NXP Semiconductors
Application Note

Document Number: AN5340
Rev. 0, 10/2016

Updating SDK Images on QorIQ LS1046ARDB

Contents

1 Introduction

This application note describes how to deploy U-Boot, Linux kernel and the root file system to the QorIQ LS1046A Reference Design Board (RDB) using NXP SDK images.

The purpose of this document is to enable the user to update and build sdk images on the QorIQ LS1046ARDB using the default boot process as well as alternative deployment methods.

For more information on deployment, see the QorIQ SDK V2.0 Documentation.

2 Install the SDK on Host Computer and Build Images (Step 1)

There are three options to get started with installing the SDK and building the images:

- No modifications are needed to images from LS1046A BSP V0.4: follow instructions on Use LS1046A BSP V0.4 Pre-built Images on page 1. Skip sections 2.2 - 2.7.
- Need to modify images from LS1046A BSP V0.4 and QorIQ SDK V2.0 is not yet installed: follow instructions starting on Install SDK V2.0 on page 2. Skip section 2.1.
- Need to modify images from LS1046A BSP V0.4 and QorIQ SDK V2.0 is already installed: follow instructions on Install LS1046A BSP V0.4 on page 4 . Skip sections 2.1, 2.2 - 2.6.

2.1 Use LS1046A BSP V0.4 Pre-built Images

Follow the instructions below, if you do not need to modify the images from LS1046A BSP V0.4 or would like to get Linux running quickly. Use the LS1046A BSP V0.4 pre-built images within `SDK2.0_LS1046ABSP_V0.4.tar`.

1. Download the `SDK2.0_LS1046ABSP_V0.4.tar` file from www.nxp.com/LS1046ARDB.
2. Untar the `SDK2.0_LS1046ABSP_V0.4.tar` file.
3.
```
cd SDK2.0_LS1046ABSP_V0.4/images/ls1046ardb
```
4. Use the following image:
   - U-Boot: `u-boot-ls1046ardb.bin`
   - Frame Manager Microcode: `fsl_fman_ucode_t2080_r1.1_106_4_18.bin`
   - Reset Configuration Word: `rcw/ls1046ardb/RR_FFPPPN_1133_5559/rcw_1600.bin`
   - Linux: `kernel-fsl-ls1046a-rdb.itb`

2.2 Install SDK V2.0

As a prerequisite for users who need to modify images from LS1046A BSP V0.4, the QorIQ SDK V2.0 source ISO needs to be installed. In order to speed up the build, the QorIQ SDK V2.0 cache ISO is also recommended to be installed. The source and cache ISO's can be downloaded from Linux SDK for QorIQ Processors.

If the QorIQ SDK V2.0 ISO's have already been installed, see Install LS1046A BSP V0.4 on page 4. Otherwise, follow the instructions below to install QorIQ SDK V2.0.

1. Mount and install the source ISO on your computer:

```
$ sudo mount -o loop QorIQ-SDK-V2.0-SOURCE-20160527-yocto.iso /mnt/cdrom
$ /mnt/cdrom/install
```

2. Mount and install the cache ISO on your computer:

```
$ sudo mount -o loop QorIQ-SDK-V2.0-AARCH64-CACHE-20160527-yocto.iso /mnt/cdrom
$ /mnt/cdrom/install
```

3. When prompted to input the install path, ensure that the current user has the correct permission for the install path.

There is no uninstall script. To uninstall Yocto Project, you can remove the `<yocto_install_path>/QorIQ-SDK-<version>-<yyyymmdd>-yocto` directory manually.

NOTE

* The *source* ISO contains the package source tarballs and Yocto Project recipes. It can be installed and used to do non-cache build.

* The *cache* ISO contains the pre-built cache binaries. To avoid a long time build, you can install the source ISO and the cache ISO in the same installation folder.

2.3 Set Up Host Environment

Yocto Project requires some packages to be installed on the host.

