System Boot from SD/MMC Card with SDK 1.6 images

This application note describes how to write u-boot in SD/MMC card with boot-format application for non-PBL platform, use QorIQ Configuration Suite (QCS) PBL tool to generate PBL images for Corenet platform SD boot, deploy rootfs filesystem to boot Kernel and filesystem from SD/MMC card.

1. Use Boot-format to Write U-boot into SD card

For Non PBL platforms, after the device has completed the reset sequence, if the ROM location selects the on chip ROM eSDHC Boot configuration, the e500 core starts to execute code from the internal on-chip ROM. The e500 core configures the eSDHC controller, enabling it to communicate with the external SD/MMC card. The SD/MMC card should contain a specific data structure with control words, device configuration information and initialization code. The on-chip ROM boot code uses the information from the SD/MMC card content to configure the device, and to copy the initialization code to a target memory device (for example, the DDR) through the eSDHC interface. After all the code has been copied, the e500 core starts to execute the code from the target memory device.

1.1 SD/MMC boot up Data Structure

The boot application boot-format implements writing the boot up code to MMC/SD cards.

For non PBL platforms SD/MMC Card Data Structure is often defined as the following.

*/*Offset Data Bits [0:31]*
0x00-0x3F Reserved
0x40-0x43 BOOT signature. This location should contain the value 0x424f_4f54,
0x44-0x47 Reserved
0x48-0x4B User's code length. Number of bytes in the user's code to be copied. It must be a multiple of the SD/MMC cards block size, It <= 2GBytes.
0x4C-0x4F Reserved
0x50-0x53 Source Address. Contains the starting address of the user's code as an offset from the SD/MMC card starting address. In Standard Capacity SD/MMC Cards, it specifies the memory address in byte address format. This must be a multiple of the SD/MMC cards block size. In High Capacity SD/MMC Cards (>2GByte), the Address specifies the memory address in block address format. Block length is fixed to 512 bytes as per the SD High Capacity specification.
0x54-0x57 Reserved
0x58-0x5B Target Address. Contains the target address in the system's local memory address space in which the user's code will be copied to.
0x5C-0x5F Reserved
0x60-0x63 Execution Starting Address. Contains the jump address in the system's local memory address space into the user's code first instruction to be executed.
0x64-0x67 Reserved
0x68-0x6B N. Number of Configuration Address/Data pairs. Must be 1<=N<=1024
0x6C-0x7F Reserved
0x80-0x83 Configuration Address 1
0x84-0x87 Configuration Data 1
0x88-0x8B Configuration Address 2
0x8C-0x8F Configuration Data 2
...

0x80 + 8(N-1) Configuration Address N*
0x80 + 8(N-1)+4 Configuration Data N*
...
User's Code

NOTE: N <= 40 (Offset can be from first page up to 24th page in a 512 block)
**/*

1.2 Use boot-format command

Usage of boot-format:

<path>/boot_format <config_file> <image> -sd <dev>

In SDK 1.6, after run “bitbake boot-format”, boot-format binary is deployed in <project>/tmp/sysroots/x86_64-linux/usr/bin/boot_format, configuration data is deployed in <project>/tmp/sysroots/x86_64-linux/usr/share/boot_format/.

Use P2020RDB as an example:

Create the partitions by "fdisk /dev/sdb", one MS-DOS partition (sdb1 and this partition usually for u-boot) and one ext2 partition(sdb2).

fdisk /dev/sdb

mkfs.vfat /dev/sdb1

<install_path>/build_p1020rdb_release/tmp/sysroots/x86_64-linux/usr/bin/boot_format \
<install_path>/build_p1020rdb_release/ tmp/sysroots/x86_64-linux/share/boot_format/config_sram_p1022ds.dat u-boot-sd-P2020RDB-PC_SDCARD.bin -sd \
/dev/sdb

Note: -sd should be specify the device name rather than partition name.

The configuration data can be got from machine configuration file meta-fsl-ppc/conf/machine/p2020rdb.conf.

BOOTFORMAT_CONFIG = "config_sram_p1022ds.dat"

2. Generating PBL images for Corenet platforms with QCS tool

The Corenet Soc integrates a pre-boot-loader(PBL) which performs configuration registers read and write to initialize external memory devices such as I2C, ESDHC, loads RCW and pre-boot initialization commands from those devices before the local cores are permitted to boot.

2.1 Corenet platforms SD boot procedure

SD boot up process can be divided into two stages.

PBL loads boot image from SD, the boot Image contains three parts, the first is RCW, the second is PBI commands, the third is u-boot image loaded to CPC.

U-boot will run in CPC and then boot up to U-boot command line.

For T4240/T2080/T1040, the 512KB u-boot size is 768KB, but CPC is 512KB not big enough to store u-boot, two stage u-boot images are introduced in SDK 1.6, the first stage u-boot is loaded

to CPC, which will init DDR and load the second stage u-boot to DDR, then jump to DDR to boot up to u-boot prompt.

2.2 Building SD u-boot images with Yocto

Generate SD u-boot as the following in Yocto build environment.

a. $ cd build_<platform>_release
b. $ source ./SOURCE_THIS
c. $ bitbake -c cleansstate u-boot
d. Configure SD u-boot configuration in machine configuration file, for example for T1040RDB. In the file meta-fsl-ppc/conf/machine/t1040rdb.conf, add T1040RDB_SDCARD as the following.
   UBOOT_MACHINES ?= "T1040RDB T1040RDB_SDCARD"
e. $ bitbake u-boot
   Get SD u-boot in build_<platform>_release/tmp/deploy/images/<platform>/

For T4240/T2080/T1040 platforms, the two stage u-boot images u-boot.bin and u-boot-spl.bin can be found in u-boot build directory.

