User's Manual

RZ/A2M Group

DRP Driver User's Manual

Renesas Electronics
www.renesas.com

Rev.1.03 Mar. 2021

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord 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 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 one with a different part number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

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 driver software and how to utilize them. It is aimed at users designing application systems making use of the software. 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 software. These notes occur within the body of the text, and 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.

Table of Contents

1. Introduction

1.1 Summary

This manual describes the functions and usage of the DRP driver software, which controls the dynamic reconfigurable processor (DRP) of RZ/A2M Group microprocessors.

1.2 Functions

DRP can be implemented a variety of functions corresponding to user's setting. In this manual the function implemented by DRP is referred to as "circuit" and the data representing the circuit information is referred to as "configuration data."*1 The configuration data consists of binary data allocated in the memory.
As a device driver for the DRP, the DRP driver performs the following functions:

- Supplies a clock to the DRP and initializes the DRP driver.
- Stops supply of the clock to the DRP and terminates the DRP driver.
- Loads configuration data in the DRP.
- Erases configuration data loaded in the DRP. (Calls "unload" in this document.)
- Supplies a clock to and enables circuits written to the DRP.
- Stops supply of the clock to and disables circuits written to the DRP.
- Sets operation parameters of circuits written to the DRP and starts operation.
- Provides notification of operation completion by circuits written to the DRP.
- Gets the status (enabled or disabled, operating or not, etc.) of circuits written to the DRP.
- Gets information (version, etc.) from configuration data in the memory.
- Performs CRC checks on configuration data in the memory.

Note 1. Configuration data provided as DRP library. For details of DRP library, refer to RZ/A2M Group DRP Library User's Manual (R01US0367).

1.3 Software Configuration

The software configuration of the DRP driver is shown below. The DRP driver comprises an interface portion and a core portion, and both are supplied as source code. The DRP driver supports FreeRTOS via an OS abstraction layer.

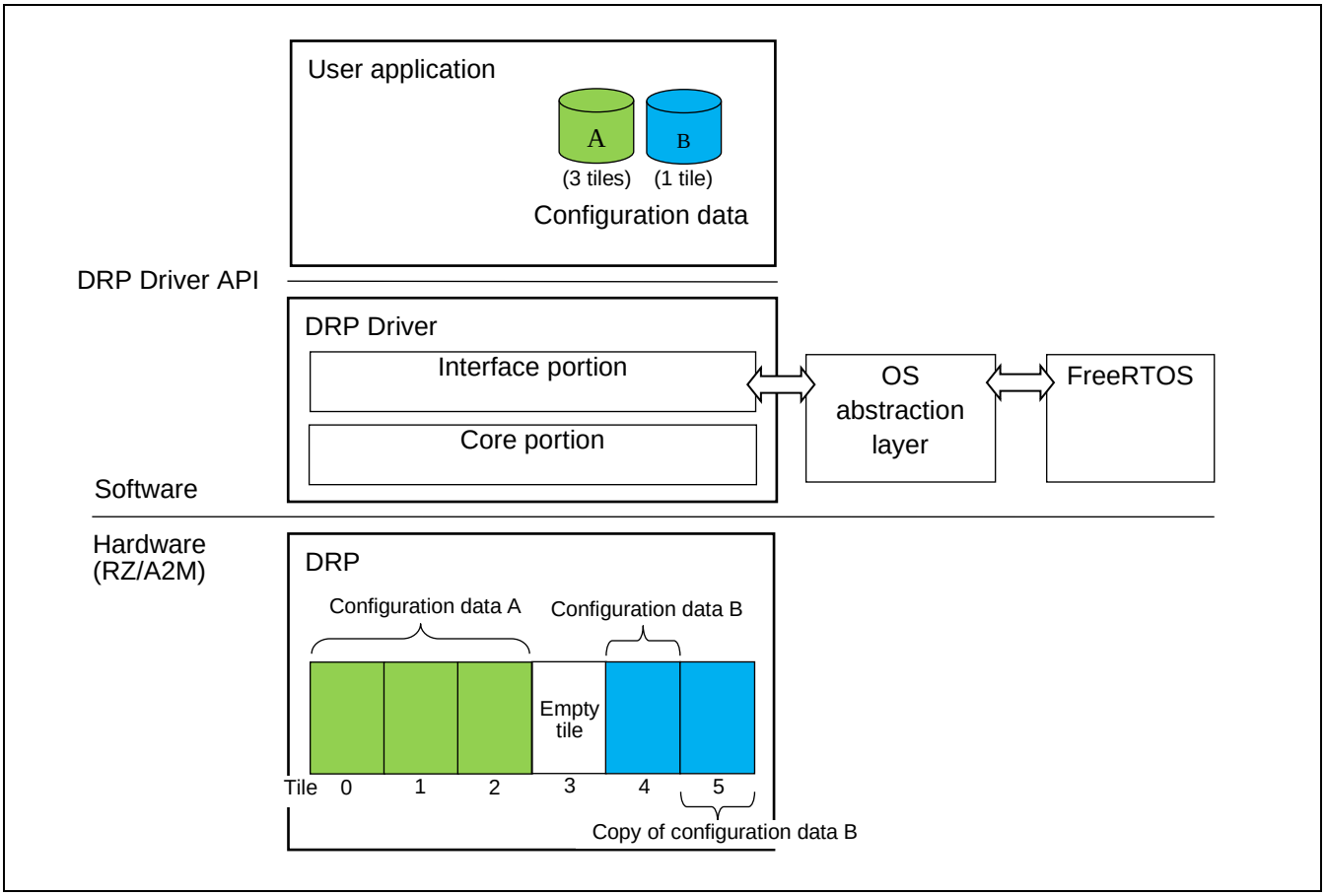


Figure 1.1 Software Configuration

- The DRP has six memory areas called “tiles” for loading configuration data.
- Configuration data is loaded in tile units.
- Each item of configuration data has its own tile count, represented as an integer value between 1 and 6. The tile count represents the number of tiles occupied by the configuration data.
- If the configuration data has a tile count of 3 or less, multiple copies can be loaded at the same time.
- In this manual the six tiles of the DRP are referred to as tile 0 to tile 5.
- In the figure above, one copy of configuration data A with tile count 3 is loaded in tile 0 through tile 2, and two copies of configuration data B with tile count 1 are loaded in tile 4 and tile 5, respectively.

2. Operation Conditions

The DRP driver operates under the conditions listed below.

