User's Manual

RENESAS

CONFIDENTIAL

RH850/D1x Device Family

Renesas Graphics Library
OctaBus Controller (OCTA) Driver
User's Manual: Software

Renesas Electronics
www.renesas.com

Rev. 1.0 April 2020

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)

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding the functions of OCTA driver. This manual is written for engineers who use OCTA driver.

> 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, and in the Usage Notes 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.

Please refer to documents of drivers and hardware for a target system implementing OCTA as necessary.

The following documents are related documents. Make sure to refer to the latest versions of these documents.

Document Type	Description	Document Title	Document No.
User's manual for Hardware	Hardware specifications (pin assignments, memory maps, peripheral function specifications, electrical characteristics, timing charts) and operation description	RH850/D1L/D1M Group User's Manual: Hardware	R01UH0451EJ0220
User's manual for Software	Description of RGL overview	Renesas Graphics Library User's Manual: Software	R01US0181ED0400
	Description of WM	Renesas Graphics Library Window Manager (WM) Driver User's Manual: Software	LLWEB-10035990
	Description of SPEA	Renesas Graphics Library Sprite Engine A (SPEA) Driver User's Manual: Software	LLWEB-10035991
	Description of VDCE	Renesas Graphics Library Video Data Controller E (VDCE) Driver User's Manual: Software	LLWEB-10035992
	Description of VOWE	Renesas Graphics Library Video Output Warping Engine (VOWE) Driver User's Manual: Software	LLWEB-10035993
	Description of JCUA	Renesas Graphics Library JPEG Codec Unit A (JCUA) Driver User's Manual: Software	LLWEB-10035994
	Description of SFMA	Renesas Graphics Library Serial Flash Memory Interface A (SFMA) Driver User's Manual: Software	LLWEB-10064753
	Description of HYPB	Renesas Graphics Library HyperBus Controller (HYPB) Driver User's Manual: Software	LLWEB-10064754
	Description of OCTA	Renesas Graphics Library OctaBus Controller (OCTA) Driver User's Manual: Software	LLWEB-10064755 (This manual)
	Description of VOCA	Renesas Graphics Library Video Output Checker A (VOCA) Driver User's Manual: Software	LLWEB-10063801

	Description of DISCOM	Renesas Graphics Library Display Output Comparator (DISCOM) Driver User's Manual: Software	LLWEB-10063802
	Description of DRW2D	Renesas Graphics Library 2D Graphics (DRW2D) Driver User's Manual: Software	LLWEB-10059472
Porting Layer Guide	Description of porting layer of RGL	Renesas Graphics Library Porting Layer Guide	LLWEB-10035995

2. Notation of Numbers and Symbols

This manual uses the following notation.

Binary	0bXXXXXXXX	(X=0 or 1)
Decimal	XXX	(X=0-9)
Hex	0xXXXXXXXX	(X=0-9,A-F)

3. List of Abbreviations and Acronyms

Abbreviation	Full Form
A0	Address bit 0
API	Application Programming Interface
CS	Chip Select
DOPI	Octa I/O DTR
DOS	DQS on STR mode
DQS	Data Strobe Signal
DTR	Double Transfer Rate
H/W	Hardware
MCLK	Memory Clock
OCTA	OctaBus
OPI	Octa I/O STR
RWW	Read-While-Write
SCLK	Serial Clock
SPI	Single I/O STR
STR	Single Transfer Rate

Table of Contents

1.Overview

1.1 Feature and Scope

The OCTA driver is a driver stack, which enables an abstract access to OctaRAM or OctaFlash memory. The abstraction shall simplify the usage by the application developer and also make it possible to use the same API for different hardware.

1.2 Component Structure

The component structure of OCTA is shown in ***Figure 1-1.***

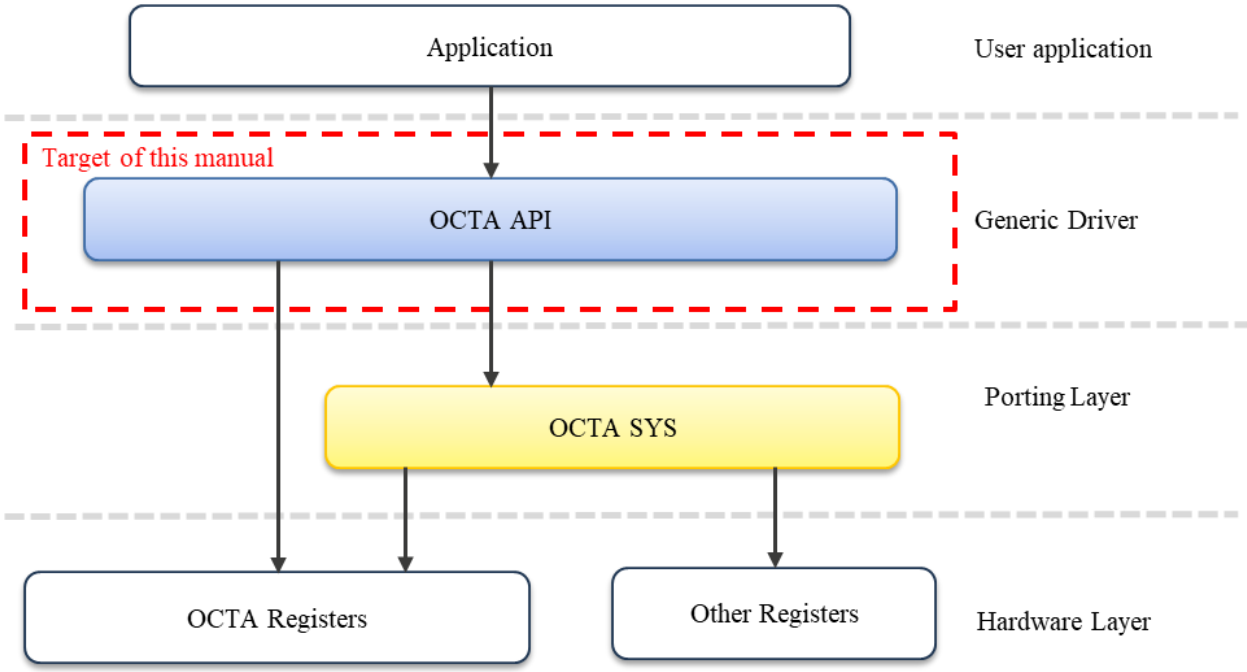

Figure 1-1 Component Structure

For the details of the API, please refer to ***Chapter 4.***

2.Basic Specification

2.1 Summary Specification

The summary of specification is described in *Table 2-1.*

Table 2-1 Summary Specification

Items	Description
Target LSI	RH850/D1M1-V2, RH850/D1M1A
Main Feature	• Number of connected devices o Up to two OctaRAM / OctaFlash memory per unit can be connected. Two device channels can be configured as: - OctaFlash only - OctaRAM only - OctaFlash / OctaRAM • Data bus width o Octa Flash - 1 bit / 8 bits o Octa RAM - 8 bits • Data transfer mode o Octa Flash - Single I/O STR (Single Transfer Rate), 1 bit per cycle mode. - Octa I/O STR (Single Transfer Rate), 8 bit per cycle mode. - Octa I/O DTR (Double Transfer Rate), 16 bit per cycle mode. o Octa RAM - Octa I/O DTR (Double Transfer Rate), 16 bit per cycle mode. • Operating mode o External address space mode o Manual mode
Semaphore / Mutex	N/A. This can be implemented with porting layer.
Interrupts	N/A.

2.2 Reserved Word

OCTA uses the following prefixes for avoiding confusion from other software. Prefixes of OCTA is described in *Table 2-2.*

Table 2-2 Prefixes

Prefix	Description
R_OCTA_*	Prefix for OCTA Module
r_octa_*	

2.3 Interrupt Handler List

None.

2.4 Error Handling

2.4.1 Return code

OCTA driver has 5 types of error codes.

2.4.1.1 Parameter level

Following errors occur by a cause such as abnormality of parameter. In this case, please set valid parameter again.

- R_OCTA_ERR_PARAM_INCORRECT
- R_OCTA_ERR_RANGE_UNIT
- R_OCTA_ERR_RANGE_PARAM

2.4.1.2 Timing level

Following errors occur by a cause such as abnormality of execution timing. In this case, please call again after changing to valid state or timing.

- R_OCTA_ERR_NOT_ACCEPTABLE
- R_OCTA_ERR_COMMAND
- R_OCTA_ERR_LATENCY
- R_OCTA_ERR_PROTECTED
- R_OCTA_ERR_ABORTED
- R_OCTA_ERR_TIMEOUT

2.4.1.3 System level

Following errors occur by a cause such as OS dependent error (e.g. system call error, resource shortage). In this case, please do recovery processing from a system layer, because this status cannot be restored only in this library.

