SONiX SN986 Serial IP Camera SoC

SDK Programming Guide

Document No.:

Version: v1.01

Revision	Date	Description
1.00	2013-11-09	Formal Release
1.01	2014-09-04	Update SDK architecture
Apply to		

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Table of Contents

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

1. Overview

SONiX SN986 Series provide a helpful software development kit (SDK) for customers to develop their own products quickly and easily. SN986 Series SDK implements a Linux 2.6.35 software development environment for SN986 Series platform, including Linux kernel distribution, SONiX drivers, SONiX middleware, and user applications.

This chapter will introduce the architecture and installation of the SN986 Series SDK.

1.1 SDK Architecture

SN986 Series SDK could be divided into three layers by the functional, which are “application layer”, “middleware layer”, and “Linux kernel and SONiX driver layer”.

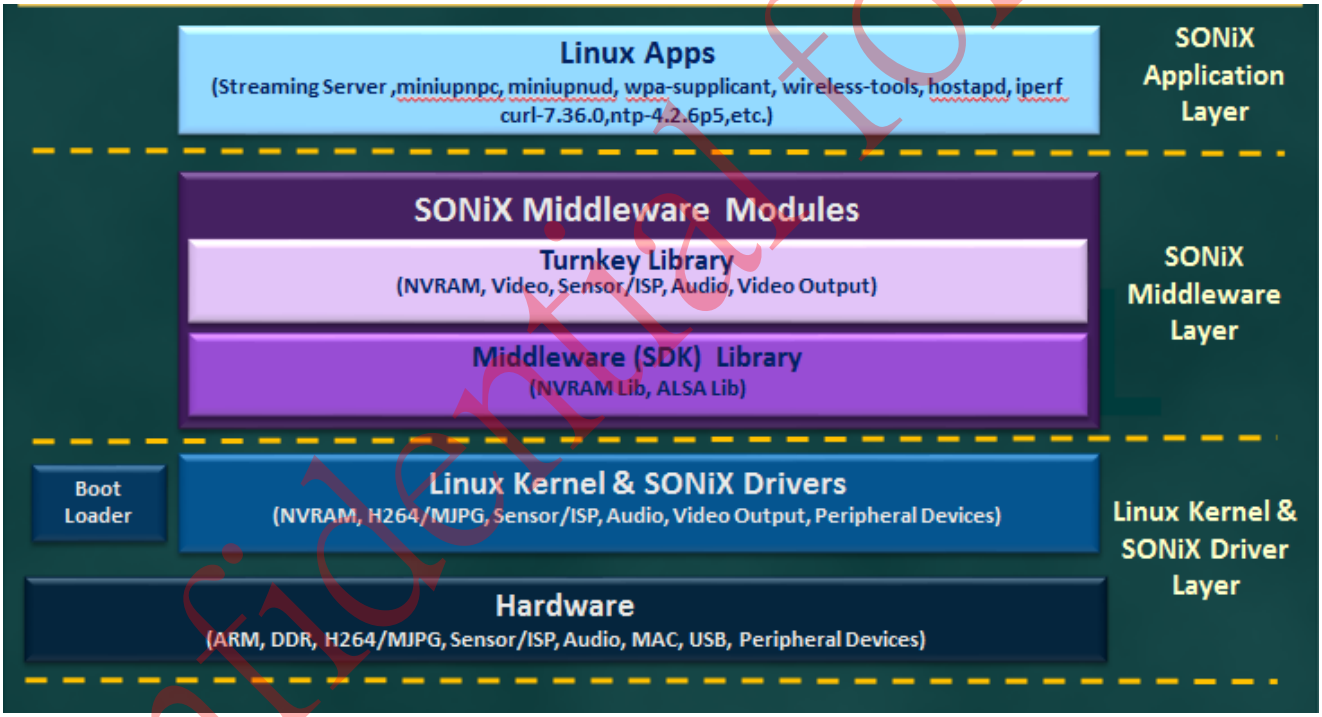

Figure 01.SN986 Series SDK architecture

On the application layer, SN986 Series SDK provides not only rich user applications based on linux, but also useful tools to assist users in developing products.

SN986 Series middleware provides APIs to communicate between user applications and SONiX drivers. Through these APIs, users can easily implement their applications to control the SONiX Audio/Video hardware. All the system and peripheral drivers will be run on the Linux kernel and SONiX driver layer.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

1.2 SDK Directory Structure

Directory	Description
app	Directory for the repository of the user applications
bootloader	Directory for the U-boot
buildscript	Directory for SDK build scripts and configurations
driver	Directory for the SONiX driver modules
filesystem	Directory for the busybox and rootfs
image	Directory for the images
kernel	Directory for the repository of the Linux kernel source
middleware	Directory for SONiX middleware
toolchain	Directory for the repository of the cross compilation tools and image tools

Table 01. SN986 Series SDK Directory Structure

1.3 Installation

Please simply follow the steps below to install and compile the SN986 series SDK on the Linux server.

1. Copy the SDK package (It would be compressed to the gzip format.) to Linux server.
2. Decompress the preceding package by the command, "*tar –xzf SDKNAME.gz*"
3. Wait for the decompression completion
4. Enter SN986 series SDK directory by typing "*cd SDKNAME*"
5. Execute the unpack shell, "*./sdk.unpack*", to unpack all the development packages.
6. After unpacking is done, all the development packages are installed.
7. Enter the unpacked code directory, "*cd snx_sdk*".
8. Enter buildscript directory, "*cd buildscrip*t"
9. Type "make XXXX_defconfig" to load the SDK configuration
   (To look up the defconfigs, please type "*ls scrips/configs*")
10. Type "make" to start compilation
11. Type "make install" to generate the images.
12. When the compilation is done, the image would be found in "image" folder.

For more information about the development environment setup, please refer to "SN986 Series Linux Environment User Guide".

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

2. Peripherals

2.1 Timer

The timer provides 3 independent sets of the sub-timers. Each sub-timer can use the system clock (OSC) for the increment or decrement counting. 2 match registers are provided for each sub-timer. Whenever a value of the match registers equals to any one of the sub-timers, the timer interrupt is triggered immediately. The issuance of the timer interrupt can be decided by the register setting when the overflow occurs. The timer features include:

There are three independent 32-bit timer programming models, and the interrupts can be issued upon the overflow and time-up. Each sub-timer has 2 match registers. Programmable decrementing/incrementing modes on the counter

In the Linux kernel, that system use timer1 for HZ tick and timer3 for high resolution timer, only timer2 for user programming.

