Freescale Semiconductor
Application Note

Document Number: AN5184

Programming eMMC/SD Card using CodeWarrior for ARMv7

1. Introduction

This document describes how to program a U-Boot image into an eMMC/SD card using CodeWarrior for QorIQ LS series for ARM v7 ISA.

This document provides:

- Introduction to eMMC/SD card architecture
- Steps to program a U-Boot image into an eMMC/SD card for board recovery

Contents

1. Introduction..1
2. Introduction to eMMC/SD card architecture...2
3. Programming a U-Boot image into an eMMC/SD card...2
4. Board recovery 4

2. Introduction to eMMC/SD card architecture

The eMMC/SD card uses the notion of blocks, not addresses. One block has 512 bits. The CodeWarrior flash programmer only allows addresses, it means, to write a U-Boot image into an eMMC/SD card, you need to compute the address by multiplying the number of the block with 512 and converting the result into hexadecimal.

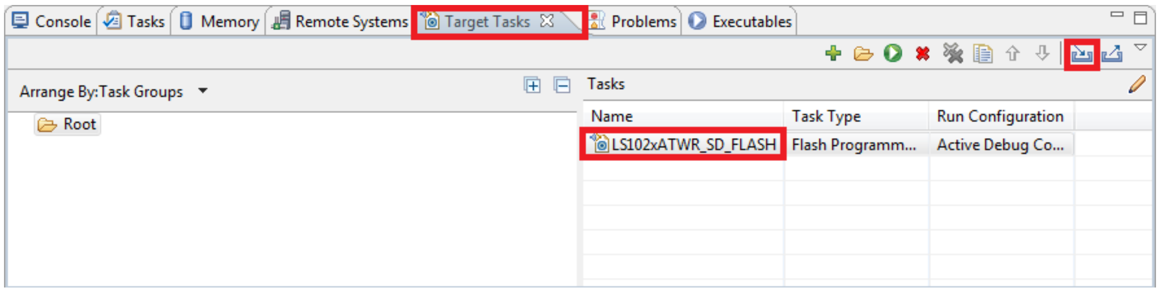
For more information, see *CodeWarrior for ARMv7 Targeting Manual* (CW_ARMv7_Targeting_Manual).

3. Programming a U-Boot image into an eMMC/SD card

To program a U-Boot image into an eMMC/SD card using CodeWarrior flash programmer, perform these steps:

1. Create a CodeWarrior for ARMv7 bareboard project.
2. Choose **Run** > **Debug** from the CodeWarrior IDE menu bar to start a new debug session.
3. Choose **Window** > **Show View** > **Other** from the CodeWarrior IDE menu bar. The **Show View** dialog appears.
4. Choose **Debug** > **Target Tasks** and click **OK**. The **Show View** dialog closes and the **Target Tasks** view appears.
5. Import the eMMC/SD card task using the **Target Tasks** view, as shown in the figure below.

