Booting Linux™ on SAM9X60 MPU from NAND Flash

TB3365

www.microchip.com Product Pages: SAM9X60

Introduction

This document describes how to boot Linux on a SAM9X60 device from a NAND Flash memory.

The device can boot from any of the external non-volatile memories (NVM) available on the board, such as an SD card, e.MMC, QSPI, SPI or NAND Flash memory. To boot from a NAND Flash memory, at91bootstrap, u-boot, the Linux Kernel and the root file system must be flashed into the NAND Flash memory. This document explains how to configure, build and program files on the SAM9X60.

Reference Documents

The following documents are available on www.microchip.com.

Table 1. Reference Documents

Type	Document Title	Literature No.
Data Sheet	SAM9X60	DS60001579
User's Guide	SAM9X60-Curiosity	DS60001783

1. Prerequisites

- Hardware:
  - A host computer running Linux operating system.
    Note: This technical brief's content was developed and tested on Ubuntu 22.04.4 LTS.
  - SAM9X60-Curiosity board (EV40E67A) Rev 4
  - One micro-B USB cable (included in the SAM9X60-Curiosity box)
  - One FTDI TTL-232R USB-to-TTL serial cable
- Software:
  This demo is based on a Linux distribution for the SAM9X60 Microchip MPU. This distribution is built using the Buildroot build system. The first step is to set up the Buildroot development environment.

2. Get Buildroot for Microchip MPU

To get the source code, the buildroot-mchp and buildroot-external-microchip repositories should be cloned. buildroot-mchp is a fork of Buildroot with a few specific patches. The external tree provides additional defconfigs and packages dedicated to our demos.

To clone buildroot-mchp from the repository:

```
$ git clone github.com/linux4microchip/buildroot-mchp
```

To clone buildroot-external-microchip from the repository:

```
$ git clone github.com/linux4microchip/buildroot-external-microchip
```

The source code should be pointing to the latest version of the buildroot-mchp and buildroot-external-microchip repositories.

Tip: We advise you to use the same linux4microchip tag for both repositories.

Note: The buildroot-mchp and buildroot-external-microchip tree is now part of linux4microchip, hence the tags on this repository are named linux4microchip-***.

To check out the latest version of buidlroot-mchp:

1. Navigate to the buildroot-mchp directory:
   ```
   $ cd buildroot-mchp
   ```
2. Enter the following command to display the versions available in 2024:
   ```
   $ git tag | grep 2024
   ```
3. Check out the latest version:
   ```
   $ git checkout linux4microchip-2024.04
   ```

To check out the latest version of buildroot-external-microchip:

1. Navigate to the buildroot-external-microchip directory:
   ```
   $ cd ..
   $ cd buildroot-external-microchip
   ```
2. Enter the following command to know the available versions in 2024:
   ```
   $ git tag | grep 2024
   ```
3. Check out the latest version:
   ```
   $ git checkout linux4microchip-2024.04
   ```

3. Configure at91bootstrap

1. Navigate to the buildroot-mchp directory:

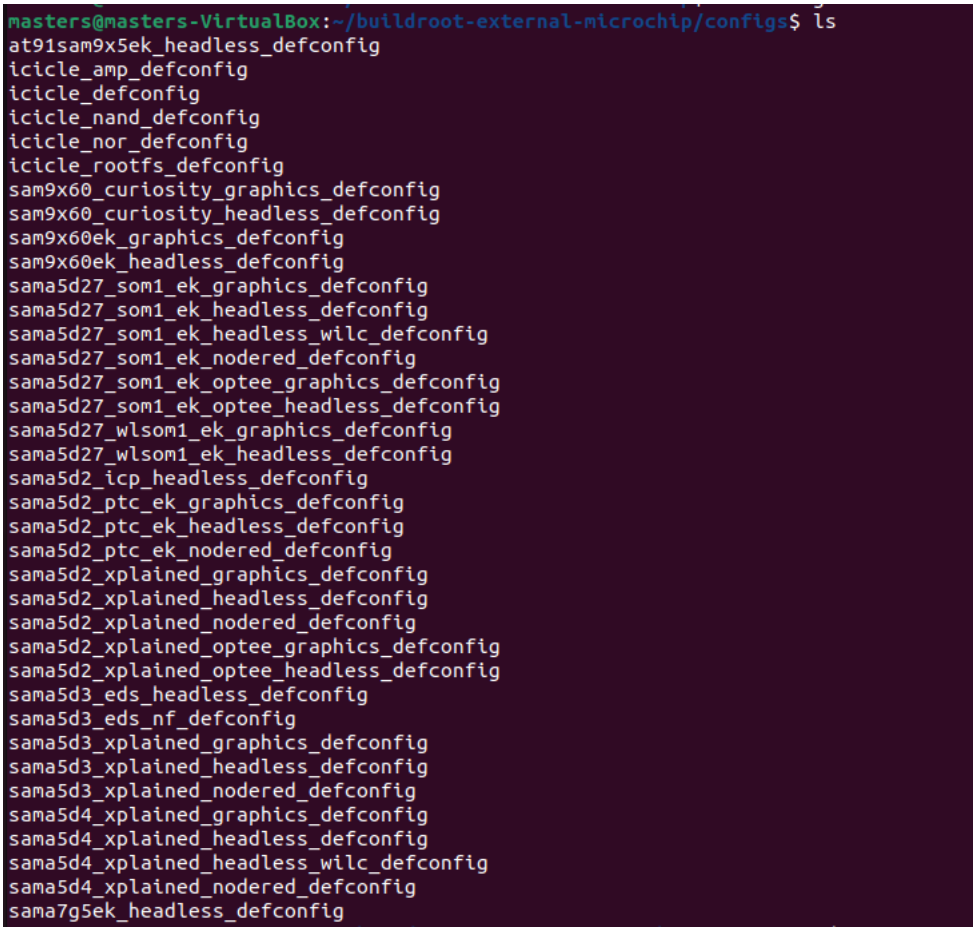
   ```
   $ cd buildroot-mchp
   ```

2. Enter the following command to export the additional defconfigs and packages from the external tree:

   ```
   $ export BR2_EXTERNAL=../buildroot-external-microchip/
   ```

3. Navigate to the configs directory in the buildroot-external-microchip directory to find the SAM9X60 Curiosity defconfig files:

   ```
   $ cd ..

   $ cd buildroot-external-microchip/configs

   $ ls
   ```



4. From the list, note the configuration file name “sam9x60_curiosity_headless_defconfig”.
5. Navigate to the buildroot-mchp directory and make the configuration files:

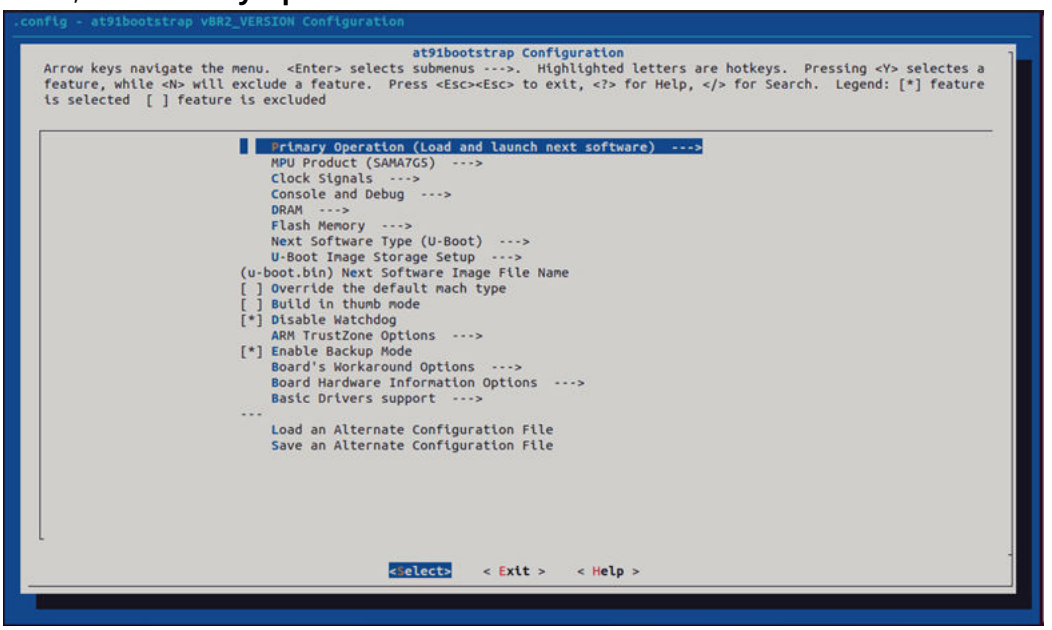
   ```
   $ cd ..

   $ cd buildroot-mchp

   $ make sam9x60_curiosity_headless_defconfig
   ```

4. Customize at91bootstrap to boot from NAND Flash

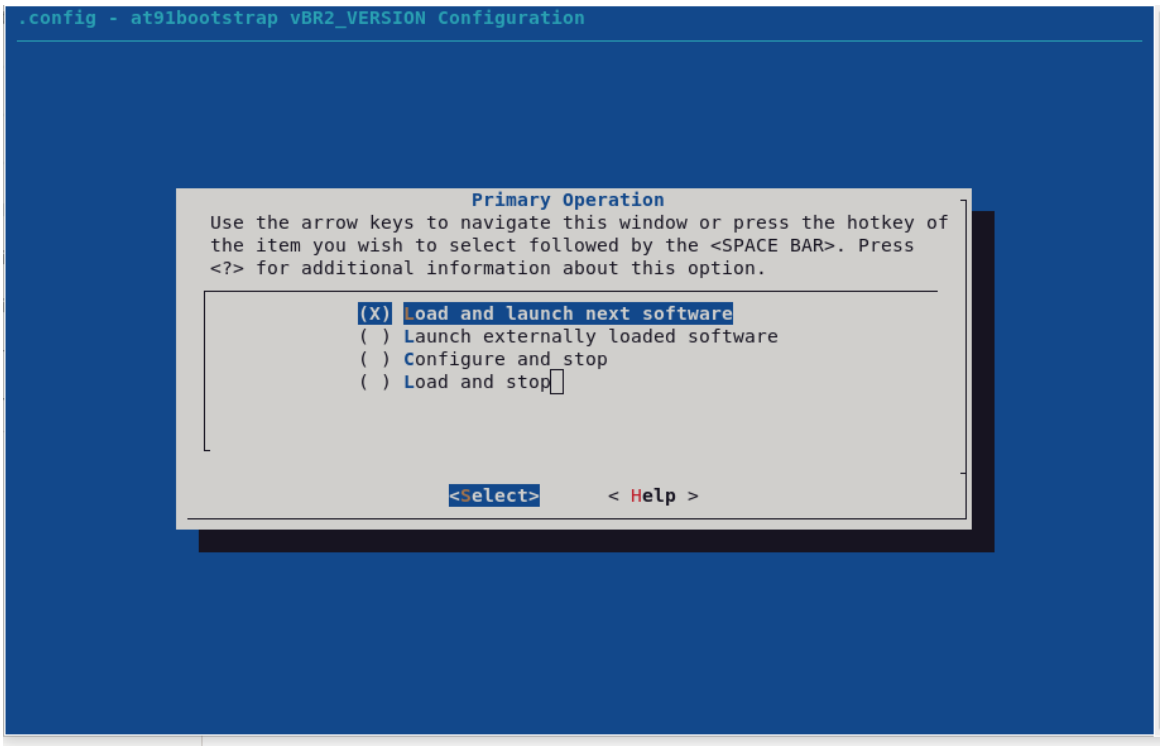
1. Enter the following command to modify the configurations:

   ```
   $ make at91bootstrap3-menuconfig
   ```
   The **at91bootstrap Configuration** window opens.

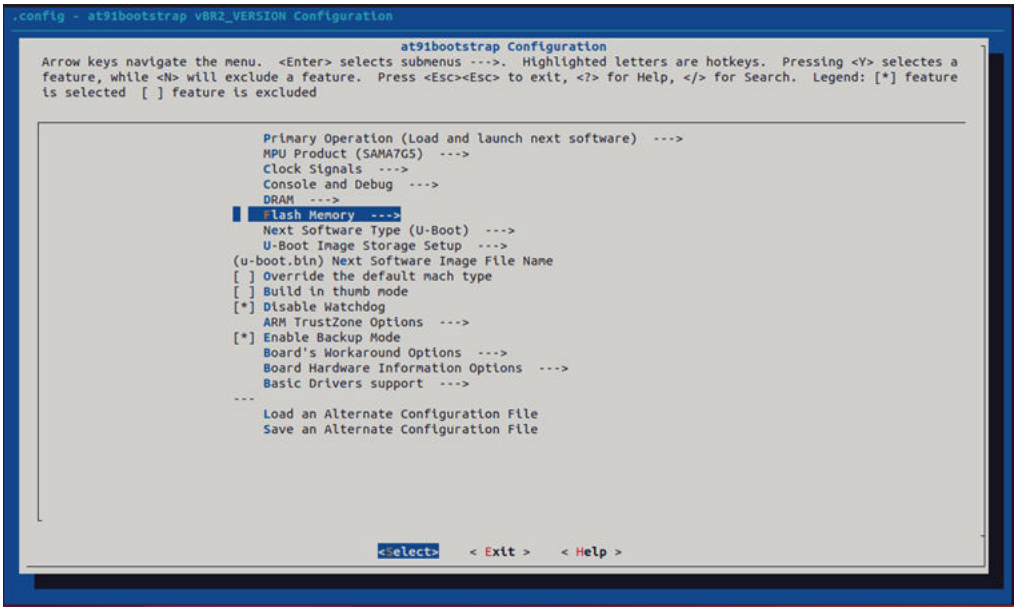
2. In the menu, click **Primary Operation.**



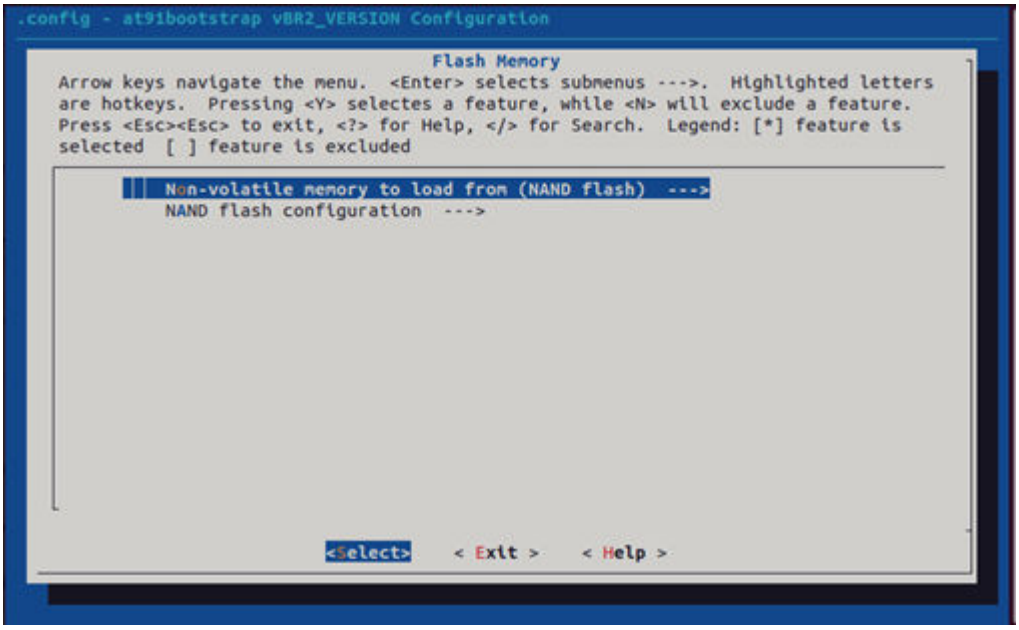
   a. In the **Primary Operation** menu, ensure the **Load and launch next software** check box is selected. If not, select it and press Enter.



