User's Manual

Linux Interface Specification Yocto recipe Start-Up Guide

User's Manual: Software

RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C

R8A77420
R8A77430
R8A77440
R8A77450
R8A77470

Renesas Electronics
www.renesas.com

Rev2.12 Dec. 2020

How to Use This Manual

- **[Readers]**
  This manual is intended for engineers who develop products which use the RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C processors.

- **[Purpose]**
  This manual is intended to give users an understanding of the functions of the RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C processor device driver and to serve as a reference for developing hardware and software for systems that use this driver.

- **[How to Read This Manual]**
  It is assumed that the readers of this manual have general knowledge in the fields of electrical
  — Engineering, logic circuits, microcontrollers, and Linux.
  → Read this manual in the order of the CONTENTS.
  — To understand the functions of a multimedia processor for RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C.
  → See the RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C User's Manual.
  — To know the electrical specifications of the multimedia processor for RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C.
  → See the RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Data Sheet.

- **[Conventions]**
  The following symbols are used in this manual.
  Data significance: Higher digits on the left and lower digits on the right
  Note: Footnote for item marked with Note in the text
  Caution: Information requiring particular attention
  Remark: Supplementary information
  Numeric representation: Binary ... ××××, 0b××××, or ××××B
  Decimal ... ××××
  Hexadecimal ... 0x×××× or ××××H
  Data type: Word … 32 bits
  Half word ... 16 bits
  Byte ... 8 bits

Table of Contents

Introduction

This start-up guide explains RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Yocto recipe package files, the system environments, the make method of kernel, the operating of U-Boot and so on.

This product RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Yocto recipe is a basic package to operate built-in Linux and basic middleware on the iW-RainboW-G21D RZ/G1H Qseven Development Platform, iW-RainboW-G20D RZ/G1M Qseven Development Platform, iW-RainboW-G20D RZ/G1N Qseven Development Platform, iW-RainboW-G22D RZ/G1E SODIMM Development Platform and iW-RainboW-G23S RZ/G1C Single Board Computer. Please contact Renesas Electronics person who provided this product to you in case of questions.

1. RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux BSP package files

This Yocto recipe will be taken

The U-Boot source code from:

git://github.com/renesas-rz/renesas-u-boot-cip.git;branch=2013.01.01/rzg1-iwave

RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux source code from:

https://github.com/renesas-rz/renesas-cip.git;branch=v4.4.*x*-cip*y* (for normal linux)
https://github.com/renesas-rz/renesas-cip.git;branch=v4.4.*x*-cip*y*-rt (for realtime linux)

1.1 Reference (RZ/G1H)

Document name	Version	Date
RZ/G1H User's Manual: Hardware	Rev.1.00	2016.09.30
RZ/G Series, User's Manual: Hardware	Rev.1.00	2016.09.30
iW-RainboW-G21M RZ/G1H Qseven SOM Hardware User Guide	-	-
iW-RainboW-G21D RZ/G1H Qseven Development Platform Hardware User Guide	-	-

1.2 Reference (RZ/G1M)

Document name	Version	Date
RZ/G1M User's Manual: Hardware	Rev.1.00	2016.09.30
RZ/G Series, User's Manual: Hardware	Rev.1.00	2016.09.30
iW-RainboW-G20M RZ/G1M Qseven SOM Hardware User Guide	-	-
iW-RainboW-G20D RZ/G1M Qseven Development Platform Hardware User Guide	-	-

1.3 Reference (RZ/G1N)

Document name	Version	Date
RZ/G1N User's Manual: Hardware	Rev.1.00	2016.09.30
RZ/G Series, User's Manual: Hardware	Rev.1.00	2016.09.30
iW-RainboW-G20M RZ/G1N Qseven SOM Hardware User Guide	-	-
iW-RainboW-G20D RZ/G1N Qseven Development Platform Hardware User Guide	-	-

1.4 Reference (RZ/G1E)

Document name	Version	Date
RZ/G1E User's Manual: Hardware	Rev.1.00	2016.09.30
RZ/G Series, User’s Manual: Hardware	Rev.1.00	2016.09.30
iW-RainboW-G22M RZ/G1E SODIMM SOM Hardware User Guide	-	-
iW-RainboW-G22M RZ/G1E SODIMM Development Kit Hardware User Guide	-	-

1.5 Reference (RZ/G1C)

Document name	Version	Date
RZ/G1C User's Manual: Hardware	Rev.1.00	2017.10.13
RZ/G Series, User’s Manual: Hardware	Rev.1.00	2016.09.30
iW-RainboW-G23S RZ/G1C SBC Hardware User Guide	-	-

2. ENVIRONMENTAL REQUIREMENT

2.1 Setting of parts

Host PC and terminal software are necessary for the operation of this product. Furthermore, Ethernet cable is required to use NFS mount function. Please refer to Table 1.

Table 1 RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux BSP Environmental Requirement

Equipment	Explanation
Linux Host PC	Ubuntu 16.04 LTS is recommended as OS It is used as building and debugging environment. It is used as TFTP server and NFS server.
Windows Host PC	Windows 7 or 10 are recommended as OS. It is used as debugging environment. Terminal software and VCP driver are executed.
Terminal software	Please use following software 1) or 2). 1) Hyperterm (Included in WindowsXP) 2) TeraTerm (Confirmed with Japanese version of Tera Term 4.72 Available at http://sourceforge.jp/projects/ttssh2)
VCP driver	Please install following VCP driver to Windows Host PC before you connect Windows Host PC and the RZ/G1 board. Latest VCP Driver for Windows: ➢ Available at http://www.ftdichip.com/Drivers/VCP.htm
TFTP server software	It is used when SPI Flash is written by U-Boot or zImage/uImage is downloaded.
NFS server software	It is used when File system is mounted by NFS.

Recommended Environment

The following shows a Recommended Environment.

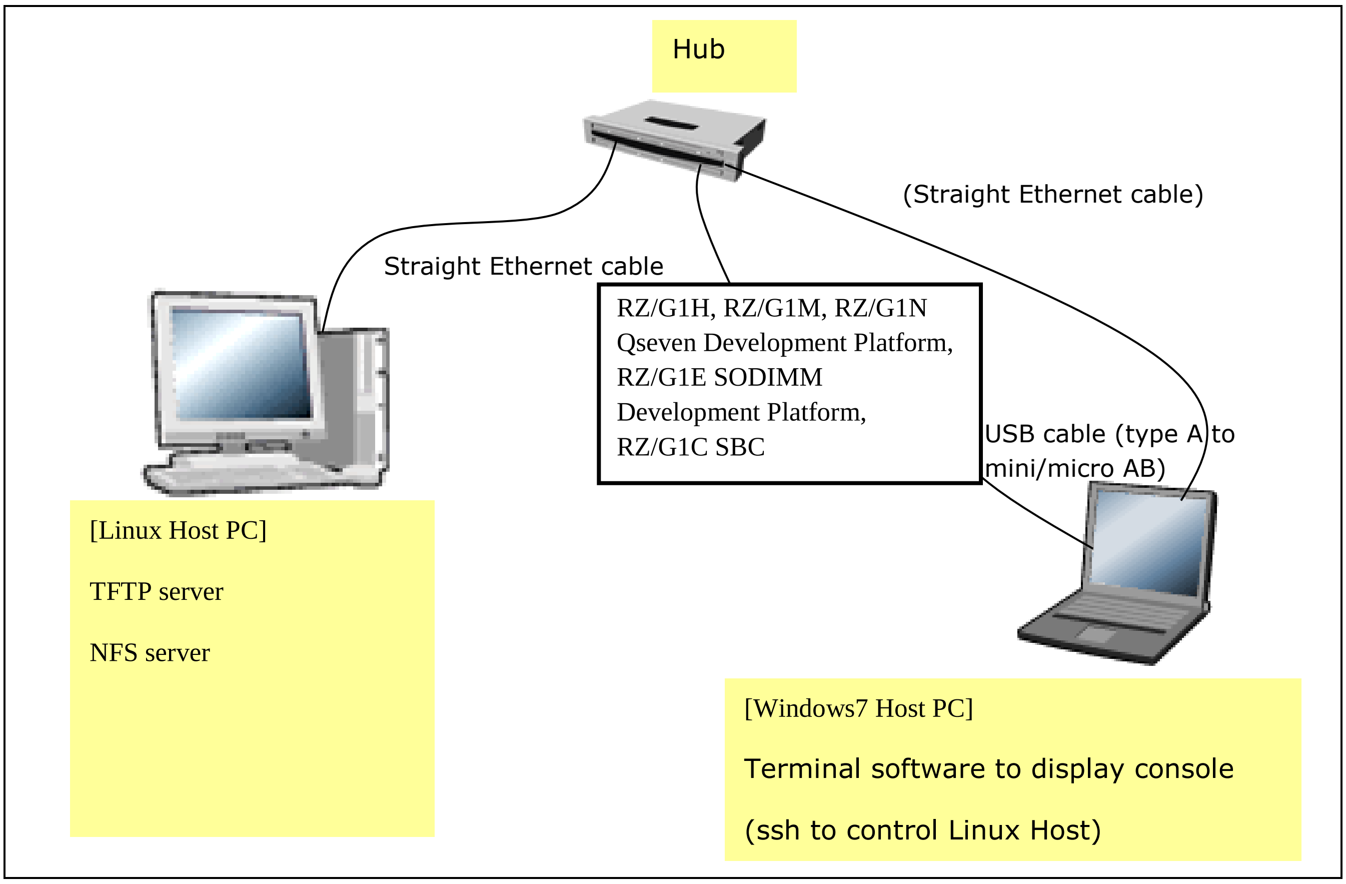

Hub
(Straight Ethernet cable)
Straight Ethernet cable
RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform, RZ/G1E SODIMM Development Platform, RZ/G1C SBC
USB cable (type A to mini/micro AB)
[Linux Host PC]
TFTP server
NFS server
[Windows7 Host PC]
Terminal software to display console
(ssh to control Linux Host)

Figure 1 Recommended Environment for RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux BSP

Note) Functions in covered with () are optional.

2.2 Setting of dip switch

The setting of RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform's dip switches is shown the following Table 2. Please refer to “iW-RainboW-G21D RZ/G1H Qseven Development Kit Hardware User Guide”, “iW-RainboW-G20D RZ/G1M RZ/G1N Qseven Development Kit Hardware User Guide” and “iW-RainboW-G22D-SODIMM RZ/G1E SODIMM Development Kit Hardware User Guide” for details.

Table 2 setting of Dip switches (RZ/G1H, RZ/G1M, RZ/G1N)

Switch2 Bits	Switch2 Bit Name	Description	
		OFF	ON
1	BIOS_DISABLE#	-	Set
2	BATLOW#	-	Set
3	LID_BTN#	-	Set
4	USB_SELECT	-	Set
5	CODEC_SELECT	-	Set
6	PCIe_SELECT	Set	-
7	DEBUG_SELECT	-	Set
8	USB_ID	Set	-

Note) SW2 Bit 1 to Bit 5 are not used in RZ/G1H/G1M/G1N carrier board.

