RZ/A2M Group
DRP Library User's Manual and Functional Design Specifications
First Edition (Rev. 1.00)

FindContoursCrop

September 24, 2019		
Renesas Electronics		
Approved by	Examined by	Author

User's Manual

RZ/A2M Group

DRP Custom Library User's Manual

FindContoursCrop

Renesas Electronics
www.renesas.com

Rev.1.00 September. 2019

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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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.

How to Use This Manual

1. Purpose and Target Readers

This manual is intended to provide the user with an understanding of the functions of the DRP library and how to utilize them. It is aimed at users designing application systems making use of the DRP library. In order to use this manual, you will need a basic knowledge of programming languages and microprocessors.

> Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section.

> The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

Contents

1. Introduction

1.1 Summary

This manual describes the functions and usage of the DRP library, which run on the dynamically reconfigurable processor (DRP) of RZ/A2M Group Microprocessors.

The DRP can perform various functions according to user's setting. In this document, the function performed by DRP is called "circuit", and the data representing circuit information is called "configuration data". Writing of the circuit to DRP can be performed by loading the configuration data using DRP Driver[*1]. DRP Library is a collection of configuration data with various functions, mainly image processing.

Note 1. For details of DRP Driver, refer to "RZ/A2M Group DRP Driver User's Manual (R01US0355)".

1.2 Functions

The functions of the configuration data contained in the DRP library are listed below.

Table 1.1 DRP Library Functions

Category	Function Name	Outline	Page
Feature Detection	FindContoursCrop	Detects contours in the image and calculates its bounding rectangle within a specified crop window. The shapes and boxes can be output directly into the video ram by a debugging interface. Various filter operations allow the limiting of the created data.	11

2. Operation Conditions

The DRP library operates under the conditions listed below.

Table 2.1 Operation Conditions

Item	Description
Microprocessor	RZ/A2M Group Microprocessors*1 • R7S921051VCBG • R7S921052VCBG • R7S921053VCBG

Note 1. The DRP library operates on RZ/A2M Group Microprocessors equipped with a DRP function module. It will not operate on RZ/A2M Group Microprocessors without a DRP function module.

This library was confirmed to operate in the following development environment:

Renesas e^2 studio 7.5.0

The following toolchain is compatible:

GCC ARM Embedded Toolchain 6-2017-q2-update

3. File Structure

Figure 3.1 shows the file structure of configuration data and header files in the DRP library.

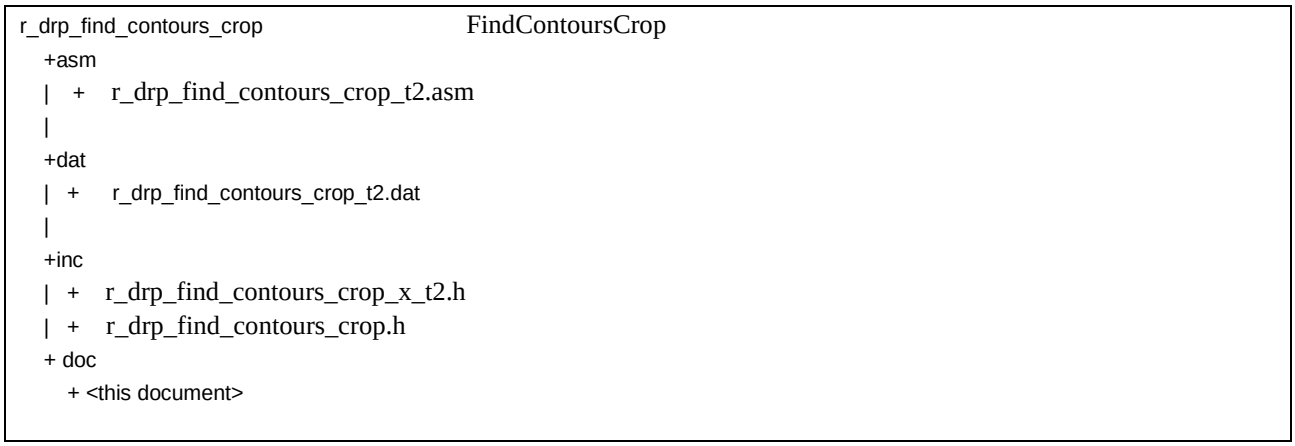


Figure 3.1 File Structure

4. DRP Library Reference

4.1 How to Read the DRP Library Reference

In this section the specifications of the configuration data contained in the DRP library are presented in the format shown below.