Use the following steps to prepare the Yocto Project environment.

In general, Yocto Project can work on most recent Linux distributions with Python-2.7.3 or greater (excluding python3 which is not supported), git-1.7.8 or greater, tar-1.24 or greater and required packages installed. The default Python is not 2.7.x on some Linux distros, e.g. CentOS 6.5 installs python 2.6.6. Follow the instructions below to install the Python 2.7.x in custom path instead of override the system default python, the override may cause system utilities breaking.

```
$ wget https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
[NOTE: Python 2.7.3 and python 2.7.5 can be used as well.]
$ tar -xf Python-2.7.6.tar.xz
$ cd Python-2.7.6
$ ./configure --prefix=/opt/python-2.7.6
$ make
$ sudo make install
```

Run the export command below to ensure python 2.7.x is used for Yocto build.

```
$ export PATH=/opt/python-2.7.6/bin:$PATH
```

Yocto Project supports typical Linux distributions: Ubuntu, Fedora, CentOS, Debian, OpenSUSE, etc. More Linux distributions are continually being verified. This SDK has been verified on following Linux distributions: Ubuntu 14.04, CentOS-7.1.1503, Debian 8.2, Fedora 22 and OpenSUSE 13.2

For a list of the Linux distributions tested by the Yocto Project community see `SANITY_TESTED_DISTROS` in `poky/meta-yocto/conf/distro/poky.conf`.

The following is the detailed package list on the CentOS hosts:

```
$ sudo yum install gawk make wget tar bzip2 gzip python unzip perl patch \
     diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath socat SDL-devel xterm
```

For the Fedora hosts:

```
$ sudo yum install gawk make wget tar bzip2 gzip python unzip perl patch \
     diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath \
     ccache perl-Data-Dumper perl-Text-ParseWords perl-Thread-Queue socat \
     findutils which SDL-devel xterm
```

For Ubuntu and Debian hosts:

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

Extra packages are needed for Ubuntu-64b:

```
$ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 ia32-libs lib32ncurses5-dev
```

For OpenSUSE host:

```
$ sudo zypper install python gcc gcc-c++ libtool subversion git chrpath automake make wget
diffstat makeinfo freeglut-devel libSDL-devel
```

2.4 Set Up Poky

Source the following poky script to set up your environment for the LS1046ARDB. This script needs to be run once for each terminal, before you begin building the source code.

```
$ . ./fsl-setup-env -m ls1046ardb
```

The following shows the usage text for the `fsl-setup-env` command:

Usage:

```
$ . ./fsl-setup-env -h
Usage: . fsl-setup-env -m <machine>

    Supported machines
    ls1046ardb

    Optional parameters:
    * [-m machine]: the target machine to be built.
    * [-b path]:    non-default path of project build folder.
    * [-j jobs]:    number of jobs for make to spawn during the compilation stage.
    * [-t tasks]:   number of BitBake tasks that can be issued in parallel.
    * [-d path]:    non-default path of DL_DIR (downloaded source)
    * [-c path]:    non-default path of SSTATE_DIR (shared state Cache)
```

```
    * [-g]:         enable Carrier Grade Linux
    * [-l]:         lite mode. To help conserve disk space, deletes the building
                    directory once the package is built.
    * [-h]:         help
```

2.5 Build fsl-image-core

Follow the steps below to build the image core using Yocto Project. Be sure to set up the host environment before doing these steps.

1.
```
$ cd <sdk-install-dir>/build_ls1046ardb
```

2.
```
$ bitbake fsl-image-core
```

NOTE

`fsl-image-core`: contains common open source packages and NXP specific packages. For additional Yocto Project usage information, see https://www.yoctoproject.org/

2.6 Build Kernel itb

Follow the steps below to build the kernel itb using Yocto Project. Be sure to set up the host environment before doing these steps.