- R_OCTA_ERR_FATAL_OS

2.4.1.4 Hardware level

Following errors occur when unexpected error occurs internally. In this case, please reset the RH850/D1x device.

- R_OCTA_ERR_NG
- R_OCTA_ERR_FATAL_HW

2.4.1.5 Device level

Following errors occur when the function is not supported with target device. In this case, please skip the function call.

- R_OCTA_ERR_DEVICE

2.5 State Transition

Each OCTA unit has following status.

Table 2-3 OCTA unit State Details

No.	State Name	Description
(1)	Uninitialized	Specifies that the OCTA driver is not initialized.
(2)	Initialized	Specifies that the OCTA driver is initialized.
(3)	Idle	Specifies that Manual mode is enabled.
(4)	Executing	Specifies that External address space mode is enabled.

The image describes state transition.
*1 : Only Octa Flash is executable.

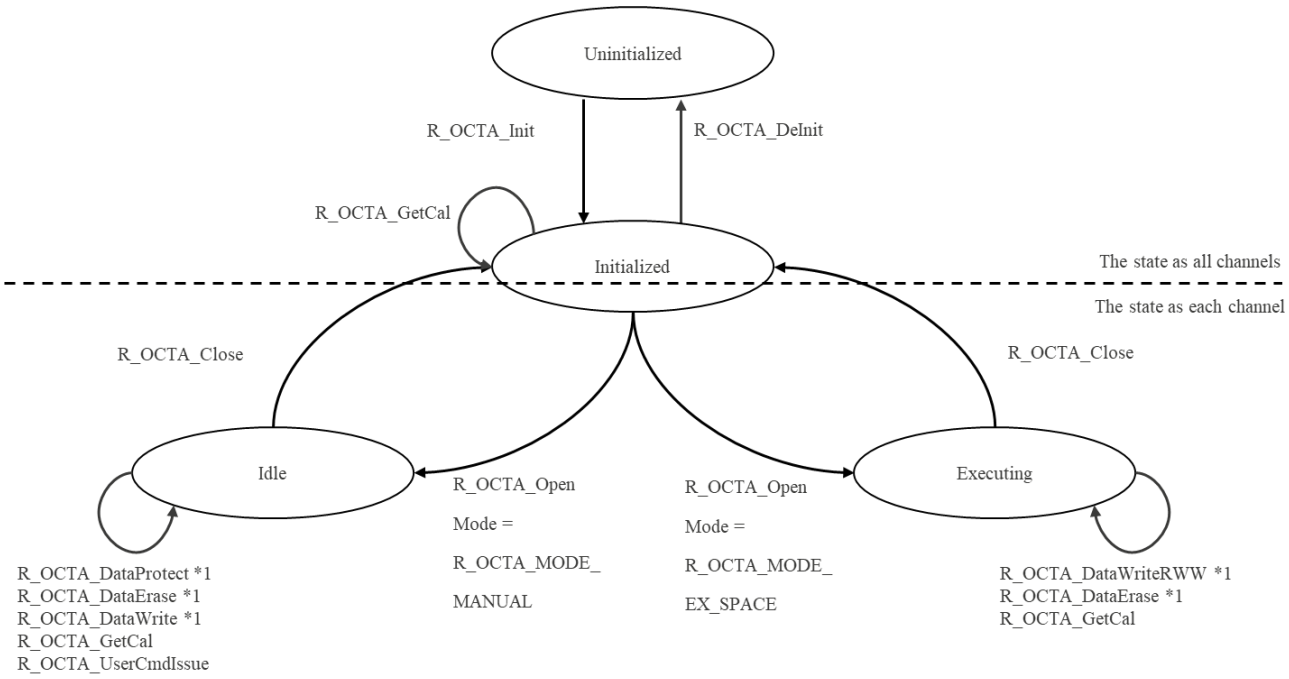

Figure 2-1 State Transition Diagram of OCTA driver

Table 2-4 State Transition Table of OCTA unit

Function Name	State			
	Uninitialized	Initialized	Idle	Executing
R_OCTA_Init	OK	NG	NG	NG
R_OCTA_DeInit	NG	OK	NG	NG
R_OCTA_Open	NG	OK	NG	NG
R_OCTA_Close	NG	NG	OK	OK
R_OCTA_DataProtect	NG	NG	OK	NG
R_OCTA_DataErase	NG	NG	OK	OK
R_OCTA_DataWrite	NG	NG	OK	NG
R_OCTA_DataWriteRWW	NG	NG	NG	OK
R_OCTA_UserCmdIssue	NG	NG	OK	NG
R_OCTA_GetCal	NG	OK	OK	OK
R_OCTA_VersionStringGet	OK	OK	OK	OK

3.Function Description

3.1 Fundamental Concepts

3.1.1 OCTA unit

RH850/D1x device has the following number of units of the OCTA.

Table 3-1 Number of units

Feature	RH850/D1x Device Name
	D1M1-V2, D1M1A
OCTA Units	1

Almost OCTA API functions have the argument “Unit”.
User specifies the OCTA H/W unit number to be controlled. The range is only 0.

3.1.2 OCTA channel

RH850/D1x device has the following number of channels of the OCTA.

Table 3-2 Number of channels

Feature	RH850/D1x Device Name
	D1M1-V2, D1M1A
OCTA channels	2
channel indexes	channel 0. channel 1

Almost OCTA API functions have the argument “Channel”.
User specifies the OCTA H/W channel number to be controlled. The range is 0 - 1.

3.1.3 System Configuration

This configuration has selected 8 bits data bus width. Then, SIO[7:0]_IN / SIO[7:0]_OUT pins are either the input pins or the output pins.

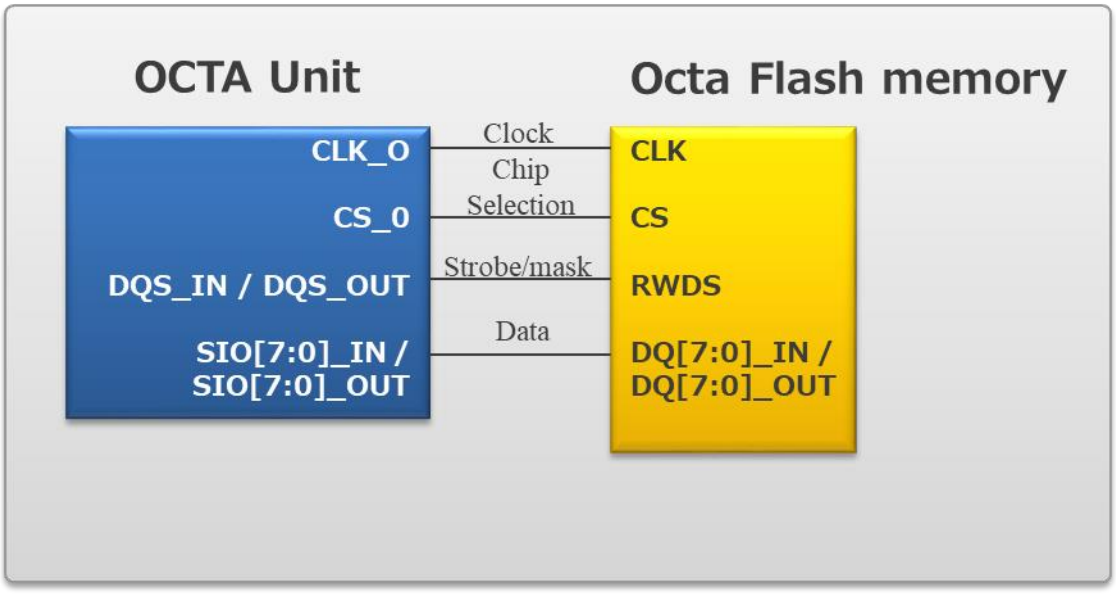

Figure 3-1 System Configuration

3.1.4 Operating Mode

The OCTA driver has two operating modes: external address space read mode and manual mode.

3.1.4.1 External address space mode

This mode is responsible for handling memory-map read/write operations. The whole space of ram memory is logically mapped to the master's address map and can be read or written directly. The flash memory can be read directly, and can be written by using Read-While-Write (RWW) function. RWW means read data one bank while another bank is programing or erasing.

In External address space mode, up to two OctaRAM / OctaFlash memory per unit can be connected. The Octa RAM / Octa Flash memory and OctaBus to be connected are assigned in the memory-map space.