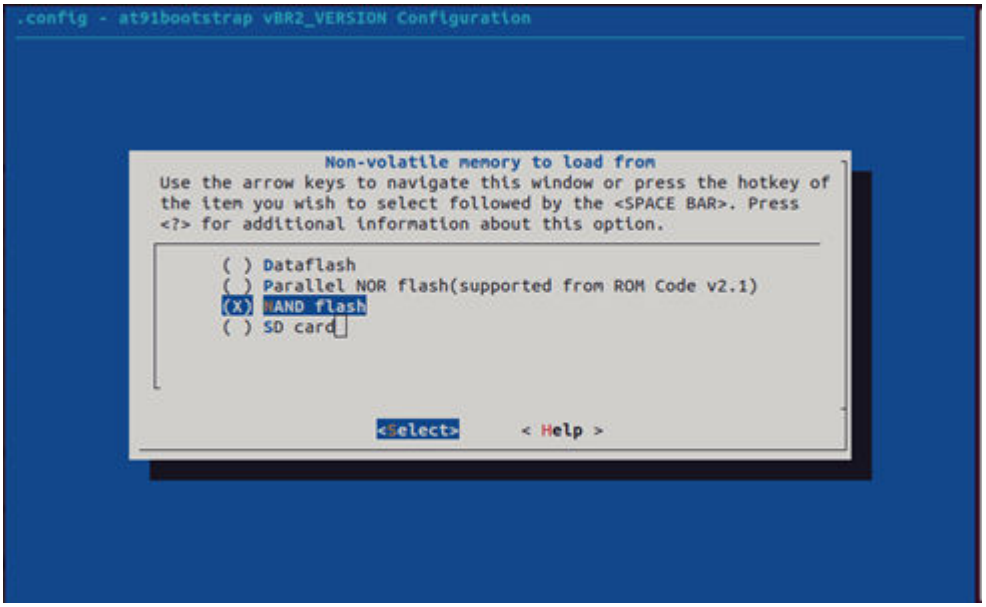
3. In the **at91bootstrap Configuration** window, select **Flash Memory** and press Enter.



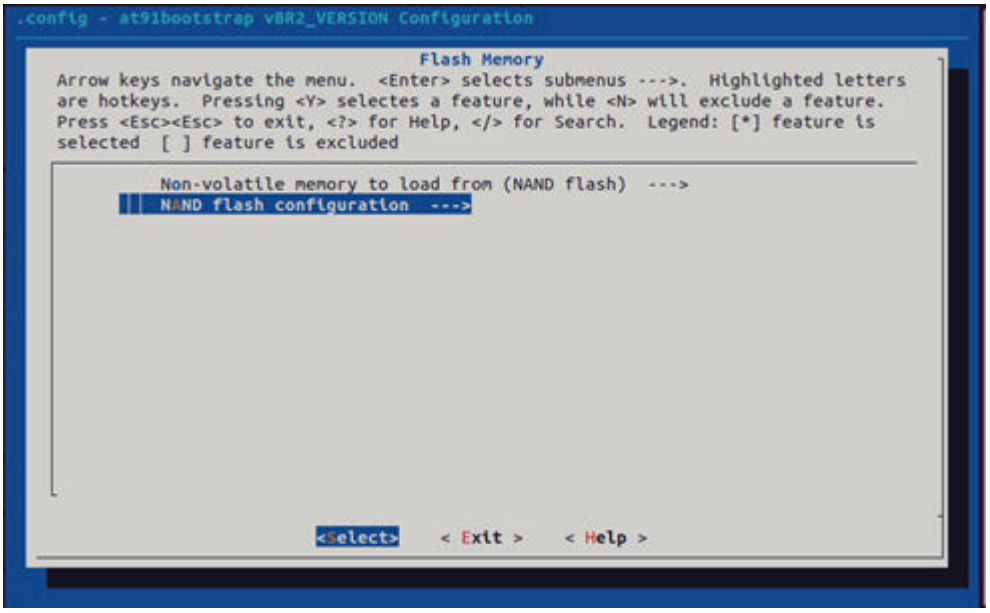
a. In the **Flash Memory** menu, select **Non-volatile memory to load from** and press Enter.



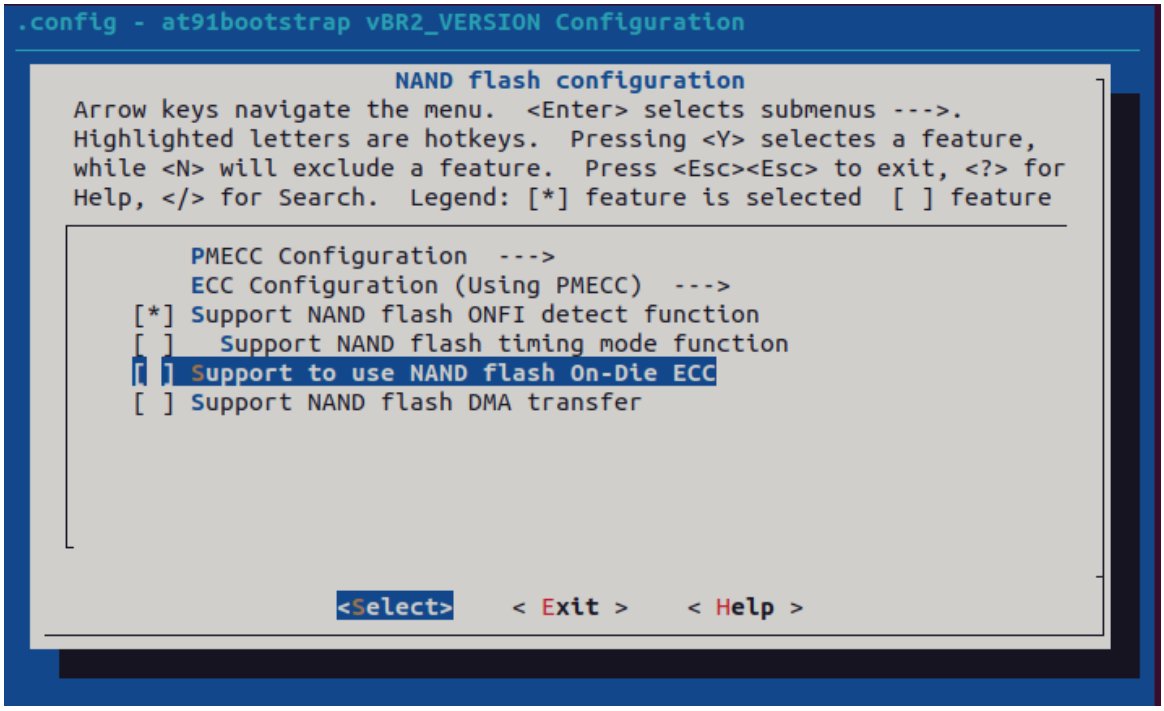
b. In the **Non-volatile memory to load from** menu, ensure **NAND flash** is selected. If not, select it and press Enter.



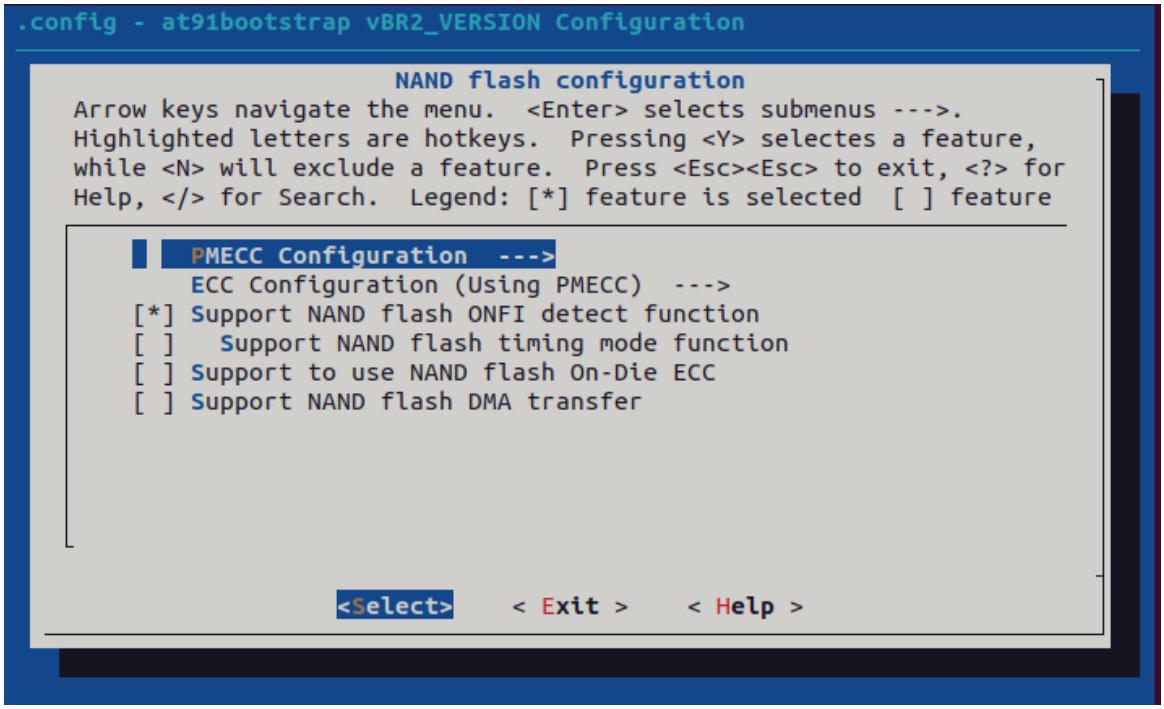
c. In the **Flash Memory** menu, select **NAND flash configuration** and press Enter.



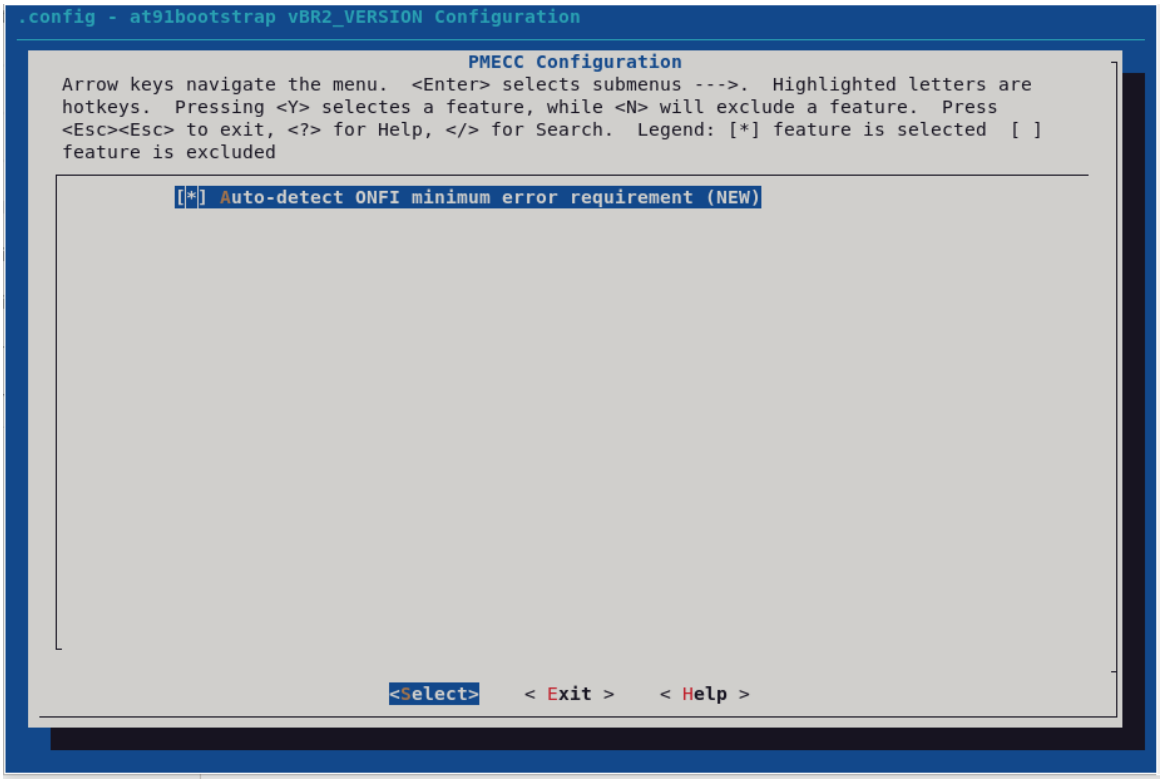
d. In the **NAND flash configuration** menu:
   i. Select **Support NAND flash ONFI detect function** and press Enter.
   ii. Deselect **Support to use NAND flash On-Die ECC** and press Enter.