2.1.1 Function Description

Function Name	Category	Description
request_hw_timer	Initial	Request one special hardware timer
free_hw_timer	release	Free one special hardware timer
set_hw_timer_alarm	Alarm	Set the alarm of special hardware timer
get_hw_timer_time	Get time	Get the time of special hardware timer
enable_hw_timer_measure_mode	measure	Enable the measure mode of special hardware timer
disable_hw_timer_measure_mode	measure	Disable the measure mode of special hardware timer
enable_hw_timer	enable	Enable special hardware timer
disable_hw_timer	Disable	Disable special hardware timer

Table 02. Timer function list

2.1.2 API Usage and Example

- int request_hw_timer(void)
  - return vaule
    - The ID of timer. (greater than or equal to 0:success, less than 0:fail

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- int free_hw_timer (int timer_id)
  - Parameter
    - timer_id: the ID of timer that will be freed
  - return value
    - Zero is success, or failed.

- int set_hw_timer_alarm (int timer_id, unsigned int ms, timer_handler_t handler, unsigned long arg)
  - Parameter
    - timer_id:the ID of timer
    - ms: The alarm time (in Microseconds units)
    - handler: Callback function
    - arg: the parameter of callback function
  - return value
    - return zero is success, or failed.

- int get_hw_timer_time (int timer_id, struct timeval* tv)
  - Parameter
    - timer_id:the ID of timer
    - tv:time value in timeval struct
  - return value
    - return zero is success, or failed.

- int enable_hw_timer_measure_mode (int timer_id)
  - Parmater
    - timer_id:the ID of timer
  - return value
    - return zero is success, or failed.

- int disable_hw_timer_measure_mode (int timer_id)
  - Parameter
    - timer_id:the ID of timer
  - return value
    - return zero is success, or failed.

- int enable_hw_timer (int timer_id)
  - Parameter
    - timer_id:the ID of timer

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- return value
  - return zero is success, or failed.

- int disable_hw_timer (int timer_id)
  - Parameter
    - timer_id:the ID of timer

  - return value
    - 0 is success, other failed.

SN986 Serial SDK provides a simple example to describe how to use timer in kernel space. Please refer to app/example/timer/ snx_timer_test_module.c

The example snx_timer_measure_test () function will calculate the time in kernel space. For the detail information, please run “timer_sh.sh” first, then “snx_timer_test -h”

Example:

```
static int snx_timer_measure_test(void)
{
    struct timeval start_time, end_time;
    struct timeval hw_start_time, hw_end_time;
    unsigned long hw_time, time;
    int retval;
    int timer_id;
    printk(KERN_INFO "timer measure test start.\n");
    timer_id = request_hw_timer();
    if(timer_id < 0)
    {
        printk(KERN_ERR "Request hardware timer fail.\n");
        return -1;
    }
    retval = enable_hw_timer_measure_mode(timer_id);
    if(retval < 0)
    {
        printk(KERN_ERR "Enable hardware timer measure mode fail.\n");
        retval = -2;
        goto err1;
    }
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
retval = enable_hw_timer(timer_id);
if(retval < 0)
{
        printk(KERN_ERR "Enable hardware timer fail.\n");
        retval = -3;
        goto err1;
}
do_gettimeofday(&start_time);
retval = get_hw_timer_time(timer_id, &hw_start_time);
if(retval < 0)
{
        printk(KERN_ERR "Get hardware timer time fail.\n");
        retval = -4;
        goto err2;
}
udelay(1000);
do_gettimeofday(&end_time);
retval = get_hw_timer_time(timer_id, &hw_end_time);
if(retval < 0)
{
        printk(KERN_ERR "Get hardware timer time fail.\n");
        retval = -5;
        goto err2;
}
time = (end_time.tv_sec - start_time.tv_sec) * 1000000;
time += end_time.tv_usec - start_time.tv_usec;
hw_time = (hw_end_time.tv_sec - hw_start_time.tv_sec) * 1000000;
hw_time += hw_end_time.tv_usec - hw_start_time.tv_usec;
printk(KERN_ERR "do_gettimeoofday measure time %ldus\n", time);
printk(KERN_ERR "hw_timer measure time %ldus\n", hw_time);
retval = disable_hw_timer(timer_id);
if(retval < 0)
{
        printk(KERN_ERR "Disable hardware timer fail.\n");
        return -7;
}
retval = free_hw_timer(timer_id);
if(retval < 0)
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
        {
            printk(KERN_ERR "Free hardware timer fail.\n");
            return -8;
        }
        return 0;
    err2:
        disable_hw_timer(timer_id);
    err1:
        free_hw_timer(timer_id);
    return retval;
}
```

2.1.3 Get Current Time

2.1.3.1 kernel space:

Call cpu_clock() API to get current time(micro second) from system startup.

2.1.3.2 User space:

Call gettimeofday() API to get current time.

2.1.4 Related Files

- kernel/linux-2.6.35.12/src/arch/arm/mach-sn986xx/time.c
- kernel/linux-2.6.35.12/src/arch/arm/mach-sn986xx/include/mach/ regs-timer.h
- app/example/timer/ snx_timer_test_module.c
- app/example/timer/ snx_timer_test.c

2.2 WDT

Software stability is a very important on our platform. Anyone who uses software has probably experienced problems that crash the computer or program in question. The crash issue of embedded programs is no user around to reset system when things go wrong. The Only One IP can do this job -- watchdog timer. It can reset the processor in case of a code crash.

Feature :

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- watchdog timer is a 32 bit counter
- The watchdog counter can reload
- watchdog timer clock source clock = EXTCLK which EXTCLK=PCLK/4
- Adjust the assert duration

2.2.1 Function Description

WDT device is “/dev/watchdog”

Function Name	Category	Description
WDIOC_GETSUPPORT	ioctl cmd	Print some feature about this driver
WDIOC_GETSTATUS	ioctl cmd	Get watchdog status
WDIOC_KEEPALIVE	ioctl cmd	Clear watchdog timer value
WDIOC_SETTIMEOUT	ioctl cmd	Set watchdog time out value
WDIOS_DISABLECARD	ioctl cmd	Disable watchdog
WDIOS_ENABLECARD	ioctl cmd	Enable watchdog

Table 03. WDT function list

2.2.2 API Usage and Example

SN986 Serial SDK provides a simple example to describe how to use watchdog. Please refer to app/example/watchdog/ snx_watchdog_test.c
The example snx_watchdog_keepalive_test () function is one of the test functions, and it will keep alive the watchdog.
For the detail usage, please run the command, “snx_watchdog_test –h”

