RZ/V2L OpenCV Accelerator Sample Application Note Revision.1.10

OpenCV Accelerator Sample Application Note

Introduction

This document describes the OpenCV Accelerator Sample Application for RZ/V2L.

Target Device

RZ/V2L

Note

This document uses the difference Linux command execution environment.

Each environment will be differentiated by the following notation.

1. Linux PC environment

```
$ <Linux PC Command>
```

2. RZ/V2L Evaluation Board Kit environment

```
# <RZ/V2L Evaluation Board Kit Command>
```

Contents

1. Overview

This document assumes that users have read the RZ/V2L OpenCV Accelerator Support Package Release Note and already executed the instructions for Boot environment and Compile environment.

This document will explain about the Sample Application which uses OpenCV Accelerator (referred to below as “OpenCVA”) on the RZ/V2L.

This sample application is to execute the following OpenCV functions in each case with the DRP disabled and enabled.

Note: If DRP is enabled, OpenCVA uses DRP to execute OpenCV function. Otherwise, OpenCV function is executed by CPU. For detail, see RZ/V2L OpenCV Accelerator User's Manual (R11UZ01346).

Case	OpenCV function	Description of process
1	cv::resize ()	Image size conversion src: FHD (BGR format) dst: XGA (BGR format)
2	cv::cvtColor ()	Color space conversion (YUV to RGB) src: FHD (YUYV format) dst: FHD (BGR format)
3	cv::cvtColorTwoPlane()	Color space conversion (NV to RGB) src: FHD (NV21 format) dst: FHD (BGR format)
4	cv::GaussianBlur()	Gaussian blur effect (kernel size = 7x7) src, dst: FHD (BGR format)
5	cv::dilate()	Image dilation (200 iterations) src, dst: FHD (BGR format)
6	cv::erode()	Image erosion (100 iterations) src, dst: FHD (BGR format)
7	cv::morphologyEx()	Image opening (50 iterations) src, dst: FHD (BGR format)
8	cv::filter2D()	Image convolution (kernel size = 3x3) src, dst: FHD (BGR format)
9	cv::Sobel()	Edge detection with Sobel filter (kernel size = 3x3) src, dst: FHD (BGR format)
10	cv::adaptiveThreshold()	Adaptive binarization of image (block size = 99x99) src, dst: FHD (Grayscale format)
11	cv::mathcTemplate()	Find parts of the src image that are similar to the small template image src: 640x360 (BGR format) template: 16x16 (BGR format)
12	cv:: warpAffine()	An affine transformation(rotate 45 degree) src, dst: FHD (BGR format)
13	cv::warpPerspective()	A perspective transformation src, dst: FHD (BGR format)
14	cv::pyrDown()	Blurs and downsamples src: FHD (BGR format) dst: QFHD (BGR format)
15	cv::pyrUp()	Blurs and upsamples src: QFHD (BGR format) dst: FHD (BGR format)

1.1 Operating Environment

Table 1-1 shows the operating environment for the sample applications.

Table 1-1 Operating Environment

Items	Version	Notes
RZ/V2 Linux Package	VLP 3.0.5	For details, please refer to the RZ/V2 Linux Package Release Note.
RZ MPU Graphics Library	1.2.0	For details, please refer to the RZ/G2L & RZ/V2L SoC Groups GPU Integration Guide.
DRP-AI Support Package	7.41	For details, please refer to the RZ/V2L DRP-AI Support Package Release Note.
OpenCV Accelerator Support Package	1.10	For details, please refer to RZ/V2L OpenCV Accelerator Support Package Release Note and RZ/V2L OpenCV Accelerator User's Manual

1.2 Sample application

OpenCVA sample application execute the OpenCV functions in sequence at OpenCV Accelerator support package installed environment. The source image data is FHD size image "image.png", that is read and converted to size and format before the OpenCV function is executed.

Each OpenCV function is disabled or enabled to use DRP by the *OCA_activate()* function. This sample application is to execute the each OpenCV features in each case with the DRP disabled and enabled by the function.

1.3 Related Documents

1. RZ/V Verified Linux Package Release Note (R01US0565)
2. RZ/V Verified Linux Package Start-Up Guide for RZ/V2L (R01US0617)
3. RZ/V2L DRP-AI Support Package Release Note (R11AN0549)
4. RZ/V2L OpenCV Accelerator Support Package Release Note (R11AN0845)
5. RZ/V2L OpenCV Accelerator User's Manual (R11UZ0346)

1.4 Memory Map

RZ/V2L Linux Package and RZ/V2L OpenCV Accelerator Support Package uses the memory map shown in the Figure 1.1.

