# Overview

## Overview of this Software

・This software can manage the memory’s data.

・Feature:

* + - 1. Acquire to physical memory address.
      2. Memory allocation

## Configuration of this Software

This software consists of the following resources.

・Source code

・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

Figure 1-1 shows the lists of these source files.

cmem

├── GPL-COPYING

├── MIT-COPYING

├── Makefile

├── cmemdrv.h

├── cmemdrv.c

**Figure 1‑1 File structure**

## Development Environments

This section describes the development environments for this software.

### Software Development Environment

Table 1‑1 Software specification

|  |  |  |
| --- | --- | --- |
| **Software Name** | **Version / Revision** | **Remarks** |
| R-Car H3/M3/M3N/E3/D3/V3U/V3H/V3M Linux BSP | - | - |

# Module Configuration

The module configuration of CMEM is as follows.

The space reserved as

CMA area

Linux Kernel

Application

CMEM Driver

Memory

Kernel space

User space

**Figure 2‑1 CMEM module configuration**

# List of API

In CMEM driver, all callback functions are registered to file\_operations function, so the user need to use file operations functions to call those callback function, the table bellow is shown the information of callback function

Table 3‑1 List of callback functions

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **File operations**  **functions** | **CMEM callback**  **functions** | **Explain** |
| 1 | read() | dev\_read () | read data from memory |
| 2 | write() | dev\_write () | write data to memory |
| 3 | ioctl() | dev\_ioctl () | io control |
| 5 | mmap () | dev\_mmap () | remap address |
| 6 | open () | dev\_open () | open device |
| 7 | close () | dev\_rls () | release device |

# API Specification

## Read data

|  |  |  |
| --- | --- | --- |
| ***Name*** |  |  |
|  | read() |  |
| ***Synopsis*** |  |  |
|  | ssize\_t read(  int *fd*,  void \**buf*,  size\_t *count*,  ) | (input)  (output)  (output) |
| ***Arguments*** |  |  |
|  | int *fd*, | File descriptor |
|  | void \**buf*, | The buffer where data is written. |
|  | size\_t *count*, | the length of data |
| ***Struct*** |  |  |
|  | - |  |
| ***Return Value*** |  |  |
|  | On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number.  On error, -1 is returned, and errno is set appropriately. | |
| ***Description*** |  | |
|  | read() attempts to read up to count bytes from memory into the buffer starting at buf. | |

## Write data

|  |  |  |
| --- | --- | --- |
| ***Name*** |  |  |
|  | write |  |
| ***Synopsis*** |  |  |
|  | ssize\_t write (  int *fd*,  void \**buf*,  size\_t *count*,  ) | (input)  (input)  (output) |
| ***Arguments*** |  |  |
|  | int *fd*,  void \**buf*,  size\_t *count*, | File descriptor  The buffer which store the written data.  the length of data |
| ***Struct*** |  |  |
|  | - |  |
| ***Return Value*** | On success, the number of bytes written is returned.  On error, -1 is returned, and errno is set to indicate the cause of the error. | |
| ***Description*** |  | |
|  | write() writes up to count bytes from the buffer starting at buf to the memory | |

## IO control function

|  |  |  |
| --- | --- | --- |
| ***Name*** |  |  |
|  | ioctl |  |
| ***Synopsis*** |  |  |
|  | int ioctl (  int *fd*,  unsigned long *request*,  void\* *arg*  ) | (input)  (input)  (input/output) |
| ***Arguments*** |  |  |
|  | int *\*filep*,  unsigned long *request*  void\* *arg* | File descriptor  The command which is used to control CMEM driver  The following command which is shown below can be used in those functions:  PARAM\_SET, M\_LOCK, M\_UNLOCK, GET\_PHYS\_ADDR,  M\_ALLOCATE, M\_UNALLOCATE, TRY\_CONV.  For more information about those command, please refer the description part.  Pointer to mem\_setpara, mem\_mlock, mem\_info structure  Depend on the type of command, the argument pointer will specify to the following structure |
| ***Struct*** | PARAM\_SET : using mem\_setpara struct to set cmem’s parameters  M\_LOCK: mem\_mlock struct is used.  M\_UNLOCK: mem\_mlock struct is used  GET\_PHYS\_ADDR: get physical address of memory using mem\_info struct  TRY\_CONV : convert virtual address to physical address, user I/F is not updated in this time  Please refer chapter 6.1, for more information about those structure | |
|  |
| ***Return Value*** | Zero is returned on success and -1 is return on error with errno is set appropriately | |
| ***Description*** |  | |
|  | The ioctl() system call io control function of CMEM driver  PARAM\_SET : setting memory’s parameters (width, height, offset, stride and tl)  M\_LOCK : cache data flush and delete cache data  M\_UNLOCK : delete cache data  GET\_PHYS\_ADDR: get physical address of memory  TRY\_CONV : convert virtual address to physical address | |

