SAMSUNG
ELECTRONICS

Porting Guide
For Post processor

S3C6400/6410

August 29, 2008

(Preliminary) REV 2.10

1Important Notice

S3C6400/6410 RISC Microprocessor
Porting Guidel for Post Processor

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City Gyeonggi-Do, Korea
446-711

Home Page: http://www.samsungsemi.com/
E-Mail: mobilesol.cs@samsung.com

Printed in the Republic of Korea

Revision History

Revision	Description of Change	Refer to	Author(s)	Date
1.00	- Master Copy	-	Jiun Yu	2007-06-22
1.20	- Commands are added		Jiun Yu	2007-10-29
1.21	- Parameter was changed		Jiun Yu	2008-01-17
2.00	-Supporting Post processor driver on S3C6410		Jiun Yu	2008-07-05
2.10	- Tested on linux kernel 2.6.24		Jiun Yu	2008-08-29

Contents

Table

1 Introduction

In this Chapter, you will understand the following:

- Section 1.1, “Overview”
- Section 1.2, “Features”

1.1 Overview

Post processor performs video/graphic scale, video format conversion and color space conversion

1.2 FEATURES

The features of the Post Processor device driver are:

- H/W feature :
  - Color space conversion from YCBCR to RGB
  - Color space conversion from RGB to YCBCR
  - Programmable source and destination image size up to 2048 x 2048 resolution
  - Free run mode operation
  - Dedicated DMA
- S/W feature :
  - The function of post processing can be used in MFC device driver

2 Setup Post Processor Driver Environment

In this Chapter, you will understand the following:

- Setup for Post Processor driver test environment

2.1 Environment Overview

We need to setup environment to use and test Post Processor device driver on evaluation board.

Category	Contents	Remark
Tool chain	Gcc 4.2.2-eabi arm cross compiler	For application compiling
	Gcc 4.2.2-eabi arm cross compiler	For kernel compiling
Kernel	Embedded linux 2.6.24	For smdk6400/6410
Test tool	Post_test.c (Post Processor test application)	Source code is included in this document
Root file system	nfs, cramfs	Add device node for Post Processor required.
Device nodes	/dev/misc/s3c-pp	Device type : c Major number : 10 Minor number : 253

Table -. Post Processor Driver Environment

3 Supported Post Processor API list

This chapter explains the post processor device driver API

3.1 API feature list

This gives information of the post processor device driver API.

File operation	Post Processor API	Remark
Open	s3c_pp_open	
Mmap	S3c_pp_mmap	
close	S3c_pp_release	
Ioctl	S3c_pp_ioctl	PPROC_SET_PARAMS PPROC_START PPROC_STOP PPROC_INTERLACE_MODE PPROC_PROGRESSIVE_MODE PPROC_GET_PHY_INBUF_ADDR PPROC_GET_INBUF_SIZE PPROC_GET_BUF_SIZE PPROC_GET_OUT_DATA_SIZE

Table - Post Processor Driver API List

3.2 API detail information

3.2.1 open

Open	
Syntax	Int open(const char * path, int oflag)
Remark	This function opens the post processor driver.
Parameters	Path [IN] : Post Processor device node path Oflag[IN] : flags of Post Processor.
Return Value	File descriptor of the Post Processor

3.2.2 mmap

Mmap	
Syntax	Void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t off);
Remark	This function maps physically continuous memory. This memory can share user and device driver. This memory is used as in/out buffer of the post processor
Parameters	Addr[IN] : none Len[IN] : mapped memory size Prot[IN] : memory access permission(PROT_READ, PROT_WRITE, etc) Flag[IN] : attribute of memory (MAP_SHARED, etc) fd [IN] : File descriptor of the Post Processor off[IN] : none
Return Value	Base address of input buffer. This address can be used in user application.

3.2.3 close

Close	
Syntax	Int close(int fd)
Remark	This function releases the post processor driver
Parameters	fd [IN] : File descriptor of the post processor
Return Value	Close success/fail

3.2.4 Ioctl

Ioctl	
Syntax	Int ioctl(int fd, int cmd, …)
Remark	Most of functions are developed in ioctl. This system call has several functions which is separated by cmd
Parameters	fd [IN] : File descriptor of the Post Processor cmd [IN] : There are several functions. Detailed information of cmd will explain below.
Return Value	It depends on cmd.