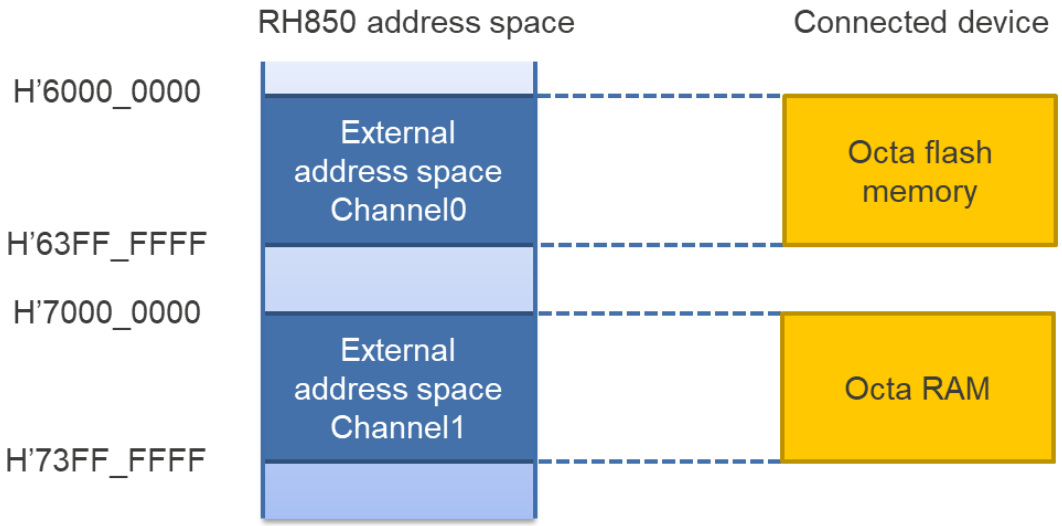

Figure 3-2 Address map (example for channel0 = Octa Flash, channel1 = Octa RAM)

3.1.4.2 Manual mode

Desired command accesses to the OctaRAM or OctaFlash memory are possible.

Note: The octa flash memory has protection function which prohibits writing and erasing. The control method of the protection function is different depending on octa flash memory.

3.1.5 Dependence command of the Octa flash / Octa RAM

The OCTA driver can support Octa Flash / Octa RAM memory if it is compatible with Macronix OctaFlash / OctaRAM family. To support various Octa Flash / Octa RAM memories, the control commands, which depended on the Octa Flash / Octa RAM memory, must be set. The command depending on the Octa Flash / Octa RAM memory sets it to r_octa_Command_t structure and it must be handed to the Open function.

3.2 Using the API

3.2.1 Initialization / De-Initialization

R_OCTA_Init initializes the driver and the hardware as far as necessary. The Unit parameter holds a number that specifies the OCTA unit number being initialized. R_OCTA_DeInit function de-initializes the driver and the hardware as far as necessary.

3.2.2 Octa RAM - External address space mode

The sample code of External address space mode for Octa RAM connected to channel 1 is shown below. After setting completion, the Octa RAM is assigned to linear memory space from 0x70000000 to 0x707FFFFF.

```
void SampleExternalAddressMode(void)
{
  uint32_t channel = 1;
  r_octa_Config_t config;
  uint32_t* read_pointer;
  uint32_t* write_pointer;
  uint32_t i;

  /* Init */
  R_OCTA_Init(LOC_OCTA_UNIT);

  /* Open */
  config.DeviceType = R_OCTA_DEVICE_RAM;
  config.OpeMode = R_OCTA_MODE_EX_SPACE;
  config.DataTransferMode = R_OCTA_MODE_DOPI;
  config.AddressMode = R_OCTA_ADDRESS_32BIT;
  config.MemorySize = 8 * 1024 * 1024; /* Byte */
  config.SectorSize = 0;
  config.PageSize = 0;
  config.Command = &r_octacdb_RamCmdTabl;
  config.RelaxSize = 0;
  config.PreCycle = R_OCTA_PRECYCLE_OFF;
  config.DQSDelay.EnableCnt = 6;
  config.DQSDelay.Delay = 0x17;
  config.CalAddress = 0xFFFFFFFF;

  R_OCTA_Open(LOC_OCTA_UNIT, channel, &config);

  /* Read & Write sample */
  read_pointer = (uint32_t*)0x70000000;
  write_pointer = (uint32_t*)0x70100000;
  for (i = 0; i < 100; i++)
  {
    write_pointer[i] = read_pointer[i];
  }
}
```

3.2.3 Octa Flash - External address space mode

The sample code of External address space mode for Octa Flash connected to channel 0 is shown below. After setting completion, the Octa Flash is assigned to linear memory space from 0x60000000 to 0x63FFFFFF.

```
void SampleExternalAddressMode(void)
{
  uint32_t channel = 0;
  r_octa_Config_t config;
  uint32_t* read_pointer;
  uint32_t* write_pointer;
  uint32_t i;

  /* Init */
  R_OCTA_Init(LOC_OCTA_UNIT);

  /* Open */
  config.DeviceType = R_OCTA_DEVICE_FLASH;
  config.OpeMode = R_OCTA_MODE_EX_SPACE;
  config.DataTransferMode = R_OCTA_MODE_DOPI;
  config.AddressMode = R_OCTA_ADDRESS_32BIT;
  config.MemorySize = 64 * 1024 * 1024; /* Byte */
  config.SectorSize = 4 * 1024; /* Byte */
  config.PageSize = 256; /* Byte */
  config.Command = &r_octacdb_FlashCmdTbl;
  config.RelaxSize = 0;
  config.PreCycle = R_OCTA_PRECYCLE_OFF;
  config.DQSDelay.EnableCnt = 5;
  config.DQSDelay.Delay = 0x17;
  config.CalAddress = 0xFFFFFFFF;

  R_OCTA_Open(LOC_OCTA_UNIT, channel, &config);

  /* Read & Write sample */
  read_pointer = (uint32_t*)0x60000000;
  write_pointer = (uint32_t*)0x60100000;
  for (i = 0; i < 100; i++)
  {
    write_pointer[i] = read_pointer[i];
  }
}
```

3.2.4 Octa Flash - Manual mode

The sample code of manual mode for Octa Flash connected to channel 0 is shown below.

```
void SampleManualMode(void)
{
  uint32_t channel = 0;
  r_octa_Config_t config;
  uint8_t data[100];
  uint32_t i;
  uint32_t byte_size = 100;
  uint32_t byte_addr = 0;

  /* Init */
  R_OCTA_Init(LOC_OCTA_UNIT);

  /* Open */
  config.DeviceType = R_OCTA_DEVICE_FLASH;
  config.OpeMode = R_OCTA_MODE_MANUAL;
  config.DataTransferMode = R_OCTA_MODE_DOPI;
  config.AddressMode = R_OCTA_ADDRESS_32BIT;
  config.MemorySize = 64 * 1024 * 1024; /* Byte */
  config.SectorSize = 4 * 1024; /* Byte */
  config.PageSize = 256; /* Byte */
  config.Command = &r_octacdb_FlashCmdTbl;
  config.RelaxSize = 0;
  config.PreCycle = R_OCTA_PRECYCLE_OFF;
  config.DQSDelay.EnableCnt = 5;
  config.DQSDelay.Delay = 0x17;
  config.CalAddress = 0xFFFFFFFF;

  R_OCTA_Open(LOC_OCTA_UNIT, channel, &config);

  /* Make write data */
  for (i = 0; i < 100; i++) {
    data[i] = i;
  }

  /* Unprotect */
  R_OCTA_DataProtect(LOC_OCTA_UNIT, channel, R_OCTA_MODE_UNPROTECT);
  /* Erase sector */
  R_OCTA_DataErase(LOC_OCTA_UNIT, channel, byte_addr, byte_size);
  /* Write data */
  R_OCTA_DataWrite(LOC_OCTA_UNIT, channel, byte_addr, data, byte_size);
  /* Protect */
  R_OCTA_DataProtect(LOC_OCTA_UNIT, channel, R_OCTA_MODE_PROTECT);
}
```

3.2.5 Calibration

OCTA driver requires DQS delay value as argument of R_OCTA_Open. DQS delay value depends on the connected Octa Flash / Octa RAM memory, transfer mode (SPI, OPI, DOPI), board design and so on. It might be useful to run the calibration routines to determine the best value for the calibration.

If you want to be executed the calibration, please calibrate with data read / data write after calling the R_OCTA_Open function by setting calibration start address to CalAddress, DQS delay value to DQSDelay.Delay.

3.3 Device difference

The following table shows the function differences depending on the device.

