**Porting of U-boot on Raspberry-pi3**

**Table of Contents**

Introduction.................................................................................................. 3

Booting Process of RPi3.............................................................................. 4

1. Steps To Partitioning SD card..................................................................... 5
2. Steps To building U-boot image for Rpi3................................................... 6
3. Steps to copying u-boot into SD card.......................................................... 7
4. Interfacing with Minicom............................................................................ 8
5. References................................................................................................... 9

**Introduction**

*The* U-Boot *utility is a multi-platform, open-source, universal boot-loader with comprehensive support for loading and managing boot images, such as the Linux kernel.*

*This article is a quick start up guide on porting U-Boot for Raspberry-PI board using SD card.*

**Booting process of RPi3**

**Stage 1**

To reduce cost, the Raspberry Pi (Model A & B) omits any on-board non-volatile memory used to store the boot loaders, Linux Kernels and file systems as seen in more traditional embedded systems. Rather, a SD/MMC card slot is provided for this purpose. (The Raspberry PI Compute Module has 4Gbyte eMMC Flash on-board).

stage 1 boot is in the **on-chip ROM***.* Loads Stage 2 in the L2 cache. The Raspberry Pi’s Broadcom BCM2835 system on a chip (SoC) powers up with its **ARM11**76JZF-S 700 MHz processor held in reset. The VideoCore IV GPU core is responsible for booting the system. It loads the first stage boot oader from a ROM embedded within the SoC. The first stage bootloader is designed to load the second stage bootloader (bootcode.bin) from a FAT32 or FAT16 file system on the SD Card.

**Stage 2**

The second stage bootloader ***- bootcode.bin*** - is executed on the VideoCore GPU and loads the third stage bootloader ***- start.elf.*** (Historically, yet another bootloader called loader.bin was loaded at this stage, but has since been phased out)

**Stage 3**

The third stage bootloader - start.elf - is where all the action happens. It starts by reading ***config.txt***, a text file containing configuration parameters for both the VideoCore (Video/HDMI modes, memory, console frame buffers etc) and loading of the Linux Kernel (load addresses, device tree, uart/console baud rates etc).

**Stage 4**

We will use **config.txt** to tell **start.elf** to load the **kernel.img** image

**kernel.img**, is the **first**! thing that runs on the ARM processor.We will use u-boot as **kernel.img** so our **config.txt** will look as:

**kernel = u-boot.bin**

After completing above steps Booting is completed and RPI board is ready to take a commands from user.

1. **Steps To Partitioning SD card**

**STEP 1:**  Insert a SD card in to the card reader and connect it to the PC ,after detecting the SD card, from the following command we can see SD card partition.

$sudo fdisk -l

**Note**:- it will show something like below location.

/dev/sdc

**STEP 2:**  For creating partitions execute the below command

sudo fdisk (location of SDcard)

**STEP 3:**  Execute the below character to list out all the partition's

p

**STEP 4:**  Execute the below character to delete the partition

d

**STEP 5:**  Execute the below character sequentially to create new partitions

i) To create a new partition

n

ii) Partition it would be in primary

p

iii) No. Of Partitions

1

iv) For default starting value of primary partition

[enter]

v) For default size of the primary partition

[enter]

**STEP 6:**  Execute the below character to activate / boot the primary partition

a

i) To activate the which partition

1

**STEP 7:**  Execute the below character to changing the partition type to FAT32

t

i) Select the partition number

1

ii) For creating FAT32 file system

c

**STEP 8:**  Execute the below character to write all information in to the new partition

w

1. **Steps To building U-boot image for Rpi-3**

**Prerequisite:**

Cross Compiler : arm-linux-gnueabi-

GCC : GCC Version should be greter than gcc 6.0

Ubuntu : Ubuntu 14.04 or more

**STEP 1:** Install cross compiler and export environment variables:

Run below command to get a ARM based linaro cross compiler.

