**Build and Boot U-Boot and Linux on a Raspberry Pi 3 Model B+**

[Linux](https://www.thegoodpenguin.co.uk/blog/category/tech-blog/linux/) [Tech Blog](https://www.thegoodpenguin.co.uk/blog/category/tech-blog/)

With more than [30 million](https://twitter.com/EbenUpton/status/1205646606504275968) units in existence, the [Raspberry Pi](https://www.raspberrypi.org/) provides for an accessible and low-cost way to play with an [ARMv8](https://en.wikipedia.org/wiki/ARM_architecture#64/32-bit_architecture) embedded device using the latest and greatest upstream sources. In this post we’ll walk through the steps required to build U-Boot, the Linux kernel and a filesystem, as well as the steps required for preparing an SD card and booting it.

Let’s start by getting a toolchain that will build 64bit ARMv8 binaries, we’ll grab the latest [Linaro toolchain](https://releases.linaro.org/components/toolchain/binaries/latest-7/aarch64-linux-gnu/), unpack it and temporarily put it in our system path:

$ wget https://releases.linaro.org/components/toolchain/binaries/latest-7/aarch64-linux-gnu/gcc-linaro-7.5.0-2019.12-x86\_64\_aarch64-linux-gnu.tar.xz

$ tar -xf gcc-linaro-7.5.0-2019.12-x86\_64\_aarch64-linux-gnu.tar.xz -C ~/tools

$ export PATH=~/tools/gcc-linaro-7.5.0-2019.12-x86\_64\_aarch64-linux-gnu/bin:$PATH

Even though the Raspberry Pi can [directly boot a kernel](https://www.raspberrypi.org/documentation/configuration/config-txt/boot.md), we’d still like to boot via U-Boot. Thus let’s grab the latest version of mainline U-Boot, configure it for our Raspberry Pi and build a u-boot.bin binary:

$ git clone ~~git://git.denx.de/u-boot.git~~ https://source.denx.de/u-boot/u-boot.git # v2021.01-rc2-121-g5b8991c667f7

$ cd u-boot

$ make ARCH=arm CROSS\_COMPILE=aarch64-linux-gnu- rpi\_arm64\_defconfig

$ make ARCH=arm CROSS\_COMPILE=aarch64-linux-gnu- -j$(nproc)

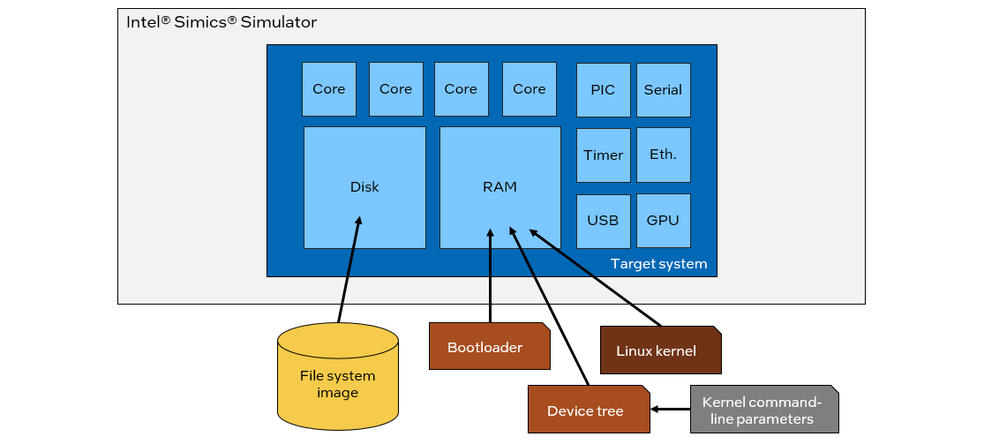
$ cd ..

make ARCH=arm CROSS\_COMPILE=aarch64-linux-gnu- -j$(nproc) để tạo ra u-boot.bin

make menuconfig để chỉnh lại các config (support ethernet)

Có 3 cách khởi động linux kernel: [Loading and Starting a Linux\* Kernel in Five Different Ways - Intel Community](https://community.intel.com/t5/Blogs/Products-and-Solutions/Software/Loading-and-Starting-a-Linux-Kernel-in-Five-Different-Ways/post/1571900)