Ioctl - PPROC_SET_PARAMS

Ioctl – PPROC_SET_PARAMS	
Syntax	Int ioctl(int fd, int cmd, scaler_params *pp_param)

Remark	This command sets parameters for postprocessing.
	
	Parameters are :
	
	typedef struct{
	unsigned int SrcFullWidth; // Source Image Full Width(Virtual screen size)
	unsigned int SrcFullHeight;// Source Image Full Height(Virtual screen size)
	unsigned int SrcStartX; // Source Image Start width offset
	unsigned int SrcStartY; // Source Image Start height offset
	unsigned int SrcWidth; // Source Image Width
	unsigned int SrcHeight; // Source Image Height
	unsigned int SrcFrmSt; // Base Address of the Source Image : Physical Address
	cspace_t SrcCSpace; // Color Space ot the Source Image
	
	unsigned int DstFullWidth; // Source Image Full Width(Virtual screen size)
	unsigned int DstFullHeight; // Source Image Full Height(Virtual screen size)
	unsigned int DstStartX; // Source Image Start width offset
	unsigned int DstStartY; // Source Image Start height offset
	unsigned int DstWidth; // Source Image Width
	unsigned int DstHeight; // Source Image Height
	unsigned int DstFrmSt;// Base Address of the Source Image : Physical Address
	cspace_t DstCSpace; // Color Space ot the Source Image
	
	unsigned int SrcFrmBufNum; // Frame buffer number
	s3c_pp_run_mode_t Mode; // POST running mode(PER_FRAME or FREE_RUN)
	s3c_pp_path_t InPath; // Data path of the source image

	s3c_pp_path_t OutPath; // Data path of the desitination image
	
	unsigned int in_pixel_size; // source format size per pixel
	unsigned int out_pixel_size; // destination format size per pixel
	}pp_params;

Parameters	fd [IN] : File descriptor of the Post Processor cmd [IN] : PPROC_SET_PARAMS pp_param[IN] : parameters for postprocessing
Return Value	None

Ioctl - PPROC_START

Ioctl – PPROC_START	
Syntax	Int ioctl(int fd, int cmd)
Remark	The post processor is started by this command.
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_START
Return Value	None

Ioctl - PPROC_STOP

Ioctl – PPROC_STOP	
Syntax	Int ioctl(int fd, int cmd)
Remark	The post processor is stopped by this command.
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_STOP
Return Value	None

Ioctl - PPROC_INTERLACE_MODE

Ioctl – PPROC_INTERLACE_MODE	
Syntax	Int ioctl(int fd, int cmd)
Remark	This command set the interlace mode
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_INTERLACE_MODE
Return Value	None

Ioctl - PPROC_PROGRESSIVE_MODE

Ioctl – PPROC_PROGRESSIVE_MODE	
Syntax	Int ioctl(int fd, int cmd)

Remark	This command set the progressive mode
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_PROGRESSIVE_MODE
Return Value	None

Ioctl - PPROC_GET_PHY_INBUF_ADDR

Ioctl – PPROC_GET_PHY_INBUF_ADDR	
Syntax	Int ioctl(int fd, int cmd)
Remark	This command get the physical address of input buffer
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_GET_PHY_INBUF_ADDR
Return Value	Physical address of input buffer

Ioctl - PPROC_GET_INBUF_SIZE

Ioctl – PPROC_GET_INBUF_SIZE	
Syntax	Int ioctl(int fd, int cmd)
Remark	This command get the size of input buffer
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_GET_ INBUF_SIZE
Return Value	Size of input buffer

Ioctl - PPROC_GET_ BUF_SIZE

Ioctl – PPROC_GET_ BUF_SIZE	
Syntax	Int ioctl(int fd, int cmd)
Remark	This command get the size of total buffer
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_GET_ BUF_SIZE
Return Value	Size of total buffer

