Working with U-boot S4SK

Rev. 1.00

April/16/2024

This document describes overview about U-Boot for S4SK

References

* [Embedded Linux Projects Using Yocto Project Cookbook.pdf](https://github.com/satishkumar1221/Project-Garuda/blob/master/Embedded%20Linux%20Projects%20Using%20Yocto%20Project%20Cookbook.pdf)
* [Porting U-Boot and Linux on new ARM boards: a step-by-step guide](https://elinux.org/images/2/2a/Schulz-how-to-support-new-board-u-boot-linux.pdf)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev. | Modified points | Approver | Checker | Author |
| 1.0 | Newly created | --  (April/--/2024) | --  (April/--/2024) | Nguyen Tran  (April/16/2024) |
|  |  |  |  |  |

Table of Contents

[1. Building the U-boot approach 2](#_Toc1692892365)

[1.1 External source development using Yocto 3](#_Toc1419502026)

[1.2 Working directory development using Yocto 3](#_Toc1091449712)

[1.3 External development 4](#_Toc1554258820)

[2. How to update the U-boot for S4SK 4](#_Toc1346314454)

[3. Working with U-Boot S4SK 6](#_Toc1623036474)

[3.1 List of working files with S4SK 7](#_Toc788973236)

[3.2 Explore file specific 7](#_Toc1666724206)

[3.3 Define u-boot environment variable for S4SK 8](#_Toc1648435903)

# Building the U-boot approach

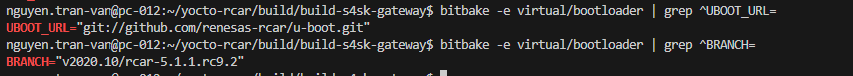
The build machine in this case is: **192.168.2.25**

The host machine of S4SK in this case is: **192.168.2.23**

## External source development using Yocto

We will use the Yocto build system from a local directory by cloning a local copy of the source used in the reference design and configuring our project to use it as an external source. **We will then develop from it, extract the patches, and add them to a bbappend file on our BSP layer**

1. *Find the upstream Git repository of u-boot S4SK:*



1. *Clone source code:*

**Build machine terminal:**

$ git clone https://github.com/renesas-rcar/u-boot.git -b v2020.10/rcar-5.1.1.rc9.2

1. *Config local.conf file*

To configure our *conf/local.conf* file to work from the cloned source, modify it as follows:

INHERIT += "externalsrc"

EXTERNALSRC:pn-myrecipe = "/path/to/my/source/tree"

EXTERNALSRC\_BUILD:pn-myrecipe = "/path/to/my/source/tree"

Please refer [**externalsrc.bbclass**](https://git.yoctoproject.org/poky/plain/meta/classes/externalsrc.bbclass)file in Yocto source tree to get more information.

Apply to S4SK board:

**local.conf file:**

INHERIT += "externalsrc"

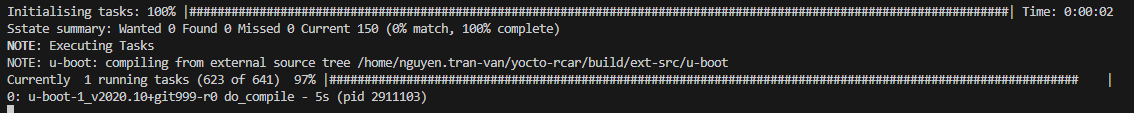
EXTERNALSRC\_pn-u-boot = "/home/nguyen.tran-van/yocto-rcar/build/ext-src/u-boot"

EXTERNALSRC\_BUILD\_pn-u-boot = "/home/nguyen.tran-van/yocto-rcar/build/ext-src/uboot-build"

1. *Build.*

**Build machine terminal:**

$ bitbake virtual/bootloader

We will see: **NOTE: u-boot: compiling from external source tree**

The newly compiled U-Boot image is available under build path: */home/nguyen.tran-van/yocto-rcar/build/ext-src/uboot-build and /tmp/deploy/images/s4sk/ (in this case)*

This approach can be applied to any recipes like Linux, ATF, etc if we need to custom. (Recommended in development phase)

## Working directory development using Yocto

A typical workflow when working on a small modification would be:

1. Start the U-Boot package compilation from scratch:

**Build machine terminal:**

$ bitbake -c cleanall virtual/bootloader

This will erase the build folder, shared state cache, and downloaded package source.