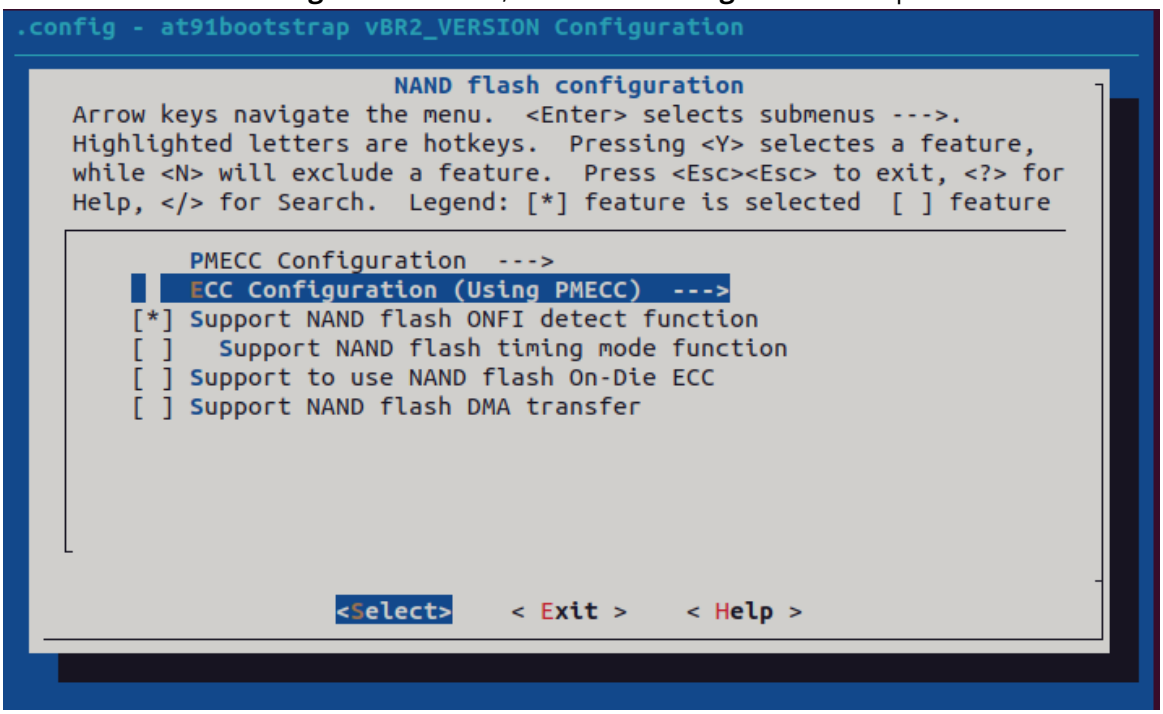
e. In the **NAND flash configuration** menu, select **PMECC Configuration** and press Enter.



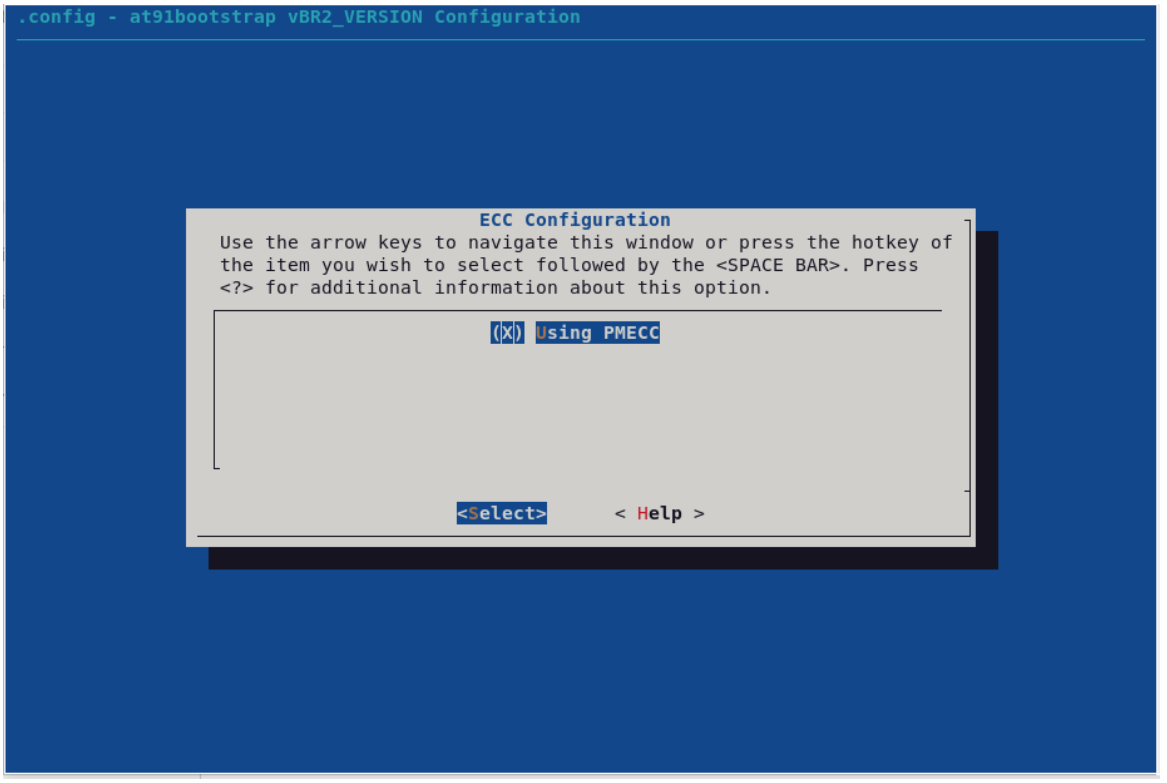
f. In the **PMECC Configuration** window, select **Auto-detect ONFI minimum error requirement** and press Enter.



g. In the **NAND flash configuration** menu, select **ECC Configuration** and press Enter.

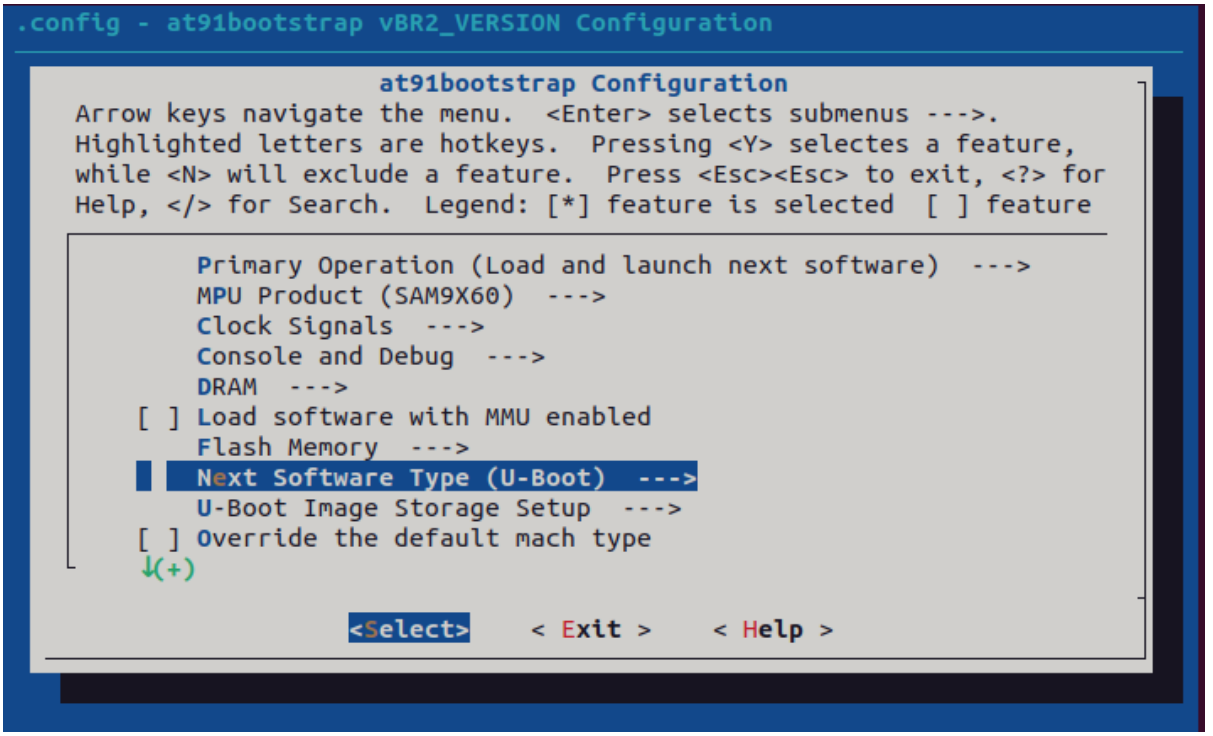


h. In the **ECC Configuration** menu, select **Using PMECC** and press Enter.

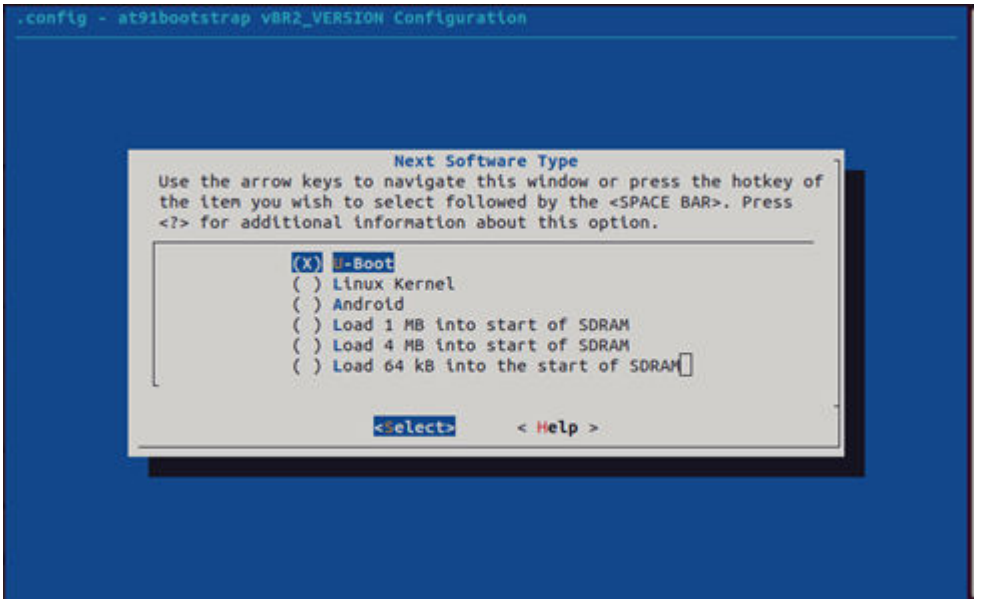


i. Press ESC twice to exit the menu.

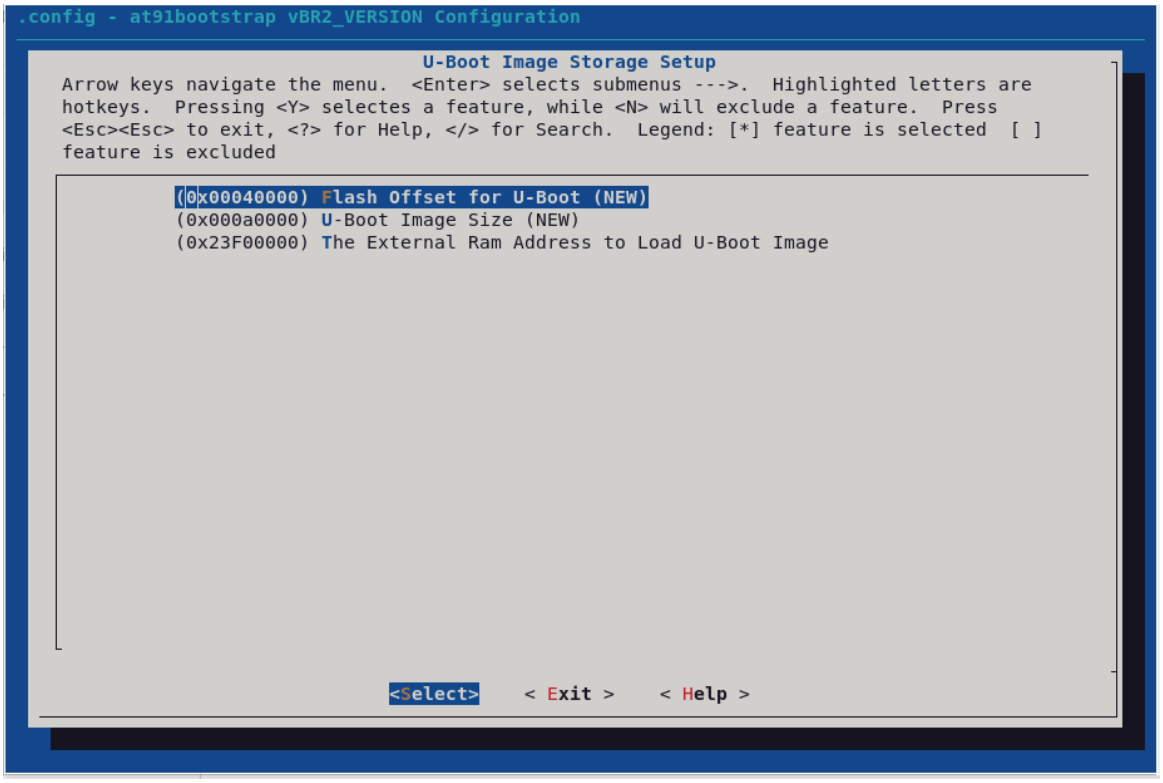
4. In the **at91bootstrap Configuration** window, select **Next Software Type** and press Enter.