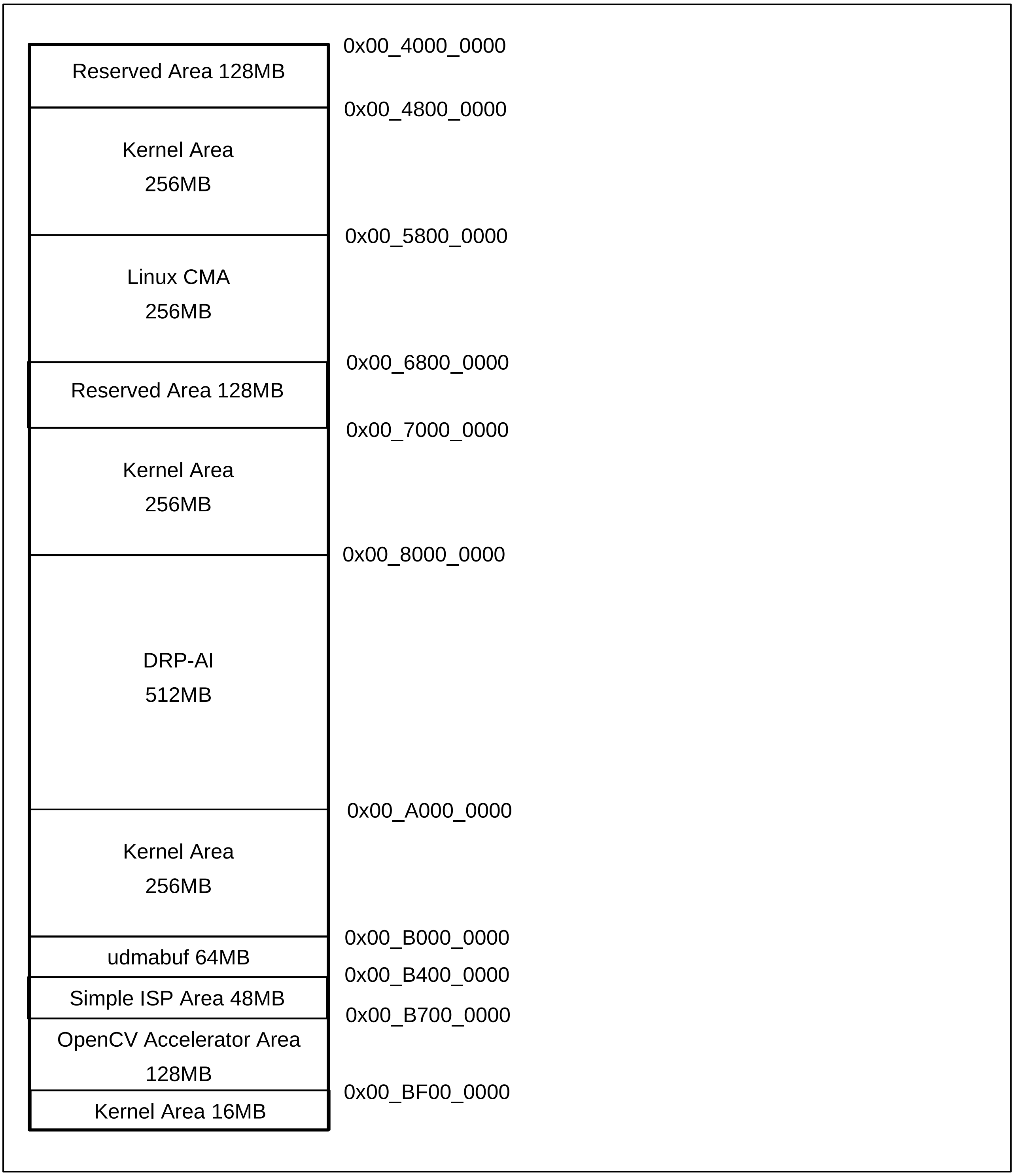


Figure 1.1 Memory map

2. Compile OpenCVA Sample Application

This chapter will explain how to compile the RZ/V2L OpenCVA Sample Application.

