mobilint

REGULUS Kernel Build Guide

Document Number: MRS003-EN
Version 2.0
Release Date: Jul. 21, 2026

Confidential. Please do not distribute.

Disclaimer and Notice

Copyright

Copyright © 2019-2026, Mobilint, Inc. All rights reserved.

WARNING No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Mobilint, Inc. (”MOBILINT”). This publication is intended for use by designated recipients only. This publication contains confidential information (including trade secrets) of MOBILINT protected by Competition Law, Trade Secrets Protection Act and other related laws, and therefore may not be, in part or in whole, directly or indirectly publicized, distributed, photocopied or used (including in a posting on the Internet where unspecified access is possible) by any unauthorized third party. MOBILINT reserves its right to take any and all measures both in equity and law available to it and claim full damages against any party that misappropriates MOBILINT’s trade secrets and/or confidential information.

Important Notice

MOBILINT, Inc. reserves the right to make changes to the information in this publication at any time without prior notice. All information provided is for reference purposes only. MOBILINT assumes no responsibility for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

This publication on its own does not convey any license, either express or implied, relating to any MOBILINT and/or third-party products, under the intellectual property rights of MOBILINT and/or any third parties.

MOBILINT makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does MOBILINT assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Customers are responsible for their own products and applications. ”Typical” parameters can and do vary in different applications. All operating parameters, including ”Typicals” must be validated for each customer application by the customer’s technical experts.

MOBILINT products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the MOBILINT product could reasonably be expected to create a situation where personal injury or death may occur. Customers acknowledge and agree that they are solely responsible to meet all other legal and regulatory requirements regarding their applications using MOBILINT products notwithstanding any information provided in this publication. Customer shall indemnify and hold MOBILINT and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim (including but not limited to personal injury or death) that may be associated with such unintended, unauthorized and/or illegal use.

1 Introduction 3

2 Linux Kernel Source Code 4

3 Kernel Build Guide 7

3.1 Apply Toolchain for Cross Compilation . . . 7
3.2 Modifying Linux Kernel Source Code . . . 7
3.3 Changing Default Configuration File . . . 7
3.4 Linux Kernel Menuconfiguration . . . 8
3.5 Build . . . 8
3.6 Build Results . . . 10

4 Kernel Image Update 12

4.1 SD-Card Booting mode . . . 12
4.2 eMMC Booting Mode . . . 12
4.2.1 GUI Mode . . . 12
4.2.2 CLI Mode . . . 14

5 Installing Module File 16

5.1 Installation using SD-Card . . . 16

6 Build Root File System 17

6.1 File System Configuration . . . 17
6.2 Creating Root File System . . . 18
6.3 Root File System Update . . . 18

7 U-boot Build 19

7.1 U-boot image for SD-Card . . . 19
7.2 U-boot Image for eMMC . . . 19

Index 21

Document Revision History

Revision Number	Date	Description
2.0	2026-06-25	Update for regulus-evk-rev01 board
1.4	2026-01-13	Update PDF Design Format
1.3	2025-07-28	change eMMC partition:7->8
1.2	2025-06-17	change the file name for toolchain
1.1	2025-06-11	Editorial update
1.0	2025-03-28	Initial release

List of Terms

All referenced brands, product names, service names and trademarks are property of their respective owners.

Term	Description
x86	a trademark of Intel
ARM	a trademark of Arm Limited
Ubuntu	a trademark of Canonical

1 Introduction

This document explains how to build a boot image using the Linux kernel source code and U-Boot source code, and how to write the generated image to an SD-Card card or eMMC. The Linux running on REGULUS is based on the `5.15.125` version of the Linux Project.

Table 1.1 Reference

No.	Document List
1	regulus_application_note.pdf
2	regulus_boot_sequence_guide.pdf
3	regulus_eMMC_burning_guide.pdf

2 Linux Kernel Source Code

Linux Kernel Source shows the the main directory and files included in the Linux kernel source code. The file that defines kernel configuration is located at **'arch/arm64/configs/regulus_evk_defconfig'**, and the device tree files, which describe the hardware configuration of the REGULUS EVK board, are located in the **'arch/arm64/boot/dts/mobilint'** directory.

