Bring up the LS1046A

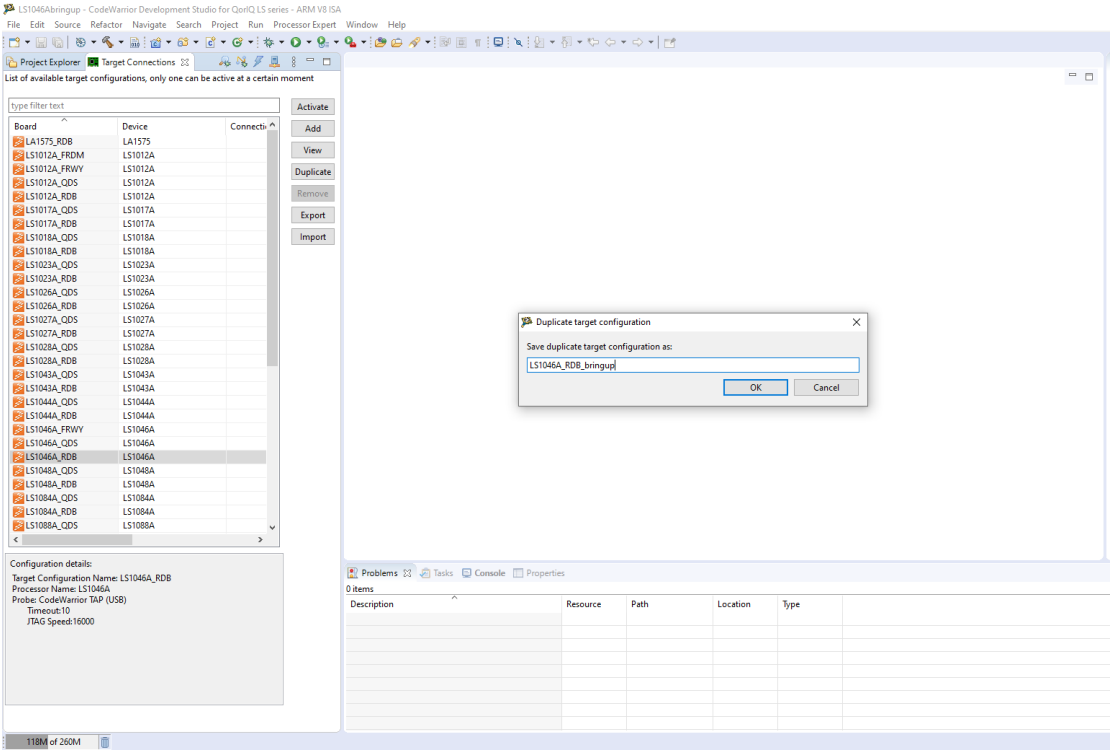
How to bring up a card when the flash is blank or the image is corrupted. How to boot cards from various boot mode when changed the RCW as requirements. This documentation will use LS1046ARDB as new board to realize the functions(all target board in the document is LS1046ARDB).

Bring up LS1046A with CodeWarrior TAP

This topic describes the steps required to perform board recovery using CodeWarrior Development Studio for LS1046ARDB, when the flash is blank or the image is corrupted.