Example:

```
int snx_watchdog_keepalive_test ()
{
    int ret,fd,i;
    unsigned long v,timeout;
    fd = open(WATCHDOG_DEVICE, O_RDONLY);
    if(!fd) {
    printf("Open Failed!\n");
    return WATCHDOG_FAIL;
    }

    timeout = 5; // in watchdog driver default is 10s
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
        ret = ioctl(fd, WDIOC_SETTIMEOUT,    &timeout);
        v = WDIOS_ENABLECARD; // enable watchdog
        ret = ioctl(fd, WDIOC_SETOPTIONS,    &v);

        for(i =0; i < 10; i++)
        {
            sleep(1);
            ret = ioctl(fd, WDIOC_KEEPALIVE,    NULL);
            printf("time %d\n", i);
        }
        v = WDIOS_DISABLECARD; // enable watchdog
        ret = ioctl(fd, WDIOC_SETOPTIONS,    &v);
        close (fd);
        // to check watchdog disable ok, no reset
        sleep (10);
        printf ("check watchdog disable ok\n");
        return WATCHDOG_SUCCESS;
}
```

2.2.3 Related Files

- driver/wdt/src/driver/regs-wdt.h
- driver/wdt/src/driver/ wdt.c
- app/example/watchdog/ snx_watchdog_test.c

2.3 I2C

The I2C is a two-wire bi-directional serial bus that provides a simple and efficient method of the data exchange while minimizing the interconnection between the devices. The I2C bus interface controller allows the host processor to serve as a master or slave residing on the I2C bus. The data are transmitted to and received from the I2C bus via a buffered interface.

The I2C features include:

- Supports the standard and fast modes through programming the clock division register
- Supports 7-bit, 10-bit, and general call addressing modes
- Glitch suppression throughout the de-bounce circuits
- Programmable slave address
- Supports the master-transmit, master-receive, slave-transmit, and slave-receive modes
- Slave mode general call address detection

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- Linux I2C sub-system

2.3.1 Function Description

2.3.1.1 Kernel space

Function Name	Category	Description
i2c_transfer		execute a single or combined I2C message

Table 04. I2C Function List

2.3.1.2 User space

device name “/dev/i2c-0

Function Name	*Category*	Description
I2C_RDWR	ioctl cmd	Combined R/W transfer

Table 05. I2C Function List

2.3.2 API Usage and Example

SN986 Serial SDK provides two simple examples to describe how to use i2c in kernel space & user space. Please refer to app/example/i2c/ ov9715.c & snx_i2c_mcu_test.c

2.3.2.1 Kernel Space

- int i2c_transfer(struct i2c_adapter *adap, struct i2c_msg *msgs, int num)
  - Parameter
    - adap:
      Handler to I2C bus
    - msgs:
      One or more messages to execute before STOP is issued to terminate the operation; each message begins with a START
    - num:
      Number of messages to be executed

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- return value
  - negative error, else the number of messages executed

Example:

```
static int ov9715_reg_read(struct i2c_client *client, u8 reg, u8 *val)
{
    int ret;
    u8 data = reg;
    struct i2c_msg msg = {
        .addr = client->addr,
        .flags= 0,
        .len   = 1,
        .buf   = &data,
    };
    ret = i2c_transfer(client->adapter, &msg, 1);
    if (ret < 0)
        goto err;
    msg.flags = I2C_M_RD;
    ret = i2c_transfer(client->adapter, &msg, 1);
    if (ret < 0)
        goto err;
    *val = data;
    return 0;
err:
    return ret;
}
```

2.3.2.2 User Space

- I2C_RDWR
  - Usage:

    ioctl(fd, I2C_RDWR, struct i2c_rdwr_ioctl_data * ioctl_data);

  - Parameters:

Element	Type	Description
nmsgs	u32	Number of i2c_msgs
msgs	struct i2c_msg	Pointers to i2c_msgs

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Table 06. The Description of i2c_rdwr_ioctl_data

Element	Type	Description
addr	u16	Slave address
flags	u16	Flags == 0 means write operations Flags == 1 means read operations
len	u16	Msg length (bytes)
buf	u8*	Pointer to msg data

Table 07. The Description of i2c_msg

- Return value:

  Error with negative value.

Example

```
int snx_i2c_write(int fd, int chip_addr, int addr, int data)
{
    struct i2c_msg msgs[1];
    struct i2c_rdwr_ioctl_data ioctl_data;
    int ret;
    __u8 buf[2];

    buf[0] = addr;
    buf[1] = data;
    msgs[0].addr = chip_addr;
    msgs[0].flags = 0;                  //Write Operation
    msgs[0].len = 2;
    msgs[0].buf = buf;
    ioctl_data.nmsgs = 1;
    ioctl_data.msgs = msgs;
    ret = ioctl(fd, I2C_RDWR, &ioctl_data);
    if (ret < 0) {
        printf("%s: ioctl return: %d\n", __func__, ret);
    }
    return ret;
}
```

2.3.3 Related Files

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- kernel/linux-2.6.35.12/src/include/linux/i2c.h
- kernel/linux-2.6.35.12/src/drivers/i2c/busses/i2c-snx.c
- kernel/linux-2.6.35.12/src/drivers/i2c/ i2c-core.c
- kernel/linux-2.6.35.12/src/arch/arm/mach-sn986xx/devices.c
- app/example/i2c/ov9715.c

2.4 SPI

SPI (Serial Peripheral Interface) bus is a synchronous serial data link *de fecto* standard with 4-wire serial bus. The frame format of SN986 Series SPI complies with Motorola. It can operate with a single master device or with one/more slave devices.
SN98600 SPI interface consists of one instance of the SPI controller. SN98610 SPI interface consists of two instances of the SPI controller.

The SPI features include:

- Supports SPI master / SPI slave / GPIO mode
- Supports read/write/write_read operation
- Programmable SPI mode (0/1/2/3)
- Programmable bits per word
- Programmable max speed
- Programmable serial bit data sequence (MSB or LSB first)

2.4.1 Function Description

Function Name	Category	Description
SPI_IOC_RD_MODE	ioctl	Get the current SPI mode
SPI_IOC_WR_MODE	ioctl	Set SPI mode
SPI_IOC_RD_LSB_FIRST	ioctl	Get the transmission setting (LSB first or not)
SPI_IOC_WR_LSB_FIRST	ioctl	Set LSB first or not in transmission
SPI_IOC_RD_BITS_PER_WORD	ioctl	Get how many bits per word for this device
SPI_IOC_WR_BITS_PER_WORD	ioctl	Set how many bits per word for this device
SPI_IOC_RD_MAX_SPEED_HZ	ioctl	Get the max speed of the spi device
SPI_IOC_WR_MAX_SPEED_HZ	ioctl	Set the max speed of the spi device
SPI_IOC_MESSAGE(num)	ioctl	SPI operation (num: numbers of the transmissions)
read	io	Read operation to SPI device
write	io	Write operation to SPI device

Table 08. SPI Function List

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

2.4.2 API Usage and Example

- SPI_IOC_RD_MODE / SPI_IOC_WR_MODE
  - Usage:

    ioctl(fd, SPI_IOC_(RD/WR)_ MODE, u8 * mode);

  - Parameters:

    mode: The value of the SPI mode.