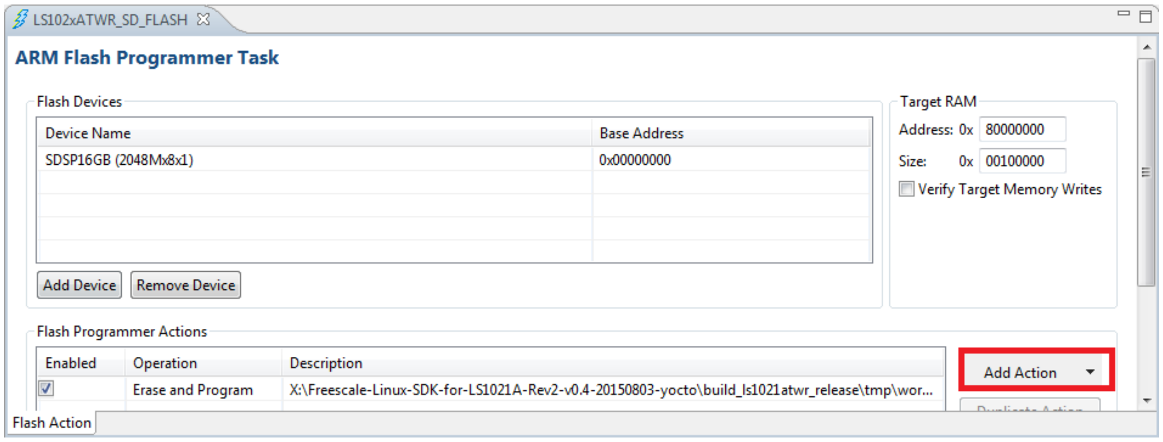
Figure 1. Target Tasks view



Name	Task Type	Run Configuration
LS102xATWR_SD_FLASH	Flash Programm...	Active Debug Co...

6. Click the **Edit Task Configuration** icon in the **Target Tasks** view. The **ARM Flash Programmer Task** editor window appears.
7. Choose **Program/Verify Action** from the **Add Action** menu (see figure below). The **Add Program / Verify Action** dialog appears.

Figure 2. ARM Flash Programmer Task editor window



Flash Devices

Device Name	Base Address
SDSP16GB (2048Mx8x1)	0x00000000

Target RAM

Address: 0x 80000000

Size: 0x 00100000

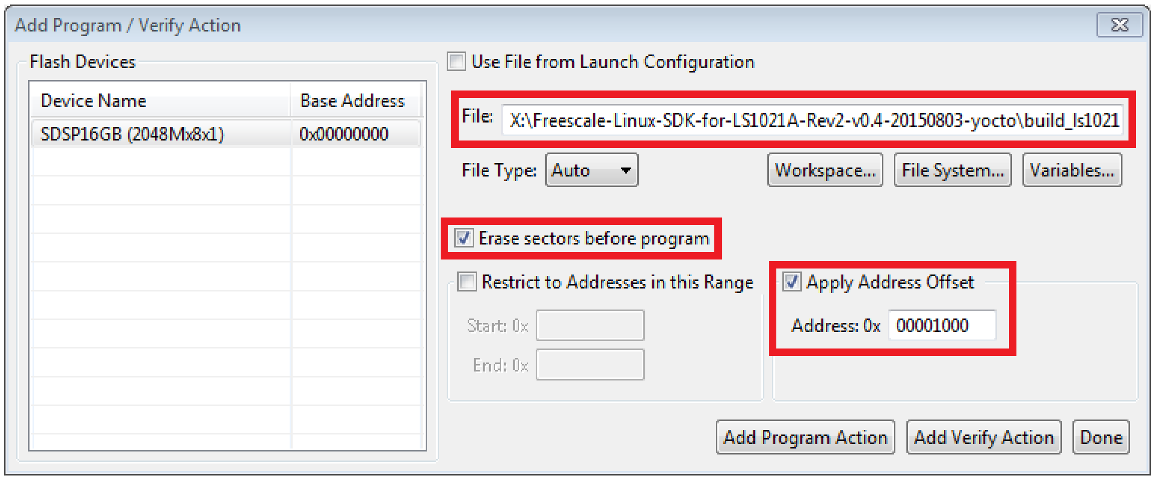
☐ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
✓	Erase and Program	X:\Freescale-Linux-SDK-for-LS1021A-Rev2-v0.4-20150803-yocto\build_ls1021atwr_release\tmp\wor...

8. In the **Add Program / Verify Action** dialog, specify the path to U-Boot image, select the **Erase sectors before program** and **Apply Address Offset** checkboxes, and set the address offset, as shown in the figure below.