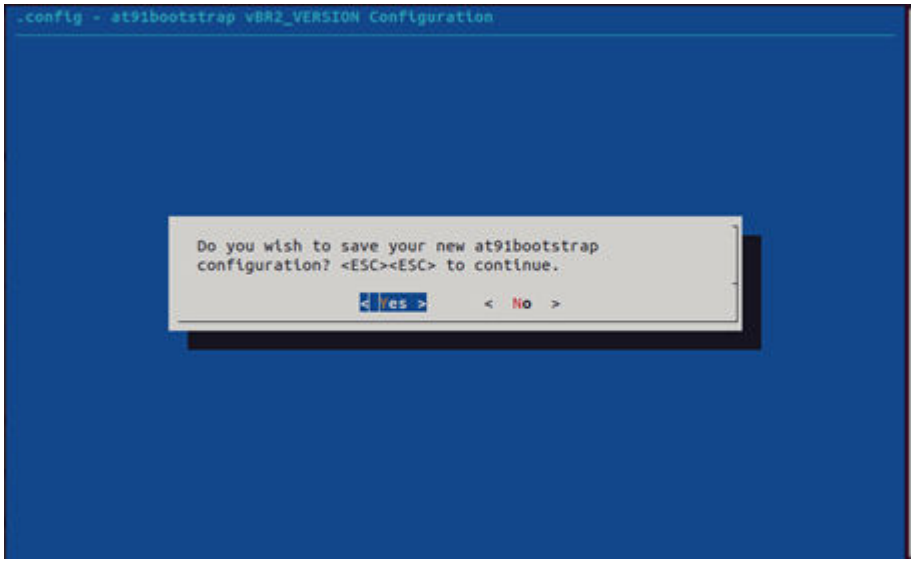
a. In the **Next Software Type** menu, ensure the **U-Boot** check box is selected. If not, select it and press Enter.



5. In the **at91bootstrap configuration** window, select **U-Boot Image Storage Setup** and press Enter.
   a. In the **U-Boot Image Storage Setup** menu, update the **Flash Offset for U-Boot, U-Boot Image Size** and **The External Ram Address to Load U-Boot Image** details as shown in the image below and press Enter.



   b. Press ESC twice to exit the menu.
6. Select **Exit** and press Enter.
   a. Click **Yes**.



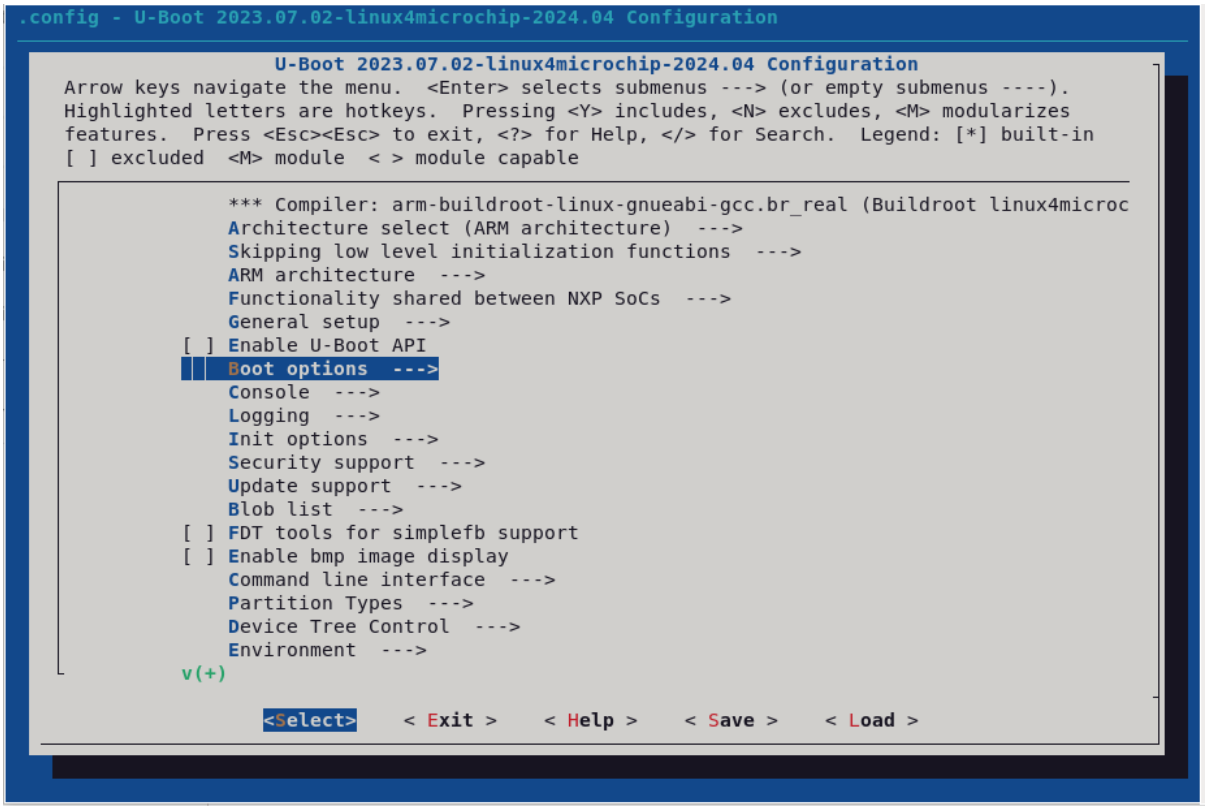
5. Modify U-Boot to boot from NAND Flash

1. Enter the following command to modify the configurations:

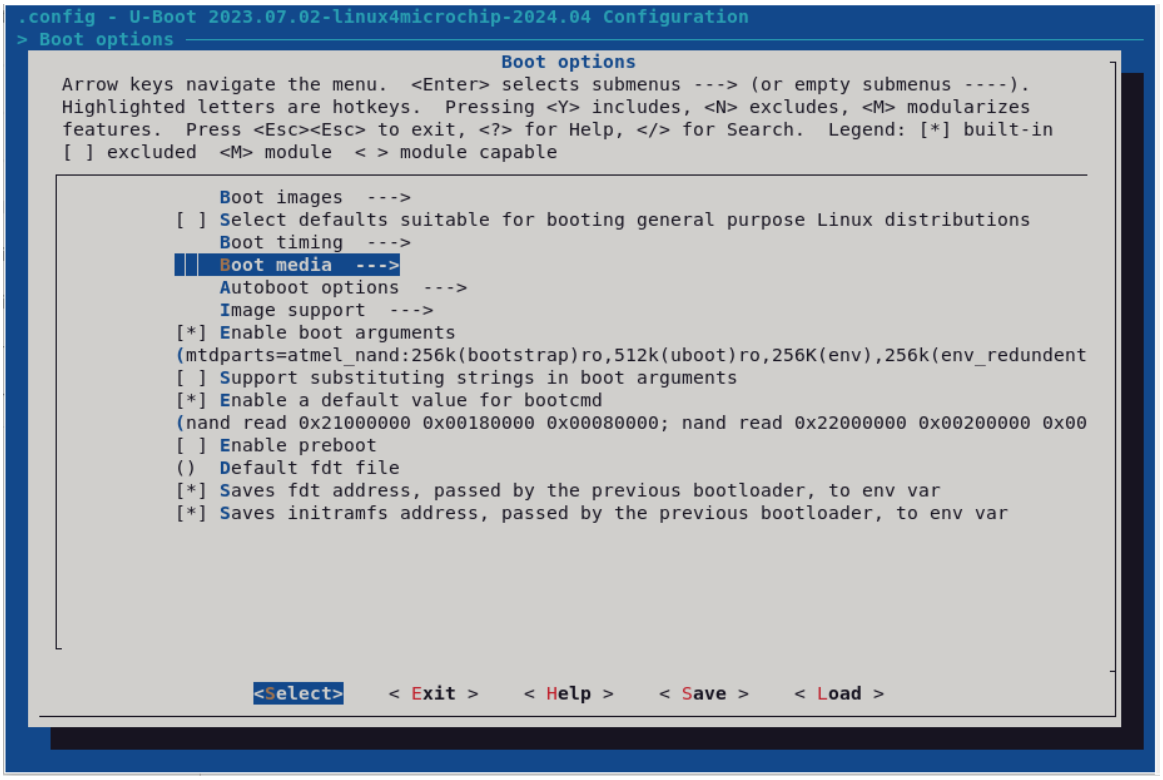
```
$ make uboot-menuconfig
```

Note that building and opening the **U-Boot configuration** window will take a few minutes.

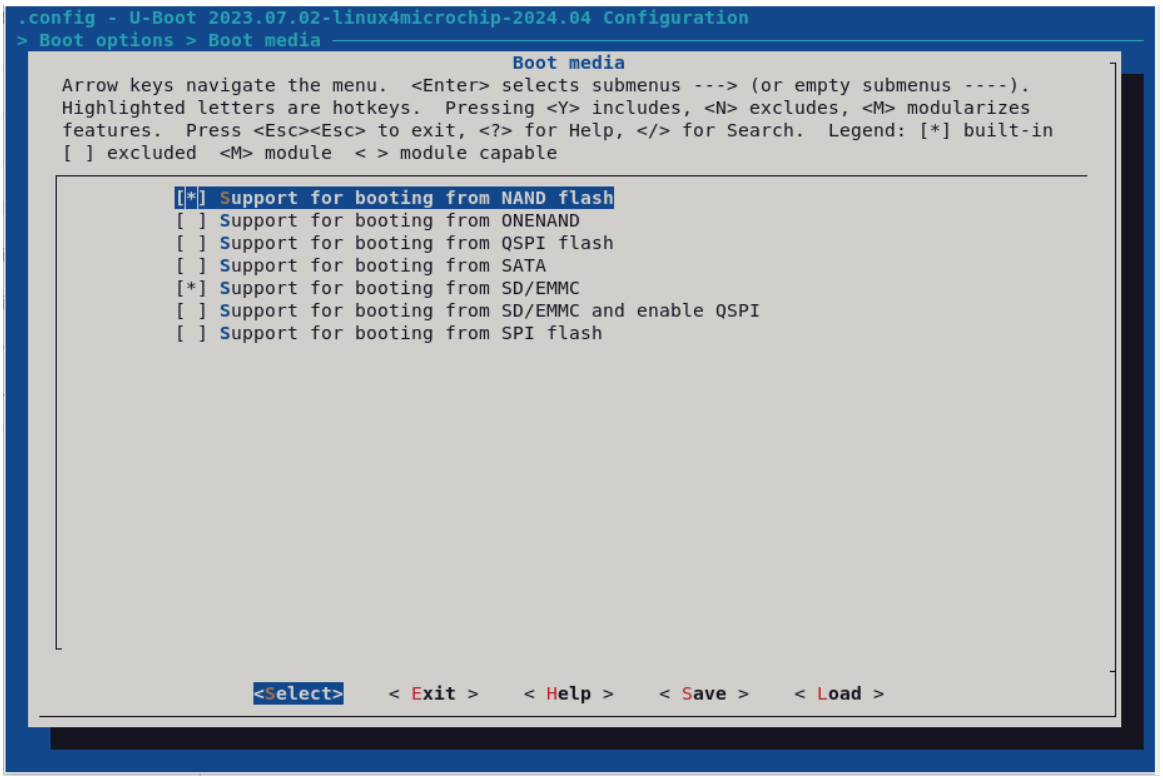
2. From the menu, select **Boot options** and press Enter.



a. In the **Boot options** window, select **Boot media** and press Enter.

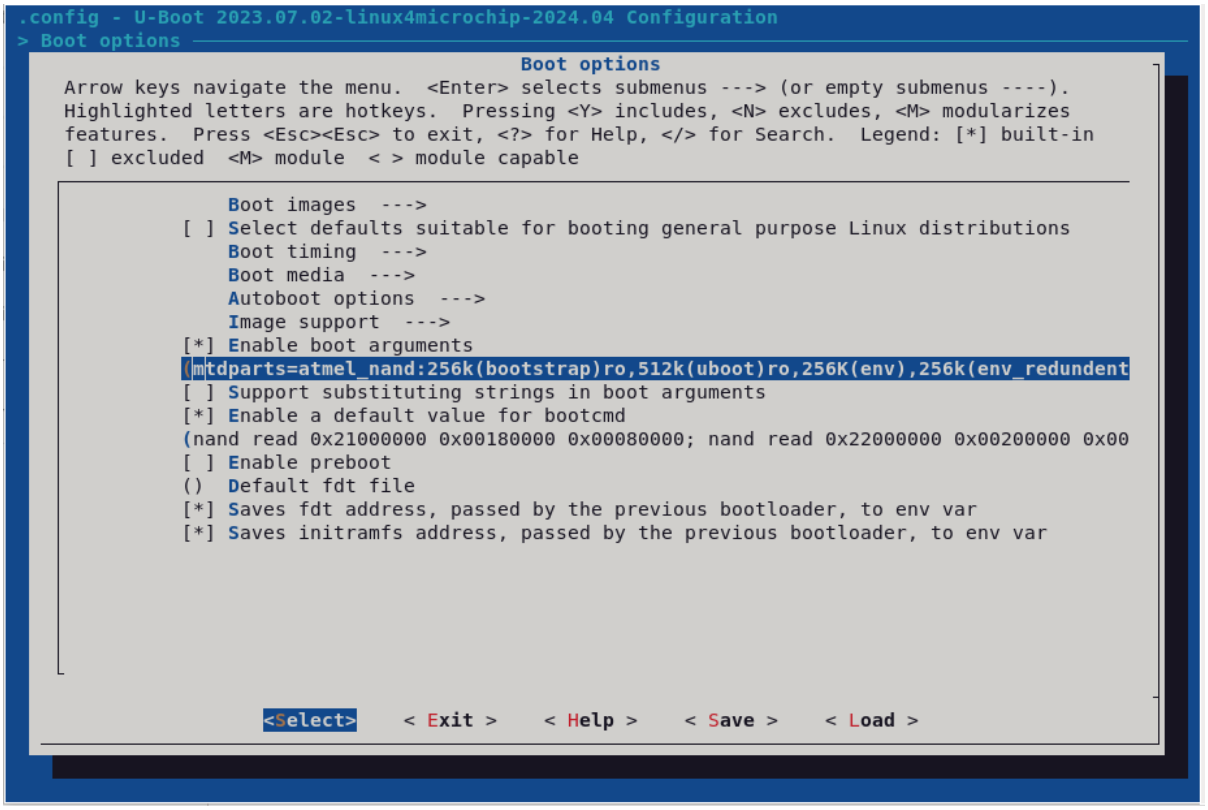


b. In the **Boot media** window, select **Support for booting from NAND flash** and press 'Y'.



c. Press ESC twice to exit the menu.