2.1 Software for Compiling

Table 2-1 shows the necessary software for compiling.

Table 2-1 Necessary software for Compiling

No	Items	Filename	Details
1	SDK	-	Generated from using RZ/V2L Linux Package and OpenCV Accelerator Support Package.
2	OpenCVA Sample Application	OpenCV_sample.zip	OpenCVA Sample Application source code, executables, and related files

2.2 SDK

SDK (Software Development Kit) is a development environment that will allow users to cross-compile the C/C++ source code for RZ/V2L Linux (ARM64).

To generate and install the SDK, please refer to the RZ/V2L OpenCV Accelerator Support Package Release Note.

2.3 Directory structure of OpenCVA Sample Application

This package provides OpenCVA Sample Applications.

The directory structure of OpenCVA is the following.

Table 2-2 Directory Structure Example of OpenCVA Sample Application

Items	Directory name	Example of stored file	Usage
Source Code	src	Makefile *.cpp *.h	A set of source code of RZ/V2L OpenCVA Sample Application. Compilation method will be explained in this chapter.
Execution Environment	exe	pre-compiled application image.png	An Execution environment for RZ/V2L OpenCVA Sample Application. This will include all necessary files to run the application. RZ/V2L OpenCVA Sample Application Note will explain how to deploy it on the RZ/V2L Evaluation Board Kit.

2.4 Setup the Working Directory

Create the working directory for compiling the OpenCVA Sample Application.

1. Set the path of working directory as an environment variable.

   Note: Change the path of working directory <WORK> accordingly.

```
$ export APP_WORK=~/<WORK>
```

2. Execute the following command to create the working directory.

```
$ mkdir $APP_WORK
```

2.5 Extract the Source Code

Use the following command to extract the OpenCVA Sample Application Package.

Notes :<PATH_to_SRC> is a path of OpenCV_sample.zip

```
$ cd <PATH_to_SRC>
$ unzip -d $APP_WORK OpenCV_sample.zip
```

After extracted the package, the working directory have the following structure.

```
$APP_WORK
    OpenCV_sample
        exe
            OCA_sample
            image.png
        src
            define.h
            Makefile
            OCA_sample.cpp
```

Figure 2-1 Working Directory Structure

2.6 Set SDK Environment Variable

In order to cross-compile the application, the environment variables need to be changed for SDK.

Execute following command to set the SDK environment variable.

Note: 1. The environment variable will be reset if the Linux terminal is closed.

2. Following command assumes the SDK is installed under the path "/opt/poky/x.x.x".

```
$ source /opt/poky/x.x.x/environment-setup-aarch64-poky-linux
```

Note: “x.x.x” is replaced by SDK version.

2.7 Cross-Compile

After the environment variable is set, execute following commands to compile the source code.

```
$ cd $APP_WORK/OpenCV_sample/src
$ make
```

Command "make" will execute the commands stated in Makefile.

For the details of compilation command, see the Makefile.

After the make, the binary file will be generated in the application source code directory.

```
$APP_WORK
  app_opencv
    exe
      OCA_sample
      image.png
    src
      define.h
      Makefile
      OCA_sample.cpp
      OCA_sample          Generated binary file
```

Figure 2-2 Directory Structure of app_opencva

3. Setup the Execution Environment for OpenCVA Sample Application

This chapter will explain how to deploy the execution environment for OpenCVA Sample Application.

Copy "OCA_sample" and "image.png" files to the root filesystem of RZ/V2L Linux.

Please use the previously compiled application or the pre-compiled sample application.

Directory structure of root filesystem will be as follows. (`Applications` are copied to `/home/root/exe`).

Note: when using SD card, please use the "sync" command after copying the file.

```
usr
bin
lib
…
home
    root
        exe
            OCA_sample
            image.png
```

4. Execute Sample Application

This section assumes the RZ/V2L Evaluation Kit is successfully booted and will explain how to execute RZ/V2L OpenCVA Sample Application.

1. Move to the directory where the application is stored.

Note: <PATH_to_FILE> is a path to the directory that the application is stored.

For the example in Chapter 3, this would be "/home/root/exe".

```
# cd <PATH_to_FILE>
```

2. Run the following command to execute the application.

```
# ./OCA_sample
```

Note: If the application is copied to the execution environment from the Ubuntu PC, the file **permission may not allow** the application to be run.