1.
```
$ cd <ISO-install-dir>/QorIQ-SDK-V2.0-20160527-yocto
```

2.
```
$ bitbake fsl-image-kernelitb
```

NOTE

`fsl-image-kernelitb`: is a FIT image that includes the Linux image, dtb and rootfs image. For additional Yocto Project usage information, see https://www.yoctoproject.org/

2.7 Install LS1046A BSP V0.4

As a prerequisite for users who need to modify images from the LS1046A BSP V0.4, the QorIQ SDK V2.0 source ISO needs to be installed. If the QorIQ SDK V2.0 ISO's have already been installed, follow the instructions below:

1. Download the `SDK2.0_LS1046ABSP_V0.4.tar` file from www.nxp.com/LS1046ARDB.
2. Untar the `SDK2.0_LS1046ABSP_V0.4.tar` file, and run the 'install' script to install the updates and sources to ISO install folder:

```
$ tar -xjf SDK2.0_LS1046ABSP_V0.4.tar
$ ./SDK2.0_LS1046ABSP_V0.4.tar/install
```

3. During the install process, the user will be prompted to input the QorIQ SDK V2.0 ISO installed location, i.e., `<ISO_INSTALL_PATH>/QorIQ-SDK-V2.0-20160527-yocto`.

2.8 Other Options

For optional targets and kernel itb, see the QorIQ SDK V2.0 Documentation.

3 Setup Board Switch Settings (Step 2)

The RDB has user selectable switches for evaluating different boot options for the LS1046A device. The table below lists the default switch settings that will come with the board. As a part of setup, make sure the settings below are correct before updating SDK images on LS1046ARDB.

Table 1. Default Switch Settings

	1	2	3	4	5	6	7	8
SW3	0	1	0	0	0	1	1	0
SW4	0	0	1	1	1	0	1	1
SW5	0	0	1	0	0	0	1	0

The table below displays additional switch settings for alternate boot devices. Note that changing the boot device configuration may require additional changes in the RCW or in other code images.

Table 2. Alternate Boot Devices Switch Settings

Boot Source	Switch
QSPI flash 0 (bank0)	SW5[1-8] +SW4[1] = 0b'00100010_0 SW3[3-5]= 0b'000
QSPI flash 1 (bank4)	SW5[1-8] +SW4[1] = 0b'00100010_0 SW3[3-5]= 0b'001
SD	SW5[1-8] +SW4[1] = 0b'00100000_0

For a complete list of switch settings, see QorIQ LS1046A Reference Design Board Getting Started Guide.

4 Program Factory Default Images to Board (Step 3)

4.1 Program All Images to Bank 4

The LS1046ARDB has 2 QSPI flash connected over the QSPI contoller. This is very helpful during development, because you can use the U-Boot image in one bank to program an image set into the alternate bank. If the new images on the alternate bank are flawed, the original images in bank0 are still functional to let you deploy corrected images.

Only one QSPI flash is available at a time depending on the board switch settings. These switch settings can also be overriden by CPLD commands.

To protect the working U-Boot in bank0, it is a convention employed by NXP to deploy newly built images into bank4 (alternate bank), and then switch to bank4 for testing. Switching to the alternate bank can be done in software, which effectively swaps bank0 with bank4, thereby putting the alternate bank in the bank0 address range until further configuration or until a reset

occurs. This process protects images in bank0, so that the LS1046ARDB board can boot from bank0 if the newly built images in bank4 do not work.