Figure 3. Add Program / Verify Action dialog



Flash Devices

Device Name	Base Address
SDSP16GB (2048Mx8x1)	0x00000000

☐ Use File from Launch Configuration

File: X:\Freescale-Linux-SDK-for-LS1021A-Rev2-v0.4-20150803-yocto\build_ls1021

File Type: Auto

☑ Erase sectors before program

☐ Restrict to Addresses in this Range

Start: 0x

End: 0x

☑ Apply Address Offset

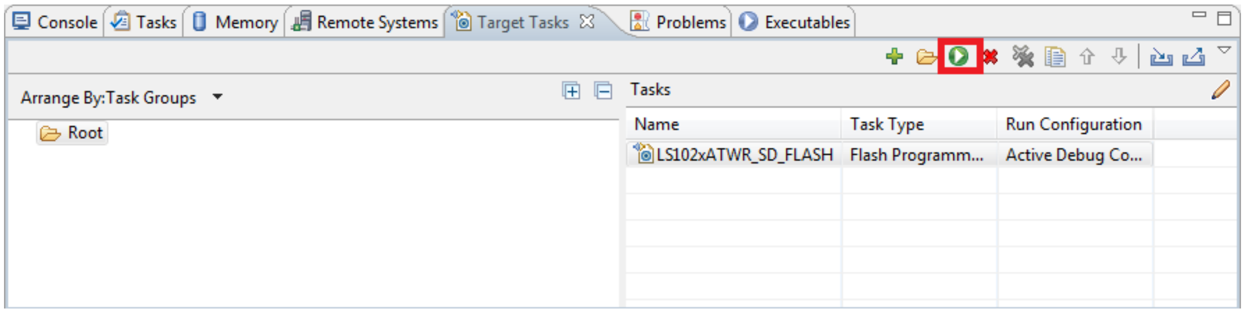
Address: 0x 00001000

NOTE The address offset is computed by multiplying the number of the block with 512 and converting the result into hexadecimal.
The U-Boot image for LS1 targets is written from block 8. In this case, the address offset will be: `8 x 512 = 4096`, which means `0x00001000` in hexadecimal.

---

9. Click **Add Program Action**, and then click **Done** to close the **Add Program / Verify Action** dialog.
10. Ensure that a valid eMMC/SD card is available on the board, and execute the target task, as shown in the figure below.

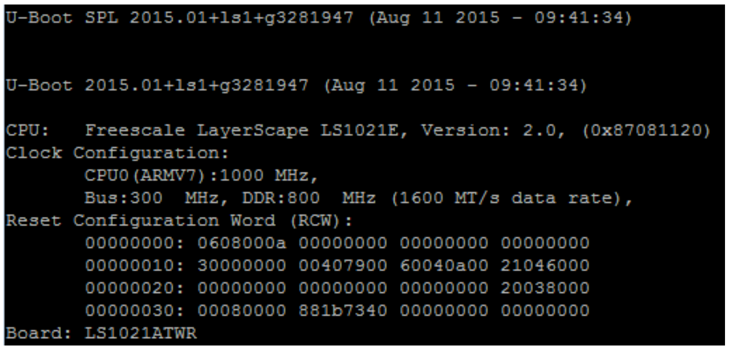
Figure 4. Executing target task



Name	Task Type	Run Configuration
LS102xATWR_SD_FLASH	Flash Programm...	Active Debug Co...

After the target task is executed, U-Boot will be ready to boot (see figure below).

Figure 5. Output window



```
U-Boot SPL 2015.01+ls1+g3281947 (Aug 11 2015 - 09:41:34)


U-Boot 2015.01+ls1+g3281947 (Aug 11 2015 - 09:41:34)

CPU:   Freescale LayerScape LS1021E, Version: 2.0, (0x87081120)
Clock Configuration:
       CPU0(ARMV7):1000 MHz,
       Bus:300  MHz, DDR:800  MHz (1600 MT/s data rate),
Reset Configuration Word (RCW):
       00000000: 0608000a 00000000 00000000 00000000
       00000010: 30000000 00407900 60040a00 21046000
       00000020: 00000000 00000000 00000000 20038000
       00000030: 00080000 881b7340 00000000 00000000
Board: LS1021ATWR
```