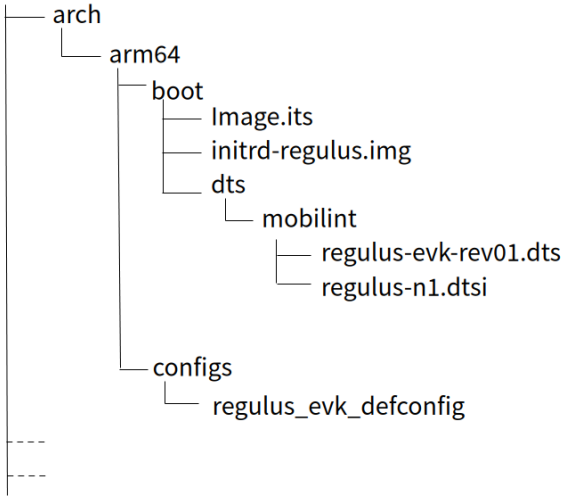

Figure 2.1 Linux Kernel Source

In Linux Kernel Source, ”Image.its” and ”initrd-regulus.img” are files for creating a boot image for eMMC, not for SD-Card. ”Image.its” is the description file for creating the boot image ”Image.fit”, and ”initrd-regulus.img” is the initramfs image file included int the ”Image.fit” file. ”dts” is the directory where the device tree file is stored. ”mobilint.dtsi” contains information about the device tree for the entire configuration of REGULUS. **”regulus-evk-rev01.dts”** contains information about the device tree for the ”evk-rev01” board configuration. In directory ”dts/mobilint”, the device tree overlay files for I/O devices, for example camera sensor and display device, are placed. Device Overlay displays several device tree overlay files and their description. Only one device belonging to the same group in Device Overlay should be selected.

The following are the descriptions of each file listed in Linux Kernel Source.

- ”Image.its” and ”initrd-regulus.img” : Files used to create a boot image for eMMC (not for SD card booting).
- ”Image.its” : description file for creating the boot image ”Image.fit”
- ”initrd-regulus.img” : the initramfs image file included in the ”Image.fit” file.
- ”dts” : Directory that contains device tree source files.
- ”mobilint.dtsi” : Defines the shared or common device tree information used across multiple board configurations for REGULUS.
- ”regulus-evk-rev01.dts” Specifies the device tree configuration for ”evk-rev01” board configuration.

In the ”dts/mobilint” directory, device tree overlay files are stored. These overlays define additional hardware components such as camera sensors or display devices. Device Overlay lists various device tree overlay files and their corresponding hardware descriptions.

Note

Only one overlay file from the same device group in Device Overlay should be selected at a time.

Table 2.1 Device Overlay

Group	File Name	Description
DSI	bridge-overlay-lontium-lt9611.dtsi	Display into HDMI monitor
DSI	panel-overlay-tdo-tl101wxv41.dtsi	Display into 10" LCD
Camera Sesnor	sensor-overlay-imx678-rx0.dtsi	Camera sensor for imx678
Camera Sensor	sensor-overlay-imx567-rx0.dtsi	Camera sensor for imx567

Table 2.2 Default Configuration

Default Configuration
CONFIG_CMDLINE="console=ttyS0,115200n8 root=/dev/ram0 rootfstype=ext4 rw rootwait ignore_loglevel"
CONFIG_BLK_DEV_INITRD=y
CONFIG_BLK_DEV_RAM=y
CONFIG_MEDIA_SUPPORT=y
# Camera sensors
CONFIG_VIDEO_IMX678=y
# REGULUS media drivers
CONFIG_VIDEO_REGULUS_MIPI_CSI2_RX=y
CONFIG_VIDEO_REGULUS_MIPI_CSI2_RX_TEST_PATTERN=0
CONFIG_VIDEO_REGULUS_MIPI_CSI2_TX=y
CONFIG_VIDEO_REGULUS_ISP=y
#CONFIG_VIDEO_REGULUS_ISP_DMA=y
CONFIG_VIDEO_REGULUS_ISP_DVP=y
# CONFIG_VIDEO_REGULUS_ISP_WITH_TUNNING is not set
CONFIG_VIDEO_REGULUS_MAPI=y
CONFIG_VIDEO_REGULUS_CODAJ=y
CONFIG_VIDEO_WAVE_VPU=m
# REGULUS Framebuffer
CONFIG_FB_REGULUS=y
CONFIG_FB_REGULUS_LCD_MIPI=y
CONFIG_CMDLINE_FORCE=y
CONFIG_STRICT_DEVMEM=n
CONFIG_MEDIA_USB_SUPPORT=y
CONFIG_USB_VIDEO_CLASS=y
CONFIG_USB_VIDEO_CLASS_INPUT_EVDEV=y

Default Configuration shows the main contents of the configuration for kernel included in the Linux Kernel Source Code. Default Configuration is for the booting image using eMMC.

In "CONFIG_CMDLINE", "root" is defined as "/dev/ram0", which means that the initramfs image is first used in the booting sequence. Initramfs proceeds with a firmware update when the specific conditions are met. If a partition with the PARTLABEL "rootfs", it performs a switch_root to that partition.

Default Configuration lists the main kernel configuration settings included in the Linux kernel source code, specifically for creating a boot image using eMMC, not SD-Card. In "CONFIG_CMDLINE", "root" is defined as "/dev/ram0", which means that the initramfs image is first used in the booting sequence. Initramfs proceeds with a firmware update when the specific conditions are met. If a partition with the PARTLABEL "rootfs" exists, it performs a switch_root to that partition.

For SD-Card boot configurations, refer to Changing Default Configuration File.

The IMX678 is defined as the default camera sensor, and both MIPI_CSI2_RX and MIPI_CSI2_TX are enabled for camera interface support. Additional image signal processing features are also configured, including the following items:

```
REGULUS_ISP
REGULUS_MAPI
REGULUS_CODAJ
REGULUS_WAVE_VPU
```

REGULUS_WAVE_VPU must be defined in module "m" style.