2. Start a development shell:

**Build machine terminal:**

$ bitbake -c devshell virtual/bootloader

This will fetch, unpack, and patch the U-Boot sources and spawn a new shell with the environment ready for U-Boot compilation. The new shell will change to the U-Boot build directory, which contains a local Git repository.

3. Perform your modifications on the local Git repository.

4. Leave the devshell open and use a different terminal to compile the source without erasing our modifications:

**Build machine terminal:**

$ bitbake -C compile virtual/bootloader

Note the capital C. This invokes the compile task but also all the tasks that follow it.

The newly compiled U-Boot image is available under /tmp/deploy/images/s4sk/

## External development

1. Clone the source u-boot code that is supported for S4SK board:

**Build machine terminal:**

$ git clone https://github.com/renesas-rcar/u-boot.git -b v2020.10/rcar-5.1.1.rc9.2

2. Build:

**Build machine terminal:**

$ cd u-boot

# Use a cross compile by your choice. In this case use **aarch64-poky-linux-**

$ make ARCH=arm CROSS\_COMPILE=aarch64-poky-linux- distclean -j$nproc

$ make ARCH=arm CROSS\_COMPILE=aarch64-poky-linux- r8a779f0\_s4sk\_defconfig -j$nproc

$ make ARCH=arm CROSS\_COMPILE=aarch64-poky-linux- -j$nproc

# We will get **u-boot-elf.srec** file after build success. Let rename that is **u-boot-elf-s4sk.srec** for burn.

$ cp u-boot-elf.srec u-boot-elf-s4sk.srec



# How to update the U-boot for S4SK

Debug interface of S4SK:

**Host terminal:**

# Open /dev/ttyUSB0 use picocom

$ picocom -b 921600 -d 8 -y n -p 1 -f n /dev/ttyUSB0

# Exit picocom us e : Ctrl A + Ctrl X

**Note: Exit picocom before it burns.**

**How to flash/update by using "Renesas BSP ROM Writer"**

Download

**Host terminal:**

$ git clone https://github.com/morimoto/renesas-bsp-rom-writer

Download ICUMX loader and copy the built binaries (u-boot-elf-s4sk.srec, tee-s4sk.srec and bl31-s4sk.srec)

[Link](https://www.renesas.com/us/en/icumx-loader-and-flash-writer-package-r-car-s4-starter-kit)

Flashing loader

**Host terminal:**

$ cd ${PATH}/ICUMX\_Loader\_and\_Flashwriter\_Package\_for\_R-Car\_S4\_Starter\_Kit\_SDKv3.16.xx/

$ ${renesas-bsp-rom-writer}/board/s4\_sk/linux/sdk\_writer

**Apply to the S4SK:**

1. Copy u-boot binary file that is used to burn to **ICUMX\_Loader\_and\_Flashwriter\_Package\_for\_R-Car\_S4\_Starter\_Kit\_SDKv3.16.0** folder of host machine of S4SK board:

**Build machine terminal:**

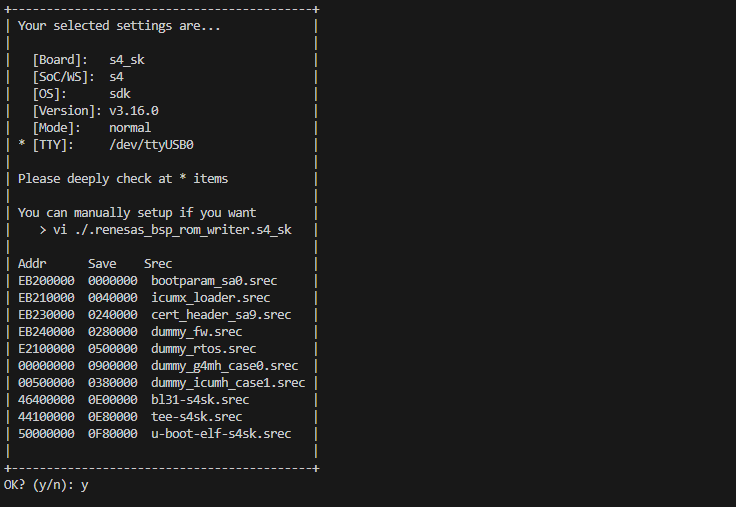
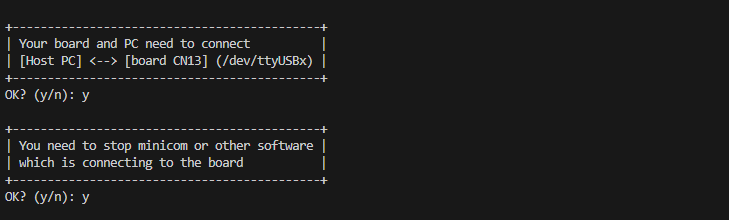
$ scp u-boot-elf-s4sk.srec <**username**>@192.168.2.23:/path/to/ICUMX\_Loader\_and\_Flashwriter\_Package\_for\_R-Car\_S4\_Starter\_Kit\_SDKv3.16.0