To determine the current bank, see the U-Boot log:

```
U-Boot 2016.01 (Aug 19 2016 - 16:59:27 +0800)

SoC:  LS1046E (0x87070010)
Clock Configuration:
       CPU0(A72):1600 MHz  CPU1(A72):1600 MHz  CPU2(A72):1600 MHz
       CPU3(A72):1600 MHz
       Bus:      600  MHz  DDR:      2100 MT/s  FMAN:     700  MHz
Reset Configuration Word (RCW):
       00000000: 0c150010 0e000000 00000000 00000000
       00000010: 11335559 40005012 40025000 c1000000
       00000020: 00000000 00000000 00000000 00238800
       00000030: 20124000 00003101 00000096 00000001
I2C:   ready
Model: LS1046A RDB Board
Board: LS1046ARDB, boot from QSPI vBank 0
CPLD:  V2.2
PCBA:  V2.0
SERDES Reference Clocks:
SD1_CLK1 = 156.25MHZ, SD1_CLK2 = 100.00MHZ
DRAM:  Initializing DDR....using SPD
Detected UDIMM 18ASF1G72AZ-2G3B1
8 GiB (DDR4, 64-bit, CL=15, ECC on)
       DDR Chip-Select Interleaving Mode: CS0+CS1
SEC0: RNG instantiated
PPA Firmware: Version 0.2
Using SERDES1 Protocol: 4403 (0x1133)
Using SERDES2 Protocol: 21849 (0x5559)
NAND:  512 MiB
MMC:   FSL_SDHC: 0
SF: Detected S25FL512S_256K with page size 256 Bytes, erase size 256 KiB, total 64 MiB
EEPROM: Invalid ID (5a 5a 5a 5a)
PCIe1: Root Complex no link, regs @ 0x3400000
PCIe2: Root Complex no link, regs @ 0x3500000
PCIe3: Root Complex no link, regs @ 0x3600000
In:    serial
Out:   serial
Err:   serial
SATA link 0 timeout.
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq pm clo only pmp fbss pio slum part ccc apst
Found 0 device(s).
SCSI:  Net:   SF: Detected S25FL512S_256K with page size 256 Bytes, erase size 256 KiB, total 64 MiB
Fman1: Uploading microcode version 106.4.15
FM1@DTSEC3 [PRIME], FM1@DTSEC4, FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1, FM1@TGEC2
Hit any key to stop autoboot:  0
=>
```

Switch banks in U-Boot using the following statements:

- Switch to QSPI bank 0 (default) for RDB:

```
=>cpld reset
```

- Switch to QSPI bank 4 for RDB:

```
=>cpld reset altbank
```

4.1.1 Program U-Boot

By default, an existing U-Boot runs in QSPI bank0 after the system is powered on, or after a hard reset is performed. To flash U-Boot to the alternate bank, first switch to bank0 by performing a hard reset or by typing *reset*. Then use the following commands to flash a new U-Boot into the alternate bank and then switch to the alternate bank where the new U-Boot is flashed:

```
=>tftp 82000000 <u-boot_file_name>.bin
=>sf probe 0:1
=>sf erase 100000 +$filesize
=>sf write 82000000 100000 $filesize
=>cpld reset altbank
```

NOTE

1. Using "sf probe 0:0" could program U-Boot to the current bank.
2. The U-Boot image generated from code needs to be byte swapped. Yocto Project generates the byte-swapped U-Boot image, which is programmed into QSPI flash. For steps to build U-Boot without using Yocto Project, see Build U-Boot and RCW without Yocto Project and Program to QSPI Flash on page 8.

4.1.2 Program Reset Configuration Word (RCW)

To program a new RCW, first switch to QSPI bank0 by performing a hard reset or by typing *reset*. Next, load the new RCW to RAM by downloading it via TFTP and then copying it to flash offset `0x0`. `0x0` is the offset address of the RCW in the QSPI flash. Then use the following commands to program the RCW to QSPI flash and reset to alternate bank:

```
=>tftp 82000000 <rcw_file_name>.bin
=>sf probe 0:1
=>sf erase 0 +$filesize
=>sf write 82000000 0 $filesize;
=>cpld reset altbank
```

NOTE

Using "sf probe 0:0" could program RCW to the current bank.