4. Board recovery

This section explains how to perform board recovery. The section is divided into the following subsections:

- RCW/U-Boot recovery
- Complete board recovery

4.1. RCW/U-Boot recovery

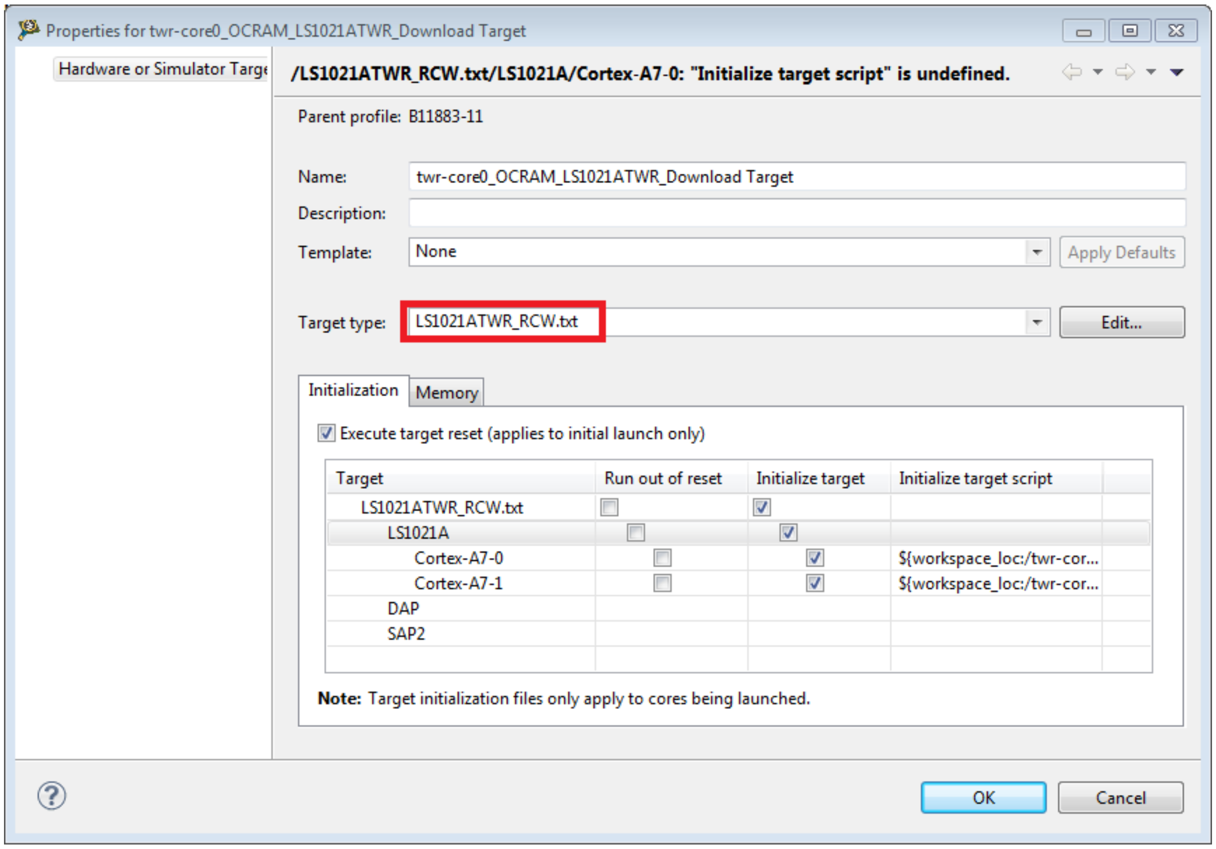
If eMMC/SD card boot is the only boot option available for the board, then you should use the CodeWarrior reset configuration word (RCW) override feature to recover the board.