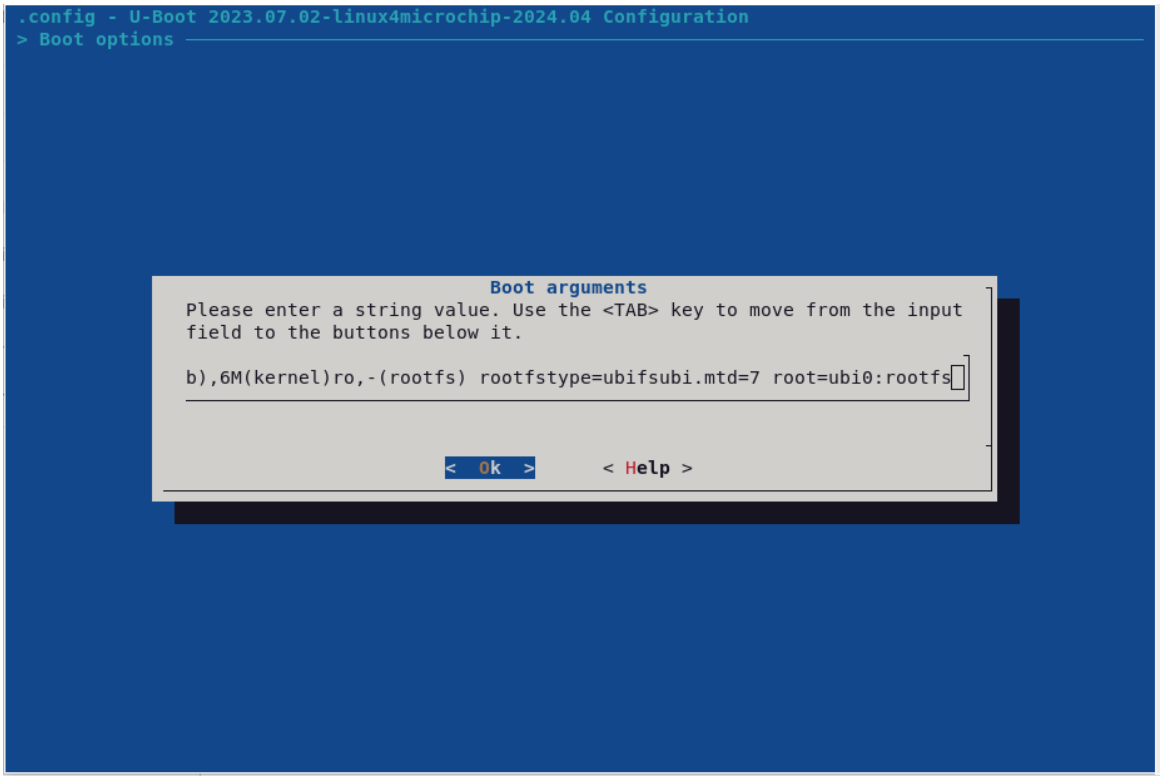
3. In the **Boot Options** window, select **Boot arguments** and press Enter.



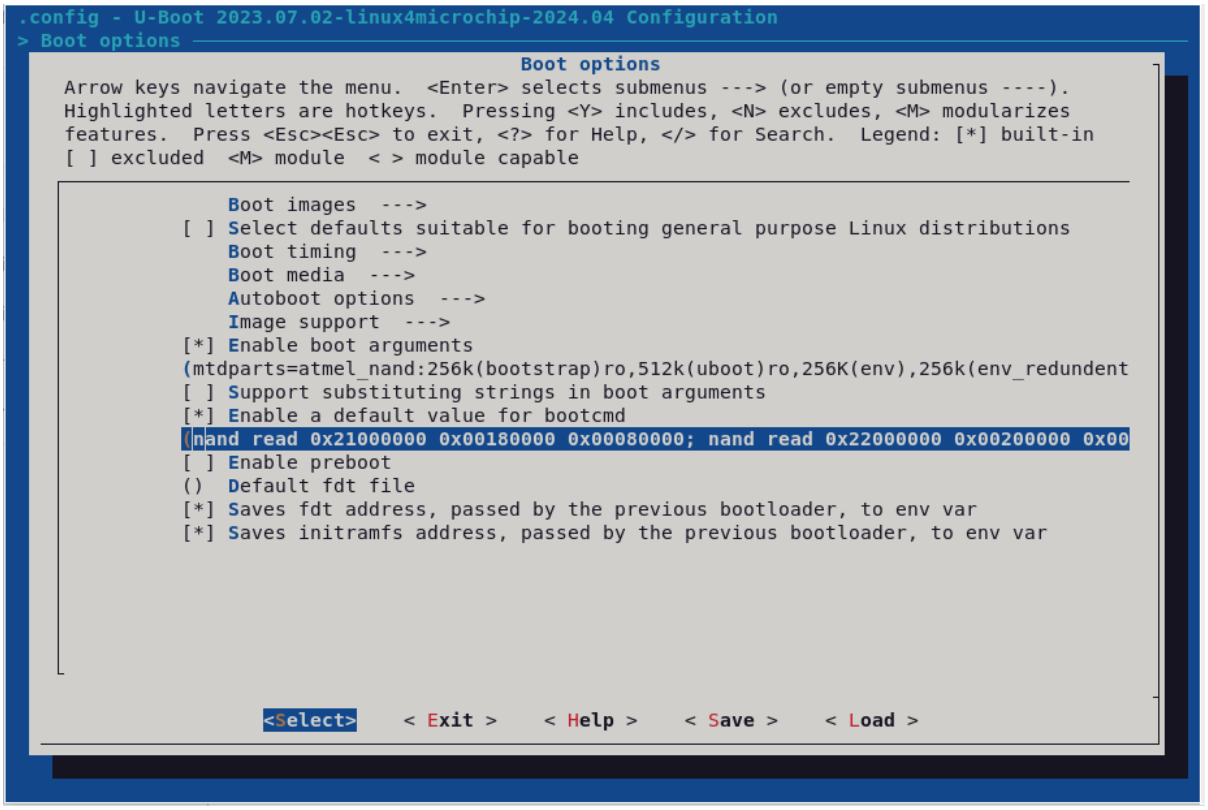
a. In the **Boot arguments** window, update the string to:

```
mtdparts=atmel_nand:256k(bootstrap)ro,512k(uboot)ro,256K(env),256k(env_r
edundent),256k(spare),512k(dtb),6M(kernel)ro,-(rootfs) rootfstype=ubifs
ubi.mtd=7 root=ubi0:rootfs
```

and press Enter.



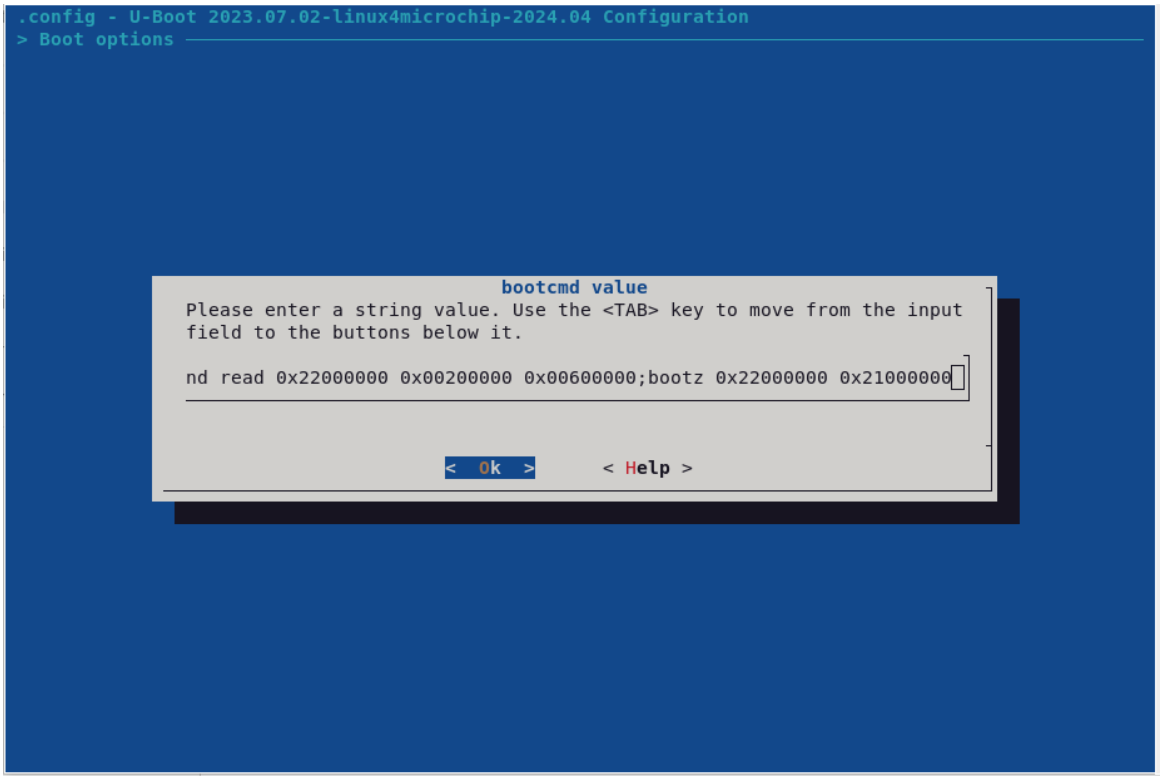
4. In the **Boot Options** window, select **Bootcmd** and press Enter.



a. In the **Bootcmd value** window, update the string to:

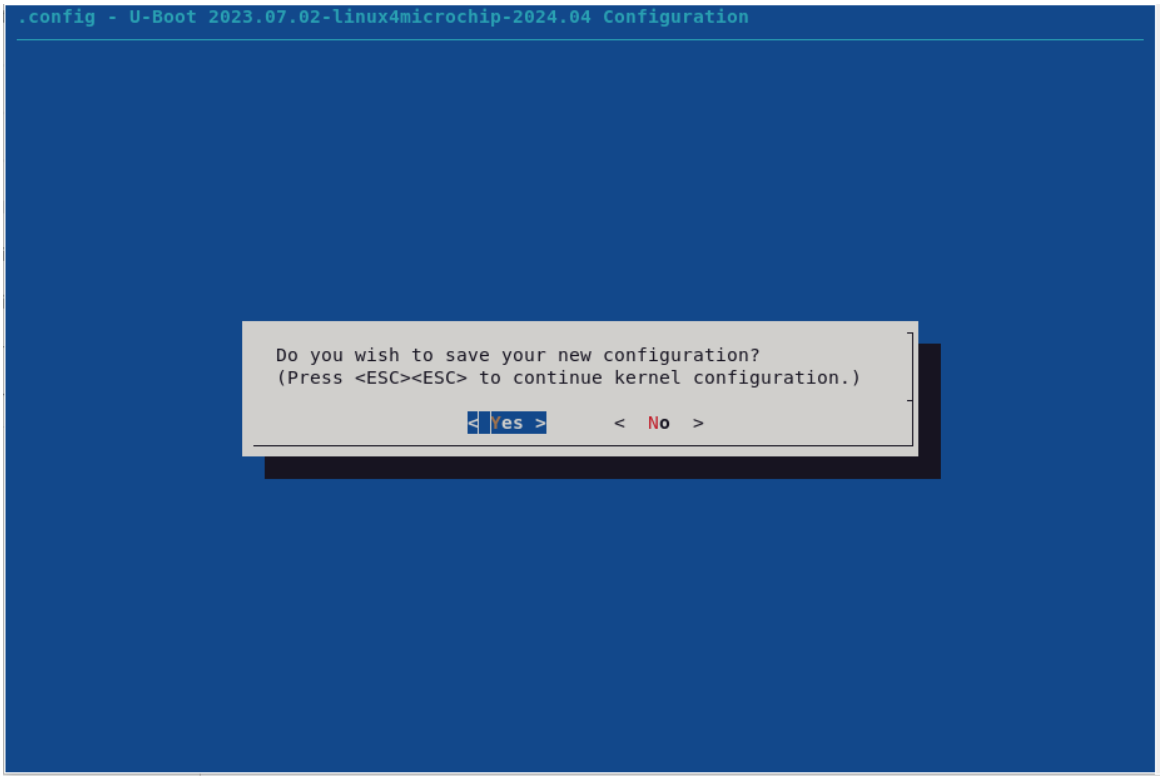
```
nand read 0x21000000 0x00180000 0x00080000; nand read 0x22000000
0x00200000 0x00600000;bootz 0x22000000 – 0x21000000
```

and press Enter.



b. Press ESC twice to exit the menu.

