User's Manual

32

Memory Manager for Linux

User's Manual: Software

R-Car H2/M2 Series

Renesas Electronics
www.renesas.com

Rev.1.40 Jan. 2015

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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

How to Use This Manual

Purpose and Target Readers

This manual is designed to provide the user with an understanding the functions of Memory Manager for Linux. This manual is written for engineers who use Memory Manager.

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

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

Please refer to documents of software and hardware for a target system implementing Memory Manager as necessary.
The following documents are related documents. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	**Description**	**Document Title**	**Notes**
User's manual for Hardware	Hardware specifications (pin assignments, memory maps, peripheral function specifications, electrical characteristics, timing charts) and operation description Note: Refer to the application notes for details on using peripheral functions.	R-Car {H/M}2 User's Manual: Hardware	
User's manual for Software	Description of Memory Manager	Memory Manager for Linux	This manual
	Description of S3CTRL Driver	S3CTRL Driver for Linux	
	Description of R-Car H2/M2 Linux BSP	Linux Interface Specification Start-Up Guide	

Notation of Numbers and Symbols

This manual use following notation.

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

List of Abbreviations and Acronyms

Abbreviation	Full Form
MMNGR	Memory Manager
S3CTL	S3CTRL Driver
CMA	Contiguous Memory Allocator
DTV	Digital Terrestrial Television
SSP	Stream processor
MV	Motion vector
LPAE	Large Physical Address Extention

Table of Contents

Memory Manager for Linux
User's Manual: Software

Jan. 2015

1. Overview

1.1 Overview of this Software

・To bring out the performance of the multimedia H/W IP,
the following features for the memory space are required.
(1) Not only the space in the kernel management, which is shared by H/W IP and CPU,
but also the space out of the kernel management, which is used by H/W IP (Video Codec IP) only.
(2) The larger contiguous space.

・Therefore Memory Manager (hereafter called in MMNGR) provides the following memory space.
(1) The space in the kernel management, which is shared by H/W IP and CPU.
(2) The space out of the kernel management, which is user by H/W IP (Video Codec IP) only.
The space is allocated out of the kernel management, if LPAE is disabled.
However the space is allocated in the reservation area of CMA according to BSP specification,
if LPAE is enabled.
In this manual, the space is called the space out of the kernel management, if the reservation area is used.
(3) The physical contiguous space over 4MB in and out of the kernel management
by one allocation request.

・The API is provided in the user layer.

1.2 Configuration of this Software

This software consists of the following resources.

- Document
- Source code
- Sample application
- Makefile

To use this software, the following additional software which is not included in this software is required.
Details of this additional software are shown below.

- Kernel module source code

This software is distributed in based on Dual MIT/GPLv2 licenses.
Figure 1-2 shows the lists of these source files.

