Sample Codes Manual

UVCS Adapted for Linux Common Engine Library

Sample Codes Manual

Renesas Electronics
www.renesas.com

Rev.0.91 2014.05

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

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

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

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

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

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the user interface of this software.
Target reader is the user which designs the applied system using this software.
For using this manual, it is required following knowledge,

- Knowledge of Moving picture.
- Knowledge of RTOS (Real time operating system).
- Knowledge of each CODEC to use.

2. About using this software

When you use this software, you need to enter into the software license agreement with us.

3. Related Manuals

As a related document, the next manual and document are prepared for using this software.
Please contact Renesas Electronics sales office if necessary.

- UVCS Common Engine Library User's Manual

4. About Revision History

Revision history is only the main point that we corrected or added it for the former edition.
It is not the thing which recorded all the revision contents.
Please refer to this manual about a detail.

5. List of Abbreviations

Abbreviation	Long Name
UVCS-CMN	UVCS Common Engine Library

6. List of Acronyms

Acronyms	Long Name
UVCS	Unified Video CODEC Server

Table of contents

1. Sample Codes

1.1 External Function Prototypes

1.1.1 uvcs_cmn_interrupt

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static ulong reg_vlc; /* mapped register address */
static struct tasklet_struct tl_vlc;

static irqreturn_t uvcs_isr_0(int irq, void *dev)
{
	if (dev == &reg_vlc[0]) {
		iowrite32(0uL, (void *)(reg_vlc + UVCS_VCPREG_IRQENB));
		tasklet_schedule(&tl_vlc);
		return IRQ_HANDLED;
	}
	return IRQ_NONE;
}

static void uvcs_tasklet_0(ulong value)
{
	uvcs_cmn_interrupt(uvcs_info, reg_vlc, 0);
}
```

Please refer to uvcs_tasklet_0 function in uvcs_lkm_uf_io.c.

1.1.2 uvcs_cmn_initialize

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_U32 * uvcs_lib_work_mem ;
static UVCS_U32 uvcs_lib_work_req_size ;
static UVCS_U32 uvcs_hdl_work_req_size ;

int sample_initialize( void )
{
	UVCS_CMN_INIT_PARAM_T uvcs_init_param ;
	UVCS_RESULT uvcs_ret ;

	/* get workarea size for uvcs-cmn */
	(void)uvcs_cmn_get_work_size( &uvcs_lib_work_req_size, &uvcs_hdl_work_req_size ) ;

	/* allocate workarea */
	uvcs_lib_work_mem = (UVCS_U32*)kmalloc( uvcs_lib_work_req_size, GFP_KERNEL ) ;
	if(uvcs_lib_work_mem == NULL ) {
		return -ENOMEM ;
	}

	uvcs_init_param.struct_size		= sizeof( UVCS_CMN_INIT_PARAM_T ) ;
	uvcs_init_param.hw_num		= NUM_OF_HARDWARE ; /* see LSI spec. ( R-CarH2:2, R-CarM2/E2:1 ) */

	/* work memory */
	uvcs_init_param.work_mem_0_size = uvcs_lib_work_req_size ;
	uvcs_init_param.work_mem_0_virt = uvcs_lib_work_mem ;

	/* ip information */
	uvcs_init_param.ip_base_addr[0][0] = VLC_BASE_ADDR ; /* see LSI spec. (module base address of VLC(VCP)) */
	uvcs_init_param.ip_base_addr[0][1] = CE_BASE_ADDR ;    /* see LSI spec. (module base address of CE(VCP)) */
	if( uvcs_lib_work_mem != 1 ) {
		/* for example R-CarH2 */
		uvcs_init_param.ip_base_addr[1][0] = VLC1_BASE_ADDR ; /* see LSI spec. (module base address of VLC(VCP1)) */
		uvcs_init_param.ip_base_addr[1][1] = CE1_BASE_ADDR ;    /* see LSI spec. (module base address of CE(VCP1)) */
	}
	. . .

	/* init UVCS-CMN library */
	uvcs_ret = uvcs_cmn_initialize( &uvcs_init_param, &uvcs_info ) ;
	if( uvcs_ret != UVCS_RTN_OK ) {
		kfree( uvcs_lib_work_mem ) ;
		return –EFAULT ;
	}
	return 0 ;
}
```

Please refer to uvcs_lkm_io_init function in uvcs_lkm_uf_io.c.