  - Return value:

    Error with negative value.

- SPI_IOC_RD_LSB_FIRST / SPI_IOC_WR_LSB_FIRST
  - Usage:

    ioctl(fd, SPI_IOC_(RD/WR)_LSB_FIRST, u8* lsb);

  - Parameters:

    lsb: The value of the transmission setting.

  - Return value:

    Error with negative value.

- SPI_IOC_RD_BITS_PER_WORD / SPI_IOC_WR_BITS_PER_WORD
  - Usage:

    ioctl(fd, SPI_IOC_(RD/WR)_ BITS_PER_WORD, u8 * bits);

  - Parameters:

    bits: The value of the bits per word in transmission.

  - Return value:

    Error with negative value.

- SPI_IOC_RD_MAX_SPEED_HZ / SPI_IOC_WR_MAX_SPEED_HZ
  - Usage:

    ioctl(fd, SPI_IOC_(RD/WR)_ MAX_SPEED_HZ, u32* speed);

  - Parameters:

    speed: The value of the max transmission speed of the device.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- Return value:
  Error with negative value.

■ SPI_IOC_MESSAGE(num)

- Usage:
  ioctl(fd, SPI_IOC_ MESSAGE, struct spi_ioc_transfer *xfer);

- Parameters:

  The description of Structure spi_ioc_transfer:

Element	Type	Description
tx_buf	u64	TX buffer
rx_buf	u64	RX buffer
len	u32	The length of buffer
speed_hz	u32	The speed hz of transmission
delay_usecs	u16	Delay between the transmissions
bits_per_word	u8	Bits per word
cs_change	u8	if change chip select
pad	u32	

Table 09. The Description of Struct spi_ioc_transfer

- Return value:
  Error with negative value.

SN986 Serial SDK provides simple examples to describe how to use SPI in user space. Please refer to app/example/src/spi/ snx_spi_ctl.c

Example:

```
fd = open("/dev/spidev0.0", O_RDWR);                  // open spi1 first
ioctl (fd, SPI_IOC_RD_MODE, &mode );                  // get SPI mode
ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits);           // get Bits per word
ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);           // get max speed hz

struct spi_ioc_transfer *xfer;                        /* read data via SPI */
xfer->len = SPI_DATA_NUM;
xfer->rx_buf = (unsigned long) buf ;
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
xfer->speed_hz = speed;
xfer->bits_per_word = bits;
ioctl ( fd , SPI_IOC_MESSAGE(1), xfer );

struct spi_ioc_transfer *xfer;                    /* Write data via SPI */
xfer->len = SPI_DATA_NUM;
xfer->tx_buf = (unsigned long) buf ;
xfer->speed_hz = speed;
xfer->bits_per_word = bits;
ioctl ( fd , SPI_IOC_MESSAGE(1), xfer );
```

2.4.3 Related Files

- kernel/linux-2.6.35.12/ src/drivers/spi/snx_spi.c
- kernel/linux-2.6.35.12/ src/drivers/spi/spidev.c

2.5 UART

SN986 Serial provides 2 UART instances.

- Supports baud rates from 907 bps up to 115.2 Kbps
- Supports 1 start bit, 8-bit data, and 1 or 2 stop bit format
- Support 32 bytes TX and 32 bytes RX FIFO buffer
- Support TX and RX ready interrupt
- Support APB DMA with 8/16/32-bit data width
- Support APB DMA burst 4 mode
- Support GPIO mode
- Without RX Timeout Interrupt

2.5.1 Function Description

None

2.5.2 API Usage and Example

None

2.5.3 Related Files

- kernel/linux-2.6.35.12/src/drivers/serial/ snx_uart.c
- kernel/linux-2.6.35.12/src/arch/arm/mach-sn986xx/include/mach/ regs-serial.h

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

2.6 PWM

There are 2 PWM outputs, and the pins are shared with GPIO.

Features:

- Supports to control Iris, IR LED and status LED.
- The formula of PWM Period TPeriod is
  - Tbase = PCLK x (PWM_CK_DIV + 1);
  - TPeriod = Tbase x (PWM_PERIOD + 1)
  - Tduty = Tbase x PWM_DUTY
  - Range of PWM_CK_DIV is from 0 ~ 255
  - Range of PWM_PERIOD is from 0 ~ 1048575
  - Range of PWM_DUTY is from 0 ~ 1048575

2.6.1 Function Description

pwm device is “ dev/pwm”

Function Name	Category	Description
SONIX_PWM_REQUEST	ioctl cmd	equest a usable pwm device for pwm fd, the arg must be pwm_ID_1 or pwm_ID_2. these pwm_ID is defined in the struct of SNX_PWM_ID
SONIX_PWM_FREE	ioctl cmd	free a usable pwm device
SONIX_PWM_ENABLE	ioctl cmd	Enable pwm device
SONIX_PWM_DISABLE	ioctl cmd	disable pwm device
SONIX_PWM_READ	ioctl cmd	use pwm read mode and read the value
SONIX_PWM_INVERSE	ioctl cmd	inverse the square wave of pwm output
SONIX_PWM_CONFIG	ioctl cmd	config pwm device duty and period time, the arg is struct of pwm_config_param, (defined in snx_pwm.h)

Table 10. PWM Function List

2.6.2 API Usage and Example

SN986 Serial SDK provides two simple examples to describe how to use pwm device. Please refer to “app/example/pwm/ snx_pwm_period_test.c”,
and ”app/example/pwm/snx_pwm_ap_test.c”.