2. Run burn script and follow steps for burn proceed:

**Host terminal:**

$ cd /path/to/ICUMX\_Loader\_and\_Flashwriter\_Package\_for\_R-Car\_S4\_Starter\_Kit\_SDKv3.16.0

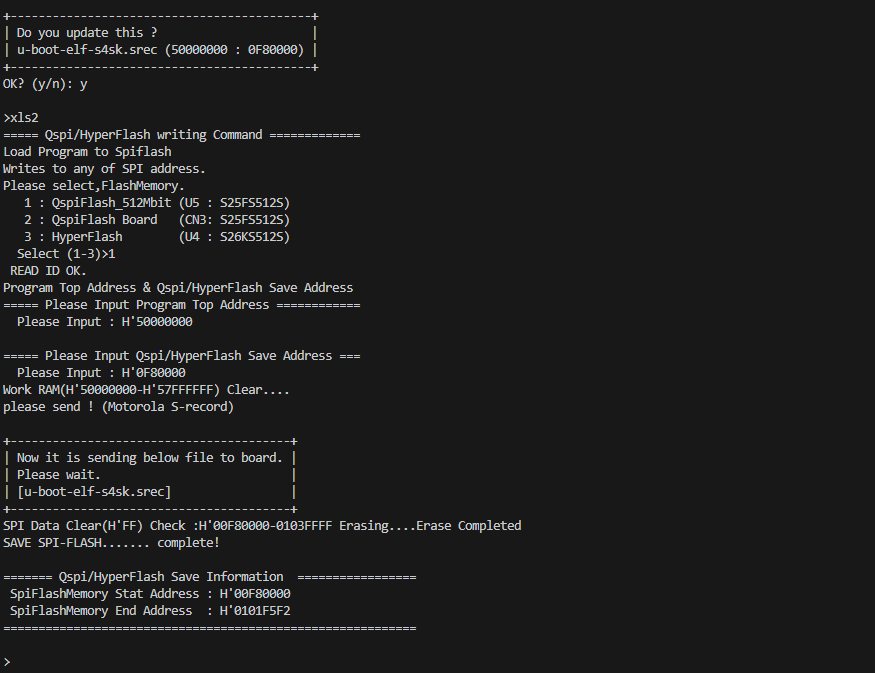