Table 2.1 Operation Conditions

Item	Description
Microprocessor	The DRP driver runs on the Cortex™-A9 processor of RZ/A2M Group microprocessors. The product numbers of compatible RZ/A2M Group microprocessors are as follows:*1 R7S921051VCBG R7S921052VCBG R7S921053VCBG
Development environment	e2 studio V7.8.0 The following toolchain is compatible: GNU Arm Embedded Toolchain 6-2017-q2-update

Note 1. The DRP driver 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.

3. File Structure

Figure 3.1 shows the file structure of the DRP driver.

```
src
    renesas
        drivers
            drp
                inc
                    r_dk2_if.h          Header file of DRP Driver interface part
                src
                    drp_iodefine.h      IO definition file of DRP
                    r_dk2_core.c        Source file of DRP Driver core part
                    r_dk2_core.h        Header file of DRP Driver core part
                    r_dk2_if.c          Source file of DRP Driver interface part
```

Figure 3.1 The File Structure of The DRP Driver

4. API Specifications

4.1 List of API Functions

Table 4.1 lists the API functions of the DRP driver.

Table 4.1 API Functions of DRP Driver

API Function Name	Outline	Page
R_DK2_Initialize	Initializes DRP driver and initializes DRP.	7
R_DK2_Uninitialize	Stops DRP and terminates DRP driver.	8
R_DK2_Load	Loads configuration data in DRP.	9
R_DK2_Unload	Unloads configuration data from DRP.	15
R_DK2_Activate	Enables circuit in DRP.	16
R_DK2_Inactivate	Disables circuit in DRP.	17
R_DK2_Start	Starts operation of circuit in DRP.	18
R_DK2_GetStatus	Gets state of circuit in DRP.	20
R_DK2_GetInfo	Gets information from configuration data and checks CRC.	21
R_DK2_GetVersion	Gets DRP driver version information.	23

None of the API functions may be called from an interrupt context. For information on the reentrancy of API functions, refer to section 8, OS-Dependent Portion.

4.2 Error Codes

A return value of 0 or a positive number from a DRP driver API function indicates a normal end, and a negative return value indicates an abnormal end. When an abnormal end occurs, an error code is returned. Table 4.2 lists the error codes. For the specific conditions under which errors are generated, refer to the descriptions of the return values of the various API functions in section 5, API Reference.

Table 4.2 Function Error Codes

Macro Name	Value	Description
R_DK2_SUCCESS	0	Normal end
R_DK2_ERR_ARG	-1	Argument error
R_DK2_ERR_FORMAT	-2	Format error
R_DK2_ERR_CRC	-3	CRC error
R_DK2_ERR_DEVICE	-4	Device error
R_DK2_ERR_BUSY	-5	Busy
R_DK2_ERR_INTERNAL	-6	Internal error
R_DK2_ERR_OVERWRITE	-7	Data overwrite error
R_DK2_ERR_OS	-8	OS error
R_DK2_ERR_STATUS	-9	Status error
R_DK2_ERR_TILE_PATTERN	-10	Tile pattern error
R_DK2_ERR_STOPPED	-11	Transfer stopped error

5. API Reference

5.1 How to Read the API Reference

API function name	Category
Function outline	Synchronous/asynchronous function
Format	Shows the format used to call the API function. The header file designated by #include “header file” is the standard header file required to run the API function. Do not fail to include this header file. The designations I and O indicate that the corresponding argument is input data or output data, respectively. The designation IO indicates input/output data.
Return values	Lists the return values of the API function. Comments following the colon (:) after the return value provide a description of the return value (such as return conditions).
Description	Describes the specifications of the API function.
Note	Any precautionary notes appear here.

5.2 R_DK2_Initialize

R_DK2_Initialize			DRP driver API
Initializes DRP driver and initializes DRP			Synchronous function
Format	#include "r_dk2_if.h" int32_t R_DK2_Initialize(void);		
Return values	R_DK2_SUCCESS	:	Normal end.
	R_DK2_ERR_DEVICE	:	Abnormal end. This error is generated when initialization of the DRP fails.
	R_DK2_ERR_OS	:	Abnormal end. This error is generated when securing of an OS resource fails.
	R_DK2_ERR_STATUS	:	Abnormal end. This error is generated when the DRP driver has already been initialized.
Description	This API function initializes internal variables and secures OS resources, putting the DRP driver into a usable state. Also, it restores the DRP from low-power mode, starts supply of the clock, and initializes the hardware.		
Note	If the error R_DK2_ERR_DEVICE occurs, check the device used. The DRP driver is compatible with RZ/A2M Group microprocessors equipped with a DRP function module. For details of the DRP driver operating conditions, refer to section 2, Operation Conditions. If the value R_DK2_ERR_OS is returned, reevaluate the OS settings.		

5.3 R_DK2_Uninitialize

R_DK2_Uninitialize			DRP driver API
Stops DRP and terminates DRP driver			Synchronous function
Format	#include "r_dk2_if.h" int32_t R_DK2_Uninitalize(void);		
Return values	R_DK2_SUCCESS	:	Normal end.
	R_DK2_ERR_OS	:	Abnormal end. This error is generated when releasing of an OS resource fails.
	R_DK2_ERR_STATUS	:	Abnormal end. This error is generated when the DRP driver has already been terminated.
Description	This API function stops supply of the clock to the DRP and transitions the DRP to low-power mode. It performs a forced stop if the DRP is operating. Also, it releases OS resources and transitions the DRP driver to the uninitialized state. After this API function runs, the DRP driver remains in an unusable state until the next time the R_DK2_Initialize function is called.		
Note	This API function performs a forced stop if the DRP is operating. Note that in this case the callback function set by the R_DK2_Load function may not be called. If the value R_DK2_ERR_OS is returned, reevaluate the OS settings.		

