**Boot using u-boot**

**SD card**

Work flow:

U-boot <- run bootcmd <- run bootcmd\_sd0 <- setenv bootargs ${bootargs\_sd0};ext4load mmc 0:1 0x48080000 /boot/Image;ext4load mmc 0:1 0x48000000 /boot/${dtb};run booti\_cmd

1. Load image(kernel) và dtb vào địa chỉ trên RAM
2. Chạy booti (boot image) để khởi chạy kernel với device tree đi kèm (sau khi boot xong sẽ chạy bootargs)

A screen shot of a computer screen

Description automatically generated

Check thẻ sd, ở đây là mmcblk1p1, lúc gắn vào nó sẽ ra /dev/mmcblk1p1

Câu lệnh sau dùng để load image và device tree lên ram

ext4load mmc <dev>[:partition] <loadAddress> <bootfilename>

ext4load(hoặc load) mmc 0:1(thẻ số 0, part 1 tức là thẻ mmcblk1, part mmcblk1p1) 0x48080000 (check địa chỉ ram trên datasheet) H3/boot/Image(đường dẫn image trong thẻ sd)

[Lệnh setenv booti\_cmd 'booti 0x48080000 - 0x48000000' trong U-Boot đang thiết lập một biến môi trường mới có tên là booti\_cmd với giá trị là 'booti 0x48080000 - 0x48000000'1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html).

[Cụ thể, 'booti 0x48080000 - 0x48000000' là một lệnh booti, được sử dụng để khởi động một kernel Linux ở định dạng ‘Image’ phẳng hoặc nén1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html). Dưới đây là ý nghĩa của từng phần:

* [booti: Là lệnh khởi động một kernel Linux1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html).
* [0x48080000: Đây là địa chỉ của hình ảnh kernel1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html).
* [-: Ký hiệu này được sử dụng khi bạn muốn khởi động một kernel với một cây thiết bị (device tree) nhưng không có một ổ đĩa RAM ban đầu (initial RAM disk)1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html).
* [0x48000000: Đây là địa chỉ của cây thiết bị (device tree)1](https://docs.u-boot.org/en/latest/usage/cmd/booti.html).

[Vì vậy, lệnh setenv booti\_cmd 'booti 0x48080000 - 0x48000000' đang thiết lập một biến môi trường để khởi động một kernel Linux tại địa chỉ 0x48080000 với một cây thiết bị tại địa chỉ 0x48000000, nhưng không có ổ đĩa RAM ban đầu](https://docs.u-boot.org/en/latest/usage/cmd/booti.html)

Ổ đĩa ram ban đầu (initial ramdisk) là để chứa thêm các driver/modules cho kernel)

Tftp boot and NFS rootfs

Với NFS rootfs khi ta khai báo bootargs

bootargs=root=/dev/nfs nfsroot=192.168.8.8:/tftpboot/long.trinh-tien/E3/rootfs,nfsvers=4 ip=192.168.8.20

Trong bootargs, nfsvers và ip đều là các tham số quan trọng:

1. [**nfsvers**: Tham số này chỉ định phiên bản của NFS mà bạn muốn sử dụng1](https://stackoverflow.com/questions/57948311/how-to-setup-bootargs-to-load-nfs-in-beaglebone-through-eth0). Trong trường hợp của bạn, nfsvers=4 nghĩa là bạn đang sử dụng NFS version 4. Việc chỉ định phiên bản NFS giúp đảm bảo tương thích giữa máy chủ và máy khách.
2. [**ip**: Tham số này cung cấp thông tin về cấu hình IP cho thiết bị1](https://stackoverflow.com/questions/57948311/how-to-setup-bootargs-to-load-nfs-in-beaglebone-through-eth0). Trong trường hợp của bạn, ip=192.168.8.20 nghĩa là bạn đang cung cấp một địa chỉ IP tĩnh cho thiết bị. Điều này cần thiết để thiết lập kết nối mạng giữa máy chủ NFS và thiết bị.

