1. Overview

1.1 Overview of this Software

・This software can manage the memory’s data.

・Feature:

1. Acquire to physical memory address.
2. Memory allocation

1.2 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

1.3 Development Environments

This section describes the development environments for this software.

1.3.1 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	-	-

2. Module Configuration

The module configuration of CMEM is as follows.

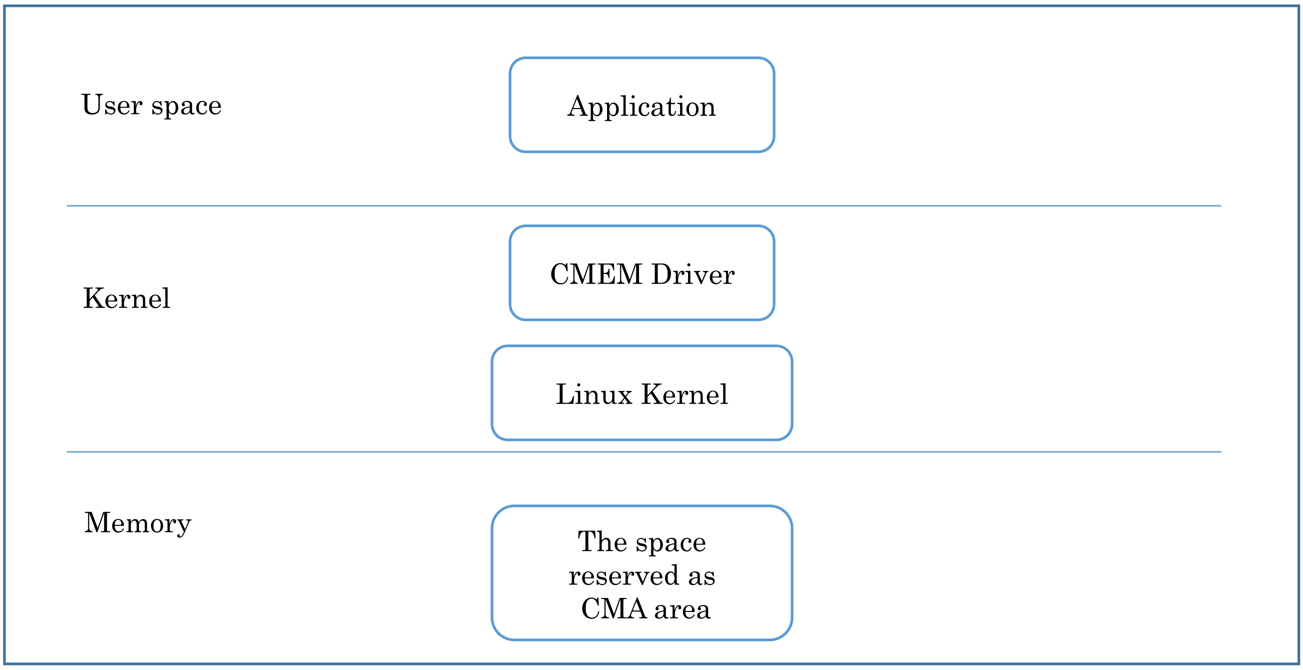

Figure 2-1 CMEM module configuration

3.

~~3.~~4. 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

~~4.~~5. API Specification

~~4.1~~5.1 Read data

Name

read()

Synopsis

ssize_t read(

int *fd*,	(input)
void ***buf*,	(output)
size_t *count*,	(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.

~~4.2~~5.2 Write data

Name

write

Synopsis

ssize_t write (

int *fd*,	(input)
void **buf*,	(input)
size_t *count*,	(output)

)

Arguments

int *fd*,	File descriptor
void **buf*,	The buffer which store the written data.
size_t *count*,	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

~~4.3~~5.3 IO control function

Name

ioctl

Synopsis

int ioctl (

int *fd*, (input)

unsigned long *request*, (input)

void* *arg* (input/output)

)

Arguments

int **filep*, File descriptor

unsigned long *request* 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.

void* *arg* 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~~5.1~~, 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

~~4.4~~5.4 Map the Address for H/W IP to the User Space

Name

mmap

Synopsis

```
void* mmap (
       void *addr,                 (input)
       size_t length,              (input)
       int prot,                   (input)
       int flags,                  (input)
       int fd,                     (input)
       off_t offset                (input)
)
```

Arguments

void **addr*,	the starting address space of mapping
size_t *length*,	the length of memory space memory
int *prot*,	protect
int *flags*,	flags, but this flags variable is not used in CMEM driver, this flags must be set to 0.
int *fd*,	file descriptor
off_t *offset*	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.

~~4.5~~5.5 Open device

Name

open

Synopsis

```
int open(
          const char *pathname,          (input)
          int flags                      (input)
)
```

Arguments

const char **pathname*,	node ID
int *flags*	file operation struct

Struct

-

Return Value

Description

The open() system call opens cmem driver function

~~4.6~~5.6 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

~~5.~~6. Definition

~~5.1~~6.1 Structure

~~5.1.1~~6.1.1 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

~~5.1.2~~6.1.2 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)

~~5.1.3~~6.1.3 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

~~5.2~~6.2 Macro

~~5.2.1~~6.2.1 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

~~6.~~7. Option Setting

~~6.1~~7.1 Module parametters

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

~~6.1.1~~7.1.1 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
```

~~6.1.2~~7.1.2 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
```

~~6.1.3~~7.1.3 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
```