User's Manual

IMR Driver

User's Manual: Software

Renesas Electronics
www.renesas.com

Rev.0.15E July.15.21

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

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

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

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

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed for users to understand the user interface specification of this software and targets for users to design application systems used by this software. Please refer to the manuals related to this software.

> 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.

2. Usage for this software

In case of using this software, the customer must make an agreement for software licensing with our company.

3. Abbreviation

- *k* with number means 1000.
- *K* means 1024.
- *NULL* means a pointer to 0 address.
- A *mod* B means a remainder with A divided by B.
- About mathematical formulas in functional description for API, *i* means index of sequences and c means channel number.

Table of Contents

IMR Driver

User's Manual :

1. Overview

1.1 Features

1.1.1 IMR related software

Figure 1-1 describes the entire structure of software to control IMR-LXn. The user application uses a library that consists of an "IMR driver".

First, the user application calls the IMR driver to set the parameters.

User app calls DLG to create Display List.

The user application then uses the display list it runs to call the IMR driver.

The IMR driver starts executing IMR-LXn, gets the execution result from IMR-LXn.

Finally, IMR Driver returns it to the user application.

This document explains about the Software Architecture Design of IMR Driver in red box.

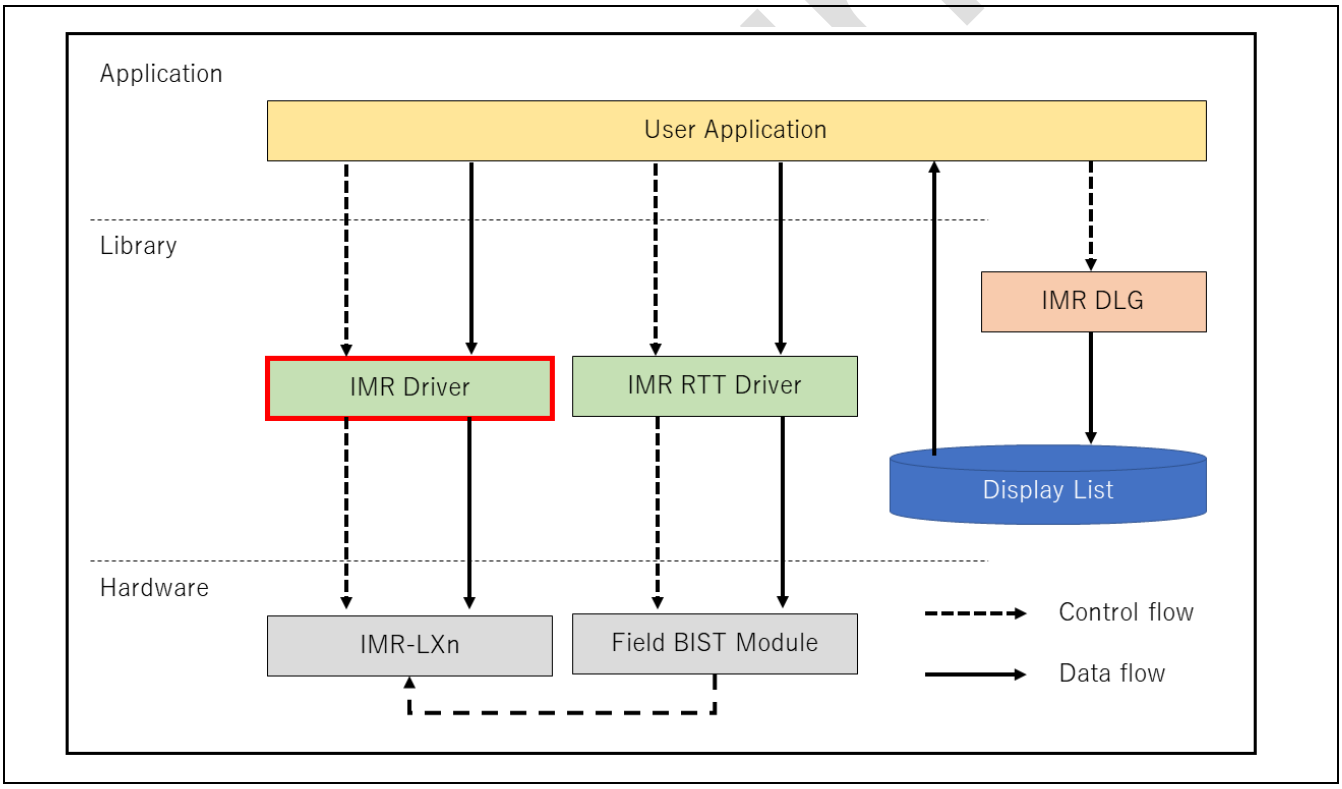

Figure 1-1 Entire structure of software to control IMR-LXn

1.1.2 IMR Driver

This manual describes IMR Driver functions. The IMR Driver is called from IMR User Application, and controls IMR-LXn.

The block diagram of IMR Driver is described in ***Figure 1-2***.

Component boxed in red is the scope of this document.

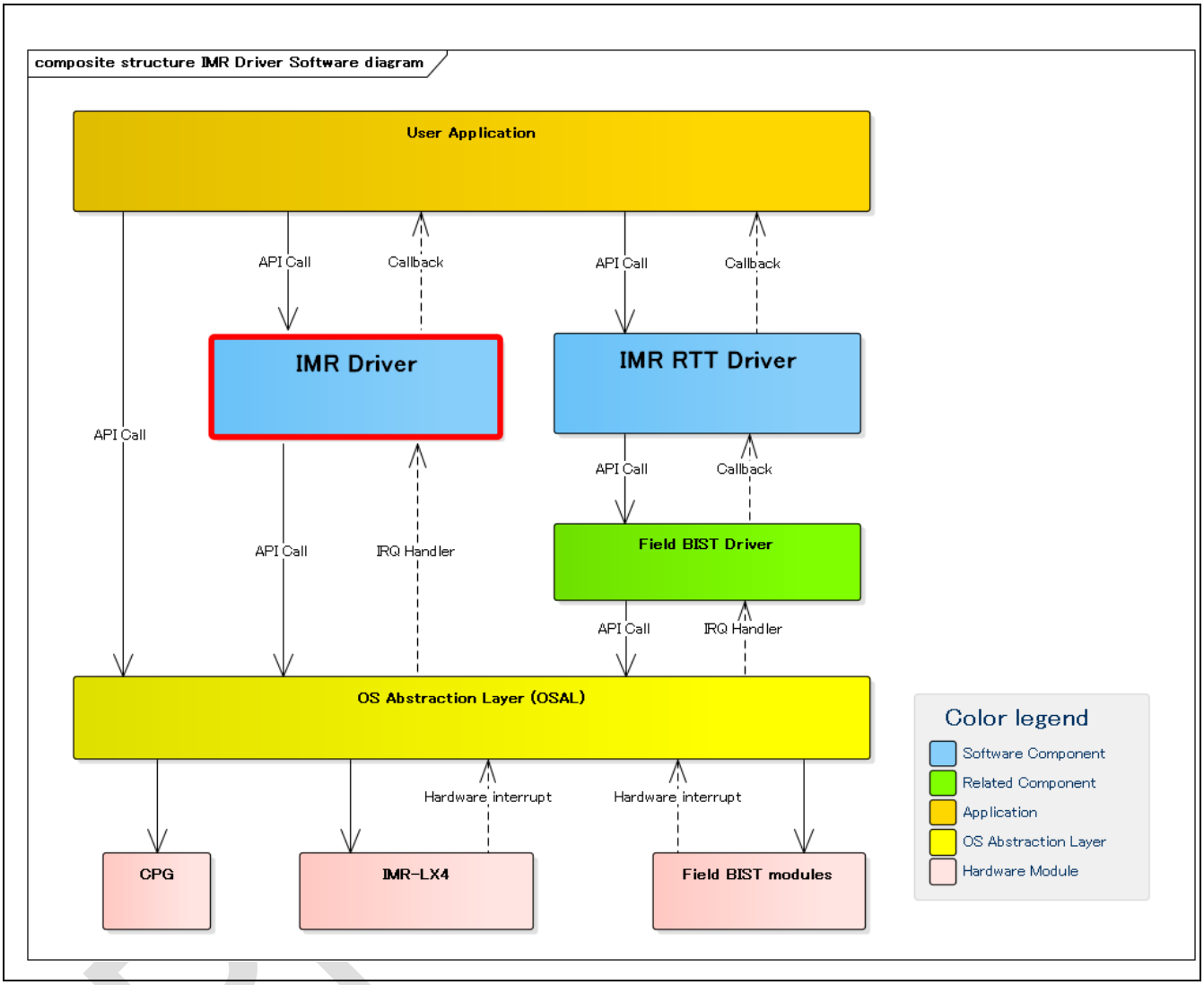

Figure 1-2 Block diagram

Table 1-1 describes the outline of each component in *Figure 1-2*.

Table 1-1 Component List

Component	Description
IMR Driver	Described in this document
OS Abstraction Layer	Refer to the *Operating System Abstraction Layer APPLICATION NOTE* for detail.
IMR-LXn,CPG	Refer to the *R-Car Series, 3rd Generation User's Manual: Hardware* for detail.
IMR RTT Driver	Out of scope of this document
Field BIST Driver	Out of scope of this document
Field BIST module	Out of scope of this document

Table 1-2 describes the outline of each interrupt in *Figure 1-2*.

Table 1-2 Interrupt List

Interrupt	Description
TRAP	Rendering operation has been completed.
IER	Illegal instruction was executed in DL.
WUPOVF	WUP signals which has received but not processed exceeds the maximum number allowed. Please refer to chapter 3.1.1.2 of *"Product Information of IMR Driver"* for SoC differences.
WUPERR	Receives the WUP signal from another module while it is not executing a display list. Please refer to chapter 3.1.1.2 of *"Product Information of IMR Driver"* for SoC differences.

1.2 References

Please refer to chapter 1.4.1 of ***"Product Information of IMR Driver".***

1.3 List of Terms

Table 1-3 List of Terms

Term	Description
IMR	Image Renderer
IMR-LXn	Image Renderer light extended n. For example, IMR-LX6 for V4H and IMR-LX4 for V3U.
DL	Display List
WUP	An instruction of DL. It is a part of Inter-Module Synchronization feature of IMR-LXn.
OSAL	Operating System Abstraction Layer.
DYP	Double luminance plane(DYP mode)

.

2. Basic Specification

2.1 Summary Specification

The IMR Driver enlarges / reduces and corrects distortion of the input image. Refer to Chapter 3.2 for the processing flow of the function. The IMR Driver provides is following functions to control the IMR-LXn hardware.

- Basic functions
  - ➢ Initialize and exit the IMR Driver
    - ✧ Provide the ***R_IMRDRV_Init*** function that initializes of IMR Driver. This function initializes the IMR Driver that controls of IMR-LXn hardware.
    - ✧ Provide the ***R_IMRDRV_Quit*** function that terminates of IMR Driver. This function releases resources used inside of IMR Driver and terminates the IMR Driver.
    - ✧ Provides the ***R_IMRDRV_Start*** function that initializes the IMR-LXn hardware. This function starts supplying clock.
    - ✧ IMR-LXn Provides the ***R_IMRDRV_Stop*** function to stop the hardware. This function shuts off the clock supply. Specifications for suspension of supply follow the "Power policy setting function (Not Implement)".
  - ➢ Attribute data setting function when executing the image rendering
    - ✧ Provides the ***R_IMRDRV_AttrSetParam*** function that sets specified basic parameters (such as source and destination image data) to the IMR Driver.
  - ➢ Execution function for image rendering
    - ✧ Provide is ***R_IMRDRV_Execute*** function to execute the image rendering.
    - ✧ Notifications such as image rendering completion (TRAP instruction decode), and image rendering execution error are specified asynchronously by the callback function of ***R_IMRDRV_Execute*** function. It is done with (***p_imrdrv_callback_func_t***). See Refer 3.4.3 for details.

- Optional functions
  - ➢ Following functions are provided to set optional attribute data, when execute the image rendering.
    - ✧ ***R_IMRDRV_AttrSetRotatorScaler***
      - ➢ Set specified rotator and scaler parameters to the IMR Driver.
    - ✧ ***R_IMRDRV_AttrSetCacheMode***
      - ➢ Set specified cache mode parameter to the IMR Driver.
    - ✧ ***R_IMRDRV_AttrSetDoubleLumaPlane***
      - ➢ Set specified double luminance plane parameter to the IMR Driver.

- Power policy setting function(Not Implement)

- Software version acquisition function
  - ➢ Provide is ***R_IMRDRV_GetVersion*** function to acquisition software version of IMR Driver.

2.2 Reserved word

The IMR Driver uses the following prefixes for avoiding confusion from other software. No rules are specified except for the following definitions.

Table 2-1 Prefixes

prefix	Description
R_IMRDRV_Xxx	External functions
r_imrdrv_xxx	Internal functions
st_imrdrv_xxx_t	Structure type
e_imrdrv_xxx_t	Enumeration type
IMRDRV_XXX	Enumerators
IMRDRV_XXX	Define
g_imrdrv_xxx	Global variable
p_imrdrv_xxx_t	Function pointer type

2.3 Data Related Overview

Figure 2.1 shows the rendering processing data related overview using this software.