5. Select **Exit** and press Enter.
   a. Click **Yes**.



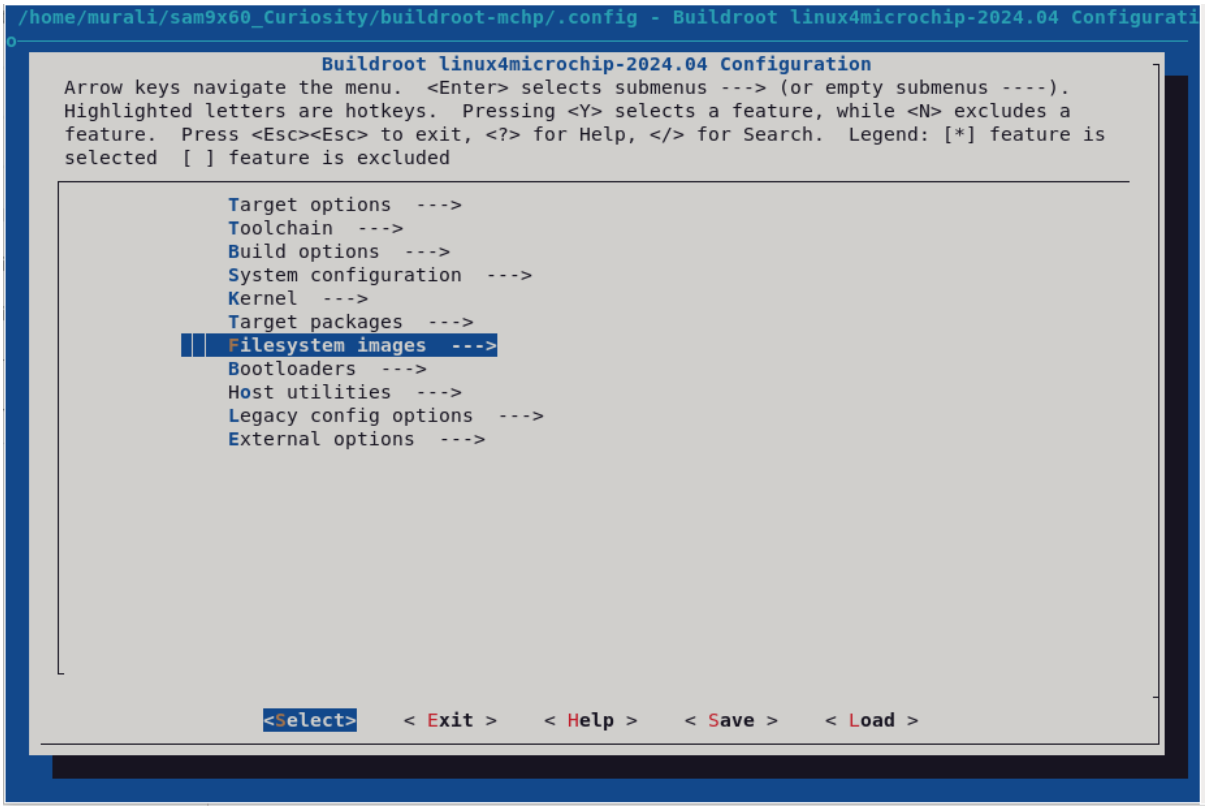
6. Modify file system details in menuconfig to boot from NAND Flash

1. Enter the following command to modify the configurations:

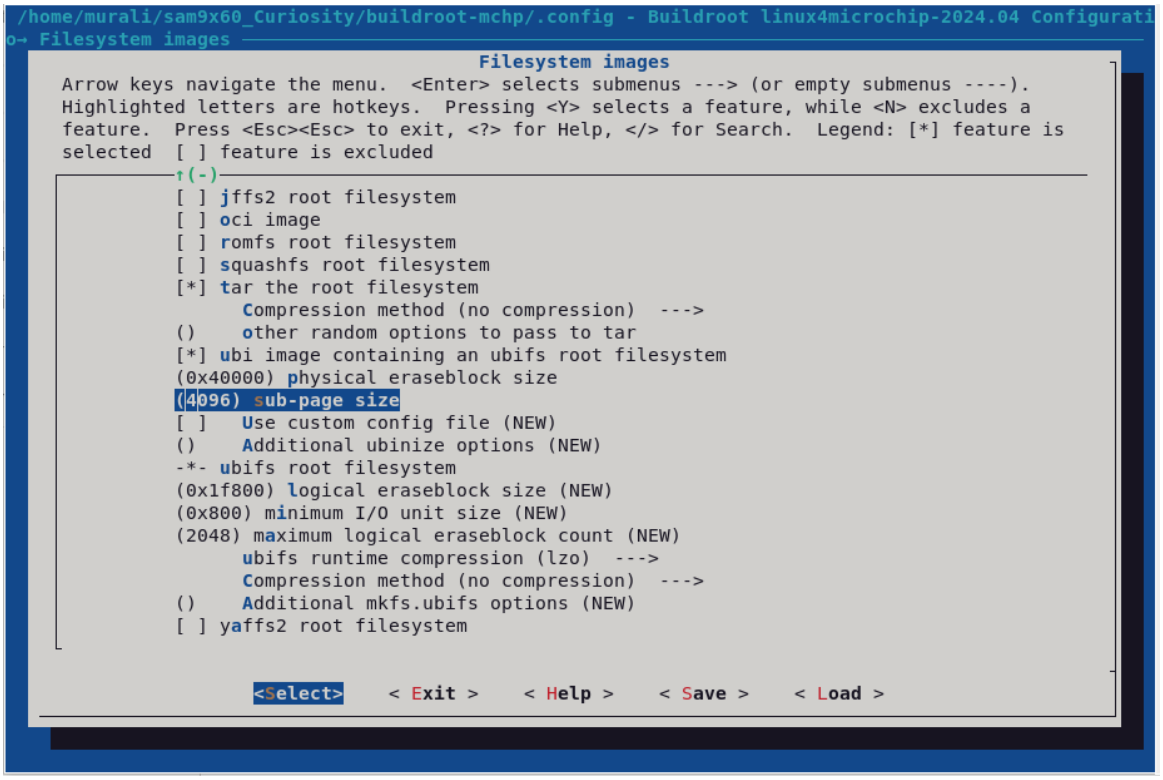
   ```
   $ make menuconfig
   ```

   The configuration window opens.

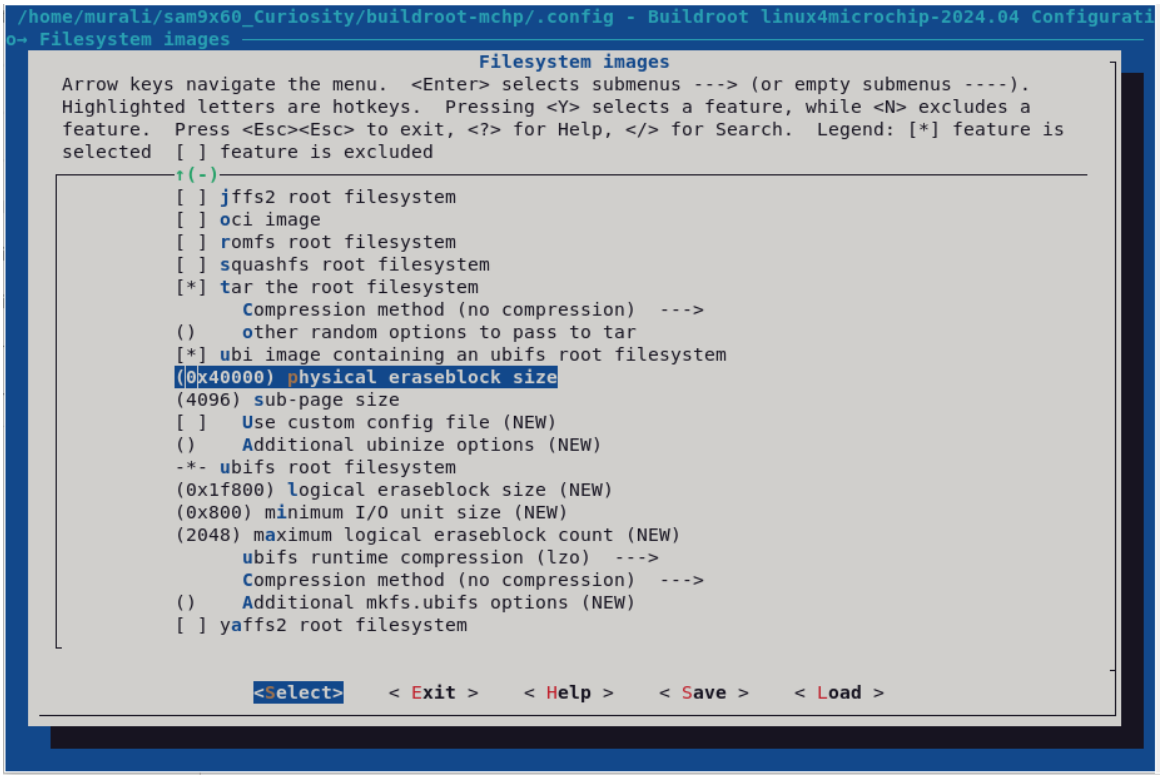
2. From the menu, select **Filesystem images** and press Enter.



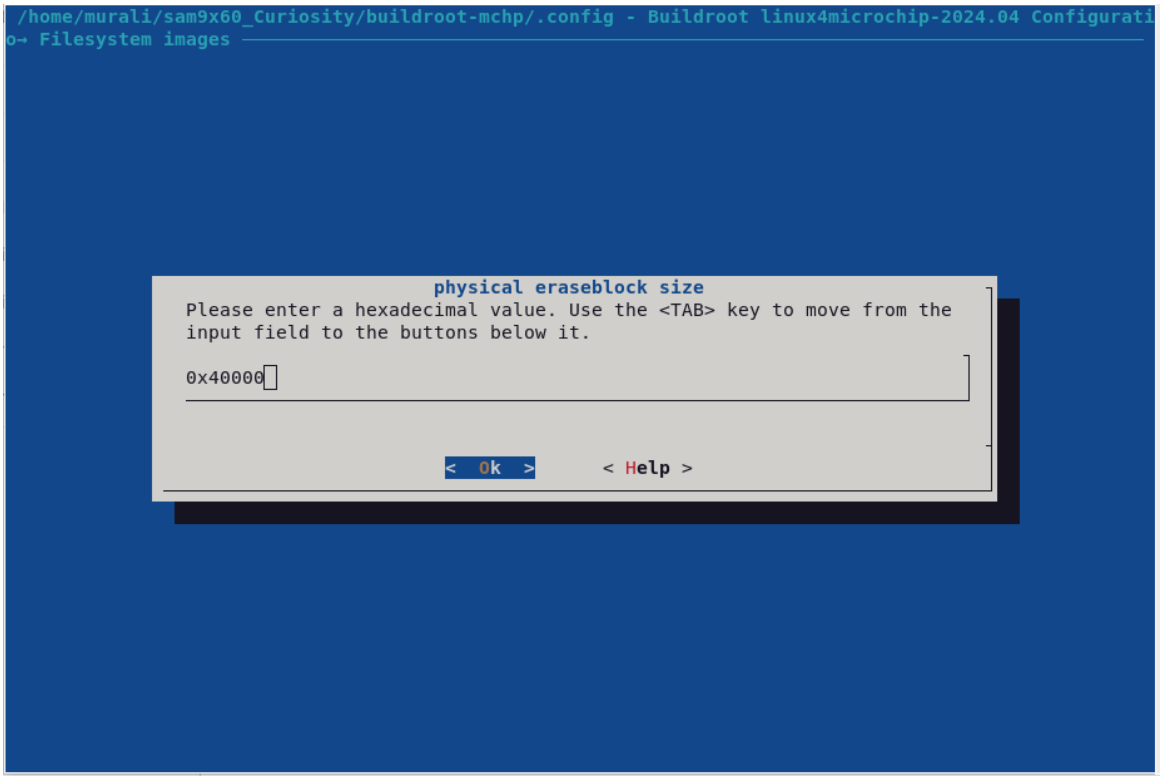
   a. In the **Filesystem images** window, ensure the **ubi image containing an ubifs root filesystem** checkbox is enabled. If not, select it and press 'Y' to enable.



b. In the **Filesystem images** window, select **physical eraseblock size** and press Enter.

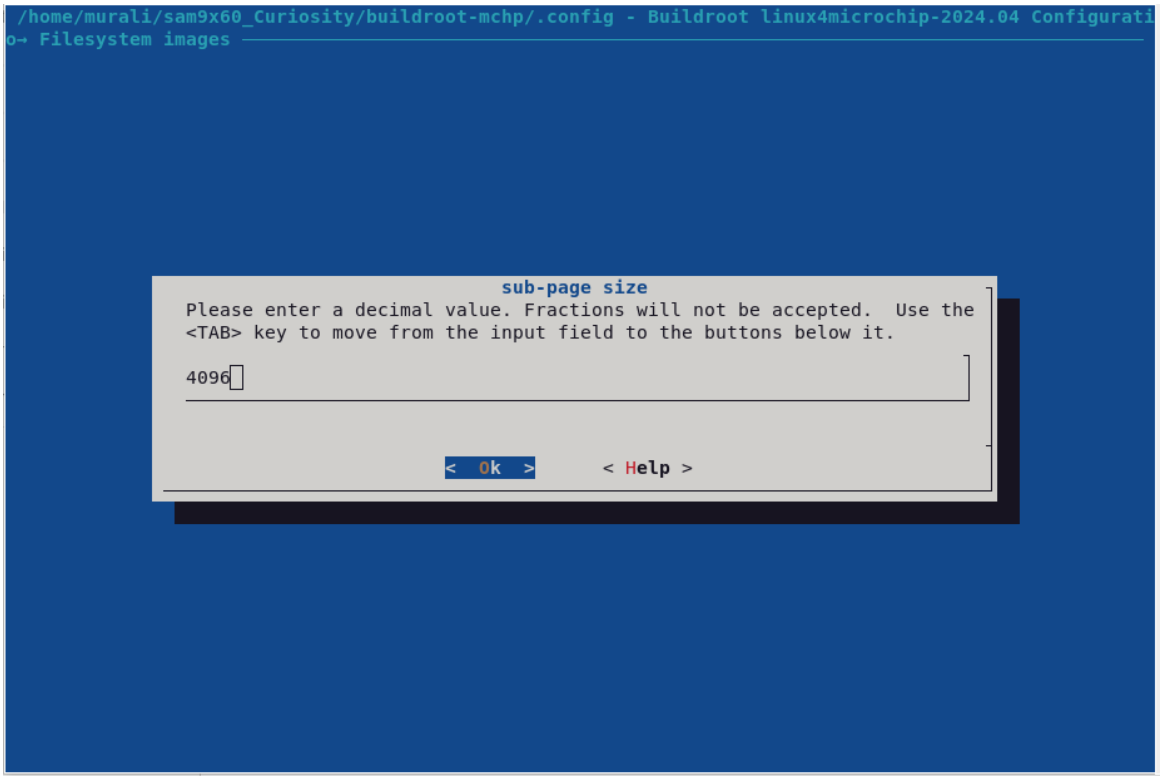


c. Update the **physical eraseblock size** to **0x40000** and press Enter. The physical erase block size is the NAND Flash block size in bytes.



d. In the **Filesystem images** window, select **sub-page size** and press Enter.