Table 3-3 APIs supported by OCTA driver

Feature	RH850/D1x Device Name			
	D1L2(H)	D1M1(H)	D1M1-V2, D1M1A	D1M2(H)
All API of OCTA driver	No	No	Full	No

3.4 Header File List

Table 3-4 Header File List

No.	Header File Name	Description
(1)	r_octa_api.h	Header file for OCTA API.
(2)	r_typedefs.h	Header file for predefined data types.

4.Functions

4.1 Function List

This section describes about the OCTA API functions which are in *Table 4-1* and executable state of each function is described in the specification of each function.

Table 4-1 List of OCTA API Functions

Function Name	Purpose
R_OCTA_Init	This function initializes the OCTA driver.
R_OCTA_DeInit	This function de-initializes the OCTA driver.
R_OCTA_Open	This function opens the OCTA driver.
R_OCTA_Close	This function closes the OCTA driver.
R_OCTA_DataProtect	This function sets the protection mode of the Octa flash memory.
R_OCTA_DataErase	This function erases the data in the Octa flash memory.
R_OCTA_DataWriteRWW	This function writes data to the Octa Flash memory by RWW (Read-While-Write) function.
R_OCTA_DataWrite	This function writes data to the Octa Flash memory.
R_OCTA_UserCmdIssue	This function receives the DQS (Data Strobe Signal) delay value.
R_OCTA_GetCal	This function executes user command sequence.
R_OCTA_VersionStringGet	This function returns the version string of this OCTA driver.
R_OCTA_MacroVersionGet	This function returns the major and minor version of the H/W macro.

4.2 OCTA API Functions

This chapter describes the application interface functions, which are required for general use of the driver.

4.2.1 Basic functions

The section describes driver functions, which are required for general use of the driver, but which are related to a specific functionality of the macro itself.

4.2.1.1 R_OCTA_Init

Function Prototypes

```
r_octa_Error_t R_OCTA_Init(const unit32_t     Unit)
```

Input Parameter

Table 4-2 Input parameter of R_OCTA_Init

Parameter	Description
Unit	Specifies the OCTA unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_DEVICE	- OCTA driver is not applicable to target device.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

Description

This function initializes the OCTA driver.
This function calls R_OCTA_Sys_Init to initialize environment-dependent setting.
If the function successfully executes, the return code will be R_OCTA_ERR_OK and the state will be in the Initialize state.

Reentrancy

Non-reentrant
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.2 R_OCTA_DeInit

Function Prototypes

```
r_octa_Error_t R_OCTA_DeInit(const uint32_t        Unit)
```

Input Parameter

Table 4-3 Input parameter of R_OCTA_DeInit

Parameter	Description
Unit	Specifies the OCTA unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

Description

This function de-initializes the OCTA driver.
This function calls R_OCTA_Sys_DeInit to de-initialize environment-dependent setting.
If the function successfully executes, the return code will be R_OCTA_ERR_OK and the state will be in the Uninitialize state.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.3 R_OCTA_Open

Function Prototypes

```
r_octa_Error_t R_OCTA_Open(const uint32_t                Unit,
                           const uint32_t                Channel,
                           const r_octa_Config_t * const Config)
```

Input Parameter

Table 4-4 Input parameter of R_OCTA_Open

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
Config	Pointer to the r_octa_Config_t structure.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error has occurred.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_RANGE_UNIT	- The unit-number is the outside of the range.
R_OCTA_ERR_RANGE_PARAM	- A parameter is out of range.
R_OCTA_ERR_FATAL_OS	- Fatal Error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.
R_OCTA_ERR_PROTECTED	- A process is aborted because of memory protection.
R_OCTA_ERR_COMMAND	- A command is not supported.
R_OCTA_ERR_LATENCY	- A latency value is invalid.

Description

This function opens the OCTA driver.
This function opens the OCTA driver with the operating mode and device type that is specified.
This function opens for each specified channel device.

In case of external address space mode, Read command and Write command are set to H/W and it enables access from memory-map space. For Octa RAM can be read / write directly.
For Octa flash memory can be read directly. Read command and Write command to be set are different depending on the data transfer mode.

In case of manual mode, the function enables access the status/configuration register for Octa RAM/flash memory. And enables Write access to memory for Octa flash memory.

If the function successfully executes, the return code will be R_OCTA_ERR_OK. and the status will be changed to Idle if R_OCTA_MODE_MANUAL is specified. The status will be changed to Executing if R_OCTA_MODE_EX_SPACE is specified.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t
r_octa_Config_t

4.2.1.4 R_OCTA_Close

Function Prototypes

```
r_octa_Error_t R_OCTA_Close(const uint32_t Unit,
                            const uint32_t Channel)
```

Input Parameter

Table 4-5 Input parameter of R_OCTA_Close

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.

Description

This function closes the OCTA driver.
In case of external address space mode, this function disables access from the memory-map space..
If the function successfully executes, the return code will be R_OCTA_ERR_OK and the state will be in the Initialize state.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.5 R_OCTA_DataProtect

Function Prototypes

```
r_octa_Error_t R_OCTA_DataProtect(const uint32_t               Unit,
                                  const uint32_t               Channel,
                                  const r_octa_ProtectionMode_t Mode)
```

Input Parameter

Table 4-6 Input parameter of R_OCTA_DataProtect

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
Mode	Protection mode.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter is incorrect.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_RANGE_UNIT	- The unit-number is the outside of the range.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_FATAL_OS	- Fatal Error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.

Description

This function sets the protection mode of the Octa flash memory.
The write and erase access to Octa Flash memory is prohibited if protection is set.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t
r_octa_ProtectionMode_t

4.2.1.6 R_OCTA_DataErase

Function Prototypes

```
r_octa_Error_t R_OCTA_DataErase(const uint32_t       Unit,
                                const uint32_t       Channel,
                                const uint32_t       Addr,
                                const uint32_t       Size )
```

Input Parameter

Table 4-7 Input parameter of R_OCTA_DataErase

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
Addr	Erase start address of the Octa flash memory.
Size	Data size to erase.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.
R_OCTA_ERR_COMMAND	- A command is not supported.
R_OCTA_ERR_PROTECTED	- A process is aborted because of memory protection.
R_OCTA_ERR_TIMEOUT	- Status polling is timeout.

Description

This function erases the data in the Octa flash memory.
This function erases the data in a unit of sector. Therefore, this function erases data of the sector including the size from the address.

This function erases the sectors in following range.

Start sector	Sector that Addr is belonged.
End sector	Sector that (Addr + Size - 1) is belonged.

In the case of Octa flash memory compatible with RWW (Read-While-Write), it can be executed while reading into the memory map space in external address space mode.
This function has the possibility that the processing takes time. Therefore, R_OCTA_Sys_Relax is sometimes executed.

If the function successfully executes, the return code will be R_OCTA_ERR_OK.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.7 R_OCTA_DataWriteRWW

Function Prototypes

```
r_octa_Error_t R_OCTA_DataWriteRWW(const uint32_t       Unit,
                                   const uint32_t       Channel,
                                   const uint32_t       Addr,
                                   const uint8_t*       Buf,
                                   const uint32_t       Size)
```

Input Parameter

Table 4-8 Input parameter of R_OCTA_DataWriteRWW

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
Addr	The parameter specifies the write address of the Octa flash memory. If the data transfer mode is R_OCTA_MODE_DOPI, Addr given must be even address.
Buf	This is a pointer to the buffer stored write data.
Size	The parameter specifies the data byte-size to write. If the data transfer mode is R_OCTA_MODE_DOPI, Size given must be even data size.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.
R_OCTA_ERR_COMMAND	- A command is not supported.
R_OCTA_ERR_PROTECTED	- A process is aborted because of memory protection.
R_OCTA_ERR_TIMEOUT	- Status polling is timeout

Description

This function writes data to the Octa Flash memory by RWW (Read-While-Write) function.
RWW function means read data one bank while another bank is programing or erasing.

This function can be executed for Octa flash memory with supporting RWW. If Octa flash memory is not supported RWW, does not guarantee the operation.

This function is executed WriteBufInitial, WriteBufContinue and WriteBufConfirm command. These commands to be set are different depending on the data transfer mode.