Table 3 setting of Dip switches (RZ/G1E)

Switch4 Bits	Switch4 Pin Name	Description	
		OFF	ON
1	GPIO974(GP1_14)	-	Set
2	GPIO865(GP4_1)	Set	-

Note) iW-RainboW-G23S RZ/G1C Single Board Computer does not have Dip switch.

3. Building Instructions

You can build BSP by using Yocto Project. Please execute following steps in $WORK directory on Linux Host PC. Filesystem by making following instruction is the one for testing current BSP package in Renesas. Please note that Renesas has not been verified with any other build configuration or modified recipes except "core-image-weston" configuration which is based on upstream Yocto Project deliverables and some additional packages correspond to gstreamer.

Note) Renesas executed following instructions with clean $WORK/build directory. You may use wipe-sysroot and/or bitbake -c cleansstate to reflect modifications of configuration files for Recipe as in open source Yocto Project's standards, however Renesas strongly recommend to use recipe with clean $WORK/build directory for each configurations because there are some implicit dependency for header files exist to keep compatibility between application build scheme with/without RZ/G Multimedia Package.

If you want to build image with demo packages, please follow the instructions in 3.1, then go to 3.2. If you want to build image without demo packages, please execute 3.1 and 3.3.

3.1 Common instructions

Step 1 installation of required commands

Ubuntu is used as Linux Host PC since Yocto Project Quick Start specifies Ubuntu as one of the distributions. In case of that you can install the required commands as follows.
Please refer to yocto website for detail:
https://www.yoctoproject.org/docs/2.4.2/yocto-project-qs/yocto-project-qs.html

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python python3 \
python3-pip python3-pexpect \
xz-utils debianutils iputils-ping libsdl1.2-dev xterm
```

Note) There is a bitbake command in $WORK/poky/scripts/. Command path is available after step 6.
Note) When you use terminal interactions to build such as menuconfig under non-X terminal (ssh, etc.), please install "screen" command package to Host PC
Note) Below commands are required in case of building HTML5 version

```
$ sudo apt-get install autoconf2.13 clang llvm clang-3.9 llvm-3.9
```

Step 2 download of required files

Required files (poky, meta-openembedded, meta-linaro, meta-renesas, meta-qt5) are downloaded by git clone.

```
$ export WORK=<path-to-your-build-directory>
$ mkdir $WORK
$ cd $WORK
$ git clone git://git.yoctoproject.org/poky
$ git clone git://git.openembedded.org/meta-openembedded
$ git clone git://git.linaro.org/openembedded/meta-linaro.git
$ git clone https://github.com/meta-qt5/meta-qt5.git
$ git clone https://github.com/renesas-rz/meta-renesas
```

Below files are required in case of building HTML5 version

```
$ git clone https://github.com/webdino/meta-gecko-embedded.git
$ git clone -b firefox-60esr https://github.com/webdino/meta-browser.git
$ git clone -b jethro-14.0.1_rust_1.24.1 https://github.com/webdino/meta-
rust.git
```

Step 3 checkout and cherry-pick patches

Please checkout available version of each git clone.

```
$ cd $WORK/poky
$ git checkout -b tmp 342fbd6a3e57021c8e28b124b3adb241936f3d9d
$ cd $WORK/meta-openembedded
$ git checkout -b tmp dacfa2b1920e285531bec55cd2f08743390aaf57
$ cd $WORK/meta-linaro
$ git checkout -b tmp 75dfb67bbb14a70cd47afda9726e2e1c76731885
$ cd $WORK/meta-qt5
$ git checkout -b tmp c1b0c9f546289b1592d7a895640de103723a0305
$ cd $WORK/meta-renesas

Please choose a specific version
e.g.   If BSP version is v2.1.1 (support all RZ/G1H,M,N,E,C)
$ git checkout certified-linux-v2.1.1

[Tag name : Feature]
certified-linux-v2.1.x        : support normal linux
certified-linux-v2.1.x-rt     : support realtime linux
```

Note) tmp is a temporary name of a local branch. We can use checkout command without branch. Please note that HEAD refers directly to commit (detached HEAD).

3.2 Build images with demo packages

When you use demo packages (Multimedia and Graphics application for demonstration) from Renesas, please execute as following steps. Otherwise please skip this section and go to section 3.3.

Step 4 download demo packages, checkout and cherry-pick patches

```
$ cd $WORK
$ git clone https://github.com/renesas-rz/meta-rzg-demos
$ cd $WORK/meta-rzg-demos

Please choose a specific version.
e.g.   If BSP version is v2.1.1
$ git checkout certified-linux-v2.1.1

[Tag name : Feature]
certified-linux-v2.1.x          : support normal and realtime linux
```

Step 5 copy RZ/G Multimedia Package into recipe directory structure

Download and extract RZ/G Multimedia Package link on **RZ/G Marketplace**:

- America: https://mp.renesas.com/en-us/rzg/marketplace/linux_package/index.html
- Europe : https://mp.renesas.com/en-eu/rzg/marketplace/linux_package/index.html
- Asia : https://mp.renesas.com/en-sg/rzg/marketplace/linux_package/index.html
- Japan : https://mp.renesas.com/ja-jp/rzg/marketplace/linux_package/index.html

Copy downloaded package into a folder and extract the downloaded package then please execute following instruction:

```
$ mkdir $WORK/MMP
$ cp RZG_Series_*_Software_Package* $WORK/MMP
$ cd $WORK/MMP
$ unzip RZG_Series_Software_Package_for_Linux*.tar.gz.zip
$ unzip RZG_Series_Software_Package_of_Linux_Drivers*.tar.gz.zip
```

To use RZ/G Multimedia Package from Renesas, please copy deliverables of those software into recipe directory structure. Renesas provide shell script to copy those software.

```
$ cd $WORK/meta-renesas/meta-rzg1
$ sh copy_mm_software_lcb.sh $WORK/MMP
$ sh copy_gfx_software_rzg1h.sh $WORK/MMP (for RZ/G1H)
$ sh copy_gfx_software_rzg1m.sh $WORK/MMP (for RZ/G1M)
$ sh copy_gfx_software_rzg1n.sh $WORK/MMP (for RZ/G1N)
$ sh copy_gfx_software_rzg1e.sh $WORK/MMP (for RZ/G1E)
$ sh copy_gfx_software_rzg1c.sh $WORK/MMP (for RZ/G1C)
```

Step 6 execute source command

Please execute source command with oe-init-build-env for setting environment.

```
$ cd $WORK
$ source poky/oe-init-build-env
```

Step 7 copy bblayers.conf and local.conf

Please copy configuration files from demo packages.

```
$ cp $WORK/meta-rzg-demos/meta-rzg1/<demo-packages-
name>/template/<board>(1)/bblayers.conf $WORK/build/conf/bblayers.conf
$ cp $WORK/meta-rzg-demos/meta-rzg1/<demo-packages-
name>/template/<board>(1)/local.conf $WORK/build/conf/local.conf
```

(1) <board> is board name: **iwg21m** (iWave RZ/G1H), **iwg20m-g1m** (iWave RZ/G1M), **iwg20m-g1n** (RZ/G1N), **iwg22m** (iWave RZ/G1E), **iwg23s** (iWave RZ/G1C).

Note) When you have to take care about the hardware computation resource of HOST PC, please review and modify definitions for capacity limitation, such as “BB_NUMBER_THREADS” or “PARALLEL_MAKE” on $WORK/build/conf/local.conf, and also please review timeout definitions of package download (wget, etc.) described in $WORK/poky/meta/conf/bitbake.conf.

Note) It’s reported some open source toolchain for HOST PC has issues on parallel build (make 3.81 e.g.), so if you have dependency error while in build sequence (such as the compilation error on unsettled package for each build), please review definitions for capacity limitation described in above. For example, reduce the number of parallel make with ‘PARALLEL_MAKE = “-j 1” ’ which described in $WORK/conf/local.conf. And also, please review the number of that in case of memory starvation occurred in a HOST PC which has many CPU Cores.

Step 8 enable Multimedia package

Please modify configurations in $WORK/build/conf/local.conf by following instructions

The following standard multimedia packages are enabled

No.	Functions	Explanation
1	MMNGR	Memory manager driver & shared libraries
2	S3CTRL driver	S3 cache controller driver & shared libraries
3	FDPM driver	FDP driver & shared libraries
4	VSPM driver	VSP driver & shared libraries
5	UVCS	UVCS Common Engine Library, UVCS Decode Engine Library
6	VCP3	VCP3 common/core parts
7	OMX	OMX common parts
8	H.264	H264 decoder
		H264 encoder

To enable optional multimedia functions, please add DISTRO_FEATURES_append to $WORK/build/conf/local.conf as DISTRO_FEATURES_append = " <function name>".

Note) These configurations exist near the end of local.conf.

Note) DISTRO_FEATURES_append are commented out by the default. To enable functions, please uncomment it.

For example
[Disable (default)]

```
#DISTRO_FEATURES_append = " h264avcenc_lib"
```

[Enable]

```
DISTRO_FEATURES_append = " h264avcenc_lib"
```

Step 9 building with bitbake

Please build as follows. The file system (core-image-weston-<supported board name>.tar.bz2) is created in $WORK/build/tmp/deploy/images/<supported board name>/ directory.

Note) <supported board name> is the one of the following: **iwg21m** (iWave RZ/G1H), **iwg20m-g1m** (iWave RZ/G1M), **iwg20m-g1n** (RZ/G1N), **iwg22m** (iWave RZ/G1E), **iwg23s** (iWave RZ/G1C).

Note) Build by bitbake might need several hours under the influence of Linux Host PC performance and network environment.

Note) The bitbake downloads some package while building. Then the bitbake might stop for network timeout or link error. In this case, please get applicable package in $WORK/build/downloads directory whenever build stops by wget command, or please review timeout definitions of package download (wget, etc.) described in $WORK/poky/meta/conf/bitbake.conf.

Note) It's reported some open source toolchain for HOST PC has issues on parallel build (make 3.81 e.g.), so if you have dependency error while in build sequence (such as the compilation error on unsettled package for each build), please review definitions for capacity limitation described in Step 7. For example, reduce the number of parallel make with 'PALLARLEL_MAKE = "-j 1" ' which described in $WORK/build/conf/local.conf.

```
$ cd $WORK/build
$ bitbake core-image-weston
```

Note) When you use a configuration for Wayland, following messages would be displayed. It's not the issue because it not used mesa in Wayland environment but just a multiple dependency description exist which designate same sub-modules of mesa.