The snx_pwm_period_test.c is a simple program to set pwm period time and duty time, and

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

the snx_pwm_ap_test.c includes some applications of pwm, such as the period / duty time inverse…etc..
For the detail usage, please execute the examples by the commands,
“snx_pwm_period_test -h” or “snx_pwm_ap_test -h”

Example:

```
fd = open(PWM_DEVICE, O_RDWR, 0);
if(fd < 0)
{
        printf("open the file %s failed\n",PWM_DEVICE);
        return PWM_FAIL;
}
if (atoi(argv[1]) != 0 && atoi(argv[1]) != 1)
{
usage (argv[0]);
        return PWM_FAIL;
}
id = atoi(argv[1]);
if (atoi(argv[2]) > 0 && atoi(argv[3]) > 0)
{
if (atoi(argv[2]) >= atoi(argv[3]))
{
                pwm_param.duty_ns= atoi(argv[3]) * ONE_MSEC;
                pwm_param.period_ns= atoi(argv[2]) * ONE_MSEC;
}
else
{
                pwm_param.duty_ns= atoi(argv[2]) * ONE_MSEC;
                pwm_param.period_ns= atoi(argv[3]) * ONE_MSEC;
}
}
else
{
usage (argv[0]);
return PWM_FAIL;
}
if(id != SNX_PWM_1 && id != SNX_PWM_2){
        printf("id = %d\n", id);
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
        close(fd);
        return PWM_FAIL;
    }

    if(ioctl(fd, SONIX_PWM_REQUEST, &id))
    {
        printf("request failed\n");
        close(fd);
        return PWM_FAIL;
    }

    if(ioctl(fd, SONIX_PWM_DISABLE, &id))
    {
        printf("disable failed\n");
        close(fd);
        return PWM_FAIL;
    }

    if(ioctl(fd, SONIX_PWM_CONFIG, &pwm_param))
    {
        printf("config failed\n");
        close(fd);
        return PWM_FAIL;
    }

    if(ioctl(fd, SONIX_PWM_ENABLE, &id))
    {
        printf("enable failed\n");
        close(fd);
        return PWM_FAIL;
    }

    if(ioctl(fd, SONIX_PWM_FREE, &id))
    {
        printf("free failed\n");
        close(fd);
        return PWM_FAIL;
    }
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
if(close(fd))
{
    printf("close the file /dev/test_timer failed.\n");
    return PWM_FAIL;
}
```

2.6.3 Related Files

- driver/pwm/src/driver/pwm.c
- driver/pwm/src/driver/pwm.h
- app/example/pwm/ snx_pwm_ap_test.c
- app/example/pwm/ snx_pwm_period_test.c
- app/example/pwm/ snx_pwm.h

2.7 GPIO

General Purpose Input/Output (a.k.a. GPIO) is a common interface of SoC. Users can configure a GPIO pin as an input or output. When the GPIO pin is configured as an output, this pin can be set to high voltage or low voltage. When GPIO is configured as an input, this pin can be used to detect the voltage. Moreover, an input pin may be configured as an external interrupt source.

GPIO support

- GPIO direction configuration
- GPIO input/output value set /get
- GPIO interrupt request

2.7.1 Function Description

Function Name	Category	Description
/sys/class/gpio/export		export control of a GPIO to user space
/sys/class/gpio/unexport		unexport control of a GPIO to user space
/sys/class/gpio/gpioN/direction		Set/get gpio pin direction. Read/write as either “in” or “out”
/sys/class/gpio/gpioN/value		Set/get gpio value. Read/write as either 0 (low) or 1 (high)

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

/sys/class/gpio/gpioN/edge	set interrupt edge none/rasing/falling/both

Table 11. GPIO function list

2.7.2 API Usage and Example

SN986 Serial SDK provides a simple example to describe how to use gpio device. Please refer to app/example/gpio/ snx_gpio_test.c. The function, snx_gpio_write(), is used to set the gpio direction and output value.
For the detail usage please execute the example by the command, “snx_gpio_test -h”

Example:

```
int snx_gpio_write(int pin_number, int value)
{
    int fd, len;
    memset (buf,0,BUFFER_SIZE);
    sprintf(buf, "/sys/class/gpio/gpio%d/direction", pin_number);
    fd = open(buf, O_RDWR);
    if(fd < 0)
        return GPIO_FAIL;
    write(fd, "out", sizeof("out"));
    close(fd);

    sprintf(buf, "/sys/class/gpio/gpio%d/value", pin_number);
    fd = open(buf, O_RDWR);
    if(fd < 0)
        return GPIO_FAIL;
    len = sprintf(buf, "%d", value);
    write(fd, buf, len);
    close(fd);
    return GPIO_SUCCESS;
}
```

2.7.3 Related Files

- driver/gpio/src/driver/gpio.c
- driver/gpio/src/driver/gpio.h
- app/example/gpio/ snx_gpio_test.c

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

2.8 CRC16

SN986 Serial supports built-in CRC16. The controller is in Cryptography Engine register space.

The CRC16 features include,

- Built-in CRC-16 ModBus Codec
- CRC-16 with polynomial is $X^{16} + X^{15} + X^2 + 1$

2.8.1 Function Description

The CRC device node is “/ dev/crypto”

Function Name	Category	Description
SNX_INIT_BUF	ioctl cmd	Initial the buffer for CRC calcuation
SNX_CRC_CALCULATE	ioctl cmd	Enable the calculate

Table 12. CRC Function List

2.8.2 API Usage and Example

- SNX_INIT_BUF
  - Usage:

    ioctl(fd, SNX_INIT_BUF, struct snx_crypto_info* info);

  - Parameters:

Element	Type	Description
crypto_mode mod	enum	operation mode
data_len	u32	data length
key[6]	u32	crypto key (for AES/DES/3DES)

Table 13. Description of snx_crypto_info structure

  - Return value:

    Error with negative value.

- SNX_CRC_CALCULATE
  - Usage:

    ioctl(fd, SNX_CRC_CALCULATE, struct snx_crc_info * info);

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- Parameters:

Element	Type	Description
data_len	u32	data length
crc	u16	CRC result

Table 14. Description of snx_crc_info structure

- Return value:

  Error with negative value.