In this case, please run the following command on RZ/V2L Evaluation Board to allow the file execution.

```
# chmod +x OCA_sample
```

3. OpenCVA Sample Application execution results.

OpenCVA sample application execute the OpenCV functions in sequence.

The sample application executes 15 OpenCV processes in sequence with DRP disabled and enabled. And displays their processing times to console, and output the results (PNG images) to storage.

An example of the information displayed in the console is as follows.

```
[6] erode                FHD(BGR) [iteration=100]
[CPU]1206.385864msec
[OCA]360.674133msec
[CPU] / [OCA] = 3.344808 times
```

OpenCV Function

[CPU] Processing time without DRP

[OCA] Processing time with DRP

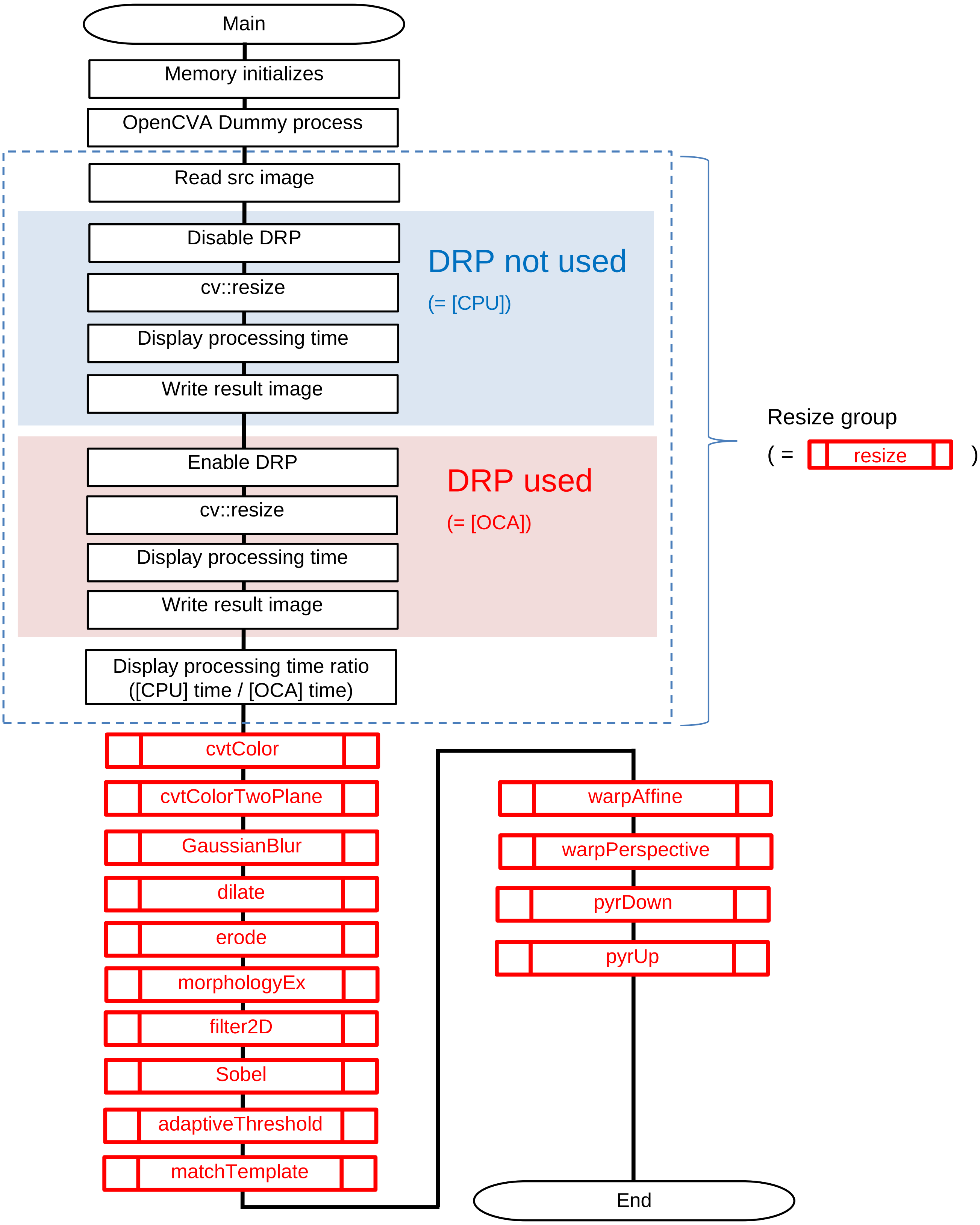
The 15 OpenCV functions are as follows.

