

|||||

IPMUX1IPMUX2IPMUX3

ipmux_interrupt.c	
ipmux_interrupt.h	
opl_cpuif.c	
opl_cpuif.h	
opl_cpuif_test.c	
opl_hw_ops.h	
opl_minte.c	
opl_minte.h	
opl_minte_test.c	
opl_reg_mmap.c	
opl_reg_mmap.h	
opl_regmmap_test.c	
opl_typedef.h	

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

ipmux_interrupt.c

```
#include <linux/wait.h> #include <asm/bitops.h>
#include <linux/sched.h>
#include <linux/kernel.h>
#include <ipmux_interrupt.h>
#include <opl_hw_ops.h>
```

local variable declaration and definition.

void	turn_on_level2_intr (u32 regID, char *bitmap) <i>description:</i> To enable the level2 interrupt.
void	turn_off_level2_intr (u32 regID) <i>description:</i>
u32	get_mask (char *bitmap)
char	intr_status [8][32]

global variable imported

u32	g_opl_chip_irq_event
wait_queue_head_t	g_opl_chip_waitq

void	ipmux_interrupt_handler (int irq, void *dev_id, struct pt_regs *regs)
void	ipmux_interrupt_handler (int irq, void *dev_id, struct pt_regs *regs) <i>description: the ipmux interrupt handler,only wake up the process,which to read the irqpending.</i>
int	ipmux_hw0_irqinit (void) <i>description: request the ipmux irq and enable the its intr.</i>
void	ipmux_hw0_irqexit (void) <i>description: free irq,and disable ipmux interrupt.</i>

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

ipmux_interrupt.h

Macro definition

#define	REG_INT_EN_POS_UPT	0x10204
#define	REG_INT_PEN_POS_UPT	0x10200
#define	REG_INT_EN_ATB	0x11404
#define	REG_INT_PEN_ATB	0x11400
#define	REG_INT_EN_IWF	0x13E00
#define	REG_INT_PEN_IWF	0x13E04
#define	REG_INT_EN_BRG	0x21000
#define	REG_INT_PEN_BRG	0x21004
#define	REG_INT_EN_PPE	0x18180
#define	REG_INT_PEN_PPE	0x18184
#define	REG_INT_EN_UPTM	0x14404
#define	REG_INT_PEN_UPTM	0x14400
#define	REG_INT_EN_DNTM	0x14F04
#define	REG_INT_PEN_DNTM	0x14F00
#define	REG_INT_EN_MC	0x21404
#define	REG_INT_PEN_MC	0x21400
#define	REG_INT_EN_DMA0	0x2B58
#define	REG_INT_PEN_DMA0	0x2B54
#define	OPL_UART0_IRQ	3
#define	OPL_UART1_IRQ	4
#define	OPL_HOSTDMA0_IRQ	5
#define	OPL_FEDMA1_IRQ	6
#define	OPL_IPMUX_IRQ	9

|||||

opl_cpuif.c

```
#include <linux/init.h> #include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched.h>
#include <linux/wait.h>
#include <asm/uaccess.h>
#include <asm/io.h>
#include <linux/slab.h>
#include <linux/types.h>
#include <linux/errno.h>
#include <ipmux_interrupt.h>
#include <opl_cpuif.h>
```

Macro definition

```
#define OPL_HOST_MAJOR 130
#define OPL_HOST_NAME "opl_cpuif"
#define IPMUX_DMA_NUMS 4
#define OPL_DMA_VAL_MAX 2048
```

global variable and function imported

```
void ipmux_irq_enable (unsigned int irq)
void ipmux_irq_disable (unsigned int irq)
```

local variable declaration and definition

	int	opl_host_open (struct inode *inode, struct file *filp)
	ssize_t	opl_host_read (struct file *filp, char *buffer, size_t length, loff_t *offset)
	ssize_t	opl_host_write (struct file *filp, const char *buffer, size_t length, loff_t *offset)
	int	opl_host_release (struct inode *inode, struct file *filp)
	int	opl_host_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, u32 arg) <i>description: it is used for api to do something associated with ioctl command.</i>
	void	host_dma0_isr (int irq, void *dev_id, struct pt_regs *regs) <i>description: the host dma0 interrupt handler, when the interrupt generated, it will wake up the process, which is waiting for the host dma0 interrupt.</i>
	u32	tx_phys_addr [IPMUX_DMA_NUMS] = {0}
	u32	rx_phys_addr [IPMUX_DMA_NUMS] = {0}
	char *	dma0_rx_buf = NULL
	char *	dma0_tx_buf = NULL
	u32	order = 0
	u32	opl_host_is_open = 0
	u32	opl_host_open_count = 0
	u32	opl_host_dma0_irq_event = 0
wait_queue_head_t		opl_host_dma0_waitq
file_operations		opl_host_fops

int __init	opl_host_dma_init (void) <i>description: the module's entry.</i>
void __exit	opl_host_dma_exit (void) <i>description:the module's exit.</i>
	module_init (opl_host_dma_init)
	module_exit (opl_host_dma_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("opulan Inc")
	MODULE_DESCRIPTION ("opulan IPMUX-e switch chip driver module")

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_cpuif.h

```
struct dma_request_phys_addr_s  
struct dma_request_data_s
```

Macro definition

#define	GET_IPMUX_DMA_TX_PHYS_ADDR	1000
#define	GET_IPMUX_DMA_RX_PHYS_ADDR	2000
#define	GET_IPMUX_DMA_RX_BUF_DATA	3000
#define	SET_IPMUX_DMA_TX_BUF_DATA	4000
#define	ENABLE_IPMUX_HOST_DMA0_INTERRUPT	5000
#define	DISABLE_IPMUX_HOST_DMA0_INTERRUPT	6000
#define	WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT	7000

type definition

```
typedef dma_request_phys_addr_s dma_request_phys_addr_t  
typedef dma_request_data_s dma_request_data_t
```

```
#define DISABLE_IPMUX_HOST_DMA0_INTERRUPT 6000
```

opl_cpuif.h26

opl_host_ioctl().

```
#define ENABLE_IPMUX_HOST_DMA0_INTERRUPT 5000
```

opl_cpuif.h25

opl_host_ioctl().

```
#define GET_IPMUX_DMA_RX_BUF_DATA 3000
```

opl_cpuif.h22

opl_host_ioctl().

```
#define GET_IPMUX_DMA_RX_PHYS_ADDR 2000
```

opl_cpuif.h20

opl_host_ioctl().

```
#define GET_IPMUX_DMA_TX_PHYS_ADDR 1000
```

opl_cpuif.h19

opl_host_ioctl().

```
#define SET_IPMUX_DMA_TX_BUF_DATA 4000
```

opl_cpuif.h23

opl_host_ioctl().

```
#define WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT 7000
```

opl_cpuif.h27

opl_host_ioctl().

```
typedef struct dma_request_data_s dma_request_data_t
```

[**opl_host_ioctl\(\)**](#).

```
typedef struct dma_request_phys_addr_s dma_request_phys_addr_t
```

[**opl_host_ioctl\(\)**](#).

|||||

opl_cpuif_test.c

```
#include <stdlib.h> #include <unistd.h>
#include <opl_typedef.h>
```

Macro definition

```
#define OPL_CPUIF_DEVICE "/dev/opl_cpuif"
```

global variable and function exported

int	opl_cpuif_test_init (int *initialized)
void	opl_cpuif_test_main (int argc, char **argv)
int	opl_cpuif_test_exit (int *initialized)

```
int main (int argc, char **argv)
```

```
#define OPL_CPUIF_DEVICE "/dev/opl_cpuif"
```

[opl_cpuif_test.c22](#)

```
int main( int      argc,
          char ** argv
        )
```

opl_cpuif_test.c109

opl_cpuif_test_exit()opl_cpuif_test_init()opl_cpuif_test_main().

```
int opl_cpuif_test_exit( int * initialized )
```

opl_cpuif_test.c96

main().

```
int opl_cpuif_test_init( int * initialized )
```

opl_cpuif_test.c89

u32.

main().

```
void opl_cpuif_test_main( int      argc,
                           char ** argv
                         )
```

opl_cpuif_test.c105

main().

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  **doxygen** 1.3.9.1

|||||

opl_hw_ops.h

Macro definition

```
#define IPMUX_REG_BASE 0xBF000000
```

```
#define IPMUX_REG_ID2ADDR(regId) ((u32)IPMUX_REG_BASE + (u32)(regId))
#define IPMUX_REG_ADDR2ID(addr) ((u32)(addr) - (u32)IPMUX_REG_BASE)
```

```
int ipMuxRegRead (u32 regID, volatile u32 *pval)  
int ipMuxRegWrite (u32 regID, u32 val)
```

```
#define IPMUX_REG_ADDR2ID( addr ) ((u32)(addr) - (u32)IPMU
```

opl_hw_ops.h93

```
#define IPMUX_REG_BASE 0xBF000000
```

opl_hw_ops.h21

```
#define IPMUX_REG_ID2ADDR( regId ) ((u32)IPMUX_REG_BAS
```

opl_hw_ops.h92

ipMuxRegRead()ipMuxRegWrite().

```
int ipMuxRegRead( u32          regID,  
                   volatile u32 * pval  
                 ) [inline, static]
```

opl_hw_ops.h95

IPMUX_REG_ID2ADDRu32.

```
int ipMuxRegWrite( u32 regID,  
                    u32 val  
                  ) [inline, static]
```

opl_hw_ops.h105

IPMUX_REG_ID2ADDRu32.

turn_off_level2_intr()turn_on_level2_intr().

|||||

opl_minte.c

```
#include <linux/init.h> #include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched.h>
#include <linux/wait.h>
#include <asm/uaccess.h>
#include <asm/io.h>
#include <linux/slab.h>
#include <linux/types.h>
#include <linux/errno.h>
#include <opl_minte.h>
#include <ipmux_interrupt.h>
```

```
#define OPL_MINTE_MAJOR 120
#define OPL_MINTE_DEVICE "opl_minte"
```

the variable and function imported.

<code>u32</code>	<code>get_irq_pending</code> (void)
<code>void</code>	<code>clear_irq_pending</code> (void)
<code>void</code>	<code>ipmux_irq_enable</code> (unsigned int irq)
<code>void</code>	<code>ipmux_irq_disable</code> (unsigned int irq)

local function declaration

int	opl_minte_open (struct inode *inode, struct file *filp)
ssize_t	opl_minte_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description: copy the opl_chip_irq_pending(indicate which interrupt occurred) to user App, which will handle its interrupt.if the no irq pending occurred,the process will sleep until the isr wake up it.</i>
ssize_t	opl_minte_write (struct file *filp, const char *buffer, size_t length, loff_t *offset) <i>description: because the enable irq or disable irq will be in atomic mode,so it should be operated in kernel mode is correct.</i>
int	opl_minte_release (struct inode *inode, struct file *filp)

the global variable declaration and definition.

u32	g_opl_chip_irq_event = 0
wait_queue_head_t	g_opl_chip_waitq

local variable declaration and definition

	u32 <code>opl_minte_open_count</code> = 0
	u32 <code>opl_minte_is_open</code> = 0
file_operations	<code>opl_minte_fops</code>

int __init	opl_minte_init (void) <i>description: The driver module entry to register the chrdev and request irq.</i>
void __exit	opl_minte_exit (void) <i>description: it is called when the module removed.</i>
	module_init (opl_minte_init)
	module_exit (opl_minte_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("stephanxu <hxu@opulan.com>")
	MODULE_DESCRIPTION ("control interface for opulan ipmux-e switch chip")

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_minte.h

Macro definition

#define	opl_minte_hw0_irqexit	ipmux_hw0_irqexit
#define	opl_minte_hw0_irqinit	ipmux_hw0_irqinit
int	ipmux_hw0_irqinit (void)	<i>description: request the ipmux irq and enable the its intr.</i>
void	ipmux_hw0_irqexit (void)	<i>description: free irq, and disable ipmux interrupt.</i>

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_minte_test.c

```
#include <stdlib.h> #include <fcntl.h>
#include <unistd.h>
#include "../include/opl_typedef.h"
```

Macro definition

```
#define OPL_MINTE_DEVICE "/dev/opl_minite"
```

local variable declaration and definition

int	opl_minte_init (int *initialized) <i>description: it is used to open the /dev/opl_minte device with flag: RDWR SYNC.</i>
void	opl_minte_main (int argc, char **argv) <i>description: it is the example of how to use it to check the value.</i>
int	opl_minte_exit (int *initialized) <i>description: close the opened file "/dev/opl_minte".</i>

local function declaration

int **opl_minte_fd**

the global fd of the "/dev/opl_minte"

```
int main (int argc, char **argv)  
    description: the testing entry for minte unit testing programming.
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_reg_mmap.c

```
#include <linux/init.h> #include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched.h>
#include <linux/wait.h>
#include <asm/uaccess.h>
#include <asm/io.h>
#include <linux/slab.h>
#include <opl_hw_ops.h>
#include <opl_reg_mmap.h>
```

```
#define OPL_REG_MAJOR 110
#define OPL_REG_DEVICE "opl_reg"
```

local function declaration

int	opl_reg_open (struct inode *inode, struct file *filp)
ssize_t	opl_reg_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description:</i>
ssize_t	opl_reg_write (struct file *filp, const char *buffer, size_t length, loff_t *offset)
int	opl_reg_mmap (struct file *file, struct vm_area_struct *vma) <i>description: memory map the ipmux register into user space to access it.</i>
int	opl_reg_release (struct inode *inode, struct file *filp)

local variable declaration and definition

	u32 opl_reg_open_count = 0
	u32 opl_reg_is_open = 0
file_operations	opl_reg_fops

int __init	opl_reg_init (void) <i>description: The driver module entry to register the chrdev and request irq.</i>
void __exit	opl_reg_exit (void) <i>description: it is called when the module removed.</i>
	module_init (opl_reg_init)
	module_exit (opl_reg_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("stephanxu <hxu@opulan.com>")
	MODULE_DESCRIPTION ("dedicated the module for mmap the opulan switch chip register.")

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_reg_mmap.h

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  doxygen 1.3.9.1

|||||

opl_regmmap_test.c

```
#include <stdlib.h> #include <sys/mman.h>
#include <fcntl.h>
#include "../include/opl_typedef.h"
```

Macro definition

```
#define OPL_REG_DEVICE "/dev/opl_reg"  
#define OPL_REG_SIZE 0x300000
```

local variable declaration and definition

u32	reg_read (char *reg_base, int offset) <i>description: use to read the value of (reg_base + offset);</i>
void	reg_write (char *reg_base, int offset, u32 value) <i>description: write the u32 value into regbase + offset, if the offset is not alignment of 4, it will return and print error message.</i>
void	opl_reg_main (int argc, char **argv) <i>description: the body unit testing of reg mmap module ,it will be completed.</i>
int	opl_reg_init (int *opl_reg_initialized) <i>description: use to open the /dev/opl_reg device and do mmap() it.</i>
int	opl_reg_exit (int *opl_reg_initialized) <i>description: do close file and unmap the memory mmaped before.</i>

global variable and function exported

char *	opl_reg_base = NULL <i>HELP.</i>
char *	OPL_REG_USAGE
int	opl_reg_fd <i>global fd of the "/dev/opl_reg"</i>

```
int main (int argc, char **argv)  
description:the Entry of the unit testing for opl_regmmap module.
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_typedef.h

type definition

typedef unsigned int	u32
typedef unsigned short	u16
typedef unsigned char	u8
typedef long	s32
typedef short	s16
typedef char	s8
typedef unsigned char	bool

typedef unsigned char bool

opl_typedef.h39

typedef short s16

opl_typedef.h37

typedef long s32

opl_typedef.h36

typedef char s8

opl_typedef.h38

typedef unsigned short u16

opl_typedef.h34

typedef unsigned int u32

opl_typedef.h33

`get_mask()ipMuxRegRead()ipMuxRegWrite()opl_cpuif_test_in
turn_on_level2_intr().`

typedef unsigned char u8

opl_typedef.h35

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  *1.3.9.1*

|||||

IPMUX1IPMUX2IPMUX3

`dma_request_data_s`

`dma_request_phys_addr_s`

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  *1.3.9.1*

|||||

dma_request_data_s

```
#include <opl_cpuif.h>
```

int	dmas
int	bd
int	len
char	buf [2048]

```
int dma_request_data_s::bd
```

opl_cpuif.h47

opl_host_ioctl().

```
char dma_request_data_s::buf[2048]
```

opl_cpuif.h49

opl_host_ioctl().

```
int dma_request_data_s::dmas
```

opl_cpuif.h46

opl_host_ioctl().

```
int dma_request_data_s::len
```

opl_cpuif.h48

opl_host_ioctl().