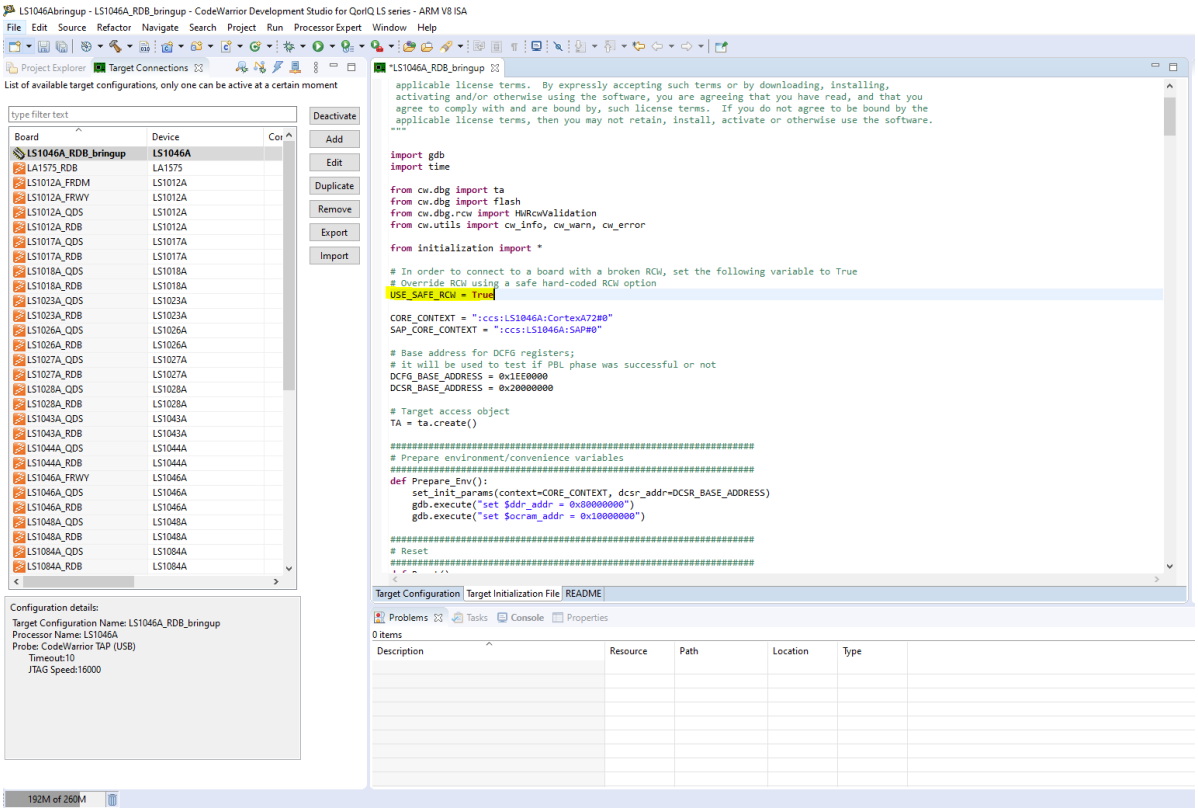
When RCW is missing or corrupted and switches for setting the board in the hard-coded RCW mode are not available, you can use the RCW override feature available with the CodeWarrior software for board recovery. For using this feature, perform these steps:

1. Open CodeWarrior for LS1046ARDB and define a Target connection configuration for the board using a pre-defined configuration.