Lưu ý ip tĩnh nên cùng subnet với nfs server (host)

# E3

setenv bootargs 'root=/dev/nfs nfsroot=192.168.1.21:/tftpboot/PT3/E3/rootfs,nfsvers=4 ip=192.168.1.20'

setenv bootcmd\_tftp 'tftp 0x48080000 PT3/E3/Image;tftp 0x48000000 PT3/E3/${dtb};booti 0x48080000 - 0x48000000'

setenv ipaddr '192.168.1.20' (ip mình tự đặt, cùng subnet là dc)

setenv serverip '192.168.1.21' (ip của máy mình)

setenv dtb 'r8a77990-ebisu-4d.dtb'

setenv bootcmd 'run bootcmd\_tftp'

saveenv

**Cách check trên datasheet:**

A diagram of a computer

Description automatically generated

Vào JUN-2024/RENESAS\_RCH3M3M3NE3D3\_YoctoStartupGuide\_UME\_v5.9.4\_06\_Body.pdf

Tới board cần load để xem memory map, ở đây là : E3 System Evaluation Board Ebisu-4D memory map (Linux)

Ở đây ta thấy dtb sẽ load từ 0x00\_4800\_0000 và kernel image load từ 0x00\_4808\_0000

Từ đó ta vào u-boot để nạp dtb vào kernel image vào 2 địa chỉ trên (thật ra không cần chính xác, chỉ cần phần địa chỉ nó ghi vào không trùng với secure region và các phần khác là ok)

Nó sẽ chạy bootcmd , load xong kernel thì sẽ chạy bootargs

**Build yocto for ULCB, Salvator and boot -up built image on board ( Deadline : June 19)**

Để burn,boot thì ta cần chuẩn bị IPL, secure boot, image, device tree, rootfs

Input:

+ Yocto startup guideline : [RENESAS\_RCH3M3M3NE3D3\_YoctoStartupGuide\_UME\_v5.9.4\_06\_Body.pdf](https://renesasgroup.sharepoint.com/:b:/r/sites/team_RCL-EX/Shared%20Documents/SW%20Maintenance/Document/Linux_BSP/RENESAS_RCH3M3M3NE3D3_YoctoStartupGuide_UME_v5.9.4_06_Body.pdf?csf=1&web=1&e=m7Rqh5)

+ Package : [FeedbackCommit\_Final](https://renesasgroup.sharepoint.com/:f:/r/sites/team_RCL-EX/Shared%20Documents/SW%20Maintenance/Prj_Output/Yocto/Final_Release/PT2/bin_dir/FeedbackCommit_Final?csf=1&web=1&e=zU2GwJ&xsdata=%3D%3D&sdata=dFpvVzBrU201RE5JUDVsdjJpc2NJb2kxcEt3eWZhak9rN0x4MS9oUVFRQT0%3D&ovuser=53d82571-da19-47e4-9cb4-625a166a4a2a%2Clong.trinh-tien%40banvien.com.vn)

**IPL và Secure Boot**

* **IPL (Initial Program Load)**: Là giai đoạn đầu tiên của quá trình khởi động, nơi các thành phần quan trọng như Boot parameter, các boot loader (BL2, BL31) được tải và thực thi để thiết lập môi trường ban đầu cho hệ thống.
* **Secure Boot**: Là một quy trình bảo mật trong đó các thành phần khởi động được xác thực bằng cách sử dụng các chứng nhận (Certification headers) để đảm bảo rằng chỉ các phần mềm đã được ký kết và ủy quyền mới được phép chạy.

**Kết luận**

Danh sách các thành phần phần mềm được ghi vào HyperFlash này bao gồm cả các thành phần của IPL và Secure Boot:

* **IPL**: Thành phần chính của IPL bao gồm **bootparam\_sa0**, **bl2**, và **bl31**. Các thành phần này chịu trách nhiệm cho việc khởi động ban đầu và thiết lập môi trường bảo mật.
* **Secure Boot**: Thành phần **cert\_header\_sa6** là phần không thể thiếu của Secure Boot, đảm bảo rằng các phần mềm khởi động đều được xác thực và bảo mật.