This sample is setting the parameter for R-CarH2. In the other LSI, please modify the parameter of UVCS_CMN_INIT_PARAM_T.

1.1.3 uvcs_cmn_deinitialize

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_U32 *lib_work_mem ;

void sample_deinitialize( void )
{
        UVCS_RESULT uvcs_ret ;

        uvcs_ret = uvcs_cmn_deinitialize( uvcs_info, UVCS_TRUE ) ;
        if( uvcs_ret != UVCS_RTN_OK ) {
                /* FATAL ERROR */
        }
        if( uvcs_lib_work_mem != NULL ) {
                vfree( uvcs_lib_work_mem ) ;
        }
}
```

Please refer to uvcs_lkm_io_cleanup function in uvcs_lkm.c.

1.1.4 uvcs_cmn_open

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_CMN_HANDLE uvcs_hdl ;
static UVCS_U32 *uvcs_hdl_work ;
static UVCS_U32 uvcs_hdl_work_req_size ;

int sample_open_dev( )
{
        UVCS_CMN_OPEN_PARAM_T open_param ;
        UVCS_RESULT uvcs_ret ;

        uvcs_hdl_work = kmalloc( uvcs_hdl_work_req_size, GFP_KERNEL ) ;
        if(uvcs_hdl_work == NULL ) {
                return –ENOMEM ;
        }
        open_param.struct_size = sizeof( UVCS_CMN_OPEN_PARAM_T ) ;
        open_param.hdl_work_0_virt = uvcs_hdl_work ;
        open_param.hdl_work_0_size = uvcs_hdl_work_req_size ;
        open_param.preempt_mode = UVCS_FALSE ;
        uvcs_ret = uvcs_cmn_open( uvcs_info, &open_param, &dev_handle ) ;

        switch( uvcs_ret ) {
                /* interrupted */
                case UVCS_RTN_CONTINUE :
                        kfree( uvcs_hdl_work ) ;
                return -ERESTARTSYS ;

                case UVCS_RTN_OK :
                return 0 ;

                default :
                     kfree( uvcs_hdl_work ) ;
                return -EFAULT ;
        }
```

Please refer to uvcs_lkm_open function in uvcs_lkm.c.

1.1.5 uvcs_cmn_close

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_CMN_HANDLE uvcs_hdl ;
static UVCS_U32 *uvcs_hdl_work ;

int sample_close_dev( )
{
	UVCS_RESULT uvcs_ret ;

	do {
		uvcs_ret = uvcs_cmn_close( uvcs_info, uvcs_hdl, UVCS_FALSE ) ;

		switch( uvcs_ret ) {
			/* interrupted */
			case UVCS_RTN_CONTINUE :
			return -ERESTARTSYS ;

			/* this handle is working */
			case UVCS_RTN_BUSY :
				if( wait_cnt >= UVCS_LKM_CLOSE_WAIT_MAX ) {
					(void)uvcs_cmn_close( uvcs_info, uvcs_hdl, UVCS_TRUE ) ;
					uvcs_ret = UVCS_RTN_OK ;
				}
				else  {
					msleep( UVCS_LKM_CLOSE_WAIT_TIME ) ;
					wait_cnt++ ;
				}
			break ;

			case UVCS_RTN_OK :
			break ;