Simple Platform Direct Linux Kernel Boot

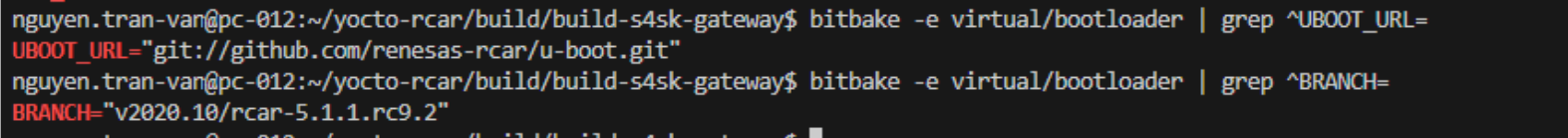


Đơn giản là load hết vào ram, sau đó load filesystem image

**Full process từ burn u-boot tới load filesystem vào s4sk  
Build U-boot (thực hiện ở máy build để có đủ tool):**

Dựa vào các step trên để build u-boot với branch sau:

git clone [https://github.com/renesas-rcar/u-boot.git -b v2020.10/rcar-5.1.1.rc9.2](https://github.com/renesas-rcar/u-boot.git -b%20v2020.10/rcar-5.1.1.rc9.2)

(làm sao để biết branch nào? Kiểm tra trong yocto build (yocto build sẽ nói ở phần sau) như sau

)

**Config local.conf file dựa trên poky version (poky là từ yocto project, đọc sau) (chỉ config khi sử dụng yocto để build u-boot)**

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.

Note that each Poky version has different syntax (In this case Poky version 4)

Apply to S4SK board: (Poky version 3)

**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"

**How to find out Yocto version?**

Check out this file it gives you full details about Yocto version

vim $POKY-DIR/meta-poky/conf/distro/poky.conf

You will get info like:

DISTRO = "poky"

DISTRO\_NAME = "Poky (Yocto Project Reference Distro)"

DISTRO\_VERSION = "2.7.2"

DISTRO\_CODENAME = "warrior"

SDK\_VENDOR = "-pokysdk"

SDK\_VERSION = "${@d.getVar('DISTRO\_VERSION').replace('snapshot-${DATE}', 'snapshot')}"

.....

.....

Now you will know the version you are actually using.

**To find the kernel version you are using:**

bitbake -e virtual/kernel | grep "^PV"

Sau khi config xong thì make bằng cross-compiler (khi compile độc lập với yocto) ở trên hoặc xài lệnh: bitbake virtual/bootloader (chỉ dùng khi đang sử dụng yocto để build)

Khi make độc lập với yocto thì sẽ ra output là file **u-boot-elf.srec,** sửa lại thành **u-boot-elf-s4sk.srec** để burn ở bước sau:

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

**Nạp u-boot bằng tool unofficial "Renesas BSP ROM Writer" (chuyển file qua máy host dùng lệnh scp)**

Ví dụ về dùng scp để chuyển file : 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](http://0.0.0.23/path/to/ICUMX_Loader_and_Flashwriter_Package_for_R-Car_S4_Starter_Kit_SDKv3.16.0)

Tải tool , folder của tool lúc này sẽ là ${renesas-bsp-rom-writer} : git clone <https://github.com/morimoto/renesas-bsp-rom-writer>

Download ICUMX loader (giải nén hết vào cùng thư mục) and copy the built binaries (u-boot-elf-s4sk.srec, tee-s4sk.srec and bl31-s4sk.srec) có được ở trên phần build u-boot

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

Ta giải nén hết ICUMX loader và đem hết 3 file của u-boot vào thư mục đó, sau đó đảm bảo ta đứng ở thư mục chứa toàn bộ file này:

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

Và từ đó trỏ vào file sdk\_writer trong tool:

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

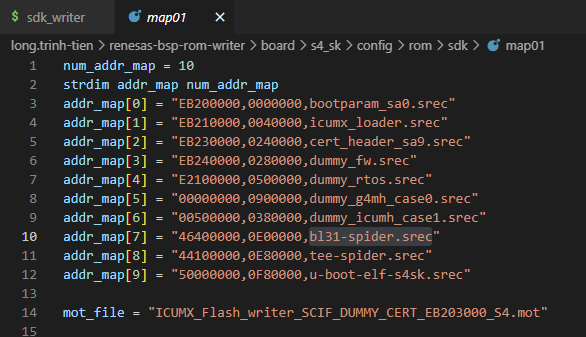


A screenshot of a computer program

Description automatically generated

Lưu ý ở bước này tool gốc bị 1 lỗi nhỏ, sửa lại như sau:

Ở tool rom-writer tại /home/long.trinh-tien/renesas-bsp-rom-writer/board/s4\_sk/config/rom/sdk/map01



Sửa spider thành s4sk tại addr\_map[7] và addr\_map[8]

A black background with white text

Description automatically generated

Ngay sau khi ấn y tại Power OFF thì bật terminal khác lên chạy cmd sau để vào burn mode (nhớ kêu mọi người thoát picocom, minicom hết nếu ko sẽ bị timeout hoặc crash):

cpld\_control\_v1.0 -w S4SK [ftdi://ftdi:2232:276697/2](ftdi://ftdi:2232:276697) 0x0008 0x00000080804922BF 0x0024 0x01