```
NOTE: multiple providers are available for virtual/mesa (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match virtual/mesa
NOTE: multiple providers are available for virtual/libgl (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match virtual/libgl
NOTE: multiple providers are available for runtime libgl (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match libgl
```

If you need to suppress above messages, please add a following line to local.conf

```
BBMASK .= "|mesa-gl"
```

3.3 Build images without demos package

Step 4 copy RZ/G Multimedia Package into recipe directory structure

Download and extract RZ/G Multimedia Package link on **RZ/G Marketplace**:

- America: https://mp.renesas.com/en-us/rzg/marketplace/linux_package/index.html
- Europe : https://mp.renesas.com/en-eu/rzg/marketplace/linux_package/index.html
- Asia : https://mp.renesas.com/en-sg/rzg/marketplace/linux_package/index.html
- Japan : https://mp.renesas.com/ja-jp/rzg/marketplace/linux_package/index.html

Copy downloaded package into a folder and extract the downloaded package then please execute following instruction:

```
$ mkdir $WORK/MMP
$ cp RZG_Series_Evaluation_Software_Package* $WORK/MMP
$ cd $WORK/MMP
$ unzip RZG_Series_Evaluation_Software_Package_for_Linux*.tar.gz.zip
$ unzip RZG_Series_Evaluation_Software_Package_of_Linux_Drivers*.tar.gz.zip
```

To use RZ/G Multimedia Package from Renesas, please copy deliverables of those software into recipe directory structure. Renesas provide shell script to copy those software.

```
$ cd $WORK/meta-renesas/meta-rzg1
$ sh copy_mm_software_lcb.sh $WORK/MMP
$ sh copy_gfx_software_rzg1h.sh $WORK/MMP (for RZ/G1H)
$ sh copy_gfx_software_rzg1m.sh $WORK/MMP (for RZ/G1M)
$ sh copy_gfx_software_rzg1n.sh $WORK/MMP (for RZ/G1N)
$ sh copy_gfx_software_rzg1e.sh $WORK/MMP (for RZ/G1E)
$ sh copy_gfx_software_rzg1c.sh $WORK/MMP (for RZ/G1C)
```

Step 5 execute source command

Please execute source command with oe-init-build-env for setting environment.

```
$ cd $WORK
$ source poky/oe-init-build-env
```

Step 6 copy bblayers.conf and local.conf

Please copy configuration files for corresponding board. There are three options:

```
### 1. No GUI framework
$ cp $WORK/meta-renesas/meta-rzg1/templates/<board>(1)/*.conf ./conf
### 2. Enable Gecko (HTML5)
$ cp $WORK/meta-renesas/meta-rzg1/templates/<board>(1)/gecko/*.conf
./conf
### 3. Enable Qt
$ cp $WORK/meta-renesas/meta-rzg1/templates/<board>(1)/qt/*.conf ./conf
```

(1) <board> is board name which can be one of the following values: **iwg21m** (iWave RZ/G1H), **iwg20m-g1m** (iWave RZ/G1M), **iwg20m-g1n** (RZ/G1N), **iwg22m** (iWave RZ/G1E), **iwg23s** (iWave RZ/G1C).

Note) When you have to take care about the hardware computation resource of HOST PC, please review and modify definitions for capacity limitation, such as "BB_NUMBER_THREADS" or "PARALLEL_MAKE" on $WORK/build/conf/local.conf, and also please review timeout definitions of package download (wget, etc.) described in $WORK/poky/meta/conf/bitbake.conf.

Note) It's reported some open source toolchain for HOST PC has issues on parallel build (make 3.81 e.g.), so if you have dependency error while in build sequence (such as the compilation error on unsettled package for each build), please review definitions for capacity limitation described in above. For example, reduce the number of parallel make with 'PARALLEL_MAKE = "-j 1" ' which described in $WORK/conf/local.conf. And also please review the number of that in case of memory starvation occurred in a HOST PC which has many CPU Cores.

Step 7 building with bitbake

Please build as follows. The file system (*core-image-weston-<supported board name>.tar.bz2*) is created in *$WORK/build/tmp/deploy/images/<supported board name>/* directory.

Note) <supported board name> is the one of the following: **iwg21m** (iWave RZ/G1H), **iwg20m-g1m** (iWave RZ/G1M), **iwg20m-g1n** (RZ/G1N), **iwg22m** (iWave RZ/G1E), **iwg23s** (iWave RZ/G1C).

Note) Build by bitbake might need several hours depending on the Linux Host PC performance and network environment.

Note) The bitbake downloads some packages while building and may stop for network timeout or link error. In this case, please get applicable package in $WORK/build/downloads directory whenever build stops by wget command, or please review timeout definitions of package download (wget, etc.) described in $WORK/poky/meta/conf/bitbake.conf.

Note) It's reported some open source toolchain for HOST PC has issues on parallel build (make 3.81 e.g.), so if you have dependency error while in build sequence (such as the compilation error on unsettled package for each build), please review definitions for capacity limitation described in Step 7. For example, reduce the number of parallel make with 'PALLARLEL_MAKE = "-j 1" ' which described in $WORK/conf/local.conf.

```
$ cd $WORK/build
$ bitbake core-image-weston
```

Note) When you use a configuration for Wayland, following messages would be displayed. It's not the issue because it not used mesa in Wayland environment but just a multiple dependency description exist which designate same sub-modules of mesa

```
NOTE: multiple providers are available for virtual/mesa (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match virtual/mesa
NOTE: multiple providers are available for virtual/libgl (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match virtual/libgl
NOTE: multiple providers are available for runtime libgl (mesa, mesa-gl)
NOTE: consider defining a PREFERRED_PROVIDER entry to match libgl
```

If you need to suppress above messages, please add a following line to local.conf

```
BBMASK .= "|mesa-gl"
```

4. Confirm starting of U-Boot and Linux

Please connect iW-RainboW-G21D RZ/G1H Qseven Development Platform, iW-RainboW-G20D RZ/G1M Qseven Development Platform, iW-RainboW-G20D RZ/G1N Qseven Development Platform, iW-RainboW-G22D RZ/G1E SODIMM Development Platform, iW-RainboW-G23S RZ/G1C Single Board Computer, Windows Host PC with terminal software for console and Linux Host PC with TFTP and NFS server as Figure 1. Then please confirm normal starting of U-Boot and Linux with following step.

In case of RZ/G1H, RZ/G1M, RZ/G1N, Ethernet connector is J22 (Top), serial connector is J3, Display connector is J10, SD slots are J2/J30, USB Host connectors are J2, J22 and J23 (Bottom), Audio out connector is J5, Audio in connector is J6.

In case of RZ/G1E, Ethernet connector is J5, serial connector is J8, Display connectors is J23, SD slot is J2/J19, USB Host connectors are J4 and J5, Audio IN/OUT connector is J3.

In case of RZ/G1C, Ethernet connector is J10, Display connectors is J12, SD slot is J15, USB Host connectors are 20 PIN Expansion connector J7.

Step 1 setting Linux Host PC

Please install TFTP server and NFS server in Linux Host PC with apt-get command and so on. Please set /etc/xinetd.d/tftp of TFTP server and /etc/exports of NFS server according to your environment.

Step 2 connect cable

Connect USB Host connector of Windows Host PC that is virtual COM port to J3 of RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform, J8 of RZ/G1E SODIMM Development Platform and 20 PIN Expansion connector of RZ/G1C SBC with USB cable for displaying console.

Step 3 setting the terminal software

Activate the Terminal Software on Windows Host PC. Configure the Terminal Software on Windows Host PC as followings. Please refer to Table 1 about the VCP driver for making a USB host connector into a virtual COM port.

For RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform, RZ/G1E SODIMM Development Platform, RZ/G1C SBC:

- [setting value] baud rate 115200, 8bit data, parity none, stop 1 bit, flow control none

Step 4 update U-boot

For more information about boot loader and its storage, please follow the instruction in the User Manual provided in the board (RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform, RZ/G1E SODIMM Development Platform, RZ/G1C SBC).

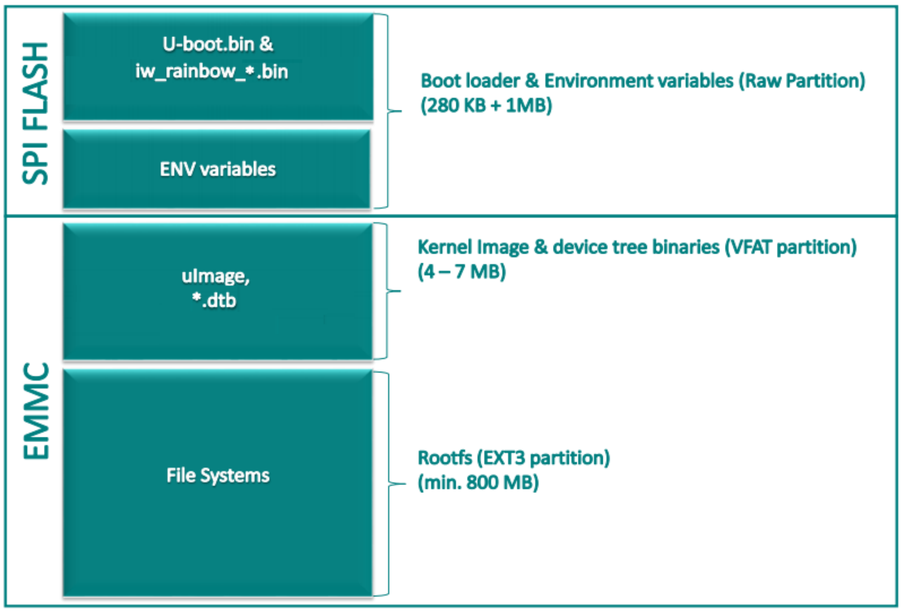

SPI FLASH
U-boot.bin & iw_rainbow_*.bin
ENV variables
Boot loader & Environment variables (Raw Partition) (280 KB + 1MB)
EMMC
uImage, *.dtb
Kernel Image & device tree binaries (VFAT partition) (4 – 7 MB)
File Systems
Rootfs (EXT3 partition) (min. 800 MB)

Figure 2 Boot device memory layout

Note:
Loader: iw_rainbow_*.bin
RZ/G1H: iw_rainbow_G21M_SPI_loader_v031.bin
RZ/G1M, RZ/G1N: iw_rainbow_g20m_SPI_loader_v020_ddr3.bin
RZ/G1E: iw_rainbow_G22M_SPI_loader_v031.bin
RZ/G1C: iW_RainboW_G23S_SPI_LOADER_V020_DDR3_E6300000_V100.bin

Kernel device tree: *.dtb
RZ/G1H: uImage-r8a7742-iwg21d-q7-dbcm-ca.dtb
RZ/G1M: uImage-r8a7743-iwg20d-q7-dbcm-ca.dtb
RZ/G1N: uImage-r8a7744-iwg20m.dtb
RZ/G1E: uImage-r8a7745-iwg22d-sodimm.dtb
RZ/G1C: uImage-r8a77470-iwg23s-sbc.dtb

This section explains the step by step procedure to flash the binaries into RZ/G1M/G1N SOM through JTAG Debugger.

Note) Example for JTAG Debugger name: lauterbach, …

- The prebuilt binaries are built by section 3.
- Copy the JTAG debugger configuration file and the binary files uboot (uboot.bin), loader (iw_rainbow_G21M_SPI_loader_v031.bin for RZ/G1H, iw_rainbow_g20m_SPI_loader_v020_ddr3.bin for RZ/G1M, RZ/G1N, iw_rainbow_G22M_SPI_loader_v031.bin for RZ/G1E, iW_RainboW_G23S_SPI_LOADER_V020_DDR3_E6300000_V100.bin for RZ/G1C) required for booting RZ/G1M/G1N SOM into the SD card, insert this SD card into JTAG Debugger.
- Open minicom for JTAG Debugger and Target board separately in host PC.
- Power on the JTAG Debugger and then target board.
- Once the JTAG Debugger is booted, set the following parameters in JTAG terminal to boot the target board.
- To set the supervisor mode interrupts, execute the below command.
  RZ/G1x > set cpsr 0x1D3
- To set the stack pointer for loader, execute the below command.
  RZ/G1x > set sp 0xE6303D00
- To set the program counter for loader, execute the below command.
  RZ/G1x > set pc 0xE6300000
- To load loader binary image into memory from SD Card, execute the below command.
  RZ/G1x > memory load card://loader.bin bin 0xE6300000
  RZ/G1x > go
  RZ/G1x > halt
- To set the supervisor mode interrupts, execute the below command.
  RZ/G1x > set cpsr 0x1D3
- To set the stack pointer for u-boot, execute the below command.
  RZ/G1x > set sp 0xE633FFFC
- To set the program counter for u-boot, execute the below command.
  - For RZ/G1H:
    RZ/G1x > set pc 0xE8080000
  - For RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C:
    RZ/G1x > set pc 0xE6304000
- To load loader.bin image into memory from SD Card, execute the below command.
  - For RZ/G1H:
    RZ/G1x > memory load card://u-boot.bin bin 0xE8080000
    RZ/G1x > go
  - For RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C:
    RZ/G1x > memory load card://u-boot.bin bin 0xE6304000
    RZ/G1x > go
- Now, target board will boot and boot prints will appear in the target board console.
- Stop in u-boot command prompt by pressing any key.
- Once command prompt appears, execute the below command in JTAG board to halt the target board core.
  RZ/G1x > *halt*

Step 5 set U-Boot environment variables

Please start U-Boot by board reset. Please set and save environment variable as follows. If U-Boot is replaced from BSP older version as YoctoRecipe Package v1.4.1, please set environment variables from default state after executing "env default -a" command. Current U-Boot has bootm_low and bootm_size which are new environment variables for device tree work area. It is necessary to reflect into environment save area of SPI Flash.