Function name*1

Function outline

Configuration data file	The name of the configuration data file. Use the DRP Driver's R_DK2_Load() function to load the data in the DRP.
Supported version	Lists the version of the configuration data that operates under present specification. Use the DRP Driver's R_DK2_GetInfo() function to get the version.
Configuration data size (byte)	Lists the size of the configuration data. Lists all versions, if there are different versions.
Header file	The name of the header file for using the configuration data. Use #include "header file" to include the file.
Parameter	Lists the parameters required by the circuit. Parameters are passed from the CPU to the DRP by means of the DRP driver's R_DK2_Start() function. Parameters are defined as a structure within the header file. Before running the circuit, set the parameters on the CPU side. The data type defined in stdint.h is used. Also, the area where parameters are stored and the area indicated by parameters representing addresses such as 'src' and 'dst' must be located in physical memory. *2
I/O details	Lists the details of the data specified by the parameters. Unless otherwise indicated, the same address may be specified for the input buffer address and output buffer address.
Number of tiles	The number of tiles used by the circuit. The DRP has 6 tiles. The DRP Driver's R_DK2_Load() function is used to assign circuits to tiles.
Segmented processing	Indicates that the function can be processed in parallel by multiple circuits. In parallel processing, the input image is divided up in the vertical direction and processed accordingly. The segmented processing can be executed by utilizing the 6 tiles of DRP and loading multiple configuration data of 3 tiles or less. For details on loading multiple configuration data of 3 tiles or less into DRP, see the explanation of R_DK2_Load () function in "RZ/A2M Group DRP Driver User's Manual".

Example: A case where the input image is divided into three portions in the vertical direction

Description	Describes the specifications of the configuration data.
Note	Additional notes appear here.

Note 1. The function name of configuration data is a character string that can be obtained from the configuration data by using the DRP Driver's R_DK2_GetInfo() function.

Note 2. If the values of physical memory in the area of parameters and input/output data of the circuit are incorrect because the values are in the Cortex-A9 cache, etc., the circuit does not work properly. It must be necessary to clean the cache before calling the DRP driver's R_DK2_Start() function or to allocate the parameters and input/output data of circuit to a non-cached area.

For information on using the API functions of the DRP Driver, refer to "RZ/A2M Group DRP Driver User's Manual (R01US0355)".

4.2 Feature Detection

4.2.1 FindContoursCrop

FindContoursCrop

Detects contours in the image and calculates its bounding rectangle within a specified crop window. The shapes and boxes can be output directly into the video ram by a debugging interface. Various filter operations allow the limiting of the created data.

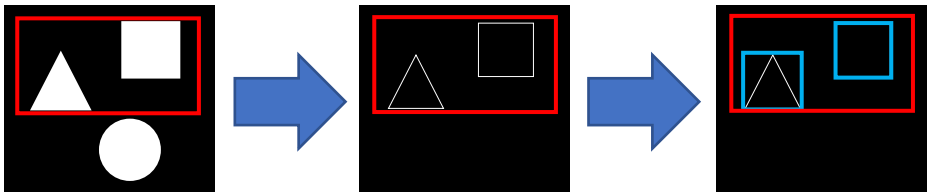
Configuration data file			r_drp_find_contours_crop_t2.dat	
Supported version			0.90	
Configuration data size (byte)			250336	
Header file			r_drp_find_contours_crop_x_t2.h (r_drp_find_contours_crop.h)	
Parameter	Structure name			
	r_drp_find_contours_t			
	Member name		Type	Description
	src		uint32_t	Input image address
	dst_rect		uint32_t	Output data (rectangle information) address
	dst_region		uint32_t	Output data (region information) address
	width		uint16_t	Input image width (pixels)
	height		uint16_t	Input image height (pixels)
	work		uint32_t	Work area address
	dst_rect_size		uint32_t	Maximum output number of Rectangle Information (0 to 20,000) (When set to 0, no output)
	dst_region_size		uint32_t	Maximum output number of Region Information (0 to 500,000) Specify the upper limit of the total of region information to be output, not the number per contour (When set to 0, no output)
	threshold_widthOrMin		uint16_t	Width threshold of rectangle to be detected (1 to width)
	threshold_heightOrMax		uint16_t	Height threshold of rectangle to be detected (1 to height)
	threshold_region		uint16_t	Output filter min contour points
	x_crop		uint16_t	Cropping window x-position
	y_crop		uint16_t	Cropping window y-position
	width_crop		uint16_t	Cropping window width
	height_crop		uint16_t	Cropping window height
	mode		uint8_t	Output mode and filter control
	dst_rgbDebug		uint32_t	Output data (debug drawing specified by dmode) address
	dmode		uint8_t	Debug mode output control
I/O details	Input image	Address: Width (pixels): Height (pixels): Format: Data size:	Specified by src. Specified by width. (16 to 1280, integer multiple of 8) Specified by height. (8 to 960) Binary image (1 byte per pixel), or 8-bits grayscale (1 byte per pixel) Refer to the description for details. (width) × (height) × 1 byte	
Crop window	Position: pixel exact (no multiple of 8 pixels required)	x_crop	x_crop+width_crop <= width	
		y_crop	y_crop+height_crop <= height	
		width_crop	5 <= width_crop <= width	
		height_crop	5 <= height_crop <=height	