In DTR OPI, the starting address given must be even address (A0=0) and data byte number must be even.
This function has the possibility that the processing takes time. Therefore, R_OCTA_Sys_Relax is sometimes executed.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.8 R_OCTA_DataWrite

Function Prototypes

```
r_octa_Error_t R_OCTA_DataWrite(const uint32_t       Unit,
                                const uint32_t       Channel,
                                const uint32_t       Addr,
                                const uint8_t*       Buf,
                                const uint32_t       Size)
```

Input Parameter

Table 4-9 Input parameter of R_OCTA_DataWrite

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
Addr	The parameter specifies the write address of the Octa flash memory. This parameter aligns in the page size of the Octa flash memory. If the data transfer mode is R_OCTA_MODE_DOPI, Addr given must be even address.
Buf	This is a pointer to the buffer stored write data.
Size	The parameter specifies the data byte-size to write. If the data transfer mode is R_OCTA_MODE_DOPI, Size given must be even data size.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.
R_OCTA_ERR_COMMAND	- A command is not supported.
R_OCTA_ERR_PROTECTED	- A process is aborted because of memory protection.
R_OCTA_ERR_TIMEOUT	- Status polling is timeout

Description

This function writes data to the Octa Flash memory.
In order to write data to Octa Flash, the data of the sector must have been erased previously.

This function executes Write command. Write commands to be set are different depending on the data transfer mode.
In DTR OPI, the starting address given must be even address (A0=0) and data byte number must be even.
This function has the possibility that the processing takes time. Therefore, R_OCTA_Sys_Relax is sometimes executed.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.9 R_OCTA_UserCmdIssue

Function Prototypes

```
r_octa_Error_t R_OCTA_UserCmdIssue(const uint32_t                      Unit,
                                   const uint32_t                      Channel,
                                   const r_octa_CmdTransaction_t* const CmdTransaction,
                                         uint8_t*                 const Buf,
                                   const uint32_t                      BufSize)
```

Input Parameter

Table 4-10 Input parameter of R_OCTA_UserCmdIssue

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.
CmdTransaction	Command sequence.
BufSize	Byte-size of data that can be stored to Buf.

Input-Output Parameter

None

Output Parameter

Table 4-11 Output parameter of R_OCTA_UserCmdIssue

Parameter	Description
Buf	This is a pointer to the Buffer to store the read/write data. If read transaction is not existed in command sequence, please set to R_NULL.

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	- Fatal error has occurred at H/W.
R_OCTA_ERR_COMMAND	- A command is not supported.

Description

This function executes user command sequence.
This function executes when user read / write the status/configuration register and read device ID for Octa RAM/flash memory.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t
r_octa_CmdTransaction_t

4.2.1.10 R_OCTA_GetCal

Function Prototypes

```
r_octa_Error_t R_OCTA_GetCal(const uint32_t         Unit,
                             const uint32_t         Channel,
                                   uint8_t* const   Delay)
```

Input Parameter

Table 4-12 Input parameter of R_OCTA_GetCal

Parameter	Description
Unit	Specifies the OCTA unit number.
Channel	Device Channel number.

Input-Output Parameter

None

Output Parameter

Table 4-13 Output parameter of R_OCTA_GetCal

Parameter	Description
Delay	Pointer to the DQS delay value.

Return Codes

R_OCTA_ERR_OK	- No error occurred.
R_OCTA_ERR_RANGE_UNIT	- The unit-number was outside the range.
R_OCTA_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_OCTA_ERR_NOT_ACCEPTABLE	- A function was called in an incorrect state.
R_OCTA_ERR_RANGE_PARAM	- A parameter is the outside of the range.
R_OCTA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

Description

This function receives the DQS (Data Strobe Signal) delay value.
The DQS delay value uses to adjusting the calibration.
This function uses to get the DQS delay value after executing the calibration. The calibration is adjusted by actually reading / writing to the connected OCTA RAM / flash memory while changing the DQS delay value.
The DQS delay value was received in this function is sets to DQSDelay.Delay member of r_octa_Config_t structure of argument for R_OCTA_Open function.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_OCTA_Sys_Lock
- R_OCTA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.1.11 R_OCTA_VersionStringGet

Function Prototypes

```
const uint8_t* R_OCTA_VersionStringGet(void)
```

Input Parameter

None

Input-Output Parameter

None

Output Parameter

None

Return Codes

Version string.

Description

This function returns version string of the OCTA driver.

Reentrancy

Reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

None

4.2.1.12 R_OCTA_MacroVersionGet

Function Prototypes

```
r_octa_Error_t R_OCTA_MacroVersionGet(uint32_t * const Major,
                                      uint32_t * const Minor)
```

Input Parameter

None

Input -Output Parameter

None

Output Parameter

Table 4-14 Output parameter of R_OCTA_MacroVersionGet

Parameter	Description
Major	The major version.
Minor	The minor version.

Return Codes

R_OCTA_ERR_OK	- No error has occurred.
R_OCTA_ERR_PARAM_INCORRECT	- Either parameter Major or parameter Minor was R_NULL

Description

This function returns the major and minor version of the H/W macro.

Reentrancy

Reentrant**.**

Sync/Async

Synchronous

Call from Interrupt

Prohibited**.**

Preconditions

See ***Table 2-4*** about OCTA unit status conditions.

See also

r_octa_Error_t

4.2.2 Interrupt functions

None.

5.Types

5.1 Basic Types

This section shows the basic types used on this library.

Table 5-1 Basic type

Types	Definition		Basic types
char_t	typedef char	char_t	signed char
int8_t	typedef signed char	int8_t	signed char
int16_t	typedef signed short	int16_t	signed short
int32_t	typedef signed int	int32_t	signed int
int64_t	typedef signed long long	int64_t	signed long long
uint8_t	typedef unsigned char	uint8_t	unsigned char
uint16_t	typedef unsigned short	uint16_t	unsigned short
uint32_t	typedef unsigned int	uint32_t	unsigned int
uint64_t	typedef unsigned long long	uint64_t	unsigned long long
float32_t	typedef float	float32_t	float
float64_t	typedef double	float64_t	double

5.2 Definition

This section shows the definitions used in OCTA API.

Table 5-2 Definition of OCTA API

Name	Description
R_OCTA_VERSION_HI	MSB byte of the version information. It is major version information. This value is changed with release version.
R_OCTA_VERSION_LO	LSB byte of the version information. It is miner version information. This value is changed with release version.

Table 5-3 Definition of Command Option

Name	Description
R_OCTA_CMD_WRITE	Use Write transaction.
R_OCTA_CMD_READ	Use Read transaction.
R_OCTA_CMD_SIZE_NONE	Size is none.
R_OCTA_CMD_SIZE_8	8 bitwise access.
R_OCTA_CMD_SIZE_16	16 bitwise access.
R_OCTA_CMD_SIZE_MASK	Mask for size get.
R_OCTA_CMD_WRITE_NONE	Use Write transaction. No Write size. Only command issue.
R_OCTA_CMD_WRITE8	Use Write transaction. 8 bitwise access..
R_OCTA_CMD_WRITE16	Use Write transaction. 16 bitwise access.
R_OCTA_CMD_READ8	Use Read transaction. 8 bitwise access..
R_OCTA_CMD_READ16	Use Read transaction. 16 bitwise access.
R_OCTA_CMD_NONE	If the command is not supported, please set to this flag.

5.3 Enumerated Type

This section shows the enumerated types used in OCTA API Function.

5.3.1 r_octa_Error_t

Description

OCTA driver error code.
If an error occurs, these enumerations give information about the reason.

Definition

```
typedef enum
{
    R_OCTA_ERR_OK = 0,
    R_OCTA_ERR_NG,
    R_OCTA_ERR_PARAM_INCORRECT,
    R_OCTA_ERR_RANGE_UNIT,
    R_OCTA_ERR_RANGE_PARAM,
    R_OCTA_ERR_NOT_ACCEPTABLE,
    R_OCTA_ERR_DEVICE,
    R_OCTA_ERR_FATAL_OS,
    R_OCTA_ERR_FATAL_HW,
    R_OCTA_ERR_PROTECTED,
    R_OCTA_ERR_TIMEOUT,
    R_OCTA_ERR_COMMAND,
    R_OCTA_ERR_LATENCY
} r_octa_Error_t;
```

Table 5-4 Enumerator of r_octa_Error_t

Name	Description
R_OCTA_ERR_OK	No error occurred.
R_OCTA_ERR_NG	An error has occurred, but no specific error code is defined for it.
R_OCTA_ERR_PARAM_INCORRECT	A parameter provided to a function was incorrect.
R_OCTA_ERR_RANGE_UNIT	The unit-number was outside the range.
R_OCTA_ERR_RANGE_PARAM	Parameter is the outside the range.
R_OCTA_ERR_NOT_ACCEPTABLE	A function was called in an incorrect state.
R_OCTA_ERR_DEVICE	OCTA driver is not applicable to target d1x device.
R_OCTA_ERR_FATAL_OS	Fatal error has occurred at OS interface.
R_OCTA_ERR_FATAL_HW	Fatal error has occurred at H/W.
R_OCTA_ERR_PROTECTED	A process is aborted because of memory protection.
R_OCTA_ERR_TIMEOUT	Status polling is timeout.
R_OCTA_ERR_COMMAND	A command is not supported.
R_OCTA_ERR_LATENCY	A latency value is invalid.