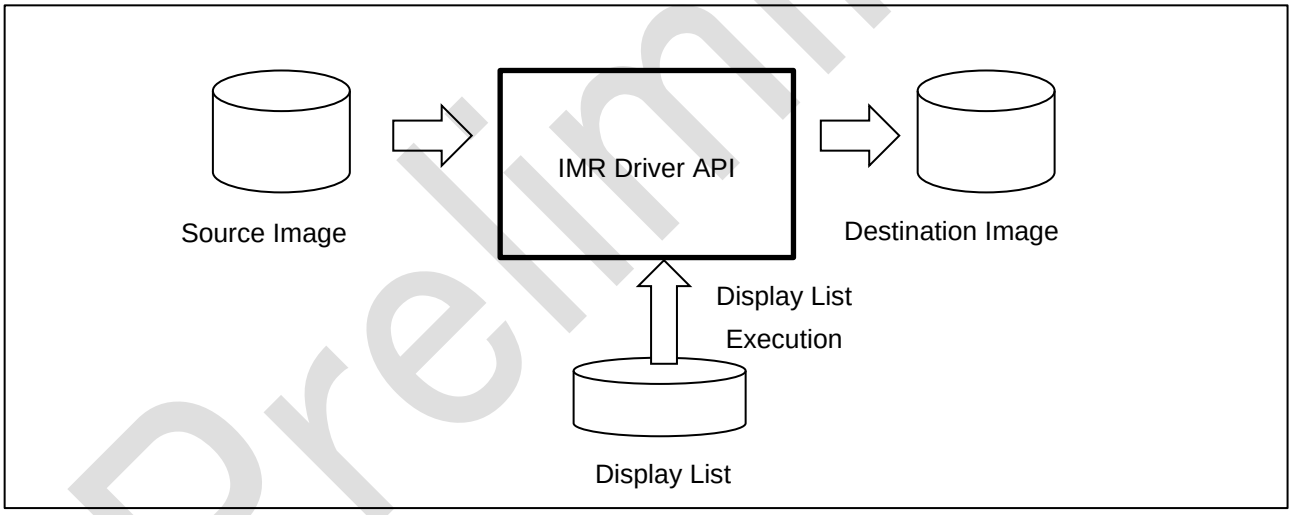

Figure 2.1 Data related overview

2.4 Image Data Specifications

Please refer to chapter 2.4 of ***"Product Information of IMR Driver"***.

2.5 Memory Region for Images

Figure 2.2 shows the memory imagery of the Y/UV separate format, Figure 2.3 shows the memory imagery of the interleaved format, Figure 2.4 shows the memory imagery of the YUV422 semi-planar format, and Figure 2.5 shows the memory imagery of the YUV420 semi-planar format.

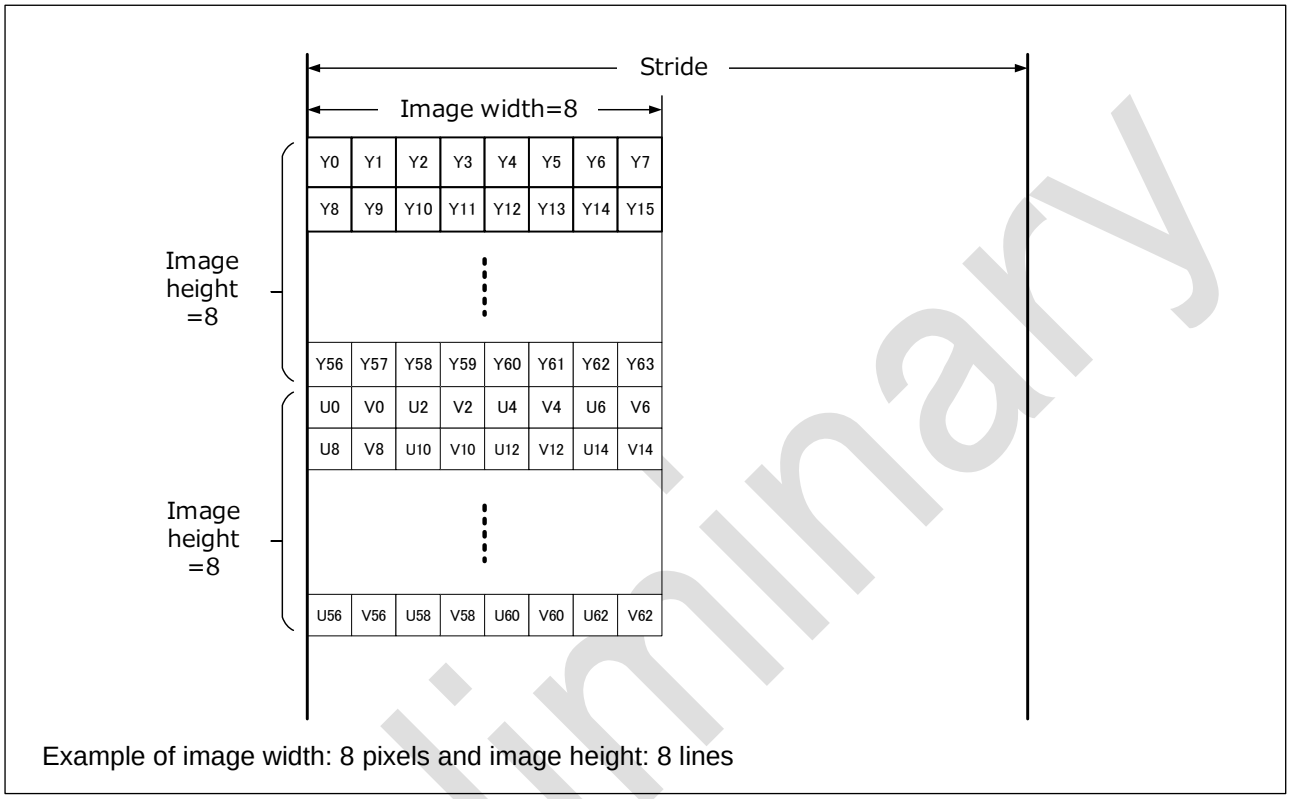

Figure 2.2 Memory Imagery of Y/UV Separate Format

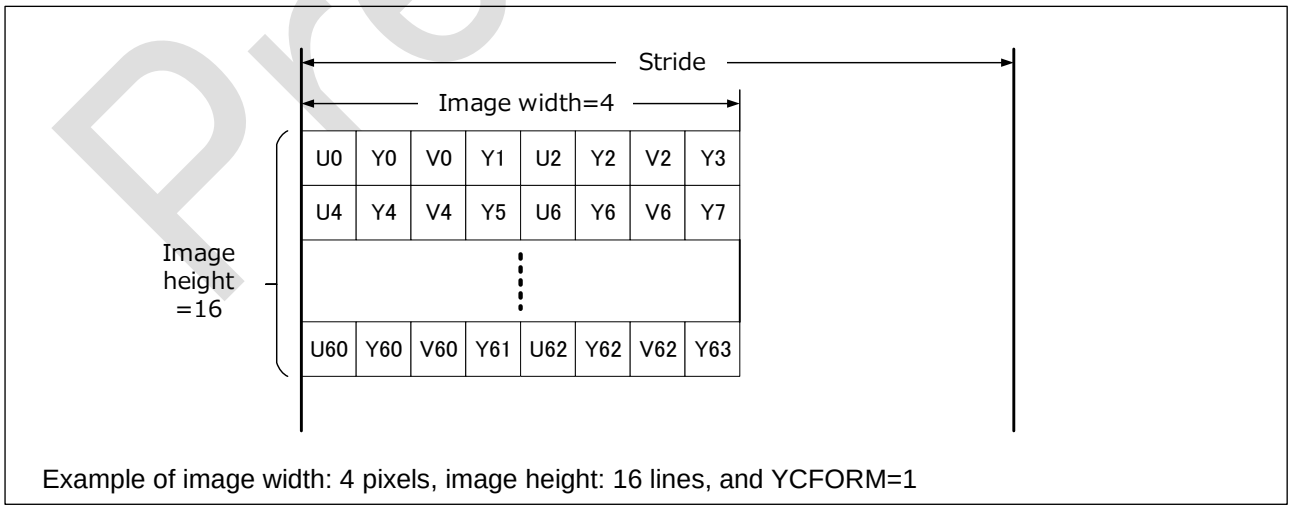

Figure 2.3 Memory Imagery of Interleaved Format

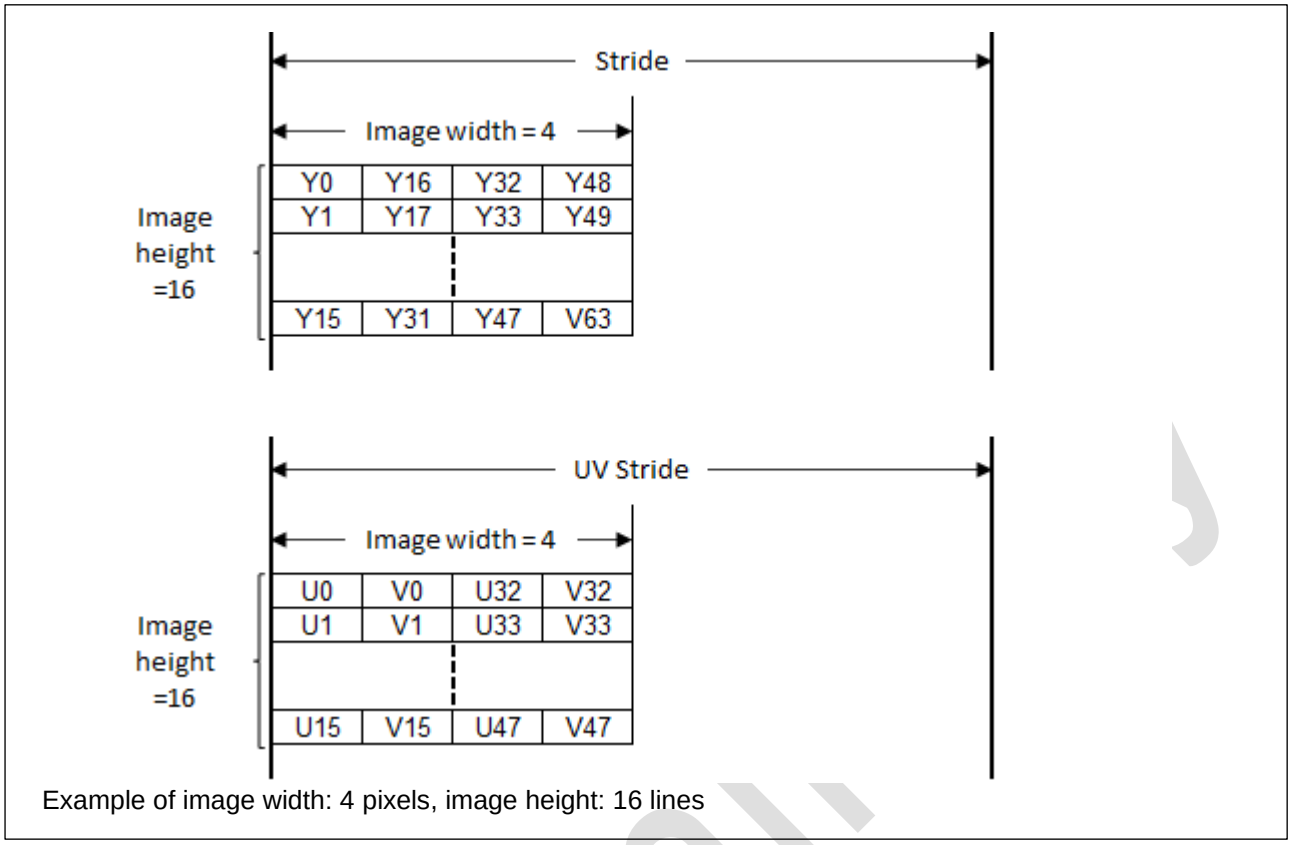

Figure 2.4 Memory Imagery of YUV422 Semi-planar Format

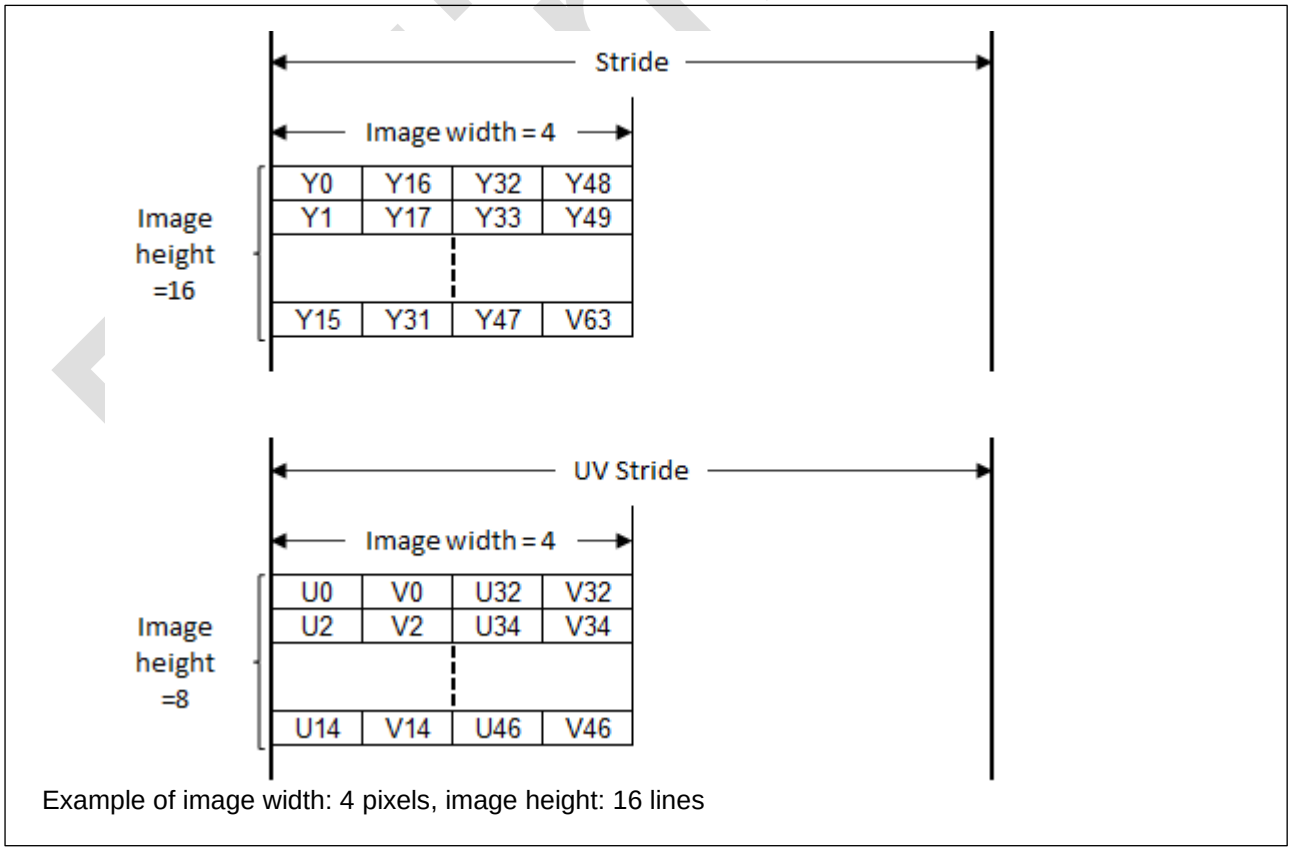

Figure 2.5 Memory Imagery of YUV420 Semi-planar Format

2.6 Display List

Rendering by IMR is performed by specifying a Display List (hereafter called “DL”), an array of data in 4-byte units, where the operation codes such as IMR register operation and conversion coordinates designation are described, to IMR. In this document, this 4-byte data is described as a DL element. Also, one or more DL elements combining the operation code and data required for that operation code is described as a DL instruction.
DL of a fixed table created in advance can be used by copying it into DL memory area.
It is necessary to include TRAP instruction in the DL.
The user app creates DL with IMR DLG.