```
=> setenv ethaddr xx:xx:xx:xx:xx:xx
=> setenv ipaddr 192.168.0.20
=> setenv serverip 192.168.0.1
(to use zImage)
=> setenv bootcmd 'tftp 0x40007fc0 zImage;tftp 0x40f00000 uImage-r8a7742-
iwg21d-q7-dbcm-ca.dtb;bootz 0x40007fc0 - 0x40f00000'      (In case of
RZ/G1H)
=> setenv bootcmd 'tftp 0x40007fc0 zImage;tftp 0x40f00000 uImage-r8a7743-
iwg20d-q7-dbcm-ca.dtb;bootz 0x40007fc0 - 0x40f00000' (In case of RZ/G1M)
=> setenv bootcmd 'tftp 0x40007fc0 zImage;tftp 0x40f00000 uImage-r8a7744-
iwg20m.dtb;bootz 0x40007fc0 - 0x40f00000'      (In case of RZ/G1N)
=> setenv bootcmd 'tftp 0x40007fc0 zImage;tftp 0x40f00000 uImage-r8a7745-
iwg22d-sodimm.dtb;bootz 0x40007fc0 - 0x40f00000' (In case of RZ/G1E)
=> setenv bootcmd 'tftp 0x40007fc0 zImage;tftp 0x40f00000 uImage-r8a77470-
iwg23s-sbc.dtb;bootz 0x40007fc0 - 0x40f00000'    (In case of RZ/G1C)

(to use uImage)
=> setenv bootcmd 'tftp 0x40007fc0 uImage;tftp 0x40f00000 uImage-r8a7742-
iwg21d-q7-dbcm-ca.dtb;bootm 0x40007fc0 - 0x40f00000'      (In case of
RZ/G1H)
=> setenv bootcmd 'tftp 0x40007fc0 uImage;tftp 0x40f00000 uImage-r8a7743-
iwg20d-q7-dbcm-ca.dtb;bootm 0x40007fc0 - 0x40f00000'      (In case of
RZ/G1M)
=> setenv bootcmd 'tftp 0x40007fc0 uImage;tftp 0x40f00000 uImage-r8a7744-
iwg20m.dtb;bootm 0x40007fc0 - 0x40f00000'      (In case of RZ/G1N)
=> setenv bootcmd 'tftp 0x40007fc0 uImage;tftp 0x40f00000 uImage-r8a7745-
iwg22d-sodimm.dtb;bootm 0x40007fc0 - 0x40f00000' (In case of RZ/G1E)
=> setenv bootcmd 'tftp 0x40007fc0 uImage;tftp 0x40f00000 uImage-
r8a77470-iwg23s-sbc.dtb;bootm 0x40007fc0 - 0x40f00000'    (In case of
RZ/G1C)
```

Note) Common device tree file (dtb) is used for both zImage and uImage.

Step 6 write Kernel device tree by U-Boot

If you would like to load Kernel device tree file (uImage-r8a7742-iwg21d-q7-dbcm-ca.dtb, uImage-r8a7743-iwg20d-q7-dbcm-ca.dtb, uImage-r8a7744-iwg20m.dtb, uImage-r8a7745-iwg22d-sodimm.dtb, uImage-r8a77470-iwg23s-sbc.dtb) from SPI Flash, please prepare as follows. If you would like to download from Linux Host PC with tftp, you don't need following steps.

Please write Kernel device tree file to address 0x100000 of SPI Flash by U-Boot. Put the Kernel device tree file in root directory of TFTP server.

Due to Data specs of SPI Flash, Size arguments in the following instructions are set to 0x400000 regardless of the size of dtb file.

```
=> tftpboot 0x41000000 uImage-r8a7742-iwg21d-q7-dbcm-ca.dtb      (In case of
RZ/G1H)
=> tftpboot 0x41000000 uImage-r8a7743-iwg20d-q7-dbcm-ca.dtb   (In case of
RZ/G1M)
=> tftpboot 0x41000000 uImage-r8a7744-iwg20m.dtb      (In case of RZ/G1N)
=> tftpboot 0x41000000 uImage-r8a7745-iwg22d-sodimm.dtb      (In case of
RZ/G1E)
=> tftpboot 0x41000000 uImage-r8a77470-iwg23s-sbc.dtb      (In case of
RZ/G1C)

sh_eth Waiting for PHY auto negotiation to complete.. done
sh_eth: 100Base/Full
Using sh_eth device
TFTP from server 192.168.0.1; our IP address is 192.168.0.20
Filename 'uImage-r8a7743-iwg20m.dtb'.
Load address: 0x41000000
Loading: #
         357.4 KiB/s
done
Bytes transferred = 22334 (573e hex)
```

Step 7 change the bootargs by U-Boot

To change bootargs which passed to the kernel in boot sequence, please modify it by “setenv bootargs” command of u-boot.

```
=> setenv bootargs console=ttySC2,115200n8 vmalloc=384M ip=192.168.0.20
:::::eth0 root=/dev/nfs nfsroot=192.168.0.1:/export/rfs rootwait' (In case of
RZ/G1H)
=> setenv bootargs console=ttySC0,115200n8 vmalloc=384M ip=192.168.0.20
root=/dev/nfs nfsroot=192.168.0.1:/export/rfs rootwait' (In case of RZ/G1M,
RZ/G1N)
=> setenv bootargs 'console=ttySC3,115200n8 vmalloc=384M ip=192.168.0.20
root=/dev/nfs nfsroot=192.168.0.1:/export/rfs rootwait' (In case of RZ/G1E)
=> setenv bootargs 'console=ttySC1,115200n8 vmalloc=384M ip=192.168.0.20
root=/dev/nfs nfsroot=192.168.0.1:/export/rfs rootwait' (In case of RZ/G1C)
=> saveenv
```

Step 8 write Kernel (zImage/uImage) by U-Boot

If you would like to load Kernel (zImage/uImage) from SPI Flash, please prepare as follows. If you would like to download from Linux Host PC with tftp, you don't need following steps.

Please write Kernel (zImage/uImage) to address 0x140000 of SPI Flash by U-Boot. Put the uImage in root directory of TFTP server.

```
=> tftpboot 0x41000000 zImage(uImage)
=> setenv kern_size 0xXXXXXX      (filesize of zImage/uImage roundup by
256KB boundary, for example "0x400000")
=> sf probe
=> sf erase 0x140000 ${kern_size}
=> sf write 0x41000000 0x140000 ${kern_size}
```

Please set U-Boot environment variable as follows.

```
(to use zImage)
=> setenv bootcmd 'sf probe;sf read 0x40007fc0 0x140000 ${kern_size};sf
read 0x40f00000 0x100000 0x40000;bootz 0x40007fc0 - 0x40f00000'
(to use uImage)
=> setenv bootcmd 'sf probe;sf read 0x40007fc0 0x140000 ${kern_size};sf
read 0x40f00000 0x100000 0x40000;bootm 0x40007fc0 - 0x40f00000'

=> saveenv
```

Step 9 set file system

Please extract file system (core-image-weston-<supported board name>.tar.bz2). Please export /export directory of NFS server.

```
$ mkdir /export/rfs
$ cd /export/rfs
$ sudo tar xvf core-image-weston-<supported board name>.tar.bz2
```

Note) <supported board name> is the following: iwg21m (iWave RZ/G1H), iwg20m-g1m (iWave RZ/G1M), iwg20m-g1n (RZ/G1N), iwg22m (iWave RZ/G1E), iwg23s (iWave RZ/G1C).

Note) Please use proper file system images, iwg21m for RZ/G1H, iwg20m-g1m for RZ/G1M, iwg20m-g1n for RZ/G1N, iwg22m for RZ/G1E and iwg23s for RZ/G1C. And also please take attention to the package configurations which defined in Section 3.

Step 10 start Linux

After board reset, U-Boot is started. After countdown, Linux boot messages are displayed. Please confirm login prompt after Linux boot messages.

5. Exporting Toolchains

Please refer Documents from Yocto Project to export Toolchains such as
https://www.yoctoproject.org/docs/2.4.2/adt-manual/adt-manual.html

And please use build target of bitbake as "core-image-weston-sdk -c populate_sdk" to generate package.

Note) When you use "ld" directly but not via gcc (in case of building Kernel, Driver or u-boot), please disable LDFLAGS with 'export LDFLAGS="" '.

Note) Please do not use same shell environment to other compilation/debugging purpose (also make menuconfig of linux kernel, e.g.) but cross compilation for RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C which shell environment with "source" command to setup environment variables for the SDK. Because some environment variables for cross compilation interferes execution of other tools on the same shell environment.

Note) Options for Compiler and its corresponds are same for each CPU core architecture (CortexA15 for RZ/G1H, RZ/G1M, RZ/G1N, CortexA7 for RZ/G1H, RZ/G1E, RZ/G1C), however header files and libraries to be linked statically are selected and assembled into SDK Toolchain Package according to the configurations of which are using for building SDK Toolchain Package.

Example of instruction:

In following examples, it's assumed that it's already extracted and prepared recipe environment such as in the instructions of Section 3 (must done just before execution of bitbake, at least). You may reuse $WORK/build while you reuse same configuration after executing bitbake as in Section 3 for this purpose.

Step 1 configure architectures of Host PC which are installed this toolchain

When users of toolchain execute different architecture of Host PC other than build environment (x86_64 and i686), please modify SDK_MACHINE description on $WORK/build/conf/local.conf

On $WORK/build/conf/local.conf