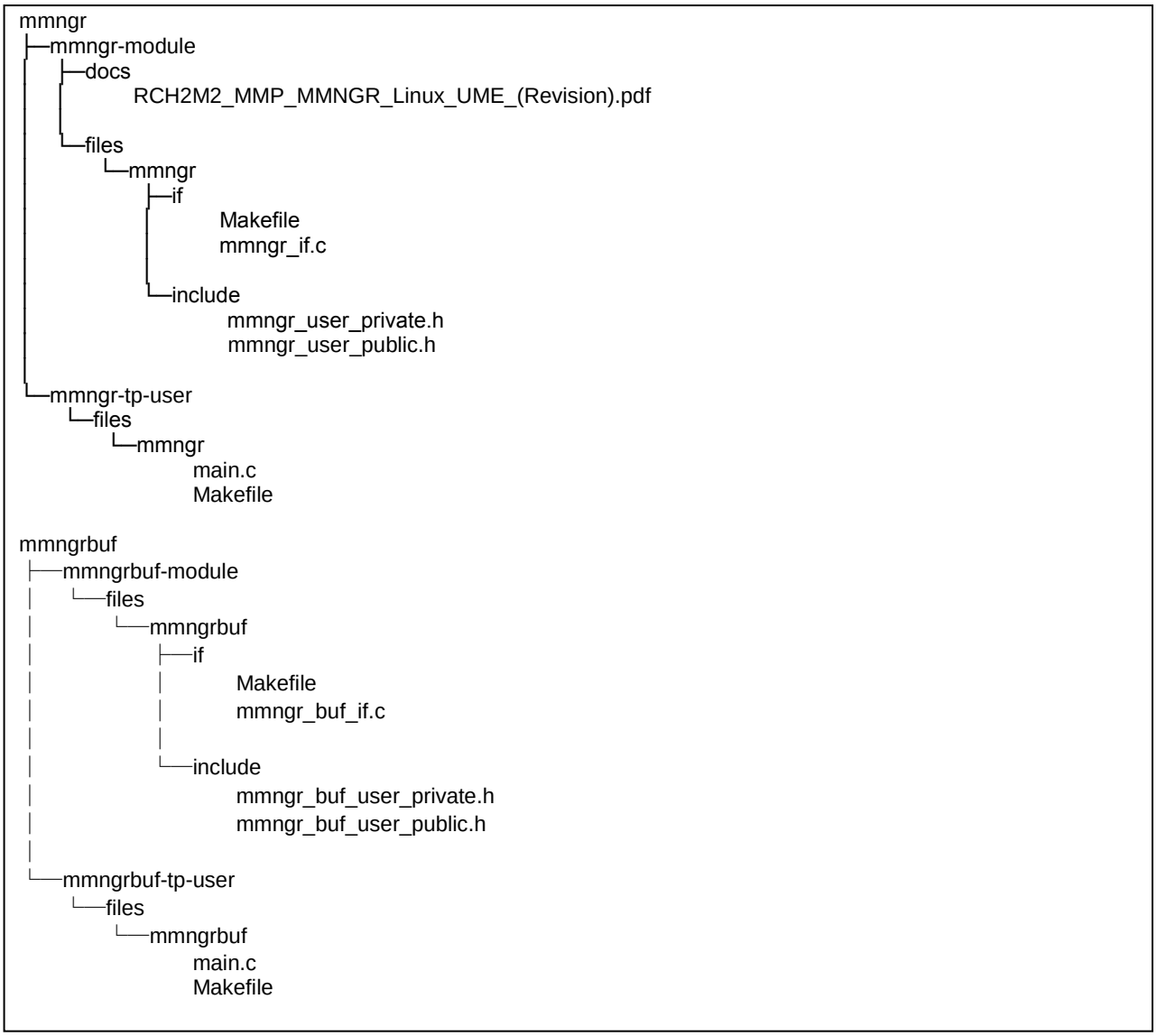


Figure 1-1 Configuration of this software

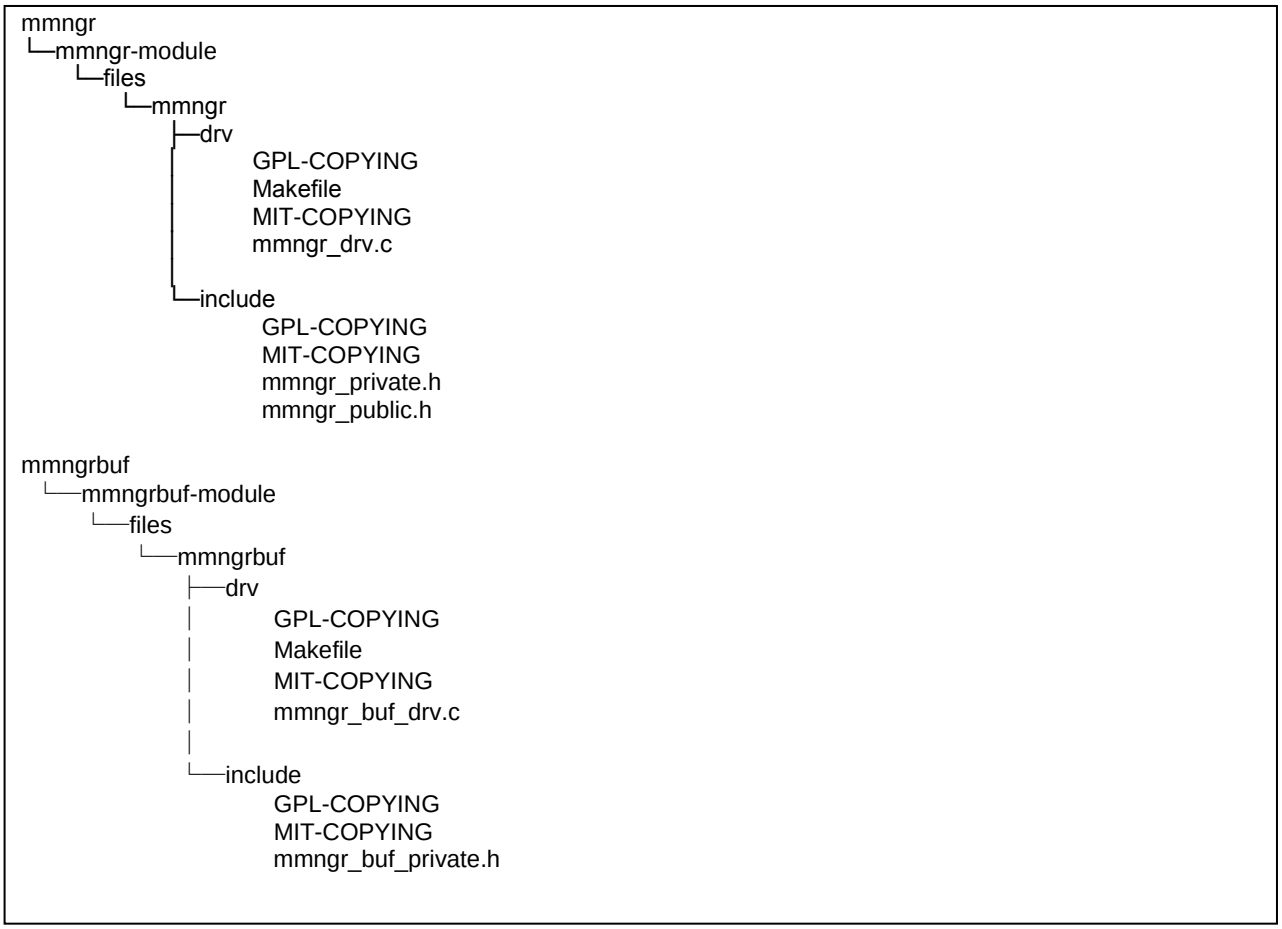


Figure 1-2 Kernel module source code not included in this software

1.3 Development Environments

This section describes the development environments for this software.

1.3.1 Hardware Development Environment

Hardware Name		Remarks
Device	R-Car H2/M2	-
Board	RENESAS LAGER/KOELSCH	-

1.3.2 Software Development Environment

Software Name	Version / Revision	Remarks
R-Car H2/M2 Linux BSP	-	-

2. Installation Procedures

2.1 Building the shared library

The following item is the procedure for building the shared library of this software.

1) Setting environment variables

Follow Linux BSP Start-Up Guide.

In addition to the guide, add the following path.

```
$export PATH=$WORK/gcc-linaro-arm-linux-gnueabihf-4.7-2013.02-01-20130221_linux/bin:$PATH
$export BUILDDIR=(the directory path storing the include directory of the library headers
                  in your environment)
```

2) Building the shared library

Execute make under mmngr/mmngr-module/files/mmngr/if.

```
$cd mmngr/mmngr-module/files/mmngr/if
$make
```

3) Verifying the shared library

Make sure that the following shared libraries are built under mmngr/mmngr-module/files/mmngr/if.

libmmngr.so.x.x.x ("x.x.x" depends on the version.)