3. Function Description

3.1 Finite-State machine

The Finite-State machine of IMR Driver is shown in ***Figure 3-1***.

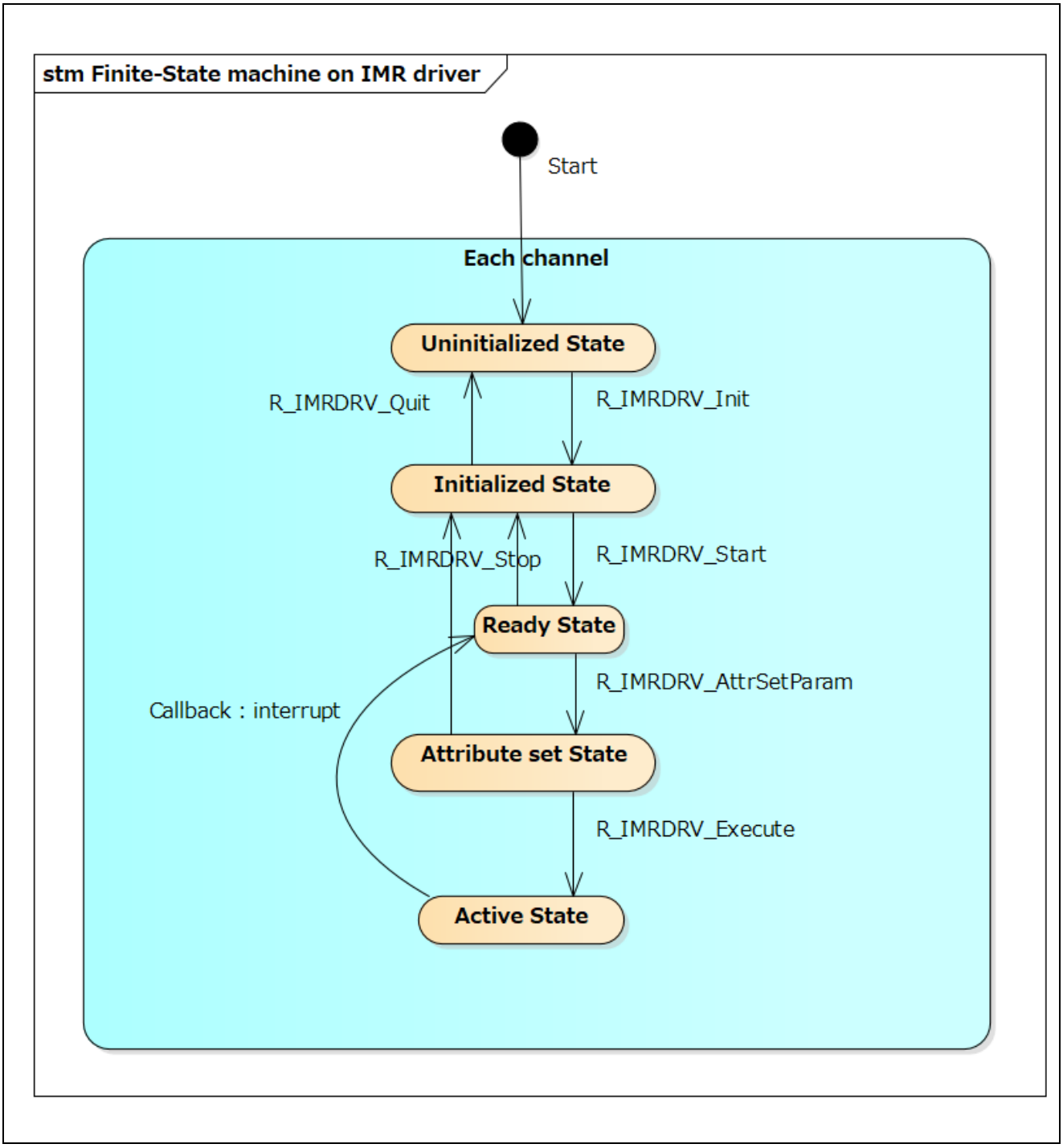

Figure 3-1 Finite State machine on IMR Driver

Note that these states are managed for each channel.

3.1.1 Uninitialized State

This is the initial state of the driver immediately after booting the system.
When ***R_IMRDRV_Init*** is called, the driver state is changed to the ***Initialized State***.

3.1.2 Initialized State

This state indicates that the IMR Driver has been initialized.

When ***R_IMRDRV_Start*** is called, the driver state is changed to the ***Ready State***.
When ***R_IMRDRV_Quit*** is called, the driver state is changed to the ***Uninitialized State***.
Note that this state is managed for each channel.

3.1.3 Ready State

This state indicates that the channel is ready to execute DL.

When ***R_IMRDRV_AttrSetParam*** is called, the driver state is changed to the ***Attribute set State***.
When ***R_IMRDRV_Stop*** is called, the driver state is changed to ***Initialized State***.

3.1.4 Attribute set State

This state indicates that the channel is set attribute parameter for executing DL.

When ***R_IMRDRV_Execute*** is called, the driver state is changed to the ***Active State***.
When ***R_IMRDRV_Stop*** is called, the driver state is changed to ***Initialized State***.

3.1.5 Active State

This state indicates that the channel is executing DL which is provided by ***R_IMRDRV_Execute***.

When the execution is completed, the driver state is changed to the ***Ready State***.
When the execution is interrupted by the TRAP, IER, WUPOVF and WUPERR interrupt, the driver state is changed to the ***Ready State***.

3.2 Function Flow

The typical function-flow is shown in ***Figure 3-2***. For more details, please refer to ***Finite-State machine*** and ***Functions***.

3.2.1 Basic Function Flow

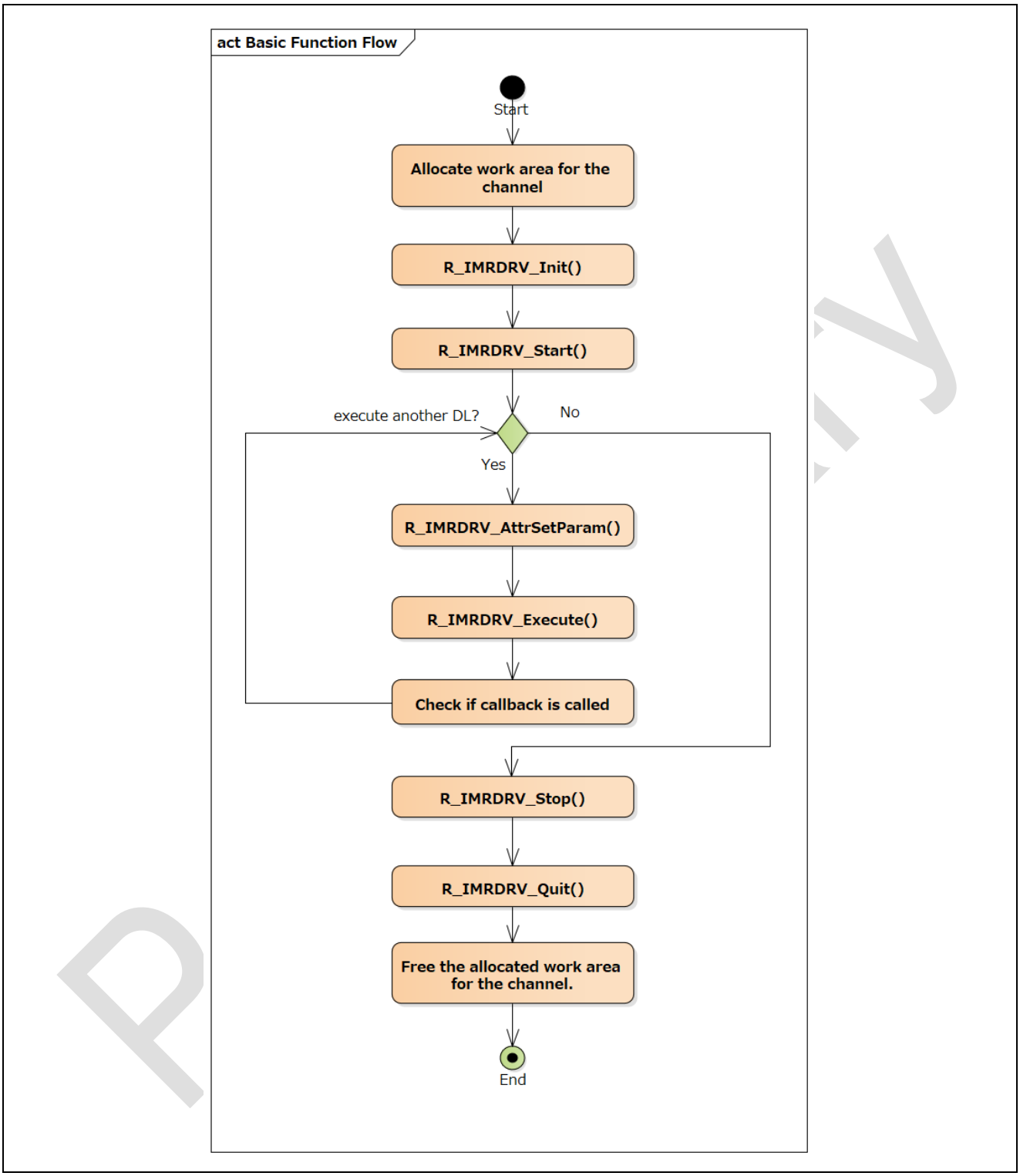

Figure 3-2 Basic Function Flow on IMR driver

3.2.2 Optional Function Flow (rotator and scaler)

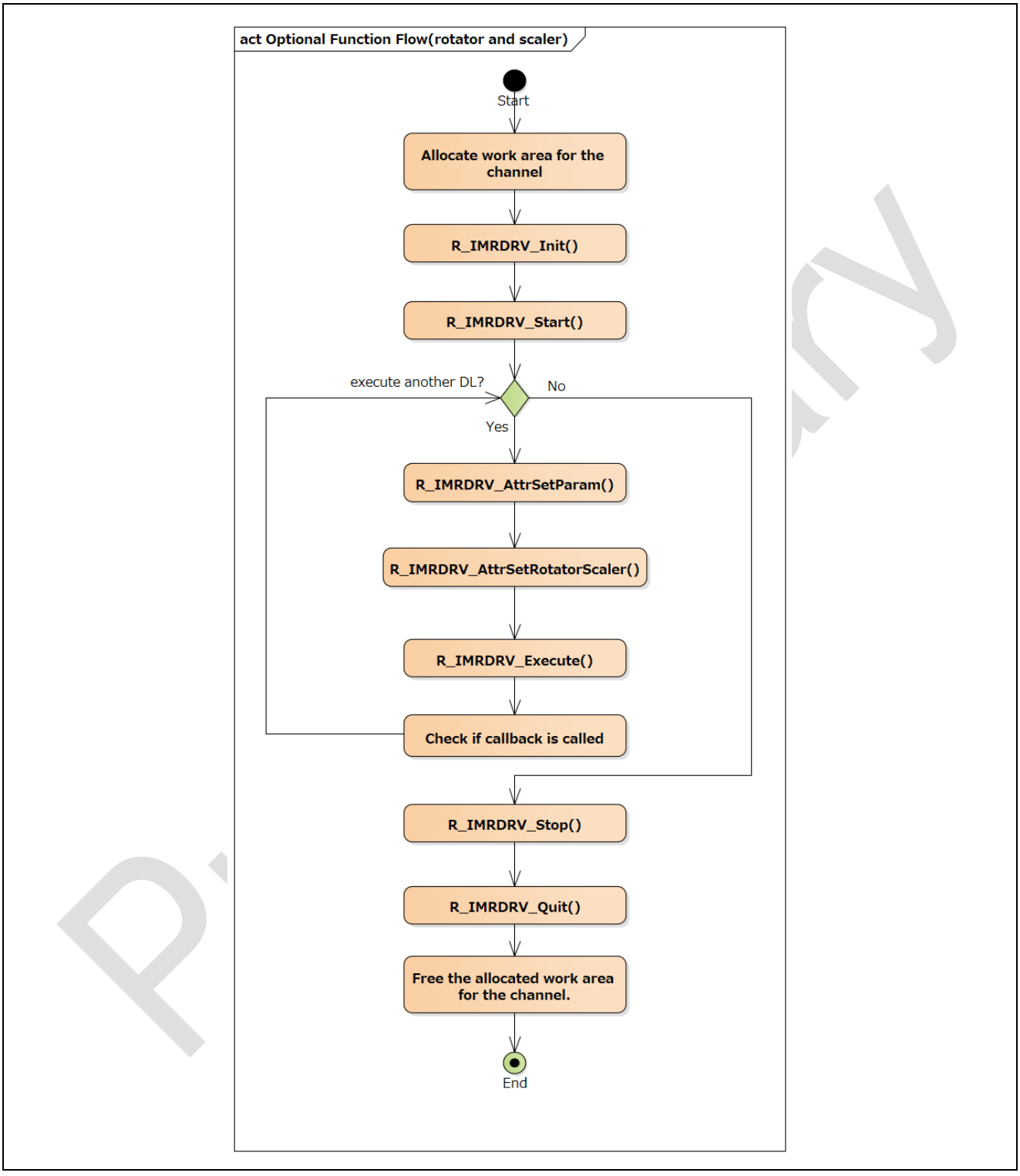

Figure 3-3 Optional Function Flow (rotator and scaler) on IMR driver

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for the rotator and scaler supported by SoC differences.