-
- opl_cpuif.h

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  *1.3.9.1*

|||||

dma_request_phys_addr_s

```
#include <opl_cpuif.h>
```

	int	dmas
unsigned long		phys_addr

```
int dma_request_phys_addr_s::dmas
```

[opl_cpuif.h41](#)

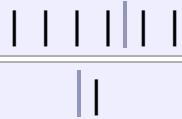
[opl_host_ioctl\(\)](#).

```
unsigned long dma_request_phys_addr_s::phys_addr
```

[opl_cpuif.h42](#)

[opl_host_ioctl\(\)](#).

-
- [opl_cpuif.h](#)
-



- bd : **dma_request_data_s**
- buf : **dma_request_data_s**
- dmas : **dma_request_data_s, dma_request_phys_addr_s**
- len : **dma_request_data_s**
- phys_addr : **dma_request_phys_addr_s**

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

IPMUX1IPMUX2IPMUX3

- **ipmux module**
 - **function_description(ipmux_interrupt.c)**
- **cpuif module**
 - **function-description(opl_cpuif.c)**
- **minite module**
 - **function_description(opl_minite.c)**
- **minite_test module**
 - **function_description(opl_minite_test.c)**
- **register mmap module**
 - **function_description(opl_reg_mmap.c)**
- **register mmap test module**
 - **function_description(opl_reg_test.c)**

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

ipmux module

THE FORMAT is FOR DOXYGEN to Generate the document opulan.com

:

stephanxu@sina.com

:

v1.0

:

June-29-2007
date description

June-29-2007 mainly from the old version and make some modification.

function_description(ipmux_interrupt.c)

Macro definition

#define	REG_INT_EN_POS_UPT	0x10204
#define	REG_INT_PEN_POS_UPT	0x10200
#define	REG_INT_EN_ATB	0x11404
#define	REG_INT_PEN_ATB	0x11400
#define	REG_INT_EN_IWF	0x13E00
#define	REG_INT_PEN_IWF	0x13E04
#define	REG_INT_EN_BRG	0x21000
#define	REG_INT_PEN_BRG	0x21004
#define	REG_INT_EN_PPE	0x18180
#define	REG_INT_PEN_PPE	0x18184
#define	REG_INT_EN_UPTM	0x14404
#define	REG_INT_PEN_UPTM	0x14400
#define	REG_INT_EN_DNTM	0x14F04
#define	REG_INT_PEN_DNTM	0x14F00
#define	REG_INT_EN_MC	0x21404
#define	REG_INT_PEN_MC	0x21400
#define	REG_INT_EN_DMA0	0x2B58
#define	REG_INT_PEN_DMA0	0x2B54
#define	OPL_UART0_IRQ	3
#define	OPL_UART1_IRQ	4
#define	OPL_HOSTDMA0_IRQ	5
#define	OPL_FEDMA1_IRQ	6
#define	OPL_IPMUX_IRQ	9

local variable declaration and definition.

u32	get_mask (char *bitmap)
char	intr_status [8][32]

global variable imported

u32	g_opl_chip_irq_event
wait_queue_head_t	g_opl_chip_waitq

```
void impux_interrupt_handler (int irq, void *dev_id, struct pt_regs *regs)
```

```
#define OPL_FEDMA1_IRQ 6
```

ipmux_interrupt.h52

```
#define OPL_HOSTDMA0_IRQ 5
```

ipmux_interrupt.h51

opl_host_dma_exit()opl_host_dma_init()opl_host_ioctl().

```
#define OPL_IPMUX_IRQ 9
```

ipmux_interrupt.h53

ipmux_hw0_irqexit()ipmux_hw0_irqinit()opl_minte_write().

```
#define OPL_UART0_IRQ 3
```

ipmux_interrupt.h49

```
#define OPL_UART1_IRQ 4
```

ipmux_interrupt.h50

```
#define REG_INT_EN_ATB 0x11404
```

ipmux_interrupt.h24

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_BRG 0x21000
```

ipmux_interrupt.h30

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_DMA0 0x2B58
```

ipmux_interrupt.h45

```
#define REG_INT_EN_DNTM 0x14F04
```

ipmux_interrupt.h39

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_IWF 0x13E00
```

ipmux_interrupt.h27

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_MC 0x21404
```

ipmux_interrupt.h42

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_POS_UPT 0x10204
```

ipmux_interrupt.h21

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_PPE 0x18180
```

ipmux_interrupt.h33

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_EN_UPTM 0x14404
```

ipmux_interrupt.h36

ipmux_hw0_irqexit()ipmux_hw0_irqinit().

```
#define REG_INT_PEN_ATB 0x11400
```

ipmux_interrupt.h25

```
#define REG_INT_PEN_BRG 0x21004
```

ipmux_interrupt.h31

```
#define REG_INT_PEN_DMA0 0x2B54
```

ipmux_interrupt.h46

```
#define REG_INT_PEN_DNTM 0x14F00
```

ipmux_interrupt.h40

```
#define REG_INT_PEN_IWF 0x13E04
```

ipmux_interrupt.h28

```
#define REG_INT_PEN_MC 0x21400
```

ipmux_interrupt.h43

```
#define REG_INT_PEN_POS_UPT 0x10200
```

ipmux_interrupt.h22

```
#define REG_INT_PEN_PPE 0x18184
```

ipmux_interrupt.h34

```
#define REG_INT_PEN_UPTM 0x14400
```

ipmux_interrupt.h37

```
u32 get_mask( char * bitmap ) [static]
```

ipmux_interrupt.c95

u32.

[turn_on_level2_intr\(\)](#).

```
void impux_interrupt_handler( int irq,  
                            void * dev_id,  
                            struct pt_regs * regs  
)
```

u32 g_opl_chip_irq_event

opl_minte.c61

ipmux_interrupt_handler()opl_minte_read().

wait_queue_head_t g_opl_chip_waitq

opl_minte.c62

ipmux_interrupt_handler()opl_minte_init()opl_minte_read().

char intr_status[8][32] [static]

```
{  
    {0, 0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0,  
     {0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
      {0, 0, 0, 0, 0, -1, 0, 0, -1, -1, -1, 0, -1, 0, -1, 0, -1, 0, 0,  
       {0, 0, 0, -1, 0, 0, 0, -1, -1, 0, 0, -1, 0, -1, 0, 0, 0, 0, 0,  
        {0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
         {0, -1, 0, -1, -1, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, -1, -1,  
          {0, -1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
           {0, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
            }
```

ipmux_interrupt.c51

ipmux_hw0_irqinit().

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  **doxygen** 1.3.9.1

|||||

function_description(ipmux_interr

[ipmux module]

local variable declaration and definition.

void	turn_on_level2_intr (u32 regID, char *bitmap) <i>description:</i> To enable the level2 interrupt.
void	turn_off_level2_intr (u32 regID) <i>description:</i>
u32	get_mask (char *bitmap)
char	intr_status [8][32]

void	ipmux_interrupt_handler (int irq, void *dev_id, struct pt_regs *regs) <i>description: the ipmux interrupt handler,only wake up the process,which to read the irqpending.</i>
int	ipmux_hw0_irqinit (void) <i>description: request the ipmux irq and enable the its intr.</i>
void	ipmux_hw0_irqexit (void) <i>description: free irq,and disable ipmux interrupt.</i>

```
u32 get_mask( char * bitmap ) [static]
```

ipmux_interrupt.c95

u32.

turn_on_level2_intr().

```
void ipmux_hw0_irqexit( void )
```

description: free irq, and disable ipmux interrupt.

:

void

:

NULL

:

ipmux_module_exit()

Deprecated:

ipmux_interrupt.c165

**OPL_IPMUX_IRQREG_INT_EN_ATBREG_INT_EN_BRGREG_INT
turn_off_level2_intr().**

```
int ipmux_hw0_irqinit( void )
```

description: request the ipmux irq and enable the its intr.

```
:  
void  
  
:  
0 success  
-1 failure  
  
:  
ipmux_module_init
```

Deprecated:

[ipmux_interrupt.c139](#)

[intr_statusipmux_interrupt_handler\(\)OPL_IPMUX_IRQREG_INT,](#)
[turn_on_level2_intr\(\)](#).

```
void ipmux_interrupt_handler( int           irq,  
                           void *        dev_id,  
                           struct pt_regs * regs  
)
```

description: the ipmux interrupt handler,only wake up the process,which to read the irqpending.

```
:  
int    irq  
void  *dev_id  
struct pt_regs regs
```

```
:  
NULL
```

```
:  
ipmux\_hw0\_irqinit
```

Deprecated:

ipmux_interrupt.c87

g_opl_chip_irq_eventg_opl_chip_waitq.

ipmux_hw0_irqinit().

void turn_off_level2_intr (u32 regID) [static]

description:

:

unsigned long regID it is the register address offset.

:

NULL

:

Deprecated:

ipmux_interrupt.c127

ipMuxRegWrite().

ipmux_hw0_irqexit().

**void turn_on_level2_intr (u32 regID,
 char * bitmap
) [static]**

description: To enable the level2 interrupt.

:

unsigned long regID it is the register address offset.
**bitmap* transfer the bitmap[] into 32bit mask.

:
NULL

Deprecated:

ipmux_interrupt.c115
get_mask()ipMuxRegWrite()u32.
ipmux_hw0_irqinit().

```
char intr_status[8][32] [static]
```

```
{
    {0, 0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0,
     {0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
      {0, 0, 0, 0, 0, -1, 0, 0, -1, -1, -1, 0, -1, 0, -1, 0, -1, 0, 0,
       {0, 0, 0, -1, 0, 0, 0, -1, -1, 0, 0, -1, 0, -1, 0, 0, 0, 0,
        {0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
         {0, -1, 0, -1, -1, -1, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, -1,
          {0, -1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
           {0, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            }
```

ipmux_interrupt.c51

ipmux_hw0_irqinit().

|||||

cpuif module

THE FORMAT is FOR DOXYGEN to Generate the document
automatically opulan.com.

:

stephanxu@sina.com

:

v0.1

:

July-30-2007 date description

----- July-30-2007

initial version for cputif module.begin data July-15-2007.

function-description(opl_cpuif.c)

Macro definition

```
#define OPL_HOST_MAJOR 130
#define OPL_HOST_NAME "opl_cpuif"
#define IPMUX_DMA_NUMS 4
#define OPL_DMA_VAL_MAX 2048
```

global variable and function imported

```
void ipmux_irq_enable (unsigned int irq)
void ipmux_irq_disable (unsigned int irq)
```

local variable declaration and definition

	int	opl_host_open (struct inode *inode, struct file *filp)
	ssize_t	opl_host_read (struct file *filp, char *buffer, size_t length, loff_t *offset)
	ssize_t	opl_host_write (struct file *filp, const char *buffer, size_t length, loff_t *offset)
	int	opl_host_release (struct inode *inode, struct file *filp)
	int	opl_host_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, u32 arg) <i>description: it is used for api to do something associated with ioctl command.</i>
	void	host_dma0_isr (int irq, void *dev_id, struct pt_regs *regs) <i>description: the host dma0 interrupt handler, when the interrupt generated, it will wake up the process, which is waiting for the host dma0 interrupt.</i>
	u32	tx_phys_addr [IPMUX_DMA_NUMS] = {0}
	u32	rx_phys_addr [IPMUX_DMA_NUMS] = {0}
	char *	dma0_rx_buf = NULL
	char *	dma0_tx_buf = NULL
	u32	order = 0
	u32	opl_host_is_open = 0
	u32	opl_host_open_count = 0
	u32	opl_host_dma0_irq_event = 0
wait_queue_head_t		opl_host_dma0_waitq
file_operations		opl_host_fops

```
#define IPMUX_DMA_NUMS 4
```

opl_cpuif.c36

```
#define OPL_DMA_VAL_MAX 2048
```

opl_cpuif.c37

opl_host_ioctl().

```
#define OPL_HOST_MAJOR 130
```

opl_cpuif.c33

opl_host_dma_exit()opl_host_dma_init().

```
#define OPL_HOST_NAME "opl_cpuif"
```

opl_cpuif.c34

opl_host_dma_exit()opl_host_dma_init().

```
void host_dma0_isr( int irq,  
                  void * dev_id,  
                  struct pt_regs * regs  
                 ) [static]
```

description: the host dma0 interrupt handler, when the interrupt generated, it will wake up the process, which is waiting for the host dma0 interrupt.

:

```
int     irq:  
void    *dev_id:  
struct pt_regs *regs:
```

:

```
NULL
```

:

Deprecated:

[opl_cpuif.c134](#)

[opl_host_dma0_irq_eventopl_host_dma0_waitq](#).

[opl_host_dma_init\(\)](#).