Perform these steps to recover the board:

1. Create a CodeWarrior for ARMv7 bareboard project. If the double data rate (DDR) memory is not functional, then create the project by selecting **Download OCRAM** as the launch configuration.
2. Choose **Run > Debug Configurations** from the CodeWarrior IDE menu bar. The **Debug Configurations** dialog appears.
3. Click **Edit** next to the **Connection** menu in the **Target Settings** panel. The **Properties for <*connection launch configuration*>** window appears.

4. Click **Edit** next to the **Target** menu. The **Properties for <*connection launch configuration*> Target** window appears.
5. Specify a JTAG configuration file for RCW override in the **Target type** menu by clicking the **Edit** button next to this menu, as shown in the figure below.

Figure 6. Selecting JTAG configuration file for RCW override


Properties for twr-core0_OCRAM_LS1021ATWR_Download Target

/LS1021ATWR_RCW.txt/LS1021A/Cortex-A7-0: "Initialize target script" is undefined.

Parent profile: B11883-11

Name: twr-core0_OCRAM_LS1021ATWR_Download Target

Description:

Template: None

Target type: LS1021ATWR_RCW.txt

Execute target reset (applies to initial launch only)

Target	Run out of reset	Initialize target	Initialize target script
LS1021ATWR_RCW.txt	☐	✓	
LS1021A	☐	✓	
Cortex-A7-0	☐	✓	${workspace_loc:/twr-cor...
Cortex-A7-1	☐	✓	${workspace_loc:/twr-cor...
DAP			
SAP2			

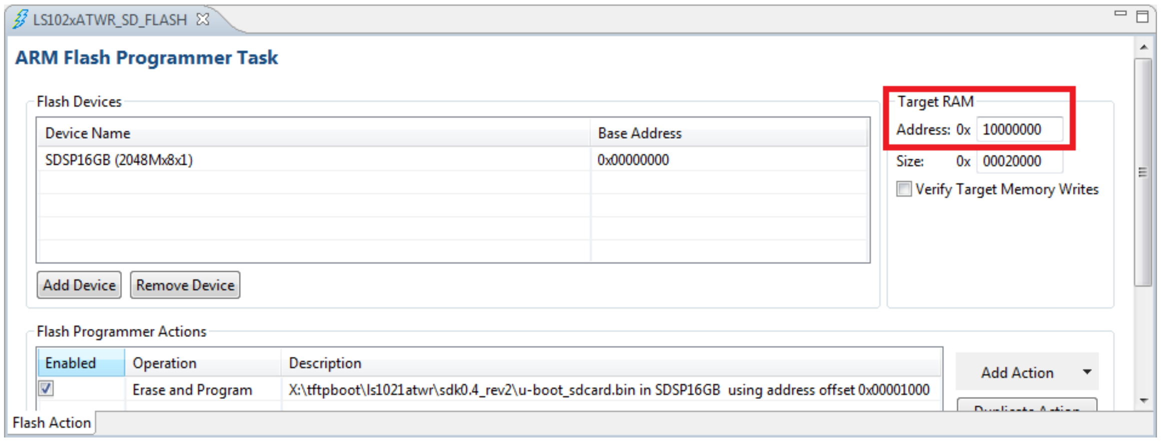
Note: Target initialization files only apply to cores being launched.

NOTE The JTAG configuration file for RCW override contains valid values for RCW registers and it allows CodeWarrior to connect to the board.

6. Select OCRAM file as the initialization file for the new target type on the **Initialization** page.
7. Select memory initialization file for the target type on the **Memory** page.
8. Click **OK** to close the **Properties for <*connection launch configuration*> Target** window.
9. Click **OK** to close the **Properties for <*connection launch configuration*>** window.
10. Click **Debug** in the **Debug Configurations** dialog to debug the project.
11. Open the flash programmer target task in the **ARM Flash Programmer Task** editor window and change the running address of the algorithm with the OCRAM address of the processor, as shown in the figure below.