3.2.3 Optional Function Flow (cache mode)

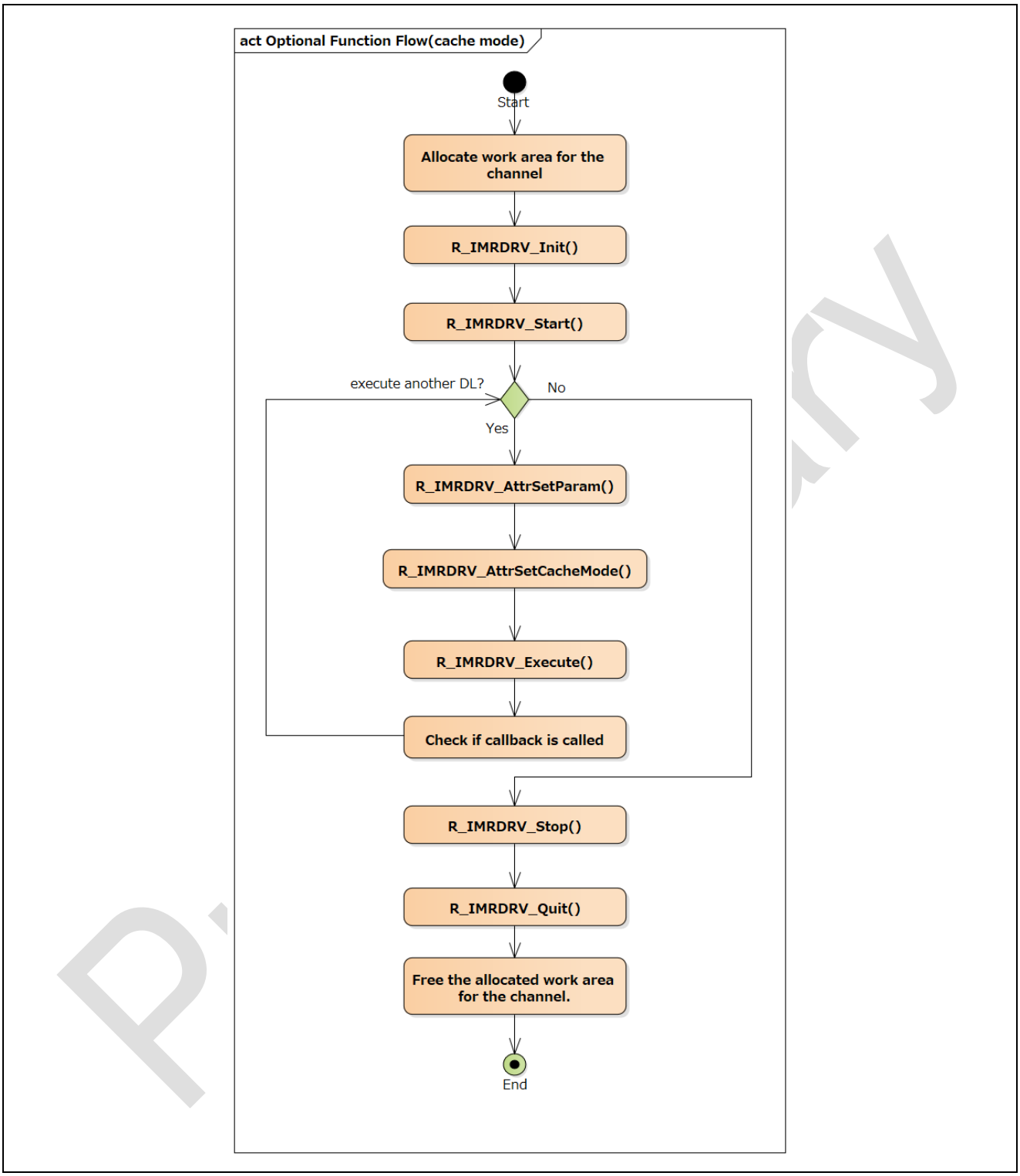

Figure 3-4 Optional Function Flow (cache mode) on IMR driver

3.2.4 Optional Function Flow (DYP)

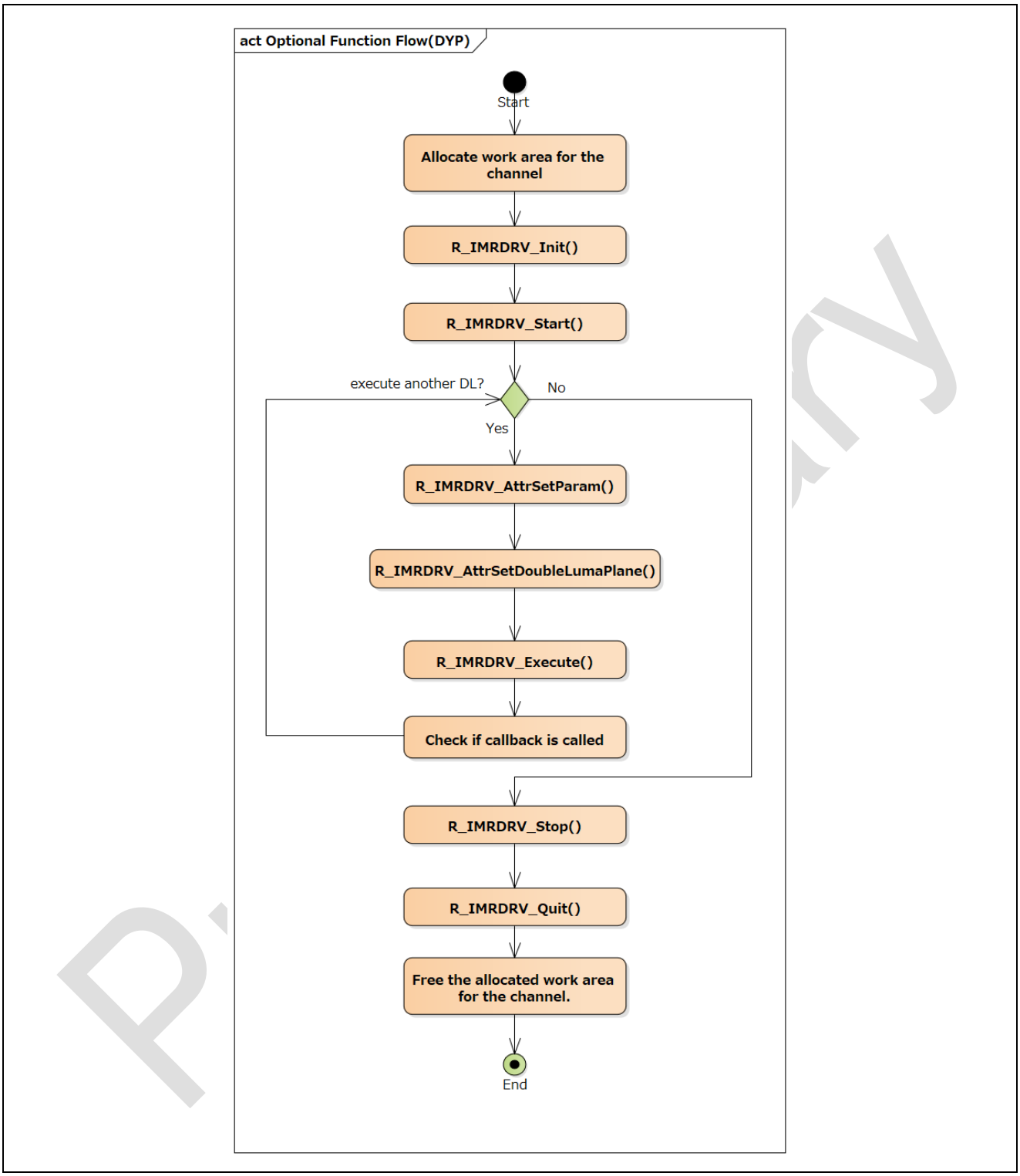

Figure 3-5 Optional Function Flow (DYP) on IMR driver

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for the DYP supported by SoC differences.

3.3 Error Processing

This driver notifies the user of an error by the result value of the function. this section shows about each error.

3.3.1 Parameter error

This error occurs by specifying an invalid parameter.
If this error occurs, the driver returns ***IMRDRV_ERROR_PAR***.
In this case, the user can continue by put the correct value to the argument.
Please refer to ***5.Functions*** for detail of each parameter.

3.3.2 Device error

This error is caused by hardware error in IMR-LXn or OSAL function error.
If this error occurs, the driver will return ***IMRDRV_ERROR_DEV***. In this case, execute system reset.

3.3.3 Configuration error

This error is caused by an invalid combination of attributes.
If this error occurs, the driver returns ***IMRDRV_ERROR_CONF***.
In this case, the user checks the combination of parameters set in ***R_IMRDRV_AttrSetParam***, ***R_IMRDRV_AttrSetRotatorScaler***, ***R_IMRDRV_AttrSetCacheMode*** and ***R_IMRDRV_AttrSetDoubleLumaPlane***.

3.3.4 Fail

This error occurs when an unexpected error is detected, such as due to stack corruption.
If this error occurs, the driver returns ***IMRDRV_ERROR_FAIL***. In this case, execute system reset.

3.3.5 I/O errorr

This error occurs when register access fails.
If this error occurs, the driver returns ***IMRDRV_ERROR_IO***. In this case, execute system reset.

3.3.6 State error

This error occurs during API calls that cannot be state transitioned.
If this error occurs, the driver returns ***IMRDRV_ERROR_STATE***.
In this case, modify the function order to call the function from the correct state machine.
Please refer to ***3.1 Finite-State machine*** and ***5 Functions*** for the correct combination of function call and state machine.

3.3.7 Handle error

This error occurs when IMR Driver internal control data ***st_imrdrv_ctrl_handle_t*** is invalid. (Not implement)
If this error occurs, the driver returns ***IMRDRV_ERROR_HANDLE***. In this case, execute system reset.

3.3.8 Timeout error

This error occurs by mutex timeout or message timeout.
If this error occurs, the driver will return ***IMRDRV_ERROR_TIME***.
In this case, modify the timeout values at calling ***R_IMRDRV_Init***.

3.3.9 Configuration register error

This error occurs by Configuration Register Check. (Not implement)
If this error occurs, the driver will return ***IMRDRV_ERROR_CONFREG***.

3.3.10 Module stop error

This error occurs by Unintended Module Stop Check. (Not implement)
If this error occurs, the driver will return ***IMRDRV_ERROR_MODULESTOP***.

3.4 Usage and Restriction

3.4.1 User side prepare data

Before using IMR Driver, show the preparation of the User Application below. When using the IMR Driver, User Application need to prepare the data in the following ***Table 3-1*** on the user side, and set it in the argument of ***R_IMRDRV_Init*** function.

Table 3-1 User side prepare data List

No.	Input Parameter	Description
1	Working memory area	The user side should allocate the work area for IMR Driver. The size is ***IMRDRV_SIZE_WORKAREA***. The user side set the start address pointer of the work area to ***p_work_addr*** in ***st_imrdrv_initdata_t*** and should use as an argument of the ***R_IMRDRV_Init*** function. IMR Driver uses this work area as Control area and return the pointer of this data in ***p_handle*** of ***R_IMRDRV_Init*** function. The user should use this pointer, ***p_handle*** in each API function. The User Application should prepare only one the work area for each the channel. If IMR Driver is operated using multiple data, operation can not be guaranteed.

The data shown above is necessary for internal control of Driver. If these data are rewritten during Driver operation(from call ***R_IMRDRV_Init*** to call ***R_IMRDRV_Quit***), operation can not be guaranteed.

3.4.2 Timeout detection of channels in IMR-LXn

The execution time of IMR-LXn must be observed by user application. For example, user application starts the timer immediately before the execution of ***R_IMRDRV_Execute*** and stops at the beginning of the ***Callback function***. If the timeout occurs, the user application should restart the system. The expected execution time of each channel depends on the DL which user created.

3.4.3 How interrupts from IMR-LXn are handled

Several kinds of interrupt occur on the channel of IMR-LXn when was initiated by Execute. Refer to ***Table 1-2***

If the interrupt occurs, the callback function passed by ***R_IMRDRV_Execute*** is executed and IMR Driver changes the driver state from ***Active State*** to ***Ready State***.

The user application checks whether the DL completed normally with the argument details code of the callback function. Also, since the callback function becomes a task context, the callback function processing should be minimized.

The user application needs to reboot the system when IER, WUPOVF and WUPERR occur.

4. Data Type definition

4.1 Data Type

Type name	Description
imrdrv_ctrl_handle_t	IMR Driver Control handle

4.2 Definition Values

Name	Value	Description
IMRDRV_SIZE_WORKAREA	2048	Lower limit of work area size.
IMRDRV_ATTR_RS_IDX_ONE_ONE	0	Index of attribute information for scale down to 1/1.
IMRDRV_ATTR_RS_IDX_ONE_HALF	1	Index of attribute information for scale down to 1/2.
IMRDRV_ATTR_RS_IDX_ONE_FOURTH	2	Index of attribute information for scale down to 1/4.
IMRDRV_ATTR_RS_IDX_ONE_EIGHTH	3	Index of attribute information for scale down to 1/8.
IMRDRV_ATTR_RS_IDX_MAX_NUM	4	Maximum number of attribute information for scale down.
IMRDRV_ATTR_RS_ADDR_INVALID	0xFFFFFFFF	Source / destination address of rotating and scaling is invalid.
IMRDRV_ATTR_DYP_ADDR_INVALID	0xFFFFFFFF	Invalid value of address for DYP.
IMRDRV_ATTR_DYP_STRIDE_INVALID	0x00000000	Invalid value of stride for DYP.
IMRDRV_SR_WERR	0x00000800	WERR interrupt notified by a callback.
IMRDRV_SR_WOVF	0x00000400	WOVF interrupt notified by a callback.
IMRDRV_SR_IER	0x00000002	IER interrupt notified by a callback.
IMRDRV_SR_TRA	0x00000001	TRA interrupt notified by a callback.