			default :
			return -EFAULT ;
		}
	} while (uvcs_ret == UVCS_RTN_BUSY ) ;
	kfree( hdl->uvcs_hdl_work ) ;
}
```

Please refer to uvcs_lkm_release function in uvcs_lkm.c.

1.1.6 uvcs_cmn_request

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;

int sample_call_uvcs_request(
        UVCS_CMN_HANDLE target, void *req_data, UVCS_U32 num_bytes )
{
    UVCS_RESULT uvcs_ret ;
    UVCS_CMN_HWPROC_PARAM_T req_param ;

    req_param.struct_size = sizeof( UVCS_CMN_HWPROC_PARAM_T ) ;
    req_param.req_serial = 0 /* user defined value */ ;
    memcpy( req_param.cmd_param, req_data, num_bytes ) ;
    uvcs_ret = uvcs_cmn_request( uvcs_info, target, &req_param ) ;

    switch( uvcs_ret ) {
        /* interrupted */
        case UVCS_RTN_CONTINUE :
            result = -ERESTARTSYS ;
        break ;

        case UVCS_RTN_OK :
            result = num_bytes ;
        break ;

        /* unacceptable request */
        case UVCS_RTN_BUSY :
            result = -EBUSY ;
        break ;

        default :
            result = -EFAULT ;
        break ;
    }
    return result;
}
```

Please refer to uvcs_lkm_write function in uvcs_lkm.c.

1.1.7 uvcs_cmn_execute

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;

void sample_ execute ( void ) (
{
        uvcs_cmn_execute ( lib_info, 0 ) ;
}
```

Please refer to uvcs_lkm_thread function in uvcs_lkm_uf_thread.c.

1.1.8 uvcs_cmn_set_preempt_mode

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_CMN_HANDLE uvcs_hdl ;

int sample_ set_preempt_mode ( unsigned int cmd )
{
	UVCS_RESULT uvcs_ret ;

	if( cmd == UVCS_IOCTL_SET_PREEMPT_MODE ) {
		uvcs_ret = uvcs_cmn_set_preempt_mode ( uvcs_info, &uvcs_hdl, UVCS_TRUE, 0 ) ;

		switch( uvcs_ret ) {
			case UVCS_RTN_BUSY :
				result = -EBUSY ;
			break ;
			case UVCS_RTN_OK :
				result = 0 ;
			break ;
			default :
				result = -EINVAL ;
			break ;
		}
	}
	else if ( cmd == UVCS_IOCTL_CLR_PREEMPT_MODE ) {
		uvcs_ret = uvcs_cmn_set_preempt_mode( uvcs_info, &uvcs_hdl, UVCS_FALSE, 0 ) ;
		if( uvcs_ret == UVCS_RTN_PARAMETER_ERROR ) {
			result = -EINVAL ;
		}
	}
	else {
		;
	}

	return result ;
}
```

Please refer to uvcs_lkm_ioctl function in uvcs_lkm.c.

1.1.9 uvcs_cmn_get_work_size

<Sample Codes>

```
static UVCS_U32 uvcs_lib_work_req_size ;
static UVCS_U32 uvcs_hdl_work_req_size ;

int sample_get_work_size ( )
{
	UVCS_RESULT	uvcs_ret ;

	uvcs_ret = uvcs_cmn_get_work_size ( &uvcs_lib_work_req_size, &uvcs_hdl_work_req_size ) ;
	if( uvcs_ret != UVCS_RTN_OK ) {
		return -EINVAL ;
	}

	return 0 ;
}
```

Please refer to uvcs_lkm_init function in uvcs_lkm.c.

1.1.10 uvcs_cmn_get_ip_info

<Sample Codes>

```
static UVCS_CMN_LIB_INFO uvcs_info ;
static UVCS_CMN_IP_INFO_T ip_info ;

int sample_ get_ip_info ( )
{
        UVCS_RESULT uvcs_ret ;

        uvcs_ret = uvcs_cmn_get_ip_info( uvcs_info, &ip_info ) ;
        if( uvcs_ret != UVCS_RTN_OK ) {
                return –EFAULT ;
        }

        return 0 ;
}
```

Please refer to uvcs_lkm_init function in uvcs_lkm.c.