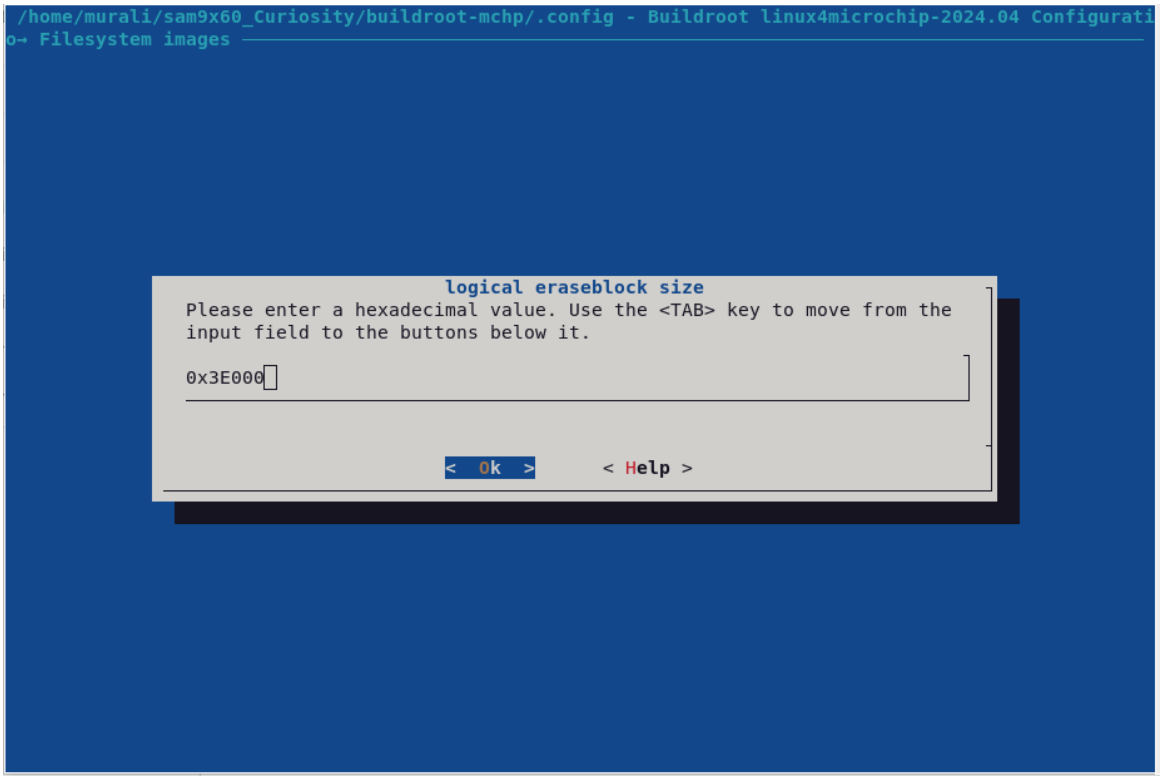
e. Update the **sub-page size** to **4096** and press Enter. Some NAND Flash devices support sub-pages,and some do not. Those that do not must be updated with the actual page size as the sub-page size.



f. In the **Filesystem images** window, ensure **ubifs root filesystem** is enabled. If not, select it and press 'Y' to enable.

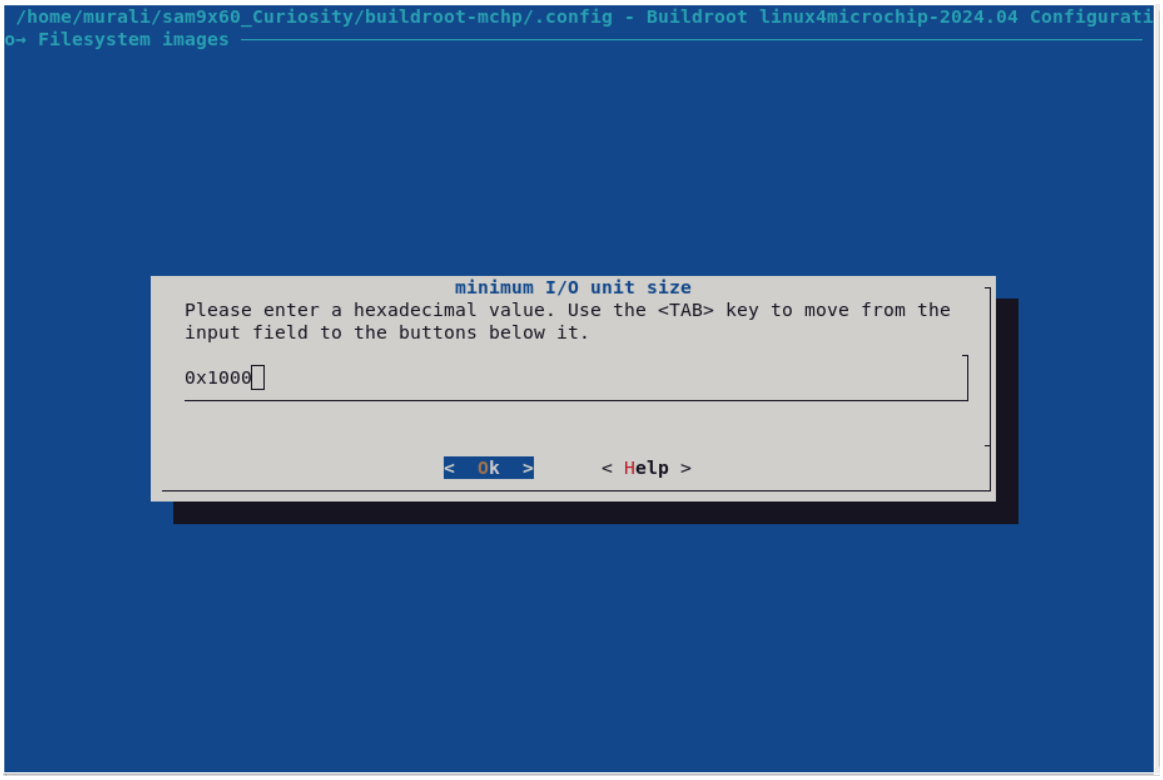
g. In the **Filesystem images** window, select **logical eraseblock size** and press Enter.

h. Update the **logical eraseblock size** to **0x3E000** and press Enter. The logical eraseblock size is calculated by subtracting two pages from the physical eraseblock size (LEB = PEB – 8192). Each of the two pages contains an EC header and a VID header.
Note: This calculation is applicable only for a NAND Flash which does not support sub-pages. For a NAND Flash supporting sub-pages, the calculation varies with the sub-page size.



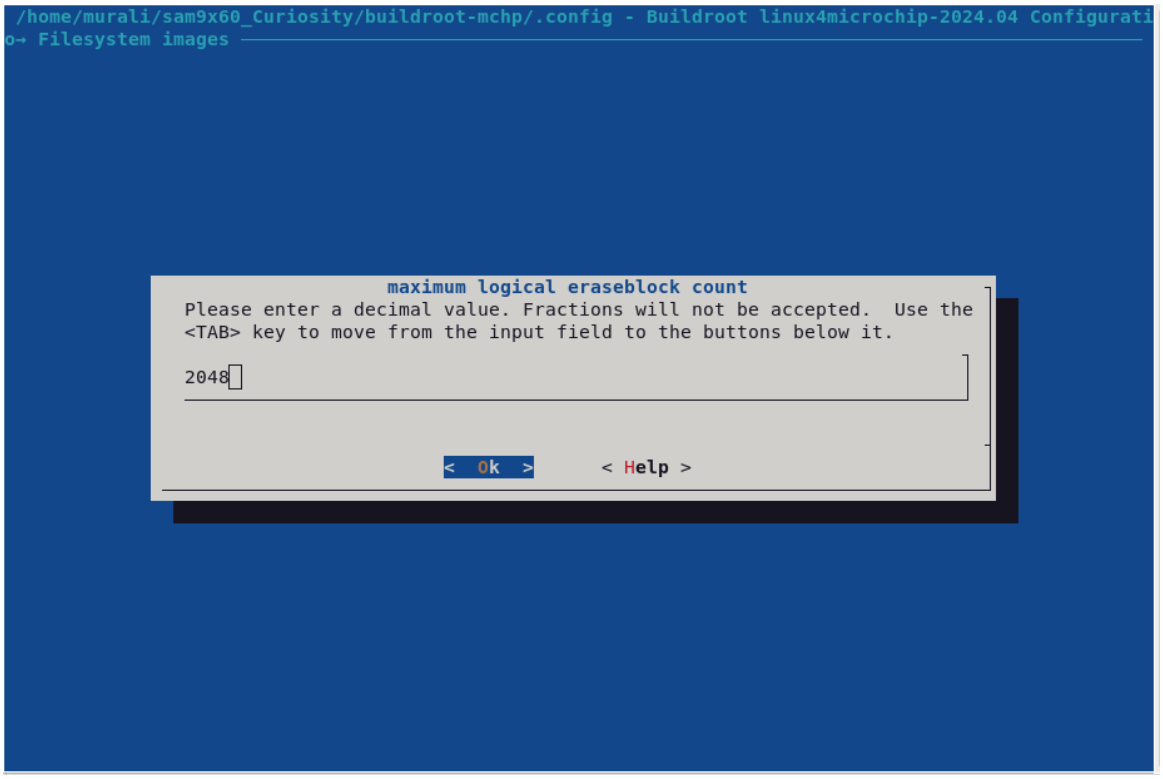
i. In the **Filesystem images** window, select **minimum I/O unit size** and press Enter.

j. Update the **minimum I/O unit size** to **0x1000** and press Enter. The minimum I/O unit size is equivalent to the NAND Flash device page size.



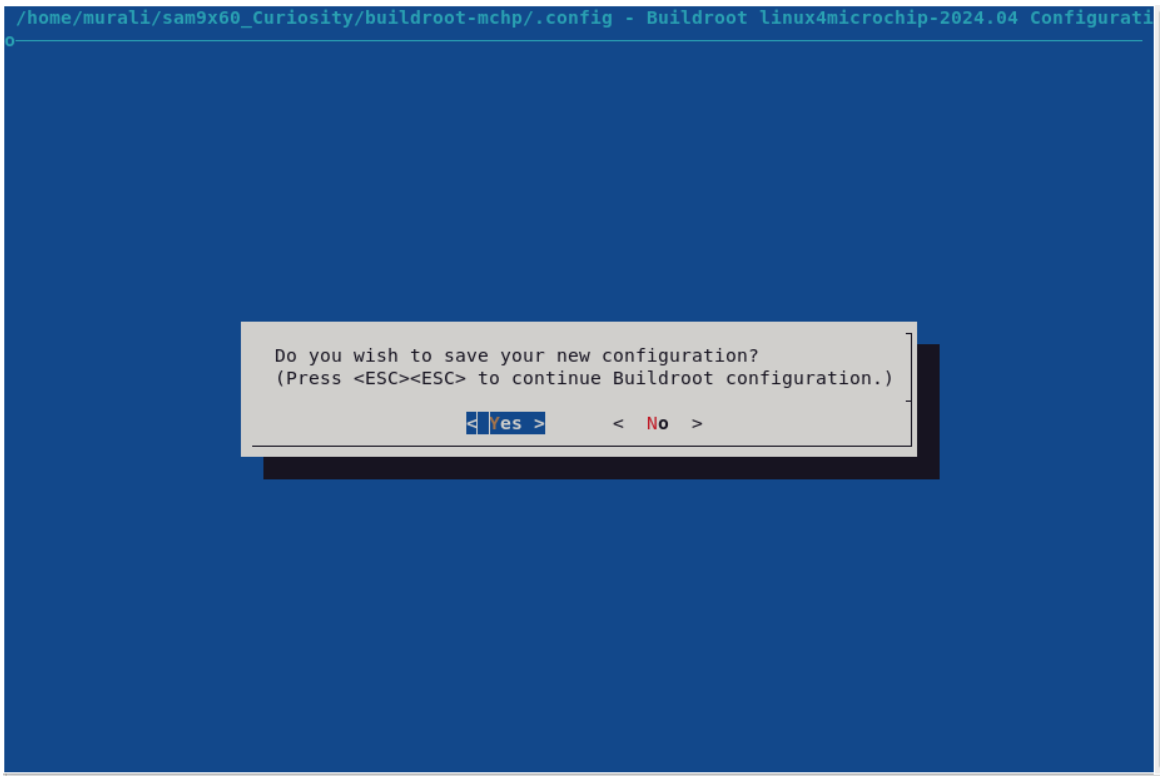
k. In the **Filesystem images** window, select **maximum logical eraseblock count** and press Enter.

l. Update the **maximum logical eraseblock count** to **2048** and press Enter. The maximum logical eraseblock count is the total number of blocks of the NAND Flash device.



m. Press ESC twice to exit the menu.