$ sudo apt-get install gcc-arm-linux-gnueabi

**STEP 2:**  Download the U-Boot source from the below link

Get the source code by cloning the U-Boot git repository :

$ git clone --depth 1 --branch v2017.11 git://git.denx.de/u-boot.git v2017.11

or download the tar file :

$ wget <ftp://ftp.denx.de/pub/u-boot/u-boot-2017.11.tar.bz2>

**STEP 3:** Compile U-Boot

Go to the folder using below command.

**$** cd V2018.11

$ sudo make -C v2017.11/ CROSS\_COMPILE=aarch64-linux-gnu- rpi\_3\_defconfig

$ sudo make -C v2017.11/ CROSS\_COMPILE=aarch64-linux-gnu-

After downloading the U-Boot source. it will create a folder, name as **V2018.11**. After executing all above steps do ' **ls**' command, you can see below images in your folder

u-boot.bin,u-boot.lds,u-boot.map,u-boot.srec

|  |  |
| --- | --- |
| **Filename** | **Description** |
| System.map | The symbol map |
| u-boot | U-Boot in ELF binary format |
| u-boot.bin | U-Boot raw binary image that can be written to the boot storage device |
| u-boot.srec | U-Boot image in Motorola's S-Record format |

1. **Steps to copying u-boot into SD card**

**STEP 1:** Use below link to download **bootcode.bin** and **start.elf** according to rpi supported images and copy in to your SDcard.

<https://github.com/raspberrypi/firmware/tree/master/boot>

**STEP 2:** Insert the SD card into card reader and connect the USB of card reader to CPU.

**STEP 3:** You will see a window on monitor, after connecting USB to the CPU

**STEP 4:** Copy the below images into SD card location using below commands from command line.

u-boot.bin , bootcode.bin, start.elf, and config.txt

**STEP 5:** Give following commands.

$ mount

By putting above command we will get sd card location on which bootable partition is mounted.

$ cp u-boot.bin (location of SDcard)

eg: In this case location of SD card is- media/abc/FCF1-DD00/

then follow below commands replacing with your sdcard location.

# cp bootcode.bin media/abc/FCF1-DD00/

# cp start.elf /media/abc/FCF1-DD00/

# vim /media/abc/FCF1-DD00/config.txt

write config.txt as below:

# Serial console output!

enable\_uart=1

# 64bit-mode

arm\_control=0x200

# Use U-Boot

kernel=u-boot.bin

dtparam=i2c\_arm=on

dtparam=spi=on

1. **Interfacing with minicom**

**Connections of RPI:-**

1. Insert the SD card containing card reader into RPI board memory card slot.
2. Connect the RPI Tx and Rx pins to RS-232 serial converter.
3. Connect the RS-232 USB to CPU.

|  |  |  |
| --- | --- | --- |
| **RPI GPIO pin number** | **USB-to-TTL pins** | **Description** |
| 8 | Tx | Rx |
| 10 | Rx | Tx |
| 6 | GND | GND |

**Steps to use minicom to check the output:-**

**STEP 1:** To open the minicom first install minicom using below command

$ sudo apt-get install minicom

**STEP 2:** put command dmesg So that you will get exact address of connecting device(USB-to-TTL)

$ dmesg

**STEP 3:** Use below command to open the minicom to see the output.

$ sudo minicom -s

**STEP 4:** you will get a configuration menu after opening the minicom.

**STEP 5:** set **Serial port setup** as below

|  |  |
| --- | --- |
| Serial Device | /dev/ttyUSB0 |
| Bps/Par/Bits | 115200 8N1 |
| Hardware Flow Control | No |
| Software Flow Control | No |

**STEP 6:** Give supply to Rpi3 board, **U-boot>** prompt should be come on minicom screen.

1. **References**

[1] <https://github.com/raspberrypi/firmware/tree/master/boot>

[2]<http://www.qnx.com/developers/docs/660/index.jsp?topic=%2Fcom.qnx.doc.bsp_update.guide%2Ftopic%2Fbsp_660_sd_card_ubuntu.html>

[3] <http://elinux.org/RPi_U-Boot>