```
# This variable specified the architecture to build SDK/ADT items for and means
# you can build the SDK packages for architectures other than the machine you are
# running the build on (i.e. building i686 packages on an x86_64 host._
# Supported values are i686 and x86_64
#SDKMACHINE ?= "i686"
SDKMACHINE ?= "i686" (or " x86_64")
# ← When toolchain used in 32bit Host PC but building toolchain with x86_64
```

Step 2 building toolchain package with bitbake

```
$ cd $WORK/build
$ bitbake core-image-weston-sdk -c populate_sdk
$ cp tmp/deploy/sdk/poky-glibc-x86_64(or i686)-core-image-weston-sdk-
cortexa15hf-(vfp-)neon-toolchain- 2.4.2.sh <shared dir. where able to
access from each Host PCs>    (for RZ/G1H, RZ/G1M, RZ/G1N)

$ cp tmp/deploy/sdk/poky-glibc-x86_64(or i686)-core-image-weston-sdk-
cortexa7hf-(vfp-)neon-toolchain- 2.4.2.sh <shared dir. where able to access
from each Host PCs>    (for RZ/G1E, RZ/G1C)
```

Step 3 Install toolchain on each Host PCs

```
(For RZ/G1H, RZ/G1M, RZ/G1N)
$ sudo <shared dir. where able to access from each Host PCs>/ poky-glibc-
x86_64(or i686)-core-image-weston-sdk-cortexa15hf-(vfp-)neon-toolchain-
2.4.2.sh
(For RZ/G1E, RZ/G1C)
$ sudo <shared dir. where able to access from each Host PCs>/ poky-glibc-
x86_64(or i686)-core-image-weston-sdk-cortexa7hf-(vfp-)neon-toolchain-
2.4.2.sh
[sudo] password for (INSTALL person): (password of your account)
Enter target directory for SDK (default: /opt/poky/2.4.2): (just a return)
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

Step 4 setup environment variables for each compilation on each Host PCs

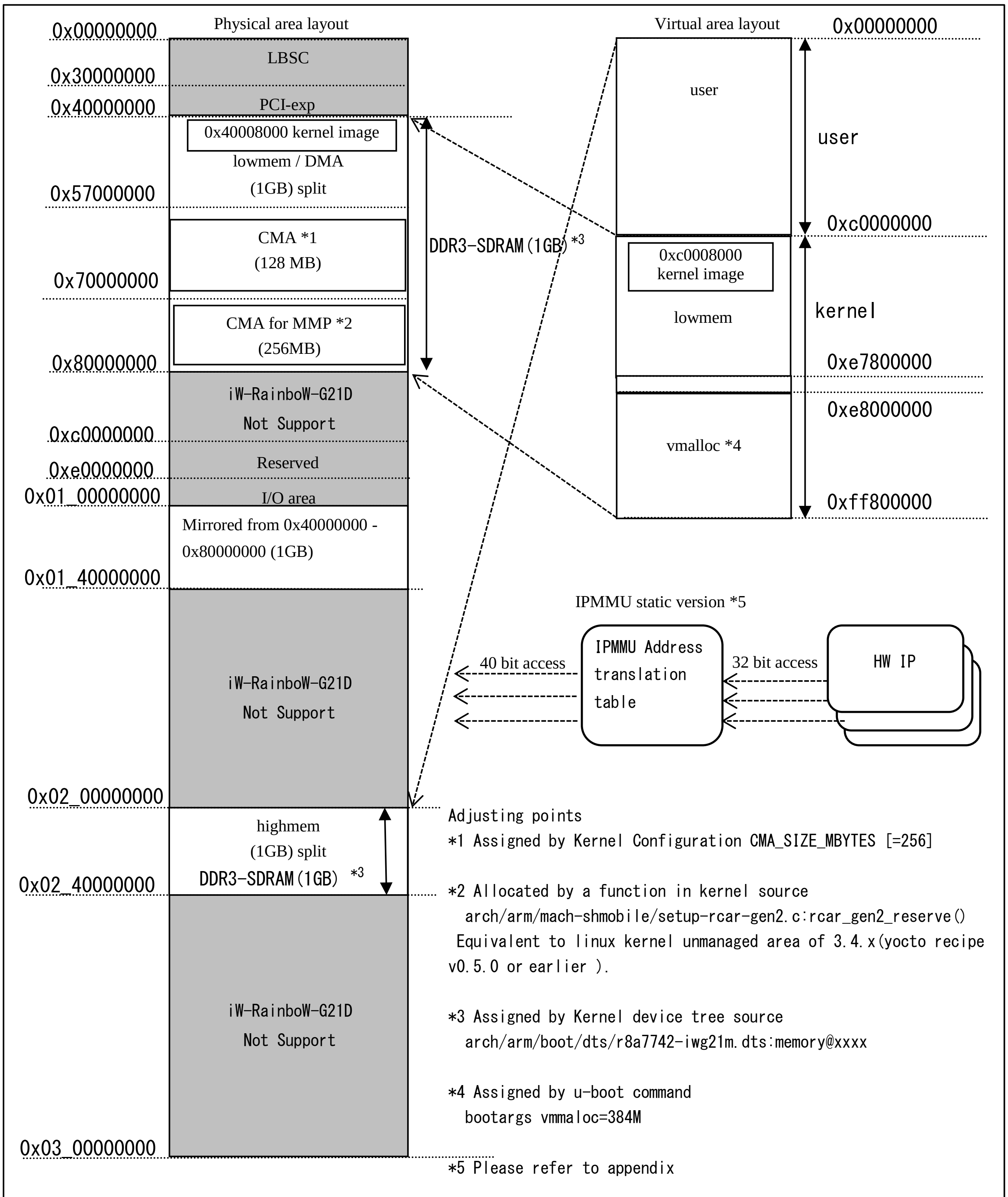
Please setup environment variables as follows or integrate set-up sequence into your build script or Makefile.

```
$ cd (Your working directory)
$ source /opt/poky/2.4.2/environment-setup-cortexa15hf(-vfp)-)neon-
poky-linux-gnueabi    (For RZ/G1H, RZ/G1M, RZ/G1N)
$ source /opt/poky/2.4.2/environment-setup-cortexa7hf(-vfp)-neon-poky-
linux-gnueabi    (For RZ/G1E, RZ/G1C)
$ export LDFLAGS=""
$ $CC (Your source code).c <optional FLAG>
```

6. Memory map

Following Figure 3, Figure 4, Figure 5, Figure 6, Figure 7 shows memory map of this RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux BSP package.

A saving area of the environment variable of U-Boot is address 0xc0000 of SPI Flash. If you would like to return default value, please use "env default -a" command, after that save by the saveenv command.



Adjusting points

*1 Assigned by Kernel Configuration CMA_SIZE_MBYTES [=256]

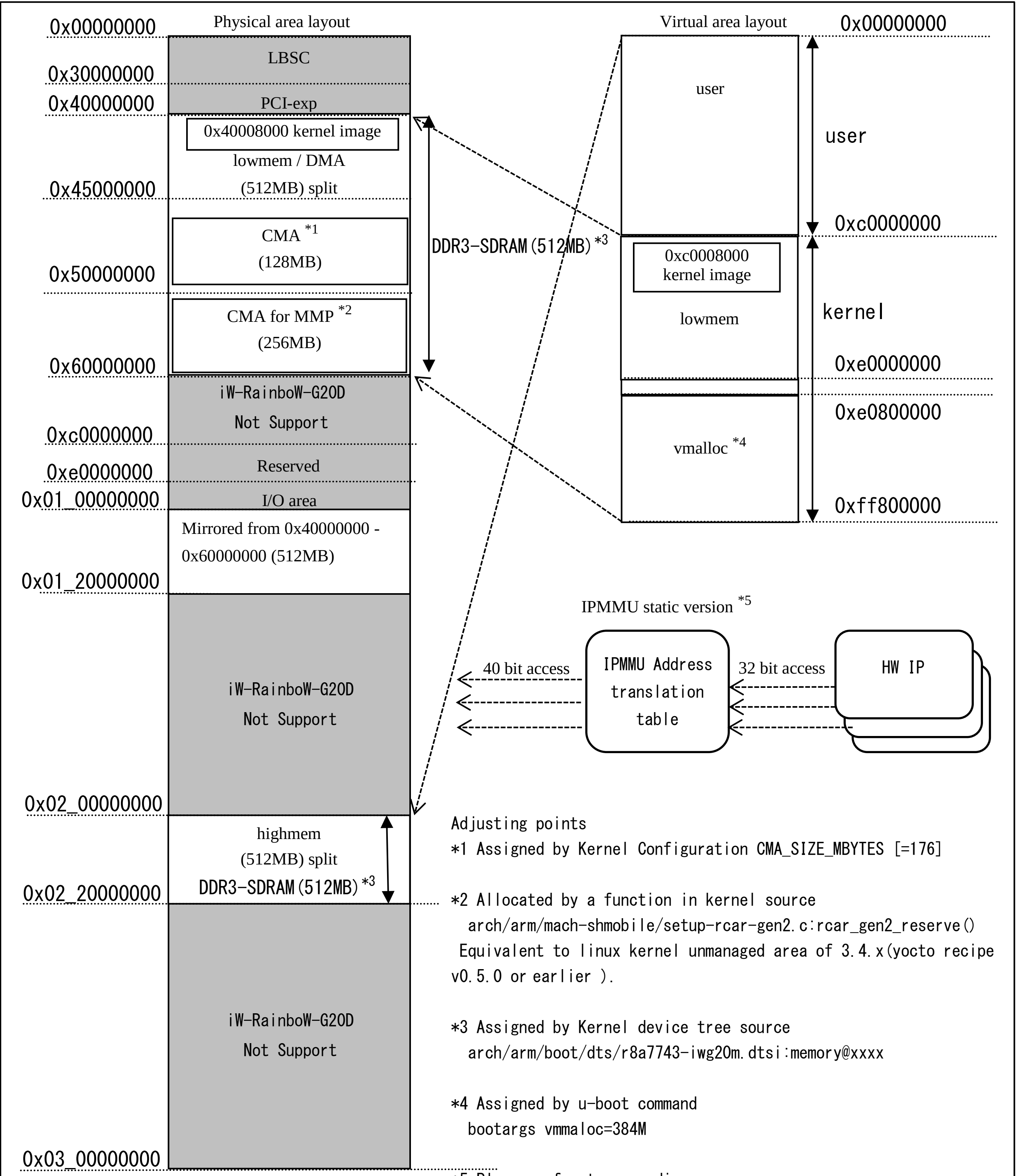
*2 Allocated by a function in kernel source
arch/arm/mach-shmobile/setup-rcar-gen2.c:rcar_gen2_reserve()
Equivalent to linux kernel unmanaged area of 3.4.x(yocto recipe v0.5.0 or earlier).