1.2 System Callback

1.2.1 UVCS_CMN_CB_REG_READ

<Sample Codes>

```
void sample_cb_register_read( UVCS_PTR udptr, volatile UVCS_U32 *reg_addr,
                                                UVCS_U32 *dst_addr, UVCS_U32 num_reg )
{
        while( num > 0 ) {
                *dst++ = ioread32( reg++ ) ;
                num-- ;
        }
        rmb() ;
}
```

Please refer to uvcs_lkm_uf_register_read function in uvcs_lkm_uf_io.c.

1.2.2 UVCS_CMN_CB_REG_WRITE

<Sample Codes>

```
void sample_cb_register_write( UVCS_PTR udptr, volatile UVCS_U32 *reg_addr,
                                                UVCS_U32 *src_addr, UVCS_U32 num_reg )
{
        while( num_reg > 0 ) {
                iowrite32( *src_addr++, reg_addr++ ) ;
                num_reg-- ;
        }
        wmb();
}
```

Please refer to uvcs_lkm_uf_register_write function in uvcs_lkm_uf_io.c.

1.2.3 UVCS_CMN_CB_HW_START

<Sample Codes>

```
void sample_cb_uf_hw_start ( UVCS_PTR udptr, UVCS_U32 hw_ip_id,
                                                UVCS_U32 hw_module_id, UVCS_U32 *baa )
{
        /* power management code or hw-cache setting code is implemented here, if needed */
        /* ex. VPC cache flush */
}
```

Please refer to uvcs_lkm_uf_hw_start function in uvcs_lkm_uf_io.c.

This sample is setting the parameter of VPC for R-CarH2. In the other LSI, please modify the parameter of VPC.

1.2.4 UVCS_CMN_CB_HW_STOP

<Sample Codes>

```
void sample_cb_hw_stop ( UVCS_PTR udptr, UVCS_U32 hw_ip_id,
                                                UVCS_U32 hw_module_id )
{
        /* power management code or hw-cache setting code is implemented here, if needed */
}
```

Please refer to uvcs_lkm_uf_hw_stop function in uvcs_lkm_uf_io.c.

1.2.5 UVCS_CMN_CB_PROC_DONE

<Sample Codes>

```
wait_queue_head_t  waitq ;

void sample_cb_proc_done ( UVCS_PTR udptr, UVCS_PTR hdl_udptr,
                           UVCS_CMN_HANDLE handle, UVCS_CMN_HW_PROC_T *res_info )
{
    wake_up_interruptible( &waitq ) ;
}
```

Please refer to uvcs_lkm_uf_hw_processing_done function in uvcs_lkm_uf_other.c.

1.2.6 UVCS_CMN_CB_SEM_LOCK

<Sample Codes>

```
static struct semaphore sem ;

UVCS_BOOL sample_cb_semaphore_lock( UVCS_PTR udptr )
{
        if( down_interruptible( &sem ) ) {
                return UVCS_FALSE ;
        }
        return UVCS_TRUE ;
}
```

Please refer to uvcs_lkm_uf_semaphore_lock function in uvcs_lkm_uf_semaphore.c.

1.2.7 UVCS_CMN_CB_SEM_UNLOCK

<Sample Codes>

```
static struct semaphore sem ;

void sample_cb_semaphore_unlock( UVCS_PTR udptr )
{
        up( &sem ) ;
}
```

Please refer to uf_uvcs_semaphore_unlock function in uvcs_lkm_uf_semaphore.c.