4.3 Enumerated Type

4.3.1 e_imrdrv_channelno_t

```
typedef enum {
        IMRDRV_CHANNELNO_INVALID        = -1,
        IMRDRV_CHANNELNO_0              = 0,
        IMRDRV_CHANNELNO_1              = 1,
        IMRDRV_CHANNELNO_2              = 2,
        IMRDRV_CHANNELNO_3              = 3,
        IMRDRV_CHANNELNO_4              = 4,
        IMRDRV_CHANNELNO_5              = 5
} e_imrdrv_channelno_t;
```

Table 4-1 Enumerator of ***e_imrdrv_channelno_t***

Name	Description
IMRDRV_CHANNELNO_INVALID	Invalid channel number
IMRDRV_CHANNELNO_0	IMR-LXn channel0
IMRDRV_CHANNELNO_1	IMR-LXn channel1
IMRDRV_CHANNELNO_2	IMR-LXn channel2
IMRDRV_CHANNELNO_3	IMR-LXn channel3
IMRDRV_CHANNELNO_4	IMR-LXn channel4
IMRDRV_CHANNELNO_5	IMR-LXn channel5

4.3.2 e_imrdrv_errorcode_t

```
typedef enum {
        IMRDRV_ERROR_OK                 = 0,
        IMRDRV_ERROR_PAR                = 1,
        IMRDRV_ERROR_DEV                = 2,
        IMRDRV_ERROR_TIME               = 3,
        IMRDRV_ERROR_CONF               = 4,
        IMRDRV_ERROR_FAIL               = 5,
        IMRDRV_ERROR_IO                 = 6,
        IMRDRV_ERROR_STATE              = 1024,
        IMRDRV_ERROR_CONFREG            = 1025,
        IMRDRV_ERROR_MODULESTOP         = 1026,
        IMRDRV_ERROR_HANDLE             = 1027
} e_imrdrv_errorcode_t;
```

Table 4-2 Enumerator of ***e_imrdrv_errorcode_t***

Name	Description
IMRDRV_ERROR_OK	Successful completion
IMRDRV_ERROR_PAR	Parameter error
IMRDRV_ERROR_DEV	Hardware fail
IMRDRV_ERROR_TIME	Timeout error
IMRDRV_ERROR_CONF	Inconsistent configuration error
IMRDRV_ERROR_FAIL	Unexpected error
IMRDRV_ERROR_IO	The register operation error
IMRDRV_ERROR_STATE	IMR Driver status error
IMRDRV_ERROR_CONFREG	Configuration Register Check error
IMRDRV_ERROR_MODULESTOP	Unintended Module Stop Check error
IMRDRV_ERROR_HANDLE	IMR Driver control data error

4.3.3 e_imrdrv_state_t

```
typedef enum {
        IMRDRV_STATE_UNINIT             = 0,
        IMRDRV_STATE_INIT               = 1,
        IMRDRV_STATE_READY              = 2,
        IMRDRV_STATE_ATTR_SET           = 3,
        IMRDRV_STATE_ACTIVE             = 4
} e_imrdrv_state_t;
```

Table 4-3 Enumerator of ***e_imrdrv_state_t***

Name	Description
IMRDRV_STATE_UNINIT	***Uninitialized State***.
IMRDRV_STATE_INIT	***Initialized State***.
IMRDRV_STATE_READY	***Ready State***.
IMRDRV_STATE_ATTR_SET	***Attribute set State***.
IMRDRV_STATE_ACTIVE	***Active State***.

4.3.4 e_imrdrv_color_format_t

```
typedef enum {
        IMRDRV_COLOR_FORM_INVALID       = -1,
        IMRDRV_COLOR_FORM_Y             = 0,
        IMRDRV_COLOR_FORM_UV            = 1,
        IMRDRV_COLOR_FORM_SEP_Y         = 2,
        IMRDRV_COLOR_FORM_SEP_UV        = 3,
        IMRDRV_COLOR_FORM_DATA          = 4,
        IMRDRV_COLOR_FORM_UYVY          = 5,
        IMRDRV_COLOR_FORM_YVYU          = 6,
        IMRDRV_COLOR_FORM_VYUY          = 7,
        IMRDRV_COLOR_FORM_YUYV          = 8,
        IMRDRV_COLOR_FORM_SEMI422       = 9,
        IMRDRV_COLOR_FORM_SEMI420       = 10
} e_imrdrv_color_format_t;
```

Table 4-4 Enumerator of ***e_imrdrv_color_format_t***

Name	Description
IMRDRV_COLOR_FORM_INVALID	Invalid value of color format.
IMRDRV_COLOR_FORM_Y	Y format.
IMRDRV_COLOR_FORM_UV	UV format.
IMRDRV_COLOR_FORM_SEP_Y	Y separate format.
IMRDRV_COLOR_FORM_SEP_UV	UV separate format.
IMRDRV_COLOR_FORM_DATA	Data format.
IMRDRV_COLOR_FORM_UYVY	YUV422 interleaved UYVY format.
IMRDRV_COLOR_FORM_YVYU	YUV422 interleaved YVYU format.
IMRDRV_COLOR_FORM_VYUY	YUV422 interleaved VYUY format.
IMRDRV_COLOR_FORM_YUYV	YUV422 interleaved YUYV format.
IMRDRV_COLOR_FORM_SEMI422	YUV422 semi-planar format.
IMRDRV_COLOR_FORM_SEMI420	YUV420 semi-planar format.

4.3.5 e_imrdrv_bpp_t

```
typedef enum {
        IMRDRV_BPP_INVALID      = -1,
        IMRDRV_BPP_8            = 0,
        IMRDRV_BPP_10           = 1,
        IMRDRV_BPP_12           = 2,
        IMRDRV_BPP_14           = 3,
        IMRDRV_BPP_16           = 4,
        IMRDRV_BPP_32           = 5
} e_imrdrv_bpp_t;
```

Table 4-5 Enumerator of ***e_imrdrv_bpp_t***

Name	Description
IMRDRV_BPP_INVALID	Invalid value of bit per pixel.
IMRDRV_BPP_8	8-bit per pixel.
IMRDRV_BPP_10	10-bit per pixel.
IMRDRV_BPP_12	12-bit per pixel.
IMRDRV_BPP_14	14-bit per pixel.
IMRDRV_BPP_16	16-bit per pixel.
IMRDRV_BPP_32	32-bit per pixel.

4.3.6 e_imrdrv_mode_t

```
typedef enum {
        IMRDRV_MODE_INVALID             = -1,
        IMRDRV_MODE_OFF                 = 0,
        IMRDRV_MODE_ON                  = 1
} e_imrdrv_mode_t;
```

Table 4-6 Enumerator of ***e_imrdrv_mode_t***

Name	Description
IMRDRV_MODE_INVALID	Invalid value.
IMRDRV_MODE_OFF	Mode off.
IMRDRV_MODE_ON	Mode on.

4.3.7 e_imrdrv_scaling_down_filter_t

```
typedef enum {
        IMRDRV_SCALE_DOWN_FILT_INVALID      = -1,
        IMRDRV_SCALE_DOWN_FILT_AVERAGE      = 0,
        IMRDRV_SCALE_DOWN_FILT_ADD          = 1
} e_imrdrv_scaling_down_filter_t;
```

Table 4-7 Enumerator of ***e_imrdrv_scaling_down_filter_t***

Name	Description
IMRDRV_SCALE_DOWN_FILT_INVALID	Invalid value.
IMRDRV_SCALE_DOWN_FILT_AVERAGE	Four times of the average of the neighboring four pixels.
IMRDRV_SCALE_DOWN_FILT_ADD	The values of the neighboring four pixels are added.

4.3.8 e_imrdrv_rounding_mode_t

```
typedef enum {
        IMRDRV_ROUND_MODE_INVALID          = -1,
        IMRDRV_ROUND_MODE_ROUND_DOWN       = 0,
        IMRDRV_ROUND_MODE_ROUND_NEAR       = 1
} e_imrdrv_rounding_mode_t;
```

Table 4-8 Enumerator of ***e_imrdrv_rounding_mode_t***

Name	Description
IMRDRV_ROUND_MODE_INVALID	Invalid value.
IMRDRV_ROUND_MODE_ROUND_DOWN	Rounding down.
IMRDRV_ROUND_MODE_ROUND_NEAR	Rounding to the nearest number.

4.3.9 e_imrdrv_rotate_mode_t

```
typedef enum {
        IMRDRV_ROTATE_MODE_INVALID          = -1,
        IMRDRV_ROTATE_MODE_UNIT_BLOCKS      = 0,
        IMRDRV_ROTATE_MODE_INTERPOLATING    = 1
} e_imrdrv_rotate_mode_t;
```

Table 4-9 Enumerator of ***e_imrdrv_rotate_mode_t***

Name	Description
IMRDRV_ROTATE_MODE_INVALID	Invalid value.
IMRDRV_ROTATE_MODE_UNIT_BLOCKS	Rotated in 2x2 block unit.
IMRDRV_ROTATE_MODE_INTERPOLATING	Rotated during interpolation.

4.3.10 e_imrdrv_cache_mode_t

```
typedef enum {
        IMRDRV_CACHE_MODE_INVALID          = -1,
        IMRDRV_CACHE_MODE_NORMAL           = 0,
        IMRDRV_CACHE_MODE_NON_BLOCKING     = 1
} e_imrdrv_cache_mode_t;
```

Table 4-10 Enumerator of ***e_imrdrv_cache_mode_t***

Name	Description
IMRDRV_CACHE_MODE_INVALID	Invalid value.
IMRDRV_CACHE_MODE_NORMAL	Normal mode.
IMRDRV_CACHE_MODE_NON_BLOCKING	Non-blocking mode.

4.3.11 e_imrdrv_double_cache_mode_t

```
typedef enum {
        IMRDRV_DOUBLE_CACHE_MODE_INVALID        = -1,
        IMRDRV_DOUBLE_CACHE_MODE_SINGLE         = 0,
        IMRDRV_DOUBLE_CACHE_MODE_DOUBLE         = 1
} e_imrdrv_double_cache_mode_t;
```

Table 4-11 Enumerator of ***e_imrdrv_double_cache_mode_t***

Name	Description
IMRDRV_DOUBLE_CACHE_MODE_INVALID	Invalid value.
IMRDRV_DOUBLE_CACHE_MODE_SINGLE	Single Cache Mode.
IMRDRV_DOUBLE_CACHE_MODE_DOUBLE	Double Cache Mode.

4.4 Structures

The structure used in this software is shown below.

4.4.1 st_imrdrv_attr_param_t

```
typedef struct {
    st_imrdrv_dl_t              dl_data;
    st_imrdrv_data_t            src_data;
    st_imrdrv_data_t            dst_data;
    st_imrdrv_triangle_mode_t   triangle_mode;
} st_imrdrv_attr_param_t;
```

Table 4-12 Structure of ***st_imrdrv_attr_param_t***

Name	In/Out	Description
dl_data	In	Display List data for execute
src_data	In	The source image data to be rendered
dst_data	In	Destination image data to be rendered
triangle_mode	In	Triangle mode

Table 4-13 Valid value of ***st_imrdrv_attr_param_t***

Name	Value
dl_data	Please refer to ***st_imrdrv_dl_t***
src_data	Please refer to ***st_imrdrv_data_t***
dst_data	Please refer to ***st_imrdrv_data_t***
triangle_mode	Please refer to ***st_imrdrv_triangle_mode_t***

4.4.2 st_imrdrv_initdata_t

```
typedef struct {
    void                *p_work_addr;
    uint32_t            work_size;
    e_imrdrv_channelno_t channel_no;
} st_imrdrv_initdata_t;
```

Table 4-14 Structure of ***st_imrdrv_initdata_t***

Name	In/Out	Description
p_work_addr	In	The pointer of the work area
work_size	In	Size of the work area (Byte)
channel_no	In	Number of channel to use

Table 4-15 Valid value of ***st_imrdrv_initdata_t***

Name	Value
p_work_addr	Must not NULL and 8 byte alignment (Not implement)
work_size	Greater than or equal to IMRDRV_SIZE_WORKAREA
channel_no	Please refer to ***e_imrdrv_channelno_t*** Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for invalid channel due to SoC differences.

4.4.3 st_imrdrv_data_t

```
typedef struct {
        uint32_t                    height;
        uint32_t                    width;
        uint32_t                    stride;
        uint32_t                    phys_addr;
        e_imrdrv_color_format_t     color_format;
        e_imrdrv_bpp_t              bpp;
        st_imrdrv_uv_setting_t      uv_set;
} st_imrdrv_data_t;
```

Table 4-16 Structure of ***st_imrdrv_data_t***

Name	In/Out	Description
height	In	Height of the picture(line)
width	In	Width of the picture(pixel)
stride	In	Stride of the picture (byte)
phys_addr	In	Start address of the physical memory area
color_format	In	Color format of the picture.
bpp	In	Bit depth of the picture(bit/pixel).
uv_set	In	UV image data.

