

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-pb	=> 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
lx2160ardb	=> setenv bootargs console=ttyAMA0,115200 root=/dev/ram0 rw rootdelay=10 earlycon=pl011,mmio32,0x21c0000 ramdisk_size=0x2000000 default_hugepagesz=1024m hugepagesz=1024m hugepages=2 pci=pcie_bus_perf

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=ttyS0,115200 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 , lx2160ardb and ls2088ardb need the below commands to enable DPAA2 ethernet in Linux

Board	Commands
ls1088ardb-pb	=> 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
lx2160ardb	=> fsl_mc start mc 0x20a00000 0x20e00000 => fsl_mc lazyapply dpl 0x20d00000
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-64b t4240rdb-64b	=> 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 ls1012ardb	=> pci enum => tftp a0000000 fitImage-fsl-image-networking-<board>.bin => pfe stop => bootm a0000000
ls1021atwr ls1043ardb ls1046ardb ls1046afrwy ls1088ardb-pb lx2160ardb ls2080ardb 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

lx2160ardb	=> setenv bootargs console=ttyAMA0,115200 root=/dev/sda* rw rootdelay=10 earlycon=pl011,mmio32,0x21c0000 ramdisk_size=0x2000000 default_hugepagesz=1024m hugepagesz=1024m hugepages=2 pci=pcie_bus_perf

-

- For ls1088ardb ,lx2160ardb 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
lx2160ardb	=> fsl_mc start mc 0x20a00000 0x20e00000 => fsl_mc lazyapply dpl 0x20d00000
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
ls1012afrwy ls1012ardb	=> pci enum => tftp a0000000 fitImage-<board>.bin => pfe stop => bootm a0000000
ls1021atwr ls1043ardb ls1046ardb ls1046afrwy ls1088ardb lx2160ardb ls2080ardb 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

For build secure boot image ,you need to set the following variables in local.conf

DISTRO_FEATURES_append = " secure"

ROOTFS_IMAGE = "fsl-image-mfgtool"

For arm64 targets:

KERNEL_ITS = "kernel-all.its"

For arm32 targets:

KERNEL_ITS = "kernel-arm32.its"

$: bitbake secure-boot-qoriq

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

How to program TF-A binaries on specific boot mode

•QSPI NOR Flash

1. Boot from QSPI NOR flash0
2. Program QSPI NOR flash1: => sf probe 0:1
3. Flash bl2_qspi.pbl:

   => tftp 0xa0000000 bl2_qspi.pbl

   => sf erase 0x0 +$filesize && sf write 0xa0000000 0x0 $filesize
4. Flash fip_uboot.bin:

   => tftp 0xa0000000 fip_uboot.bin

   => sf erase 0x100000 +$filesize && sf write 0xa0000000 0x100000 $filesize
5. Boot from QSPI NOR flash1. The board will boot with TF-A

•SD Card

1. Boot from QSPI NOR flash0.
2. Flash bl2_sd.pbl on SD card:

   => tftp 82000000 bl2_sd.pbl

   => mmc write 82000000 8 <blk_cnt>

Here, blk_cnt refers to number of blocks in SD card that need to be written as per the file size. For example, when you load bl2_sd.pbl from the TFTP server, if the bytes transferred is 82809 (14379 hex), then blk_cnt is calculated as82809/512 = 161 (A1 hex). For this example, mmc write command will be: => mmc write 82000000 8 A1.

3. Flash fip_uboot.bin on SD card:

   => tftp 82000000 fip_uboot.bin

   => mmc write 82000000 800 <blk_cnt>

4. Boot from SD card. The board will boot with TF-A

•NOR Flash

1. Boot from default bank.
2. Flash bl2_nor.pbl on alternate bank:

   => tftp 82000000 $path/bl2_nor.pbl;

   => pro off all;erase 64000000 +$filesize;cp.b 82000000 64000000 $filesize

3. Flash fip_uboot.bin on alternate bank:

   => tftp 82000000 $path/fip_uboot.bin;

   => pro off all;erase 64100000 +$filesize;cp.b 82000000 64100000 $filesize

4. Boot the board from alternate bank. The board will boot with TF-A.

•NAND Flash

1. Flash bl2_nand.pbl:

   => tftp 82000000 $path/bl2_nand.pbl

   => nand erase 0x0 $filesize;nand write 0x82000000 0x0 $filesize;