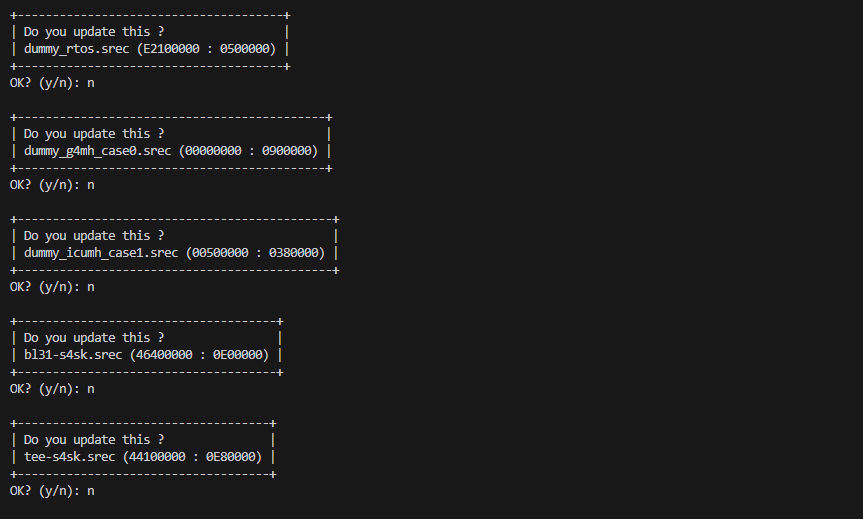
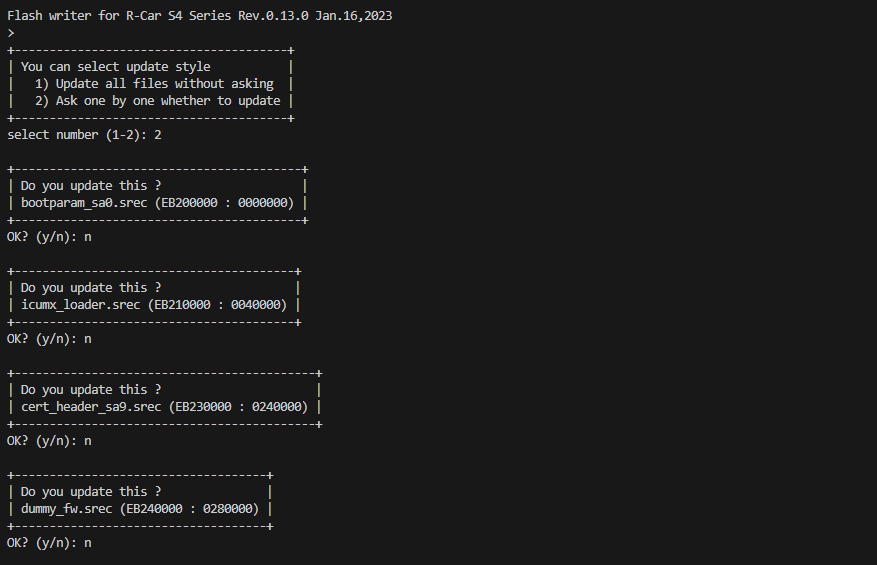
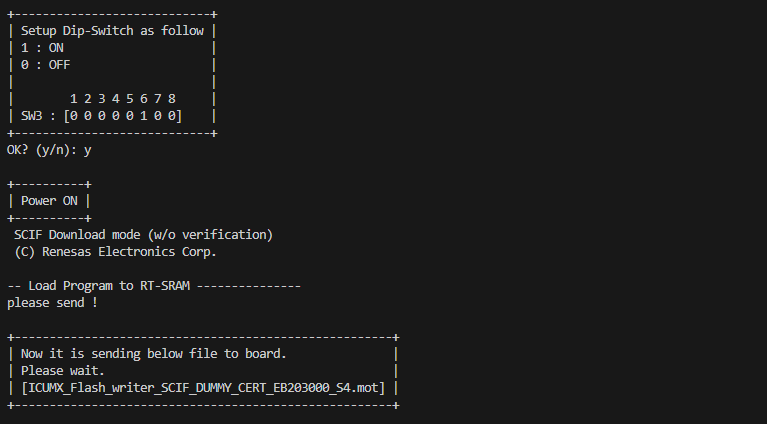
$ ../renesas-bsp-rom-writer/board/s4\_sk/linux/sdk\_writer