```
void ipmux_irq_disable( unsigned int irq )
```

```
void ipmux_irq_enable( unsigned int irq )
```

```
int opl_host_ioctl ( struct inode * inode,
                     struct file * filp,
                     unsigned int cmd,
                     u32 arg
                   ) [static]
```

description: it is used for api to do something associated with ioctl comm

:

```
    struct inode *inode:
    struct file *filp:
    unsigned int cmd: $:GET_IPMUX_DMA_RX_PHYS_ADDR:return
    u32 arg: $for PHY_ADDR: it is dma_request_phys_addr_t $for
```

:

```
    0 is success
    !0 is fail
```

:

Deprecated:

opl_cpuif.c194

**dma_request_data_s::bddma_request_data_s::bufDISABLE_IPI
WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT.**

```
int opl_host_open ( struct inode * inode,
                    struct file * filp
                  ) [static]
```

opl_cpuif.c143

opl_host_is_openopl_host_open_count.

```
ssize_t opl_host_read ( struct file * filp,
                      char *      buffer,
                      size_t       length,
                      loff_t *    offset
                    ) [static]
```

opl_cpuif.c163

```
int opl_host_release ( struct inode * inode,
                      struct file *   filp
                    ) [static]
```

opl_cpuif.c153

opl_host_is_openopl_host_open_count.

```
ssize_t opl_host_write ( struct file *   filp,
                        const char *   buffer,
                        size_t        length,
                        loff_t *     offset
                      ) [static]
```

opl_cpuif.c168

```
char* dma0_rx_buf = NULL [static]
```

opl_cpuif.c88

opl_host_dma_exit()opl_host_dma_init().

```
char* dma0_tx_buf = NULL [static]
```

opl_cpuif.c89

opl_host_dma_init().

```
u32 opl_host_dma0_irq_event = 0 [static]
```

opl_cpuif.c94

host_dma0_isr()opl_host_ioctl().

```
wait_queue_head_t opl_host_dma0_waitq [static]
```

opl_cpuif.c95

host_dma0_isr()opl_host_dma_init()opl_host_ioctl().

```
struct file_operations opl_host_fops
```

```
{  
    .open = opl_host_open,  
    .read = opl_host_read,  
    .write = opl_host_write,  
    .release = opl_host_release,  
    .ioctl = opl_host_ioctl,  
}
```

opl_cpuif.c96

opl_host_dma_init().

```
u32 opl_host_is_open = 0 [static]
```

opl_cpuif.c92

opl_host_ioctl()opl_host_open()opl_host_release().

```
u32 opl_host_open_count = 0 [static]
```

opl_cpuif.c93

opl_host_open()opl_host_release().

```
u32 order = 0 [static]
```

opl_cpuif.c91

opl_host_dma_exit()opl_host_dma_init().

```
u32 rx_phys_addr[IPMUX_DMA_NUMS] = {0} [static]
```

opl_cpuif.c86

opl_host_dma_init()opl_host_ioctl().

u32 tx_phys_addr[IPMUX_DMA_NUMS] = {0} [static]

opl_cpuif.c85

opl_host_dma_init()opl_host_ioctl().

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  **1.3.9.1**

|||||

function-description(opl_cpuif.c)

[cpuif module]

local variable declaration and definition

void	host_dma0_isr (int irq, void *dev_id, struct pt_regs *regs) <i>description: the host dma0 interrupt handler, when the interrupt generated, it will wake up the process, which is waiting for the host dma0 interrupt.</i>
int	opl_host_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, u32 arg) <i>description: it is used for api to do something associated with ioctl command.</i>

int __init	opl_host_dma_init (void) <i>description: the module's entry.</i>
void __exit	opl_host_dma_exit (void) <i>description:the module's exit.</i>
	module_init (opl_host_dma_init)
	module_exit (opl_host_dma_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("opulan Inc")
	MODULE_DESCRIPTION ("opulan IPMUX-e switch chip driver module")

```
void host_dma0_isr( int irq,  
                    void * dev_id,  
                    struct pt_regs * regs  
                    ) [static]
```

description: the host dma0 interrupt handler, when the interrupt generated, it will wake up the process, which is waiting for the host dma0 interrupt.

:

```
int irq:  
void *dev_id:  
struct pt_regs *regs:
```

:

```
NULL
```

:

Deprecated:

[opl_cpuif.c134](#)

[opl_host_dma0_irq_eventopl_host_dma0_waitq](#).

[opl_host_dma_init\(\)](#).

```
MODULE_AUTHOR( "opulan Inc" )
```

```
MODULE_DESCRIPTION( "opulan IPMUX-e switch chip driver mod
```

```
module_exit( opl_host_dma_exit )
```

```
module_init( opl_host_dma_init )
```

```
MODULE_LICENSE( "GPL2" )
```

```
void __exit opl_host_dma_exit( void ) [static]
```

description:the module's exit.

it is used for free resouces:irq,memory and unregister the device.

:

void

:

void

:

Deprecated:

opl_cpuif.c333

**dma0_rx_bufOPL_HOST_MAJOROPL_HOST_NAMEOPL_HOST
u32.**

```
int __init opl_host_dma_init( void ) [static]
```

description: the module's entry.

register the char device,request irq for host dma0 and malloc memory fo

:

void

:

0 is success.

!0 is fail

:

Deprecated:

opl_cpuif.c289

dma0_rx_bufdma0_tx_bufhost_dma0_isr()opl_host_dma0_wait.

```
int opl_host_ioctl( struct inode * inode,
                    struct file * filp,
                    unsigned int cmd,
                    u32 arg
                  ) [static]
```

description: it is used for api to do something associated with ioctl comm

:

*struct inode *inode:*

*struct file *filp:*

unsigned int cmd: \$:GET_IPMUX_DMA_RX_PHYS_ADDR:return

u32 arg: \$for PHY_ADDR: it is dma_request_phys_addr_t \$for

O is success

!O is fail

Deprecated:

opl_cpuif.c194

**dma_request_data_s::bddma_request_data_s::bufDISABLE_IPI
WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT.**

|||||

minte module

THE FORMAT is FOR DOXYGEN to Generate the document
opulan.com.

:

stephanxu@sina.com

:

v0.2

:

June-29-2007 date description

----- June-29-2007

mainly from the old version and rewrite the opulan_read function.

July-05-2007 distill the interrupt handle way from the old ipmux driver module.

July-09-2007 change transfer global g_opl_chip_irq_pending to
get_irq_pending() and **clear_irq_pending()**.

	function_description(opl_minte.c)
#define	OPL_MINTE_MAJOR 120
#define	OPL_MINTE_DEVICE "opl_minte"

Macro definition

#define	opl_minte_hw0_irqexit	ipmux_hw0_irqexit
#define	opl_minte_hw0_irqinit	ipmux_hw0_irqinit
int	ipmux_hw0_irqinit (void)	<i>description: request the ipmux irq and enable the its intr.</i>
void	ipmux_hw0_irqexit (void)	<i>description: free irq, and disable ipmux interrupt.</i>

the variable and function imported.

<code>u32</code>	<code>get_irq_pending</code> (void)
<code>void</code>	<code>clear_irq_pending</code> (void)
<code>void</code>	<code>ipmux_irq_enable</code> (unsigned int irq)
<code>void</code>	<code>ipmux_irq_disable</code> (unsigned int irq)

local function declaration

int	opl_minte_open (struct inode *inode, struct file *filp)
int	opl_minte_release (struct inode *inode, struct file *filp)
ssize_t	opl_minte_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description: copy the opl_chip_irq_pending(indicate which interrupt occurred) to user App, which will handle its interrupt.if the no irq pending occurred,the process will sleep until the isr wake up it.</i>
ssize_t	opl_minte_write (struct file *filp, const char *buffer, size_t length, loff_t *offset) <i>description: because the enable irq or disable irq will be in atomic mode,so it should be operated in kernel mode is correct.</i>

the global variable declaration and definition.

u32	g_opl_chip_irq_event = 0
wait_queue_head_t	g_opl_chip_waitq

local variable declaration and definition

	u32 <code>opl_minte_open_count</code> = 0
	u32 <code>opl_minte_is_open</code> = 0
file_operations	<code>opl_minte_fops</code>

```
#define OPL_MINTE_DEVICE "opl_minte"
```

opl_minte.c36

opl_minte_exit()opl_minte_init().

```
#define opl_minte_hw0_irqexit ipmux_hw0_irqexit
```

opl_minte.h23

opl_minte_exit().

```
#define opl_minte_hw0_irqinit ipmux_hw0_irqinit
```

opl_minte.h24

opl_minte_init().

```
#define OPL_MINTE_MAJOR 120
```

opl_minte.c35

opl_minte_exit()opl_minte_init().

```
void clear_irq_pending( void )
```

opl_minte_read().

```
u32 get_irq_pending( void )
```

opl_minte_read().

```
void ipmux_hw0_irqexit( void )
```

description: free irq, and disable ipmux interrupt.

:

void

:

NULL

:

ipmux_module_exit()

Deprecated:

ipmux_interrupt.c165

**OPL_IPMUX_IRQREG_INT_EN_ATBREG_INT_EN_BRGREG_INT
turn_off_level2_intr()**.

```
int ipmux_hw0_irqinit( void )
```

description: request the ipmux irq and enable the its intr.

:

void

:

0 success

-1 failure

:

`ipmux_module_init`

Deprecated:

`ipmux_interrupt.c139`

`intr_statusipmux_interrupt_handler()OPL_IPMUX_IRQREG_INT,`
`turn_on_level2_intr()`.

`void ipmux_irq_disable(unsigned int irq)`

`opl_host_ioctl()opl_minte_write()`.

`void ipmux_irq_enable(unsigned int irq)`

`opl_host_ioctl()opl_minte_write()`.

`int opl_minte_open(struct inode * inode,`
`struct file * filp`
`) [static]`

opl_minte.c163

opl_minte_is_openopl_minte_open_count.

```
ssize_t opl_minte_read( struct file * filp,
                        char *      buffer,
                        size_t       length,
                        loff_t *    offset
                      ) [static]
```

description: copy the opl_chip_irq_pending(indicate which interrupt occu its interrupt.if the no irq pending occured,the process will sleep until the i

:

struct file **filp*
 char **buffer*
 size_t *length*
 loff_t *offset*

:

 the length of bytes process read.

:

 opl_minte_interrupt_handler

Deprecated:

opl_minte.c109

clear_irq_pending()g_opl_chip_irq_eventg_opl_chip_waitqget_i u32.

```
int opl_minte_release( struct inode * inode,
                       struct file *   filp
```

```
) [static]
```

opl_minte.c172

opl_minte_is_openopl_minte_open_count.

```
ssize_t opl_minte_write ( struct file * filp,
                           const char * buffer,
                           size_t length,
                           loff_t * offset
) [static]
```

description: because the enable irq or disable irq will be in atomic mode, should be operated in kernel mode is correct.

the value in user *buffer it is the enable or disable. 1. enable 0. disable if value is enable, to enable the ipmux interrupt, call the function in malta_in

:

@param const char *buffer: contain the value(0-disable or 1-enable)

:

0 is success

!0 failure. the device is not open or copy *buf from user to kernel fa

:

Deprecated:

opl_minte.c146

```
ipmux_irq_disable()ipmux_irq_enable()OPL_IPMUX_IRQ
opl_minte_is_open.
```

```
u32 g_opl_chip_irq_event = 0
```

opl_minte.c61

ipmux_interrupt_handler()opl_minte_read().

```
wait_queue_head_t g_opl_chip_waitq
```

opl_minte.c62

ipmux_interrupt_handler()opl_minte_init()opl_minte_read().

```
struct file_operations opl_minte_fops
```

```
{  
    .read = opl_minte_read,  
    .write = opl_minte_write,  
    .open = opl_minte_open,  
    .release = opl_minte_release,  
}
```

opl_minte.c83

opl_minte_init().

```
u32 opl_minte_is_open = 0 [static]
```

opl_minte.c82

**opl_minte_open()opl_minte_read()opl_minte_release()
opl_minte_write().**

```
u32 opl_minte_open_count = 0 [static]
```

opl_minte.c81

opl_minte_open()opl_minte_release().

|||||

function_description(opl_minte.c)

[minte module]

local function declaration

ssize_t	opl_minte_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description: copy the opl_chip_irq_pending(indicate which interrupt occurred) to user App, which will handle its interrupt.if the no irq pending occurred,the process will sleep until the isr wake up it.</i>
ssize_t	opl_minte_write (struct file *filp, const char *buffer, size_t length, loff_t *offset) <i>description: because the enable irq or disable irq will be in atomic mode,so it should be operated in kernel mode is correct.</i>

int __init	opl_minte_init (void) <i>description: The driver module entry to register the chrdev and request irq.</i>
void __exit	opl_minte_exit (void) <i>description: it is called when the module removed.</i>
	module_init (opl_minte_init)
	module_exit (opl_minte_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("stephanxu <hxu@opulan.com>")
	MODULE_DESCRIPTION ("control interface for opulan ipmux-e switch chip")

```
MODULE_AUTHOR( "stephanxu <hxu@opulan.com>" )
```

```
MODULE_DESCRIPTION( "control interface for opulan ipmux-e sw" )
```

```
module_exit( opl_minte_exit )
```

```
module_init( opl_minte_init )
```

```
MODULE_LICENSE( "GPL2" )
```

```
void __exit opl_minte_exit( void ) [static]
```

description: it is called when the module removed.

do the reverse of [opl_minte_init\(\)](#).

:

void

:

NULL

:

Deprecated:

opl_minte.c219

OPL_MINTE_DEVICEopl_minte_hw0_irqexit
OPL_MINTE_MAJOR.

main().

```
int __init opl_minte_init( void ) [static]
```

description: The driver module entry.to register the chrdev and request ir
and some essential initialization.

:

void

:

0 success

!0 failure.the module cant be inserted into os.

:

insmod rmmod

Deprecated:

opl_minte.c189

g_opl_chip_waitqOPL_MINTE_DEVICEopl_minte_fopsopl_minte
OPL_MINTE_MAJOR.

main().

```
ssize_t opl_minte_read ( struct file * filp,
                        char *      buffer,
                        size_t       length,
                        loff_t *    offset
                      ) [static]
```

description: copy the opl_chip_irq_pending(indicate which interrupt occurs its interrupt.if the no irq pending occurred,the process will sleep until the interrupt occurs)

:

*struct file *filp*
 *char *buffer*
 size_t length
 loff_t offset

:

 the length of bytes process read.

:

 opl_minte_interrupt_handler

Deprecated:

[opl_minte.c109](#)

[clear_irq_pending\(\)](#)[g_opl_chip_irq_event](#)[g_opl_chip_waitqget_i](#)
[u32.](#)

```
ssize_t opl_minte_write ( struct file * filp,
                         const char * buffer,
                         size_t       length,
                         loff_t *    offset
                       ) [static]
```

description: because the enable irq or disable irq will be in atomic mode,

should be operated in kernel mode is correct.

the value in user *buffer it is the enable or disable. 1. enable 0. disable if
value is enable, to enable the ipmux interrupt, call the function in malta_in

:

*@param const char *buffer:* contain the value(0-disable or 1-enable)

:

0 is success

!0 failure. the device is not open or copy *buf from user to kernel fa

:

Deprecated:

opl_minte.c146

ipmux_irq_disable()ipmux_irq_enable()OPL_IPMUX_IRQ
opl_minte_is_open.

|||||

minte_test module

THE FORMAT is FOR DOXYGEN to Generate the document
automatically opulan.com

.

:

stephanxu@sina.com

:

v1.0

:

July-07-2007

author date description

----- July-07-2007 the
initial version of testing the minte module.