In addition, if you use the dmabuf function of MMNGR, follow 4) and 5).

4) Building the shared library

Execute make under mmngrbuf/mmngrbuf-module/files/mmngrbuf/if.

```
$cd mmngrbuf/mmngrbuf-module/files/mmngrbuf/if
$make
```

5) Verifying the shared library

Make sure that the following shared libraries are built under mmngrbuf/mmngrbuf-module/files/mmngrbuf/if.

libmmngrbuf.so.x.x.x("x.x.x" depends on the version.)

2.2 Building the Kernel and the Kernel Module

The following is the procedure for building the kernel and the kernel module that is not included in this software.

1) Setting environment variables

Follow Linux Interface Specification for R8A77900 Start-Up Guide.

In addition to the guide, set the following value.

```
$export KERNELDIR=(the kernel path in your environment) (for example, $WORK/linux-stable)
$export KERNELSRC=(the directory path storing the include directory in your environment)
                    (for example, $WORK/linux-stable)
$export CP=cp
$export MMNGR_CONFIG=MMNGR_LAGER or MMNGR_KOELSCH (this definition depends on your environment)
$export MMNGR_SSP_CONFIG=MMNGR_SSP_DISABLE
```

2) Building the Kernel

MMNGR uses CMA (Contiguous Memory Allocator) of the kernel.
If you change the size of the space, follow the following steps.
If you don't have to change the size, follow Step.2 only.

Step.1

Modify the config with "make menuconfig".

Set * to "Contiguous Memory Allocator (EXPERIMENTAL)".
Unset * of "CMA debug message DEVELOPMENT) (NEW).
Modify the size in "Size in Mega Byte (NEW)". This size is not only the size used in MMNGR, but the size used in CMA. Therefore if the other function uses CMA, consider not only the size used in MMNGR but also the size used in the other function.

[The target parameters]

```
Device Drivers
  ---> Generic Driver Options
    --->Contiguous Memory Allocator (EXPERIMENTAL)
Device Drivers
  ---> Generic Driver Options
    --->CMA debug message (DEVELOPMENT) (NEW)
Device Drivers
  ---> Generic Driver Options
    ---> Size in Mega Byte (NEW)
```

Step.2

Build the kernel according to 5. Kernel Build in Linux BSP Start-Up Guide.

3) Building the kernel module

About the space out of the kernel management, MMNGR uses MM_OMXBUF_ADDR as the start address, and uses MM_OMXBUF_SIZE as the size.
If you change the start address or the size of the space out of the kernel management, follow the following steps.
If you don't have to change the values, follow Step.2 only.

Step.1

Modify the definition int mmngr_private.h.

[The target definitions: Disable LPAE]

```
MM_OMXBUF_ADDR => the start address of OMXBUF
                  Set the address aligned on 128MB.
                  (Default H2: 0xB0000000, M2: 0x78000000)
MM_OMXBUF_SIZE => the size of OMXBUF.
                  Set the size aligned on 4KB.
                  (Default H2:256MB, M2:128MB ) (Max H2:256MB, M2:128MB)
MM_OMXBUF_MXI_ADDR=> the start address of OMXBUF (32bit address space)
```

[The target definitions: Enable LPAE]

```
MM_OMXBUF_ADDR => the start address of OMXBUF.
                  Set theaddress aligned on 128MB.
                  (Default H2: 0x70000000, M2: 0x78000000)
MM_OMXBUF_SIZE => the size of OMXBUF.
                  Set the size aligned on 4KB.
                  (Default H2:256MB, M2:128MB ) (Max H2:256MB, M2:128MB)
MM_OMXBUF_MXI_ADDR=> the start address of OMXBUF (32bit address space)
```

Step.2
Execute make under mmngr/mmngr-module/files/mmngr/drv.
$cd mmngr/mmngr-module/files/mmngr/drv
$make

In addition, if you want to use the space out of the kernel management for DTV (SSP), follow the followint steps.
About the space out of the kernel management for DTV (SSP),
MMNGR uses MM_SSPBUF_ADDR as the start address, and uses MM_SSPBUF_SIZE as the size.
If you don't have to change the values, follow Step.3 only.

Step.1
Add the definition for DTV (SSP).
$export MMNGR_SSP_CONFIG=MMNGR_SSP_ENABLE

Step.2
Modify the definition int mmngr_private.h.
[The target definitions: Disable LPAE]
MM_SSPBUF_ADDR => the start address of SSPBUF.
Set the address aligned on 4KB.
(Default H2:0xAF000000, M2:0x77000000)
MM_SSPBUF_SIZE => the size of SSPBUF.
Set the size aligned on 4KB.
(Default H2:16MB, M2:16MB)
[The target definitions: Enable LPAE]
MM_SSPBUF_ADDR => the start address of SSPBUF.
Set the address aligned on 4KB.
(Default H2:0x6F000000, M2:0x6F000000)
MM_SSPBUF_SIZE => the size of SSPBUF.
Set the size aligned on 4KB.
(Default H2:16MB, M2:16MB)

In addition, don't include 1GB order from MM_SSPBUF_ADDR to (MM_SSPBUF_ADDR + MM_SSPBUF_SIZE - 1).
Don't overlap the area of OMXBUF, if you use SSPBUF.
Don't overlap the area of MVBUF, if you use SSPBUF and MVBUF is allocated

Step.3
Execute make under mmngr/mmngr-module/files/mmngr/drv.
$cd mmngr/mmngr-module/files/mmngr/drv
$make

In addition, when you set MMNGR_CONFIG to MMNGR_KOELSCH,
MVBUF is allocated in the space out of the kernel management.

About the space out of the kernel management for MV,
MMNGR uses MM_MVBUF_ADDR as the start address, and uses MM_MVBUF_SIZE as the size.
If you have to change the values, follow Step.1 and Step.2.
If you don't have to change the values, follow Step.2 only.