*3 Assigned by Kernel device tree source
arch/arm/boot/dts/r8a7742-iwg21m.dts:memory@xxxx

*4 Assigned by u-boot command
bootargs vmmaloc=384M

*5 Please refer to appendix

Figure 3: RZ/G1H memory map (Linux)



Adjusting points

*1 Assigned by Kernel Configuration CMA_SIZE_MBYTES [=176]

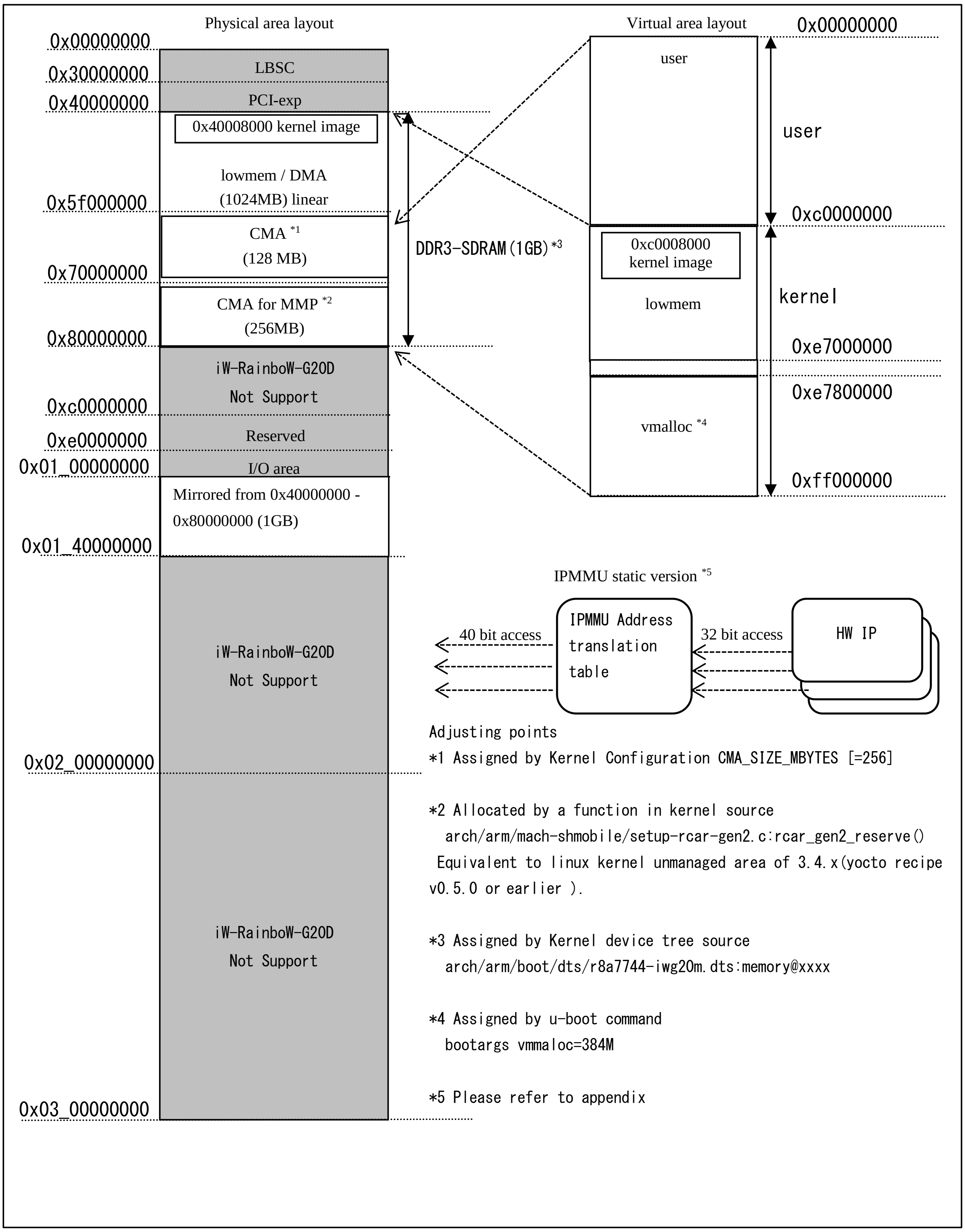
*2 Allocated by a function in kernel source
  arch/arm/mach-shmobile/setup-rcar-gen2.c:rcar_gen2_reserve()
 Equivalent to linux kernel unmanaged area of 3.4.x(yocto recipe v0.5.0 or earlier).

*3 Assigned by Kernel device tree source
  arch/arm/boot/dts/r8a7743-iwg20m.dtsi:memory@xxxx

*4 Assigned by u-boot command
  bootargs vmmaloc=384M

*5 Please refer to appendix

Figure 4: RZ/G1M memory map (Linux)



Adjusting points
*1 Assigned by Kernel Configuration CMA_SIZE_MBYTES [=256]

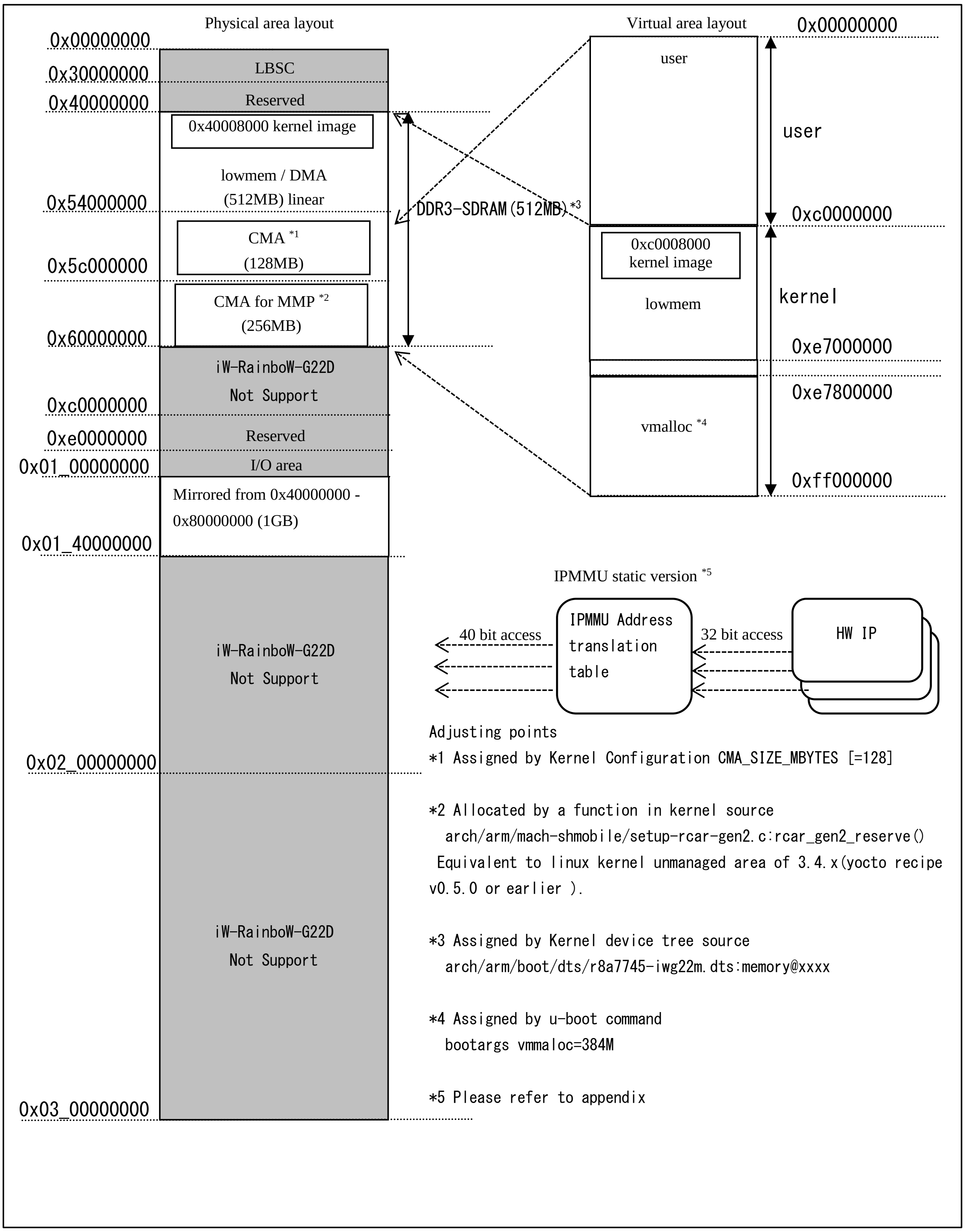
*2 Allocated by a function in kernel source
  arch/arm/mach-shmobile/setup-rcar-gen2.c:rcar_gen2_reserve()
 Equivalent to linux kernel unmanaged area of 3.4.x(yocto recipe v0.5.0 or earlier).

*3 Assigned by Kernel device tree source
  arch/arm/boot/dts/r8a7744-iwg20m.dts:memory@xxxx

*4 Assigned by u-boot command
  bootargs vmmaloc=384M

*5 Please refer to appendix

Figure 5: RZ/G1N memory map (Linux)



Adjusting points

*1 Assigned by Kernel Configuration CMA_SIZE_MBYTES [=128]