	Output data (Rectangle Information)	Address:	Specified by dst_rect (Specify an address that differs from src.)
		Format:	From the top address, specifications are made in the following order. Upper-left corner x coordinate of Bounding rectangle (2 bytes) Upper-left corner y coordinate of Bounding rectangle (2 bytes) Bounding rectangle width (2 bytes) Bounding rectangle height (2 bytes) Count of region information (4 bytes) Start address of region information (4 bytes) Refer to the description for details.
		End Data:	End of rectangle information. All fields are 0 (16 bytes) Refer to the description for details.
		Data size:	(Count of rectangles detected + 1) × 16 bytes “+ 1” means the size of End Data The maximum size is (dst_rect_size) × 16 bytes
	Output data (Region Information)	Address:	Specified by dst_region (Specify an address that differs from src.)
		Format:	From the top address, specifications are made in the following order. x coordinate of one pixel constituting the contour (2 bytes) y coordinate of one pixel constituting the contour (2 bytes) Refer to the description for details.
		Data size:	(The total of pixels constituting every contour) × 4 bytes The maximum size is (dst_region_size) × 4 bytes
	Work area	Address:	Specified by work.
		Data size:	((width) × (height) × 1 / 4) + 8 byte
		Description	The area used to store data during FindContoursCrop processing.
Number of tiles	2		
Segmented processing	Not supported.		

Description

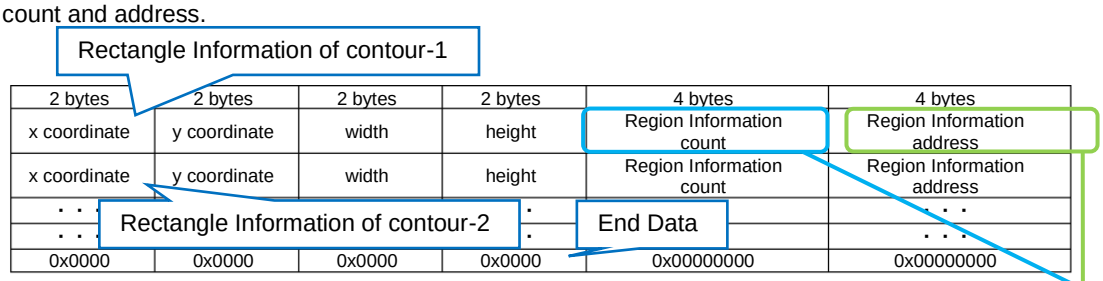
This function performs FindFontoursCrop processing of the image at the address specified by src and outputs the bounding rectangle information of detected contours to the address specified by dst_rect, and the pixel coordinate constituting detected contours to the address specified by dst_region.

The red box in the left image (input image) specifies the crop window in which the FindFontoursCrop will be processed. FindContoursCrop detects two contours within the crop window (middle image). The right image shows the found contours and the surrounding boxes(blue).