1.2.8 UVCS_CMN_CB_SEM_CREATE

<Sample Codes>

```
static struct semaphore sem ;

UVCS_BOOL sample_cb_semaphore_create( UVCS_PTR udptr )
{
        sema_init( &sem, 1 ) ;
        return UVCS_TRUE ;
}
```

Please refer to uvcs_lkm_uf_semaphore_create function in uvcs_lkm_uf_semaphore.c.

1.2.9 UVCS_CMN_CB_SEM_DESTROY

<Sample Codes>

```
static struct semaphore sem ;

void sample_cb_semaphore_destroy( UVCS_PTR udptr )
{
        /* nothing */
}
```

Please refer to uvcs_lkm_uf_semaphore_destroy function in uvcs_lkm_uf_semaphore.c.

1.2.10 UVCS_CMN_CB_THREAD_EVENT

<Sample Codes>

```
wait_queue_head_t wait_q ;

void sample_cb_thread_event( UVCS_PTR udptr )
{
        evt_req = true ;
        wake_up_interruptible( &wait_q ) ;
}
```

Please refer to uvcs_lkm_uf_thread_event function in uvcs_lkm_uf_thread.c.

1.2.11 UVCS_CMN_CB_THREAD_CREATE

<Sample Codes>

```
wait_queue_head_t wait_q ;

UVCS_BOOL sample_cb_thread_create( UVCS_PTR udptr )
{
        evt_req = false ;
        evt_stop = false ;
        init_waitqueue_head( &wait_q ) ;
        thread = kthread_run(    thread_func , udptr, "uvcs" ) ;
        if( IS_ERR( thread ))
        {
                return UVCS_FALSE ;
        }
        return UVCS_TRUE ;
}
```

Please refer to uvcs_lkm_uf_thread_create function in uvcs_lkm_uf_thread.c.

1.2.12 UVCS_CMN_CB_THREAD_DESTROY

<Sample Codes>

```
task_struct  thread ;

void sample_cb_thread_destroy ( UVCS_PTR udptr )
{
        /* please terminate thread loop by some method */
        kthread_stop( thread ) ;
        thread = NULL ;
}
```

Please refer to uvcs_lkm_uf_thread_destroy function in uvcs_lkm_uf_thread.c.

1.3 Makefile

1.3.1 Linux

<Sample >

```
CUR_DIR := $(shell pwd)

obj-m := uvcs_cmn.o
uvcs_cmn-y := uvcs_lkm.o
uvcs_cmn-y += uvcs_lkm_uf_semaphore.o
uvcs_cmn-y += uvcs_lkm_uf_thread.o
uvcs_cmn-y += uvcs_lkm_uf_other.o
uvcs_cmn-y += uvcs_lkm_uf_io.o
uvcs_cmn-y += uvcs_cmn_api.o
uvcs_cmn-y += uvcs_cmn_dump.o
uvcs_cmn-y += mcvx_api.o

EXTRA_CFLAGS := -DUVCS_DEBUG=1
EXTRA_CFLAGS += -mno-unaligned-access

.PHONY: all
all:
@cp -p $(UVCS_DRV_SRC_DIR)/*.c .
@cp -p $(UVCS_DRV_SRC_DIR)/*.h .
@cp -p $(UVCS_CMN_SRC_DIR)/*.c .
@cp -p $(UVCS_CMN_SRC_DIR)/*.h .
@cp -p $(UVCS_CMN_INC_DIR)/*.h .
@cp -p $(DRV_CORE_SRC_DIR)/*.c .
@cp -p $(DRV_CORE_SRC_DIR)/*.h .
@make --no-print-directory -C $(KERNELDIR) M=$(PWD) modules
@rm -f *.c
@rm -f *.h

clean:
@make --no-print-directory -C $(KERNELDIR) M=$(PWD) clean
```

(*1) UVCS_***_DIR has to set the each folder. The setting example is shown in following.