Table 4-17 Valid value of ***st_imrdrv_data_t***

Name	Value
height	Within the range from 1 to 2048.
width	Within the range from 2 to 2048.
stride	256 to 8192 and must be 256 byte alignment [This value is apply for Source Image] 64 to 8192 and must be 64 byte alignment [This value is apply for Destination Image] Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the upper limit of stride supported by SoC differences.
phys_addr	256 byte alignment [This value is apply for Source Image] 64 byte alignment [This value is apply for Destination Image]
color_format	Within the range of ***e_imrdrv_color_format_t*** except for invalid value Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the color format supported by SoC differences.
bpp	Within the range of ***e_imrdrv_bpp_t*** except for invalid value Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the bit per pixel supported by SoC differences.
uv_set	Please refer to ***st_imrdrv_uv_setting_t***

4.4.4 st_imrdrv_dl_t

```
typedef struct {
        uint32_t                phys_addr;
} st_imrdrv_dl_t ;
```

Table 4-18 Structure of ***st_imrdrv_dl_t***

Name	In/Out	Description
phys_addr	In	Physical address of DL storage area

Table 4-19 Valid value of ***st_imrdrv_dl_t***

Name	Value
phys_addr	8 byte alignment

4.4.5 st_imrdrv_os_config_t

```
typedef struct {
        osal_mutex_id_t             mutex_id;
        osal_milli_sec_t            mutex_wait_period;
        e_osal_interrupt_priority_t dev_irq_priority;
} st_imrdrv_os_config_t;
```

Table 4-20 Structure of ***st_imrdrv_os_config_t***

Name	In/Out	Description
mutex_id	In	OSAL mutex ID.
mutex_wait_period	In	The wait time (msec) of locking mutex. When an interrupt is generated from IMR-LXn, the mutex lock is used during the processing of the interrupt in the IMR Driver, but the timeout time at that time is 0. User should specify the sufficient value to complete each function.
dev_irq_priority	In	Interrupt priority

Table 4-21 Valid value of ***st_imrdrv_os_config_t***

Name	Value
mutex_id	Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver".***
mutex_wait_period	Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver".***
dev_irq_priority	Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver".***

4.4.6 st_imrdrv_triangle_mode_t

```
typedef struct {
        e_imrdrv_mode_t             uvshfval;
        e_imrdrv_mode_t             shfval;
        e_imrdrv_mode_t             uvshfe;
        e_imrdrv_mode_t             shfe;
        e_imrdrv_mode_t             rde;
        e_imrdrv_mode_t             tfe;
        e_imrdrv_mode_t             tcm;
        e_imrdrv_mode_t             autosg;
        e_imrdrv_mode_t             autodg;
        e_imrdrv_mode_t             bfe;
        e_imrdrv_mode_t             tme;
} st_imrdrv_triangle_mode_t;
```

Table 4-22 Structure of ***st_imrdrv_triangle_mode_t***

Name	In/Out	Description
uvshfval	In	Specifies one bit of logical right-shifting after filtering when the extended filtering pipeline is in use. This setting is applied to the UV plane when inputting YUV422/YUV420 semi-planar only.
shfval	In	Specifies one bit of logical right-shifting after filtering when the extended filtering pipeline is in use. This setting applies to YUV422/YUV420 Semi-planar Y-planes and other formats.
uvshfe	In	Specifies whether to truncate the fractional part or to retain the specified fractional digits following filtering by the extended filtering pipeline. This setting is applied to the UV plane when inputting YUV422/YUV420 semi-planar only. IMRDRV_MODE_OFF: The fractional part is truncated after filtering. IMRDRV_MODE_ON: Some of the digits of the fractional part after filtering are retained.
shfe	In	Specifies whether to truncate the fractional part or to retain the specified fractional digits following filtering by the extended filtering pipeline. This setting applies to YUV422/YUV420 Semi-planar Y-planes and other formats. IMRDRV_MODE_OFF: The fractional part is truncated after filtering. IMRDRV_MODE_ON: Some of the digits of the fractional part after filtering are retained.
rde	In	Specifies how to round the fractional part in filtering. IMRDRV_MODE_OFF: Truncation IMRDRV_MODE_ON: Rounding up or down
tfe	In	Pyramidal Image Filtering Enable IMRDRV_MODE_OFF: Disables the filtering between two adjacent pyramidal images. IMRDRV_MODE_ON: Enables the filtering between two adjacent pyramidal images.
tcm	In	Specifies the order of the first three vertexes of the strip-type triangle with N vertexes that is drawn by the TRI instruction. Any TRI instruction not consistent with the order specified in this bit is not executed. 0: Specifies triangle vertexes counterclockwise. 1: Specifies triangle vertexes clockwise.
autosg	In	Automatic Source Coordinate Generation Mode IMRDRV_MODE_OFF: Disables the automatic source coordinate generation function. IMRDRV_MODE_ON: Enables the automatic source coordinate generation function.
autodg	In	Automatic Source Coordinate Generation Mode IMRDRV_MODE_OFF: Disables the automatic source coordinate generation function. IMRDRV_MODE_ON: Enables the automatic source coordinate generation function.
bfe	In	Bilinear Filter Enable Indicates whether bilinear filtering is used for texture mapping. IMRDRV_MODE_OFF: Bilinear filtering is not used.

		IMRDRV_MODE_ON: Bilinear filtering is used.
tme	In	Texture Mapping Enable IMRDRV_MODE_OFF: The single color specified by TRICR, TRICR2, and TRICR3 is used. IMRDRV_MODE_ON: Texture mapping is used.

Table 4-23 Valid value of ***st_imrdrv_triangle_mode_t***

Name	Value
uvshfval	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
shfval	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
uvshfe	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
shfe	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
rde	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
tfe	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
tcm	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
autosg	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
autodg	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
bfe	Within the range of ***e_imrdrv_mode_t*** except for invalid value.
tme	Within the range of ***e_imrdrv_mode_t*** except for invalid value.

Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the triangle mode supported by SoC differences.

4.4.7 st_imrdrv_attr_rs_unit_t

```
typedef struct {
        e_imrdrv_scaling_down_filter_t  scaling_down_filter;
        e_imrdrv_rounding_mode_t        rounding_mode;
        e_imrdrv_bpp_t                  dst_bpp;
        uint32_t                        dst_addr_norotate;
        uint32_t                        dst_addr_rotate;
        uint32_t                        dst_stride_norotate;
        uint32_t                        dst_stride_rotate;
} st_imrdrv_attr_rs_unit_t;
```

Table 4-24 Structure of ***st_imrdrv_attr_rs_unit_t***

Name	In/Out	Description
scaling_down_filter	In	Scaling down filter.
rounding_mode	In	Rounding mode.
dst_bpp	In	Bit depth of destination image.
dst_addr_norotate	In	Destination address for not rotate.
dst_addr_rotate	In	Destination address for rotate.
dst_stride_norotate	In	Destination stride for not rotate.
dst_stride_rotate	In	Destination stride for rotate.

Table 4-25 Valid value of ***st_imrdrv_attr_rs_unit_t***

Name	Value
scaling_down_filter	Within the range of ***e_imrdrv_scaling_down_filter_t*** except for invalid scaling down filter. Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the scaling down filter supported by SoC differences.
rounding_mode	Within the range of ***e_imrdrv_rounding_mode***_t except for invalid rounding mode. Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the rounding mode supported by SoC differences.
dst_bpp	Within the range of ***e_imrdrv_bpp***_t except for invalid output image format and 32bpp format. Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the bit per pixel supported by SoC differences.
dst_addr_norotate	The value should be specific byte aligned. Please Refer to Table 4-26 ***IMRDRV_ATTR_RS_ADDR_INVALID*** means an invalid value.
dst_addr_rotate	The value should be specific byte aligned. Please Refer to Table 4-27 ***IMRDRV_ATTR_RS_ADDR_INVALID*** means an invalid value.
dst_stride_norotate	Within the range from 1 to 4096. The value should be specific byte aligned. Please Refer to Table 4-26 0 means an invalid value.
dst_stride_rotate	Within the range from 1 to 4096. The value should be specific byte aligned. Please Refer to Table 4-27 0 means an invalid value.

Table 4-26 Valid alignment for not rotate

Index of rs_param[]	Destination bpp (rs_param[].dst_bpp)				Valid align
	IMRDRV_ATTR_RS_IDX_ONE_ONE	***IMRDRV_ATTR_RS_IDX_ONE_HALF***	***IMRDRV_ATTR_RS_IDX_ONE_FOURTH***	***IMRDRV_ATTR_RS_IDX_ONE_EIGHTH***	
IMRDRV_ATTR_RS_IDX_ONE_ONE	N/A	N/A	N/A	N/A	256byte
IMRDRV_ATTR_RS_IDX_ONE_HALF	N/A	8bpp	8bpp	8bpp	256byte
	N/A		Other than above		128byte
	N/A	Other than 8bpp	N/A	N/A	256byte
IMRDRV_ATTR_RS_IDX_ONE_FOURTH	N/A	N/A	8bpp	N/A	64byte
	N/A	N/A	Other than 8bpp	N/A	128byte
IMRDRV_ATTR_RS_IDX_ONE_EIGHTH	N/A	N/A	N/A	8bpp	32byte
	N/A	N/A	N/A	Other than 8bpp	64byte

Table 4-27 Valid alignment for rotate

Index of rs_param[]	Destination bpp (rs_param[].dst_bpp)				Valid align
	IMRDRV_ATTR_RS_IDX_ONE_ONE	***IMRDRV_ATTR_RS_IDX_ONE_HALF***	***IMRDRV_ATTR_RS_IDX_ONE_FOURTH***	***IMRDRV_ATTR_RS_IDX_ONE_EIGHTH***	
IMRDRV_ATTR_RS_IDX_ONE_ONE	N/A	N/A	N/A	N/A	256byte
IMRDRV_ATTR_RS_IDX_ONE_HALF	N/A	8bpp	N/A	N/A	128byte
	N/A	Other than 8bpp	N/A	N/A	256byte
IMRDRV_ATTR_RS_IDX_ONE_FOURTH	N/A	N/A	8bpp	N/A	64byte
	N/A	N/A	Other than 8bpp	N/A	128byte
IMRDRV_ATTR_RS_IDX_ONE_EIGHTH	N/A	N/A	N/A	8bpp	32byte
	N/A	N/A	N/A	Other than 8bpp	64byte

For example, if rs_param[IMRDRV_ATTR_RS_IDX_ONE_HALF].dst_bpp is IMRDRV_BPP_8 and rs_param[IMRDRV_ATTR_RS_IDX_ONE_FOURTH].dst_bpp is not IMRDRV_BPP_8, then dst_addr_norotate, dst_stride_norotate, dst_addr_rotate, and dst_stride_rotate in rs_param[IMRDRV_ATTR_RS_IDX_ONE_HALF] must be aligned to 128 byte.

Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the attribute of rotator and scaler by size supported by SoC differences.

4.4.8 st_imrdrv_attr_rs_param_t

```
typedef struct {
        uint32_t                valid_rs_idx;
        st_imrdrv_attr_rs_unit_t    rs_param[IMRDRV_ATTR_RS_IDX_MAX_NUM]
        e_imrdrv_rotate_mode_t      rotate_mode_yuv422
} st_imrdrv_attr_rs_param_t;
```

Table 4-28 Structure of ***st_imrdrv_attr_rs_param_t***

Name	In/Out	Description
valid_rs_idx	In	Valid index of attribute of rotator and scaler.
rs_param	In	Attribute of rotator and scaler.
rotate_mode_yuv422	In	Algorithm to rotate the UV plane in YUV422 format. If color format is not YUV422, set IMRDRV_ROTATE_MODE_UNIT_BLOCKS.

Table 4-29 Valid value of ***st_imrdrv_attr_rs_param_t***

Name	Value
valid_rs_idx	Less than ***IMRDRV_ATTR_RS_IDX_MAX_NUM.***
rs_param	Refer to ***st_imrdrv_attr_rs_unit_t***
rotate_mode_yuv422	Within the range of ***e_imrdrv_rotate_mode_t*** except for invalid rotate mode. Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the rotate mode supported by SoC differences.

4.4.9 st_imrdrv_attr_cache_mode_t

```
typedef struct {
        e_imrdrv_cache_mode_t           cache_mode;
        e_imrdrv_double_cache_mode_t    double_cache_mode;
} st_imrdrv_attr_cache_mode_t;
```

Table 4-30 Structure of ***st_imrdrv_attr_cache_mode_t***

Name	In/Out	Description
cache_mode	In	The cache mode.
double_cache_mode	In	The double cache mode.

Table 4-31 Valid value of ***st_imrdrv_attr_cache_mode_t***

Name	Value
cache_mode	Within the range of ***e_imrdrv_cache_mode_t*** except for invalid cache mode.
double_cache_mode	Within the range of ***e_imrdrv_double_cache_mode_t*** except for invalid cache mode. Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the double cache mode supported by SoC differences.

4.4.10 st_imrdrv_uv_setting_t

```
typedef struct {
    uint32_t            uv_stride;
    uint32_t            uv_phys_addr;
    e_imrdrv_bpp_t      uv_bpp;
} st_imrdrv_uv_setting_t;
```

Table 4-32 Structure of ***st_imrdrv_uv_setting_t***

Name	In/Out	Description
uv_stride	In	Stride of the picture (byte)
uv_phys_addr	In	Start address of the physical memory area
uv_bpp	In	Bit depth of the picture(bit/pixel)

Table 4-33 Valid value of ***st_imrdrv_uv_setting_t***

Name	Value
uv_stride	256 to 65536 and must be 256 byte alignment [This value is apply for semi-planar format.]
uv_phys_addr	256 byte alignment (Except 0) [This value is apply for semi-planar format.]
uv_bpp	Within the range of ***e_imrdrv_bpp_t*** except for invalid value [This value is apply for semi-planar format.]

Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the uv setting supported by SoC differences.

4.4.11 st_imrdrv_attr_dyp_param_t

```
typedef struct {
    uint32_t            stride;
    uint32_t            phys_addr;
} st_imrdrv_attr_dyp_param_t;
```

Table 4-34 Structure of ***st_imrdrv_attr_dyp_param_t***

Name	In/Out	Description
stride	In	Stride of the picture (byte)
phys_addr	In	Start address of the physical memory area

Table 4-35 Valid value of ***st_imrdrv_attr_dyp_param_t***

Name	Value
stride	256 to 65536 and must be 256 byte alignment [This value is apply for Second Y Plane Image]
phys_addr	256 byte alignment (Except 0) [This value is apply for Second Y Plane Image]

Please refer to chapter 3.1.1.1 of ***"Product Information of IMR Driver"*** for the attribute of DYP parameter supported by SoC differences.

4.4.12 st_imrdrv_version_t

```
typedef struct {
        uint32_t                major;
        uint32_t                minor;
        uint32_t                patch;
} st_imrdrv_version_t;
```

Table 4-36 Structure of ***st_imrdrv_version_t***

Name	In/Out	Description
major	Out	Major version
minor	Out	Minor version
patch	Out	Patch version

5. Functions

5.1 Function List

5.1.1 IMR Driver function

Table 5-1 List of IMR Driver function

Function Name	Purpose	Mandatory Function
R_IMRDRV_Init	Initialize this driver.	Mandatory
R_IMRDRV_Quit	Uninitialize this driver.	Mandatory
R_IMRDRV_Start	Start this driver.	Mandatory
R_IMRDRV_Stop	Stop this driver	Mandatory
R_IMRDRV_Execute	Execute DL.	Mandatory
R_IMRDRV_AttrSetParam	Parameter settings for the Execute function	Mandatory
R_IMRDRV_GetVersion	Get version	Optional
R_IMRDRV_AttrSetRotatorScaler	Set attribute of rotator and scaler	Optional
R_IMRDRV_AttrSetCacheMode	Set attribute of cache mode	Optional
R_IMRDRV_AttrSetDoubleLumaPlane	Set attribute of double luminance plane	Optional

5.1.2 Callback function

Table 5-2 List of Callback function

Function prototype	Purpose
p_imrdrv_callback_func_t	Notify interrupt.