3 Kernel Build Guide

3.1 Apply Toolchain for Cross Compilation

The Cross Compile environment described in this document is an example for a PC using an x86 CPU and Linux Ubuntu 22.04. Among the files in the distributed **`regulus-release_v4.x.x.tar.gz`**, run **`install-regulus-toolchain.sh`** to install the toolchain. The command to use the installed toolchain is as follows, where v4.X.X indicates the version of the distributed toolchain.

$ source /opt/crosstools/mobilint/2.0.0/v4.X.X/environment-setup-cortexa53-mobilint-linux

Once this environment setup script is applied, environment variables such as ARCH, SDKTARGETSYSROOT, and GCC will be automatically configured, allowing you to use the toolchain.

3.2 Modifying Linux Kernel Source Code

After modification is complete, to change the Linux Kernel Version, modify it as follows. The part that displays the version in **"Makefile"** is as follows.

```
VERSION = 5
PATCHLEVEL = 15
SUBLEVEL = 125
EXTRAVERSION = -1
```

The VERSION, PATCHLEVEL, and SUBLEVEL values are fixed values, so do not modify them. The Linux Kernel version can be distinguished by changing the **EXTRAVERSION** value. For example, if "EXTRAVERSION=-2" is set, the "uname -r" command will be displayed as shown below in the modified Linux Kernel environment.

```
5.15.125-2
```

3.3 Changing Default Configuration File

The default configuration file for the Linux Kernel is stored in the location below.

```
Directory : arch/arm64/configs
File : regulus_evk_defconfig
```

The boot device should be specified in the default configuration file. The devices that can be used as booting devices are as follows:

1. SD-Card
2. eMMC
3. NFS

Define the booting device in **CONFIG_CMDLINE** of Default Configuration. The **CONFIG_CMDLINE** settings for each booting device are the same as CMDLINE Settings.

Refer to CMDLINE Settings for the CONFIG_CMDLINE settings for each booting device.

Table 3.1 CMDLINE Settings

CMDLINE Settings	
SD-Card	CONFIG_CMDLINE="console=ttyS0,115200n8 root=/dev/mmcblk0p2 rootfstype=ext4 rw rootwait ignore_loglevel"
eMMC	CONFIG_CMDLINE="console=ttyS0,115200n8 root=/dev/ram0 rootfstype=ext4 rw rootwait ignore_loglevel"
NFS	CONFIG_CMDLINE="console=ttyS0,115200n8 root=/dev/nfs nfsroot=192.↩ 168.10.174:/srv/regulus_nfsroot,tcp,vers=3 rw ip=::::root:eth0:dhcp rootwait init=/sbin/init ignore_loglevel"

3.4 Linux Kernel Menuconfiguration

To change boot options in menu configuration or select the desired configuration, use the menu configuration. When the Linux Kernel configuration is changed to the required settings, execute the command below.

```
$ make ARCH=arm64 menuconfig
```

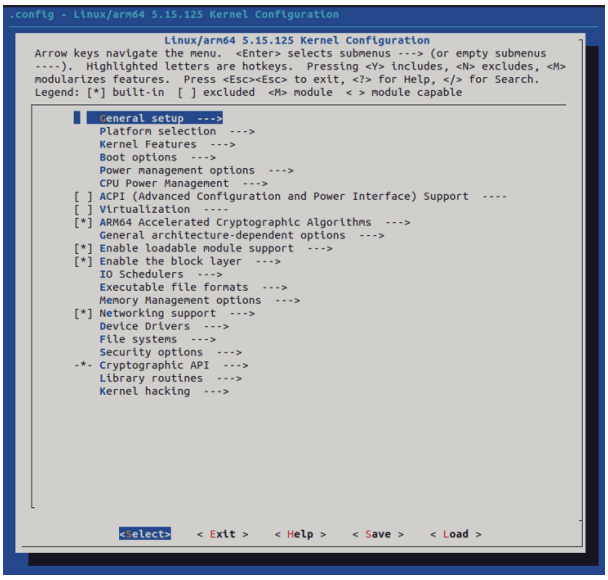

Figure 3.1 Kernel Menuconfig

3.5 Build

The command to generate the **".config"** file based on **regulus_evk_defconfig** is as follows:

```
$ make regulus_evk_defconfig

 HOSTCC scripts/basic/fixdep
 HOSTCC scripts/kconfig/conf.o
 HOSTCC scripts/kconfig/confdata.o
 HOSTCC scripts/kconfig/expr.o
 LEX scripts/kconfig/lexer.lex.c
```

```
 YACC scripts/kconfig/parser.tab.[ch]
 HOSTCC scripts/kconfig/lexer.lex.o
 HOSTCC scripts/kconfig/menu.o
 HOSTCC scripts/kconfig/parser.tab.o
 HOSTCC scripts/kconfig/preprocess.o
 HOSTCC scripts/kconfig/symbol.o
 HOSTCC scripts/kconfig/util.o
 HOSTLD scripts/kconfig/conf
#

# configuration written to .config

#
```