3. Switch board to burn mode

**Host terminal:**

$ cpld-control-1.8.11082022 -w S4SK 276697 0x0008 0x00000080804922BF 0x0024 0x01



4. Switch board to boot mode:

**Host terminal:**

$ cpld-control-1.8.11082022 -w S4SK 276697 0x0008 0x00000080804922A9 0x0024 0x01

How to reset board:

**Host terminal:**

$ cpld-control-1.8.11082022 -w S4SK 276697 0x0024 1

# Working with U-Boot S4SK

## List of working files with S4SK

|  |  |  |
| --- | --- | --- |
| **No** | **Path** | **Remarks** |
| **1** | u-boot/arch/arm/dts/r8a779f0.dtsi | R-Car S4 (R8A779F0) SoC specific device tree file |
| **2** | u-boot/arch/arm/dts/r8a779f0-s4sk.dts | S4SK board specific device tree file |
| **3** | u-boot/board/renesas/s4sk/s4sk.c | S4SK board support. |
| **4** | u-boot/configs/r8a779f0\_s4sk\_defconfig | S4SK defconfig file |
| **5** | u-boot/include/configs/rcar-gen4-common.h | S4SK board header file. This file is R-Car Gen4 common configuration file. |

## Explore file specific

**Explore the S4SK defconfig file**: [r8a779f0\_s4sk\_defconfig](https://github.com/renesas-rcar/u-boot/blob/v2020.10/rcar-5.1.1.rc9.2/configs/r8a779f0_s4sk_defconfig)

* CONFIG\_BOOTARGS: This config support config default u-boot environment variable **bootargs.**

**Example:** CONFIG\_BOOTARGS="rw root=/dev/mmcblk0p1 rootwait ignore\_loglevel cma=560M"

It set bootargs =rw root=/dev/mmcblk0p1 rootwait ignore\_loglevel cma=560M

With:

* + Rootfs located in mmc0, partition 1 (SD Card interface of S4SK), read/write file system.
  + Rootwait: wait for rootfs availble then mount it
  + [CMA memory](https://developer.toradex.com/software/linux-resources/linux-features/contiguous-memory-allocator-cma-linux/) allocation is 560MB, this memory region is used to GPU, VPU or etc. Usually, it configured around 25% RAM size, also depending on application.
* CONFIG\_DEFAULT\_FDT\_FILE: This config supports config default device tree that we want to use.

**Example**: CONFIG\_DEFAULT\_FDT\_FILE="r8a779f0-s4sk.dtb"

It set default dtb file is r8a779f0-s4sk.dtb

* CONFIG\_CMD\_BOOTZ=y: This config support for boot zImage
* CONFIG\_CMD\_MMC=y: This config support for mmc boot
* CONFIG\_CMD\_PING=y: This config enables ping command
* CONFIG\_CMD\_EXT4=y: This config enables ext4load command
* CONFIG\_CMD\_FAT=y: This config enables fatload command
* CONFIG\_ENV\_IS\_IN\_SPI\_FLASH=y: This config direct u-boot environment variable store in FLASH (in case of S4SK)
* CONFIG\_SYS\_PROMPT: This config support for sys-prompt

**Example**: CONFIG\_SYS\_PROMPT="U-Boot S4SK # "

**U-boot S4SK terminal:**



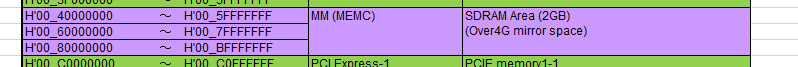
* CONFIG\_CMD\_TFTPBOOT=y: This config enables tftpboot command
* CONFIG\_CMD\_PXE=y: This config enables [pxe](https://github.com/u-boot/u-boot/blob/master/doc/README.pxe) boot support

More information can be referring documentation in u-boot source tree:

* [README](https://github.com/u-boot/u-boot/blob/master/README)
* [doc](https://github.com/u-boot/u-boot/tree/master/doc)

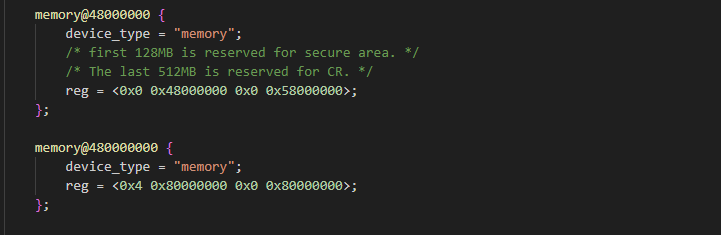
**Explore the DRAM of S4SK:**

Refer: R8A779F0 SoC specific



Base address of SDRAM is 0x40000000

Refer memory node in S4SK device tree file: [r8a779f0-s4sk.dts](https://github.com/renesas-rcar/u-boot/blob/v2020.10/rcar-5.1.1.rc9.2/arch/arm/dts/r8a779f0-s4sk.dts)



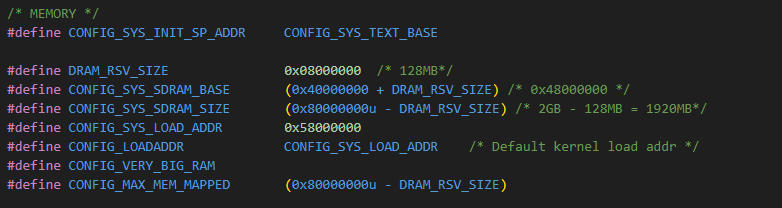
This describes 2 bank RAM with total size 3456MB

Bank1: Base address is 0x48000000 and size is 1408MB (58000000 in hex)

Bank2: Base address = 0x480000000 and size is 2048MB (80000000 in hex)

**Explore the board header file**: [rcar-gen4-common.h](https://github.com/renesas-rcar/u-boot/blob/v2020.10/rcar-5.1.1.rc9.2/include/configs/rcar-gen4-common.h)

Let concentrate on memory config:



SDRAM\_BASE is 0x40000000 but reserve first 128MB for secure area, so Rcar's DRAM memory always starts at 0x48000000, and u-boot shall have relocated itself to higher in memory by the time this value is used.