5.2 IMR Driver function

5.2.1 IMR Control function

5.2.1.1 R_IMRDRV_Init

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_Init (
        const st_imrdrv_initdata_t *const p_initdata,
        const st_imrdrv_os_config_t *const p_os_config,
        const p_imrdrv_callback_func_t callback_func,
        void *const p_callback_arg,
        imrdrv_ctrl_handle_t *const p_handle
);
```

<Input Parameters>

Parameter	Description
p_os_config	The pointer of data used by OSAL. Must not be NULL.
callback_func	Callback function Must not be NULL.
p_callback_arg	The pointer of argument of callback function. This argument is optional.

<Input-Output Parameters>

Parameter	Description
p_initdata	Initialize structure. Must not be NULL.

<Output Parameters>

Parameter	Description
p_handle	The pointer of *imrdrv_ctrl_handle_t* Must not be NULL.

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	■Permitted / □Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	□Permitted / ■Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_FAIL

<Description>

This function initializes channel for IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.

This function clears the work area.
This function sets initialize data of ***st_imrdrv_initdata_t***, ***p_imrdrv_callback_func_t*** and void (the pointer of argument of the callback function) to work area of ***imrdrv_ctrl_handle_t.***
This function creates the semaphore resource.
This function changes state to ***Initialized State*** for specified channel.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument p_initdata is NULL.
  - The argument p_os_config is NULL.
  - The argument p_os_config->mutex_wait_period is less than 0.
  - The argument callback_func is NULL.
  - The argument p_handle is NULL.
  - The member of argument p_initdata is invalid value.
- If it corresponds to following conditions, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.

<Notes>

After call, this API must not be called until R_IMRDRV_Quit is completed.

5.2.1.2 R_IMRDRV_Quit

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_Quit (
        const imrdrv_ctrl_handle_t handle
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	■Permitted / □Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	□Permitted / ■Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function uninitializes channel for IMR Driver.

This function check if specified channel is in permitted state before executing body of this function.

This function destroys the semaphore resource.
This function changes state to ***Uninitialized State*** for the channel.
This function cleans work area of ***imrdrv_ctrl_handle_t***.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
- If it corresponds to following conditions, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following conditions, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.

<Notes>

None

5.2.1.3 R_IMRDRV_Start

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_Start (
        const imrdrv_ctrl_handle_t handle
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	■Permitted / □Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	□Permitted / ■Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_IO
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function starts channel for IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.

This function creates the IO resource.
This function enables clock for IMR-LXn channel.
This function executes the software reset for IMR-LXn channel.
This function enables all the interrupt for IMR-LXn channel, but all interrupt is masked.
This function registers the callback function and interrupt handler.
This function changes state to ***Ready State*** for the channel.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
- If it corresponds to following conditions, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following conditions, ***IMRDRV_ERROR_FAIL*** is returned.
  - HW error or OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following conditions, ***IMRDRV_ERROR_IO*** is returned.
  - The register access error.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

None

5.2.1.4 R_IMRDRV_Stop

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_Stop (
        const imrdrv_ctrl_handle_t handle
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	■Permitted / □Prohibited
Attribute Set State	■Permitted / □Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_IO
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function stops channel for IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.

This function disables all the interrupt for IMR-LXn channel.
This function disables clock for IMR-LXn channel.
This function destroys the IO resource.
This function changes state to ***Initialized State*** for the channel.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - HW error or OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following condition, ***IMRDRV_ERROR_IO*** is returned.
  - The register access error.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

None

5.2.1.5 R_IMRDRV_Execute

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_Execute (
        const imrdrv_ctrl_handle_t handle
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	□Synchronous function / ■Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	■Permitted / □Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_CONF (Not implement)
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_IO
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function sets parameter to the register and executes the DL.

This function checks if specified channel is in permitted state before executing body of this function.

This function executes the software reset. for IMR-LXn channel.
This function sets the parameter of work area to the register.
This function executes the DL.
The IMR driver calls the callback function specified by ***p_imrdrv_callback_func_t*** when the interrupt occurred.
If this function does not end normally, the IMR driver will not call the callback function.

The IMR driver changes the state to ***Active State*** for the channel.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
- If it corresponds to following condition, ***IMRDRV_ERROR_CONF*** is returned.
  - The combination of attributes set by ***R_IMRDRV_AttrSetParam***, ***R_IMRDRV_AttrSetRotatorScaler***, ***R_IMRDRV_AttrSetCacheMode*** or ***R_IMRDRV_AttrSetDoubleLumaPlane*** is invalid..
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - HW error or OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following condition, ***IMRDRV_ERROR_IO*** is returned.
  - The register access error.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

None

5.2.1.6 R_IMRDRV_AttrSetParam

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_AttrSetParam (
        const imrdrv_ctrl_handle_t handle,
        const st_imrdrv_attr_param_t *const p_param
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL
p_param	The pointer of *st_imrdrv_attr_param_t* Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	■Permitted / □Prohibited
Attribute Set State	□Permitted / ■Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function sets specified parameter to the work area of IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.
The IMR driver changes the state to ***Attribute set State*** for the channel.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following conditions, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
  - The argument p_param is NULL.
  - The member of argument p_param is invalid value.
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

None

5.2.1.7 R_IMRDRV_GetVersion

<Function Prototypes>

```
const st_imrdrv_version_t R_IMRDRV_GetVersion (
                void
);
```

<Input Parameters>

None

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	□Yes / ■No

<Return Value>

Pointer of structure ***st_imrdrv_version_t*** is returned.

<Description>

This function returns IMR driver version.

<Notes>

None

5.2.1.8 R_IMRDRV_AttrSetRotatorScaler

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_AttrSetRotatorScaler (
        const imrdrv_ctrl_handle_t handle,
        const st_imrdrv_attr_rs_param_t *const p_param
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL
p_param	Attribute parameter Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function /□Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	■Permitted / □Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function sets specified parameter of rotating and scaling to the work area of IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.
This function does not change state.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
  - The argument p_param is NULL.
  - The member of argument p_param is invalid value.
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for the rotator and scaler supported by SoC differences.

5.2.1.9 R_IMRDRV_AttrSetCacheMode

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_AttrSetCacheMode (
        const imrdrv_ctrl_handle_t handle,
        const st_imrdrv_attr_cache_mode_t *const p_param
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL
p_param	Attribute parameter Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function /□Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	□Permitted / ■Prohibited
Attribute Set State	■Permitted / □Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function sets specified parameter of cache mode to the work area of IMR Driver.

This function checks if specified channel is in permitted state before executing body of this function.
This function does not change state.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following condition, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
  - The argument p_param is NULL.
  - The member of argument p_param is invalid value.
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for the cache mode supported by SoC differences.

5.2.1.10 R_IMRDRV_AttrSetDoubleLumaPlane

<Function Prototypes>

```
e_imrdrv_errorcode_t R_IMRDRV_AttrSetDoubleLumaPlane (
        const imrdrv_ctrl_handle_t handle,
        const st_imrdrv_attr_dyp_param_t *const p_param
);
```

<Input Parameters>

Parameter	Description
handle	Control structure handle Valid range: Must not be NULL
*p_param	The pointer of *st_imrdrv_attr_dyp_param_t* Valid range: Must not be NULL

<Input-Output Parameters>

None

<Output Parameters>

None

<Function Attribute>

Attributes	Value
Categories	■Synchronous function / □Asynchronous function
Call from interrupt	□Permitted / ■Prohibited
Call from callback	□Permitted / ■Prohibited
Reentrant	□Permitted / ■Prohibited
Driver state restriction	■Yes(see Executable State) / □No

<Executable State>

Library State	Permission
Uninitialized State	□Permitted / ■Prohibited
Initialized State	□Permitted / ■Prohibited
Ready State	□Permitted /■Prohibited
Attribute Set State	■Permitted / □Prohibited
Active State	□Permitted / ■Prohibited

<Return Value>

IMRDRV_ERROR_OK
IMRDRV_ERROR_PAR
IMRDRV_ERROR_TIME
IMRDRV_ERROR_FAIL
IMRDRV_ERROR_STATE
IMRDRV_ERROR_HANDLE (Not implement)

<Description>

This function sets specified parameter of double luminance plane to the work area of IMR Driver.

This function check if specified channel is in permitted state before executing body of this function.
This function does not change state.
This function is exclusively controlled by the semaphore.

- When the process of this function ends normally, ***IMRDRV_ERROR_OK*** is returned.
- If it corresponds to following conditions, ***IMRDRV_ERROR_PAR*** is returned.
  - The argument handle is NULL.
  - The argument p_param is NULL.
  - The member of argument p_param is invalid value.
- If it corresponds to following condition, ***IMRDRV_ERROR_STATE*** is returned.
  - Illegal state transition.
- If it corresponds to following condition, ***IMRDRV_ERROR_FAIL*** is returned.
  - OS system call error.
  - Lock the semaphore resource fails.
  - Unlock the semaphore resource fails.
- If it corresponds to following conditions, ***IMRDRV_ERROR_TIME*** is returned.
  - Lock the semaphore resource timeouts.

<Notes>

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for the DYP supported by SoC differences.

5.3 Callback function

5.3.1 p_imrdrv_callback_func_t

<Function Prototypes>