After creating the .config file as shown above, you can proceed with compilation using the following command. To use the commands needed to create a boot image on the host PC, install "u-boot-tools" and "device-tree-compiler" using the "apt" command and then run "make".

```
$ sudo apt install u-boot-tools device-tree-compiler
 (type sudo password)
$ make -j<proc>

 WRAP arch/arm64/include/generated/uapi/asm/kvm_para.h
 WRAP arch/arm64/include/generated/uapi/asm/errno.h
 WRAP arch/arm64/include/generated/uapi/asm/ioctl.h
 WRAP arch/arm64/include/generated/uapi/asm/ioctls.h
 WRAP arch/arm64/include/generated/uapi/asm/ipcbuf.h
 WRAP arch/arm64/include/generated/uapi/asm/msgbuf.h
...
 LD [M] lib/crypto/libdes.ko
 LD [M] lib/crypto/libsha256.ko
 LD [M] lib/crypto/libsm4.ko
 LD [M] lib/libcrc32c.ko
 LD [M] lib/lzo/lzo_compress.ko
 LD [M] lib/raid6/raid6_pq.ko
 LD [M] net/dsa/dsa_core.ko
 LD [M] lib/zstd/zstd_compress.ko
 LD [M] net/ipv4/netfilter/nf_defrag_ipv4.ko
 LD [M] net/netfilter/nf_conntrack.ko
 LD [M] sound/soc/codecs/snd-soc-hdmi-codec.ko
```

When the Linux Kernel is normally compiled, the driver code loaded in the module format must be copied to **/lib/modules/** of the root file System To do this, module-related files must be prepared. The procedure for installing module-related files on the REGULUS EVK board is explained in Installing Module File.

When the Linux kernel is successfully compiled, driver code built as loadable modules must be copied to the **/lib/modules/** directory of the root file system. To do this, the necessary module-related files must first be prepared.

The procedure for installing these module files on the REGULUS EVK board is described in Installing Module File.

The command to create module-related files in the "modules" directory is as follows.

```
$ mkdir -p ~/modules
$ make modules_install INSTALL_MOD_PATH=~/modules
```

When the command above is executed, the **~/modules** directory is created and module-related files are stored into it.

3.6 Build Results

The results generated after completing the build process are as follows. Build Images shows where the built files are stored. In the following example, the base board is "regulus-evk-rev01" board.

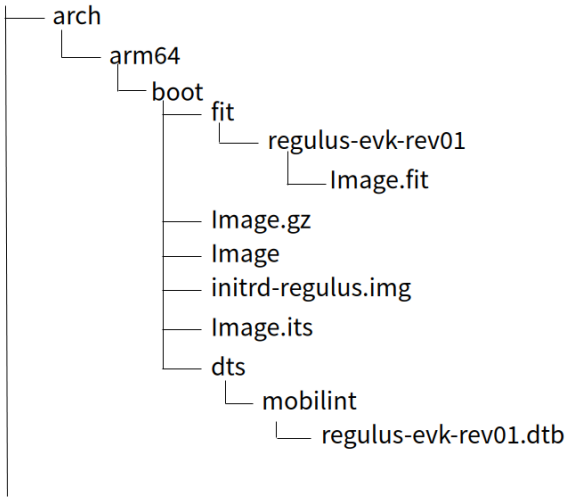

Figure 3.2 Build Images

Boot Images shows the booting images and their contents.

Table 3.2 Boot Images

Name	Contents
Image	Linux kernel image
Image.gz	compressed Linux kernel image
Image.its	description file for creating the boot image "Image.fit" in the "fit" directory
Initrd-regulus.img	initramfs image file included in the "Image.fit" file
fit	flattened image tree directory
regulus-evk-rev01	directory for "regulus-evk-rev01" board
Image.fit	image file emerged image header, Image, device tree and Initrd-regulus.img
dts	device tree directory
mobilint	directory for regulus
regulus-evk-rev01.dtb	device tree blob file for "regulus-evk-rev01" board

In REGULUS, booting images are stored on SD-Card or eMMC. SD-Card stores the boot image using the MBR (Master Root Record) partition method. eMMC stores the boot image using the GPT(GUID Partition Table) partition method. Performing a kernel build will create boot images for both the SD-Card and eMMC. In Boot Images, select and use only the necessary files according to the boot method.

Boot Images for SD-Card are booting images stored on the SD card.

Table 3.3 Boot Images for SD-Card

Name	Contents
Image	Linux kernel image
dts	device tree directory
mobilint	directory for regulus
regulus-evk-rev01.dtb	device tree blob file for ”regulus-evk-rev01” board

Boot Images for eMMC refers to the booting image used for eMMC burning. ”Image.fit” is the image used during eMMC booting and is stored in the 5th partition of the eMMC.