NOTE Using OCRAM address for flash programmer algorithm enables the internal buffer of the eSDHC controller. However, this reduces the performance as direct memory access (DMA) is not used in this mode.

Figure 7. Changing flash programmer algorithm address



ARM Flash Programmer Task

Flash Devices

Device Name	Base Address
SDSP16GB (2048Mx8x1)	0x00000000

Target RAM

Address: 0x 10000000

Size: 0x 00020000

☐ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
✓	Erase and Program	X:\tftpboot\ls1021atwr\sdk0.4_rev2\u-boot_sdcard.bin in SDSP16GB using address offset 0x00001000

12. Execute the target task.

After the eMMC/SD card has been programmed successfully with the U-Boot image, the board is recovered and is ready for use.

4.2. Complete board recovery

One of the major problems in board production is that eMMC is not removable and sometimes serial and network interfaces are not available. For such as a situation, this section provides a solution to perform complete board recovery, including U-Boot, U-Boot environment, kernel, device tree, and rootfs.

To create a complete board image, including U-Boot, U-Boot environment, kernel, device tree, and rootfs, perform these steps:

1. On a Linux host, create a temporary file on a hard disk with the capacity of the eMMC device.

```
b11883@fsr-ub1264-120:/$ sudo dd if=/dev/zero of=/tmp/myfs count=204800
204800+0 records in
204800+0 records out
104857600 bytes (105 MB) copied, 0.415984 s, 252 MB/s
```

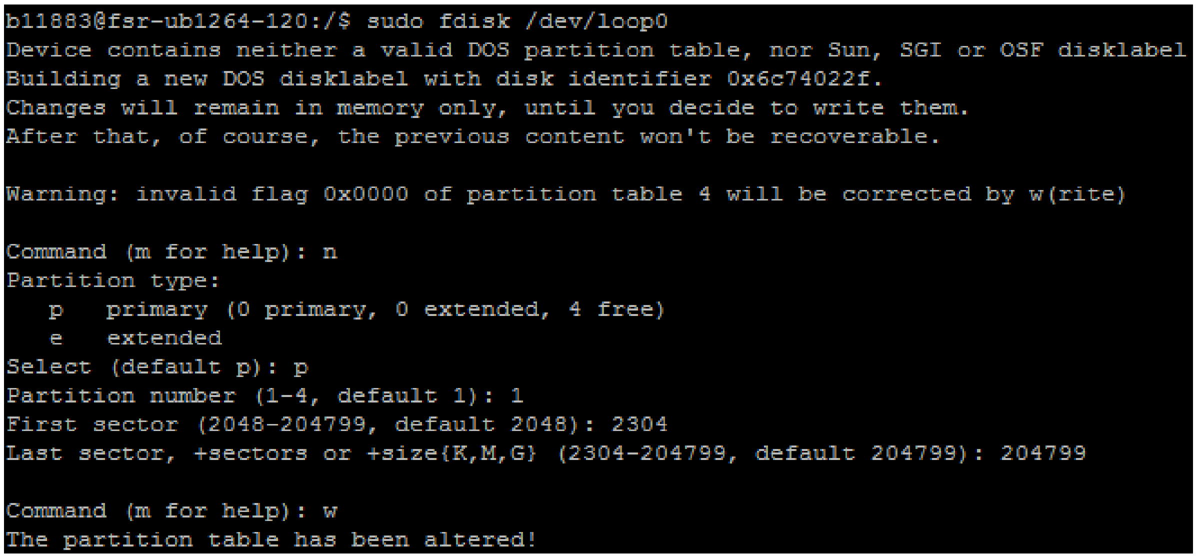
NOTE File size is calculated using block size, which is 512 bytes multiplied by the count. For the current example, a temporary file of 100 MB size was created.