Ioctl - PPROC_GET_ OUT_DATA_SIZE

Ioctl – PPROC_GET_ OUT_DATA_SIZE	
Syntax	Int ioctl(int fd, int cmd)
Remark	This command get the output size
Parameters	fd [IN] : File descriptor of the post processor cmd [IN] : PPROC_GET_ OUT_DATA_SIZE

Return Value	Size of output data

4 Sample code

In this Chapter, you will understand the following:

- Post processing sample code

4.1 Post processing sample code

Below source code explains post processing process

Post processing test	Post_test.c

```
int main(int argc, char **argv)
{
      int                 dev_fd, in_fd, out_fd;
      int                 file_size;
      int                 buf_size;
      int                 out_size;
      char         *in_addr;
      char         *in_buf, *out_buf;
      struct stat  s;
      pp_params    pp_param;


      if(argc != 11 ) {
             printf("Check number of arguments!!!\n");
             printf("Usage   :   [src_width]   [src_height]   [src_format]
[dst_width] [dst_height] [dst_format] ");
             printf("[out_path]   [mode]   [in   file   name]   [out   file
name]\n\n");
             printf("[src/dst_format] : 6(RGB16), 9(RGB24), 12(420YCbCr),
14(422YCBYCR)\n");
             printf("                        15(422YCRYCB),  16(422CBYCRY),
17(422CRYCBY)\n");
             printf("[out_path] : 0(DMA), 1(FIFO)\n");
             printf("[mode] : 0(ONE-SHOT), 1(FREE-RUN)\n");

             return -1;
      }

      // set post processor configuration
      pp_param.SrcFullWidth      = atoi(argv[1]);
      pp_param.SrcFullHeight     = atoi(argv[2]);
      pp_param.SrcCSpace         = atoi(argv[3]);
      pp_param.DstFullWidth      = atoi(argv[4]);
      pp_param.DstFullHeight     = atoi(argv[5]);
      pp_param.DstCSpace         = atoi(argv[6]);
      pp_param.OutPath           = atoi(argv[7]);
      pp_param.Mode              = atoi(argv[8]);


      // open in/out file
      in_fd = open(argv[9], O_RDONLY);
      out_fd = open(argv[10], O_RDWR | O_CREAT | O_TRUNC, 0644);
      if((in_fd < 0) || (out_fd < 0)) {
             printf("input/output file open error\n");
```

```
		return -1;
	}

	// get input file size
	fstat(in_fd, &s);
	file_size = s.st_size;

	// mapping input file to memory
	in_addr = (char *)mmap(0, file_size, PROT_READ, MAP_SHARED, in_fd, 
0);
	if(in_addr == NULL) {
		printf("input file memory mapping failed\n");
		return -1;
	}

	// open post processor
	dev_fd = open(DEVICE_FILE_NAME, O_RDWR|O_NDELAY);
	if(dev_fd < 0)
	{
		printf("Post processor open error\n");
		return -1;
	}

	// in_buf is post processor input buffer
	buf_size = ioctl(dev_fd, PPROC_GET_BUF_SIZE);
	in_buf   =   (char   *)   mmap(0,   buf_size,   PROT_READ   |   PROT_WRITE, 
MAP_SHARED, dev_fd, 0);
	if(in_buf == NULL) {
		printf("Post processor mmap failed\n");
		return -1;
	}
	out_buf = in_buf + ioctl(dev_fd, PPROC_GET_INBUF_SIZE);

	memcpy(in_buf, in_addr, file_size);

	pp_param.SrcFrmSt = ioctl(dev_fd, PPROC_GET_PHY_INBUF_ADDR);
	pp_param.DstFrmSt = pp_param.SrcFrmSt        +        ioctl(dev_fd, 
PPROC_GET_INBUF_SIZE);

	ioctl(dev_fd, PPROC_SET_PARAMS, &pp_param);
	ioctl(dev_fd, PPROC_START);

	out_size = ioctl(dev_fd, PPROC_GET_OUT_DATA_SIZE);
	write(out_fd, out_buf, out_size);

	munmap(in_buf, buf_size);

	close(dev_fd);
	close(in_fd);
	close(out_fd);

	return 0;
}
```