RZ/A2M

DRP Linux Application Development

R01ANxxxxEx0100
Rev.1.00
Oct 23, 2019

Introduction

This document explains application development using the DRP and the RZ/A2M Linux BSP.

Target Device

RZ/A2M

Contents

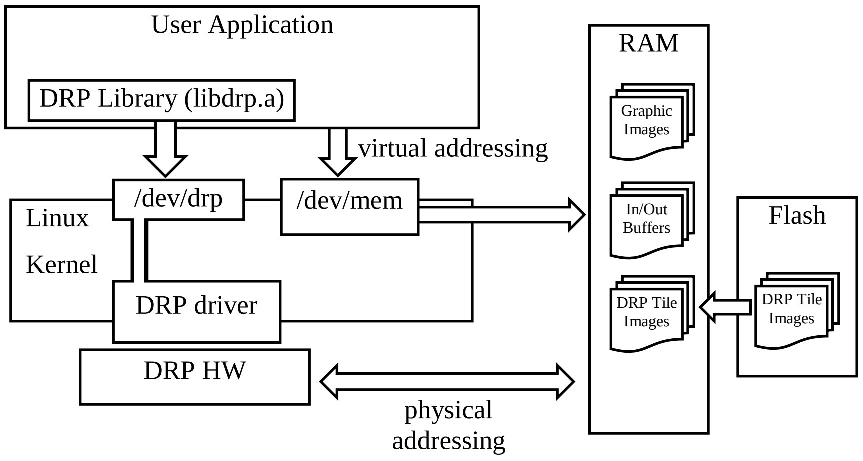
1. Overview

The Dynamically Reconfigurable Processor (DRP) hardware is included in the RZ/A2M.

This document does not describe what DRP is, it only describes the driver and API to the software provided by Renesas.

Pre-built DRP libraries are provided by Renesas along with a simple software API for using them.

Below is a block diagram showing the general architecture of the system.



Software Architecture

2. API Documentation

The software interface (API) for Linux driver is the same as the RTOS version. Therefore, the RTOS version documentation is included in this package under the directory:

rza2m_drp/drp-lib/doc/

There are 4 documents: (2 in English, 2 in Japanese). Those documents are described below:

DRP Driver User's Manual
rza2m_drp/drp-lib/doc/r01us0355ej0100-rza2.pdf
This manual explains the software API to the DRP driver. Even though it refers to FreeRTOS in the document, the API and definitions will be the same for Linux.

DRP Library User's Manual
rza2m_drp/drp-lib/doc/r01us0367ej0100-rza2.pdf
This manual explains in details the paramters that are to be passed for each DRP Library provided by Renesas.

In addition to the API functions in those documents, there are some additional functions for the Linux version.

3. DRP Package Installation

3.1 RZ/A Linux-4.19 BSP Requirement

This software requires that the RZ/A Linux-4.19 BSP is already installed.

Please follow the instructions here to install that BSP:

https://elinux.org/RZ-A/Boards/RZA2MEVB

Also follow the instructions to do the default BSP build.

3.2 Downloading DRP Package

Once the 4.19 BSP has been configured and built for RZ/A2M, please enter the following commands in order to download the DRP source code from github and please it under the 4.19 BSP directory.

```
$ cd rza_linux-4.19_bsp
$ git clone https://github.com/renesas-rz/rza2m_drp
```

3.3 Installing the driver into the kernel

Instead of copying all the DRP files under the kernel source tree, we will make a symbolic link to point to our driver directory in rza2m_drp package. This way we can keep the driver within the DRP package directory, but make it part of the kernel build source in another directory.