2. Associate a loop device with the temporary file.

```
b11883@fsr-ub1264-120:/$ sudo losetup /dev/loop0 /tmp/myfs
```

3. Partition the loop device, as shown in the figure below.

Figure 8. Partitioning loop device



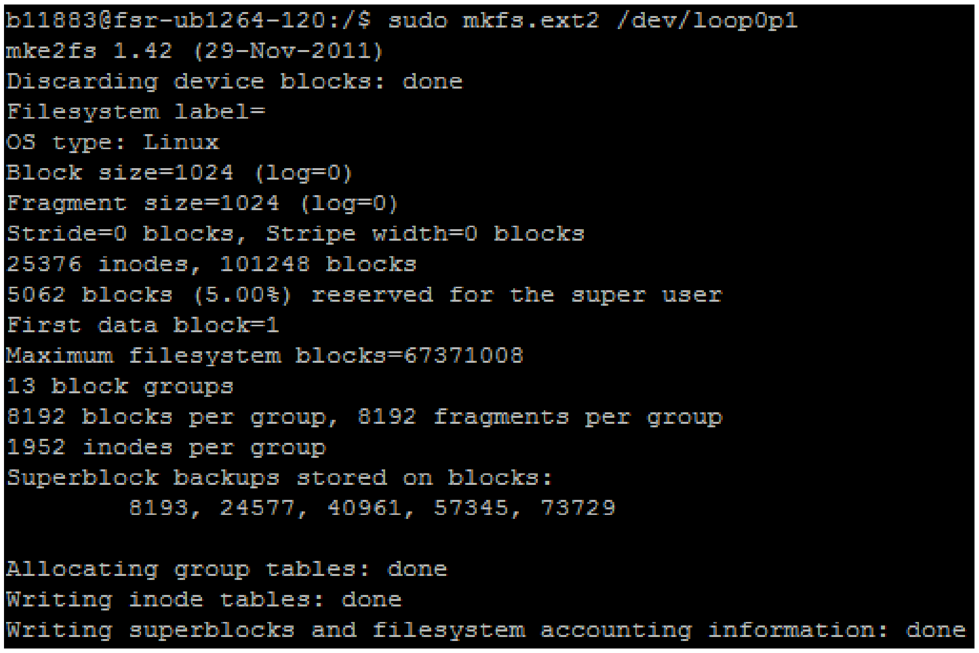
```
b11883@fsr-ub1264-120:/$ sudo fdisk /dev/loop0
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x6c74022f.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-204799, default 2048): 2304
Last sector, +sectors or +size{K,M,G} (2304-204799, default 204799): 204799

Command (m for help): w
The partition table has been altered!
```

NOTE For loop 0, first sector will be 2304 because up to this sector, U-Boot and U-Boot environment will be placed.

If after writing table to disk and exiting, you get a warning "Re-reading the partition table failed with error 22: Invalid argument", then run the `partprobe` command on the partition created (*`sudo partprobe /dev/loop0`*).

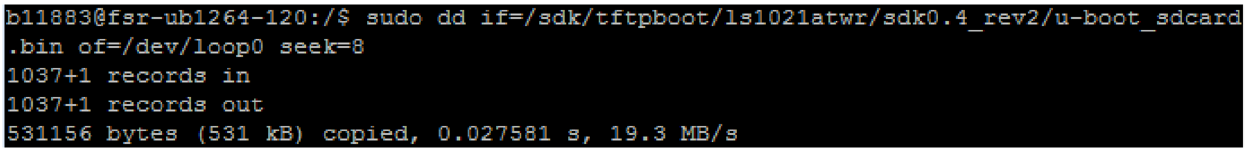
4. Format the loop device, as shown in the figure below.