See also

None

5.3.2 r_octa_DeviceType_t

Description

This type describes the control device type.

Definition

```
typedef enum
{
    R_OCTA_DEVICE_FLASH = 0,
    R_OCTA_DEVICE_RAM
} r_octa_DeviceType_t;
```

Table 5-5 Enumerator of r_octa_DeviceType_t

Name	Description
R_OCTA_DEVICE_FLASH	Octa Flash.
R_OCTA_DEVICE_RAM	Octa RAM.

See also

None

5.3.3 r_octa_OperatingMode_t

Description

This type describes the operating mode.

Definition

```
typedef enum
{
    R_OCTA_MODE_EX_SPACE = 0,
    R_OCTA_MODE_MANUAL
} r_octa_OperatingMode_t;
```

Table 5-6 Enumerator of r_octa_OperatingMode_t

Name	Description
R_OCTA_MODE_EX_SPACE	External address space mode. Read/Write access to memory-mapped space is possible for Octa RAM/flash memory.
R_OCTA_MODE_MANUAL	Manual mode. In case of Octa flash, Read/Write access to register area and Write access to memory is possible. In case of Octa RAM, Read/Write access to register area is possible.

See also

None

5.3.4 r_octa_DataTransferMode_t

Description

This type describes the data transfer mode.

Definition

```
typedef enum
{
    R_OCTA_MODE_SPI = 0,
    R_OCTA_MODE_OPI,
    R_OCTA_MODE_DOPI
} r_octa_DataTransferMode_t;
```

Table 5-7 Enumerator of r_octa_DataTransferMode_t

Name	Description
R_OCTA_MODE_SPI	Single I/O STR (Single Transfer Rate), 1 bit per cycle mode. In case of Octa Flash, Read/Write to memory/register access is possible. In case of Octa RAM, it is not supported.
R_OCTA_MODE_OPI	Octa I/O STR (Single Transfer Rate), 8 bit per cycle mode. In case of Octa Flash, Read/Write to memory/register access is possible. In case of Octa RAM, it is not supported.
R_OCTA_MODE_DOPI	Octa I/O DTR (Double Transfer Rate), 16 bit per cycle mode. In case of Octa Flash, Read/Write to memory/register access is possible. In case of Octa RAM, Read/Write to memory/register access is possible.

See also

None

5.3.5 r_octa_AddressMode_t

Description

This type describes the format of address output to the Octa flash memory. In case of Octa RAM, please set to R_OCTA_ADDRESS_32BIT.

Definition

```
typedef enum
{
    R_OCTA_ADDRESS_24BIT = 0,
    R_OCTA_ADDRESS_32BIT
} r_octa_AddressMode_t;
```

Table 5-8 Enumerator of r_octa_AddressMode_t

Name	Description
R_OCTA_ADDRESS_24BIT	24 bits address output.
R_OCTA_ADDRESS_32BIT	32 bits address output.

See also

None

5.3.6 r_octa_ProtectionMode_t

Description

This type describes the protection mode of the Octa flash memory.

Definition

```
typedef enum
{
    R_OCTA_MODE_PROTECT = 0,
    R_OCTA_MODE_UNPROTECT
} r_octa_ProtectionMode_t;
```

Table 5-9 Enumerator of r_octa_ProtectionMode_t

Name	Description
R_OCTA_MODE_PROTECT	Protection mode.
R_OCTA_MODE_UNPROTECT	Un-protection mode.

See also

None

5.3.7 r_octa_Reg_t

Description

This type describes the register of the Octa RAM/flash memory.

Definition

```
typedef enum
{
    R_OCTA_STATUS_REG = 0,
    R_OCTA_CONFIG_REG,
    R_OCTA_CONFIG2_REG,
    R_OCTA_NONE_REG
} r_octa_Reg_t;
```

Table 5-10 Enumerator of r_octa_Reg_t

Name	Description
R_OCTA_STATUS_REG	Status Register.
R_OCTA_CONFIG_REG	Configuration Register.
R_OCTA_CONFIG2_REG	Configuration2 Register.
R_OCTA_NONE_REG	Un used.

See also

None

5.3.8 r_octa_PreCycle_t

Description

This type describes the pre-cycle mode of the Octa RAM/flash memory. This type is used when the data transfer mode is R_OCTA_MODE_DOPI.

Definition

```
typedef enum
{
    R_OCTA_PRECYCLE_OFF = 0,
    R_OCTA_PRECYCLE_ON,
} r_octa_PreCycle_t;
```

Table 5-11 Enumerator of r_octa_PreCycle_t

Name	Description
R_OCTA_PRECYCLE_OFF	pre-cycle mode off.
R_OCTA_PRECYCLE_ON	pre-cycle mode on.

See also

None

5.3.9 r_octa_LowPeriod_t

Description

This type describes the setting of the period for issuing command when the device chip select pull down. (from the last SCLK low to CS high)
Its unit is MCLK cycle.
In case of DOPI mode, Actual Cycle is set value - 0.5 clk. If Low=3 is set, actual delay is 2.5 clock (for DOPI mode). (e.g. It is described as tSLCH in data sheet of MX25LW51245G.)

Definition

```
typedef enum
{
    R_OCTA_LOWPERIOD_2 = 0,
    R_OCTA_LOWPERIOD_3,
    R_OCTA_LOWPERIOD_4,
    R_OCTA_LOWPERIOD_5
} r_octa_LowPeriod_t;
```

Table 5-12 Enumerator of r_octa_LowPeriod_t

Name	Description
R_OCTA_LOWPERIOD_2	2 cycles.
R_OCTA_LOWPERIOD_3	3 cycles.
R_OCTA_LOWPERIOD_4	4 cycles.
R_OCTA_LOWPERIOD_5	5 cycles.

See also

None

5.3.10 r_octa_HighPeriod_t

Description

This type describes the setting of the period for the device chip select pull up after the command is finished. (from the last SCLK low to CS high)
Its unit is MCLK cycle.
In case of DOPI mode, Actual Cycle is set value - 0.5 clk. If High=3 is set, actual delay is 2.5 clock (for DOPI mode). (e.g. It is described as tCLSH in data sheet of MX25LW51245G.)

Definition

```
typedef enum
{
    R_OCTA_HIGHPERIOD_2 = 0,
    R_OCTA_HIGHPERIOD_3,
    R_OCTA_HIGHPERIOD_4,
    R_OCTA_HIGHPERIOD_5,
    R_OCTA_HIGHPERIOD_6,
    R_OCTA_HIGHPERIOD_7,
    R_OCTA_HIGHPERIOD_8,
    R_OCTA_HIGHPERIOD_9
} r_octa_HighPeriod_t;
```

Table 5-13 Enumerator of r_octa_HighPeriod_t

Name	Description
R_OCTA_HIGHPERIOD_2	2 cycles.
R_OCTA_HIGHPERIOD_3	3 cycles.
R_OCTA_HIGHPERIOD_4	4 cycles.
R_OCTA_HIGHPERIOD_5	5 cycles.
R_OCTA_HIGHPERIOD_6	6 cycles.
R_OCTA_HIGHPERIOD_7	7 cycles.
R_OCTA_HIGHPERIOD_8	8 cycles.
R_OCTA_HIGHPERIOD_9	9 cycles.

See also

None

5.3.11 r_octa_BetweenPeriod_t

Description

This type describes the setting of the period between two commands. (from CS high to the next CS low).
Its unit is MCLK cycle.
(e.g. It is described as tSHSL in data sheet of MX25LW51245G.)

Definition

```
typedef enum
{
    R_OCTA_BETWEENPERIOD_2 = 0,
    R_OCTA_BETWEENPERIOD_5,
    R_OCTA_BETWEENPERIOD_7,
    R_OCTA_BETWEENPERIOD_9,
    R_OCTA_BETWEENPERIOD_11,
    R_OCTA_BETWEENPERIOD_13,
    R_OCTA_BETWEENPERIOD_15,
    R_OCTA_BETWEENPERIOD_17
} r_octa_BetweenPeriod_t;
```

Table 5-14 Enumerator of r_octa_BetweenPeriod_t

Name	Description
R_OCTA_BETWEENPERIOD_2	2 cycles.
R_OCTA_BETWEENPERIOD_5	5 cycles.
R_OCTA_BETWEENPERIOD_7	7 cycles.
R_OCTA_BETWEENPERIOD_9	9 cycles.
R_OCTA_BETWEENPERIOD_11	11 cycles.
R_OCTA_BETWEENPERIOD_13	13 cycles.
R_OCTA_BETWEENPERIOD_15	15 cycles.
R_OCTA_BETWEENPERIOD_17	17 cycles.