Step.1
Modify the definition in mmngr_private.h.
MM_MVBUF_ADDR (Default H2, M2: 0x7500 0000)
Restrictions of MM_MVBUF_ADDR are as follows.
(a) Set MM_MVBUF_ADDR to the address in the linear area set by BSP.
(b) Set MM_MVBUF_ADDR to the address aligned on 4KB.

MM_MVBUF_SIZE (Default H2, M2: 40 * 1024 * 1024)
Restrictions of MM_MVBUF_SIZE are as follows.
(a) Set MM_MVBUF_SIZE the size aligned on 4KB.
MM_MVBUF_SIZE depends on the number of the simultaneous playback of video decode
and video decoder specification. The maximum number of the simultaneous playback is 4.
The maximum size per video decoder is 10MB. Therefore the default size is maximum size.
If you want to decrease the size, decrease the number of the simultaneous playback,
or decrease the size in based on decoder specification.
About relationship between decoder specification and size,
refer to Memory Requirement of each decode part user's manual.
Don't overlap the area of OMXBUF
Don't overlap the area of SSPBUF, if you use SSPBUF.

Step.2
Execute make under mmngr/mmngr-module/files/mmngr/drv.
$cd mmngr/mmngr-module/files/mmngr/drv

$make

Set OMXBUF, SSPBUF and MVBUF you use within CMA reservation area.
Set the same size as CMA reservation size in setup-rcar-gen2.c to MM_KERNEL_RESERVE_SIZE
Set the total size of OMXBUF, SSPBUF or MVBUF within MM_KERNEL_RESERVE_SIZE.

4) Verifying the kernel and the kernel module

Make sure that the following kernel in the space pointed by Linux BSP Start-Up Guide.

uImage

Make sure that the following kernel module is built under mmngr/mmngr-module/files/mmngr/drv.

mmngr.ko

In addition, if you use the dmabuf function of MMNGR, follow 5) and 6).

5) Building the kernel module

Execute make under mmngrbuf/mmngrbuf-module/files/mmngrbuf/drv

$cd mmngrbuf/mmngrbuf-module/files/mmngrbuf/drv

$make

6) Verifying the kernel module

Make fure that the following kernel module is built under mmngrbuf/mmngrbuf-module/files/mmngrbuf/drv.

mmngrbuf.ko

2.3 Storing the sharead library, the Kernel module and uImage.

The following procedure shows storing the shared library, the kernel module and uImage that are built at 2.1. and 2.2..

1) Storing the shared library
   Create /local/lib under /usr in the rootfs of the target board.
      $mkdir /usr/local
      $mkdir /usr/local/lib
   Store libmmngr.so.x.x.x under /usr/local/lib in the rootfs of the target board.
      $cp libmmngr.so.x.x.x /usr/local/lib

2) Storing the kernel module
   Store the kernel module under the directory in the rootfs of the target board.
      $mkdir /tmp (/tmp is a example. Therefore change the directory accoding to your environment.)
      $cp mmngr.ko /tmp

3) Storing uImage
   Store uImage in the target board according to the start-up guide.

4) Setting environment variable
   Set /usr/local/lib to LD_LIBRARY_PATH at the target board.
      $export LD_LIBRARY_PATH=/usr/local/lib

5) Copying the link to the target rootfs
   Copy the links to the target rootfs of the target board at PC.
      $cp -d libmmngr.so.x (the rootfs of the target board)/usr/local/lib/
      $cp -d libmmngr.so (the rootfs of the target board)/usr/local/lib

6) Install the kernel module at the target board.
   Execute insmod under the directory storing the kernel module at the target board.
      $cd /tmp
      $insmod mmngr.ko

In addition, if you use the dmabuf function of MMNGR, follow from 7) to 10)..

7) Storing the shared library.
   Store libmmngrbuf.so.x.x.x under /usr/local/lib in the rootfs of the target board.
      $cp libmmngrbuf.so.x.x.x /usr/local/lib

8) Storing the kernel module.
   Store the kernel module under the directory in the rootfs of the target board.
      $cp mmngr.ko /tmp (/tmp is a example. Therefore change the directory accoding to your environment.)

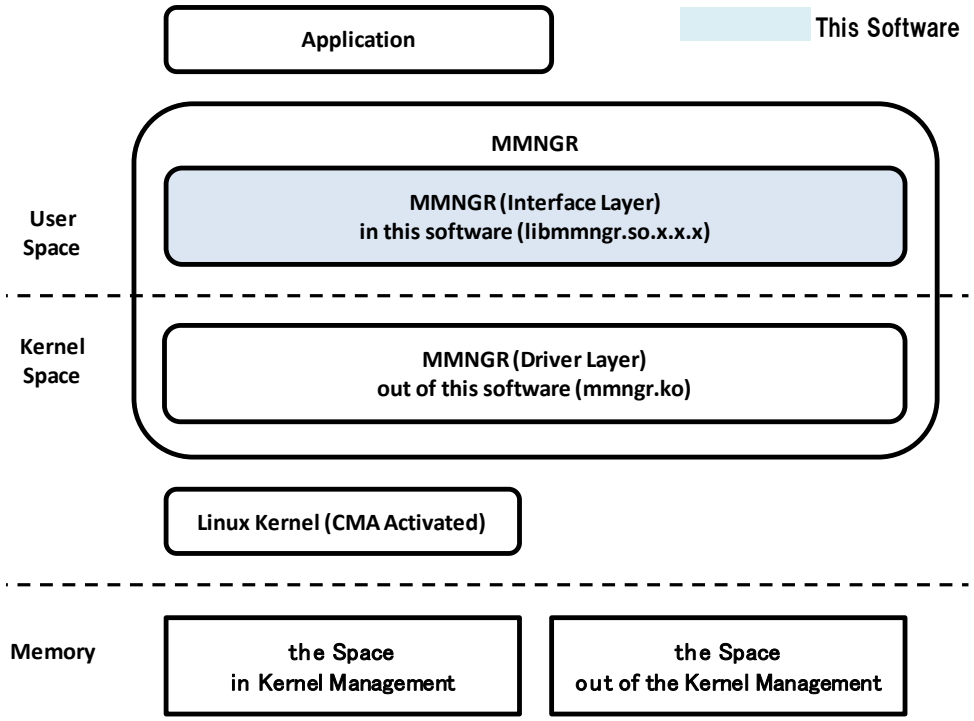
9) Copying the link to the target rootfs.
   Copy the links to the target rootfs of the target board at PC.
      $cp –d libmmngrbuf.so.x (the rootfs of the target board)/usr/local/lib/
      $cp –d libmmngrbuf.so (the rootfs of the target board)/usr/local/lib/