Table 3.4 Boot Images for eMMC

Name	Contents
Image.its	description file for creating the boot image ”Image.fit” in the ”fit” directory
Initrd-regulus.img	initramfs image file included in the ”Image.fit” file
fit	flattened image tree directory
regulus-evk-rev01	directory for ”regulus-evk-rev01” board
Image.fit	image file emerged image header, Image, device tree and Initrd-regulus.img

4 Kernel Image Update

4.1 SD-Card Booting mode

To use the booting mode via SD card, you can simply copy the following files generated from the kernel build to the first partition of the SD card.

1. Image
2. regulus-evk-rev01.dtb

4.2 eMMC Booting Mode

The file used for the kernel image update is as follows:

```
arch/arm64/boot/fit/regulus-evk-rev01/Image.fit
```

Kernel image updates are performed using the firmware update tool comamnd (”firmwareupdatetool”), which is built into the distributed rootfs. This section assumes that eMMC full burning has been completed according to the document specified in Reference Document.

Table 4.1 Reference Document

No.	Document List
1	regulus_application_note.pdf
2	regulus_boot_sequence_guide.pdf
3	regulus_eMMC_burning_guide.pdf

4.2.1 GUI Mode

Run the following commands to create the required directory and navigate into it:

```
$ mkdir -p /root/.firmware_update
$ cd /root/.firmware_update
```

Prepare the Image.fit file depending on the availability of Ethernet:

- Ethernet available

```
$ scp user-id@host-pc-ip-address:~/Image.fit .
```

- Ethernet not available

```
# Insert micro SD card containing Image.fit into the REGULUS board

$ mkdir -p /mnt/sdcard
$ mount /dev/mmcblk0p1 /mnt/sdcard
$ cp /mnt/sdcard/Image.fit .
$ sync
$ umount /mnt/sdcard
```

Run the Firmware Update Tool with **`firmwareupdatetool`**, and sequentially select "Firmware Update (from Local)" and "Kernel Image".

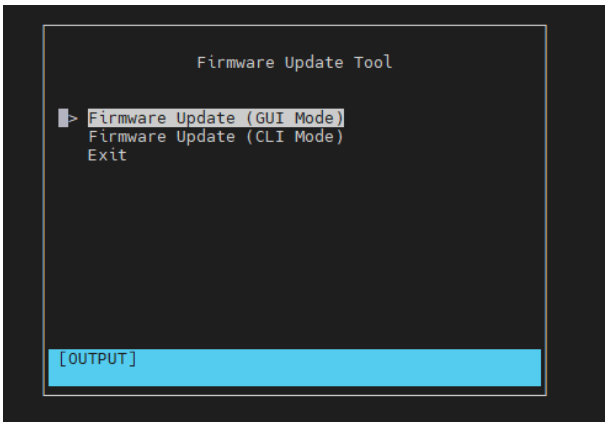

Figure 4.1 Main Menu

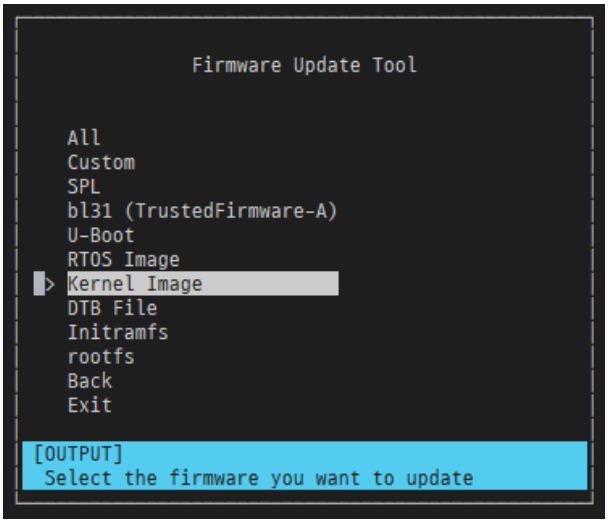

Figure 4.2 Update Menu

In the Update Menu, "All" is a menu to update all images, and "Custom" is a menu to update multiple selected images.

When you select "Custom", the items to be updated are listed. Use the arrow keys to move up and down, and press the "SPACE" key on the item you want to update to select it, which will display an "X" mark. After selecting all the items you want to update, press the "s" key to update all the selected items. When selecting the "Custom" menu, all the images to be updated must exist in the **"~/.firmware_update"** directory.

Once the update is complete, the following screen will appear. Press any key to reboot.

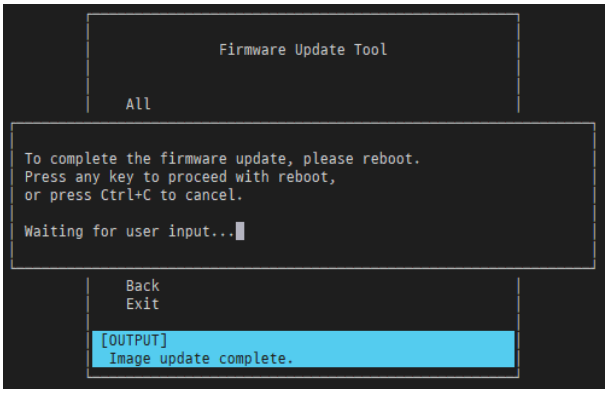

Figure 4.3 Reboot to Complete

After rebooting, if the following screen is displayed, the kernel image update was successful.