```
function_description(opl_minte_test.c)
```

Macro definition

```
#define OPL_MINTE_DEVICE "/dev/opl_minite"
```

local function declaration

```
int opl_minte_fd  
the global fd of the "/dev/opl_minte"
```

```
#define OPL_MINTE_DEVICE "/dev/opl_minte"
```

opl_minte_test.c22

```
int opl_minte_fd
```

the global fd of the "/dev/opl_minte"

opl_minte_test.c53

opl_minte_exit()opl_minte_init()opl_minte_main().

|||||

function_description(opl_minte_te
[minte_test module]

local variable declaration and definition

int	opl_minte_init (int *initialized) <i>description: it is used to open the /dev/opl_minte device with flag: RDWR SYNC.</i>
int	opl_minte_exit (int *initialized) <i>description: close the opened file "/dev/opl_minte".</i>
void	opl_minte_main (int argc, char **argv) <i>description: it is the example of how to use it to check the value.</i>

```
int main (int argc, char **argv)
```

description: the testing entry for minte unit testing programming.

```
int main( int      argc,
          char ** argv
        )
```

description: the testing entry for minte unit testing programming.

:

@param

:

@retval

:

Deprecated:

if the testing will be intergrated into main testing, it will be removed.

opl_minte_test.c162

opl_minte_exit()opl_minte_init()opl_minte_main().

```
int opl_minte_exit( int * initialized )
```

description: close the opened file "/dev/opl_minte".

:

**initialized,that* means close it or not depend on initialized value.then clear it.

:

0 success

-X close the file failed.

:

Deprecated:

opl_minte_test.c120

opl_minte_fd.

int opl_minte_init (int * *initialized*)

description: it is used to open the /dev/opl_minte device with flag:
RDWR|SYNC.

:

**initialized* is the output value to indicated it is initialized or not.

:

0 success

-X failed.

:

Deprecated:

opl_minte_test.c101

OPL_MINTE_DEVICEopl_minte_fd.

**void opl_minte_main (int argc,
 char ** argv
)**

description: it is the example of how to use it to check the value.
and it is the body of unit testing programing, which will be completed.

```
:
```

int argc
char **argv

```
:
```

void *

```
:
```

Deprecated:

opl_minte_test.c139

opl_minte_fdu32.

main().

|||||

register mmap module

THE FORMAT is FOR DOXYGEN to Generate the document
opulan.com.

:

stephanxu@sina.com

:

v0.2

:

June-29-2007 author date description -----
----- hxu June-29-2007 mainly from the old version
and rewrite the opulan_read function.
hxu July-05-2007 distill the piece of the ipmux driver code dedicated
for the opl switch chip register map.

	function_description(opl_reg_mmap.c)
#define	OPL_REG_MAJOR 110
#define	OPL_REG_DEVICE "opl_reg"

local function declaration

int	opl_reg_open (struct inode *inode, struct file *filp)
ssize_t	opl_reg_write (struct file *filp, const char *buffer, size_t length, loff_t *offset)
int	opl_reg_release (struct inode *inode, struct file *filp)
ssize_t	opl_reg_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description:</i>
int	opl_reg_mmap (struct file *file, struct vm_area_struct *vma) <i>description: memory map the ipmux register into user space to access it.</i>

local variable declaration and definition

	u32	opl_reg_open_count = 0
	u32	opl_reg_is_open = 0
file_operations		opl_reg_fops

```
#define OPL_REG_DEVICE "opl_reg"
```

opl_reg mmap.c32

opl_reg_exit()opl_reg_init().

```
#define OPL_REG_MAJOR 110
```

opl_reg mmap.c31

opl_reg_exit()opl_reg_init().

```
int opl_reg_mmap( struct file * file,  
                  struct vm_area_struct * vma  
) [static]
```

description: memory map the ipmux register into user space to access it.

which will reduce the spense on copy value between kernel space with user space. and reduce the dependency on interface of OS. the ipmux register must be NOcached and NO swap out and NO Coredump.

:

*struct file ***file*
 *struct vm_area_struct ***vma*

:

 0 success.
 -(x) failure.

:

 mmap manual.

Deprecated:

[opl_reg_mmap.c124](#)

```
int opl_reg_open( struct inode * inode,  
                  struct file * filp  
) [static]
```

[opl_reg_mmap.c150](#)

`opl_reg_is_openopl_reg_open_count.`

```
ssize_t opl_reg_read( struct file * filp,
                      char *      buffer,
                      size_t       length,
                      loff_t *    offset
                    ) [static]
```

description:

:

@param
@return

:

@retval

:

Deprecated:

`opl_reg_mmap.c102`

```
int opl_reg_release( struct inode * inode,
                     struct file *   filp
                   ) [static]
```

`opl_reg_mmap.c159`

`opl_reg_is_openopl_reg_open_count.`

```
ssize_t opl_reg_write ( struct file * filp,  
                      const char * buffer,  
                      size_t length,  
                      loff_t * offset  
) [static]
```

opl_reg_mmap.c107

struct file_operations opl_reg_fops

```
{  
    .read = opl_reg_read,  
    .write = opl_reg_write,  
    .mmap = opl_reg_mmap,  
    .open = opl_reg_open,  
    .release = opl_reg_release,  
}
```

opl_reg_mmap.c75

opl_reg_init().

u32 opl_reg_is_open = 0 [static]

opl_reg_mmap.c74

opl_reg_open()opl_reg_release().

u32 opl_reg_open_count = 0 [static]

opl_reg_mmap.c73

opl_reg_open()opl_reg_release().

|||||

function_description(opl_reg mmap)

[register mmap module]

local function declaration

ssize_t	opl_reg_read (struct file *filp, char *buffer, size_t length, loff_t *offset) <i>description:</i>
---------	---

int	opl_reg_mmap (struct file *file, struct vm_area_struct *vma) <i>description: memory map the ipmux register into user space to access it.</i>
-----	--

int __init	opl_reg_init (void) <i>description: The driver module entry to register the chrdev and request irq.</i>
void __exit	opl_reg_exit (void) <i>description: it is called when the module removed.</i>
	module_init (opl_reg_init)
	module_exit (opl_reg_exit)
	MODULE_LICENSE ("GPL2")
	MODULE_AUTHOR ("stephanxu <hxu@opulan.com>")
	MODULE_DESCRIPTION ("dedicated the module for mmap the opulan switch chip register.")

```
MODULE_AUTHOR( "stephanxu <hxu@opulan.com>" )
```

```
MODULE_DESCRIPTION( "dedicated the module for mmap the op
```

```
module_exit( opl_reg_exit )
```

```
module_init( opl_reg_init )
```

```
MODULE_LICENSE( "GPL2" )
```

```
void __exit opl_reg_exit( void ) [static]
```

description: it is called when the module removed.

do the reverse of opl_reg_module_init.

:

void

:

NULL

:

Deprecated:

opl_reg_mmap.c193

OPL_REG_DEVICEOPL_REG_MAJOR.

main().

```
int __init opl_reg_init( void ) [static]
```

description: The driver module entry.to register the chrdev and request irq.

and some essential initialization.

:

void

:

0 success

!0 failure.the module cant be inserted into os.

:

insmod rmmod

Deprecated:

opl_reg_mmap.c176

OPL_REG_DEVICEopl_reg_fopsOPL_REG_MAJOR.

main().

```
int opl_reg_mmap( struct file *           file,
```

```
    struct vm_area_struct * vma  
) [static]
```

description: memory map the ipmux register into user space to access it.

which will reduce the spense on copy value between kernel space with user space. and reduce the dependency on interface of OS. the ipmux register must be NOcached and NO swap out and NO Coredump.

:

```
    struct file *file  
    struct vm_area_struct *vma
```

:

```
    0    success.  
-(x) failure.
```

:

```
    mmap manual.
```

Deprecated:

[opl_reg_mmap.c124](#)

```
ssize_t opl_reg_read( struct file * filp,  
                      char *      buffer,  
                      size_t       length,  
                      loff_t *    offset  
) [static]
```

description:

:

```
@param
```

@return

:

@retval

:

Deprecated:

opl_reg_mmap.c102

|||||

register mmap test module

THE FORMAT is FOR DOXYGEN to Generate the document
automatically opulan.com.

:

stephanxu@sina.com

:

v1.0

:

July-07-2007 author date description -----
----- hxu July-07-2007 the initial version of the register
mmap unit testing code

```
function_description(opl_reg_test.c)
```

Macro definition

```
#define OPL_REG_DEVICE "/dev/opl_reg"  
#define OPL_REG_SIZE 0x300000
```

global variable and function exported

char *	opl_reg_base = NULL <i>HELP.</i>
char *	OPL_REG_USAGE
int	opl_reg_fd <i>global fd of the "/dev/opl_reg"</i>

```
#define OPL_REG_DEVICE "/dev/opl_reg"
```

opl_reg mmap _test.c23

```
#define OPL_REG_SIZE 0x300000
```

opl_reg mmap _test.c24

opl_reg_exit()opl_reg_init().

```
char* opl_reg_base = NULL
```

HELP.

```
opl_regmmmap_test.c85
```

```
opl_reg_exit()opl_reg_init()opl_reg_main().
```

```
int opl_reg_fd
```

global fd of the "/dev/opl_reg"

```
opl_regmmmap_test.c94
```

```
opl_reg_exit()opl_reg_init().
```

```
char* OPL_REG_USAGE
```

```
"opl_reg_test -r offset or\n\  
        opl_reg_test --read offset or\n\  
        opl_reg_test -w offset value or\n\  
        opl_reg_test --write offset value \n\  
Note: the offset and value can be decimal or hex  
      "
```

```
opl_regmmmap_test.c86
```

```
opl_reg_main().
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  **1.3.9.1**

|||||

function_description(opl_reg_test

[register mmap test module]

local variable declaration and definition

u32	reg_read (char *reg_base, int offset) <i>description: use to read the value of (reg_base + offset);</i>
void	reg_write (char *reg_base, int offset, u32 value) <i>description: write the u32 value into regbase + offset, if the offset is not alignment of 4, it will return and print error message.</i>
void	opl_reg_main (int argc, char **argv) <i>description: the body unit testing of reg mmap module ,it will be completed.</i>
int	opl_reg_init (int *opl_reg_initialized) <i>description: use to open the /dev/opl_reg device and do mmap() it.</i>
int	opl_reg_exit (int *opl_reg_initialized) <i>description: do close file and unmap the memory mmaped before.</i>

```
int main (int argc, char **argv)
```

description:the Entry of the unit testing for opl_regmmap module.

```
int main( int      argc,
          char ** argv
        )
```

description:the Entry of the unit testing for opl_regmmap module.

:

@param

:

@retval

:

Deprecated:

[opl_regmmap_test.c247](#)

[opl_reg_exit\(\)opl_reg_init\(\)opl_reg_main\(\)](#).

```
int opl_reg_exit( int * opl_reg_initialized )
```

description: do close file and unmap the memory mmaped before.

:

**opl_reg_initialized*: if initialized,exit will close it and do unmap it.

:

0 success

-1 failed

:

Deprecated:

opl_reg mmap _test.c227

opl_reg_baseopl_reg_fdOPL_REG_SIZE.

```
int opl_reg_init( int * opl_reg_initialized )
```

description: use to open the /dev/opl_reg device and do mmap() it.

:

**opl_reg_initialized*,1 indicated open device and mmap successfully.

:

0 success
-x failed.

:

Deprecated:

opl_reg mmap _test.c203

opl_reg_baseOPL_REG_DEVICEopl_reg_fdOPL_REG_SIZE.

```
void opl_reg_main( int      argc,  
                   char ** argv  
                 )
```

description: the body unit testing of reg mmap module ,it will be completed.

```
:  
    int    argc  
    char **argv
```

```
:  
    void
```

Deprecated:

opl_reg mmap _test.c151

opl_reg_baseOPL_REG_USAGereg_read()reg_write()u32.
main().

```
u32 reg_read( char * reg_base,  
             int      offset  
             )
```

description: use to read the value of (reg_base + offset);

```
:  
    *reg_base: the base address of mmap().  
    offset      the offset of the reg_base.
```

```
:  
    the value(u32) of the (reg_base +offset)
```

```
:  
    -1 indicate the offset is not alignment of 4. or the value is 0xffff,ffff.
```

```
:
```

Deprecated:

opl_reg mmap _test.c115

u32.

opl_reg_main().

```
void reg_write( char * reg_base,
                int     offset,
                u32    value
)
```

description: write the u32 value into regbase + offset,if the offset is not alignment of 4, it will return and print error message.

:
 **reg_base*: the base address of mmap().
 offset the offset of the *reg_base*.