The default kernel load address is set to a 256MB offset (CONFIG\_SYS\_LOAD\_ADDR 0x58000000).

**Note:** Please do not conflict load address and bank 1 RAM size value that defined at memory node

We can use region 0x48000000 – 0x58000000 (256MB) for device tree, device tree overlay, script, init ramdisk, or etc (fdtaddr, dtboaddr, pxefile\_addr\_r, scriptaddr, initrd\_addr, splashimage, etc)

## Define u-boot environment variable for S4SK

We can define FDT at 0x48000000 or above (0x48000000 – 0x58000000 = 256MB)

*fdtaddr=0x48000000*

*fdt\_addr\_r=0x48000000*

We can define DTBO at 0x49000000 or above (0x48000000 - 0x49000000 = 16MB)

*dtboaddr=0x49000000*

*fdtoverlay\_addr\_r=0x49000000*

We can define pxefile\_addr\_r at 0x4A000000 or above (0x49000000 - 4A0000000 = 16MB)

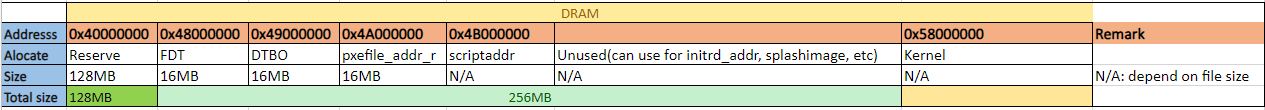
*pxefile\_addr\_r=0x4A000000*

We can define scriptaddr at 0x4B000000 or above

*scriptaddr=0x4B000000*

We can define more if needed (initrd\_addr, splashimage, etc)

Make sure we have enough space to grow the base file without overlapping anything.



Define u-boot environment variable for S4SK approach:

1. Define manually during boot time

**U-boot S4SK terminal:**

$ setenv fdt\_addr\_r 0x48000000

$ setenv fdtaddr 0x48000000

$ setenv dtboaddr 0x49000000

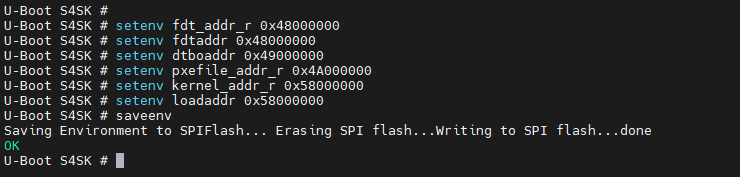
$ setenv fdtoverlay\_addr\_r 0x49000000

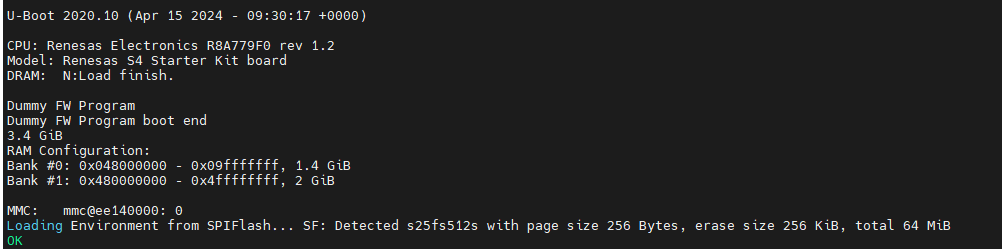
$ setenv pxefile\_addr\_r 0x4A000000

$ setenv kernel\_addr\_r 0x58000000

$ setenv loadaddr 0x58000000

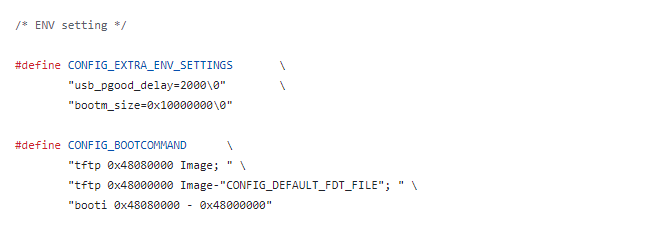
$ saveenv

Some env variables are stored in FLASH of S4SK, after power on reset, u-boot loads some env form FLASH and uses them for target boot.