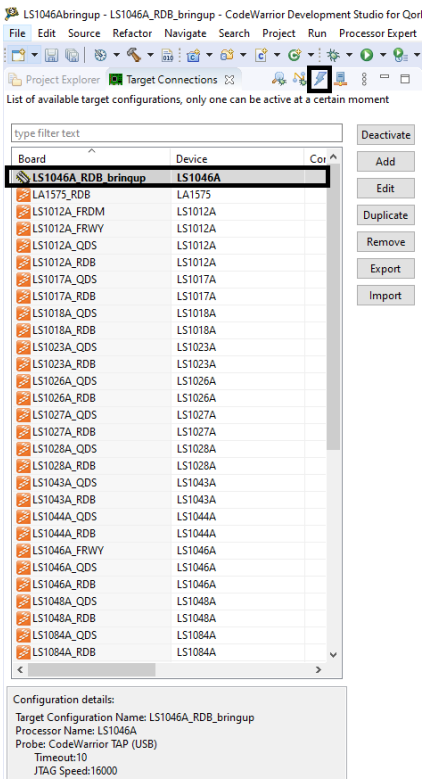


2. Select the LS1046A_RDB, click the right button select “Duplicate”, input the name in popup windows.

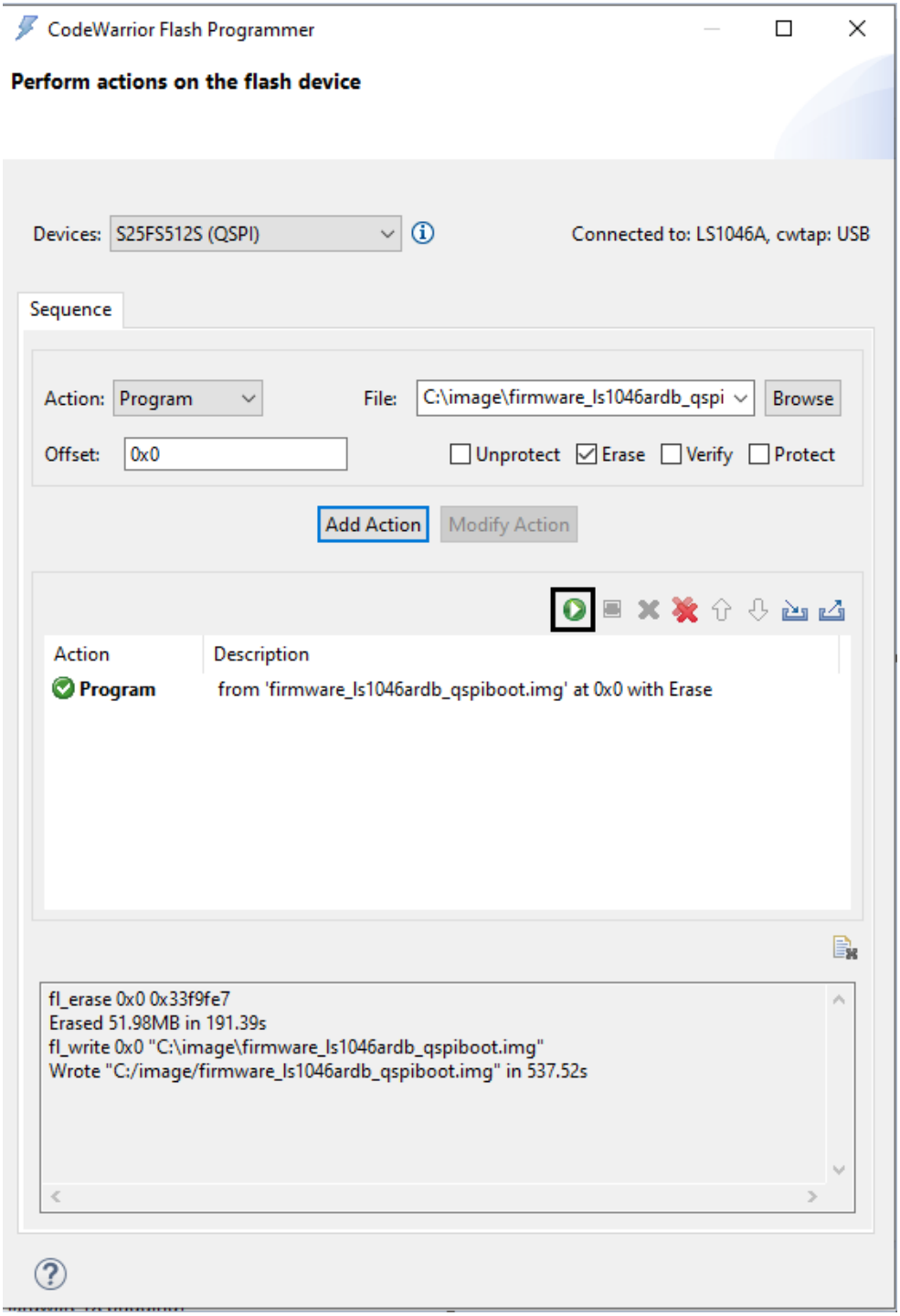
3. In Target Configuration tab, select the probe type.
4. Click the Target Init File tab and search for USE_SAFE_RCW. To connect to the board with a missing or corrupted RCW, set the USE_SAFE_RCW variable to True.



5. Click OK to save the configuration.
6. Click the Flash Programmer button to connect to the board.



7. When the CodeWarrior software connects to the board, the CodeWarrior Flash Programmer window appears.
8. Select the appropriate flash device, the Program action, the RCW file to be programmed and the Offset.
9. Click Add Action and execute the flash programmer sequence.



10. Select the switches according to the firmware place(QSPI, in the step 9), the LS1046ARDB can bring up.

When change the RCW as requirement, how to generate the firmware of different boot modes. Content below will list the ways to boot from all supported boot modes of LS1046ARDB, including SD card, QSPI and eMMC.