5.4 R_DK2_Load

R_DK2_Load

DRP driver API

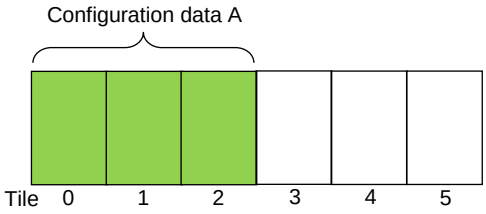
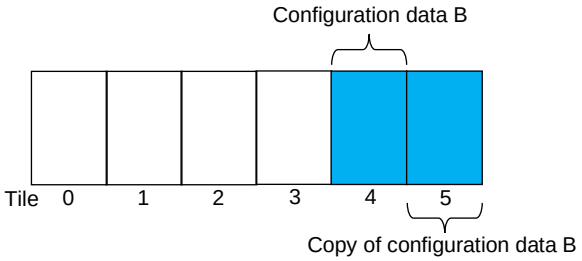
Loads configuration data in DRP

Synchronous/asynchronous function

Format

#include "r_dk2_if.h"

int32_t R_DK2_Load(const void *const pconfig, const uint8_t top_tiles, const uint32_t tile_pattern, const load_comp_t pload, const process_comp_t pprocess, uint8_t *const paid);

pconfig	I	Specifies the address of the configuration data to be loaded. The configuration data must be aligned with a 32-byte boundary. Also, the configuration data must exist in physical memory.
top_tiles	I	Specifies the start tile position where the configuration data is allocated using macros R_DK2_TILE_0 to R_DK2_TILE_5, which represent the six tiles of the DRP, tile 0 to tile 5. When loading multiple configuration data items, obtain the logical sum (logical OR) of each bit of the above macros. For example, to allocate configuration data A with tile count 3 to tile 0 through tile 2, specify “R_DK2_TILE0”.  For example, to allocate two copies of configuration data B with tile count 1 to tile 4 and tile 5, respectively, specify “R_DK2_TILE_4 | R_DK2_TILE_5”. 
tile_pattern	I	Specifies the tile pattern. For setting values, refer to 5.4.1, Tile Patterns. Once the tile pattern has been set, use the same tile pattern setting until the configuration data for all tiles has been unloaded using the R_DK2_Unload function. When an attempt is made to change the tile pattern when the DRP is in a state in which configuration data has already been loaded, the API function returns a value of R_DK2_ERR_TILE_PATTERN.

pload	I	Specifies the address of the callback function used to provide notification when loading of configuration data completes. For detailed specifications of the callback function specified by the argument pload, refer to 5.4.2, Load Completion Callback Function. When a value other than NULL is specified for this argument, loading of configuration data can be halted by the R_DK2_Unload function. When NULL is specified for this argument, the R_DK2_Unload function cannot halt loading of configuration data, and this API function finishes only when loading is complete.
pprocess	I	Specifies the address of the callback function used to provide notification when the processing started using the R_DK2_Start function completes. For detailed specifications of the callback function specified by the argument pprocess, refer to 5.8.1, Processing Completion Callback Function. This notification does not occur if NULL is specified.
paid	O	Specifies the address of the six-element array used to perform notification of the ID for identifying the loaded configuration data. Index 0 to index 5 of the array represent the six tiles of the DRP, tile 0 to tile 5, and the array elements represent the IDs of the configuration data items loaded in the corresponding tiles. If a configuration data item occupies multiple tiles, the same ID is stored in all the array elements representing the corresponding tiles. Each ID is a unique positive number corresponding to a single circuit, and a value of 0 means that no configuration data is loaded. If multiple copies of a configuration data item are loaded, each copy is assigned a different ID. When notification of IDs is made by this argument, the notification covers the IDs for all six tiles following execution of the R_DK2_Load function, including all configuration data that has been written to that point. This notification does not occur if NULL is specified. For example, if configuration data A with tile count 3 is allocated to tile 0 through tile 2, and two copies of configuration data B with tile count 1 are allocated to tile 4 and tile 5, respectively, the contents of the array are as shown below.

Index	Description
0	Circuit ID of configuration data A circuit information
1	Same as index 0
2	Same as index 0
3	0
4	Circuit ID of configuration data B circuit information
5	Circuit ID of configuration data B circuit information (different from index 4)

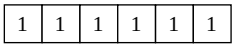
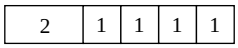
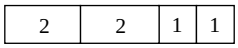
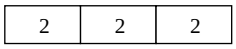
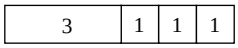
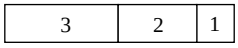
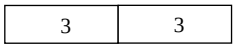
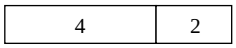
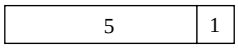
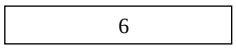
Return values	R_DK2_SUCCESS	:	Normal end.
	R_DK2_ERR_ARG	:	Abnormal end. This error is generated in the following cases: • NULL is specified for argument pconfig. • A value that is not aligned with a 32-byte boundary is specified for argument pconfig. • The argument top_tiles is not in the format of the logical sum (logical OR) of each bit of R_DK2_TILE_0 to R_DK2_TILE_5. • A macro other than those listed in Table 5.1 is specified for argument tile_pattern.
	R_DK2_ERR_FORMAT	:	Abnormal end. This error is generated when a format error is detected in the configuration data.
	R_DK2_ERR_DEVICE	:	Abnormal end. This error is generated when NULL is specified for argument pload and a transfer error occurs during loading of configuration data.
	R_DK2_ERR_BUSY	:	Abnormal end. This error is generated when a value other than NULL is specified for argument pload and, during loading of configuration data, an attempt is made to load other configuration data.
	R_DK2_ERR_OVERWRITE	:	Abnormal end. This error is generated when other configuration data has already been written to the load position of the specified configuration data.
	R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
	R_DK2_ERR_STATUS	:	Abnormal end. This error is generated when the DRP driver has not been initialized.
	R_DK2_ERR_TILE_PATTERN	:	Abnormal end. This error is generated in the following cases: • The tile pattern is changed when the DRP is in a state in which configuration data has already been loaded. • The tile position or tile count in the configuration data do not match the tile pattern.

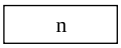
Description	When a value other than NULL is specified for the argument pload, this API function starts loading the configuration data in the DRP and notifies when loading completes by means of a callback function. At this time, other configuration data cannot be loaded until loading completes. In such cases the value R_DK2_ERR_BUSY is returned, and this API function fails. Also, if a value other than NULL is specified for the argument pload, it is possible to halt loading of configuration data with the R_DK2_Unload function. When NULL is specified for the argument pload, loading of the configuration data continues until completion when this API function is run. In this case, loading of configuration data cannot be halted by the R_DK2_Unload function. It is also possible for this API function to load configuration data to multiple tile positions. For details of the callback function specified by the argument pload, refer to 5.4.2, Load Completion Callback Function, and for details of the callback function specified by the argument pprocess, refer to 5.8.1, Processing Completion Callback Function. This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.
Note	If the value R_DK2_ERR_FORMAT is returned, check to make sure the address specified for argument pconfig is the correct address of the configuration data. A return value of R_DK2_ERR_DEVICE indicates that an error occurred during transfer of the configuration data. Reevaluate the memory settings, etc., for the allocation of the configuration data. If the configuration data specified by the argument pconfig exists in the Cortex-A9 cache and the data in the physical memory does not match the configuration data, proper loading will not be possible. It may be necessary to clear the cache before calling this API function or to allocate the configuration data to a non-cached area.