Figure 9. Formatting loop device



```
b11883@fsr-ub1264-120:/$ sudo mkfs.ext2 /dev/loop0p1
mke2fs 1.42 (29-Nov-2011)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
25376 inodes, 101248 blocks
5062 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
13 block groups
8192 blocks per group, 8192 fragments per group
1952 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

5. Copy U-Boot and U-Boot environment into the loop device.

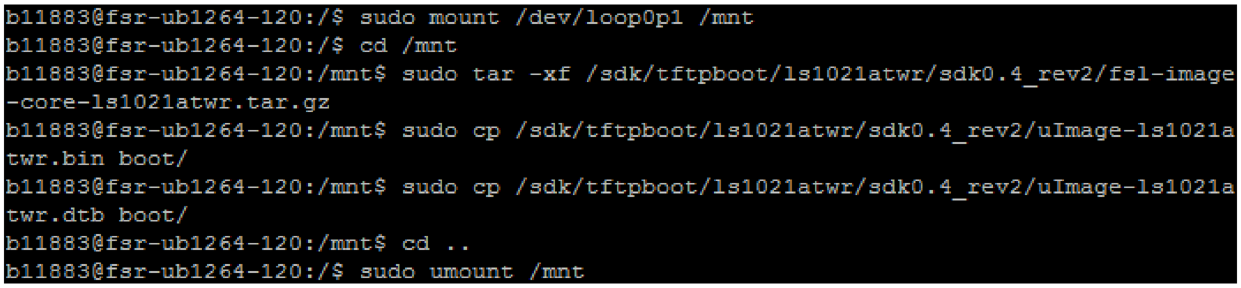


```
b11883@fsr-ub1264-120:/$ sudo dd if=/sdk/tftpboot/ls1021atwr/sdk0.4_rev2/u-boot_sdcard
.bin of=/dev/loop0 seek=8
1037+1 records in
1037+1 records out
531156 bytes (531 kB) copied, 0.027581 s, 19.3 MB/s
```

NOTE Ensure that the first eight sectors from loop 0 are skipped; otherwise the partition table will be overwritten by U-Boot.

6. Mount the loop device partition and untar rootfs into it.
7. Copy kernel image and device tree into the `/boot` folder, as shown in the figure below.

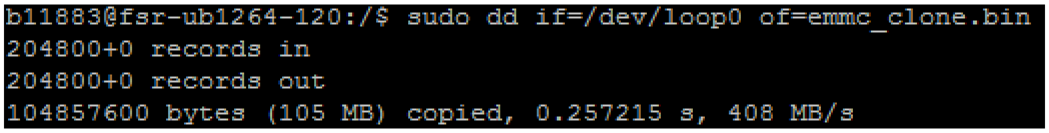
Figure 10. Copying kernel image and device tree



```
b11883@fsr-ub1264-120:/$ sudo mount /dev/loop0p1 /mnt
b11883@fsr-ub1264-120:/$ cd /mnt
b11883@fsr-ub1264-120:/mnt$ sudo tar -xf /sdk/tftpboot/ls1021atwr/sdk0.4_rev2/fsl-image
-core-ls1021atwr.tar.gz
b11883@fsr-ub1264-120:/mnt$ sudo cp /sdk/tftpboot/ls1021atwr/sdk0.4_rev2/uImage-ls1021a
twr.bin boot/
b11883@fsr-ub1264-120:/mnt$ sudo cp /sdk/tftpboot/ls1021atwr/sdk0.4_rev2/uImage-ls1021a
twr.dtb boot/
b11883@fsr-ub1264-120:/mnt$ cd ..
b11883@fsr-ub1264-120:/$ sudo umount /mnt
```

8. Create a clone of the loop device.

Figure 11. Creating a clone of loop device



```
b11883@fsr-ub1264-120:/$ sudo dd if=/dev/loop0 of=emmc_clone.bin
204800+0 records in
204800+0 records out
104857600 bytes (105 MB) copied, 0.257215 s, 408 MB/s
```

9. Use flash programmer to flash the clone into eMMC. You are recommended not to specify any offset so that the entire device, starting from the first address, is programmed. See RCW/U-Boot recovery for more details on using flash programmer.

NOTE Programming large images using flash programmer may take several hours. Therefore, the procedure described in this section should be used as the last option for board recovery.

Document Number: AN5184

18 January 2016