SN986 Serial SDK provides simple examples to describe how to use CRC in user space. Please refer to app/example/src/crypto/ snx_crypto_example.c

Example:

```
/* open crypto device */
crypto_fd = open ("/dev/crypto", O_RDWR, 0);
info.mode = snx_crypto_conf->mode;
info.data_len = snx_crypto_conf->data_len;
if (ioctl (crypto_fd, SNX_INIT_BUF, &info)) {          //Init crypto driver and allocate buffer
        printf ("ioctl(SNX_INIT_BUF) fail.\n");
        return -1;
}

mmap_size = snx_crypto_conf->data_len;
data_in = mmap (NULL, mmap_size, PROT_READ |PROT_WRITE, MAP_SHARED, crypto_fd, 0);

/* Copy the input data to the input buffer */
memcpy (data_in, snx_crypto_conf->buffer_in, snx_crypto_conf->data_len);
crc_info.data_len = snx_crypto_conf->data_len;

ioctl(crypto_fd, SNX_CRC_CALCULATE, &crc_info)

/* Copy the CRC result */
memcpy (snx_crypto_conf->buffer_out, &crc_info.crc, sizeof(unsigned short));
```

2.8.3 Related Files

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Confidential for Tozar

- driver/crypto/src/driver/snx_crypto.c
- driver/crypto/src/driver /snx_crypto.h
- driver/crypto/src/driver /snx_crypto_regs.h

2.9 AES/3DES/DES

SN986 Serial supports cryptography engine, including AES, DES and 3DES operations.
Features:

- AES
  - Built-in AES Codec
  - Cipher key only with lengths of 128 bits (AES-128)
  - Supports ECB mode
  - Supports DMA function
- DES/3DES
  - Built-in DES/3DES Codec
  - Supports ECB mode
  - Supports DMA function

2.9.1 Function Description

AES/3DES/DES device is “/dev/ snx_cipher”

Function Name	Category	Description
SNX_INIT_BUF	Ioctl cmd	Initial buffer
SNX_CRYPTO_CRYPT	Ioctl cmd	Enable the encrypt/decrypt, use struct snx_cipher_info (define in snx_cipher.h)

Table 15. AES/DES/3DES Function List

2.9.2 API Usage and Example

- SNX_INIT_BUF
  - Usage:

    ioctl(fd, SNX_INIT_BUF, struct snx_crypto_info* info);

  - Parameters:

Element	Type	Description
crypto_mode mod	enum	operation mode

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

data_len	u32	data length
key[6]	u32	crypto key (for AES/DES/3DES)

Table 16. Description of snx_crypto_info structure

- Return value:
  Error with negative value.

■ SNX_CRYPTO_CRYPT

- Usage:
  ioctl(fd, SNX_CRYPTO_CRYPT, struct snx_crypto_info* info);

- Parameters:

Element	Type	Description
crypto_mode mod	enum	operation mode
data_len	u32	data length
key[6]	u32	crypto key (for AES/DES/3DES)

Table 17. Description of snx_crypto_info structure

- Return value:
  Error with negative value.

In SN986 Serial SDK, we provide simple examples to describe how to use AES/DES/3DES in user space. Please refer to app/example/src/crypto/ snx_crypto_example.c

Example:

```
crypto_fd = open ("/dev/crypto", O_RDWR, 0);
info.mode = snx_crypto_conf->mode;
info.data_len = snx_crypto_conf->data_len;
ioctl (crypto_fd, SNX_INIT_BUF, &info)

/* in AES/DES/3DES, two driver buffers would be allocated, one is for input data, the other is for the result */
mmap_size = 2 * snx_crypto_conf->data_len;

data_in = mmap (NULL, mmap_size, PROT_READ |PROT_WRITE, MAP_SHARED, crypto_fd, 0);

data_out = data_in + snx_crypto_conf->data_len;
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
/* Copy the input data to the input buffer */
memcpy ((unsigned char*)info.key, snx_crypto_conf->key, KEY_SIZE);
memcpy (data_in, snx_crypto_conf->buffer_in, snx_crypto_conf->data_len);
info.mode = snx_crypto_conf->op;
info.data_len = snx_crypto_conf->data_len;

ioctl(crypto_fd, SNX_CRYPTO_CRYPT, &info)
/* Copy the result to the output buffer */
memcpy (snx_crypto_conf->buffer_out, data_out, snx_crypto_conf->data_len);
```

2.9.3 Related Files

- driver/crypto/src/driver/snx_crypto.c
- driver/crypto/src/driver /snx_crypto.h
- driver/crypto/src/driver /snx_crypto_regs.h

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

2.10 RTC

SN986 Serial has a built-in real-time clock (RTC), which is a flexible, low area, and low power real time clock. The RTC provides the second, minute, hour, and day counters separately. The RTC also provides a programmable auto-alarm function. When the second-auto-alarm function was turned on, the interrupt would be triggered at each second, and the automatic minute/hour alarm would be turned on as well.

Features:

- 30-bit timer, once per second, so count range is 0 ~ 1073741823 second, year, month, date, hours, minutes, and seconds used in F/W transfer.
- 30-bit alarm timer, once per second, supports interrupts and flag.
- 24-bit wakeup timer, once per 1/128's second, supports interrupts and flag

2.10.1 Function Description

RTC device is "/dev/rtc"

Function Name	Category	Description
RTC_SET_TIME	ioctl cmd	set rtc time counter
RTC_RD_TIME	ioctl cmd	get rtc time counter
RTC_ALM_SET	ioctl cmd	set alarm time counter
RTC_ALM_READ	ioctl cmd	get alarm time counter which set by RTC_ALM_SET
RTC_WKALM_SET	ioctl cmd	set wake up time counter
RTC_WKALM_RD	ioctl cmd	get wake up time counter which set by RTC_WKALM_SET
RTC_AIE_ON	ioctl cmd	Enable alarm or wake up interrupt
RTC_AIE_OFF	ioctl cmd	disable alarm or wake up interrupt

Table 18. RTC Function List

2.10.2 API Usage and Example

SN986 Serial SDK provides a simple example to describe how to use RTC. Please refer to "app/example/rtc/ snx_rtc_test.c"

In the example, "snx_rtc_wakeup_test ()" is used to set the RTC timer counter and get current time.