:

void

Deprecated:

opl_reg mmap _test.c134

u32.

opl_reg_main().

b	c	d	e	g	h	i	m	o	r	s	t	u	w

- b -

- bool : **opl_typedef.h**

- c -

- clear_irq_pending() : **opl_minte.c**

- d -

- DISABLE_IPMUX_HOST_DMA0_INTERRUPT : **opl_cpuif.h**
- dma0_rx_buf : **opl_cpuif.c**
- dma0_tx_buf : **opl_cpuif.c**
- dma_request_data_t : **opl_cpuif.h**
- dma_request_phys_addr_t : **opl_cpuif.h**

- e -

- ENABLE_IPMUX_HOST_DMA0_INTERRUPT : **opl_cpuif.h**

- g -

- g_opl_chip_irq_event : **opl_minte.c**, **ipmux_interrupt.c**
- g_opl_chip_waitq : **opl_minte.c**, **ipmux_interrupt.c**
- GET_IPMUX_DMA_RX_BUF_DATA : **opl_cpuif.h**
- GET_IPMUX_DMA_RX_PHYS_ADDR : **opl_cpuif.h**
- GET_IPMUX_DMA_TX_PHYS_ADDR : **opl_cpuif.h**
- get_irq_pending() : **opl_minte.c**
- get_mask() : **ipmux_interrupt.c**

- h -

- host_dma0_isr() : **opl_cpuif.c**

- i -

- impux_interrupt_handler() : **ipmux_interrupt.c**
- intr_status : **ipmux_interrupt.c**
- IPMUX_DMA_NUMS : **opl_cpuif.c**
- ipmux_hw0_irqexit() : **opl_minte.h, ipmux_interrupt.c**
- ipmux_hw0_irqinit() : **opl_minte.h, ipmux_interrupt.c**
- ipmux_interrupt_handler() : **ipmux_interrupt.c**
- ipmux_irq_disable() : **opl_minte.c, opl_cpuif.c**
- ipmux_irq_enable() : **opl_minte.c, opl_cpuif.c**
- IPMUX_REG_ADDR2ID : **opl_hw_ops.h**
- IPMUX_REG_BASE : **opl_hw_ops.h**
- IPMUX_REG_ID2ADDR : **opl_hw_ops.h**
- ipMuxRegRead() : **opl_hw_ops.h**
- ipMuxRegWrite() : **opl_hw_ops.h**

- m -

- main() : **opl_regmmap_test.c, opl_minte_test.c, opl_cpuif_test.c**
- MODULE_AUTHOR() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- MODULE_DESCRIPTION() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- module_exit() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- module_init() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- MODULE_LICENSE() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**

- o -

- OPL_CPUIF_DEVICE : **opl_cpuif_test.c**
- opl_cpuif_test_exit() : **opl_cpuif_test.c**
- opl_cpuif_test_init() : **opl_cpuif_test.c**
- opl_cpuif_test_main() : **opl_cpuif_test.c**
- OPL_DMA_VAL_MAX : **opl_cpuif.c**
- OPL_FEDMA1_IRQ : **ipmux_interrupt.h**

- opl_host_dma0_irq_event : **opl_cpuif.c**
- opl_host_dma0_waitq : **opl_cpuif.c**
- opl_host_dma_exit() : **opl_cpuif.c**
- opl_host_dma_init() : **opl_cpuif.c**
- opl_host_fops : **opl_cpuif.c**
- opl_host_ioctl() : **opl_cpuif.c**
- opl_host_is_open : **opl_cpuif.c**
- OPL_HOST_MAJOR : **opl_cpuif.c**
- OPL_HOST_NAME : **opl_cpuif.c**
- opl_host_open() : **opl_cpuif.c**
- opl_host_open_count : **opl_cpuif.c**
- opl_host_read() : **opl_cpuif.c**
- opl_host_release() : **opl_cpuif.c**
- opl_host_write() : **opl_cpuif.c**
- OPL_HOSTDMA0_IRQ : **ipmux_interrupt.h**
- OPL_IPMUX_IRQ : **ipmux_interrupt.h**
- OPL_MINTE_DEVICE : **opl_minte_test.c, opl_minte.c**
- opl_minte_exit() : **opl_minte_test.c, opl_minte.c**
- opl_minte_fd : **opl_minte_test.c**
- opl_minte_fops : **opl_minte.c**
- opl_minte_hw0_irqexit : **opl_minte.h**
- opl_minte_hw0_irqinit : **opl_minte.h**
- opl_minte_init() : **opl_minte_test.c, opl_minte.c**
- opl_minte_is_open : **opl_minte.c**
- opl_minte_main() : **opl_minte_test.c**
- OPL_MINTE_MAJOR : **opl_minte.c**
- opl_minte_open() : **opl_minte.c**
- opl_minte_open_count : **opl_minte.c**
- opl_minte_read() : **opl_minte.c**
- opl_minte_release() : **opl_minte.c**
- opl_minte_write() : **opl_minte.c**
- opl_reg_base : **opl_reg mmap_test.c**
- OPL_REG_DEVICE : **opl_reg mmap_test.c, opl_reg mmap.c**
- opl_reg_exit() : **opl_reg mmap_test.c, opl_reg mmap.c**
- opl_reg_fd : **opl_reg mmap_test.c**
- opl_reg_fops : **opl_reg mmap.c**
- opl_reg_init() : **opl_reg mmap_test.c, opl_reg mmap.c**
- opl_reg_is_open : **opl_reg mmap.c**
- opl_reg_main() : **opl_reg mmap_test.c**

- OPL_REG_MAJOR : [opl_reg_mmap.c](#)
- opl_reg_mmap() : [opl_reg_mmap.c](#)
- opl_reg_open() : [opl_reg_mmap.c](#)
- opl_reg_open_count : [opl_reg_mmap.c](#)
- opl_reg_read() : [opl_reg_mmap.c](#)
- opl_reg_release() : [opl_reg_mmap.c](#)
- OPL_REG_SIZE : [opl_regmmap_test.c](#)
- OPL_REG_USAGE : [opl_regmmap_test.c](#)
- opl_reg_write() : [opl_reg_mmap.c](#)
- OPL_UART0_IRQ : [ipmux_interrupt.h](#)
- OPL_UART1_IRQ : [ipmux_interrupt.h](#)
- order : [opl_cpuif.c](#)

- r -

- REG_INT_EN_ATB : [ipmux_interrupt.h](#)
- REG_INT_EN_BRG : [ipmux_interrupt.h](#)
- REG_INT_EN_DMA0 : [ipmux_interrupt.h](#)
- REG_INT_EN_DNTM : [ipmux_interrupt.h](#)
- REG_INT_EN_IWF : [ipmux_interrupt.h](#)
- REG_INT_EN_MC : [ipmux_interrupt.h](#)
- REG_INT_EN_POS_UPT : [ipmux_interrupt.h](#)
- REG_INT_EN_PPE : [ipmux_interrupt.h](#)
- REG_INT_EN_UPTM : [ipmux_interrupt.h](#)
- REG_INT_PEN_ATB : [ipmux_interrupt.h](#)
- REG_INT_PEN_BRG : [ipmux_interrupt.h](#)
- REG_INT_PEN_DMA0 : [ipmux_interrupt.h](#)
- REG_INT_PEN_DNTM : [ipmux_interrupt.h](#)
- REG_INT_PEN_IWF : [ipmux_interrupt.h](#)
- REG_INT_PEN_MC : [ipmux_interrupt.h](#)
- REG_INT_PEN_POS_UPT : [ipmux_interrupt.h](#)
- REG_INT_PEN_PPE : [ipmux_interrupt.h](#)
- REG_INT_PEN_UPTM : [ipmux_interrupt.h](#)
- reg_read() : [opl_regmmap_test.c](#)
- reg_write() : [opl_regmmap_test.c](#)
- rx_phys_addr : [opl_cpuif.c](#)

- s -

- s16 : [opl_typedef.h](#)
- s32 : [opl_typedef.h](#)
- s8 : [opl_typedef.h](#)
- SET_IPMUX_DMA_TX_BUF_DATA : [opl_cpuif.h](#)

- t -

- turn_off_level2_intr() : [ipmux_interrupt.c](#)
- turn_on_level2_intr() : [ipmux_interrupt.c](#)
- tx_phys_addr : [opl_cpuif.c](#)

- u -

- u16 : [opl_typedef.h](#)
- u32 : [opl_typedef.h](#)
- u8 : [opl_typedef.h](#)

- w -

- WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT : [opl_cpuif.h](#)

|||||

IPMUX1IPMUX2IPMUX3

•

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  *1.3.9.1*

|||||

turn_on_level2_intr (u32 regID, char *bitmap)

turn_off_level2_intr (u32 regID)

ipmux_hw0_irqexit (void)

ipmux_hw0_irqinit (void)

ipmux_interrupt_handler (int irq, void *dev_id, struct pt_regs *regs)

opl_host_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, u32 arg)

host_dma0_isr (int irq, void *dev_id, struct pt_regs *regs)

opl_host_dma_exit (void)

opl_host_dma_init (void)

opl_minte_read (struct file *filp, char *buffer, size_t length, loff_t *offset)

opl_minte_write (struct file *filp, const char *buffer, size_t length, loff_t *offset)

opl_minte_exit (void)

opl_minte_init (void)

opl_minte_init (int *initialized)

opl_minte_exit (int *initialized)

opl_minte_main (int argc, char **argv)

main (int argc, char **argv)

if the testing will be intergrated into main testing, it will be removed.

opl_reg_read (struct file *filp, char *buffer, size_t length, loff_t *offset)

opl_reg_mmap (struct file *file, struct vm_area_struct *vma)

opl_reg_exit (void)

opl_reg_init (void)

reg_read (char *reg_base, int offset)

reg_write (char *reg_base, int offset, u32 value)

opl_reg_main (int argc, char **argv)

opl_reg_init (int *opl_reg_initialized)

opl_reg_exit (int *opl_reg_initialized)

main (int argc, char **argv)

|||||

ipmux_interrupt.c

```
00001
00014 #include <linux/wait.h>
00015 #include <asm/bitops.h>
00016 #include <linux/sched.h>
00017 #include <linux/kernel.h>
00018
00019
00020 #include <ipmux_interrupt.h>
00021 #include <opl_hw_ops.h>
00025 /*-----Macro definition-----*/
00026
00027
00033 /*-----type definition-----*/
00034
00040 /*-----global variable/function declaration-----*/
00041 extern u32 g_opl_chip_irq_event;
00042 extern wait_queue_head_t g_opl_chip_waitq;
00043
00049 /*-----local variable declaration and definition----*/
00050 /* why I reserve it? consistency.change from int to char. */
00051 static char intr_status[8][32] = {
00052 /* bit 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
00053 /*upt*/ {0, 0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0
00054 /*misc*/ {0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
00055 /*iwf*/ {0, 0, 0, 0, 0, -1, 0, 0, -1, -1, -1, 0, -1, 0, -1, 0, -1, 0
00056 /*brg*/ {0, 0, 0, -1, 0, 0, 0, -1, -1, 0, 0, -1, 0, -1, 0, 0, 0
00057 /*ppe*/ {0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
00058 /*uptm*/ {0, -1, 0, -1, -1, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, -1
00059 /*dntm*/ {0, -1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
00060 /*mc*/ {0, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1
00061 };
00062
00063
00064
00065 static void turn_on_level2_intr(u32 regID,char *bitmap);
00066 static void turn_off_level2_intr(u32 regID);
00067 static u32 get_mask(char *bitmap);
00068
00071 void impux_interrupt_handler(int irq,void *dev_id,struct pt_r
00072
00073
00077 /*-----local function declaration and definition-----
```

```

00087 void ipmux_interrupt_handler(int irq, void *dev_id, struct pt_r
00088 {
00089     if(test_and_set_bit(0,&g_opl_chip_irq_event)){
00090         /* TBD: add some debug message.if it is set, some bugs ex
00091     }
00092     wake_up_interruptible(&g_opl_chip_waitq);
00093 }
00094
00095 static u32 get_mask(char *bitmap)
00096 {
00097     int i = 0;
00098     u32 mask = 0;
00099     for (i = 0; i < 32; ++i) {
00100         if (bitmap[i] == -1) {
00101             mask |= (1<<i);
00102         }
00103     }
00104     return ~mask;
00105 }
00106
00115 static void turn_on_level2_intr(u32 regID, char *bitmap)
00116 {
00117     u32 mask = get_mask(bitmap);
00118     ipMuxRegWrite(regID,mask);
00119 }
00127 static void turn_off_level2_intr(u32 regID)
00128 {
00129     ipMuxRegWrite(regID,0);
00130 }
00139 int ipmux_hw0_irqinit(void)
00140 {
00141     /* request the ipmux_irq for ipmux switch,all of them share
00142     if(request_irq(OPL_IPMUX_IRQ,ipmux_interrupt_handler,
00143                     0,"IPMUX interrupt",NULL) != 0){
00144         return -1;
00145     }
00146     /* turn on irqs of ipmux:pos_upt,atb,iwf,bridge,ppe,uptom,d
00147     turn_on_level2_intr(REG_INT_EN_POS_UPT, intr_status[0]);
00148     turn_on_level2_intr(REG_INT_EN_ATB, intr_status[1]);
00149     turn_on_level2_intr(REG_INT_EN_IWF, intr_status[2]);
00150     turn_on_level2_intr(REG_INT_EN_BRG, intr_status[3]);
00151     turn_on_level2_intr(REG_INT_EN_PPE, intr_status[4]);
00152     turn_on_level2_intr(REG_INT_EN_UPTM, intr_status[5]);
00153     turn_on_level2_intr(REG_INT_EN_DNTM, intr_status[6]);
00154     turn_on_level2_intr(REG_INT_EN_MC, intr_status[7]);
00155
00156     return 0;
00157 }
00165 void ipmux_hw0_irqexit(void)
00166 {

```

```
00167 /* turn off the irqs of ipmux:pos,atb,iwf,bridge,ppe,uptm,d
00168 turn_off_level2_intr(REG_INT_EN_POS_UPT);
00169 turn_off_level2_intr(REG_INT_EN_ATB);
00170 turn_off_level2_intr(REG_INT_EN_IWF);
00171 turn_off_level2_intr(REG_INT_EN_BRG);
00172 turn_off_level2_intr(REG_INT_EN_PPE);
00173 turn_off_level2_intr(REG_INT_EN_UPTM);
00174 turn_off_level2_intr(REG_INT_EN_DNTM);
00175 turn_off_level2_intr(REG_INT_EN_MC);
00176
00177 /*free the irq. */
00178 free_irq(OPL_IPMUX_IRQ,NULL);
00179 }
00180
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  doxygen 1.3.9.1

|||||

ipmux_interrupt.h

```
00001 #ifndef _IPMUX_INTERRUPT_H_
00002 #define _IPMUX_INTERRUPT_H_
00003
00020 /*-----Macro definition-----*/
00021 #define REG_INT_EN_POS_UPT      0x10204
00022 #define REG_INT_PEN_POS_UPT    0x10200
00023
00024 #define REG_INT_EN_ATB        0x11404
00025 #define REG_INT_PEN_ATB      0x11400
00026
00027 #define REG_INT_EN_IWF        0x13E00
00028 #define REG_INT_PEN_IWF      0x13E04
00029
00030 #define REG_INT_EN_BRG       0x21000
00031 #define REG_INT_PEN_BRG     0x21004
00032
00033 #define REG_INT_EN_PPE       0x18180
00034 #define REG_INT_PEN_PPE     0x18184
00035
00036 #define REG_INT_EN_UPTM     0x14404
00037 #define REG_INT_PEN_UPTM    0x14400
00038
00039 #define REG_INT_EN_DNTM     0x14F04
00040 #define REG_INT_PEN_DNTM    0x14F00
00041
00042 #define REG_INT_EN_MC       0x21404
00043 #define REG_INT_PEN_MC      0x21400
00044
00045 #define REG_INT_EN_DMA0      0x2B58
00046 #define REG_INT_PEN_DMA0     0x2B54
00047
00048
00049 #define OPL_UART0_IRQ      3
00050 #define OPL_UART1_IRQ      4
00051 #define OPL_HOSTDMA0_IRQ    5
00052 #define OPL_FEDMA1_IRQ      6
00053 #define OPL_IPMUX_IRQ       9
00054
00055
00056 /*
00057 #define IPMUX_REG_BASE   0xbf000000
00058 #define IPMUX_REG_ID2ADDR(regId) ((u32)IPMUX_REG_BASE + (u3
```