10) Install the kernel module at the target board.
   Execute insmod under the directory storing the kernel module at the target board.

```
$cd /tmp
$insmod mmngrbuf.ko
```

3. Module Configuration

The module configuration of MMNGR is as follows.



・MMNGR consists of the interface layer and the driver layer.

Interface Layer (in this software)

This layer is in the user space and provides API.

Driver Layer (out of this software)

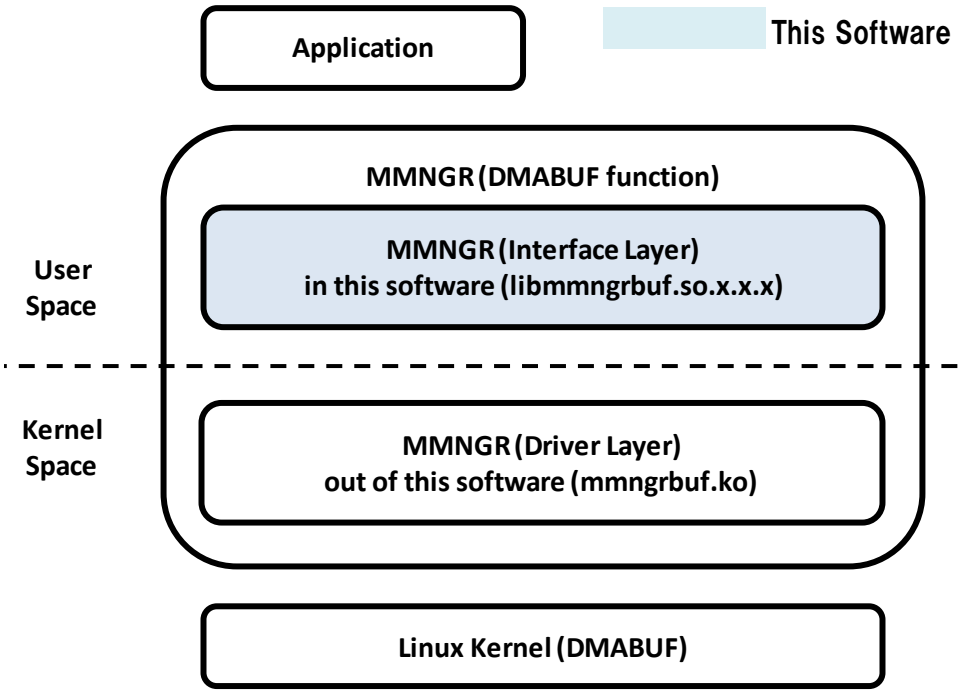
This layer is in the kernel space and calls the kernel API.

・The execution context of MMNGR is the process or the thread calling MMNGR.

That is, MMNGR does not have the process and the thread.

The module configuration of the dmabuf function of MMNGR is as follows.

The modulesof the dmabuf function are idependent from the above modules.



・MMNGR consists of the interface layer and the driver layer.

Interface Layer (in this software)

This layer is in the user space and provides API.

Driver Layer (out of this software)

This layer is in the kernel space and calls the kernel API.

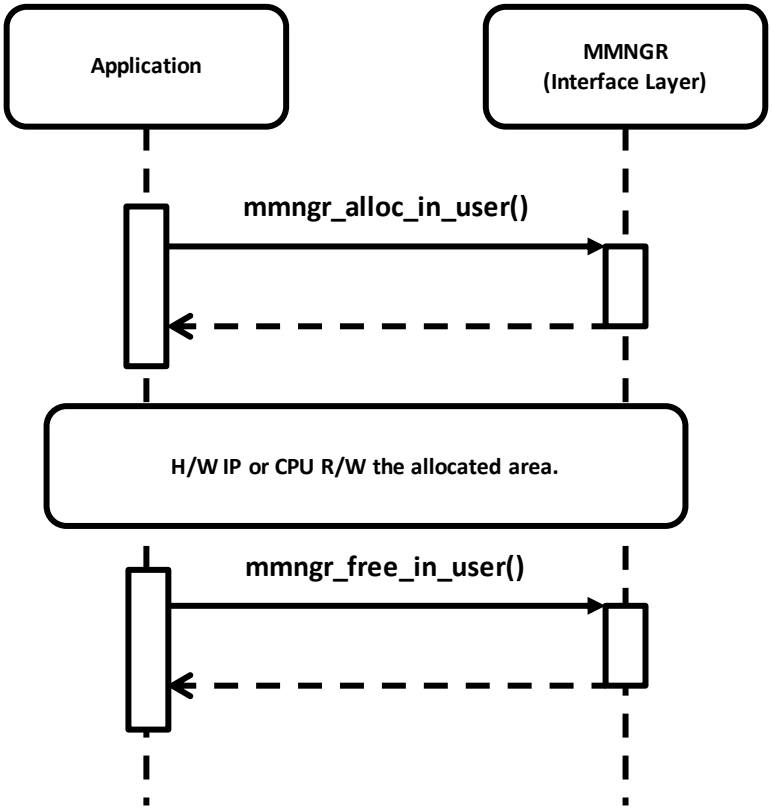
・The execution context of MMNGR is the process or the thread calling MMNGR.