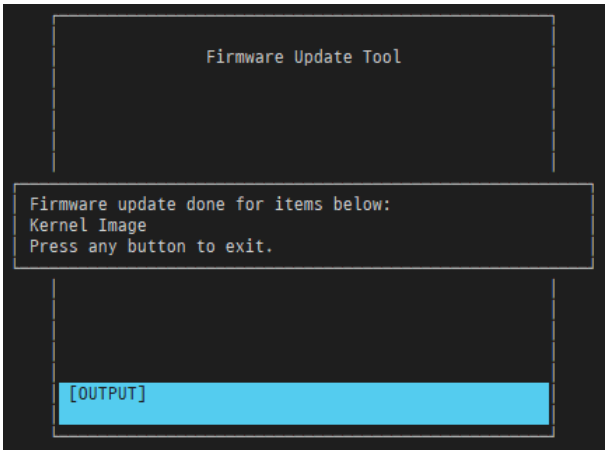

Figure 4.4 Update Done

4.2.2 CLI Mode

Follow the same steps as in chapter GUI Mode to prepare the Image.fit file so that it exists on the REGULUS board.

Note

The Image.fit file can be placed in any directory, but this guide assumes it is located at `/root/.firmware_update` for consistency.

Run the Firmware Update Tool in CLI mode using the following command:

```
$ firmwareupdatetool -c
or
$ firmwareupdatetool --cli
```

Then, execute the update and reboot using the command below to complete the process:

```
Firmware Update CLI Mode
Type 'help' to see available commands.
```

```
cli> update /root/.firmware_update/Image.fit
Updating Image...
Note: Image update is complete.
Reboot is required to complete the update.
cli> quit

$ reboot
```

5 Installing Module File

This section describes the process of installing the module files created during Kernel Compile (mentioned in Build) onto the REGULUS EVK board. On the Host PC, it is necessary to compress the module files into a single file using the **"tar"** command. This chapter explains how to install the module files on the REGULUS EVK board using SD-Card and Ethernet.

On the Host PC, verify that the files exist in the "~/modules" directory as follows. The file names should be the same as the version name applied when modifying the Linux source code. This document describes the process assuming the version name is "5.15.125-2".

```
lib/modules/5.15.125-2
```

Create a compressed file by executing the following command.

```
$ cd ~/modules/lib/modules
$ tar czvf ~/modules.tar.gz 5.15.125-2
 ==> ~/modules.tar is created
```

5.1 Installation using SD-Card

Copy the module file to the REGULUS EVK board using the SD-Card and then install it. In Host PC, copy the module file to the SD-Card. The SD-Card must be formatted as fat32. Below, "???" is the Volume Name corresponding to the SD-Card.

```
$ cp ~/modules.tar.gz /media/{whoami}/???/.
```

Install the SD-Card containing the changed module file into REGULUS EVK board.

```
In REGULUS EVK board
$ mkdir -p /mnt/sdcard
$ mount /dev/mmcblk0p1 /mnt/sdcard
$ cd /root
$ cp /mnt/sdcard/modules.tar.gz .
$ cd /usr/lib/modules
$ tar -xzvf /root/modules.tar.gz
$ ls -l /usr/lib/modules
==> If a directory starting with 5.15.125-2 exists, it was installed normally.
```

To apply the installed module, execute the "poweroff" command and restart the REGULUS EVK board. ## Installation using Ethernet Using "scp" command, copy the module file from the Host PC and then perform the installation.

```
In REGULUS EVK board,
$ cd /root
$ scp user-id@hostpc-ip_address:~/modules.tar.gz .
 ==> type passwd for user-id in host pc
$ ls -l modules.tar.gz <== for confirmation
$ cd /usr/lib/modules
$ tar -xzvf /root/modules.tar.gz
$ ls -l /usr/lib/modules
 ==> If a directory starting with 5.15.125-2 exists, it was installed normally.
```

To apply the installed module, execute the "poweroff" command and restart the REGULUS EVK board.

6 Build Root File System

6.1 File System Configuration

REGULUS uses SD-Card or eMMC as auxiliary storage. SD-Card uses MBR partition and eMMC uses GPT partition respectively. SD-Card is divided into two partitions, the first partition is in fat32 format and the second partition is in ext4 format. The first partition stores the image files required for REGULUS to boot, and the second partition stores the root file system. SD-Card Partition shows the Partitions and related information for the SD-Card device.

	Partition No.	Capacity	File System	Files	Device Name
SD-Card 32 GB	1st Partition	400 MB	fat32	SPL.bin bl31.bin u-boot.bin Image regulus-base-b.dtb	/dev/mmcblk1p1
	2nd Partition	~31.6 GB	ext4	Root File System	/dev/mmcblk1p2

Figure 6.1 SD-Card Partition

eMMC, GPT partition, is divided into 8 partitions. eMMC Partition shows the Partitions and related information for the eMMC device.