```
00059 #define IPMUX_REG_ADDR2ID(addr) ((u32)(addr) - (u32)IPMUX_
00060 */
00066 /*-----type definition-----*/
00067
00073 /*-----global variable/function declaration-----*/
00074
00080 /*-----local variable declaration and definition-----*/
00081
00088 /*-----inline function declaration and definition-----*/
00089 extern int ipmux_hw0_irqinit(void);
00090 extern void ipmux_hw0_irqexit(void);
00091
00092
00093 /*
00094 inline void ipMuxRegRead(u32 regId, u32 *val)
00095 {
00096     *val = *(volatile u32 *)IPMUX_REG_ID2ADDR(regId);
00097 }
00098
00099 inline void ipMuxRegWrite(u32 regId, u32 val)
00100 {
00101     *((volatile u32 *)IPMUX_REG_ID2ADDR(regId)) = (u32)(val);
00102 }
00103 */
00104
00107 #endif /* _IPMUX_INTERRUPT_H_ */
```

|||||

opl_cpuif.c

```
00001
00015 #include <linux/init.h>
00016 #include <linux/kernel.h>
00017 #include <linux/module.h>
00018 #include <linux/sched.h>
00019 #include <linux/wait.h>
00020 #include <asm/uaccess.h>
00021 #include <asm/io.h>
00022 #include <linux/slab.h>
00023 #include <linux/types.h>
00024 #include <linux/errno.h>
00025
00026
00027 #include <ipmux_interrupt.h>
00028 #include <opl_cpuif.h>
00032 /*-----Macro definition-----
00033 #define OPL_HOST_MAJOR 130
00034 #define OPL_HOST_NAME "opl_cpuif"
00035
00036 #define IPMUX_DMA_NUMS 4
00037 #define OPL_DMA_VAL_MAX 2048
00038
00039
00045 /*-----type definition-----
00046
00047
00048
00049
00055 /*-----global varible/function declaration-----
00056 extern void ipmux_irq_enable(unsigned int irq);
00057 extern void ipmux_irq_disable(unsigned int irq);
00058
00064 /*-----local function declaration-----
00065
00066
00067
00068
00069
00076 /*-----local variable declaration and definition---
00077 static int opl_host_open(struct inode *inode, struct file *fi
00078 static ssize_t opl_host_read(struct file *filp, char *buffer
00079 static ssize_t opl_host_write(struct file *filp, const char *
```

```
00080 static int opl_host_release(struct inode *inode, struct file *
00081 static int opl_host_ioctl(struct inode *inode,struct file *fi
00082
00083 static void host_dma0_isr(int irq,void *dev_id,struct pt_regs
00084
00085 static u32 tx_phys_addr[IPMUX_DMA_NUMS]={0};
00086 static u32 rx_phys_addr[IPMUX_DMA_NUMS]={0};
00087
00088 static char *dma0_rx_buf=NULL;
00089 static char *dma0_tx_buf=NULL;
00090
00091 static u32 order = 0;
00092 static u32 opl_host_is_open = 0;
00093 static u32 opl_host_open_count = 0;
00094 static u32 opl_host_dma0_irq_event = 0;
00095 static wait_queue_head_t opl_host_dma0_waitq;
00096 struct file_operations opl_host_fops = {
00097     .open = opl_host_open,
00098     .read = opl_host_read,
00099     .write = opl_host_write,
00100     .release = opl_host_release,
00101     .ioctl = opl_host_ioctl,
00102 };
00103
00104
00111 /*-----global variable and function exported-----
00112
00113
00121 /*-----local  function definition-----
00122
00134 static void host_dma0_isr(int irq,void *dev_id,struct pt_regs
00135 {
00136     if(test_and_set_bit(0,&opl_host_dma0_irq_event)){
00137         /* TBD: add some debug message here */
00138     }
00139     wake_up_interruptible(&opl_host_dma0_waitq);
00140 }
00141
00142
00143 static int opl_host_open(struct inode *inode,struct file *fil
00144 {
00145     if(test_and_set_bit(0,&opl_host_is_open)){
00146         printk(KERN_DEBUG "the device is opened\n");
00147     }
00148     opl_host_open_count++;
00149     MOD_INC_USE_COUNT;
00150     return 0;
00151 }
00152
00153 static int opl_host_release(struct inode *inode,struct file *
```

```
00154 {
00155     opl_host_open_count--;
00156     if(!opl_host_open_count)
00157         clear_bit(0,&opl_host_is_open);
00158
00159     MOD_DEC_USE_COUNT;
00160     return 0;
00161 }
00162
00163 static ssize_t opl_host_read(struct file *filp, char *buffer
00164 {
00165     return 0;
00166 }
00167
00168 static ssize_t opl_host_write(struct file *filp,const char *b
00169 {
00170     return 0;
00171 }
00172
00173 static int opl_host_ioctl(struct inode *inode,struct file *fi
00174 {
00175     int status = 0;
00176     if(!test_bit(0,&opl_host_is_open)){
00177         return -ENODEV;
00178     }
00179     switch(cmd){
00180     case GET_IPMUX_DMA_TX_PHYS_ADDR:
00181     {
00182         int dmas= 0;
00183         dma_request_phys_addr_t *req = (dma_request_phys_addr_t
00184         if(copy_from_user(&dmas,&req->dmas,4)){
00185         }
00186         if(tx_phys_addr[dmas]){
00187             if(copy_to_user(&req->phys_addr,&tx_phys_addr[dmas],4
00188             }
00189         }else{
00190             /* for others dma,not for dma0. */
00191         }
00192     }
00193     break;
00194     case GET_IPMUX_DMA_RX_PHYS_ADDR:
00195     {
00196         int dmas= 0;
00197         dma_request_phys_addr_t *req = (dma_request_phys_addr_t
00198         if(copy_from_user(&dmas,&req->dmas,4)){
00199         }
00200         if(rx_phys_addr[dmas]){
00201             if(copy_to_user(&req->phys_addr,&rx_phys_addr[dmas],4
00202             }
00203     }
00204 }
```

```

00224     }else{
00225         /* for others dma,not for dma0. */
00226     }
00227 }
00228 break;
00229 case SET_IPMUX_DMA_TX_BUF_DATA:
00230 {
00231     dma_request_data_t req;
00232     if(copy_from_user(&req,(char *)arg,sizeof(dma_request_d
00233     })
00234     int dmas = req.dmas;
00235     int bd = req.bd;
00236     int len = req.len;
00237     if(copy_from_user((char *)(tx_phys_addr[dmas]+bd*OPL_DM
00238     })
00239 }
00240 break;
00241 case GET_IPMUX_DMA_RX_BUF_DATA:
00242 {
00243     dma_request_data_t req;
00244     if(copy_from_user(&req,(char *)arg,sizeof(dma_request_d
00245     })
00246     int dmas = req.dmas;
00247     int bd = req.bd;
00248     int len = req.len;
00249     if(dmas<2){
00250         if(copy_to_user(((dma_request_data_t *)arg)->buf,
00251                         (char *)(rx_phys_addr[dmas]+bd*OPL_DM
00252         )
00253     }
00254 }
00255 break;
00256 case ENABLE_IPMUX_HOST_DMA0_INTERRUPT:
00257 {
00258     ipmux_irq_enable(OPL_HOSTDMA0_IRQ);
00259 }
00260 break;
00261 case DISABLE_IPMUX_HOST_DMA0_INTERRUPT:
00262 {
00263     ipmux_irq_disable(OPL_HOSTDMA0_IRQ);
00264 }
00265 break;
00266 case WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT:
00267 {
00268     wait_event_interruptible(opl_host_dma0_waitq, opl_host_d
00269     if(!test_and_clear_bit(0,&opl_host_dma0_irq_event)){
00270         printk(KERN_ALERT "clear opl_host_dma0_irq_event fail
00271         return -1;
00272     }
00273 }

```

```

00274     break;
00275 default:
00276     break;
00277 }
00278 return status;
00279 }
00280
00289 static int __init opl_host_dma_init(void)
00290 {
00291     int ret = 0;
00292     int count = 0;
00293     int size = 2*64*OPL_DMA_VAL_MAX;
00294
00295     ret = register_chrdev(OPL_HOST_MAJOR, OPL_HOST_NAME, &opl_hos
00296     if(ret < 0){
00297         /* add debug Message TBD */
00298         return -1;
00299     }
00300
00301     for (count = PAGE_SIZE, order = 0; count < size; order++, c
00302         ;
00303     dma0_rx_buf =(char*) __get_free_pages(GFP_KERNEL| GFP_DMA,
00304     if(!dma0_rx_buf){
00305         goto fail1;
00306     }
00307     memset(dma0_rx_buf,0,size);
00308     dma0_tx_buf = &dma0_rx_buf[64*OPL_DMA_VAL_MAX];
00309     rx_phys_addr[0] = (u32)dma0_rx_buf|0xa0000000; /* for mips
00310     tx_phys_addr[0] = (u32)dma0_tx_buf|0xa0000000;
00311
00312     init_waitqueue_head(&op1_host_dma0_waitq);
00313     ret = request_irq(OPL_HOSTDMA0_IRQ,host_dma0_isr,0,"IPMux d
00314     if(ret){
00315         /* add error message */
00316         goto fail2;
00317     }
00318     return 0;
00319 fail1:
00320     ret = unregister_chrdev(OPL_HOST_MAJOR,OPL_HOST_NAME);
00321 fail2:
00322     free_pages((u32)dma0_rx_buf,order);
00323     return -1;
00324 }
00325
00333 static void __exit opl_host_dma_exit(void)
00334 {
00335     int ret = 0;
00336
00337     free_irq(OPL_HOSTDMA0_IRQ,NULL);

```

```
00338     free_pages((u32)dma0_rx_buf,order);
00339     ret = unregister_chrdev(OPL_HOST_MAJOR,OPL_HOST_NAME);
00340     if(ret<0){
00341         /* add debug Message TBD */
00342     }
00343 }
00344
00345 module_init(opl_host_dma_init);
00346 module_exit(opl_host_dma_exit);
00347 MODULE_LICENSE("GPL2");
00348 MODULE_AUTHOR("opulan Inc");
00349 MODULE_DESCRIPTION("opulan IPMUX-e switch chip driver module"
00350
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_cpuif.h

```
00001
00018 /*-----Macro definition-----
00019 #define GET_IPMUX_DMA_TX_PHYS_ADDR 1000
00020 #define GET_IPMUX_DMA_RX_PHYS_ADDR 2000
00021
00022 #define GET_IPMUX_DMA_RX_BUF_DATA      3000
00023 #define SET_IPMUX_DMA_TX_BUF_DATA      4000
00024
00025 #define ENABLE_IPMUX_HOST_DMA0_INTERRUPT 5000
00026 #define DISABLE_IPMUX_HOST_DMA0_INTERRUPT 6000
00027 #define WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT 7000
00028
00029
00030
00031
00032
00038 /*-----type definition-----
00039
00040 typedef struct dma_request_phys_addr_s{
00041     int dmas;
00042     unsigned long phys_addr;
00043 }dma_request_phys_addr_t;
00044
00045 typedef struct dma_request_data_s{
00046     int dmas;
00047     int bd;
00048     int len;
00049     char buf[2048];
00050 }dma_request_data_t;
00051
00052
00053
00059 /*-----global variable/function declaration-----
00060
00061
00062
00063
00064
00070 /*-----local function declaration-----
00071
00072
00073
```

```
00074
00075
00082 /*-----local variable declaration and definition---
00083
00084
00085
00086
00087
00094 /*-----global variable and function exported-----
00095
00096
00097
00098
00099
00107 /*-----local function definition-----
00108
00109
00110
00111
00112
00113
```

|||||

opl_cpuif_test.c

```
00001
00014 #include <stdlib.h>
00015 #include <unistd.h>
00016
00017 #include <opl_typedef.h>
00021 /*-----Macro definition-----
00022 #define OPL_CPUIF_DEVICE "/dev/opl_cpuif"
00023
00024
00025
00026
00032 /*-----type definition-----
00033
00034
00035
00036
00037
00043 /*-----global varible/function declaration-----
00044
00050 /*-----local function declaration-----
00051
00052
00053
00054
00055
00062 /*-----local variable declaration and definition---
00063
00064
00071 /*-----global variable and function exported-----
00072 int opl_cpuif_test_init(int *initialized);
00073 void opl_cpuif_test_main(int argc,char **argv);
00074 int opl_cpuif_test_exit(int *initialized);
00075
00076
00077
00085 /*-----local function definition-----
00086
00087
00088
00089 int opl_cpuif_test_init(int *initialized)
00090 {
00091     u32 value = 0;
```

```
00092     *initialized = 1;
00093     return 0;
00094 }
00095
00096 int opl_cpuif_test_exit(int *initialized)
00097 {
00098     int ret = 0;
00099     if(*initialized){
00100         *initialized = 0;
00101     }
00102     return ret;
00103 }
00104
00105 void opl_cpuif_test_main(int argc,char **argv)
00106 {
00107 }
00108
00109 int main(int argc,char **argv)
00110 {
00111     int initialized = 0;
00112     opl_cpuif_test_init(&initialized);
00113     if(initialized){
00114         opl_cpuif_test_main(argc,argv);
00115     }
00116     opl_cpuif_test_exit(&initialized);
00117 }
00118
00119
```

|||||

opl_hw_ops.h

```
00001 #ifndef _OPL_HW_H_
00002 #define _OPL_HW_H_
00003
00020 /*-----Macro definition-----*/
00021 #define IPMUX_REG_BASE           0xBF000000      /* de
00022
00023
00024
00025
00031 /*-----type definition-----*/
00032
00033
00034
00035
00036
00042 /*-----global varible/function declaration-----*/
00043
00044
00045
00046
00047
00053 /*-----local  function declaration-----*/
00054
00055
00056
00057
00058
00065 /*-----local  variable declaration and definition---*/
00066
00067
00068
00069
00070
00077 /*-----global variable and function exported---*/
00078
00079
00080
00081
00082
00090 /*-----local  function definition-----*/
00091
00092 #define IPMUX_REG_ID2ADDR(regId)      ((u32)IPMUX_REG_BASE
```

```
00093 #define IPMUX_REG_ADDR2ID(addr) ((u32)(addr) - (u32)I
00094
00095 static inline int ipMuxRegRead(u32 regID, volatile u32 *pval)
00096 {
00097     if (regID % 4 != 0)
00098         return -1;
00099
00100     *(u32*)(pval) = *(volatile u32 *)IPMUX_REG_ID2ADDR(regID);
00101
00102     return 0;
00103 }
00104
00105 static inline int ipMuxRegWrite(u32 regID, u32 val)
00106 {
00107     if (regID % 4 != 0)
00108         return -1;
00109
00110     *((volatile u32 *)IPMUX_REG_ID2ADDR(regID)) = (u32)(val);
00111
00112     return 0;
00113 }
00114
00115
00116
00117
00120 #endif/* _OPL_HW_H_ */
```