That is, MMNGR does not have the process and the thread.

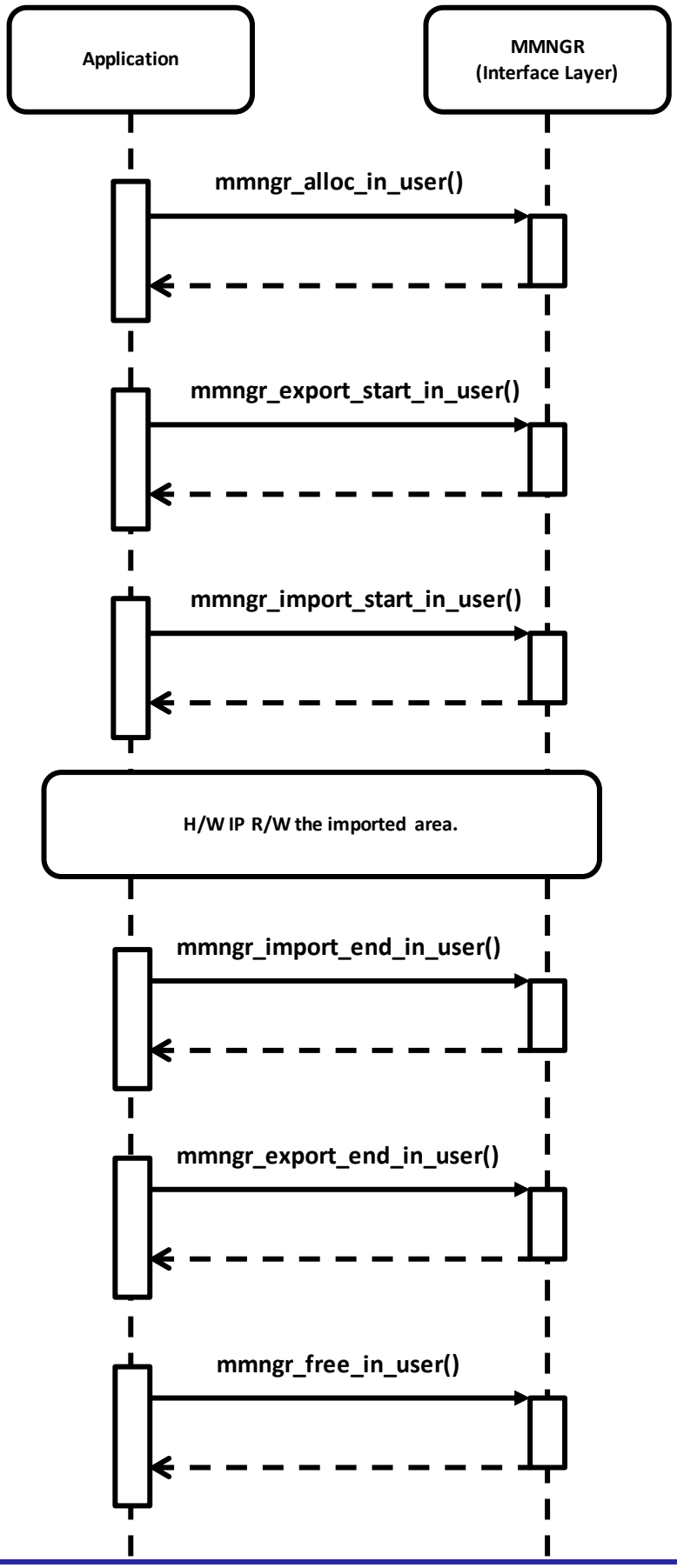
4. List of API

No	Name	Function
1	mmngr_alloc_in_user()	Allocate the memory space
2	mmngr_free_in_user()	Free the memory space
3	mmngr_debug_map_va()	Map the memory space for H/W IP to the user space. (Debug API for the space out of the kernel management)
4	mmngr_debug_unmap_va()	Unmap the memory space mapped to the user space (Debug API for the space out of the kernel management)
5	mmngr_export_start_in_user()	Start the export of dmabuf fd.
6	mmngr_export_end_in_user()	End the export of dmabuf fd.
7	mmngr_import_start_in_user()	Start the import of dmabuf fd.
8	mmngr_import_end_in_user()	End the import of dmabuf fd.

5. Sample Sequence



If you use the dmabuf function of MMNGR, sample sequence is as follows.



6. API Specification

6.1 Allocate Memory Space

Name

mmngr_alloc_in_user

Synopsis

```
int mmngr_alloc_in_user(
    MMNGR_ID *pid,                  (output)
    unsigned long size,             (input)
    unsigned long *pphy_addr,       (output)
    unsigned long *phard_addr,      (output)
    unsigned long *puser_virt_addr, (output)
    unsigned long flag              (input)
)
```

Arguments

MMNGR_ID *pid	The address storing the id of the allocated memory
unsigned long size	The size of the allocation [Byte]
unsigned long *pphy_addr	The address storing the physical address that the lower 12bit address is shifted to right.
unsigned long *phard_addr	The address storing the address for H/W IP of the allocated memory
unsigned long *puser_virt_addr	The address storing the address for CPU of the allocated memory
unsigned long flag	The flag to select the target of the allocation. MMNGR_VA_SUPPORT The space in the kernel management MMNGR_PA_SUPPORT The space out of the kernel management MMNGR_PA_SUPPORT_SSP The space out of the kernel Management for DTV (SSP) (This definition is valid, only when MMNGR_SSP_ENABLE is defined.)