	Partition No.	Capacity	File System	Files	Device Name
eMMC 32GB	1st Partition	1 MB	-	SPL.bin	/dev/mmcblk2p1
	2nd Partition	1 MB	-	bl31.bin.img	/dev/mmcblk2p2
	3rd Partition	1 MB	-	u-boot.bin.img	/dev/mmcblk2p3
	4th Partition	1 MB	-	u-boot env	/dev/mmcblk2p4
	5th Partition	31 MB	-	reserved	/dev/mmcblk2p5
	6th Partition	64 MB	-	Image.fit	/dev/mmcblk2p6
	7th Partition	6 GB	ext4	root file system	/dev/mmcblk2p7
	8th Partition	≈23 GB	ext4	user file system	/dev/mmcblk2p8

Figure 6.2 eMMC Partition

A file ending with ".img" is a ".bin" file with a header prepended to the beginning, containing information such as the file size.

The root file system uses a file system provided by the open-source Yocto Project. The directory below shows the Root file system configuration.

```
lrwxrwxrwx 1 root root 7 Sep 12 13:29 bin -> usr/bin
drwx------ 2 root root 4096 Sep 12 13:53 boot
drwxr-xr-x 2 root root 4096 Sep 24 06:00 dev
drwxr-xr-x 18 root root 4096 Sep 23 13:32 etc
drwxr-xr-x 2 root root 4096 Sep 12 13:29 home
lrwxrwxrwx 1 root root 7 Sep 12 13:29 lib -> usr/lib
drwxr-xr-x 2 root root 4096 Sep 24 06:00 media
drwxr-xr-x 2 root root 4096 Sep 24 06:00 mnt
drwxr-xr-x 2 root root 4096 Sep 24 06:00 proc
drwx------ 3 root root 4096 Sep 12 13:53 root
drwxr-xr-x 2 root root 4096 Sep 12 13:53 run
lrwxrwxrwx 1 root root 8 Sep 12 13:29 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Sep 12 13:29 srv
drwxr-xr-x 2 root root 4096 Sep 24 06:00 sys
drwxrwxr-x 2 root root 4096 Sep 24 06:00 tmp
drwxr-xr-x 8 root root 4096 Sep 23 13:38 usr
drwxr-xr-x 9 root root 4096 Sep 12 13:53 var
```

In REGULUS, ”ssh” and ”scp” commands are supported for file exchange and remote terminal capabilities with the remote device. Since the application field involves applying a Deep Learning Model to video input from an external camera to obtain the desired results, OpenCV and Gstreamer are supported to support this application field.

6.2 Creating Root File System

This section describes the procedure for creating a new **rootfs.tar.gz** on the Host PC. It explains the process of creating a new root file system by adding the developed application to the root file system provided by the toolchain. The process involves including the added files in the root file system using the **rootfs.tar.gz** included in the toolchain

```
In Host PC
$ mkdir new-rootfs
$ cd new-rootfs
$ tar -xzf ../rootfs.tar.gz
==> Copy the application to the desired directory.
$ tar -czf ../newrootfs.tar.gz .
$ cd ..
$ mv rootfs.tar.gz oldrootfs.tar.gz
$ mv newrootfs.tar.gz rootfs.tar.gz
```

If the above command is executed normally, new **rootfs.tar.gz** will be created.

6.3 Root File System Update

The procedure for applying the newly created root file system to the **regulus-evk-rev01** board is as follows.

```
In Host PC
1. copy ”rootfs.tar.gz” into  ”~/.firmware_update” directory in
regulus-evk-rev01 board

In regulus-evk-rev01 board
$ firmwareupdatetool
  ==> select ”Firmware Update (GUI Mode)”
  ==> select ”rootfs”
```

7 U-boot Build

In REGULUS, booting can be done using either an SD card or eMMC, and the u-boot image used differs for each booting mode. This chapter describes the procedure for building the u-boot image for SD card or eMMC.

7.1 U-boot image for SD-Card

To create a u-boot image for an SD card, you need to modify the following file in the u-boot source tree.

```
include/configs/reguls_evk.h
```

The definition of "CONFIG_EXTRA_ENV_SETTINGS" from the file above is applied as follows:

```
#define CONFIG_EXTRA_ENV_SETTINGS   "bootdelay=2\0"     "hostname=REGULUS\0"
"fdt_file="CONFIG_DEFAULT_DEVICE_TREE".dtb\0"   "fdt_addr=0x900000000\0"
"load_addr=0x903000000\0"   "ipaddr=192.168.0.67\0"
"serverip=192.168.0.22\0"   "netmask=255.255.255.0\0"
"ethaddr=00:0a:35:00:06:49\0"   "eth1addr=00:a8:44:1f:1f:11\0"
"sd_rootdev=/dev/mmcblk0p2\0"   "mmc_rootdev=/dev/mmcblk2p2\0"
"console=ttyS0\0"   "baudrate=115200n8\0"   "othbootargs=ignore_loglevel\0"
"setbootargs=setenv bootargs "                          "rdinit=/sbin/init rw
console=${console},"              "${baudrate} ${othbootargs}\0"
"load_sd_image=setenv rootdev ${sd_rootdev}; fatload mmc 0:1 ${load_addr} Image;
fatload mmc 0:1 ${fdt_addr} ${fdt_file};\0"    "load_emmc_image=setenv rootdev
${mmc_rootdev}; fatload mmc 2:1 ${load_addr} Image; fatload mmc 2:1 ${fdt_addr}
${fdt_file};\0"     "load_script=run load_sd_image \0"  "bootcmd=run
load_script; run setbootargs; booti ${load_addr} - ${fdt_addr}; \0"
```

To compile the modified u-boot source code, execute the cross-compilation environment setup script and perform the compilation to create the u-boot image.