|||||

opl_minte.c

```
00001
00017 #include <linux/init.h>
00018 #include <linux/kernel.h>
00019 #include <linux/module.h>
00020 #include <linux/sched.h>
00021 #include <linux/wait.h>
00022 #include <asm/uaccess.h>
00023 #include <asm/io.h>
00024 #include <linux/slab.h>
00025 #include <linux/types.h>
00026 #include <linux/errno.h>
00027
00028 #include <opl_minte.h>
00029 #include <ipmux_interrupt.h>
00030
00034 /*-----Macro definition-----*/
00035 #define OPL_MINTE_MAJOR    120
00036 #define OPL_MINTE_DEVICE   "opl_minte"
00037
00038
00044 /*-----type definition-----*/
00045
00051 /*-----global variable/function declaration-----*/
00052 extern u32 get_irq_pending(void);
00053 extern void clear_irq_pending(void);
00054 extern void ipmux_irq_enable(unsigned int irq);
00055 extern void ipmux_irq_disable(unsigned int irq);
00061 u32 g_opl_chip_irq_event = 0;
00062 wait_queue_head_t g_opl_chip_waitq;
00063
00069 /*-----local function declaration----- */
00070 static int opl_minte_open(struct inode *inode, struct file *f
00071 static ssize_t opl_minte_read(struct file *filp, char *buffe
00072 static ssize_t opl_minte_write(struct file *filp, const char
00073 static int opl_minte_release(struct inode *inode, struct file
00074
00080 /*-----local variable declaration and definition-----*/
00081 static u32 opl_minte_open_count = 0;
00082 static u32 opl_minte_is_open = 0;
00083 struct file_operations opl_minte_fops = {
00084     .read = opl_minte_read,
00085     .write = opl_minte_write,
```

```

00086     .open = opl_minte_open,
00087     .release = opl_minte_release,
00088 };
00089
00095 /*-----local function declaration and definition-----*/
00096
00109 static ssize_t opl_minte_read(struct file *filp, char *buffe
00110 {
00111     u32 pending = 0;
00112     if(test_bit(0,&opl_minte_is_open)){
00113         /* TBD: add debug message */
00114         wait_event_interruptible(g_opl_chip_waitq,g_opl_chip_irq_
00115         if(!test_and_clear_bit(0,&g_opl_chip_irq_event)){
00116             /* TBD: add debug message ,if return 0, indicate more pro
00117             printk("clear g_opl_chip_irq_event event failed\n");
00118             return -ENODEV;
00119         }
00120         pending = get_irq_pending();
00121         /* should add lock later */
00122         clear_irq_pending();
00123         if(!(copy_to_user((void *)buffer,(void *)&pending,4))){
00124             /* TBD: add debug message. */
00125             return 4;
00126         }
00127     }
00128     return -ENODEV;
00129 }
00146 static ssize_t opl_minte_write(struct file *filp, const char
00147 {
00148     int irq_enable = 0;
00149     if(test_bit(0,&opl_minte_is_open)){
00150         if(!(copy_from_user(&irq_enable,buffer,4))){
00151             if(irq_enable){
00152                 ipmux_irq_enable(OPL_IPMUX_IRQ);
00153             }else{
00154                 ipmux_irq_disable(OPL_IPMUX_IRQ);
00155             }
00156             return 0;
00157         }
00158     }
00159     return -ENODEV;
00160 }
00161
00162
00163 static int opl_minte_open(struct inode *inode, struct file *f
00164 {
00165     if (test_and_set_bit(0, &opl_minte_is_open))
00166         printk(KERN_DEBUG "the/dev/opl_minte is opened\n");
00167     opl_minte_open_count++;
00168     MOD_INC_USE_COUNT;

```

```
00169     return 0;
00170 }
00171
00172 static int opl_minte_release(struct inode *inode, struct file
00173 {
00174     opl_minte_open_count--;
00175     if(!opl_minte_open_count)
00176         clear_bit(0, &opl_minte_is_open);
00177     MOD_DEC_USE_COUNT;
00178     return 0;
00179 }
00180 static int __init opl_minte_init(void)
00181 {
00182     int ret;
00183     printk("enter minte module\n");
00184     ret = register_chrdev(OPL_MINTE_MAJOR, OPL_MINTE_DEVICE, &opl
00185     if(ret<0){
00186         /* TBD add debug message */
00187         printk("error1\n");
00188         goto fail1;
00189     }
00190     ret = opl_minte_hw0_irqinit();
00191     if(ret){
00192         /*TBD add debug message */
00193         goto fail2;
00194     }
00195     init_waitqueue_head(&g_opl_chip_waitq);
00196     return 0;
00197 fail2:
00198     unregister_chrdev(OPL_MINTE_MAJOR, OPL_MINTE_DEVICE);
00199 fail1:
00200     return ret;
00201 }
00202 static void __exit opl_minte_exit(void)
00203 {
00204     int ret;
00205     opl_minte_hw0_irqexit();
00206     ret = unregister_chrdev(OPL_MINTE_MAJOR, OPL_MINTE_DEVICE);
00207     if(ret < 0){
00208         /*TBD:add debug message */
00209     }
00210     return;
00211 }
00212 module_init(opl_minte_init);
00213 module_exit(opl_minte_exit);
00214 MODULE_LICENSE("GPL2");
00215 MODULE_AUTHOR("stephanxu <hxu@opulan.com>");
00216 MODULE_DESCRIPTION("control interface for opulan ipmux-e switch")
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  *1.3.9.1*

|||||

opl_minte.h

```
00001 #ifndef _OPL_MINTE_H_
00002 #define _OPL_MINTE_H_
00003
00020 /*-----Macro definition-----*/
00021 extern int ipmux_hw0_irqinit(void);
00022 extern void ipmux_hw0_irqexit(void);
00023 #define opl_minte_hw0_irqexit ipmux_hw0_irqexit
00024 #define opl_minte_hw0_irqinit ipmux_hw0_irqinit
00025
00026
00027
00028
00034 /*-----type definition-----*/
00035
00036
00037
00038
00039
00045 /*-----global variable/function declaration-----*/
00046
00047
00048
00049
00050
00056 /*-----local function declaration-----*/
00057
00058
00059
00060
00061
00068 /*-----local variable declaration and definition---*/
00069
00070
00071
00072
00073
00080 /*-----global variable and function exported----*/
00081
00082
00083
00084
00085
```

```
00093 /*-----local function definition-----*/
00094
00095
00096
00097
00098
00099
00102 #endif /* _OPL_MINTE_H */
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  doxygen 1.3.9.1

|||||

opl_minte_test.c

```
00001
00014 #include <stdlib.h>
00015 #include <fcntl.h>
00016 #include <unistd.h>
00017 #include "../include/opl_typedef.h"
00021 /*-----Macro definition-----
00022 #define OPL_MINTE_DEVICE "/dev/opl_minte"
00023
00024
00030 /*-----type definition-----
00031
00032
00033
00034
00035
00041 /*-----global variable/function declaration-----
00042
00043
00044
00050 /*-----local function declaration-----
00051
00053 int opl_minte_fd;
00054
00055
00056
00057
00064 /*-----local variable declaration and definition---
00065
00066 int opl_minte_init(int *initialized);
00067 void opl_minte_main(int argc,char **argv);
00068 int opl_minte_exit(int *initialized);
00069
00076 /*-----global variable and function exported-----
00077
00078
00079
00080
00081
00089 /*-----local function definition-----
00090
00101 int opl_minte_init(int *initialized)
00102 {
```

```

00103     opl_minte_fd = open(OPL_MINTE_DEVICE,O_RDWR|O_SYNC);
00104     if(opl_minte_fd<0){
00105         printf("open the %s device failed\n",OPL_MINTE_DEVICE);
00106         return -1;
00107     }
00108     *initialized = 1;
00109     return 0;
00110 }
00111
00120 int opl_minte_exit(int *initialized)
00121 {
00122     int ret = 0;
00123     if(*initialized){
00124         ret = close(opl_minte_fd);
00125         *initialized = 0;
00126     }
00127     return ret;
00128 }
00129
00139 void opl_minte_main(int argc,char **argv)
00140 {
00141     int len = 0;
00142     u32 irq_pending = 0;
00143     len = read(opl_minte_fd,&irq_pending,4);
00144     if(len == 4){
00145         printf("read the irq pending is correct. pending = 0x%x",
00146             return;
00147     }else{
00148         printf("read the irq pending error\n");
00149     }
00150 }
00162 int main(int argc,char **argv)
00163 {
00164     int initialized = 0;
00165     opl_minte_init(&initialized);
00166     if(initialized){
00167         opl_minte_main(argc,argv);
00168     }
00169     opl_minte_exit(&initialized);
00170     return 0;
00171 }

```

|||||

opl_reg_mmap.c

```
00001
00015 #include <linux/init.h>
00016 #include <linux/kernel.h>
00017 #include <linux/module.h>
00018 #include <linux/sched.h>
00019 #include <linux/wait.h>
00020 #include <asm/uaccess.h>
00021 #include <asm/io.h>
00022 #include <linux/slab.h>
00023
00024 #include <opl_hw_ops.h>
00025 #include <opl_reg_mmap.h>
00026
00030 /*-----Macro definition-----*/
00031 #define OPL_REG_MAJOR      110
00032 #define OPL_REG_DEVICE    "opl_reg"
00033
00034
00040 /*-----type definition-----*/
00041
00047 /*-----global variable/function declaration-----*/
00048
00060 /*-----local function declaration----- */
00061 static int opl_reg_open(struct inode *inode, struct file *fil
00062 static ssize_t opl_reg_read(struct file *filp, char *buffer,
00063 static ssize_t opl_reg_write(struct file *filp, const char *b
00064 static int opl_reg_mmap(struct file * file, struct vm_area_st
00065 static int opl_reg_release(struct inode *inode, struct file *
00066
00072 /*-----local variable declaration and definition----*/
00073 static u32 opl_reg_open_count = 0;
00074 static u32 opl_reg_is_open = 0;
00075 struct file_operations opl_reg_fops = {
00076     .read = opl_reg_read,
00077     .write = opl_reg_write,
00078     .mmap = opl_reg_mmap,
00079     .open = opl_reg_open,
00080     .release = opl_reg_release,
00081 };
00082
00088 /*-----local function declaration and definition----*/
00089
```

```
00102 static ssize_t opl_reg_read(struct file *filp, char *buffer,
00103 {
00104     return 0;
00105 }
00106
00107 static ssize_t opl_reg_write(struct file *filp, const char *b
00108 {
00109     return 0;
00110 }
00111
00112 static int opl_reg_mmap(struct file * file, struct vm_area_st
00113 {
00114     unsigned long offset = vma->vm_pgoff << PAGE_SHIFT;
00115     /*
00116     * Accessing memory above the top the kernel knows about or
00117     * through a file pointer that was marked O_SYNC will be
00118     * done non-cached.
00119     */
00120     offset += 0x1f000000;
00121     if ((offset>__pa(high_memory)) || (file->f_flags & O_SYNC))
00122         vma->vm_page_prot = pgprot_noncached(vma->vm_page_prot);
00123     }
00124
00125     /* Don't try to swap out physical pages.. */
00126     vma->vm_flags |= VM_RESERVED;
00127
00128     /* Don't dump addresses that are not real memory to a core
00129     if (offset >= __pa(high_memory) || (file->f_flags & O_SYNC))
00130         vma->vm_flags |= VM_IO;
00131     if (remap_page_range(vma->vm_start, offset, vma->vm_end-vma
00132                         vma->vm_page_prot)){
00133         return -EAGAIN;
00134     }
00135     return 0;
00136 }
00137
00138 static int opl_reg_open(struct inode *inode, struct file *fil
00139 {
00140     if (test_and_set_bit(0, &opl_reg_is_open))
00141         printk(KERN_DEBUG "the /dev/opl_reg is already opened\n");
00142     opl_reg_open_count++;
00143     MOD_INC_USE_COUNT;
00144     return 0;
00145 }
00146
00147 static int opl_reg_release(struct inode *inode, struct file *
00148 {
00149     opl_reg_open_count++;
00150     if(!opl_reg_open_count)
00151         clear_bit(0, &opl_reg_is_open);
00152 }
```

```
00164     MOD_DEC_USE_COUNT;
00165     return 0;
00166 }
00176 static int __init opl_reg_init(void)
00177 {
00178     int ret;
00179     ret = register_chrdev(OPL_REG_MAJOR, OPL_REG_DEVICE, &opl_reg);
00180     if(ret<0){
00181         /* TBD add debug message */
00182     }
00183     return ret;
00184 }
00193 static void __exit opl_reg_exit(void)
00194 {
00195     int ret;
00196     ret = unregister_chrdev(OPL_REG_MAJOR, OPL_REG_DEVICE);
00197     if(ret < 0){
00198         /*TBD: add debug message */
00199     }
00200     return;
00201 }
00202
00203 module_init(opl_reg_init);
00204 module_exit(opl_reg_exit);
00205 MODULE_LICENSE("GPL2");
00206 MODULE_AUTHOR("stephanxu <hxu@opulan.com>");
00207 MODULE_DESCRIPTION("dedicated the module for mmap the opulan
```

|||||

opl_reg_mmap.h

```
00001 #ifndef _OPL_REG_MMAP_H_
00002 #define _OPL_REG_MMAP_H_
00003
00020 /*-----Macro definition-----*/
00021
00022
00023
00024
00025
00031 /*-----type definition-----*/
00032
00033
00034
00035
00036
00042 /*-----global varible/function declaration-----*/
00043
00044
00045
00046
00047
00053 /*-----local  function declaration-----*/
00054
00055
00056
00057
00058
00065 /*-----local  variable declaration and definition---*/
00066
00067
00068
00069
00070
00077 /*-----global variable and function exported---*/
00078
00079
00080
00081
00082
00090 /*-----local  function definition-----*/
00091
00092
```

```
00093
00094
00095
00096
00099 #endif /* _OPL_REG_MMAP_H_ */
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1

|||||

opl_regmap_test.c

```
00001
00014 #include <stdlib.h>
00015 #include <sys/mman.h>
00016 #include <fcntl.h>
00017
00018 #include "../include/opl_typedef.h"
00022 /*-----Macro definition-----
00023 #define OPL_REG_DEVICE "/dev/opl_reg"
00024 #define OPL_REG_SIZE 0x300000
00025
00026
00027
00028
00034 /*-----type definition-----
00035
00036
00037
00038
00039
00045 /*-----global variable/function declaration-----
00046
00047
00048
00049
00050
00056 /*-----local function declaration-----
00057
00058
00059
00060
00061
00068 /*-----local variable declaration and definition---
00069 u32 reg_read(char *reg_base,int offset);
00070 void reg_write(char *reg_base, int offset, u32 value);
00071 void opl_reg_main(int argc,char ** argv);
00072 int opl_reg_init(int *opl_reg_initialized);
00073 int opl_reg_exit(int *opl_reg_initialized);
00074
00075
00076
00083 /*-----global variable and function exported-----
00085 char *opl_reg_base = NULL;
```