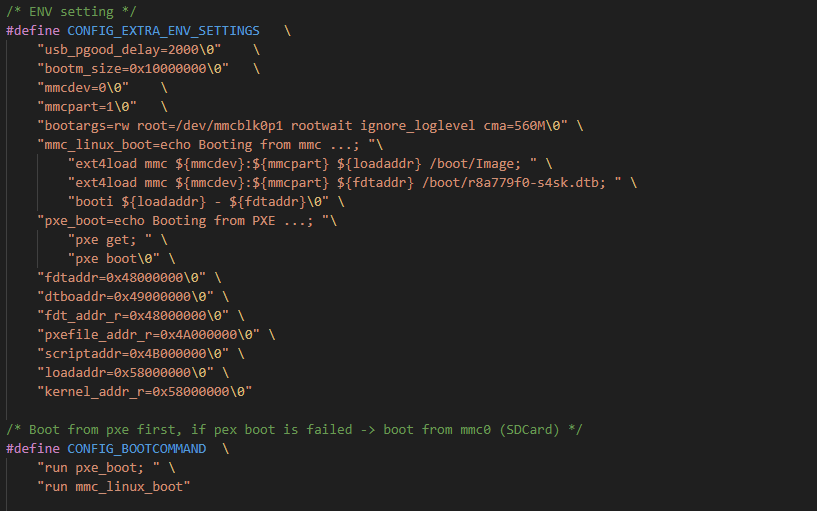


1. Define as default env variable

Open file **rcar-gen4-common.h** in u-boot S4SK source tree:

Go to ENV Setting This is origin default env form Renesas.

Let define some default env variable follow CONFIG\_EXTRA\_ENV\_SETTINGS

We can define more env variables if needed.

We also define default **bootcmd** variable follow CONFIG\_BOOTCOMMAND

In this case, **bootcmd** is pxe boot, and mmc0 (S4SK SD card). If pxe boot failed, u-boot trying to boot next command is mmc0.

To reset default env we use command:

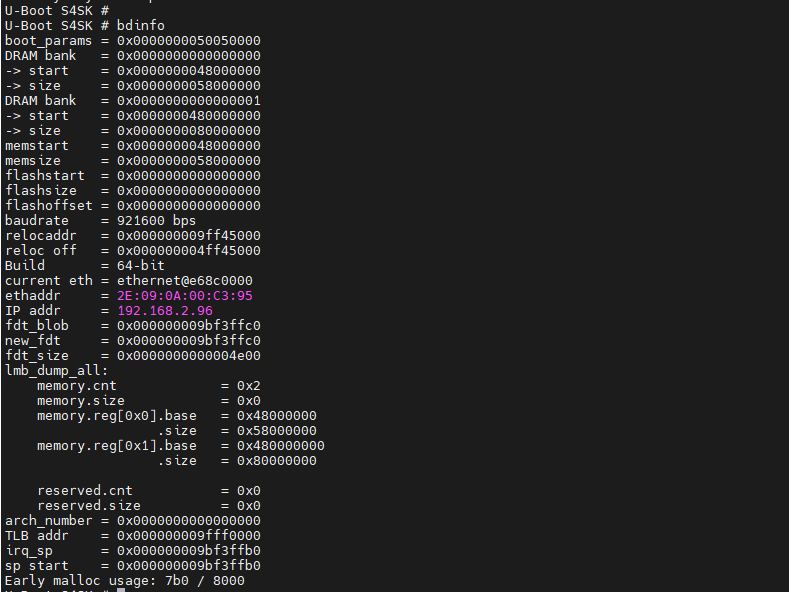
**U-boot S4SK terminal:**

$ env default –a

Default env available is some env that defined at CONFIG\_EXTRA\_ENV\_SETTINGS

Use [**bdinfo**](https://docs.u-boot.org/en/latest/usage/cmd/bdinfo.html)command in u-boot to display information of board:

Enable this command if u-boot not yet built-in: CONFIG\_CMD\_BDI=y

We can easily see more information about our board.