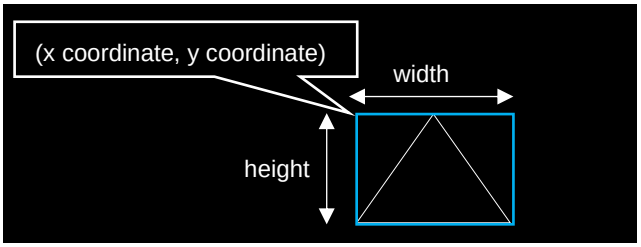


A simplified representation of detection of rectangle in the input image

“Rectangle Information”
This function outputs the bounding rectangle information of detected contours in input image and the corresponding Region Information address and count as shown below. After outputs this data set for the number of detected contours, this function outputs the End Data means end of data.
Keep reading the Rectangle Information until you find the End Data to obtain all the Rectangle Information. Also, you can obtain the corresponding Region Information using the Region Information count and address.

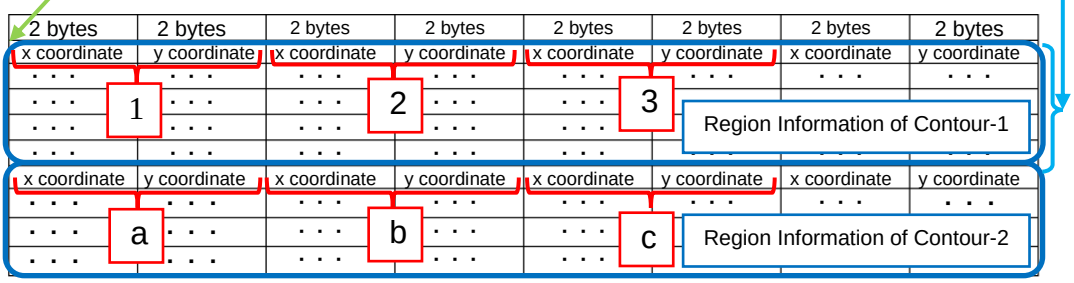


Rectangle Information

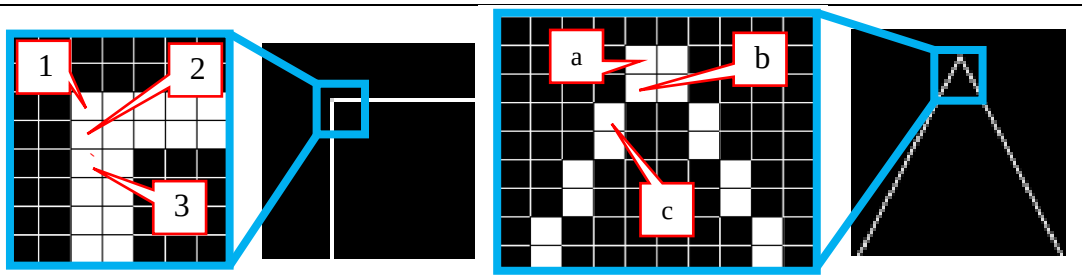


Description of Rectangle Information

“Region Information”
This function outputs (x, y) coordinate of all pixels constituting contours every contour as shown below. This coordinate is expressed in the coordinate system has upper-left corner pixel is (0, 0) in input image. If dst_region_size==0 then the “region information address” will be 0, the “region information count” always shows the number of region elements.



Region Information



Contour-1 Contour-2

If output data count reaches the value set in dst_rect_size or dst_region_size, the data output of the one that reached the upper limit stops, but the other data output doesn't stop. Moreover, if Both data count reaches upper limit, both data output stop, then DRP terminates.
If the Rectangle Information output count reaches upper limit before output the End Data, The End Data is not output.

This function supposes binary image as input image format.
When input 8-bit grayscale, this function treats pixels with a pixel value of 1 or more as pixels with a pixel value of 1. Format should be white on black background.

The mode control allows to filter Rectangle Information and Region Information so that only relevant figures will be used for output generation. For aspect ratio (mode = 011, 100 and 101) a support function aspectfloat4contourfixedint(float) is available to create the required int value.