```
00086 char *OPL_REG_USAGE = "opl_reg_test -r offset or\n\
00087          opl_reg_test --read offset or\n\
00088          opl_reg_test -w offset value or\n\
00089          opl_reg_test --write offset value \n\
00090          Note: the offset and value can be deci
00091
00092
00094 int opl_reg_fd;
00095
00103 /*-----local function definition-----*/
00104
00115 u32 reg_read(char *reg_base,int offset)
00116 {
00117     u32 value = 0;
00118     if(offset%4){
00119         printf("the alignment is not 4 bytes \n");
00120         return -1;
00121     }
00122     value = *(volatile u32 *)(reg_base + offset);
00123     return value;
00124 }
00134 void reg_write(char *reg_base, int offset, u32 value)
00135 {
00136     if(offset%4){
00137         printf("the alignment is not 4 bytes \n");
00138         return ;
00139     }
00140     *(volatile u32 *)(reg_base + offset) = value;
00141 }
00151 void opl_reg_main(int argc,char ** argv)
00152 {
00153     int i = 0;
00154     int offset = 0;
00155     u32 value = 0;
00156
00157     if(argc < 3){
00158         printf("the number of argument is not correct\n");
00159         printf(OPL_REG_USAGE);
00160         return;
00161     }
00162
00163     if((argv[2][0] == '0') && ((argv[2][1] == 'x')||(argv[2][1]
00164     /* TBD:should be check the ranged */
00165     sscanf(argv[2],"%x",&offset);
00166 }else{
00167     sscanf(argv[2],"%d",&offset);
00168 }
00169
00170     if(!strcmp(argv[1],"--read")||!strcmp(argv[1], "-r")){
00171         if(argc != 3){
```

```

00172     printf("the number of argument is not correct\n");
00173     printf(OPL_REG_USAGE);
00174     return;
00175 }
00176     printf("the %x register's value is 0x%x\n", offset, reg_rea
00177
00178 }else if(!strcmp(argv[1],"--write")||!strcmp(argv[1], "-w"))
00179 {
00180     if(argc != 4){
00181         printf("the number of argument is not correct\n");
00182         printf(OPL_REG_USAGE);
00183         return;
00184     }
00185     if(argv[3][0] == '0' && ((argv[3][1] == 'x')||(argv[3][1]
00186     /* TBD:should be check the ranged */
00187     sscanf(argv[3], "%x", &value);
00188     }else{
00189         sscanf(argv[3], "%d", &value);
00190     }
00191     reg_write(opl_reg_base, offset, value);
00192     printf("write the %x register, write_value = 0x%x, check_va
00193 }
00194
00203 int opl_reg_init(int *opl_reg_initialized)
00204 {
00205     opl_reg_fd = open(OPL_REG_DEVICE, O_RDWR);
00206     if(opl_reg_fd<0){
00207         printf("open device file failed\n", OPL_REG_DEVICE);
00208         return -1;
00209     }
00210     opl_reg_base = (char *)mmap(0, OPL_REG_SIZE, PROT_READ|PROT_W
00211     if(!opl_reg_base){
00212         printf("mmap the opl register failed\n");
00213         close(opl_reg_fd);
00214         return -1;
00215     }
00216     *opl_reg_initialized = 1;
00217     return 0;
00218 }
00227 int opl_reg_exit(int *opl_reg_initialized)
00228 {
00229     int ret = 0;
00230     if(*opl_reg_initialized == 1){
00231         *opl_reg_initialized = 0;
00232         munmap(opl_reg_base, OPL_REG_SIZE);
00233         ret = close(opl_reg_fd);
00234     }
00235     return ret;
00236 }
```

```
00247 int main(int argc, char **argv)
00248 {
00249     int init = 0;
00250     opl_reg_init(&init);
00251     if(init){
00252         opl_reg_main(argc,argv);
00253     }
00254     opl_reg_exit(&init);
00255     return 0;
00256 }
00257
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  doxygen 1.3.9.1

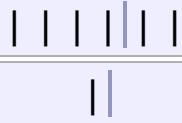
|||||

opl_typedef.h

```
00001 #ifndef _OPL_TYPEDEF_H_
00002 #define _OPL_TYPEDEF_H_
00003
00020 /*-----Macro definition-----*/
00021
00022
00023
00024
00025
00031 /*-----type definition-----*/
00032 #ifndef __KERNEL__
00033 typedef unsigned int u32;
00034 typedef unsigned short u16;
00035 typedef unsigned char u8;
00036 typedef long s32;
00037 typedef short s16;
00038 typedef char s8;
00039 typedef unsigned char bool;
00040 #endif /* __KERNEL__ */
00041
00046 /*-----global variable/function declaration-----*/
00047
00048
00049
00050
00051
00057 /*-----local function declaration-----*/
00058
00059
00060
00061
00062
00069 /*-----local variable declaration and definition---*/
00070
00071
00072
00073
00074
00081 /*-----global variable and function exported----*/
00082
00083
00084
```

```
00085
00086
00094 /*-----local function definition-----
00095
00096
00097
00098
00099
00100
00103 #endif /* _OPL_TYPEDEF_H_ */
```

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1



- bd : **dma_request_data_s**
- buf : **dma_request_data_s**
- dmas : **dma_request_data_s**, **dma_request_phys_addr_s**
- len : **dma_request_data_s**
- phys_addr : **dma_request_phys_addr_s**

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by doxygen 1.3.9.1



- C -

- clear_irq_pending() : **opl_minte.c**

- g -

- get_irq_pending() : **opl_minte.c**
- get_mask() : **ipmux_interrupt.c**

- h -

- host_dma0_isr() : **opl_cpuif.c**

- i -

- ipmux_interrupt_handler() : **ipmux_interrupt.c**
- ipmux_hw0_irqexit() : **opl_minte.h, ipmux_interrupt.c**
- ipmux_hw0_irqinit() : **opl_minte.h, ipmux_interrupt.c**
- ipmux_interrupt_handler() : **ipmux_interrupt.c**
- ipmux_irq_disable() : **opl_minte.c, opl_cpuif.c**
- ipmux_irq_enable() : **opl_minte.c, opl_cpuif.c**
- ipMuxRegRead() : **opl_hw_ops.h**
- ipMuxRegWrite() : **opl_hw_ops.h**

- m -

- main() : **opl_regmmap_test.c, opl_minte_test.c, opl_cpuif_test.c**
- MODULE_AUTHOR() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- MODULE_DESCRIPTION() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- module_exit() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**
- module_init() : **opl_reg mmap.c, opl_minte.c, opl_cpuif.c**

- MODULE_LICENSE() : **opl_reg_mmap.c, opl_minte.c, opl_cpuif.c**

- O -

- opl_cpuif_test_exit() : **opl_cpuif_test.c**
- opl_cpuif_test_init() : **opl_cpuif_test.c**
- opl_cpuif_test_main() : **opl_cpuif_test.c**
- opl_host_dma_exit() : **opl_cpuif.c**
- opl_host_dma_init() : **opl_cpuif.c**
- opl_host_ioctl() : **opl_cpuif.c**
- opl_host_open() : **opl_cpuif.c**
- opl_host_read() : **opl_cpuif.c**
- opl_host_release() : **opl_cpuif.c**
- opl_host_write() : **opl_cpuif.c**
- opl_minte_exit() : **opl_minte_test.c, opl_minte.c**
- opl_minte_init() : **opl_minte_test.c, opl_minte.c**
- opl_minte_main() : **opl_minte_test.c**
- opl_minte_open() : **opl_minte.c**
- opl_minte_read() : **opl_minte.c**
- opl_minte_release() : **opl_minte.c**
- opl_minte_write() : **opl_minte.c**
- opl_reg_exit() : **opl_regmmap_test.c, opl_reg_mmap.c**
- opl_reg_init() : **opl_regmmap_test.c, opl_reg_mmap.c**
- opl_reg_main() : **opl_regmmap_test.c**
- opl_reg_mmap() : **opl_reg_mmap.c**
- opl_reg_open() : **opl_reg_mmap.c**
- opl_reg_read() : **opl_reg_mmap.c**
- opl_reg_release() : **opl_reg_mmap.c**
- opl_reg_write() : **opl_reg_mmap.c**

- r -

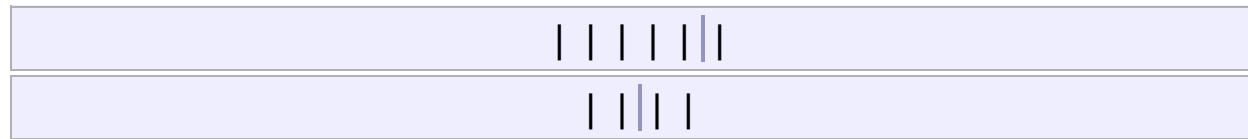
- reg_read() : **opl_regmmap_test.c**
- reg_write() : **opl_regmmap_test.c**

- t -

- turn_off_level2_intr() : **ipmux_interrupt.c**

- turn_on_level2_intr() : **ipmux_interrupt.c**
-

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  1.3.9.1



- dma0_rx_buf : [opl_cpuif.c](#)
- dma0_tx_buf : [opl_cpuif.c](#)
- g_opl_chip_irq_event : [opl_minte.c](#), [ipmux_interrupt.c](#)
- g_opl_chip_waitq : [opl_minte.c](#), [ipmux_interrupt.c](#)
- intr_status : [ipmux_interrupt.c](#)
- opl_host_dma0_irq_event : [opl_cpuif.c](#)
- opl_host_dma0_waitq : [opl_cpuif.c](#)
- opl_host_fops : [opl_cpuif.c](#)
- opl_host_is_open : [opl_cpuif.c](#)
- opl_host_open_count : [opl_cpuif.c](#)
- opl_minte_fd : [opl_minte_test.c](#)
- opl_minte_fops : [opl_minte.c](#)
- opl_minte_is_open : [opl_minte.c](#)
- opl_minte_open_count : [opl_minte.c](#)
- opl_reg_base : [opl_RegMmap_test.c](#)
- opl_reg_fd : [opl_RegMmap_test.c](#)
- opl_reg_fops : [opl_reg_mmap.c](#)
- opl_reg_is_open : [opl_reg_mmap.c](#)
- opl_reg_open_count : [opl_reg_mmap.c](#)
- OPL_REG_USAGE : [opl_RegMmap_test.c](#)
- order : [opl_cpuif.c](#)
- rx_phys_addr : [opl_cpuif.c](#)
- tx_phys_addr : [opl_cpuif.c](#)

||| |||

||| ||

- bool : [opl_typedef.h](#)
- dma_request_data_t : [opl_cpuif.h](#)
- dma_request_phys_addr_t : [opl_cpuif.h](#)
- s16 : [opl_typedef.h](#)
- s32 : [opl_typedef.h](#)
- s8 : [opl_typedef.h](#)
- u16 : [opl_typedef.h](#)
- u32 : [opl_typedef.h](#)
- u8 : [opl_typedef.h](#)



- d -

- DISABLE_IPMUX_HOST_DMA0_INTERRUPT : [opl_cpuif.h](#)

- e -

- ENABLE_IPMUX_HOST_DMA0_INTERRUPT : [opl_cpuif.h](#)

- g -

- GET_IPMUX_DMA_RX_BUF_DATA : [opl_cpuif.h](#)
- GET_IPMUX_DMA_RX_PHYS_ADDR : [opl_cpuif.h](#)
- GET_IPMUX_DMA_TX_PHYS_ADDR : [opl_cpuif.h](#)

- i -

- IPMUX_DMA_NUMS : [opl_cpuif.c](#)
- IPMUX_REG_ADDR2ID : [opl_hw_ops.h](#)
- IPMUX_REG_BASE : [opl_hw_ops.h](#)
- IPMUX_REG_ID2ADDR : [opl_hw_ops.h](#)

- o -

- OPL_CPUIF_DEVICE : [opl_cpuif_test.c](#)
- OPL_DMA_VAL_MAX : [opl_cpuif.c](#)
- OPL_FEDMA1_IRQ : [ipmux_interrupt.h](#)
- OPL_HOST_MAJOR : [opl_cpuif.c](#)
- OPL_HOST_NAME : [opl_cpuif.c](#)
- OPL_HOSTDMA0_IRQ : [ipmux_interrupt.h](#)
- OPL_IPMUX_IRQ : [ipmux_interrupt.h](#)
- OPL_MINTE_DEVICE : [opl_minte_test.c](#), [opl_minte.c](#)
- opl_minte_hw0_irqexit : [opl_minte.h](#)

- opl_minte_hw0_irqinit : [opl_minte.h](#)
- OPL_MINTE_MAJOR : [opl_minte.c](#)
- OPL_REG_DEVICE : [opl_reg mmap test.c](#), [opl_reg mmap.c](#)
- OPL_REG_MAJOR : [opl_reg mmap.c](#)
- OPL_REG_SIZE : [opl_reg mmap test.c](#)
- OPL_UART0_IRQ : [ipmux_interrupt.h](#)
- OPL_UART1_IRQ : [ipmux_interrupt.h](#)

- r -

- REG_INT_EN_ATB : [ipmux_interrupt.h](#)
- REG_INT_EN_BRG : [ipmux_interrupt.h](#)
- REG_INT_EN_DMA0 : [ipmux_interrupt.h](#)
- REG_INT_EN_DNTM : [ipmux_interrupt.h](#)
- REG_INT_EN_IWF : [ipmux_interrupt.h](#)
- REG_INT_EN_MC : [ipmux_interrupt.h](#)
- REG_INT_EN_POS_UPT : [ipmux_interrupt.h](#)
- REG_INT_EN_PPE : [ipmux_interrupt.h](#)
- REG_INT_EN_UPTM : [ipmux_interrupt.h](#)
- REG_INT_PEN_ATB : [ipmux_interrupt.h](#)
- REG_INT_PEN_BRG : [ipmux_interrupt.h](#)
- REG_INT_PEN_DMA0 : [ipmux_interrupt.h](#)
- REG_INT_PEN_DNTM : [ipmux_interrupt.h](#)
- REG_INT_PEN_IWF : [ipmux_interrupt.h](#)
- REG_INT_PEN_MC : [ipmux_interrupt.h](#)
- REG_INT_PEN_POS_UPT : [ipmux_interrupt.h](#)
- REG_INT_PEN_PPE : [ipmux_interrupt.h](#)
- REG_INT_PEN_UPTM : [ipmux_interrupt.h](#)

- s -

- SET_IPMUX_DMA_TX_BUF_DATA : [opl_cpuif.h](#)

- w -

- WAIT_FOR_IPMUX_HOST_DMA0_INTERRUPT : [opl_cpuif.h](#)
-

Generated at Mon Jul 30 16:43:48 2007 for IPMUX1IPMUX2IPMUX3 by  **1.3.9.1**