For more information see RCW (Reset Configuration Word) and Ethernet Interfaces

4.1.3 Program Frame Manager Microcode (FMan Ucode)

To program a new microcode, first switch to QSPI bank0 by performing a hard reset or by typing *cpld reset*. Next, load the new microcode to RAM by downloading it via TFTP and then writing it to flash offset `0x300000`. `0x300000` is the offset of ucode in the QSPI flash. Then use following commands to program the ucode to the boot device:

```
=>tftp 83000000 <ucode_file_name>.bin
=>sf probe 0:1
=>sf erase 300000 +$filesize
=>sf write 83000000 300000 $filesize
```

NOTE

Using "sf probe 0:0" could program microcode to the current bank.

4.2 Other Options

Once the board is set-up and configured appropriately, the following sections will step you through alternative deployment methods.

4.2.1 Build U-Boot and RCW without Yocto Project and Program to QSPI Flash

Execute the commands below to program a new U-Boot and RCW to QSPI boot without using Yocto Project:

1. Compile QSPI boot image (enable QSPI):

```
$make distclean ARCH=aarch64 CROSS_COMPILE=${toolchain_path}/gcc-linaro-aarch64-linux-
gnu-4.9-2014.07_linux/bin/aarch64-linux-gnu-
$make ARCH=aarch64  $ls1046ardb_qspi_defconfig
$make CROSS_COMPILE=${toolchain_path}/gcc-linaro-aarch64-linux-gnu-4.9-2014.07_linux/bin/
aarch64-linux-gnu- -j4
```

2. Swap the bytes for QSPI boot:

```
$tclsh byte_swap.tcl rcw_1600_qspiboot.bin rcw_1600_qspiboot_swap.bin 8
$tclsh byte_swap.tcl u-boot-dtb.bin u-boot_swap.bin 8
```

   The `byte_swap.tcl` script is a shareable tool and can be found under `rcw/ls1046ardb/` directory.

3. Write U-Boot images to QSPI flash 1 (altbank is bank4) under QSPI:

```
=>sf probe 0:1
```

   SF: Detected S25FL512S_256K with page size 256 Bytes, erase size 256 KiB, total 64 MiB

```
=>tftp 81000000 rcw_1600_qspiboot_swap.bin
=>sf erase 0 +$filesize
=>sf write 81000000 0 $filesize
=>tftp 82000000 u-boot_swap.bin
=>sf erase 100000 +$filesize;sf write 82000000 100000 $filesize
```

4. Switch to QSPI altbank:

```
=>cpld reset altbank
```

   Or set switches referring to board configurations Setup Board Switch Settings (Step 2) and power on the board from QSPI boot.

4.2.2 Program All Images to SD Card

4.2.2.1 Program U-Boot to SD Card (SD Boot)

To program the U-boot image, first boot the board to U-Boot. Next, load the new U-Boot SD boot image to RAM by downloading it via TFTP and then copying it to SD card with blk offset `0x8`. Execute the following commands at the U-Boot prompt to program the U-Boot image to SD card and reset to SD boot.

```
=>tftp 82000000 u-boot-sdcard.bin
=>mmc erase 8 0x800
```

```
=>mmc write 0x82000000 8 0x800
=>cpld reset sd
```

Program the image to SD card in Linux.

```
dd if=u-boot-with-spl-pbl-sd.bin of=/dev/sdb bs=512 seek=8
```

4.2.2.2 Program FMan Ucode to SD card

Execute the commands below to program a new microcode to SD Card:

```
=>tftp 83000000 <ucode_file_name>.bin
=>mmc write 83000000 820 50
```

5 Boot up Linux (Step 4)

5.1 Program Kernel itb from TFTP

1. **Set U-Boot Environment**

   Before performing FIT image deployment, configure U-Boot environment variables. See Configuring U-Boot Network Parameters on page 11 for more information on how to set U-Boot environment variables. In addition, execute the following commands at the U-Boot prompt to prepare for FIT image deployment from TFTP:

   ```
   =>setenv bootargs 'root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500 console=ttyS0,115200'
   =>saveenv
   ```

2. **Boot Up the System**

   Execute the following commands to TFTP the image to the board, then boot into Linux:

   ```
   =>tftp a0000000 <fit_image_name>
   =>bootm a0000000
   ```

   The board will now boot into Linux .