Please setup the host machine environment in the Ubuntu 20.04 LTS:

$ tar xf flexbuild_lsdk2108.tar

$ cd flexbuild_lsdk2108/

$ source setup.env

$ flex-builder -h

Various compile path below, you can choose any one of them.

Boot up from the SD card

Compile PBL binary from RCW source file

In this use case, the requirement is changing the SPI_EXT from 1 to 0.

In the host machine:

1. $ git clone https://source.codeaurora.org/external/qoriq/qoriq-components/rcw
2. $ cd rcw
3. $ git checkout -b LSDK-21.08 LSDK-21.08
4. $ cd ls1046ardb/ RR_FFSSPPPH_1133_5559

Change the SPI_EXT from 1 to 0 in rcw_1800_sdboot.rcw

5. $ cd ls1046ardb
6. $../rcw.py -i RR_FFSSPPPH_1133_5559/rcw_1800_sdboot.rcw -o rcw_1800_sdboot.bin

Now, you will get a new rcw_1800_sdboot.bin of RCW as the requirement.

Compile the PBL binary into firmware

Put the rcw_1800_sdboot.bin */flexbuild_lsdk2108/build/rcwnew

Change the */flexbuild_lsdk2108/configs/board/ls1046ardb/manifest select rcw_sd,

rcw_sd="rcwnew/rcw_1800_sdboot.bin"(make sure the path is the required rcw_1800_sdboot.bin)

7. $ cd flexbuild_lsdk2108/
8. $ source setup.env
9. $ flex-builder -i clean-firmware
10. $ flex-builder -c atf -m ls1046ardb -b sd
11. $ flex-builder -i mkfw -m ls1046ardb -b sd

Now the required image is available in*/flexbuild_lsdk2108/build/images/firmware_ls1046ardb_sd.img

Put the firmware_ls1046ardb_sd.img into tftp-server file.

Program the firmware into the target board(LS1046ARDB)

In the target board, to partition and format target storage device with specified number and size of partitions instead of using the default

```
12. $ flex-installer -i pf -p 4P=128M:2G:8G:-1 -d /dev/mmcblk0
```

Use the command fdisk -l to check the partitions list as required.

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/mmcblk0p1		139264	401407	262144	128M	83	Linux
/dev/mmcblk0p2		403456	4597759	4194304	2G	83	Linux
/dev/mmcblk0p3		4599808	21377023	16777216	8G	83	Linux
/dev/mmcblk0p4		21379072	30253055	8873984	4.2G	83	Linux

Reboot the target board(LS1046ARDB), and in the U-Boot prompt:

Set the target board IP address correctly to make sure it can connect with server IP.

```
13. => dhcp
14. => setenv serverip 192.168.3.54(my tftp-server ip: 192.168.3.54 )
15. => saveenv
```

Download the firmware_ls1046ardb_sdboot.img from the server to the target board, program it into the SD card.

```
16. => tftp $load_addr firmware_ls1046ardb_sdboot.img
    (Bytes transferred = 54499987 (33f9a93 hex))
17. => mmc dev 0; mmc write $load_addr 8 0x33f9a93(0x33f9a93 file received in the tftp command)
```

Power off the board and put the switch to the SD card, the board will boot up with the required firmware.

Boot up from the QSPI

Compile firmware from RCW source file

In this use case, the requirement is changing the SPI_EXT from 1 to 0.

In the host machine

Change the SPI_EXT from 1 to 0 in the */flexbuild_lsdk2108/flexbuild_lsdk2108/components/firmware/rcw/ls1046ardb/rcw_1800_qspiboot.rcw

```
1.  $ cd flexbuild_lsdk2108/
```

2. $ source setup.env
3. $ flex-builder -i clean-firmware
4. $ flex-builder -c atf -m ls1046ardb -b qspi
5. $ flex-builder -i mkfw -m ls1046ardb -b qspi

Now the required image is available in */flexbuild_lsdk2108/build/images/firmware_ls1046ardb_qspiboot.img

Put the firmware_ls1046ardb_qspiboot.img into tftp-server file.