Mode	0000RFFF	Filter definition register
	R	origin of output coordinates
	0	coordinates are relative to src picture
	1	coordinates are relative to crop window
	FFF	filter mode
	000	boxWidth >= threshold_widthOrMin && boxHeight > threshold_heightOrMax
	001	regionCnt > threshold_region
	010	boxWidth >= threshold_widthOrMin && boxHeight > threshold_heightOrMax && regionCnt > threshold_region
	011	boxWidth/boxHeight*256 >= threshold_widthOrMin && boxWidth/boxHeight*256 <= threshold_heightOrMax && regionCnt >= threshold_region
	100	boxWidth/boxHeight*256 >= threshold_widthOrMin && regionCnt >= threshold_region
	101	boxWidth/boxHeight*256 <= threshold_heightOrMax && regionCnt >= threshold_region
	110	undefined
	111	undefined

The processing performs by the function is equivalent to that of the OpenCV cv::findcontours function with the specifying of CV_RETR_LIST for mode and CV_CHAIN_APPROX_NONE for method. However, this function output is unique format.
Reference URL: https://opencv.org/

The crop window and always output of the "region information count" and the filter operations are an add on and not part of the cv::findcontours function, so in case opencv should be used several functions have to be combined to get the same behavior.

This function allows the same address to be specified for both src and work. However, input image is broken because this function writes out data at the area specified by work during processing.

The function reports in addition the error status in the first box record
Condition: x_coordinate== y_coordinate== width == height == 0 && ((count &0x07) != 0)
count & 0x01 → Error: window out of right border
count & 0x02 → Error: window out of bottom border
count & 0x04 → Error: crop window box to small

The debug interface allows the output of the input pictures (src), found boxes and shapes. The target address is specified by dst_rgbDebug address. Thr the input grey picture will be converted into an RGB picture by copying the grey value int R, G and B channel. The crop window will be drawn in blue color. The found region boxes in red and the shape pixel in green.

Dmode	WCBP000A	debug mode register
	W	draw cropping window
	0	off
	1	on
	C	draw contour data stored in dst_region
	0	off
	1	on
	B	draw bounding box
	0	off
	1	on
	P	Copy src picture
	0	off
	1	on
	A	output format
	0	RGB (24bit)
	1	ARGB (32bit) – A channel will be set to 0

Note Please take care to keep the crop Window within the allowed range within the input picture.

Note This function output size is depending of input image structure. Please allocate sufficient memory area for dst_rect and dst_region to avoid the memory broken by referring Rectangle Information and Region Information of I/O details and setting appropriate values to dst_rect_size and dst_region_size .

5. Using the DRP Library

To use this library, it is necessary to initialize the DRP, load configuration data, etc. Also, since the parameters are different for each configuration data, set the parameters based on the specification of the configuration data to be used. For application example of DRP library, refer to "RZ/A2M Group 2D Barcode Application Note (R01AN4503)".

Usage example:

```
#include " r_drp_find_contours_crop_x_t2.h"
#define FIND_CONTOURS_NUM (150)
```

….