For example for T1040, u-boot images are in build_t1040rdb_release/tmp/work/t1040rdb-fsl-linux/u-boot/2014.01+fslgit-r0/git/T1040RDB_SDCARD/.

2.3 Generate PBL images with QCS Tool

QorIQ Configuration Tool can be downloaded from http://www.freescale.com/QCS.

Create and modify QCS project as the following.

A. Following new project wizards to create a project
   PBL - Preboot Loader RCW Configuration Component
   select the proposed RCW Hard-coded configuration
B. Once the project is created, open a Component Inspector window for the PBL1:PBL component
C. In the **Import** tab:
   Select the Binary input format
   Hit the Browse button to choose a QorIQ SDK release pre-build RCW
   Hit the Import button
   The message "Successfully Imported" appears, if the RCW contains any PBI data, it will also be imported.
D. In the Properties tab change the following properties:
   Reset Configuration Word (RCW)->RCW Source
   Boot Configuration->PBI_SRC = 0b0110 - SD/MMC
   Boot Configuration->BOOT_LOC = 0b10000 - Memory Complex 1
   Note: for T208xRDB BOOT_LOC need to set to 0b11000 - IFC.
E. Putting together the PBI data
   Go to the **PBI Data input -> PBI Data input** property
   Click on the Value field then on the ellipsis dots [...] button on the right
   The PBI Data Input panel is now shown, if the RCW imported above has any PBI commands, they will be shown in the panel;
   Append the common PBI commands below into the text panel:
   PBI Commands (Lines started with '#' is a comment):
   Below are common PBI commands for P-series SoCs which have 1MB CPC.
   #Clear CPC1
   09010000 00200400

09138000 00000000
091380c0 00000100
#Initialize CPC1 as 1MB SRAM
09010100 00000000
09010104 fff0000b
09010f00 08000000
09010000 80000000
#Configure LAW for CPC1
09000d00 00000000
09000d04 fff00000
09000d08 81000013
#Initialize eSPI controller
09110000 80000403
09110020 2d170008
09110024 00100008
09110028 00100008
0911002c 00100008
#Configure alternate space
09000010 00000000
09000014 ff000000
09000018 81000000
#Flush data
09138000 00000000
091380c0 00000000
For T-/B-series SoCs (with 512 KB CPC SRAM):
#Initialize CPC1
09010000 00200400
09138000 00000000
091380c0 00000100
#512KB SRAM
09010100 00000000
09010104 fff80009
09010f00 08000000
#enable CPC1
09010000 80000000
#Configure LAW for CPC1
09000d00 00000000
09000d04 fff80000
09000d08 81000012
#Initialize eSPI controller
09110000 80000403
09110020 2d170008
09110024 00100008
09110028 00100008
0911002c 00100008
#Configure alternate space
09000010 00000000
09000014 ff000000
09000018 81000000
#Flush PBL data
09138000 00000000
091380c0 00000000

Hit the Apply button

F. Insert the u-boot image file:

Select ACS File (data from binary file) then file Browse appears.
For P-series SoCs:
Change Offset: 0xf40000
Browse to File: u-boot-sd.bin

For T4240/T2080/T1040 SoCs:
Change offset: 0xfd000
Browse to File: u-boot-spl.bin
Hit the Add button, then the Apply button

G. Select and click Add for these 2 PBI commands:
Flush
Wait
Hit the Apply button

H. In Properties tab goto PBL data->Output Format property, click value field and select Binary.
From the QCS menu, click on Project - > Generate Processor Expert Code button.
The generated file can be found in the QCS project under : Generated Code -> PBL1.bin

2.4 Deploy SD u-boot images on the target

For P-series SoCs:

=>tftp 100000 PBL1.bin
=>mmcinfo
=>mmc write 100000 8 block_number

=>tftp 100000 ucode.bin

=>mmc write 100000 690 block_number

For T2080/T4240/T1040

=>tftp 100000 PBL1.bin

=>mmcinfo

=>mmc write 100000 8 block_number

=>tftp 100000 u-boot.bin
=>mmc write 100000 208 block_number (for T4240QDS the offset is 0x200)

=>tftp 100000 ucode.bin
=>mmc write 100000 820 block_number

3. Deploy Rootfs to boot Kernel and filesystem from SD/MMC

1. Use the mkfs.ext2 command to create the filesystems.

   # mkfs.ext2 /dev/sdb2

2. Copy the file system to SD card by extracting the QorIQ_SDK_V1.6_<core>_<date>_ROOTFS_Image.tar.gz from rebuild image ISO, remove the tarball after extracting rootfs.
   # cp QorIQ_SDK_V1.6_<core>_<date>_ROOTFS_Image.tar.gz .

 # tar -zxvf _SDK_V1.6_<core>_<date>_ROOTFS_Image.tar.gz

 # rm _SDK_V1.6_<core>_<date>_ROOTFS_Image.tar.gz

3. Plug in the SD card to the target board and power on.
4. Set the environment in uboot for SD boot.
   # setenv bootfile uImage
   # setenv fdtfile uImage.dtb
   # setenv bootcmd 'setenv bootargs root=/dev/mmcblk0p2 rw rootdelay=5 console=$consoledev,$baudrate;mmcinfo;ext2load mmc 0:2 $loadaddr /boot/$bootfile;ext2load mmc 0:2 $fdtaddr /boot/$fdtfile;bootm $loadaddr - $fdtaddr'
   # boot