5.4.1 Tile Patterns

The tile count and tile position combinations used when loading configuration data in the DRP are limited to the 11 patterns listed in Table 5.1. Set the appropriate macro value below in the argument tile_pattern of the R_DK2_Load function to match the combination to be used.

Table 5.1 Tile Patterns

Tile Pattern	Macro Setting of Argument tile_pattern of R_DK2_Load Function
	R_DK2_TILE_PATTERN_1_1_1_1_1_1
	R_DK2_TILE_PATTERN_2_1_1_1_1
	R_DK2_TILE_PATTERN_2_2_1_1
	R_DK2_TILE_PATTERN_2_2_2
	R_DK2_TILE_PATTERN_3_1_1_1
	R_DK2_TILE_PATTERN_3_2_1
	R_DK2_TILE_PATTERN_3_3
	R_DK2_TILE_PATTERN_4_1_1
	R_DK2_TILE_PATTERN_4_2
	R_DK2_TILE_PATTERN_5_1
	R_DK2_TILE_PATTERN_6

 : Configuration data with tile count n

5.4.2 Load Completion Callback Function

Load completion callback function

Callback function

Completion of loading of configuration data

Synchronous function

Format

```
#include "r_dk2_if.h"
void load_comp(uint8_t id, int32_t result);
Note: This function can be given any name.
```

id	I	ID of circuit that has finished loading
result	I	R_DK2_SUCCESS: Indicates that loading has completed successfully. R_DK2_ERR_DEVICE: Indicates that a transfer error occurred while loading configuration data. R_DK2_ERR_STOPPED: Indicates that while loading configuration data the transfer was stopped by calling the R_DK2_Unload function.

Return values: None

Description: This is the callback function specified by the argument pload of the R_DK2_Load function. It provides notification when the loading of configuration data finishes. When multiple configuration data items are loaded, this callback function is called once for each item loaded. This function is executed in the interrupt context. This function must not call any DRP driver function.

Note: If the value of the argument result is R_DK2_ERR_DEVICE, reevaluate the memory settings, etc., for the allocation of the configuration data.

5.5 R_DK2_Unload

R_DK2_Unload

DRP driver API

Unloads configuration data from DRP

Synchronous function

Format

```
#include "r_dk2_if.h"
int32_t R_DK2_Unload(const uint8_t id, uint8_t *const paid);
```

id	I	Specifies the ID of the circuit to be unloaded. To unload multiple circuits, specify the logical sum (logical OR) of each bit of the IDs of each of the circuits. Specifying 0 causes all loaded circuits to be unloaded.
paid	O	To obtain notification of the DRP load status following execution of this function, specify the address of a six-element array prepared by the user. Index 0 to index 5 of the array represent the six tiles of the DRP, tile 0 to tile 5, and the array elements represent the IDs of the configuration data items loaded in the corresponding tiles. If a configuration data item occupies multiple tiles, the same ID is stored in all the array elements representing the corresponding tiles. This ID is a unique positive number corresponding to a single circuit, and a value of 0 means that no configuration data is loaded. If multiple copies of the same configuration data item are loaded, each copy is assigned a different ID. When notification of IDs is made by this argument, the notification covers the IDs for all six tiles following execution of the R_DK2_Unload function, including all configuration data that has been written to that point. This notification does not occur if NULL is specified.

Return values

R_DK2_SUCCESS	:	Normal end.
R_DK2_ERR_ARG	:	Abnormal end. This error is generated when the argument id does not correspond to a circuit currently loaded in the DRP.
R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
R_DK2_ERR_STATUS	:	Abnormal end. This error is generated in the following cases: The DRP driver has not been initialized.

Description

This API function unloads the circuit corresponding to the specified ID from the DRP. After the circuit is unloaded, configuration data can once again be loaded in the same tile position. This API function will forcibly unload the circuit even if it is in the process of being loaded or if it is operating.

If this API function is called during loading of configuration data, loading of data is canceled and the callback function specified by the pload argument of the R_DK2_Load function is called. At this point, the value of the callback function's result argument is R_DK2_ERR_STOPPED. Also, if this API function is called during circuit operation, the circuit stops operating and the callback function specified by the pprocess argument of the R_DK2_Load function is called. At this point, the value of the callback function's result argument is R_DK2_ERR_STOPPED.

It is also possible to unload multiple circuits or all currently loaded circuits.

This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.

Note

None.

5.6 R_DK2_Activate

R_DK2_Activate

DRP driver API

Enables circuit in DRP

Synchronous function

Format

```
#include "r_dk2_if.h"
int32_t R_DK2_Activate(const uint8_t id, const uint32_t freq);
```

id	I	Specifies the ID of the circuit to be enabled. To enable multiple circuits, specify the logical sum (logical OR) of each bit of the IDs of each of the circuits. Specifying 0 causes all loaded circuits to be enabled.
freq	I	Specifies 0.

Return values

R_DK2_SUCCESS	:	Normal end.
R_DK2_ERR_ARG	:	Abnormal end. This error is generated when the value of the argument id does not correspond to a circuit currently loaded in the DRP.
R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
R_DK2_ERR_STATUS	:	Abnormal end. This error is generated in the following cases: • The DRP driver has not been initialized. • The circuit specified by the argument id is not in the loaded state. • 0 was specified for the argument id and no circuit is currently in the loaded state. (For information on circuit states, refer to 6.2, State Transitions of Individual Circuits.)

Description

This API function enables a circuit currently loaded in the DRP, supplies a clock to the corresponding tile, and puts the circuit into a usable state.

It is also possible to activate multiple circuits or all currently loaded circuits. When 0 is specified as the argument id in order to enable all circuits, only circuits currently in the loaded state are affected. (For information on circuit states, refer to 6.2, State Transitions of Individual Circuits.)

This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.

Note

None.