// variable declaration

```
uint8_t  *output_bufadr;

int32_t ret_val;

static uint8_t drp_lib_id[R_DK2_TILE_NUM] = {0};
static volatile uint8_t drp_lib_status[R_DK2_TILE_NUM] = {DRP_NOT_FINISH};

static uint8_t frame_RAM_A[R_BCD_CAMERA_HEIGHT * R_BCD_CAMERA_WIDTH]
                __attribute__ ((section("ImageWork_RAM")));  /* canny edge picture */

static contours_rect_t   contours_rect_adr[FIND_CONTOURS_NUM + 1]
                          __attribute__ ((section("Uncache_IRAM")));
static contours_region_t contours_region_adr[FIND_CONTOURS_NUM + 1]
                          __attribute__ ((section("Uncache_IRAM")));
static contours_rect_t   cropped_contours_rect_adr[FIND_CONTOURS_NUM + 1]
                          __attribute__ ((section("Uncache_IRAM")));
static contours_region_t cropped_contours_region_adr[FIND_CONTOURS_NUM + 1]
                          __attribute__ ((section("Uncache_IRAM")));
static r_drp_find_contours_crop_t  param_find_contours_crop
                          __attribute__ ((section("Uncache_IRAM")));


static void cb_drp_finish(uint8_t id)
{
    uint32_t tile_no;
    /* Change the operation state of the DRP library notified by
       the argument to finish */
    for (tile_no = 0; tile_no < R_DK2_TILE_NUM; tile_no++)
    {
        if (drp_lib_id[tile_no] == id)
        {
            drp_lib_status[tile_no] = DRP_FINISH;
            break;
        }
    }
    return;
}

/***********************************************************************
* End of function cb_drp_finish
***********************************************************************/
…
```

```
/************************************/
/* Load DRP Library                 */
/*        +-----------------------+ */
/* tile 0 |                       | */
/*        +                       + */
/* tile 1 |                       | */
/*        +                       + */
/* tile 2 |                       | */
/*        +-----------------------+ */
/* tile 3 |                       | */
/*        +   FindContoursCrop    + */
/* tile 4 |                       | */
/*        +-----------------------+ */
/* tile 5 |                       | */
/*        +-----------------------+ */
/************************************/

/*#########################################*/
/* Load DRP Library FindContoursCrop    */
/*#########################################*/

ret_val = R_DK2_Load(&g_drp_lib_find_contours_crop_t2[0],
                     R_DK2_TILE_3,
                     R_DK2_TILE_PATTERN_3_2_1, NULL,
                     &cb_drp_finish, &drp_lib_id[0]);
DRP_DRV_ASSERT(ret_val);

/*#####################################*/
/* Activate DRP Lib FindContoursCrop */
/*#####################################*/
ret_val = R_DK2_Activate(drp_lib_id[TILE_3], 0);
DRP_DRV_ASSERT(ret_val);

/*************************************************************/
/* Set R_DK2_Start function parameters for FindContourCrop   */
/*************************************************************/

/* input image*/
param_find_contours_crop.src = (uint32_t)&frame_RAM_A[0]; /* Address of image 1-Layer*/
                   R_MMU_VAtoPA((uint32_t)param_find_contours_crop.src,
                    &(param_find_contours_crop.src ));

/* Address of rectangle output array*/
param_find_contours_crop.dst_rect = (uint32_t)&contours_rect_adr[0];
R_MMU_VAtoPA((uint32_t)param_find_contours_crop.dst_rect,
                     &(param_find_contours_crop.dst_rect ));

/* max number of rectangles to be generated */
param_find_contours_crop.dst_rect_size = FIND_CONTOURS_NUM;

/* Set Input-Image size */
/* The horizontal size (pixels) of image */
param_find_contours_crop.width  = R_BCD_CAMERA_WIDTH;
/* The vertical size (pixels) of image    */
param_find_contours_crop.height = R_BCD_CAMERA_HEIGHT;
```

```
/* Address of work area; minimum size = width * height / 4 + 8 */
param_find_contours_crop.work = (uint32_t)&work2[0];
R_MMU_VAtoPA((uint32_t)param_find_contours_crop.work,
                     &(param_find_contours_crop.work ));

/* Address of region information */
param_find_contours_crop.dst_region = (uint32_t)&contours_region_adr[0];
R_MMU_VAtoPA((uint32_t)param_find_contours_crop.dst_region,
                     &(param_find_contours_crop.dst_region ));
param_find_contours_crop.dst_region_size       = 0; /* 0: no region output */

/* Threshold of width or min value            */
param_find_contours_crop.threshold_widthOrMin  = aspectfloat4contourfixedint(3.1);
/* Threshold of height or max value           */
param_find_contours_crop.threshold_heightOrMax = aspectfloat4contourfixedint(5.9);

/* Crop window x position     */
param_find_contours_crop.x_crop                = 4;
/* Crop window y position     */
param_find_contours_crop.y_crop                = 4 ;
/* Crop window width  (min 3) */
param_find_contours_crop.width_crop            = R_BCD_CAMERA_WIDTH-8;
/* Crop window height (min 3) */
param_find_contours_crop.height_crop           = R_BCD_CAMERA_HEIGHT-8;

// Filter mode setting
 /* 0000RFFF (bit field)                                                    */
 /*  R - origin of output coordinates                                       */
 /*              0 - off (coordinates are relative to src picture)          */
 /*              1 - on  (coordinates relative to crop  window upper left (0 0))  */
 /*  F - filter for box/region found                                        */
 /*            000 - boxWidth              >=  threshold_widthOrMin &&
                     boxHeight             >=  threshold_heightOrMax       */
 /*            001 - regionCnt             >=  threshold_region            */
 /*            010 - boxWidth              >=  threshold_widthOrMin &&
                     boxHeight             >=  threshold_heightOrMax &&
                     regionCnt             >=  threshold_region            */
 /*            011 - boxWidth/boxHeight*256 >=  threshold_widthOrMin  &&
                     boxWidth/boxHeight*256 <=  threshold_heightOrMax &&
                     regionCnt             >=  threshold_region            */
 /*            100 - boxWidth/boxHeight*256 >=  threshold_widthOrMin  &&
                     regionCnt             >=  threshold_region            */
 /*            101 - boxWidth/boxHeight*256 <=  threshold_heightOrMax &&
                     regionCnt             >=  threshold_region            */
 /*            110 - reserved                                               */
 /*            111 - reserved                                               */
param_find_contours_crop.mode                  = 0b00000011 ;

// debug mode setting
/* WCBP000A (bit field)                                                     */
/*  W - draw cropping window                  -> dst_rgbDebug(RGB/ARGB)     */
/*              0 - off, 1 -on                                              */
/*  C - draw contour data stored in dst_region into   -> dst_rgbDebug(RGB/ARGB)  */
/*              0 - off, 1 -on                                              */
/*  B - boundingBox       draw bounding box into      -> dst_rgbDebug(RGB/ARGB)  */
/*              0 - off, 1 -on                                              */
/*  P - debug             copy picture src            -> dst_rgbDebug(RGB/ARGB)  */
/*              0 - off, 1 -on                                              */
/*  A - output format for debug               -> dst_rgbDebug(RGB/ARGB)     */
/*              0 - RGB (24bit), 1 -ARGB(32bit) A channel set to 0          */
param_find_contours_crop.dmode                   = 0b00000001 ;
```

```
/* Threshold contour count (number of found contour pixel) (output box is created only
if the surrounding shape has more than 50 pixel) */
param_find_contours_crop.threshold_region       = 50 ;

/* Output address for RGB/LRGB picture                                    */
/* debug bounding box(blue), found region(red), found pixel(green)        */
/* set to null if not used                                                */
/* Size is same as "width" * "height" of "src" * [3,4]                    */
param_find_contours_crop.dst_rgbDebug           = 0 ;


/* Initialize variables to be used in termination judgment of the DRP library */
drp_lib_status[TILE_3] = DRP_NOT_FINISH;

/***************************************/
/* Start DRP Library FindContoursCrop */
/***************************************/
ret_val = R_DK2_Start(drp_lib_id[TILE_3], (void *)&param_find_contours_crop,
                                           sizeof(r_drp_find_contours_crop_t));
DRP_DRV_ASSERT(ret_val);

/***************************************/
/* Wait until FindContoursCrop finished */
/***************************************/
while (drp_lib_status[TILE_3] == DRP_NOT_FINISH);
```