Make sure you are at the base of the 4.19 BSP directory.

```
$ cd rza_linux-4.19_bsp
```

Create a symbolic link under the 'drivers' directory in the kernel:

```
$ cd output/linux-4.19/drivers
$ ln -s ../../../rza2m_drp/driver/drp
$ cd ..                             # back to the kernel directory
```

Now we will patch the kernel to add our driver (add to menuconfig)

```
$ patch -p1 -i ../../rza2m_drp/kernel_patch/*.patch
```

Finally, we will enable the driver in the kernel's menuconfig:

```
$ cd ../..      # back to root of BSP
```

```
$ ./build.sh kernel menuconfig
```

Use the keyboard sequence below to search and enable the DRP driver in the kernel.

- PRESS the '/' key on the keyboard (open search window)
- Type "DRP", press Enter
- Press '1'
- Press 'y'
- Press ESC + ESC
- Press ESC + ESC
- Press ESC + ESC
- Press Enter (select "YES" to save new configuration)

Now rebuild the kernel

```
$ ./build.sh kernel xipImage
```

Program the new kernel

(RESET your board into u-boot)

```
  $ ./build.sh jlink xipImage 0x20200000
```

3.4 Device Tree with DRP Enabled

Even though the driver has been added to the kernel, we need to also enable it in the Device Tree. That can be done by adding the following code to your board's dts file:

```
drp: drp@ea000000 {
  compatible = "renesas,r7s9210-drp";
  reg = <0xea000000 0x1000000>;
  interrupts = <GIC_SPI 415 IRQ_TYPE_LEVEL_HIGH>;
  clock = <&cpg CPG_MOD 90>;
  power-domains = <&cpg>;
  status = "okay";

  /* Physical address range of config data */
  /* You can also register these in your application */
  //config-area-phys = <0x23C00000>;/* Physical address of DRP config data */
  //config-area-size = <(4*1024*1024)>; /* Size of DRP config data */
};
```

When you are using the RZ/A2M EVB, the downloaded DRP package contains a Device Tree that already has the DRP driver enabled. The following steps are needed to build and program a Device Tree that will enable the DRP driver for the RZ/A2M EVB:

```
$ cd rza_linux-4.19_bsp
$ ./build.sh env
$ export ROOTDIR=$(pwd) ; source ./setup_env.sh
$ cd rza2m_drp
$ cd devicetree
$ make
```

Now program the device tree into your board.

Connect your **J-Link** to your board and **RESET** your board into u-boot.

```
$ make program
```

4. DRP User API

4.1 Building the DRP User API library

Pre-built DRP libraries are provided by Renesas along with a simple software API for using them.

The DRP Linux library will allow you to call the standard Renesas DRP driver API calls (designed for RTOS systems) in your Linux application.

For the DRP library images, you will select only the ones you require before you build the API library that will be used by the application. Also, the DRP library images will be bundled together into a single image that you will program separately into flash.

By default, the config_flash_create.sh is already configured for the DRP demo that is included in this package. Below are the steps for build that library:

```
$ cd rza_linux-4.19_bsp
$ ./build.sh env
$ export ROOTDIR=$(pwd) ; source ./setup_env.sh

(rza_bsp)$ cd rza2m_drp
(rza_bsp)$ cd drp-lib
```

First you will need to create the binary (drplib-config.bin) with all the DRP images you need. A header file (drplib-config-api.h) will also be created that your application will use to reference the data inside the binary.

```
(rza_bsp)$ ./config_flash_create.sh
```

4.2 Build the API library that will be used by the User Application

A libdrp.a library will be created for your user application to statically link against. This is the recommended method. Also, a libdrp.so runtime library is created that can be copied to the target file system for your user application to dynamically link against at runtime. By default, the demo application's Makefile will statically link against the libdrp.a library, so copying the libdrp.so file is not required to run the demo.

```
(rza_bsp)$ make
```

To program the binary image that contains all the DRP images (drplib-config.bin) that was created by ‘config_flash_create.sh’ script into QSPI flash, do the following. Note that this will program the image into the last 4MB of QSPI flash area.