	MMNGR_PA_SUPPORT_MV The space out of the kernel Management for MV

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_PARE	Parameter Error (Condition) (1) When an application set NULL to *pid, pphy_addr, phard_addr* or *puser_virt_addr*. (2) When an application set 0 to *size*. (3) When an application set the undefined value at Parameter (2) to *flag*.
R_MM_NOMEM	Lack of Memory (Condition) When MMNGR can not allocate the space according to *size*.

Description

(1) This function allocates the memory space according to *size*.
(2) The allocated size is rounded up to 4KB alignment.
(3) The alignment of the address pointed by *pphy_addr, phard_addr* and *puser_virt_addr*is 4KB.
(4) This function allocates the memory space with CMA.
According to the specification of CMA,
the allocation time may be long, and the size of the total allocation may be smaller than the size set with "make menuconfig".
(5) MMNGR does not have the defragmentation.
(6) Only when MMNGR_VA_SUPPORT is set to *flag*,
the address is set to *puser_virt_addr*.
(7) The space pointed by *puser_virt_addr* is the user space and the non-cache space.

Notes

(1) Allocate the space set to *pid, pphy_addr, phard_addr and puser_virt_addr* in the application.

The type of the space set to *pid* is MMNGR_ID.

Example

```
MMNGR_ID id;
unsigned long phy_addr;
unsigned long hard_addr;
unsigned long user_virt_addr;
mmngr_alloc_in_user(&id, 64 * 1024, &phy_addr,& hard_addr,
                                    &user_virt_addr);
```

(2) Align the address pointed by *pphy_addr*, and set the aligned address to S3CTL.

(3) Align the address pointed by *phard_addr*, and set the aligned address to H/W IP.

(4) Set the address *puser_virt_addr* to CPU.

(5) Use the address pointed by *puser_virt_addr* in the same process calling this API.

(6) This function wastes one file descriptor per one allocation. Therefore decide the number of the allocation within the maximum of the file descriptor per one process in the application.

(7) Use the address allocated with MMNGR_PA_SUPPORT to set to S3CTL only. That is, use the address allocated with MMNGR_PA_SUPPORT for the frame buffer of the video codec IP only.

(8) Don't refer to the address pointed by *pphy_addr*, when the space is allocated with MMNGR_VA_SUPPORT.

(9) Don't refer to the address pointed by *puser_virt_addr*, when the space is allocated with MMNGR_PA_SUPPORT, MMNGR_PA_SUPPORT_SSP and MMNGR_PA_SUPPORT_MV.

(10) Map the address to the user space with mmngr_debug_map_va() and unmap the mapped address with mmngr_debug_unmap_va(), when you want to refer to the space allocated with MMNGR_PA_SUPPORT, MMNGR_PA_SUPPORT_SSP or MMNGR_PA_SUPPORT_MV in the debug.

(11) Don't call this API at the context of a signal handler.

(12) Use the address allocated with MMNGR_PA_SUPPORT_SSP to set to DTV (SSP) only. That is, use the address allocated with MMNGR_PA_SUPPORT_SSP for the buffer of DTV DEMUX IP only.

(13) Use the address allocated with MMNGR_PA_SUPPORT_MV to set to MV only. That is, use the address allocated with MMNGR_PA_SUPPORT_MV for the buffer of MV area only.

See Also

-

6.2 Free Memory Space

Name

mmngr_free_in_user

Synopsis

```
int mmngr_free_in_user(
    MMNGR_ID id                    (input)
)
```

Arguments

MMNGR_ID *id*	ID of the space freed.

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_SEQE	Sequence Error (Condition) When the space of ID has already been freed.

Description

(1) This function frees the space of ID.

Notes

(1) Set ID, which is allocated by mmngr_alloc_in_user() and is not freed, to *id*.

(2) Don't call this API at the context of a signal handler.

See Also

-

6.3 Map the Address for H/W IP to the User Space (Debug API)

Name

mmngr_debug_map_va

Synopsis

```
int mmngr_debug_map_va(
    MMNGR_ID *pid,                  (output)
    unsigned long size,             (input)
    unsigned long hard_addr,        (input)
    unsigned long *puser_virt_addr  (output)
)
```

Arguments

MMNGR_ID **pid*	The address storing the id of the mapping space
unsigned long *size*	The size of the mapping space [Byte]
unsigned long *hard_addr*	The address for H/W IP to map to the user space.
unsigned long **puser_virt_addr*	The address storing the address for CPU

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_PARE	Parameter Error (Condition) (1) When the application set NULL to *pid* or *puser_virt_addr*. (2) When the application set 0 to *size*.

Description

(1) This function map the address for H/W IP to the user space according to *size* and *hadr_addr*.

(2) The size of the mapping is rounded up to 4KB.

(3) The alignment of the address pointed by *puser_virt_addr* is 4KB.

(4) The space pointed by *puser_virt_addr* is the user space and the non-cache space.

Notes

(1) Don't use this function except for the debug.

(2) Allocate the space set to pid and *puser_virt_addr* in the application.
The type of the space set to pid is MMNGR_ID.

(3) Set size and phard addr of the space allocated by mmngr_alloc_in_user()
with MMNGR_PA_SUPPORT, MMNGR_PA_SUPPORT_SSP

or MMNGR_PA_SUPPORT_MV to *size* and *hard_addr*.

(4) Use the address pointed by *puser_virt_addr* in the same process calling this API.

(5) This function wastes one file descriptor per one allocation. Therefore decide the number of the allocation within the maximum of the file descriptor per one process in the application.

(6) Don't call this API at the context of a signal handler.

See Also

-

6.4 Unmap the Mapping Space (Debug API)

Name

mmngr_debug_unmap_va

Synopsis

int mmngr_debug_unmap_va(

MMNGR_ID *id* (input)

)

Arguments

MMNGR_ID *id*	ID of the unmapping space

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_SEQE	Sequence Error (Condition) When the space of ID has already been unmapped.

Description

(1) This function unmaps the space of ID.

Notes

(1) Don't use this function except for the debug.