…..

// clean up

```
/***************************************/
/* Unload DRP library FindContoursCrop */
/***************************************/
ret_val = R_DK2_Unload(drp_lib_id[TILE_3], &drp_lib_id[TILE_3]);
DRP_DRV_ASSERT(ret_val);
```

6. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware (R01UH0746)

(Download the latest version of the update or news from the Renesas Electronics website.)

User's Manual: Software

RZ/A2M Group DRP Driver User's Manual (R01US0355)

(Download the latest version of the update or news from the Renesas Electronics website.)

RZ/A2M Group 2D Barcode Sample Program Application Note (R01AN4503)

(Download the latest version of the update or news from the Renesas Electronics website.)

User's Manual: Development environment

For the Renesas Electronics integrated development environment (e^2 studio), visit the Renesas Electronics website to download the latest version.

Technical Update/Technical News

(Download the latest version of the update or news from the Renesas Electronics website.)

Revision History	RZ/A2M Group DRP Library User's Manual

Rev.	Date	Description	
		Page	Summary
1.00	Sep 24, 2019	—	First Edition issued

RZ/A2M Group DRP Custom Library User’s Manual

Publication Date: Rev.1.00 Sept. 24, 2019

Published by: Renesas Electronics Corporation

RZ/A2M Group

Renesas Electronics Corporation