See also

None

5.3.12 r_octa_StateType_t

Description

This type describes the state type of the register.

Definition

```
typedef enum
{
    R_OCTA_TYPE_WEL = 0,
    R_OCTA_TYPE_WIP,
    R_OCTA_TYPE_BP,
    R_OCTA_TYPE_PCM,
    R_OCTA_TYPE_DTM,
    R_OCTA_TYPE_DOS,
    R_OCTA_TYPE_DL,
} r_octa_StateType_t;
```

Table 5-15 Enumerator of r_octa_StateType_t

Name	Description
R_OCTA_TYPE_WEL	State of Write enable latch.
R_OCTA_TYPE_WIP	State of Write In Progress.
R_OCTA_TYPE_BP	State of Block Protect.
R_OCTA_TYPE_DTM	State of Data Transfer mode.
R_OCTA_TYPE_PCM	State of Pre-cycle mode.
R_OCTA_TYPE_DOS	State of DOS.
R_OCTA_TYPE_DL	State of Dummy Length.

See also

None

5.4 Structure Type

This section shows the structure used in OCTA API Function.

5.4.1 r_octa_CmdTransaction_t

Description

This type describes the Octa RAM/flash commands.

Definition

```
typedef struct
{
    uint16_t                    Cmd;
    uint32_t                    Address;
    uint8_t                     CmdLength;
    uint8_t                     AddressLength;
    uint8_t                     DummyLength;
    uint8_t                     DataLength;
    uint32_t                    OpeFlags;
} r_octa_CmdTransaction_t;
```

Table 5-16 Member of r_octa_CmdTransaction_t

Name	Description
Cmd	Command for Octa RAM/flash memory. In case of Octa flash memory, Cmd is set 1st byte for SPI register commands or 1st byte ~ 2nd byte for OPI register commands. In case of Octa RAM, Cmd is set command address. If the command is not supported, please set to 0xFFFF.
Address	Address of the command for Octa RAM/flash memory. In case of Octa flash memory, Address is set 3rd byte~6th byte for OPI register commands. In case of Octa RAM, Address is set Row address and Column address. If the address is not defined, please set to 0.
CmdLength	Command length of the command. Its unit is byte. If the command is not supported, please set to 0.
AddressLength	Address length of the command. Its unit is byte. If the address is not defined, please set to 0.
DummyLength	Dummy length of the command. Its unit is clock cycle. The range is 3 to 20. In case of Octa flash memory, Set Dummy Cycle. In case of Octa RAM, Set Latency Counter. If dummy length is not needed, please set to 0.
DataLength	Data length of the command. If data length is not needed, please set to 0.
OpeFlags	Operation Flags. Multiple flags can be set with '|'. In case of DOPI mode, if R_OCTA_CMD_SIZE_16 is set, the data gets in units of 2-bytes. See *Table 5-3*.

See also

None

5.4.2 r_octa_RegInfo_t

Description

This type describes register information of the Octa RAM/flash memory.

Definition

```
typedef struct
{
    r_octa_Reg_t          Reg;
    uint32_t              Address;
    uint8_t               AddressLength;
    uint16_t              BitPosition;
} r_octa_RegInfo_t;
```

Table 5-17 Member of r_octa_RegInfo_t

Name	Description
Reg	The register of the Octa RAM/flash memory
Address	Address of the command for Octa RAM/flash memory. In case of Octa flash memory, Address is set 2nd byte~5th byte for SPI register commands or Address is set 3rd byte~6th byte for OPI register commands. In case of Octa RAM, Address is set Row address and Column address. If the address is not defined, please set to 0.
AddressLength	Address length of the command. Its unit is byte. If the address is not defined, please set to 0.
BitPosition	Bit position of status register or configuration register.

See also

r_octa_Reg_t

5.4.3 r_octa_RegSetParam_t

Description

This type describes register information and set parameter of the Octa RAM/flash memory.

Definition

```
typedef struct
{
    r_octa_Reg_t          Reg;
    uint32_t              Address;
    uint8_t               AddressLength;
    uint16_t              BitMask;
    uint16_t              BitSet;
} r_octa_RegSetParam_t;
```

Table 5-18 Member of r_octa_RegSetParam_t

Name	Description
Reg	The register of the Octa RAM/flash memory
Address	Address of the command for Octa RAM/flash memory. In case of Octa flash memory, Address is set 2nd byte~5th byte for SPI register commands or Address is set 3rd byte~6th byte for OPI register commands. In case of Octa RAM, Address is set Row address and Column address. If the address is not defined, please set to 0.
AddressLength	Address length of the command. Its unit is byte. If the address is not defined, please set to 0.
BitMask	Bit mask of status register or configuration register.
BitSet	Value of status register or configuration register.

See also

r_octa_Reg_t

5.4.4 r_octa_Timing_t

Description

This type describes the latency.

Definition

```
typedef struct
{
    r_octa_LowPeriod_t          Low;
    r_octa_HighPeriod_t         High;
    r_octa_BetweenPeriod_t      Between;
} r_octa_Timing_t;
```

Table 5-19 Member of r_octa_Timing_t

Name	Description
Low	Set the period for issuing command when the device chip select pull down.
High	Set the period for the device chip select pull up after the command is finished.
Between	Set the period between two commands.

See also

r_octa_LowPeriod_t
r_octa_HighPeriod_t
r_octa_BetweenPeriod_t

5.4.5 r_octa_Command_t

Description

This type describes the Octa RAM/flash commands.

Definition

```
typedef struct
{
    r_octa_RegInfo_t            BlockProtect;
    r_octa_RegInfo_t            WriteEnableLatch;
    r_octa_RegInfo_t            WriteInProgress;
    r_octa_RegInfo_t            PreCycle;
    r_octa_RegInfo_t            FlashTransferType;
    r_octa_RegInfo_t            DOS;
    r_octa_RegSetParam_t        DummyCycle;
    r_octa_Timing_t             ReadTiming;
    r_octa_Timing_t             WriteTiming;
    r_octa_Timing_t             CfgTiming;
    r_octa_CmdTransaction_t     ReadSPI3B;
    r_octa_CmdTransaction_t     ReadSPI4B;
    r_octa_CmdTransaction_t     ReadOPI;
    r_octa_CmdTransaction_t     ReadDOPI;
    r_octa_CmdTransaction_t     WriteSPI3B;
    r_octa_CmdTransaction_t     WriteSPI4B;
    r_octa_CmdTransaction_t     WriteOPI;
    r_octa_CmdTransaction_t     WriteDOPI;
    r_octa_CmdTransaction_t     WriteBufInitialSPI;
    r_octa_CmdTransaction_t     WriteBufInitialOPI;
    r_octa_CmdTransaction_t     WriteBufContinueSPI;
    r_octa_CmdTransaction_t     WriteBufContinueOPI;
    r_octa_CmdTransaction_t     WriteBufConfirmSPI;
    r_octa_CmdTransaction_t     WriteBufConfirmOPI;
    r_octa_CmdTransaction_t     WriteEnableSPI;
    r_octa_CmdTransaction_t     WriteEnableOPI;
    r_octa_CmdTransaction_t     EraseSPI3B;
    r_octa_CmdTransaction_t     EraseSPI4B;
    r_octa_CmdTransaction_t     EraseOPI;
    r_octa_CmdTransaction_t     ReadStsSPI;
    r_octa_CmdTransaction_t     ReadStsOPI;
    r_octa_CmdTransaction_t     ReadCfgSPI;
    r_octa_CmdTransaction_t     ReadCfgOPI;
    r_octa_CmdTransaction_t     ReadCfg2SPI;
    r_octa_CmdTransaction_t     ReadCfg2OPI;
    r_octa_CmdTransaction_t     WriteStsCfgSPI;
    r_octa_CmdTransaction_t     WriteStsOPI;
    r_octa_CmdTransaction_t     WriteCfgOPI;
    r_octa_CmdTransaction_t     WriteCfg2SPI;
    r_octa_CmdTransaction_t     WriteCfg2OPI;
    r_octa_CmdTransaction_t     ResetEnableSPI;
    r_octa_CmdTransaction_t     ResetEnableOPI;
    r_octa_CmdTransaction_t     ResetSPI;
    r_octa_CmdTransaction_t     ResetOPI;
    r_octa_CmdTransaction_t     ReadIDSPI;
    r_octa_CmdTransaction_t     ReadIDOPI;
} r_octa_Command_t;
```