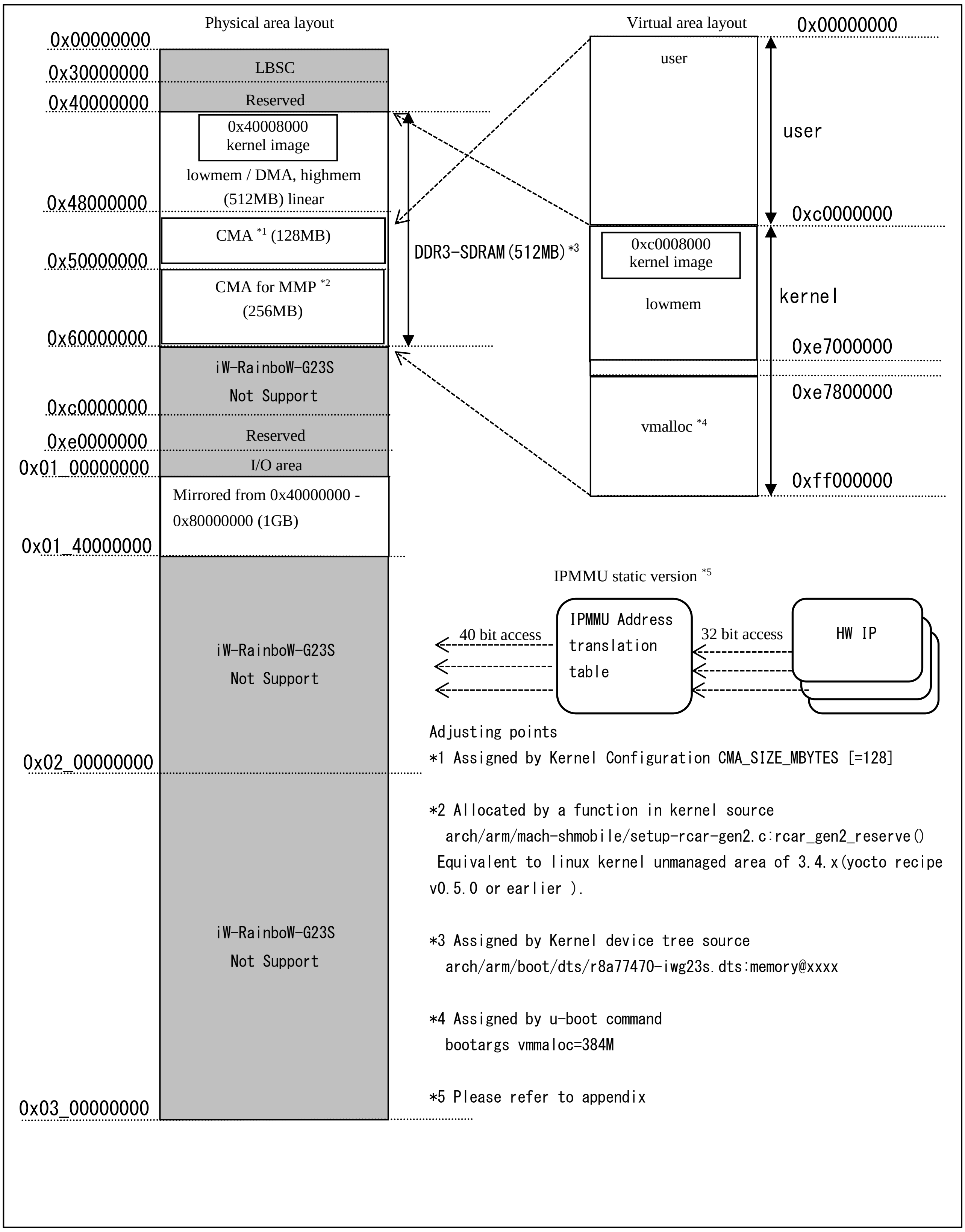
*2 Allocated by a function in kernel source
arch/arm/mach-shmobile/setup-rcar-gen2.c:rcar_gen2_reserve()
Equivalent to linux kernel unmanaged area of 3.4.x(yocto recipe v0.5.0 or earlier).

*3 Assigned by Kernel device tree source
arch/arm/boot/dts/r8a7745-iwg22m.dts:memory@xxxx

*4 Assigned by u-boot command
bootargs vmmaloc=384M

*5 Please refer to appendix

Figure 6: RZ/G1E memory map (Linux)



Adjusting points

*1 Assigned by Kernel Configuration CMA_SIZE_MBYTES [=128]

*2 Allocated by a function in kernel source
arch/arm/mach-shmobile/setup-rcar-gen2.c:rcar_gen2_reserve()
Equivalent to linux kernel unmanaged area of 3.4.x(yocto recipe v0.5.0 or earlier).

*3 Assigned by Kernel device tree source
arch/arm/boot/dts/r8a77470-iwg23s.dts:memory@xxxx

*4 Assigned by u-boot command
bootargs vmmaloc=384M

*5 Please refer to appendix

Figure 7: RZ/G1C memory map (Linux)

7. How to write binary file by TFTP

How to write some binary data is described in this section. The binary data is downloaded from TFTP server. It assumes that U-Boot setting is done by section 4.

Note) Please use Mini-Monitor for writing U-Boot. Please refer to section 4 step4.

Procedure 1

Refer to the following figure and set the network system.

IP address in this figure is an example. Please use suitable IP address.

The TFTP server must have been started in advance.

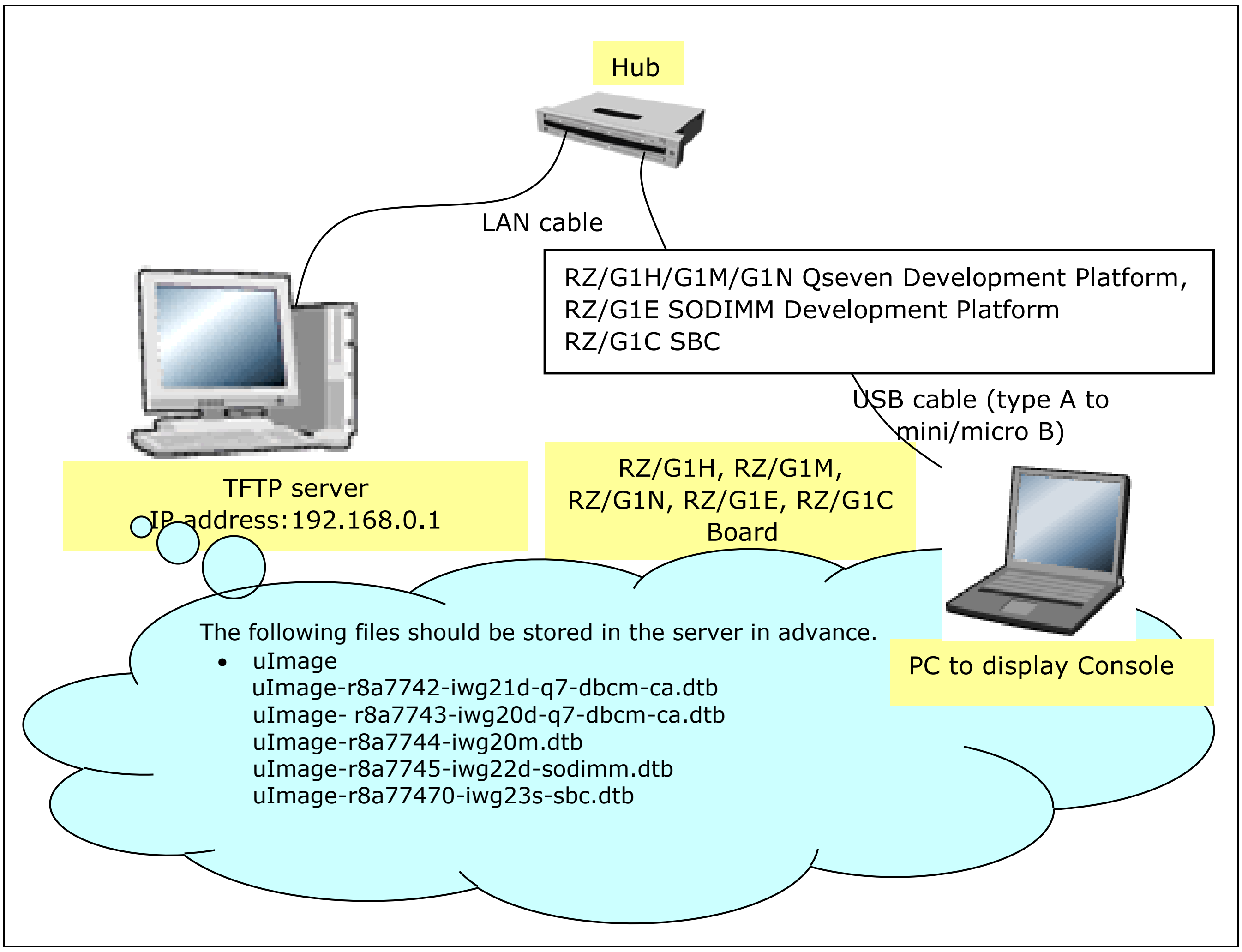

Hub
LAN cable
RZ/G1H/G1M/G1N Qseven Development Platform,
RZ/G1E SODIMM Development Platform
RZ/G1C SBC
USB cable (type A to mini/micro B)
TFTP server
IP address:192.168.0.1
RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Board
PC to display Console
The following files should be stored in the server in advance.
• uImage
uImage-r8a7742-iwg21d-q7-dbcm-ca.dtb
uImage- r8a7743-iwg20d-q7-dbcm-ca.dtb
uImage-r8a7744-iwg20m.dtb
uImage-r8a7745-iwg22d-sodimm.dtb
uImage-r8a77470-iwg23s-sbc.dtb

Figure 8 Download environment from TFTP server

The following files should be stored in the server in advance.

- uImage (Kernel)

The directory to store is the setting directory (for example "/tftpboot") of TFTP server.

Procedure 2

"Hit any key to stop autoboot" is displayed, after starting RZ/G1H, RZ/G1M, RZ/G1N Qseven Development Platform, RZ/G1E SODIMM Development Platform, RZ/G1C SBC. And the countdown of three seconds starts. Please hit enter key once while this. Then console appears.

```
U-Boot 2013.01.01-xxxxxxxx (xxx xx xxxx - xx:xx:xx)

CPU: Renesas Electronics R8A7743 rev 3.0
CPU: Temperature 70 C
Board: RZ/G1M iW-RainboW-G20M-Q7

DRAM:   1 GiB
MMC:    sh-sdhi: 0, sh-sdhi: 1, sh_mmcif: 2
SF: Detected SST25VF016B with page size 4 KiB, total 2 MiB
In:     serial
Out:    serial
Err:    serial

Board Info:
        SOM Version        : iW-PREWZ-AP-01-R3.3

Net:    ether_avb
Hit any key to stop autoboot:  0
=>
```

Procedure 3

Download a binary file from TFTP server to SDRAM. After that, write the binary file to SPI Flash.

The following example shows that the binary file uImage is downloaded to empty area address 0x41000000 on SDRAM and written by size 0x300000 (that is bigger than downloaded size of uImage) from address 0x140000 of SPI Flash. Settable size of "sf erase" command is multiple of the sector unit (256Kbyte) of SPI Flash.

Please confirm with the memory map about the address of the empty area of RAM and the write-in address of SPI Flash.

```
=> tftpboot 0x41000000 uImage
sh_eth:1 is connected to sh_eth.  Reconnecting to sh_eth
sh_eth Waiting for PHY auto negotiation to complete. done
sh_eth: 100Base/Full
Using sh_eth device
TFTP from server 192.168.0.1; our IP address is 192.168.0.20
Filename 'uImage'.
Load address: 0x41000000
Loading: #############################################
         #############################################
         #############################################
         #############################################
           ....
           ....
         #############################################
         2.8 MiB/s
done
Bytes transferred = 3094000 (2f35f0 hex)

=> setenv kern_size 0xXXXXXX     (set filesize of uImage, round up by
256KB boundary, for example "0x400000"
=> sf probe
SF: Detected S25FL512S with page size 256 KiB, total 64 MiB
=> sf erase 0x140000 ${kern_size}
=> sf write 0x41000000 0x140000 ${kern_size}
=>
```

Note) After writing is completed, if an Enter key is inputted the writing will start again.