<Sample >

```
UVCS_DRV_SRC_DIR := ../uvcs_lkm
UVCS_CMN_SRC_DIR := ../uvcs_cmn
UVCS_CMN_INC_DIR := ../../include
DRV_CORE_SRC_DIR := ../driver_core
```

(*1) The root folder is “makefile” in product CD.

1.3.2 Android

<Sample >

```
CUR_DIR := $(shell pwd)

obj-m := uvcs_cmn.o
uvcs_cmn-y := uvcs_lkm.o
uvcs_cmn-y += uvcs_lkm_uf_semaphore.o
uvcs_cmn-y += uvcs_lkm_uf_thread.o
uvcs_cmn-y += uvcs_lkm_uf_other.o
uvcs_cmn-y += uvcs_lkm_uf_io.o
uvcs_cmn-y += uvcs_cmn_api.o
uvcs_cmn-y += uvcs_cmn_dump.o
uvcs_cmn-y += mcvx_api.o

EXTRA_CFLAGS := -DUVCS_DEBUG=1
#EXTRA_CFLAGS += -mno-unaligned-access

.PHONY: all
all:
@cp -p $(UVCS_DRV_SRC_DIR)/*.c .
@cp -p $(UVCS_DRV_SRC_DIR)/*.h .
@cp -p $(UVCS_CMN_SRC_DIR)/*.c .
@cp -p $(UVCS_CMN_SRC_DIR)/*.h .
@cp -p $(UVCS_CMN_INC_DIR)/*.h .
@cp -p $(DRV_CORE_SRC_DIR)/*.c .
@cp -p $(DRV_CORE_SRC_DIR)/*.h .
@make --no-print-directory CFLAGS_MODULE=-fno-pic -C $(KERNELDIR) M=$(PWD) modules
@rm -f *.c
@rm -f *.h

clean:
@make --no-print-directory -C $(KERNELDIR) M=$(PWD) clean
```

(*1) UVCS_***_DIR has to set the each folder. The setting example is shown in following.

<Sample >

```
UVCS_DRV_SRC_DIR := ../uvcs_lkm
UVCS_CMN_SRC_DIR := ../uvcs_cmn
UVCS_CMN_INC_DIR := ../../include
DRV_CORE_SRC_DIR := ../driver_core
```

(*1) The root folder is “makefile” in product CD.

1.4 VPC setting

1.4.1 vpc_init

This function is the parameter setting for VPC. The detail of registers refers to the LSI manual (30A.2 Register Descriptions).

<Sample Codes>

```
static void uvcs_vpc_init(struct uvcs_drv_info *local, UVCS_U32 hw_ip_id)
{
	if (reg_vpc[hw_ip_id] != 0uL) {
		void *vpcctl = (void *)(reg_vpc[hw_ip_id] + UVCS_VPCREG_VPCCTL);
		void *vpccfg = (void *)(reg_vpc[hw_ip_id] + UVCS_VPCREG_VPCCFG);
		void *vpcsts = (void *)(reg_vpc[hw_ip_id] + UVCS_VPCREG_VPCSTS);
		UVCS_U32 regdata;

		uvcs_vpc_wait_end(vpcsts);
		regdata = ioread32(vpcctl);
		rmb();
		regdata |= 0x100CuL;
		iowrite32(regdata, vpcctl);
		wmb();
		uvcs_vpc_clear(vpcctl);

		switch (local->module_param.lsi_type) {
		case UVCS_LSITYPE_H2_VX:
			regdata = 0x0002030AuL | UVCS_VPCCFG_MODE;
			iowrite32(regdata, vpccfg);
			wmb();
			if ((regdata & 0x00000001) != 1uL)
				regdata = 0x0uL;
			else
				regdata = 0x3uL;
			iowrite32(regdata, (void *)reg_vpcxy[hw_ip_id]);
			break;

		case UVCS_LSITYPE_M2W_VX:
		case UVCS_LSITYPE_M2N:
		case UVCS_LSITYPE_E2:
			regdata = 0x0002030AuL | UVCS_VPCCFG_MODE;
			iowrite32(regdata, vpccfg);
			break;

		case UVCS_LSITYPE_M2W_V1:
			regdata = 0x0002030BuL;
			iowrite32(regdata, vpccfg);
			break;

		case UVCS_LSITYPE_H2_V1:
			iowrite32(0, vpccfg);
			break;

		default: /* same as M2W_Vx, including V2H */
			regdata = 0x0002030AuL | UVCS_VPCCFG_MODE;
			iowrite32(regdata, vpccfg);
			break;
		}
		wmb();
	}
}
```