5.2 Program Kernel itb from Flash

1. **Set U-Boot Environment**

   Before performing FIT image from flash, configure U-Boot environment variables. See Configuring U-Boot Network Parameters for more information on how to set U-Boot environment variables. In addition, execute the following commands at the U-Boot prompt to prepare for deployment from flash:

   ```
   =>setenv bootargs root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500 console=ttyS0,115200
   =>setenv bootcmd sf probe 0:0;sf read <fit_image_addr> 1000000 2800000;bootm <fit_image_addr>
   =>saveenv
   ```

2. **Program FIT image to QSPI Flash**

The FIT image should be downloaded to the RAM address using TFTP then copied to flash offset `0x1000000` as per "QSPI memory map" in Flash Bank Usage. At the U-Boot prompt, use the following commands to program the image to QSPI current bank:

```
=>tftp <fit_image_addr> <fit_image_name>
=>sf probe 0:0
=>sf erase 0x1000000 +$filesize
=>sf write <fit_image_addr> 0x1000000 $filesize
```

3. **Boot Up the System**

   The kernel can boot up automatically after the board is powered on, or the following command can be used to boot up the board at U-Boot prompt:

```
=>boot
```

   or

```
=> bootm <fit_image_addr>
```

5.3 Program kernel itb to SD card

Execute the commands below to program the kernel itb to the SD card:

1. Insert SD card into the Linux Host PC.
2. Create temp directory in host PC and mount the ext2 partition to the temp

```
#mkdir temp
#mount /dev/sdb1 temp
```

3. Copy the FIT Kernel Image to the SD card partition.

```
#cp  kernel.itb temp/
```

4. Copy the Root File System to the SD card partition.

```
#cp fsl-image-core-ls1046ardb_<release date>.rootfs.tar.gz  temp/
#tar xvfz fsl-image-core-ls1046ardb_<release date>.rootfs.tar.gz
#rm  fsl-image-core-ls1046ardb_<release date>.rootfs.tar.gz  temp/
```

5. Umount the temp directory

```
#umount temp
```

6 Additional Reference

The following sections are additional references for the default and alternative boot methods.

6.1 Configuring U-Boot Network Parameters

To support TFTP based deployments, set up the U-Boot environment once, and save it, so that settings persist on subsequent resets.

```
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv ethaddr <mac addr0>
=>setenv eth1addr <mac addr1>
=>setenv eth2addr <mac addr2>
=>setenv eth3addr <mac addr3>
=>setenv ethprime <ethx>
=>setenv ethact <ethx>
=>setenv netmask 255.255.x.x
=>saveenv
```

NOTE

* <ethx> is the Ethernet port on the board connected to the Linux boot server. "netmask" is subnet mask for the Linux boot server's network.

Below is one example of the MAC address configuration corresponding to the set up above. Change these values to approriate MAC addresses for your board.

```
=>setenv ethaddr 00:e0:0c:00:89:00
=>setenv eth1addr 00:e0:0c:00:89:01
=>setenv eth2addr 00:e0:0c:00:89:02
=>setenv eth3addr 00:e0:0c:00:89:03
=>saveenv
```

NOTE

1. For boards with more network interfaces, additional environment variables need to be set (e.g., eth6addr, eth7addr,...).
2. In the overwhelming majority of cases, eth<*>addr can be autoset.

The flashed version of U-Boot is now ready for TFTP based deployments.

6.2 RCW (Reset Configuration Word) and Ethernet Interfaces

The RCW directories' names conform to the following naming convention:

```
ab_cdef_ghij
```

Table 3. RCW directories Naming Convention Legend

Slot	Convention

Table continues on the next page...