3. Select **Exit** and press Enter.
   a. Click **Yes**.



7. Modify the uboot-env and genimage files to boot from NAND Flash

1. In the explorer, navigate to **buildroot-external-microchip** > **board** > **microchip** > **sam9x60_curiosity**.
2. Open the **uboot-env.txt** file.
3. In the **bootargs** section, modify the **mtdparts** values as follows:
```
mtdparts=atmel_nand:256k(bootstrap)ro,512k(uboot)ro,256K(env),256k(env_redu
ndent),256k(spare),512k(dtb),6M(kernel)ro,-(rootfs) rootfstype=ubifs
ubi.mtd=7 root=ubi0:rootfs
```
4. Modify **bootcmd_boot** as follows:
```
Bootcmd_boot=nand read 0x21000000 0x00180000 0x00080000; nand read
0x22000000 0x00200000 0x00600000;bootz 0x22000000 – 0x21000000
```
5. Click **Save** and close the file.
6. In the same directory, open the **genimage.cfg** file.
7. Modify the **files** section as follows:
```
Files = {
“zImage”,
“at91-sam9x60_curiosity.dtb”,
“boot.bin”,
“u-boot.bin”
}
```
8. Click **Save** and close the file.

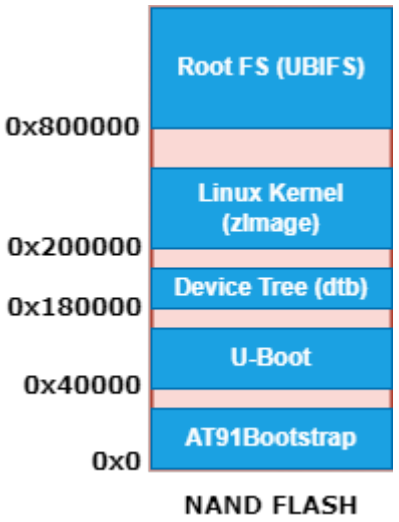
8. Final Build

After all required changes are made, the build can be performed.

1. Navigate to the buildroot-mchp directory:
   ```
   $ cd buildroot-mchp
   ```
2. Build the at91bootstrap with the following command:
   ```
   $ make
   ```
   The build will take a few hours to complete.
3. Once the build is complete, navigate to buildroot-mchp/output/Images.
4. Verify that the following files are generated:
   - at91bootstrap.bin
   - u-boot.bin
   - at91-sam9x60_curiosity.dtb
   - zImage
   - rootfs.ubi

9. NAND Flash Memory Map

Figure 9-1. NAND Flash Memory Map



10. Program the files into the target

1. Install the SAM-BA software version 3.8.1[(a)] on your PC:
   - Download and install SAM-BA from: www.linux4sam.org/bin/view/Linux4SAM/SoftwareTools#SAM_BA
   - Connect one end of the FTDI USB-to-TTL serial cable to the PC and the other end to connector J11 of the SAM9X60 curiosity board. Ensure that the cable black wire is connected to pin 1 (GND) on jumper J11. Open the serial terminal and establish a connection with the baud rate set to 115200.

   Note:

   a. The examples and command lines provided in this document are extracts from SAM-BA software version 3.8.1, but the user can use a later version. In such a case, information in this document may vary slightly.
2. By default, the SAM9X60 boot sequence tries to boot from NAND Flash. To learn about the boot process and first stage boot configuration, refer to the section “Boot Strategies” in the SAM9X60 data sheet (see Reference Documents).
3. Before powering up the board, remove jumper J4 (NAND_BOOT) to disable booting from NAND Flash.
4. Power up the board by connecting one end of the micro-B USB cable to board jumper J1 and the other end to the PC.
5. Press the START button and then the RESET button on the board.
6. When "RomBOOT" is displayed on the serial terminal, connect the J4 (NAND BOOT) jumper.
7. Flash the files into the NAND Flash memory:
   a. In the terminal, navigate to the buildroot-mchp/output/Images directory:

      ```
      $ cd buildroot-mchp/output/Images
      ```

   b. Erase the NAND Flash memory:

      ```
      $ sam-ba -p serial - b sam9x60-curiosity -a nandflash -c erase
      ```

   c. Write the bootstrap into the NAND Flash memory:

      ```
      sam-ba -p serial -b sam9x60-curiosity -a nandflash -c writeboot:
      at91bootstrap.bin
      ```

   d. Write u-boot into the NAND Flash memory:

      ```
      sam-ba -p serial -b sam9x60-curiosity -a nandflash -c write:u-
      boot.bin:0x00040000
      ```

   e. Write the device tree into the NAND Flash memory:

      ```
      sam-ba -p serial -b sam9x60-curiosity -a nandflash -c write:at91-
      sam9x60_curiosity.dtb:0x00180000
      ```

   f. Write the kernel into the NAND Flash memory:

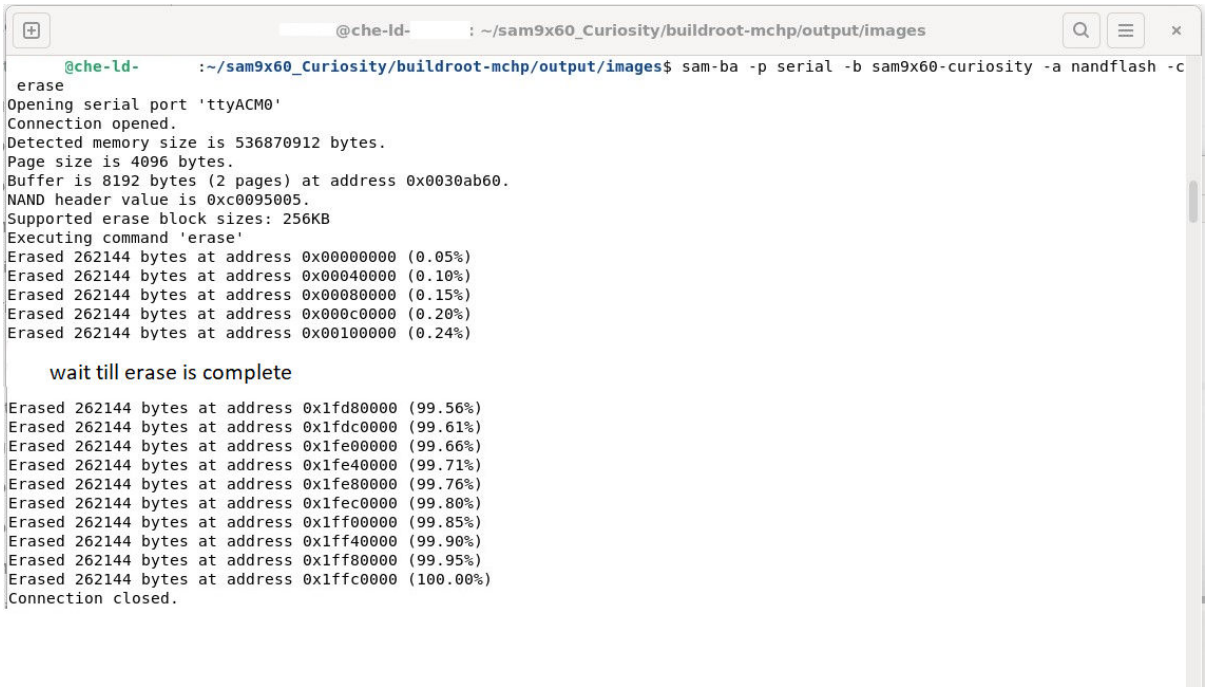
      ```
      sam-ba -p serial -b sam9x60-curiosity -a nandflash -c
      write:zImage:0x00200000
      ```

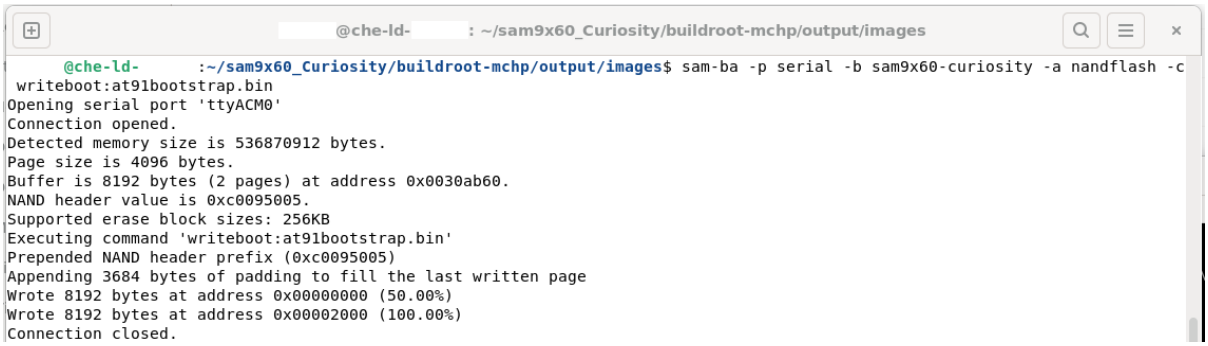
   g. Write rootfs into the NAND Flash memory:

      ```
      sam-ba -p serial -b sam9x60-curiosity -a nandflash -c
      write:rootfs.ubi:0x00800000
      ```

   h. Reset the board by pressing the reset (RESET) button.

Refer to the following screenshots:





```
@che-ld-      : ~/sam9x60_Curiosity/buildroot-mchp/output/images
@che-ld-      :~/sam9x60_Curiosity/buildroot-mchp/output/images$ sam-ba -p serial -b sam9x60-curiosity -a nandflash -c
 write:at91-sam9x60_curiosity.dtb:0x00180000
Opening serial port 'ttyACM0'
Connection opened.
Detected memory size is 536870912 bytes.
Page size is 4096 bytes.
Buffer is 8192 bytes (2 pages) at address 0x0030ab60.
NAND header value is 0xc0095005.
Supported erase block sizes: 256KB
Executing command 'write:at91-sam9x60_curiosity.dtb:0x00180000'
Appending 432 bytes of padding to fill the last written page
Wrote 8192 bytes at address 0x00180000 (22.22%)
Wrote 8192 bytes at address 0x00182000 (44.44%)
Wrote 8192 bytes at address 0x00184000 (66.67%)
Wrote 8192 bytes at address 0x00186000 (88.89%)
Wrote 4096 bytes at address 0x00188000 (100.00%)
Connection closed.
```

```
@che-ld-      : ~/sam9x60_Curiosity/buildroot-mchp/output/images
@che-ld-      :~/sam9x60_Curiosity/buildroot-mchp/output/images$ sam-ba -p serial -b sam9x60-curiosity -a nandflash -c
 write:zImage:0x00200000
Opening serial port 'ttyACM0'
Connection opened.
Detected memory size is 536870912 bytes.
Page size is 4096 bytes.
Buffer is 8192 bytes (2 pages) at address 0x0030ab60.
NAND header value is 0xc0095005.
Supported erase block sizes: 256KB
Executing command 'write:zImage:0x00200000'
Appending 752 bytes of padding to fill the last written page
Wrote 8192 bytes at address 0x00200000 (0.15%)
Wrote 8192 bytes at address 0x00202000 (0.30%)
Wrote 8192 bytes at address 0x00204000 (0.45%)
Wrote 8192 bytes at address 0x00206000 (0.60%)
```

wait till flashing is completed

```
Wrote 8192 bytes at address 0x00718000 (98.72%)
Wrote 8192 bytes at address 0x0071a000 (98.87%)
Wrote 8192 bytes at address 0x0071c000 (99.02%)
Wrote 8192 bytes at address 0x0071e000 (99.17%)
Wrote 8192 bytes at address 0x00720000 (99.32%)
Wrote 8192 bytes at address 0x00722000 (99.47%)
Wrote 8192 bytes at address 0x00724000 (99.62%)
Wrote 8192 bytes at address 0x00726000 (99.77%)
Wrote 8192 bytes at address 0x00728000 (99.92%)
Wrote 4096 bytes at address 0x0072a000 (100.00%)
Connection closed.
```

```
@che-ld-      : ~/sam9x60_Curiosity/buildroot-mchp/output/images
@che-ld-      :~/sam9x60_Curiosity/buildroot-mchp/output/images$ sam-ba -p serial -b sam9x60-curiosity -a nandflash -c write:rootfs.ubi:0x00800000
Opening serial port 'ttyACM0'
Connection opened.
Detected memory size is 536870912 bytes.
Page size is 4096 bytes.
Buffer is 8192 bytes (2 pages) at address 0x0030ab60.
NAND header value is 0xc0095005.
Supported erase block sizes: 256KB
Executing command 'write:rootfs.ubi:0x00800000'
Wrote 8192 bytes at address 0x00800000 (0.01%)
Wrote 8192 bytes at address 0x00802000 (0.01%)
Wrote 8192 bytes at address 0x00804000 (0.02%)
Wrote 8192 bytes at address 0x00806000 (0.03%)
```

wait till flashing is completed

```
Wrote 8192 bytes at address 0x07582000 (99.79%)
Wrote 8192 bytes at address 0x07584000 (99.79%)
Wrote 8192 bytes at address 0x07586000 (99.80%)
Wrote 8192 bytes at address 0x07588000 (99.81%)
Wrote 8192 bytes at address 0x0758a000 (99.81%)
Wrote 8192 bytes at address 0x0758c000 (99.82%)
Wrote 8192 bytes at address 0x0758e000 (99.83%)
Wrote 4096 bytes at address 0x07590000 (99.83%)
Skipped 47 empty page(s) at address 0x07591000
Connection closed.
```

11. Revision History

11.1 Rev. A - 11/2024

First issue.

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0131-6

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.