5.7 R_DK2_Inactivate

R_DK2_Inactivate			DRP driver API
Disables circuit in DRP			Synchronous function
Format	#include "r_dk2_if.h" int32_t R_DK2_Inactivate(const uint8_t id);		
	id	I	Specifies the ID of the circuit to be disabled. To disable multiple circuits, specify the logical sum (logical OR) of each bit of the IDs of each of the circuits. Specifying 0 causes all loaded circuits to be disabled.
Return values	R_DK2_SUCCESS	:	Normal end.
	R_DK2_ERR_ARG	:	Abnormal end. This error is generated when the value of the argument id does not correspond to a circuit currently loaded in the DRP.
	R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
	R_DK2_ERR_STATUS	:	Abnormal end. This error is generated in the following cases: • The DRP driver has not been initialized. • The circuit specified by the argument id is not in the activated or started state. • 0 was specified for the argument id and no circuit is currently in the activated or started state. (For information on circuit states, refer to 6.2, State Transitions of Individual Circuits.)
Description	This API function disables a circuit currently loaded in the DRP, stops supply of the clock to the corresponding tile, and puts the circuit into the low-power state. It is also possible to disable multiple circuits or all currently loaded circuits. When 0 is specified as the argument id in order to disable all circuits, only circuits currently in the activated or started state are affected. This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.		
Note	None		

5.8 R_DK2_Start

R_DK2_Start

DRP driver API

Starts operation of circuit in DRP

Asynchronous function

Format

```
#include "r_dk2_if.h"
int32_t R_DK2_Start(const uint8_t id, const void *const pparam, const uint32_t size);
```

id	I	Specifies the ID of the circuit that will start operating.
pparam	I	Specifies the area for storing parameters for circuit operation. The area where parameters are stored must exist in physical memory. The parameter storage area for each circuit is read independently, so it is not possible for one area to be shared by multiple circuits. The parameter specifications are different for each configuration data. For the parameter specifications of each configuration data, refer to RZ/A2M Group DRP Library User's Manual (R01US0367).
size	I	Specifies the size of the parameter area specified by the argument pparam.

Return values

R_DK2_SUCCESS	:	Normal end.
R_DK2_ERR_ARG	:	Abnormal end. This error is generated in the following cases: • The value of the argument id does not correspond to a circuit currently loaded in the DRP. • NULL is specified for the argument pparam. • 0 is specified for the argument size.
R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
R_DK2_ERR_STATUS	:	Abnormal end. This error is generated in the following cases: • The DRP driver has not been initialized. • The circuit specified by the argument id is not in the activated state. (For information on circuit states, refer to 6.2, State Transitions of Individual Circuits.)

Description

This API function starts operation of a circuit loaded in the DRP. Notification of the completion of processing is provided by the processing completion callback function specified by the argument pprocess of the R_DK2_Load function. For details of the processing completion callback function, refer to 5.8.1, Processing Completion Callback Function.

This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.

Note

If the DRP is in a state where the area set by the argument pparam for storing parameters or the circuit's I/O data exists in the cache of the Cortex-A9, and the parameters or circuit I/O data in physical memory do not match, the circuit will not operate properly. It may be necessary to clear the cache before calling this API function or to allocate the parameters and circuit I/O data to a non-cached area.

5.8.1 Processing Completion Callback Function

Processing completion callback function

Callback function

Completion of processing started by R_DK2_Start

Synchronous function

Format

```
#include "r_dk2_if.h"
void process_comp(uint8_t id, int32_t result);
```

Note: This function can be given any name.

id	I	ID of circuit whose processing has finished
result	I	R_DK2_SUCCESS: Indicates that processing has completed successfully. R_DK2_ERR_DEVICE: Indicates that a transfer error occurred while transferring parameters set by the R_DK2_Start function or while transferring circuit I/O data. R_DK2_ERR_STOPPED: Indicates that while transferring parameters set by the R_DK2_Start function or while transferring circuit I/O data the transfer was stopped by calling the R_DK2_Unload function or the R_DK2_Inactivate function.

Return values: None

Description: This is the callback function specified by the argument pprocess of the R_DK2_Load function. It provides notification when the processing started by R_DK2_Start function finishes. The number of times this callback function is called is the same as the number of times the R_DK2_Start function is called, unless an event such as a forced unload by the R_DK2_Unload function occurs.

This function is executed in the interrupt context. This function must not call any DRP driver function.

Note: If the value of the argument result is R_DK2_ERR_DEVICE, reevaluate the memory settings, etc., for the allocation of the parameters set by the R_DK2_Start function or circuit I/O data.

5.9 R_DK2_GetStatus

R_DK2_GetStatus			DRP driver API
Gets state of circuit in DRP			Synchronous function
Format	#include "r_dk2_if.h" int32_t R_DK2_GetStatus(const uint8_t id);		
	id	I	Specifies the ID of the circuit to be whose state is to be acquired.
Return values	R_DK2_STATUS_LOADED	:	Normal end. Indicates that the specified circuit is in the loaded state.
	R_DK2_STATUS_ACTICATED	:	Normal end. Indicates that the specified circuit is in the activated state
	R_DK2_STATUS_STARTED	:	Normal end. Indicates that the specified circuit is in the started state.
	R_DK2_STATUS_LOADING	:	Normal end. Indicates that the specified circuit is in the loading state.
	R_DK2_ERR_ARG	:	Abnormal end. This error is generated when the value of the argument id does not correspond to a circuit currently loaded in the DRP.
	R_DK2_ERR_OS	:	Abnormal end. This error is generated when exclusive control by the OS fails.
Description	This API function gets the state of a circuit currently loaded in the DRP. A positive return value means that the function completed successfully, and the value returned indicates the state of the circuit. A negative return value means that the function failed, and the value returned represents an error code. For information on circuit states in the DRP, refer to 6.2, State Transitions of Individual Circuits. This API function uses OS functionality to provide exclusive control so that multiple DRP driver API functions are not executed at the same time. If a failure occurs because resource acquisition times out during exclusive control, the value R_DK2_ERR_OS is returned and the API function fails.		
Note	None		

5.10 R_DK2_GetInfo

R_DK2_GetInfo

DRP driver API

Gets information from configuration data and checks CRC. Synchronous function

Format

```
#include "r_dk2_if.h"
int32_t R_DK2_GetInfo(const void *const pconfig, config_info_t *const pinfo, const bool crc_check);
```

pconfig	I	Specifies the address of the configuration data from which information is obtained. The configuration data must be aligned with a 32-byte boundary.
pinfo	O	Specifies the address of the structure config_info_t type variable. This API function stores the following information from the configuration data in the members of the structure:

Member Name	Type	Description
type	uint8_t	This area is reserved. The data stored here consists of zeros.
pname	char *	Stores a pointer to a character string of up to 31 bytes representing the circuit name.
ver	uint32_t	Stores the version of the configuration data.*1
cid	uint32_t	Stores a unique ID representing the circuit stored in the configuration data.