Program the firmware into the target board(LS1046ARDB)

In the target board

Boot the LS1046ARDB into the U-Boot

6. => dhcp fm1-mac5(Now I use the eth2 to download from tftp-server, my tftp-server ip: 192.168.3.48)
7. => setenv serverip 192.168.3.48
8. => saveenv
9. => tftp $load_addr firmware_ls1046ardb_qspiboot.img
10. => sf probe 0:1
11. => sf erase 0 +$filesize && sf write $load_addr 0 $filesize

Power off the board and put the switch to the another QSPI flash, the board will boot up with the required firmware.

Boot up from the eMMC

Enable the on board eMMC

On the LS1046ARDB, eSDHC can either be connected to a secure digital (SD) card socket or an eMMC flash memory. SD card is default to be connected. So need to change the RCW to make the U-Boot can recognize the eMMC.

Change the */flexbuild_lsdk2108/configs/board/ls1046ardb/manifest select rcw_qspi,

rcw_qspi="firmware/rcw/ls1046ardb/RR_FFSSPPPH_1133_5559/rcw_1600_qspiboot_emmc.bin"(enable the eMMC on board)

1. $ cd flexbuild_lsdk2108/
2. $ source setup.env
3. $ flex-builder -i clean-firmware
4. $ flex-builder -c atf -m ls1046ardb -b qspi

Now the required bl2_qspi.pbl is available in firmware/atf/ls1046ardb/bl2_qspi.pbl

Put the firmware_ls1046ardb_sd.img into tftp-server file.

In the target board, put the switch and boot from QSPI NOR flash0

Boot the LS1046ARDB into the U-Boot

```
=> dhcp fm1-mac5(Now I use the eth2 to download from tftp-server, my tftp-server ip: 192.168.3.48 )
=> setenv serverip 192.168.3.48
=> saveenv
=> sf probe 0:1
=> tftp 0xa0000000 bl2_qspi.pbl
=> sf erase 0x0 +$filesize && sf write 0xa0000000 0x0 $filesize
```

Put the switch and boot from QSPI NOR flash1

Now the eMMC is ready(SD card is disabled).

Compile firmware from RCW source file

In the host machine

Change the SPI_EXT from 1 to 0 in the */flexbuild_lsdk2108/flexbuild_lsdk2108/components/firmware/rcw/ls1046ardb/rcw_1800_emmcboot.rcw

```
$ cd flexbuild_lsdk2108/
$ source setup.env
$ flex-builder -i clean-firmware
$ flex-builder -c atf -m ls1046ardb -b emmc
$ flex-builder -i mkfw -m ls1046ardb -b emmc
```

Now the required image is available in */flexbuild_lsdk2108/build/images/firmware_ls1046ardb_emmcboot.img.

Put the firmware_ls1046ardb_emmcboot.img into tftp-server file.

Partition and format target storage device with specified number and size of partitions instead of using the default

```
$ flex-installer -i pf -p 4P=128M:1G:2G:-1 -d /dev/mmcblk0
```

Number	Start	End	Size	Type	File system	Flags
1	71.3MB	206MB	134MB	primary	ext4	
2	207MB	1280MB	1074MB	primary	ext4	
3	1281MB	3429MB	2147MB	primary	ext4	
4	3430MB	3909MB	479MB	primary	ext4	

Program the firmware into the target board(LS1046ARDB)

Reboot the target board(LS1046ARDB), and in the U-Boot prompt:

Set the target board IP address correctly to make sure it can connect with server IP.

```
17. => dhcp
18. => setenv serverip 192.168.3.54
19. => saveenv
```

Download the firmware_ls1046ardb_emmcboot.img from the server to the target board, program it into the eMMC.

```
20. => tftp $load_addr firmware_ls1046ardb_emmcboot.img
    (Bytes transferred = 54499987 (33f9a93 hex))
21. => mmc dev 0; mmc write $load_addr 8 0x33f9a93(0x33f9a93 file received in the tftp command)
```

Power off the board and put the switch to the eMMC/SD card, the board will boot up with the required firmware.

Please see LS1046ARDBRM.pdf page 34 Table 19 for switch to various BOOMODE.