For the detail usage, please execute the example by the command, "snx_rtc_test -h"

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

Example:

```
int snx_rtc_timer_test ()
{
    int ret, i;
    int fd;
    struct rtc_time tm;
    memset (&tm, 0, sizeof (tm));
    tm.tm_year = (2013 - 1900); // set year = tm_year + 1900
    tm.tm_mon = (12 - 1);       // set mon = tm_mon + 1
    tm.tm_mday = 10;
    tm.tm_hour = 10;
    tm.tm_min = 10;
    tm.tm_sec = 50;
    fd = open (RTC_DEVICE, O_RDONLY);
    if (!fd) {
        printf ("Open Failed!\n");
        return RTC_FAIL;
    }
    if (ioctl (fd, RTC_SET_TIME, &tm))
    {
        printf ("RTC_SET_TIME fail\n");
        return RTC_FAIL;
    }
    // sleep 10s   so output = 2013-12-10 10:11:00
    if (ioctl (fd, RTC_RD_TIME, &tm))
    {
        return RTC_FAIL;
    }
    printf ("now set time => (%04d-%02d-%02d %02d:%02d:%02d)\n",
        tm.tm_year + 1900, tm.tm_mon + 1 , tm.tm_mday,
        tm.tm_hour, tm.tm_min, tm.tm_sec);
    for (i = 0; i < 10; i++) {
        sleep (1);
        printf (".");
        fflush (stdout);
    }
    // sleep 10s   so output = 2013-12-10 10:11:00
```

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

```
    if (ioctl (fd, RTC_RD_TIME, &tm))
    {
        return RTC_FAIL;
    }
    printf ("\nafter 10s, now get time => (%04d-%02d-%02d %02d:%02d:%02d)\n",
        tm.tm_year + 1900, tm.tm_mon + 1 , tm.tm_mday,
        tm.tm_hour, tm.tm_min, tm.tm_sec);
    close (fd);
    return RTC_SUCCESS;

}
```

2.10.3 Related Files

- driver/rtc/src/driver/regs-rtc.h
- driver/rtc/src/driver/ rtc.c
- app/example/rtc/ snx_rtc_test.c

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

3. Sensor & ISP

SONiX SN986 Serial Sensor and ISP functions are supported by SONiX SN986 Serial Middleware and Turnkey library. SONiX Middleware and Turnkey library provide full-function APIs of sensor and ISP for customers.
Application can capture image using standard v4l2 interface. Isp's function interface is exported via standard proc file system. Through the v4l2 interface, users can configure the image format, size, frame rate, scaling and data buffer mode; through the proc interface the user can configure function module of the ISP, including OSD, Mirror, Flip, Private Mask, Motion Detection, hue, brightness, contrast, etc.

3.1 Function Description

Please refer to "*SN986 Serial Sensor_ISP Programming Guide*" for the detail information.

3.2 API Usage and Example

Please refer to "*SN986 Serial Sensor_ISP Programming Guide*" for the detail information.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

4. Video Engine

4.1 Video Codec

Video codec and ISP are based on Video for Linux 2 (V4L2) memory to memory architecture, Video for Linux 2 use videobuf2 memory allocate method, Sonix Codec driver support videobuf2 DMA contiguous memory allocation (vb2-dma-conting) method and Sonix memory allocation method (snx-vb2). The driver stack diagram as follow. Video codec driver includes H264, MJPEG, Scale down and Codec Data Stamp.

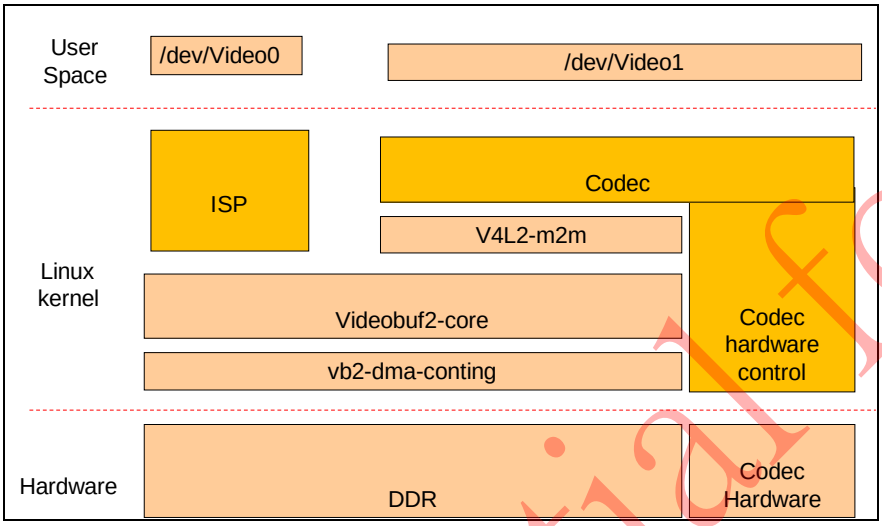

Figure 02. Video Codec connection diagram

4.1.1 Function Description

Please refer to "*SN986 Serial Video Codec Programming Guide*" for the detail information.

4.1.2 API Usage and Example

Please refer to "*SN986 Serial Video Codec Programming Guide*" for the detail information.

4.2 Video Scaling and Time Stamp

SONiX Middleware and Turnkey library provide full-function APIs of the time stamp.

Scaling

- Scaling function build in codec driver
- Scaling function only support 1, 1/2 or 1/4 size.
- Scaling function only support one scale size output.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