Case	OpenCV function	OpenCV Parameters	Output file name (DRP not used)	Output file name (DRP used)
1	cv::resize	src: (1080,1920, CV_8UC3) dst: (768,1024, CV_8UC3) dsize: {1024,768} fx: 0 fy: 0 interpolation: cv::INTER_LINEAR	OCA1_cpu_out.png	OCA1_oca_out.png
2	cv::cvtColor	src: (1080,1920, CV_8UC2) dst: (1080,1920, CV_8UC3) code: cv::COLOR_YUV2BGR_YUYV	OCA2_cpu_out.png	OCA2_oca_out.png
3	cv::cvtColorTwoPlane	src1: (1080,1920, CV_8UC1) src2: (540,960, CV_8UC2) dst: (1080,1920, CV_8UC3) code: cv::COLOR_YUV2BGR_NV21	OCA3_cpu_out.png	OCA3_oca_out.png
4	cv::GaussianBlur	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) ksize: {7,7} sigmaX: 0 sigmaY: 0	OCA4_cpu_out.png	OCA4_oca_out.png
5	cv::dilate	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) kernel: cv::Mat() anchor: cv::Point(-1,-1) iterations: 200	OCA5_cpu_out.png	OCA5_oca_out.png
6	cv::erode	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) kernel: cv::Mat() anchor: cv::Point(-1,-1) iterations: 100	OCA6_cpu_out.png	OCA6_oca_out.png
7	cv::morphologyEx	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) op: cv::MORPH_OPEN kernel: cv::Mat() anchor: cv::Point(-1,-1) iterations: 50	OCA7_cpu_out.png	OCA7_oca_out.png
8	cv::filter2D	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) ddepth: -1 kernel: (3,3, CV_32FC1)	OCA8_cpu_out.png	OCA8_oca_out.png
9	cv::Sobel	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) ddepth: -1 xorder: 1 yorder: 0	OCA9_cpu_out.png	OCA9_oca_out.png

10	cv::adaptiveThreshold	src: (1080,1920, CV_8UC1) dst: (1080,1920, CV_8UC1) maxValue: 0xFF adaptiveMethod: cv::ADAPTIVE_THRESH_MEAN_C thresholdType: cv::THRESH_BINARY blockSize: 99 C: 0	OCA10_cpu_out.png	OCA10_oca_out.png
11	cv::matchTemplate	src: (360,640, CV_8UC3) template: (16,16, CV_8UC3) dst: (345,625, CV_32FC1) mode: cv::TM_SQDIFF	OCA11_cpu_out.png	OCA11_oca_out.png
12	cv::warpAffine	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) M: {0.7071, -0.7071, 649, 0.7071, 0.7071,510} dsize:{1920, 1080} flags: INTER_LINEAR borderMode: BORDER_CONSTANT borderValue: Scalar()	OCA12_cpu_out.png	OCA12_oca_out.png
13	cv::warpPerspective	src: (1080,1920, CV_8UC3) dst: (1080,1920, CV_8UC3) M: {0.5, 0.2, 20, -0.1, 0.8, 50, -0.001, 0.001, 1.0} dsize:{1920, 1080} flags: INTER_LINEAR borderMode: BORDER_CONSTANT borderValue: Scalar()	OCA13_cpu_out.png	OCA13_oca_out.png
14	cv::pyrDown	src: (1080,1920, CV_8UC3) dst: (540,960, CV_8UC3) dstsize: Size() borderType: BORDER_DEFAULT	OCA14_cpu_out.png	OCA14_oca_out.png
15	cv::pyrUp	src: (540,960, CV_8UC3) dst: (1080,1920, CV_8UC3) dstsize: Size() borderType: BORDER_DEFAULT	OCA15_cpu_out.png	OCA15_oca_out.png

5. Application Contents

This chapter described application contents.



Revision History

Rev.	Date	Description	
		Page	Summary
1.00	18 Jan, 2024	-	Issued.
1.10	5 Feb, 2024	5	Changed the version name of each file.
		6	Added simple ISP area to memory map.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.