1.4.2 vpc_setup

This function is the initialize operation for VPC. The detail of process refers to the LSI manual (30A.3.1 Initialization).

<Sample Codes>

```
static void uvcs_vpc_setup(  struct uvcs_drv_info *local,  UVCS_U32 hw_ip_id,  UVCS_U32 *baa)
{
        void *vpcctl = (void *)(reg_vpc[hw_ip_id] + UVCS_VPCREG_VPCCTL);
        void *vpcsts = (void *)(reg_vpc[hw_ip_id] + UVCS_VPCREG_VPCSTS);
        ulong regdata;

        uvcs_vpc_wait_end(vpcsts);

        regdata = ioread32(vpcctl);
        rmb();
        regdata &= ~0xCuL;

        if (baa[UVCS_BAAIDX_STRIDE] > 1024)
                regdata |= 0x1000000CuL;
        else if (baa[UVCS_BAAIDX_STRIDE] > 512)
                regdata |= 0x10000008uL;
        else if (baa[UVCS_BAAIDX_STRIDE] > 256)
                regdata |= 0x10000004uL;
        else
                regdata |= 0x10000000uL;

        iowrite32(regdata, vpcctl);
        wmb();

        uvcs_vpc_wait_end(vpcsts);
        uvcs_vpc_clear(vpcctl);
        uvcs_vpc_wait_end(vpcsts);
        regdata = ioread32(vpcctl);
        regdata |= 1uL;
        rmb();
        iowrite32(regdata, vpcctl);
        wmb();
}
```

1.4.3 vpc_clear

This function is the clear operation for VPC. The detail of process refers to the LSI manual (30A.3.2 Clearing VPC at the Beginning of a Frame).

<Sample Codes>

```
static void uvcs_vpc_clear(void *vpcctl)
{
        ulong reg;
        ulong wait_cnt = 0uL;

        reg = ioread32(vpcctl);
        rmb();

        reg |= 0x2uL;
        iowrite32(reg, vpcctl);
        wmb();

        reg = ioread32(vpcctl);
        rmb();

        while (((reg & 0x2uL) != 0uL) && (wait_cnt < UVCS_VPC_WAIT_MAX)) {
                udelay(UVCS_VPC_WAIT_TIME);
                wait_cnt++;
                reg = ioread32(vpcctl);
                rmb();
        }
}
```

REVISION HISTORY	UVCS Adapted for Linux Common Engine Library Sample Codes Manual

Rev.	date	revision	
		pages	point
0.10	2013.05.31	－	alpha version 1.0
0.20	2013.06.30	－	alpha version 2.0
0.30	2013.07.26	－	alpha version 3.0
0.40	2013.08.26	－	alpha version 4.0
0.50	2013.09.09	－	alpha version 5.0
0.60	2013.10.21	－	alpha version 6.0
0.70	2013.11.28	－	alpha version 7.0
0.80	2014.01.31	－	alpha version 8.0
0.90	2014.03.06	－	alpha version 9.0
0.91	2014.05.12	－	beta version 6.0

UVCS Adapted for Linux Common Engine Library
Samples Code Manual

Publication Date Rev.0.91 May 12, 2014

Published by Renesas Electronics Corporation

UVCS Adapted for Linux Common Engine Library

Sample Codes Manual

Renesas Electronics Corporation