Table 5-20 Member of r_octa_Command_t

Name	Description
BlockProtect	Block Protection information.
WriteEnableLatch	Write Enable Latch.
WriteInProgress	Write in progress information.
PreCycle	Pre-cycle enable information.
FlashTransferType	Transfer type information of connected Octa flash memory.
DOS	DOS (DQS on STR mode) information. DQS is “Data Strobe Signal”.
DummyCycle	Dummy cycle setting information.
ReadTiming	Read timing setting of connected Octa RAM/flash memory.
WriteTiming	Write timing setting of connected Octa RAM/flash memory.
CfgTiming	Configuration timing setting of connected Octa RAM/flash memory.
ReadSPI3B	Read SPI 3 bytes address command.
ReadSPI4B	Read SPI 4 bytes address command.
ReadOPI	Read OPI command.
ReadDOPI	Read DTR (Double Transfer Rate) OPI command.
WriteSPI3B	Write SPI 3 bytes address command.
WriteSPI4B	Write SPI 4 bytes address command.
WriteOPI	Write OPI command.
WriteDOPI	Write DTR (Double Transfer Rate) OPI command.
WriteBufInitialSPI	Write buffer Initial SPI command.
WriteBufInitialOPI	Write buffer Initial OPI command.
WriteBufContinueSPI	Write buffer Continue SPI command.
WriteBufContinueOPI	Write buffer Continue OSPI command.
WriteBufConfirmSPI	Write buffer Confirm SPI command.
WriteBufConfirmOPI	Write buffer Confirm OPI command.
WriteEnableSPI	Write Enable SPI command.
WriteEnableOPI	Write Enable OPI command.
EraseSPI3B	Erase SPI 3 bytes address command.
EraseSPI4B	Erase SPI 4 bytes address command.
EraseOPI	Erase OPI command.
ReadStsSPI	Read Status SPI command.
ReadStsOPI	Read Status OPI command.
ReadCfgSPI	Read Configuration SPI command.
ReadCfgOPI	Read Configuration OPI command.
ReadCfg2SPI	Read Configuration2 SPI command.
ReadCfg2OPI	Read Configuration2 OPI command.
WriteStsCfgSPI	Write Status / Configuration SPI command.
WriteStsOPI	Write Status OPI command.
WriteCfgOPI	Write Configuration OPI command.
WriteCfg2SPI	Write Configuration2 SPI command.
WriteCfg2OPI	Write Configuration2 OPI command.
ResetEnableSPI	Reset Enable SPI command.
ResetEnableOPI	Reset Enable OPI command.
ResetSPI	Reset SPI command.
ResetOPI	Reset OPI command.
ReadIDSPI	Read ID SPI command.

ReadIDOPI	Read ID OPI command.

See also

r_octa_RegInfo_t
r_octa_RegSetParam_t
r_octa_Timing_t
r_octa_CmdTransaction_t

5.4.6 r_octa_DQSDelay_t

Description

This type describes the DQS (Data Strobe Signal) delay.

Definition

```
typedef struct
{
    uint8_t          EnableCnt;
    uint8_t          Delay;
} r_octa_DQSDelay_t;
```

Table 5-21 Member of r_octa_DQSDelay_t

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EnableCnt</td><td>DQS enable counter.
During read operation with DQS clock input (RAM/OPI/DOPI mode), DQS clock will transition from high-impedance to zero after command/address phase is finished.
To prevent using such invalid DQS clock, user can adjust these configuration to guarantee data correctness.
The range is 0 to 15. Actual enable DQS clock input is set value + 1 clock.
If EnableCnt=0 is set, Enable DQS clock input at 1st SCLK.
If EnableCnt=1 is set, Enable DQS clock input at 2nd SCLK.
If EnableCnt=2 is set, Enable DQS clock input at 3rd SCLK.
If EnableCnt=3 is set, Enable DQS clock input at 4th SCLK.
...
If EnableCnt=15 is set, Enable DQS clock input at 16th SCLK.

Reference DQS enable counter settings (when SCLK frequency < 120 MHz)
<table>
<tr><th rowspan="2">DQS delay [SCLK]</th><th colspan="2">RAM</th><th>Flash OPI</th><th colspan="2">Flash DOPI</th></tr>
<tr><th>Pre-cycle ON</th><th>Pre-cycle OFF</th><th>Pre-cycle ON</th><th>Pre-cycle ON</th><th>Pre-cycle OFF</th></tr>
<tr><td>0 - 0.5</td><td>3</td><td>3 - 4</td><td>7 - 8</td><td>5</td><td>5 - 6</td></tr>
<tr><td>0.5 - 1.5</td><td>4</td><td>5</td><td>8</td><td>6</td><td>6 - 7</td></tr>
<tr><td>1.5 - 2.5</td><td>5</td><td>5 - 6</td><td>9</td><td>7</td><td>7 - 8</td></tr>
</table>
</td></tr>
<tr><td>Delay</td><td>DQS delay value.
Delay value = SCLK output pad delay + SCLK wire bonding + DQS wire bonding + DQS input pad delay
(Memory device delay has considered in this table, maximum 2 SCLK cycle)</td></tr>
</table>

See also

None

5.4.7 r_octa_Config_t

Description

This type describes the configuration of the unit.

Definition

```
typedef struct
{
    r_octa_DeviceType_t         DeviceType;
    r_octa_OperatingMode_t      OpeMode;
    r_octa_DataTransferMode_t   DataTransferMode;
    r_octa_AddressMode_t        AddressMode;
    uint32_t                    MemorySize;
    uint32_t                    SectorSize;
    uint32_t                    PageSize;
    r_octa_Command_t*           Command;
    uint32_t                    RelaxSize;
    r_octa_PreCycle_t           PreCycle;
    r_octa_DQSDelay_t           DQSDelay;
    uint32_t                    CalAddress;
} r_octa_Config_t;
```

Table 5-22 Member of r_octa_Config_t

Name	Description
DeviceType	Control device type.
OpeMode	Operating mode.
DataTransferMode	Data transfer mode.
AddressMode	Address mode.
MemorySize	Total memory byte-size of connected Octa RAM/flash memory. MemorySize should be multiple of SectorSize if SectorSize is defined. (e.g. This size is 64 MBytes (64*1024*1024), when MX25LW51245G is connected.)
SectorSize	Erase Sector byte-size of connected Octa flash memory. Sector size must be power-of-two value (2^n). If not required, please set to 0. (e.g. This size is 4 Kbytes when MX25LW51245G is connected.)
PageSize	Page byte-size of connected Octa flash memory. Page byte-size must be power-of-two value (2^n). If not required, please set to 0. (e.g. This size is 256 bytes when MX25LW51245G is connected.)
Command	Command setting. This memory must be kept allocating till R_OCTA_Close is finished.
RelaxSize	Relax byte-size. Relax size must be multiple of 4-byte. This size is used in case of manual mode. Some APIs call R_OCTA_Sys_Relax in units of this size. If not required, please set to 0.
PreCycle	Pre-cycle mode. This mode is used when the data transfer mode is R_OCTA_MODE_DOPI.
DQSDelay	DQS delay.
CalAddress	Calibration Start address. This specifies the start address for executing the calibration. The calibration is adjusted by actually reading / writing to the connected OCTA RAM / flash memory while changing the DQS delay value. This driver doesn't executed the calibration. If you want to execute the calibration, please calibrate with data read / data write after calling the R_OCTA_Open function by setting calibration start address to CalAddress, DQS delay value to DQSDelay.Delay. CalAddress must be 32 bit address aligned. If not required, please set to 0xFFFFFFFF. See *3.2.5*.

See also

r_octa_DeviceType_t
r_octa_OperatingMode_t
r_octa_DataTransferMode_t
r_octa_AddressMode_t
r_octa_Command_t
r_octa_PreCycle_t
r_octa_DQSDelay_t

Revision History	Renesas Graphics Library OctaBus Controller (OCTA) Driver User's Manual: Software

Rev.	Date	Description	
		Page	Summary
0.1	Nov 22, 2019	-	First edition.
1.0	April 24, 2020	-	Update Revision.

Renesas Graphics Library
OctaBus Controller (OCTA) Driver
User's Manual: Software
Publication Date: Rev.0.1 Nov 22, 2019
Rev.1.0 April 24, 2020

Published by: Renesas Electronics Corporation

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

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1

Renesas Graphics Library
OctaBus Controller (OCTA) Driver

Renesas Electronics Corporation