Note 1. The storage format of the member ver is as follows:

Bit Position	Description
0 to 7	Stores the build number.
8 to 15	Stores the minor version.
16 to 23	Stores the major version.
24 to 31	This area is reserved. The data stored here consists of zeros.

For example, a ver value of 0x00010201 represents version 1.21.

crc_check	I	Specifies as a truth value whether or not a CRC check is performed on the configuration data when getting information.

Return values

R_DK2_SUCCESS	:	Normal end.
R_DK2_ERR_ARG	:	Abnormal end. This error is generated when pconfig has a value of NULL or pinfo has a value of NULL.
R_DK2_ERR_FORMAT	:	Abnormal end. This error is generated when a format error is detected in the configuration data.
R_DK2_ERR_CRC	:	Abnormal end. This error is generated when the argument crc_check is set to true and a CRC error is detected in the configuration data.

Description	This API function gets information from the configuration data at the address specified by the argument pconfig. It writes the information obtained from the configuration data to the address specified by the argument pinfo. This API function also performs a CRC check on the configuration data. If the CRC check fails, the value R_DK2_ERR_CRC is returned and an abnormal end occurs.
Note	If a value of R_DK2_ERR_FORMAT is returned, confirm that the address specified by the argument pconfig is the correct address of the configuration data.

5.11 R_DK2_GetVersion

R_DK2_GetVersion

DRP driver API

Gets DRP driver version information

Synchronous function

Format

```
#include "r_dk2_if.h"
uint32_t R_DK2_GetVersion(void);
```

Return values

DRP driver version information : The storage format is as shown below.

Bit Position	Description
0 to 7	Stores the build number.
8 to 15	Stores the minor version.
16 to 23	Stores the major version.
24 to 31	This area is reserved. The data stored here consists of zeros.

For example, a return value of 0x00010201 represents version 1.21.

Description: This API function gets the version number of the DRP driver.

Note: None

6. State Transitions

6.1 State Transitions of the DRP Driver Overall

Figure 6.1 shows state transitions and the clock supply status of the DRP driver overall.

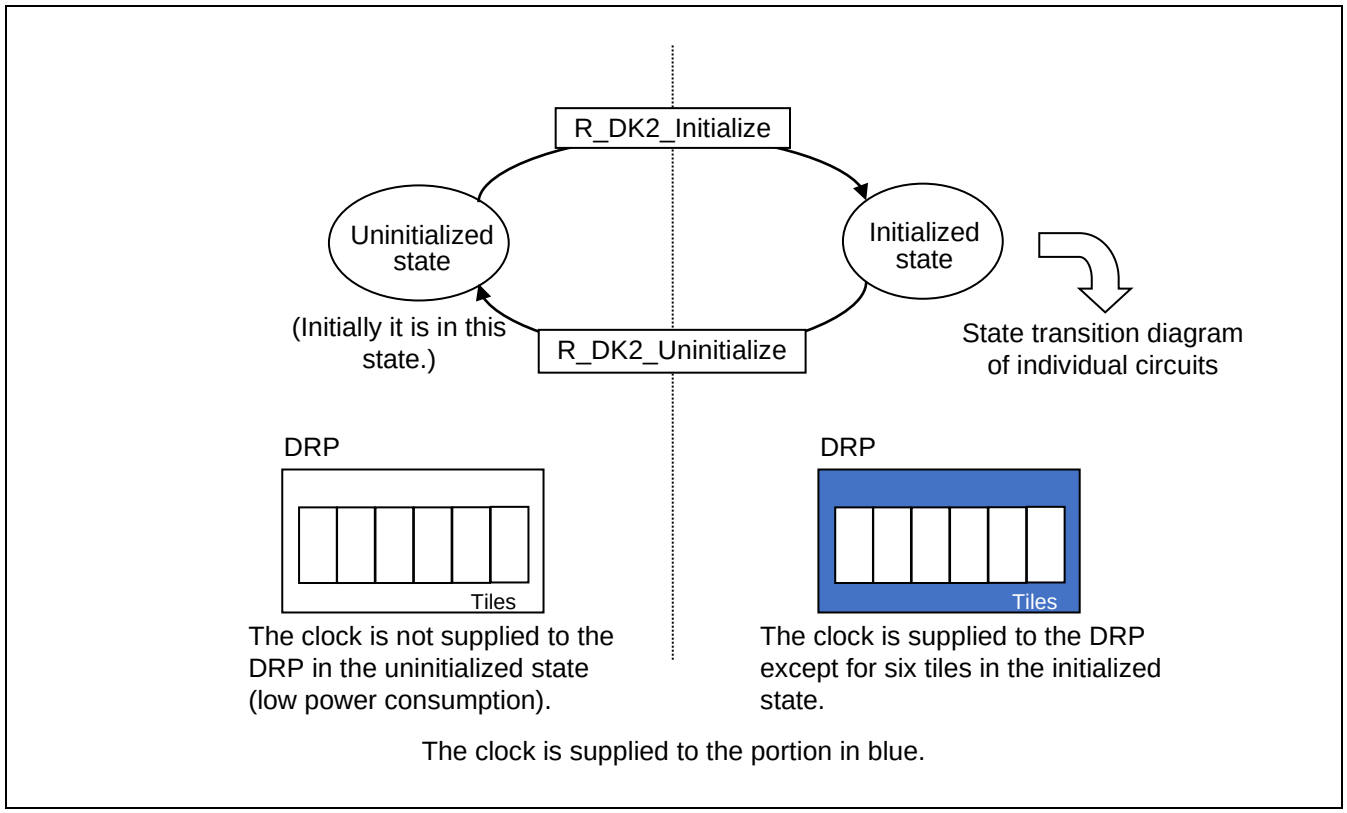
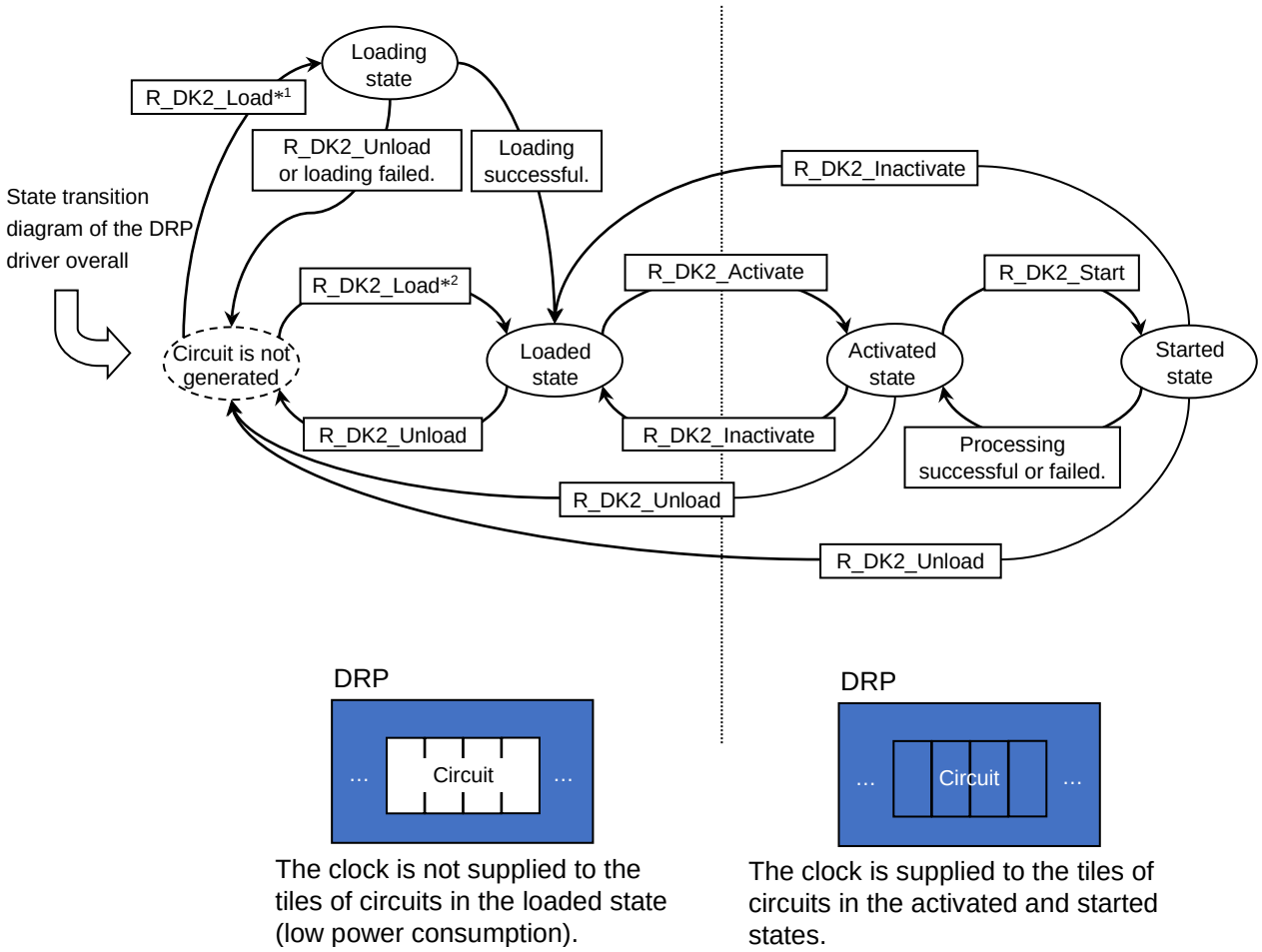