8. U-Boot command

Please refer to U-Boot user's manual about available U-boot command for RZ/G1H, RZ/G1M, RZ/G1N, RZ/G1E, RZ/G1C Linux BSP. The help or "?" command shows U-Boot command list, but be careful that it includes some unsupported command.

9. Appendix

9.1 IPMMU Example

The following example illustrates how to handle static mapped PTE for IPMMU.

```
enum {
    IPMMUSY0_DOMAIN = 0,
    IPMMUSY1_DOMAIN,
    IPMMUDS_DOMAIN,
    IPMMUMP_DOMAIN,
    IPMMUMX_DOMAIN,
    IPMMURT_DOMAIN,
    IPMMUGP_DOMAIN,
    IPMMU_DOMAIN_MAX
};

#define IPMMUSY0_ADDR   (0xe6280000 + 0x800)
#define IPMMUSY1_ADDR   (0xe6290000 + 0x800)
#define IPMMUDS_ADDR    (0xe6740000 + 0x800)
#define IPMMUMP_ADDR    (0xec680000 + 0x800)
#define IPMMUMX_ADDR    (0xfe951000 + 0x800)
#define IPMMURT_ADDR    (0xffc80000 + 0x800)
#define IPMMUGP_ADDR    (0xe62a0000 + 0x800)

#define IMCTR       0x0000
#define IMCAAR      0x0004
#define IMTTBCR     0x0008
#define IMBUSCR     0x000c
#define IMTTLBR0    0x0010
#define IMTTUBR0    0x0014
#define IMTTLBR1    0x0018
#define IMTTUBR1    0x001c
#define IMSTR       0x0020
#define IMMAIR0     0x0028
#define IMMAIR1     0x002c
#define IMUCTR      0x0300
#define IMUASID     0x0308

#define IPMMU_PGDVAL(phys_addr) ((phys_addr & PGDIR_MASK) | 0x721)

#ifdef MMNGR_KOELSCH

#define IPMMU_ADDR_SECTION_0    0x0100000000ULL
#define IPMMU_ADDR_SECTION_1    0x0200000000ULL
#define IPMMU_ADDR_SECTION_2    0x0ULL
#define IPMMU_ADDR_SECTION_3    0x0ULL
#define IPMMU_PGDVAL_SECTION_0  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_0)
#define IPMMU_PGDVAL_SECTION_1  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_1)
#define IPMMU_PGDVAL_SECTION_2  0x0ULL
#define IPMMU_PGDVAL_SECTION_3  0x0ULL

#else

#define IPMMU_ADDR_SECTION_0    0x0100000000ULL
#define IPMMU_ADDR_SECTION_1    0x0140000000ULL
#define IPMMU_ADDR_SECTION_2    0x0180000000ULL
#define IPMMU_ADDR_SECTION_3    0x01c0000000ULL
#define IPMMU_PGDVAL_SECTION_0  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_0)
#define IPMMU_PGDVAL_SECTION_1  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_1)
#define IPMMU_PGDVAL_SECTION_2  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_2)
#define IPMMU_PGDVAL_SECTION_3  IPMMU_PGDVAL(IPMMU_ADDR_SECTION_3)

#endif

#define IPMMU_KERNEL_PHY_ADDR       0x0040000000ULL
#define IPMMU_KERNEL_PHY_MIRROR_ADDR    0x0100000000ULL
#define IPMMU_KERNEL_LEGACY_MEM_SIZE    0x0040000000ULL
```

```
static unsigned long r8a779x_ipmmu_reg_base[IPMMU_DOMAIN_MAX] = {
    IPMMUSY0_ADDR,
    IPMMUSY1_ADDR,
    IPMMUDS_ADDR,
    IPMMUMP_ADDR,
    IPMMUMX_ADDR,
    IPMMURT_ADDR,
    IPMMUGP_ADDR,
};

static phys_addr_t r8a779x_ipmmu_trans_table[IPMMU_DOMAIN_MAX][4] = {
/* IPMMUSY0 */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMUSY1 */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMUDS */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMUMP */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMUMX */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMURT */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
/* IPMMUGP */
{ IPMMU_ADDR_SECTION_0, IPMMU_ADDR_SECTION_1, IPMMU_ADDR_SECTION_2, IPMMU_ADDR_SECTION_3 },
};

static pgdval_t* r8a779x_ipmmu_pgd[IPMMU_DOMAIN_MAX];

static void r8a779x_ipmmu_startup( void )
{
    int i;
    pgdval_t *pgdval_addr = NULL;

    for ( i = 0; i < IPMMU_DOMAIN_MAX; i++ ){
        pgdval_addr = kmalloc(PAGE_SIZE, GFP_KERNEL);
        if ( pgdval_addr != NULL ){
            pgdval_addr[0] = IPMMU_PGDVAL_SECTION_0;
            pgdval_addr[1] = IPMMU_PGDVAL_SECTION_1;
            pgdval_addr[2] = IPMMU_PGDVAL_SECTION_2;
            pgdval_addr[3] = IPMMU_PGDVAL_SECTION_3;
            r8a779x_ipmmu_pgd[i] = pgdval_addr;
        }
    }
}

static void r8a779x_ipmmu_cleanup( void )
{
    int i;

    for ( i = 0; i < IPMMU_DOMAIN_MAX; i++ ){
        if ( r8a779x_ipmmu_pgd[i] != NULL ){
            kfree(r8a779x_ipmmu_pgd[i]);
        }
    }
}

static int r8a779x_ipmmu_initialize( int domain )
{
    void __iomem *base;
    int i, utlb_num;

    if ( domain < 0 || domain >= IPMMU_DOMAIN_MAX ){
        return -1;
    }

    if ( r8a779x_ipmmu_pgd[domain] == NULL ){
        return -1;
    }

    pr_debug( "ipmmu initialize. domain:%d¥n", domain );
    base = ioremap( r8a779x_ipmmu_reg_base[domain], PAGE_SIZE );

    iowrite32( 0x80000000, base + IMTTBCR );
    iowrite32( 0x00000000, base + IMTTUBR0 );
    iowrite32( __pa(r8a779x_ipmmu_pgd[domain]), base + IMTTLBR0 );
    iowrite32( 0x00000003, base + IMCTR );

    utlb_num = ( domain != IPMMUGP_DOMAIN ) ? 32: 1;

    for ( i = 0; i < utlb_num; i++ ){
        iowrite32( 0x00000003, base + IMUCTR + (0x10*i) );
    }

    pr_debug("ipmmu : IMTTBCR: %x, IMTTUBR0: %x, IMTTLBR0: %x, IMCTR: %x ¥n",
        ioread32(base + IMTTBCR), ioread32(base + IMTTUBR0),
        ioread32(base + IMTTLBR0), ioread32(base + IMCTR));

    iounmap( base );

    return 0;
}

static unsigned long r8a779x_ipmmu_phys_to_virt( int domain, phys_addr_t paddr )
{
    unsigned long vaddr = 0;
    int section = 0;

    if ( domain < 0 || domain >= IPMMU_DOMAIN_MAX ){
        return 0;
    }

    if ( paddr >= IPMMU_KERNEL_PHY_ADDR &&
         paddr < IPMMU_KERNEL_PHY_ADDR + IPMMU_KERNEL_LEGACY_MEM_SIZE ) {
        /* If a physical address indicates the legacy space,
         * adjust the offset for the IPMMU table conversion.
         */
        paddr = paddr - IPMMU_KERNEL_PHY_ADDR + IPMMU_KERNEL_PHY_MIRROR_ADDR;
    }

    for ( section = 0; section < 4; section++ ){
        if ( (paddr >= r8a779x_ipmmu_trans_table[domain][section]) &&
             (paddr < r8a779x_ipmmu_trans_table[domain][section]+SZ_1G ) ){
            vaddr = section * SZ_1G;
            vaddr |= paddr & 0x3fffffff;
        }
    }

    return vaddr;
}

static phys_addr_t r8a779x_ipmmu_virt_to_phys( int domain, unsigned long vaddr )
{
    phys_addr_t paddr = 0;
    int section = 0;

    if ( domain < 0 || domain >= IPMMU_DOMAIN_MAX ){
        return 0;
    }

    section = (vaddr & 0xc0000000) >> 30;
    paddr = vaddr & 0x3fffffff;
    paddr |= r8a779x_ipmmu_trans_table[domain][section];

    if ( paddr >= IPMMU_KERNEL_PHY_MIRROR_ADDR &&
         paddr < IPMMU_KERNEL_PHY_MIRROR_ADDR + IPMMU_KERNEL_LEGACY_MEM_SIZE ) {
        /* If a physical address indicates the legacy space,
         * adjust the offset for the IPMMU table conversion.
         */
        paddr = paddr - IPMMU_KERNEL_PHY_MIRROR_ADDR + IPMMU_KERNEL_PHY_ADDR;
    }

    return paddr;
}
```

REVISION HISTORY	Linux Interface Specification Yocto recipe Start-Up Guide User’s Manual: Software

Rev.	Date	Description	
		Page	Summary
0.10	Mar. 08, 2015	—	First Edition issued (Preliminary)
1.00	Apr. 01, 2016	—	Update version
1.10	Jan. 11, 2017	—	Added RZ/G1H
2.00	Sep. 15, 2017	—	Update kernel version to linux-4.4.y-cip for RZ/G1M
2.01	Nov. 02, 2017	7, 13, 14, 15, 22, 24, 25	Remove command or description related to x11: we have no support for x11
		8, 9	Fix build command
2.02	Nov. 22, 2017	9,14,25	Fix build command
		11	Remove command or description related to x11
2.03	Dec. 15, 2017	—	Update kernel version to linux-4.4.y-cip for RZ/G1E
2.04	Feb. 15, 2018	—	Update kernel version to linux-4.4.y-cip for RZ/G1C
2.05	Mar. 30, 2018	—	Update kernel version to linux-4.4.y-cip for RZ/G1N
2.06	May. 31, 2018	—	Update kernel version to linux-4.4.y-cip for RZ/G1H
2.07	Jul. 31, 2018	—	Add support VLP2.1.x
2.08	Sep. 30, 2018	—	Add support VLP2.1.x-RT Remove VLP2.0.x related information Separate iWave-G1M and iWave-G1N Update ‘certified-linux-v2.1.0-update1’
2.09	Mar. 27, 2019	2, 9, 10	Change descriptions of ‘certified-linux’ version
2.10	Dec. 23, 2019	15, 16, 18, 20, 21	Update build instruction for Gecko, Qt, and no UI framework Update device-tree names of RZ/G1C and RZ/G1E
2.11	Mar. 13, 2020	10	Update build instruction for demo packages
2.12	Dec. 31, 2020	—	Update device-tree names of RZ/G1H

Linux Interface Specification Yocto recipe Start-Up Guide
User’s Manual: Software

Publication Date: Rev2.12 Dec 31, 2020

Published by: Renesas Electronics Corporation

Linux Interface Specification
Yocto recipe Start-Up Guide

Renesas Electronics Corporation