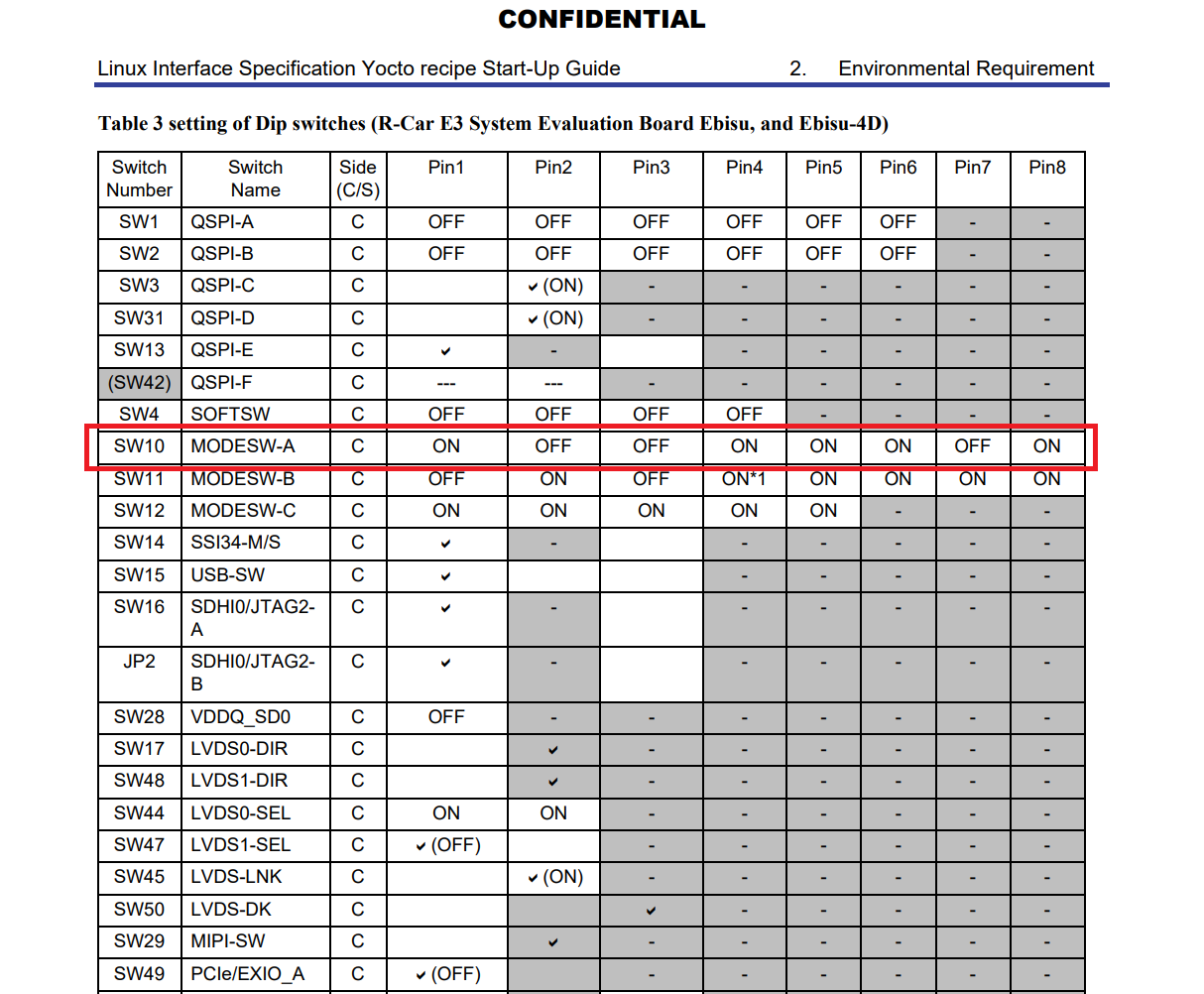
+ Burn IPL document:

**++ Manual burn:** [loadIPL\_manual.docx](https://renesasgroup.sharepoint.com/:w:/r/sites/team_RCL-EX/Shared%20Documents/SW%20Maintenance/Training_Output/BV/2023/Boot%20Guide/loadIPL_manual.docx?d=w16be6b9dec4e48b2be5b00f1e7a28494&csf=1&web=1&e=CXewe0)