$ source /opt/crosstools/mobilint/2.0.0/v4.X.X/environment-setup-cortexa53-mobilint-linux
$ make ARCH=arm

When cross-compilation completes successfully, "u-boot.bin" is generated. Copy the newly generated "u-boot.bin" to the first partition of the booting SD card. Afterward, you can boot the REGULUS EVK board using the newly created u-boot image.

7.2 U-boot Image for eMMC

To create a u-boot image for eMMC, modify the following file in the u-boot source tree.

```
include/configs/reguls_evk.h
```

The definition of "CONFIG_EXTRA_ENV_SETTINGS" from the file above is applied as follows:

```
#define CONFIG_EXTRA_ENV_SETTINGS   "bootdelay=2\0"     "hostname=REGULUS\0"
"fit_start_lba=0x12000\0"   "fit_part_size=0x20000\0"   "fit_addr=0x800000000\0"
   "fdt_file="CONFIG_DEFAULT_DEVICE_TREE".dtb\0"   "fdt_addr=0x900000000\0"
"kernel_file=Image\0"   "kernel_addr=0x903000000\0"
"initramfs_file=initrd.img\0" "initramfs_addr=0x905000000\0"
"ipaddr=192.168.0.67\0"     "serverip=192.168.0.22\0"
"netmask=255.255.255.0\0"   "ethaddr=00:0a:35:00:22:01\0"
"eth1addr=00:a8:44:1f:1f:11\0"  "console=ttyS0\0"   "baudrate=115200n8\0"
"othbootargs=ignore_loglevel LABEL=install\0"   "setbootargs=setenv bootargs "
                      "root=/dev/ram0 rdinit=/init rw console=${console},"
          "${baudrate} ${othbootargs}\0"              "find_fit_part=mmc dev 2;
gpt setenv mmc 2 Image; setenv fit_start_lba ${gpt_partition_addr}; setenv
fit_part_size ${gpt_partition_size};\0"    "load_fit_image=mmc read ${fit_addr}
${fit_start_lba} ${fit_part_size};\0"  "extract_fit_image=imxtract ${fit_addr}
kernel ${kernel_addr}; imxtract ${fit_addr} ramdisk ${initramfs_addr}; imxtract
${fit_addr} fdt ${fdt_addr};\0"     "load_emmc_image=run find_fit_part; run
load_fit_image; run extract_fit_image\0"    "load_sd_image=fatload mmc 0:1
${kernel_addr} ${kernel_file}; fatload mmc 0:1 ${fdt_addr} ${fdt_file}; fatload
mmc 0:1 ${initramfs_addr} ${initramfs_file};\0"  "bootcmd_emmc=run
load_emmc_image; run setbootargs; booti ${kernel_addr} ${initramfs_addr}
${fdt_addr}; \0"     "bootcmd_sd=run load_sd_image; run setbootargs; booti
${kernel_addr} ${initramfs_addr} ${fdt_addr}; \0"     "bootcmd=run bootcmd_sd
\0"
```

To compile the modified u-boot source code, execute the cross-compilation environment setup script and then perform the compilation to create the u-boot image

$ source /opt/crosstools/mobilint/2.0.0/v4.X.X/environment-setup-cortexa53-mobilint-linux
$ make ARCH=arm

After cross-compilation completes successfully, "u-boot.bin" will be generated. Copy the newly generated u-boot.bin to the REGULUS EVK board as follows:

```
in REGULUS EVK board

1. copy "u-boot.bin" in Host PC into "~/.firmware_update"
2. $ firmwareupdatetool
3. select "Firmware Update (from Local)"
4. select "U-Boot"
5. After "Press any key to proceed with reboot" message, press "Enter" key
6. REGULUS EVK board will be rebooted
```

When the above command is executed, the new "u-boot.bin" will be stored in the 3rd partition of the eMMC, and the changed "u-boot.bin" will be used during booting.

Index

Build Root File System, 17

Document Revision History, 1

Installing Module File, 16
Introduction, 3

Kernel Build Guide, 7
Kernel Image Update, 12

Linux Kernel Source Code, 4
List of Terms, 2

U-boot Build, 19

Mobilint, Inc.

3F, Narakium Yeoksam B Building, 35 Seolleung-ro 93-gil, Gangnam-gu, Seoul, Republic of Korea (06151)

Corporate URL
www.mobilint.com

Contact
tech-support@mobilint.com
+82 2 6281 0489