(Connect your J-Link to your board)

(RESET your board into u-boot)

```
(rza_bsp)$ ./config_flash_program.sh
```

5. Selecting/Customizing the DRP Library

For the DRP Library (libdrp.a), you select what DRP images you plan to use in your application by editing the file "drp-lib/config_flash_create.sh" instead of including all the DRP images that are released by Renesas.

Simply comment out the DRP images you do not not need by placing a "#" symbol at the beginning of the line.

```
#CONFIG1=drp_lib/r_drp_argb2grayscale/r_drp_argb2grayscale.dat
CONFIG2=drp_lib/r_drp_bayer2grayscale/r_drp_bayer2grayscale.dat
#CONFIG3=drp_lib/r_drp_binarization_adaptive_bit/r_drp_binarization_adaptive_bit.dat
#CONFIG4=drp_lib/r_drp_binarization_adaptive/r_drp_binarization_adaptive.dat
CONFIG5=drp_lib/r_drp_binarization_fixed/r_drp_binarization_fixed.dat
```

Also in the config_flash_create.sh, you can specify what Flash address you would like to store this image

```
# Address in QSPI Flash (last 4MB of 64MB QSPI Flash)
DRP_FLASH_ADDR=0x23C00000
```

After you have made your modifications to config_flash_create.sh, please:

1. Execute the **config_flash_create.sh** script again to generate a new drplib-config.bin and drplib-config-api.h file
2. Create a new libdrp.a library by executing '**make**'
3. Program the new drplib-config.bin file into flash using the **config_flash_program.sh** script.
4. Rebuild your application using the updated libdpr.a library (because drplib-config-api.h will have changed).

6. Sample Application

6.1 Building the Sample Application

This package comes with a sample application located in the "DRP_sample" directory . To build the application, please execute the following commands.

Set up the build environment:

```
  $ cd rza_linux-4.19_bsp
  $ ./build.sh env
  $ export ROOTDIR=$(pwd) ; source ./setup_env.sh
```

Build the application:

```
  (rza_bsp)$ cd rza2m_drp
  (rza_bsp)$ cd DRP_sample
  (rza_bsp)$ make
```

Add the application to your root file system (located in the buildroot directory):

```
  (rza_bsp)$ make install
```

Rebuild the root file system:

```
  (rza_bsp)$ cd ../..
  (rza_bsp)$ ./build.sh buildroot
```

Reprogram the root file system into the QSPI flash:

(Connect your J-Link to your board)

(RESET your board into u-boot)

```
  (rza_bsp)$ ./build.sh jlink rootfs_axfs 0x20800000
```

6.2 Running the Sample Application

The application was designed to be used with an RSK (800x480) LCD.

Boot your board (=> run xha_boot) and log into the kernel as root.

To run the demo, enter:

```
$ /root/drp/DRP_sample -mipi
```

6.3 About the Sample Application

Below are some comments about the sample application.

- The application was designed to be used with an RSK (800x480) LCD.
- It is recommended to use the 'Raspberry Pi' MIPI camera that comes with the RZ/A2M EVB kit. The application can work with an OV7670 parallel camera, but addition hardware configurations are needed.
- There are 2 VDC6 'layers' used. One layer is configured as an 8-bit Color Lookup Table (CLUT8) and shows black and white images from the camera. The other layer is configured as a 4-bit Color Lookup Table (CLUT4) and is an 'overlay' of the camera image in order to visually show the results of the DRP processing. The CLUT8 and CLUT4 formats were chosen in order to use as little RAM as possible.
- The 4MByte internal RAM was used for the MIPI camera image capture, the DRP buffers and the LCD frame buffers. The 8MB HyperRAM was used for the "Linux" system (kernel and application space). You can see that all the internal RAM buffers used in the application were hardcoded at the top of the file main.c in the section label "Memory Map".
- By default, the DRP images are copied from flash into RAM (Hyper RAM) in order to make switching between DRP images faster. However, this uses more system RAM. If the command line option " -qo " is used, you can see that the amount of time to switch between DRP images dynamically increases.

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Oct 28, 2019	—	First edition issued