1. Dùng tăm chỉnh switch về **QSPI Mode**

4.2.1 For H3/M3/M3N/E3 
a) QSPI Mode 
Switch 
Number 
SWI 
SW2 
SW3 
SWIO 
SW31*1 
Switch 
Name 
QSPI-A 
QSPI-B 
QSPI-C 
MODESW-A 
QSPI-D 
Pini 
ON 
ON 
set(OFF) 
OFF 
Pin2 
ON 
ON 
Pin3 
ON 
ON 
Refer to 2.2 
*I: Only E3 System Evaluation Board needs to set SW31. 
b) Hyper Flash Mode 
Pin4 
ON 
ON 
Pin4 
OFF 
OFF 
Refer to 2.2 
Pin5 
ON 
ON 
ON 
Pin5 
OFF 
OFF 
Pin6 
ON 
ON 
OFF 
Pin6 
OFF 
OFF 
Pin7 
ON 
Pin7 
Pin8 
ON 
Pin8 
Switch 
Number 
SWI 
SW2 
SW3 
SW10 
SW31*1 
Switch 
Name 
QSPI-A 
QSPI-B 
QSPI-C 
MODESW-A 
QSPI-D 
Pinl 
OFF 
OFF 
ON 
Pin2 
OFF 
OFF 
Set(ON) 
Pin3 
OFF 
OFF 

 From yocto guilde ta tìm qua 2.2 /JUN-2024/RENESAS\_RCH3M3M3NE3D3\_YoctoStartupGuide\_UME\_v5.9.4\_06\_Body.pdf



1. Bật nguồn
2. Gõ trên tetra term

--> xls2 --> Bấm Enter --> Chọn 3

1. Chỉnh sw1, sw2 off, sw3 on
2. Nhấn Y trên tetra term
3. Nhập Program Top Address cho từng file

4.1 
Writing data 
Filename 
bootparam_sa0* 
b12-* 
cert_header_sa6* 
b131 
u-boot-elf-* 
Program Top 
Address 
OxE6320000 
OxE6304000 
OxE6320000 
Flash Save 
Address 
oxoooooo 
oxl 80000 
oxl COOOO 
Description 
Loader(Boot parameter) 
Loader 
Loader(Certification) 
ARM Trusted Firmware 
OP-Tee 
U-Boot 

1. Chọn file tương ứng
2. File -> Send file... rồi chọn file

4.1 
Writing data 
Filename 
bootparam_sa0* 
b12-* 
cert_header_sa6* 
b131 
u-boot-elf-* 
Program Top 
Address 
OxE6320000 
OxE6304000 
OxE6320000 
Flash Save 
Address 
oxoooooo 
oxl 80000 
oxl COOOO 
Description 
Loader(Boot parameter) 
Loader 
Loader(Certification) 
ARM Trusted Firmware 
OP-Tee 
U-Boot 

1. Please repeat the xls2 command, if other files are written.
2. Power OFF.
3. Set dip switch to “Hyper Flash Mode”. Note) SW1, SW2, and SW3 are already Hyper Flash mode. Please set SW10.

**++ Auto burn :** [IPL 2.zip](https://renesasgroup.sharepoint.com/:u:/r/sites/team_RCL-EX/Shared%20Documents/SW%20Maintenance/Training_Output/BV/2024/IPL%202.zip?csf=1&web=1&e=rRiC32), [SW-burn\_IPL\_auto.docx](https://renesasgroup.sharepoint.com/:w:/r/sites/team_RCL-EX/Shared%20Documents/SW%20Maintenance/Training_Output/BV/2024/SW-burn_IPL_auto.docx?d=wc3194359c6744c3ba6c6d35384883421&csf=1&web=1&e=B419HL) ( You should try this first )

A diagram of a computer

Description automatically generated

Như ta thấy các địa chỉ Hyper Flash được liệt kê trong **E3 System Evaluation Board Ebisu-4D memory map (Boot)** , từ đó ta load lên SDRAM

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

+ Linux-bsp for cross compile: [GitHub - renesas-rcar/linux-bsp at v5.10.194/rcar-5.3.1](https://github.com/renesas-rcar/linux-bsp/tree/v5.10.194/rcar-5.3.1)

# install poky

Link tải toolchain: [Index of /releases/yocto/yocto-3.1.11/toolchain/x86\_64/ (yoctoproject.org)](https://downloads.yoctoproject.org/releases/yocto/yocto-3.1.11/toolchain/x86_64/)  
Tải về xong bash \*.sh (nhớ chmod 777 cho file)