(2) Set ID, which is mapped by mmngr_debug_unmap_va() and is not unmapped, to *id*.

(3) Don't call this API at the context of a signal handler.

See Also

-

6.5 Start the export of dmabuf fd

Name

mmngr_export_start_in_user

Synopsis

```
int mmngr_export_start_in_user(
    int *pid,                   (output)
    unsigned long size,         (input)
    unsigned long hard_addr,    (input)
    int *pbuf                   (output)
)
```

Arguments

int **pid*	The address storing the id of the export.
unsigned long *size*	The size to connect to dmabuf fd
unsigned long *hard_addr*	The address for H/W IP to connect to dmabuf fd
int **pbuf*	The address storing dmabuf fd

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_PARE	Parameter Error (Condition) (1) When an application set NULL to *pid* or *pbuf*. (2) When an application set 0 to *size*.

Description

(1) This function allocates dmabuf fd from the address and the address of the memory space.

Notes

(1) Allocate the space set to *pid* and *pbuf* in the application.

(2) Set the size of the space, which is allocated by mmngr_alloc_in_user and is not freed, to *size*.

(3) Set the address of the space, which is allocated by mmngr_alloc_in_user and is not freed, to *hard_addr*.

(4) Set the size aligned by 4KB to *size*

(5) Set the address aligned by 4KB to *hard_addr*.

(6) This function wastes two file descriptores per one export.
One is used for ID. The other is used for dmabuf fd.

Therefore decide the number of the allocation within the maximum of the file descriptor per one process in the application.

(7) Use dmabuf fd pointed by *pbuf* in the same process calling this API.

(8) Don't access the space pointed by *hard_addr* with CPU

(9) Don't call this API at the context of a signal handler.

(10) Don't free the space to set to *size* and *hard_addr* until mmngr_export_end_in_user finishes.

(11) Don't execute mmap() for dmabuf fd pointed by *pbuf*.

See Also

-

6.6 End the export of dmabuf fd

Name

mmngr_export_end_in_user

Synopsis

```
int mmngr_export_end_in_user(
    int id                              (input)
)
```

Arguments

int *id*	ID of export ended

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error

Description

(1) This function frees dmabuf fd for id.

Notes

(1) Set ID, which is started by mmngr_export_start_in_user and is not ended, to *id*.

(2) Don't set ID for the dmabuf fd, while dmabuf fd is imported.

(3) Don't call this API at the context of a signal handler.

See Also

-

6.7 Start the import of dmabuf fd

Name

mmngr_import_start_in_user

Synopsis

```
int mmngr_import_start_in_user(
    int *pid,                   (output)
    unsigned long *psize,       (output)
    unsigned long *phard_addr,  (output)
    int buf                     (input)
)
```

Arguments

int **pid*	The address storing the id of the import.
unsigned long **psize*	The address storing the size connected to dmabuf fd
unsigned long **phard_addr*	The address storing the address for H/W IP connected to dmabuf fd
int *buf*	dmabuf fd

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error
R_MM_PARE	Parameter Error (Condition) (1) When an application set NULL to *pid*, *psize* or *phard_addr*.

Description

(1) This function gets the address and the size of the memory space from dmabuf fd.

Notes

(1) Set the dmabuf fd, which is started by mmngr_import_start_in_user and is not ended, to *buf*.

(2) Set the dmabuf fd, which is exported by the other software and is not closed, to *buf*.

(3) Set the dmabuf fd connected to one physical contiguous space only.

(4) Don't call this API at the context of a signal handler.

See Also

-

6.8 End the import of dmabuf fd

Name

mmngr_import_end_in_user

Synopsis

```
int mmngr_import_end_in_user(
    int id                          (input)
)
```

Arguments

int *id*	ID of import ended

Struct

-

Return Value

R_MM_OK	Normal End
R_MM_FATAL	Fatal Error

Description

(1) This function ends using the address and the size got from dmabuf fd for id.

Notes

(1) Set ID, which is started by mmngr_import_start_in_user and is not ended, to id.

(2) Don't set ID for the dmabuf fd, while the space to connected to dmabuf fd is used.

(3) Don't call this API at the context of a signal handler.

See Also

-

7. Header Files

Include mmngr_user_public.h in the application, when the application calls the APIs of MMNGR.
Include mmngr_buf_user_public.h in the application, when the application calls the APIs of DMABUF function of MMNGR.

8. Definition

8.1 Return Value Definition

Definition	Value	Content
R_MM_OK	0	Normal End
R_MM_FATAL	-1	Fatal Error
R_MM_SEQE	-2	Sequence Error
R_MM_PARE	-3	Parameter Error
R_MM_NOMEM	-4	Lack of Memory

8.2 Parameter Definition

Definition	Value	Content
MMNGR_VA_SUPPORT	0	The space in the kernel management
MMNGR_PA_SUPPORT	1	The space out of the kernel management
MMNGR_PA_SUPPORT_SSP	3	The space out of the kernel management for DTV (SSP)
MMNGR_PA_SUPPORT_MV	4	The space out of the kernel management for MV

Revision History	Memory Manager for Linux User's Manual: Software

Rev.	Date	Description	
		Page	Summary
1.00	May.2014	—	First edition issued
1.10	Jun.2014	1, 5, 6	Add the explanation of the space out of the kernel management at LPAE.
1.21	Jul.2014	6, 13, 14, 18, 21	Add the explanation of the space out of the kernel management for MV.
1.30	Aug.2014	2, 4, 7, 11,12,23, 24, 25, 26, 28	Add the explanation of the dmabuf function.
1.40	Jan.2015	6, 7	Add the explanation of the space out of the kernel management.

Memory Manager for Linux User's Manual: Software

Publication Date: Rev.1.40 Jan. 2015

Published by: Renesas Electronics Corporation

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

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.3

Memory Manager for Linux
User's Manual: Software

Renesas Electronics Corporation