Figure 6.1 State Transitions and Clock Supply Status of DRP Driver Overall

6.2 State Transitions of Individual Circuits

Figure 6.2 shows state transitions and the clock supply status of individual circuits.



Note 1. When argument pload is set to anything other than NULL.
Note 2. When argument pload is set to NULL.

Figure 6.2 State Transitions and Clock Supply Status of Individual Circuits

7. Control Flowchart

Figure 7.1 is a flowchart of a DRP driver usage example.

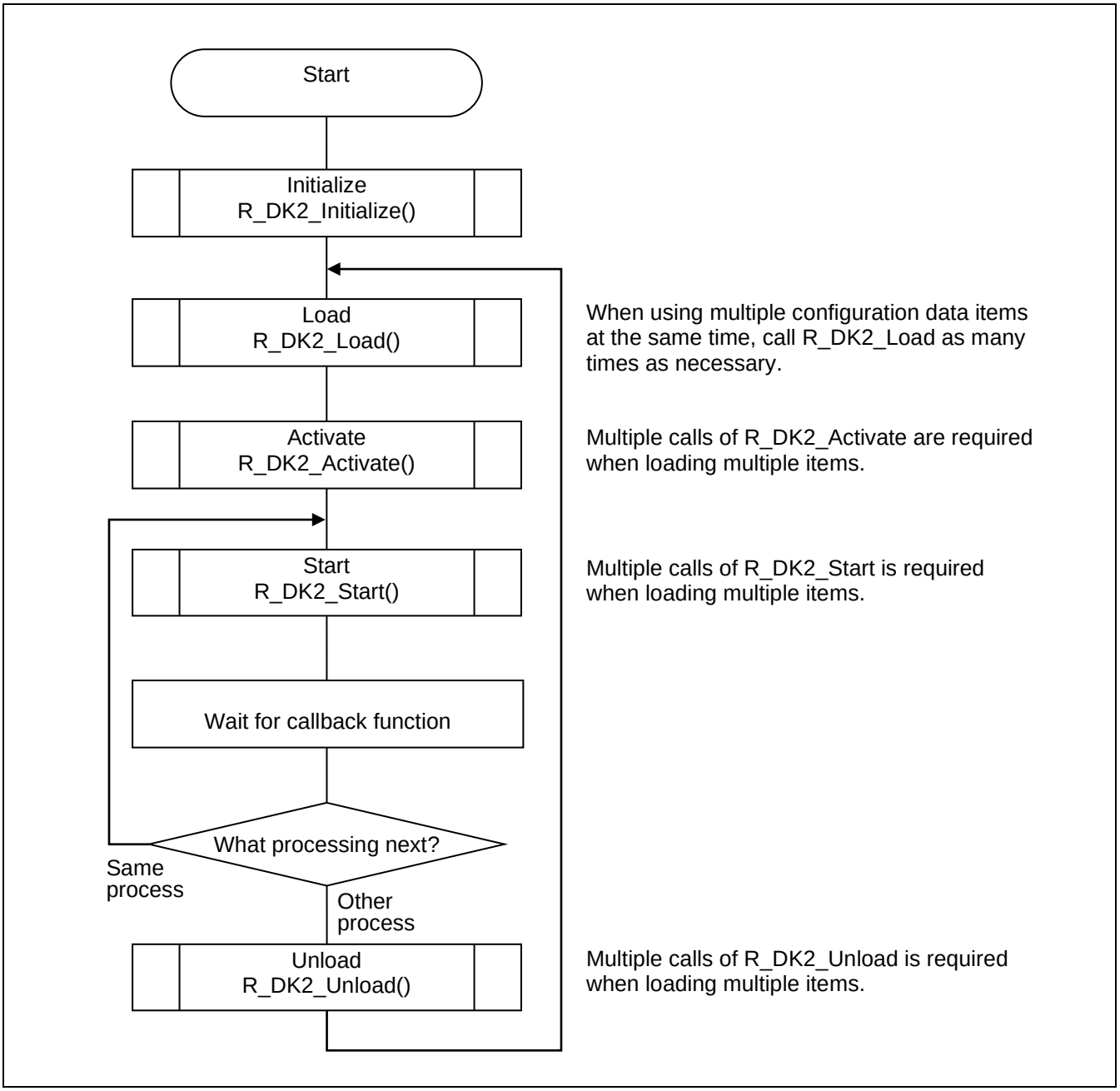


Figure 7.1 DRP Driver Usage Example

8. OS-Dependent Portion

8.1 Support for reentrancy of API functions

The OS-dependent portion of the DRP driver is separated from the rest as an OS abstraction layer. The DRP driver supports FreeRTOS via this OS abstraction layer.
The functionality provided by the DRP driver by means of the OS-dependent portion is support for reentrancy of API functions. Exclusive control employing the mutual exclusion (Mutex) capability of FreeRTOS is used to enable reentrancy for some of the API functions, as indicated in Table 8.1.
To implement reentrancy the DRP driver uses a single Mutex to provide exclusive control. When an API function supporting reentrancy is running and another API function supporting reentrancy is called, the second API function waits until the first API function finishes.
It is possible to use the macro MUTEX_WAIT defined in r_dk2_if.c to set the timeout duration during exclusive control. To specify the timeout duration, assign an integer between 0 and 0xFFFFFFFF to the macro MUTEX_WAIT. The setting value represents the timeout duration in millisecond units. A value of 0 means no wait. The default timeout duration setting is 100 milliseconds.

Table 8.1 Reentrancy Support of DRP Driver API Functions

API Function Name	Reentrancy Support	Page
R_DK2_Initialize	Reentrancy not supported	7
R_DK2_Uninitialize	Reentrancy not supported	8
R_DK2_Load	Reentrancy supported	9
R_DK2_Unload	Reentrancy supported	15
R_DK2_Activate	Reentrancy supported	16
R_DK2_Inactivate	Reentrancy supported	17
R_DK2_Start	Reentrancy supported	18
R_DK2_GetStatus	Reentrancy supported	20
R_DK2_GetInfo	Reentrancy not supported	21
R_DK2_GetVersion	Reentrancy not supported	23

8.2 DRP Driver Interrupt Priority

The DRP Driver interrupt priority levels are defined in the macros in Table 8.2.
FreeRTOS API functions cannot be called in interrupts that have a higher priority than the value of configMAX_API_CALL_INTERRUPT_PRIORITY defined in FreeRTOSConfig.h. Be careful when using FreeRTOS service calls to wait for DRP to complete.

Table 8.2 DRP Driver Interrupt Priority Macro Definition (r_dk2_if.h)

Macro Name	Value	Description
DRP_INTERRUPT_PRIORITY	26	DRP Driver interrupt priority level

9. Memory footprint

Table 9.1 lists the approximate sizes of memory used by the DRP Driver.

Table 9.1 Memory Resources

Section name	Size (approx.)
Code	12k bytes
Constant Data	0.1 Kbytes or less
Data	0.5k bytes
Stack size	400 bytes

10. Reference Documents

User's Manual: Hardware

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

(Download the latest version of the manual from the Renesas Electronics website.)

User's Manual: Software

RZ/A2M Group DRP Library User's Manual (R01US0367)

(Download the latest version of the manual from the Renesas Electronics website.)

User's Manual: Development Environment

For the Renesas Electronics integrated development environment (e2 studio), please 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.)

11. How to Import the Driver

11.1 e² studio

Please refer to the RZ/A2M Smart Configurator User's Guide: e² studio R20AN0583EJ for details on how to import drivers into projects in e² studio using the Smart Configurator tool.

11.2 For Projects created outside e² studio

This section describes how to import the driver into your project.

Generally, there are two steps in any IDE:

1) Copy the driver to the location in the source tree that you require for your project.
2) Add the link to where you copied your driver to the compiler.

Other required drivers, e.g. r_cbuffer, must be imported similarly.

REVISION HISTORY	RZ/A2M Group DRP Driver User's Manual

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 14, 2018	–	First Edition Issued
1.01	May. 31, 2019	29	Added the chapter of "10. How to Import the Driver".
1.02	Jun. 30, 2020	3	2 Operation Conditions, the version of RENESAS e2 studio was changed to 7.8.0.
		25	6.2 State Transitions of Individual Circuits, updated State transition diagram.
1.03	Mar. 31, 2021	–	Changed the DRP interrupt priority level from 8 to 26.
		27	Added the chapter of "8.2 DRP Driver Interrupt Priority".
		28	Added the chapter of "9. Memory footprint".

RZ/A2M Group DRP Driver User's Manual

Publication Date: Rev.1.00 Sep. 14, 2018
Rev.1.03 Mar. 31, 2021

Published by: Renesas Electronics Corporation

RZ/A2M Gruop

Renesas Electronics Corporation

R01US0355EJ0103

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc. Milpitas Campus
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics America Inc. San Jose Campus
6024 Silver Creek Valley Road, San Jose, CA 95138, USA
Tel: +1-408-284-8200, Fax: +1-408-284-2775

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai 200333, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, #06-02 Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0