Table 3. RCW directories Naming Convention Legend (continued)

a	'R' indicates RGMII1@DTSEC3 is supported 'N' if not available/not used
b	'R' indicates RGMII2@DTSEC4 is supported 'N' if not available/not used
c	What is available in SerDes1 Lane 0
d	What is available in SerDes1 Lane 1
e	What is available in SerDes1 Lane 2
f	What is available in SerDes1 Lane 3
g	What is available in SerDes2 Lane 0
h	What is available in SerDes2 Lane 1
i	What is available in SerDes2 Lane 2
j	What is available in SerDes2 Lane 3

Table 4. For Lanes (C through H)

Flag	Convention
'N'	NULL, not available/not used
'P'	PCIe
'X'	XAUI
'S'	SGMII
'Q'	QSGMII
'F'	XFI
'H'	SATA
'A'	AURORA

For example,

```
RR_FFPPPN_1133_5559
```

means:

- RGMII1@DTSEC3 on board
- RGMII2@DTSEC4 on board
- XFI9 on Copper port
- XFI10 on SFP cage

- SGMII5 on SGMII1 port
- SGMII6 on SGMII2 port
- PCIe1 on miniPCIe slot
- PCIe2 on Slot 2
- PCIe3 on Slot 3
- SATA
- SERDES1 Protocol is 0x1133
- SERDES2 Protocol is 0x5559

The RCW file names for the ls1046ardb conform to the following naming convention:

```
rcw_<frequency>_<specialsetting>.rcw
```

Table 5. RCW Files Naming Convention Legend

Code		Convention
frequency		Core frequency(MHZ)
specialsetting	bootmode	QSPI/SD/EMMC
	special support	emmc: eMMC boot sdboot: SD boot sben: Secure boot qspiboot: QSPI boot

For example,

```
rcw_1600_sdboot.rcw means rcw for core frequency of 1600MHz with sd boot.
```

```
ls1046ardb/RR_FFPPPN_1133_5559/rcw_1600_qspiboot.rcw means rcw for core frequency 0f 1600MHz with
QSPI boot.
```

The following RCW binaries are used on the ls1046ardb:

```
RR_FFPPPN_1133_5559/rcw_1600_qspiboot.bin
```

```
RR_FFPPPN_1133_5559/rcw_1600_qspiboot_sben.bin
```

6.3 Flash Bank Usage

The table below shows the QSPI flash memory map for LS1046ARDB:

Table 6. QSPI Flash Memory Map for LS1046ARDB

Start	End Offset	Description	Size

Table continues on the next page...

Table 6. QSPI Flash Memory Map for LS1046ARDB (continued)

0x40000000	0x400FFFFF	bank0 rcw + pbi	1 M
0x40100000	0x401FFFFF	bank0 U-Boot image	1 M
0x40200000	0x402FFFFF	bank0 U-Boot Env	1 M
0x40300000	0x403FFFFF	bank0 Fman ucode	1 M
0x40400000	0x404FFFFF	bank0 UEFI	1 M
0x40500000	0x406FFFFF	bank0 Primary Protected Application (PPA)	2 M
0x40700000	0x408FFFFF	bank0 Secure boot header + bootscript	2 M
0x40900000	0x40FFFFFF	bank0 reserved	7 M
0x41000000	0x43FFFFFF	bank0 FIT Image	48 M
0x44000000	0x440FFFFF	bank4 rcw + pbi	1 M
0x44100000	0x441FFFFF	bank4 U-Boot image	1 M
0x44200000	0x442FFFFF	bank4 U-Boot Env	1 M
0x44300000	0x443FFFFF	bank4 Fman ucode	1 M
0x44400000	0x444FFFFF	bank4 UEFI	1 M
0x44500000	0x446FFFFF	bank4 PPA	2 M
0x44700000	0x448FFFFF	bank4 Secure boot header + bootscript	2 M
0x44900000	0x44FFFFFF	bank4 reserved	7 M
0x45000000	0x47FFFFFF	bank4 FIT Image	48 M

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

AN5340
Rev. 0
10/2016