2. Flash fip_uboot.bin:

   => tftp 82000000 $path/fip_uboot.bin

   => nand erase 0x100000 $filesize;nand write 0x82000000 0x100000 $filesize;

3. Then boot from NAND flash. The board will boot with TF-A

Qoriq memory layout

Table-1: The unified 64MB memory layout of NOR/QSPI/NAND/SD media on all QorIQ platforms

Firmware Definition	MaxSize	Flash Offset	SD Start Block#
RCW + PBI + BL2 (bl2.pbl)	1MB	0x00000000	0x00008

ATF FIP Image (fip.bin) BL31 + BL32 + BL33		4MB	0x00100000	0x00800
Bootloader environment		1MB	0x00500000	0x02800
Secure boot headers		2MB	0x00600000	0x03000
DDR PHY FW or reserved		512KB	0x00800000	0x04000
Fuse provisioning header		512KB	0x00880000	0x04400
DPAA1 FMAN ucode		256KB	0x00900000	0x04800
QE firmware or DP firmware		256KB	0x00940000	0x04A00
Ethernet PHY firmware		256KB	0x00980000	0x04C00
Script for flashing image		256KB	0x009C0000	0x04E00
DPAA2-MC or PFE firmware		3MB	0x00A00000	0x05000
DPAA2 DPL		1MB	0x00D00000	0x06800
DPAA2 DPC		1MB	0x00E00000	0x07000
Device tree(needed by uefi)		1MB	0x00F00000	0x07800
Kernel	lsdk_linux.itb	16MB	0x01000000	0x08000
Ramdisk rfs		30MB	0x02000000	0x10000
CA or other uses		2MB	0x03e00000	0x1F000

Table-2: The unified 2MB memory layout (e.g. on LS1012AFRWY platform)

Firmware Definition	MaxSize	Location	SD Start Block#
RCW+PBI+BL2 (bl2.pbl)	64KB	0x0000_0000 - 0x0000_FFFF	0x00008
Reserved	64KB	0x0001_0000 - 0x0001_FFFF	0x00080
PFE firmware	256KB	0x0002_0000 - 0x0005_FFFF	0x00100
FIP (BL31+BL32+BL33)	1MB	0x0006_0000 - 0x000D_FFFF	0x00300
Environment varialbes	64KB	0x001D_0000 - 0x001D_FFFF	0x00E80
Reserved	64KB	0x001E_0000 - 0x001E_FFFF	0x00F00
Secureboot headers	64KB	0x001F_0000 - 0x001F_FFFF	0x00F80

Prebuilt Toolchain

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

ARM32: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-cortexa7hf-neon-ls1021atwr-toolchain-2.7.sh

ARM64: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-aarch64-ls2088ardb-toolchain-2.7.sh

PPCE500V2: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce500v2-p1020rdb-toolchain-2.7.sh

PPCE500MC: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce500mc-p4080ds-toolchain-2.7.sh

PPCE5500: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce5500-t1024rdb-toolchain-2.7.sh

PPCE5500-64B: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppc64e5500-t1024rdb-64b-toolchain-2.7.sh

PPCE6500: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppce6500-t4240rdb-toolchain-2.7.sh

PPCE6500-64B: https://www.nxp.com/lgfiles/sdk/lsdk1903-yocto27/fsl-qoriq-glibc-x86_64-fsl-toolchain-ppc64e6500-t4240rdb-64b-toolchain-2.7.sh

Known Issue

ID	Description	Disposition	Opened In	Workarounds
QYOCTO-583	crconf update command can't finish by itself	Open	Yocto 2.7	
QYOCTO-586	guest rootfs boot failed on T2080RDB and T4240RDB with ext2.gz on Yocto 2.6 and Yocto 2.7	Open	Yocto 2.7	
QYOCTO-554	the priority of ceetm doesn't work well on dpaa1 platform	Open	Yocto 2.6	
QYOCTO-581	Bridging can't work in YOCTO2.7 full rootfs	Open	Yocto 2.7	
QYOCTO-584	optee testing failed on yocto 2.7	Open	Yocto 2.7	

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/