```
typedef int32_t (*p_imrdrv_callback_func_t)(
    const e_imrdrv_errorcode_t    ret_code,
    const uint32_t    details_code,
    void  *const p_callback_arg
 );
```

<Input Parameters>

None.

<Input-Output Parameters>

None.

<Output Parameters>

Parameter	Description
ret_code	The result of executing the DL.
details_code	The factor of the interrupt for DL execution.
p_callback_arg	The pointer of argument (If user specifies with ***R_IMRDRV_Init***)

<Function Attribute>

NA.

<Executable State>

None.

<Return Value>

0 : This means process of callback function is succeed.
Other than above : This means process of callback function is failed.

If the return value of this callback is other than zero, the user should reboot the system. IMR driver disables interrupts in order not to generate unnecessary interrupts in the future.

<Description>

This is the function prototype for callback. The callback registration is performed by ***R_IMRDRV_Init***.
The user can confirm what interrupts were occurred by details_code in ***p_imrdrv_callback_func_t***.
The callback function is performed by task context.
The callback function processing should be minimized.

<Notes>

Don't run any other IMR Driver API in this callback.

To check details_code, mask with the following definition values.

Table 5-2 Mask value of details_code

Interrupt	Definition values
TRAP	*IMRDRV_SR_TRA*
IER	*IMRDRV_SR_IER*
WUPOVF	*IMRDRV_SR_WOVF*
WUPERR	*IMRDRV_SR_WERR*

Please refer to chapter 3.1.1.2 of ***"Product Information of IMR Driver"*** for SoC differences of WUPOVF and WUPERR.

When the DL execution ends normally, TRAP is set to details_code.
If details_code is not TRAP, an error occurred.
In this case, execute system reset.

Please Refer to ***Table 1-2*** for details of each interrupt.

Index

Revision History	IMR Driver User's Manual: Software

Rev.	Date	Description	
		Page	Summary
0.10E	Dec14, 2020	ALL	Newly created as User's Manual of IMRDRV for xOS2.
0.11E	Feb 26, 2021	8	Modify Table 2-2 to add "Data 16bpp and 32bpp".
0.12E	Mar 16, 2021	8	Modify Chapter 2.4 to Refer Product Information document. Change title of document from R-Car V3U IMR Driver to IMR Driver.
0.13E	Apr 09, 2021	ALL	Update User's Manual structure, due to unifying the purpose and writing style.
	Apr 12, 2021	3, 18	Add sentence "[valid only V3H, V3H_2 and V3U]" for "IMRDRV_SR_WERR or IMRDRV_SR_WOVF"
		21	Update enum e_imrdrv_channelno_t name and add Invalid channel enum.
		22	Remove error code IMRDRV_ERROR_MEM from e_imrdrv_errorcode_t and update enum numbers order.
		24	Update IF API. Remove error code IMRDRV_ERROR_STATE from all R_IMRDRV_Init.
		24-36	Add "const" to the argument to follow the coding guidelines.
		37	Renamed the callback function to follow the coding guidelines.
	Apr 15, 2021	15	Change stride valid value from "128 to 8192" to "256 to 8192 and must be 256 alignment [This value is apply for Source Image), 64 to 8192 and must be 64 alignment [This value is apply for Destination Image)".
		15	Change the phys_addr valid value from "256 alignment (Except for 0)" to 256 alignment (Except 0) [This value is apply for Source Image), 64 alignment (Except 0) [This value is apply for Destination Image)""
		15	Change the bpp valid value from "0 to 16" to "8 / 10 / 12/ 14/ 16 [valid only on V3H, V3H_2 and V3U], 8 / 10 / 12 / 16 [valid only on V3M]"
		15	Change the height valid value from "2 to 2048" to "1 to 2048".
		16	Change the phys_addr of dl valid value from "256 alignment (Except for 0)" to "8 alignment (Except 0)"
		34	In description of R_IMRDRV_AttrSetParam() at the instruction for setting opt and bpp.
	Apr 16, 2021	28-29	Add return error code IMRDRV_ERROR_TIME for R_IMRDRV_Start().
	May 10, 2021	17	Update description of mutex_wait_period
		24	Modify the 4th argument of R_IMRDRV_Init: const void -> void.
		37	Add: "Note: Don't run any other API while calling this callback."
	May 19, 2021	7	Change Prefix.
		19	Add "imrdrv_ctrl_handle_t" to "4.1 Data Type".
		21	Change Description of IMRDRV_SIZE_WORKAREA to "Lower limit of work area size.".
		22	Add "4.3.4 e_imrdrv_state_t".
		24	Remove "st_imrdrv_ctrl_handle_t".
		25-26	Modify Value or Description according to Component Specification.
		29-39	Change Parameters from "const st_imrdrv_ctrl_handle_t. *const p_handle" to "const imrdrv_ctrl_handle_t. handle" and correct of type name due to the change.
		30	Add error handling "The argument p_os_config->mutex_wait_period is less than IMRDRV_MUTEX_WAIT_TIME_MIN" to return IMRDRV_ERROR_PAR..

		31-36	Remove error code MRDRV_ERROR_HANDLE in R_IMRDRV_Quit, R_IMRDRV_Start and R_IMRDRV_Stop.
		33-40	Add sequence of lock/unlock semaphore resource in R_IMRDRV_Start, R_IMRDRV_Stop, R_IMRDRV_Execute and R_IMRDRV_AttrSetParam. Updated error handling (IMRDRV_ERROR_FAIL and IMRDRV_ERROR_TIME) with the above.
		42	Change Input parameter to “void *const p_callback_arg”.
	May 21, 2021	ALL	Font color “red” indicating the corrected part.
		14	Update Figure 3-2.
		19	Update the description to unify to task context.
		5, 20, 21	Update “Description” to add sentence to refer to Product Information.
		20	Remove item of “Definition” in section 4.1.
		20	Remove non-user-published macros in section 4.2.
		23	Change Member name of e_imrdrv_exec_opt_t.
		25	Update “Description” of work_size to add “(Byte)”.
		27	Update “Value” to specify the value during T.B.D.
		31	Change <Notes> to add limitations in R_IMRDRV_Init.
		31	Update Description to change sentences of set initialize data to work area.
		30,40, 43	Update Parameter to remove information other than parameter name.
		43	Update Description about task context to remove “of IMR Driver”.
0.14E	Jun 11, 2021	9, 10, 23,36, 37,39, 41,43, 45	Added status of Attribute set State - Update Figure 3-1 - Change description of state machine when R_IMRDRV_AttrSetParam is called. - Added 3.1.4 Attribute set State - Added IMRDRV_STATE_ATTR_SET to 4.3.4 e_imrdrv_state_t - Change executable State in 5.2.1.1 R_IMRDRV_Init, 5.2.1.2 R_IMRDRV_Quit, 5.2.1.3 R_IMRDRV_Start, 5.2.1.4 R_IMRDRV_Stop, 5.2.1.5 R_IMRDRV_Execute, and 5.2.1.6 R_IMRDRV_AttrSetParam
		12	Change Figure 3-2 Function Flow on IMR driver. - Added function R_IMRDRV_AttrSetRotorScaler - Added function R_IMRDRV_AttrSetCacheMode
		20	Added definitions related rotating and scaling - IMRDRV_ATTR_RS_IDX_ONE_ONE - IMRDRV_ATTR_RS_IDX_ONE_HALF - IMRDRV_ATTR_RS_IDX_ONE_FOURTH - IMRDRV_ATTR_RS_IDX_ONE_EIGHTH - IMRDRV_ATTR_RS_IDX_MAX_NUM - IMRDRV_ATTR_RS_ADDR_INVALID

		23, 24, 27	Change parameters related R_IMRDRV_AttrSetParam - Added enumeration 4.3.5 e_imrdrv_color_format - Added enumeration 4.3.6 e_imrdrv_bpp_t - Added enumeration 4.3.7 e_imrdrv_mode_t - Added member “triangle_mode” to 4.4.1 st_imrdrv_attr_param_t - Deleted member “opt” from 4.4.1 st_imrdrv_attr_param_t - Deleted member “interrupt_info” from 4.4.1 st_imrdrv_attr_param_t - Added member “color_format” to 4.4.3 st_imrdrv_data_t - Added member “bpp” to 4.4.3 st_imrdrv_data_t - Changed the valid value of “phys_addr” in 4.4.4 st_imrdrv_dl_t - Deleted structure 4.4.6 st_imrdrv_interrupt_info_t - Added structure 4.4.6 st_imrdrv_triangle_mode_t
		25	Added function of rotating and scaling. - Added enumeration 4.3.8 e_imrdrv_scaling_down_filter_t - Added enumeration 4.3.9 e_imrdrv_rounding_mode_t - Added structure 4.4.7 st_imrdrv_attr_rs_unit_t - Added structure 4.4.7 st_imrdrv_attr_rs_param_t - Added function 5.2.1.8 R_IMRDRV_AttrSetRotatorScaler
		26	Added the function of cache mode setting. - Added enumeration 4.3.10 e_imrdrv_cache_mode_t - Added enumeration 4.3.11 e_imrdrv_double_cache_mode_t - Added structure 4.4.8 st_imrdrv_attr_cache_mode_t - Added structure 4.4.8 st_imrdrv_attr_cache_mode_t - Added function 5.2.1.9 R_IMRDRV_AttrSetCacheMode
		18,19, 23,24, 25,28, 30,32	Add sentence to refer to Product Information. - IMRDRV_ATTR_RS_IDX_ONE_ONE - IMRDRV_ATTR_RS_IDX_ONE_HALF - IMRDRV_ATTR_RS_IDX_ONE_FOURTH - IMRDRV_ATTR_RS_IDX_ONE_EIGHTH - IMRDRV_ATTR_RS_IDX_MAX_NUM - IMRDRV_ATTR_RS_ADDR_INVALID - 4.3.5 e_imrdrv_color_format_t - 4.3.6 e_imrdrv_bpp_t - 4.3.8 e_imrdrv_scaling_down_filter_t - 4.3.9 e_imrdrv_rounding_mode_t - 4.3.10 e_imrdrv_cache_mode_t - 4.3.11 e_imrdrv_double_cache_mode_t - 4.4.3 st_imrdrv_data_t - 4.4.6 st_imrdrv_triangle_mode_t - 4.4.7 st_imrdrv_attr_rs_unit_t
		44	Modify the status changed by 5.2.1.5 R_IMRDRV_Execute
		19	Added definitions related callback function. - IMRDRV_SR_WERR - IMRDRV_SR_WOVF - IMRDRV_SR_IER - IMRDRV_SR_TRA

		52,53	Modify the description of parameter in callback function. - Delete reference to Definition values in description of 5.3.1 p_imrdrv_callback_func_t - Add details of the factor of the interrupt
	Jun 16, 2021	3,4,5,6,13,20,21,23,24,25,26,27,28,29,31,32,33,50,54	Change the text of the reference to Product Information. Add chapter of Product Information. - Table エラー! 指定したスタイルは使われていません。-1 Interrupt List - 1.2 References - 2.1 Summary Specification - 2.4 Image Data Specifications - 4.4.5 st_imrdrv_os_config_t - 4.4.6 st_imrdrv_triangle_mode_t - 4.4.7 st_imrdrv_attr_rs_unit_t Delete the reference to Product Information. - IMRDRV_ATTR_RS_IDX_ONE_ONE - IMRDRV_ATTR_RS_IDX_ONE_HALF - IMRDRV_ATTR_RS_IDX_ONE_FOURTH - IMRDRV_ATTR_RS_IDX_ONE_EIGHTH - IMRDRV_ATTR_RS_IDX_MAX_NUM - IMRDRV_ATTR_RS_ADDR_INVALID - 4.3.1 e_imrdrv_channelno_t - 4.3.4 e_imrdrv_color_format_t - 4.3.5 e_imrdrv_bpp_t - 4.3.7 e_imrdrv_scaling_down_filter_t - 4.3.8 e_imrdrv_rounding_mode_t - 4.3.9 e_imrdrv_cache_mode_t - 4.3.10 e_imrdrv_double_cache_mode_t Add text of the reference to Product Information. - 4.4.2 st_imrdrv_initdata_t - 4.4.3 st_imrdrv_data_t - 4.4.7 st_imrdrv_attr_rs_unit_t - 4.4.9 st_imrdrv_attr_cache_mode_t - 5.2.1.8 R_IMRDRV_AttrSetRotatorScaler - 5.3.1 p_imrdrv_callback_func_t
		12,13,14	Change Function Flow for optional functions. - Remove optional function in 3.2.1 Basic Function Flow - Add 3.2.2 Optional Function Flow (rotator and scaler) - Add 3.2.3 Optional Function Flow (cache mode)
		28,29	Add unit of alignment in 4.4.3 st_imrdrv_data_t and 4.4.4 st_imrdrv_dl_t.

		41,43, 45,47, 50,52	Change the expression regarding the use of semaphores in following API. - 5.2.1.3 R_IMRDRV_Start - 5.2.1.4 R_IMRDRV_Stop - 5.2.1.5 R_IMRDRV_Execute - 5.2.1.6 R_IMRDRV_AttrSetParam - 5.2.1.8 R_IMRDRV_AttrSetRotatorScaler - 5.2.1.9 R_IMRDRV_AttrSetCacheMode
		45	Add the condition not to callback in 5.2.1.5 R_IMRDRV_Execute.
		20	Delete unused data type. - IMRDRV_YUV - IMRDRV_YUVSEP - IMRDRV_8BIT - IMRDRV_16BIT - IMRDRV_EXE_MODE_TEXTUREMAPPING - IMRDRV_EXE_MODE_BILINEAR_ENABLE - IMRDRV_EXE_MODE_AUTODST - IMRDRV_EXE_MODE_AUTOSRC - IMRDRV_EXE_MODE_CLOCKWISE - IMRDRV_EXE_MODE_EFP_RDE - IMRDRV_EXE_MODE_EFP_SHFE - IMRDRV_EXE_MODE_EFP_SHFVAL - e_imrdrv_exec_opt_t
		10,17, 19,27, 28,32, 33,47, 53,54	Change clerical errors. - Correct typo in 3.1 Finite-State machine - Correct typo in 3.4.3 How interrupts from IMR-LX4 are handled - Correct use driver in description of 3.4.1 User side prepare data - Correct typo in 4.4.2 st_imrdrv_initdata_t - Remove * from name of Table 4-13, 4-14 in 4.4.2 st_imrdrv_initdata_t. - Correct typo in 4.4.3 st_imrdrv_data_t - Correct the index of rs_param[] of Table 4-25 and 4-26 in 4.4.7 st_imrdrv_attr_rs_unit_t - Remove * from name of input parameter in 5.2.1.6 R_IMRDRV_AttrSetParam - Correct typo in 5.3.1 p_imrdrv_callback_func_t
		23	Change the Description of Table 4-3 in 4.3.3 e_imrdrv_state_t to add links to each states.
		36	Change table of function list. - Correct purpose of R_IMRDRV_Init and R_IMRDRV_Execute in Table 5 1 List of IMR Driver function - Correct the title line in Table 5-2 List of Callback function.
		38	Change description in 5.2.1.1R_IMRDRV_Init. - Correct the sentence of work area initialization.

		38,40, 48,51, 53	Change the condition of returning IMRDRV_ERROR_FAIL. - Remove HW error from following. - R_IMRDRV_Init - R_IMRDRV_Quit - R_IMRDRV_AttrSetParam - R_IMRDRV_AttrSetRotatorScaler - R_IMRDRV_AttrSetCacheMode
		54	Change API specifications of callback in 5.3.1 p_imrdrv_callback_func_t. - Correct description of output Parameter. - Correct restriction of other API in notes. - Added the operation when other than 0 is returned. - Change function attribute.
		15,16	Change what to do if error occurs. - Modify sentence to handle error in 3.3.3 CONF. - Add sentence to handle error in 3.3.8 TIME.
		27	Change value of size of work area in 4.4.2 st_imrdrv_initdata_t.
	Jun 21, 2021	1,2,3, 4,17,2 0,21,2 3,32,4 5,47,4 9	Modify 1.3 List of Terms - Add DYP - Change IMR-LX4 to IMR-LXn
		7,8	Added Memory Imagery of semi-planar format - Added Figure 2.4 Memory Imagery of YUV422 Semi-planar Format - Added Figure 2.5 Memory Imagery of YUV420 Semi-planar Format
		27,30, 35,36, 37	Change parameters related R_IMRDRV_AttrSetRotatorScaler. - Added enumeration 4.3.9 e_imrdrv_rotate_mode_t - Added member “uv_set” to 4.4.3 st_imrdrv_data_t - Added member “rotate_mode_yuv422” to 4.4.8 st_imrdrv_attr_rs_param_t - Added structure 4.4.10 st_imrdrv_uv_setting_t
		37,57	Added function of double luminance plane. - Added definitions IMRDRV_ATTR_DYP_ADDR_INVALID - Added definitions IMRDRV_ATTR_DYP_STRIDE_INVALID - Added Figure 3-5 Optional Function Flow (DYP) on IMR driver in 3.2.4 Optional Function Flow (DYP) - Added structure 4.4.11 st_imrdrv_attr_dyp_param_t - Added function 5.2.1.10 R_IMRDRV_AttrSetDoubleLumaPlane
		17,49	Modify the factor that returns IMRDRV_ERROR_CONF in 3.3.3 CONF and 5.2.1.5 R_IMRDRV_Execute.
0.15E	July 7, 2021	1	Updated Figure 1.1. - DLG added. - RTT added.
		5	Updated Summary Specification.
		11	Added sentence about creating DL with DLG.
	July 9, 2021	17,18	Fixed chapter title in 3.3 Error Processing.

		22, 61	Correct about callback funtion. - Modified the description about interrupt notified by callback in 4.2 Definition Values. - Modified the operation when other than 0 is returned in 5.3.1 p_imrdrv_callback_func_t.
		26,31, 34,35, 39	Correct descriptions about R_IMRDRV_AttrSetParam. - Modified the description of 4.3.4 e_imrdrv_color_format_t. - Modified the description of bpp in 4.4.3 st_imrdrv_data_t, 4.4.7 st_imrdrv_attr_rs_unit_t, and 4.4.10 st_imrdrv_uv_setting_t. - Modified the description of 4.4.6 st_imrdrv_triangle_mode_t.
		29	Correct description about R_IMRDRV_AttrSetCacheMode. - Mofied the description of 4.3.10 e_imrdrv_double_cache_mode_t.
		37	Correct description about R_IMRDRV_AttrSetRotatorScaler. - Added an example about how to read the table 4-26 and 4-27 in 4.4.7 st_imrdrv_attr_rs_unit_t.
		31.44	Change clerical errors. - Correct typo in 4.3.4 e_imrdrv_color_format_t. - Correct typo in 4.4.2 st_imrdrv_initdata_t. - Correct typo in 5.2.1.2 R_IMRDRV_Quit.
		3,4,5, 6,7, 13-16, 22-42, 50,55, 5761	Appearance adjustment.

IMR Driver User’s Manual: Software

Publication Date: Rev.0.10E Dec.14.20
Rev.0.15E July.15.21

Published by: Renesas Electronics Corporation

IMR Driver

Renesas Electronics Corporation

LLWEB-20105857