# clone source  
$ git clone <https://github.com/renesas-rcar/linux-bsp.git>  
$ cd linux-bsp  
$ git checkout **v5.10.194/rcar-5.3.1**  
  
#build  
$ source /opt/poky/3.1.11/environment-setup-aarch64-poky-linux

$ make clean

$ make distclean

$ unset LD\_LIBRARY\_PATH

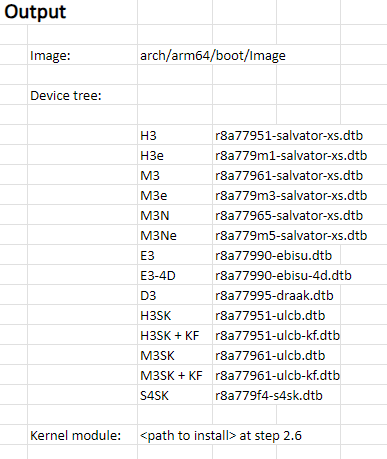
$ unset PKG\_CONFIG\_PATH

$ touch .scmversion

$ make defconfig

$ make -j16

Đây là device tree cho từng loại (source: practice\_linux.xlsx)



+ Poky SDK : [team\_RVSSW-EXT - SDK - All Documents (sharepoint.com)](https://renesasgroup.sharepoint.com/sites/team_RCL-EX/Shared%20Documents/Forms/AllItems.aspx?csf=1&web=1&e=Xc2dXR&ovuser=53d82571%2Dda19%2D47e4%2D9cb4%2D625a166a4a2a%2Cnguyen%2Enguyen%2Eyj%40renesas%2Ecom&CID=8b7d33a1%2D80a3%2D3000%2D75f7%2D4c0a7502aee3&cidOR=SPO&FolderCTID=0x012000F08F79BEB761744CA3AB82463B16F66B&id=%2Fsites%2Fteam%5FRCL%2DEX%2FShared%20Documents%2FSW%20Maintenance%2FPrj%5FOutput%2FYocto%2FFinal%5FRelease%2FPT2%2FSDK&viewid=d490f8fb%2D536f%2D4dee%2Dbe14%2D12854e843942)

**Set up toolchain on Linux**

* Download toolchain from [team\_RVSSW-EXT - SDK - All Documents (sharepoint.com)](https://renesasgroup.sharepoint.com/sites/team_RCL-EX/Shared%20Documents/Forms/AllItems.aspx?csf=1&web=1&e=Xc2dXR&ovuser=53d82571%2Dda19%2D47e4%2D9cb4%2D625a166a4a2a%2Cnguyen%2Enguyen%2Eyj%40renesas%2Ecom&CID=8b7d33a1%2D80a3%2D3000%2D75f7%2D4c0a7502aee3&cidOR=SPO&FolderCTID=0x012000F08F79BEB761744CA3AB82463B16F66B&id=%2Fsites%2Fteam%5FRCL%2DEX%2FShared%20Documents%2FSW%20Maintenance%2FPrj%5FOutput%2FYocto%2FFinal%5FRelease%2FPT2%2FSDK&viewid=d490f8fb%2D536f%2D4dee%2Dbe14%2D12854e843942)
* Unzip this folder

**$ unzip Toolchain.zip**

**$ cd Toolchain**

* Merge all splited files, then unzip the merged file

**$ cat poky-glibc-x86\_64-core-image-weston-sdk-aarch64-salvator-x-toolchain-3.1.11.7z.00\* > file.7z**

**$ 7za x file.7z**

* Execute the script file

**$ chmod 777 poky-glibc-x86\_64-core-image-weston-sdk-aarch64-salvator-x-toolchain-3.1.11.sh**

**$ ./poky-glibc-x86\_64-core-image-weston-sdk-aarch64-salvator-x-toolchain-3.1.11.sh**

Output:

+ Can burn IPL for H3SK/E3

+ Can build yocto for H3SK/E3 (bv\_gen3\_maintenance build for H3SK, hvn\_gen3\_maintenance build for E3 )

+ Can boot -up built image on board

+ Can cross compile kernel for H3SK/E3

**Set ip tĩnh cho máy host:**

sudo ifconfig enx00e04c126819 192.168.8.8 up

sudo ifconfig enp0s31f6 down