A screenshot of a computer program

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Sau đó chuyển qua terminal khác chạy cmd vào boot mode:

cpld\_control\_v1.0 -w S4SK [ftdi://ftdi:2232:276697/2](ftdi://ftdi:2232:276697) 0x0008 0x00000080804922A9 0x0024 0x01

More information:

Mình chia sẻ thêm một số thông tin về burn boot

**Manualy workflow:**  
Power off -> Gạt SW để chọn burn mode -> Turn on board -> Board đã vào burn mode và tiến hành burn -> Burn xong thì Power off -> Gạt SW để chọn boot mode và boot board

**Tooling workflow:**  
Dùng cpld\_tool để chuyển sang burn mode -> Board đã vào burn mode và tiến hành burn -> Burn xong thì dùng cpld\_tool để chuyển sang boot mode và boot board

Ở command:  
$ cpld\_control\_v1.0 -w S4SK ftdi://ftdi:2232:276697/2 0x0008 0x00000080804922BF 0x0024 0x01  
Command này sẽ làm 2 việc đó là chuyển board sang burn mode và reset board

$ cpld\_control\_v1.0 -w S4SK ftdi://ftdi:2232:276697/2 0x0008 0x00000080804922A9 0x0024 0x01  
Command này sẽ làm 2 việc đó là chuyển board sang boot mode và reset board

Nên việc power off hay power on ở tooling workflow có thể skip

Tool cpld này sẽ giúp chúng ta đổi mode mà không phải vào lab để gạt DIP SW trên board

**Load image**

Trên máy 2.23 mình note lại một số đường dẫn:

* rootfs ở đường dẫn:  /data/nfs/s4sk/
  + -> tìm file rcar-image-gateway-s4sk-20240502032430.rootfs.tar.bz2 (hoặc lấy file mà ta đã build trong yocto)
  + Tạo thư mục long.trinh-tien tại s4sk/ và chạy command để bung rootfs : tar -xjf rcar-image-gateway-s4sk-20240502032430.rootfs.tar.bz2
* Image + dtb ở đường dẫn:  /data/tftpboot/s4sk/01-2e-09-0a-00-c3-95/ (hoặc lấy file đã build trong yocto)
  + Copy trong default ra để tạo 1 thư mục riêng long.trinh-tien và copy vào đó (optional)
* Pxe boot (chưa rõ cách sử dụng) mọi người edit file 01-2e-09-0a-00-c3-95 ở đường dẫn:  /data/tftpboot/pxelinux.cfg/

Sau khi setup xong quay lại u-boot set các environment sau:

setenv serverip 192.168.2.23

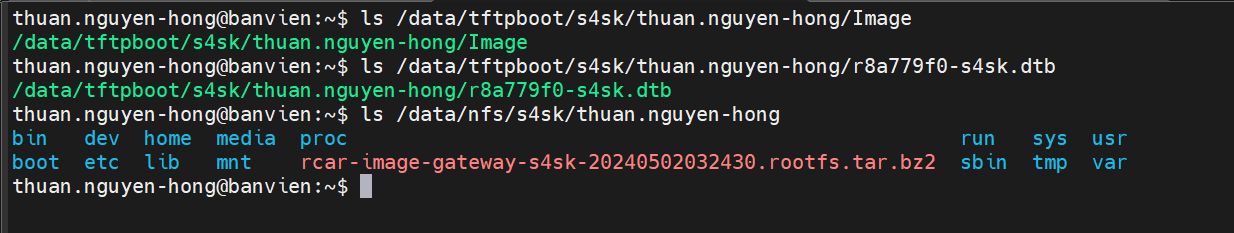
setenv ipaddr 192.168.2.95

setenv bootargs "rw root=/dev/nfs nfsroot=${serverip}:**/data/nfs/s4sk/long.trinh-tien**,nfsvers=4 ip=${ipaddr}:::::tsn0 ignore\_loglevel cma=560M"

setenv bootcmd 'tftpboot ${loadaddr} **s4sk/long.trinh-tien/Image**; tftpboot ${fdtaddr} **s4sk/long.trinh-tien/r8a779f0-s4sk.dtb**;booti ${loadaddr} - ${fdtaddr}'

saveenv

reset



giải thích các env trong U-boot:

+ipaddr : IP của board S4SK trong U-boot

+serverip: IP của server TFTP

+bootargs: dùng để boot board khi vào kernel

+bootcmd: sẽ được U-boot chạy khi boot board

Sau đó image của ta sẽ tự động được load