## Map the Address for H/W IP to the User Space

|  |  |  |
| --- | --- | --- |
| ***Name*** |  |  |
|  | mmap |  |
| ***Synopsis*** |  |  |
|  | void\* mmap (  void \**addr*,  size\_t *length*,  int *prot*,  int *flags*,  int *fd*,  off\_t *offset*  ) | (input)  (input)  (input)  (input)  (input)  (input) |
| ***Arguments*** |  |  |
|  | void \**addr*, | the starting address space of mapping |
|  | size\_t *length*, | the length of memory space memory |
|  | int *prot*,  int *flags*,  int *fd*,  off\_t *offset* | protect  flags, but this flags variable is not used in CMEM driver, this flags must be set to 0.  file descriptor  offset |
| ***Struct*** |  |  |
|  | - |  |
| ***Return Value*** |  |  |
|  | On success, mmap() returns a pointer to the mapped area.  On error, the value MAP\_FAILED (that is, (void \*) -1) is returned, and errno is set to indicate the cause of the error. | |
|  |
| ***Description*** |  | |
|  | mmap() creates a new mapping in the virtual address space of the calling process. | |
| **Note** | the offset value can be set from 0. | |

## Open device

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name*** |  |  | |
|  | open |  | |
| ***Synopsis*** |  |  | |
|  | int open(  const char \**pathname*,  int *flags*  ) | (input)  (input) | |
| ***Arguments*** |  |  | |
|  | const char \**pathname*,  int *flags* | node ID  file operation struct | |
| ***Struct*** |  |  | |
|  | - |  | |
| ***Return Value*** |  |  | |
|  |  | | |
|  |
|  | | |
| ***Description*** |  | | |
|  | The open() system call opens cmem driver function | | |

## Close

|  |  |  |
| --- | --- | --- |
| ***Name*** |  |  |
|  | close |  |
| ***Synopsis*** |  |  |
|  | int close (  int *fd*  ) | (input) |
| ***Arguments*** |  |  |
|  | int *fd* | file descriptor |
| ***Struct*** |  |  |
|  | - |  |
| ***Return Value*** |  |  |
|  | Return 0 on success and -1 on error with errno is set appropriately | |
| ***Description*** |  | |
|  | Close device | |

# Definition

## Structure

### mem\_setpara

struct mem\_setpara {

int offset;

int width;

int height;

int stride;

int tl;

};

Table 5‑1 Members of mem\_setpara structure

|  |  |  |
| --- | --- | --- |
| **Member** | **Direction** | **Contents** |
| int offset | Input | Setting the offset of memory |
| int width | Input | Setting the width of memory |
| int height | Input | Setting the height of memory |
| int stride | Input | Setting the stride of memory |
| int tl | Input | Translation lookaside |

### mem\_mlock

struct mem\_mlock {

size\_t offset;

size\_t size;

size\_t dir;

};

Table 5‑2 Members of mem\_mlock structure

|  |  |  |
| --- | --- | --- |
| **Member** | **Direction** | **Contents** |
| size\_t offset | Input | Setting the offset of memory |
| size\_t size | Input | Setting the size of memory |
| size\_t dir | Input | Setting the direction. (IOCTL\_FROM\_DEV\_TO\_CPU and IOCTL\_FROM\_CPU\_TO\_DEV are used when setting this variable) |

### mem\_info

struct mem\_info {

size\_t phys\_addr;

};

Table 5‑3 Members of mem\_info structure

|  |  |  |
| --- | --- | --- |
| **Member** | **Direction** | **Contents** |
| size\_t phys\_addr | Input | Getting the physical memory address |

## Macro

### Parameter Definition

Table 6‑4 List of Parameter Definition

|  |  |  |
| --- | --- | --- |
| **Definition** | **Value** | **Content** |
| PARAM\_SET | 1 | Setting cmem’s parameters |
| M\_LOCK | 3 | Cache data flush and delete cache data |
| M\_UNLOCK | 4 | Delete cache data |
| GET\_PHYS\_ADDR | 5 | Get physical address |
| M\_ALLOCATE | 6 | No support |
| M\_UNALLOCATE | 7 | No support |
| TRY\_CONV | 8 | Convert virtual address to physical address |

# Option Setting

## Module parametters

When load CMEM driver as loadable module, the following parameters can be used to change the CMEM setting

### Setting the size of memory

The memory size can be specify by the following command

insmod cmemdrv.ko bsize= *<value>*

when: <value> is the size of memory that you want to create

If this option is not specified, the default value (16\*1024\*1024) will be used

Example: if you want to create memory with the size = 0x7000000

# insmod cmemdrv.ko bsize=0x7000000

### Enable/Disable cache

The cache can be enabled or disable by the following command.

insmod cmemdrv.ko cached=*<value>*

When <value> is 0 or 1. 0 mean disable, and 1 mean enable.

The cache is enabled by default when this option is not specified.

Example: if you want to disable cache

# insmod cmemdrv.ko cached=0

### Change the cmem’s device file major number

The device file major number can be changed by the following command

insmod cmemdrv.ko cmem\_major=*<value>*

When *<value>* is the number which you want to set.

If this option is not specified, 88 is set as default number for cmem device file.

Example: if you want to set cmem device file major number to 70

# insmod cmemdrv.ko cmem\_major=70