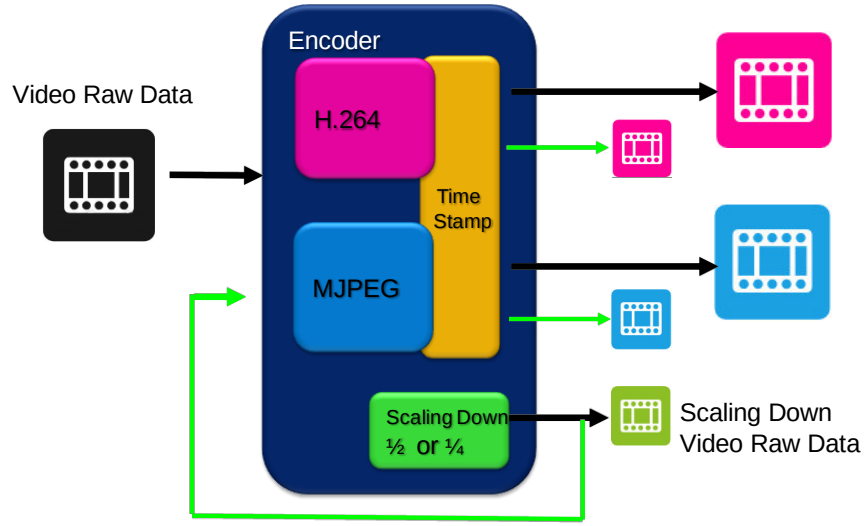

Figure 03. Video Scaling connection diagram

Time stamp

- Time stamp function build in codec driver
- Scaling down function

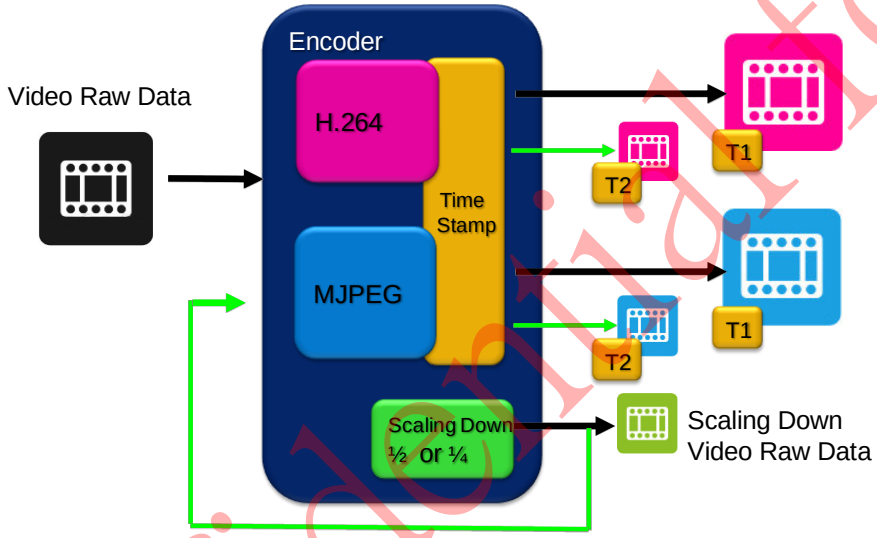

Figure 04. Time Stamp connection diagram

4.2.1 Function Description

Please refer to "*SN986 Serial Video Codec Programming Guide*" for the detail information.

4.2.2 API Usage and Example

Please refer to "*SN986 Serial Video Codec Programming Guide*" for the detail information.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

5. Video Output

5.1 OSD

SONiX Library provides the full-function APIs of OSD.
OSD can support 255 colors, applications use OSD function through interface of the framebuffer. The maximum image size of OSD is determined by video output working mode

5.1.1 Function Description

Please refer to "*SN986 Serial Video Output Programming Guide*" for the detail information.

5.1.2 API Usage and Example

Please refer to "*SN986 Serial Video Output Programming Guide*" for the detail information.

5.2 Video Output

SONiX Library provides the full-function APIs of Video Output, and the implementations are based on the interface of V4L2.

5.2.1 Function Description

Please refer to "*SN986 Serial Video Output Programming Guide*" for the detail information.

5.2.2 API Usage and Example

Please refer to "*SN986 Serial Video Output Programming Guide*" for the detail information.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

6. Audio

SN986 Series SDK provides audio codec driver and audio middleware. Audio codec supported formats are hardware audio codec (MS-ADPCM, A-LAW) and software audio codec (A-LAW, Mu-LAW, G.722, G.726). Audio codec supported sample rate are 8/16/24/32/44.1/48K.

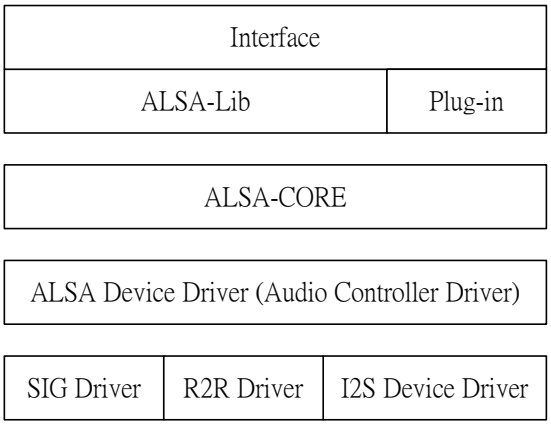

Figure 05. Audio Software Stack

6.1 Audio Codec Driver

Audio Codec Driver are based on Advanced Linux Sound Architecture (ALSA). Audio Codec driver includes audio controller driver, Sigma-Delta ADC driver, R2R DAC driver, AIC23 driver and tw2866 driver.

6.2 Audio Middleware

Audio Middleware includes ALSA library, software audio codec (A-LAW, Mu-LAW, G.722, G.726) plugin, sonix audio library and audio test tools。

6.3 Function Description

Please refer to "*SN986 Serial Audio Codec Programming Guide*" for the detail information.

6.4 API Usage and Example

Please refer to "*SN986 Serial Audio Codec Programming Guide*" for the detail information.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

7. Memory Interface

7.1 NAND Flash/Serial Flash

In SN986 Serial we support both NAND flash & Serial flash. It is both in MS1 (SD/NAND Flash/Serial Flash Controller).
The NAND/SF is compliant with AMBA 2.0 AHB master and slave interface.

7.1.1 Function Description

None

7.1.2 API Usage and Example

None

7.1.3 Related Files

- kernel/linux-2.6.35.12/src/drivers/mtd/nand/ snx_nand.c
- st58600\kernel/linux-2.6.35.12/src/include/linux/mtd/nand_reg.h
- kernel/linux-2.6.35.12/src/drivers/mtd/maps/mssf/ mssf.c
- kernel/linux-2.6.35.12/src/drivers/mtd/maps/mssf/ mssf.h
- kernel/linux-2.6.35.12/src/drivers/mtd/maps/mssf/ ms.c
- kernel/linux-2.6.35.12/src/drivers/mtd/maps/mssf/ ms.h

7.2 SD/SDHC

The SD/SDHC is compliant with AMBA 2.0 AHB master and slave interface and standard SDIO 2.0 specification. It also support multiple block read/write for large data by DMA transfer. The SD/SDHC includes CRC 16 for SPI/SD data and CRC7 for SPI/SD command in order to data verifying. For safety consideration, it support write-protect feature.

7.2.1 Function Description

CPU r/w mode

- Support command or data transfer by byte.

DMA r/w mode

- Support large data transfer in block based.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

- Support multiple blocks transfer.

CRC check

- In order to data transfer verify.

Card detect

- Support card detect interrupt feature.

Error handling

- Error interrupt flag for transfer checking.

7.2.2 API Usage and Example

None

7.2.3 Related Files

- driver/sdc/src/driver/sdcard/snx_ms.h
- driver/sdc/src/driver/sdcard/mmc.h
- driver/sdc/src/driver/sdcard/snx_mssd.h
- driver/sdc/src/driver/sdcard/snx_ms.c

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

8. USB

This USB20Host function contains not only a Universal Serial Bus (USB) 2.0 Host controller, but also build-in USB 2.0 Host PHY.
The USB 2.0 Host Controller provides a link between the AMBA on-chip bus (AHB) and the USB. The host controller supports High-, Full- and Low- Speed USB traffic.
USB 2.0 High-Speed functionality is supplied by an enhanced host controller implementing the Enhanced Host Controller Interface (EHCI). The companion Host Controller for Full- and Low-Speed functionalities are fully supported, but it is not implemented based on OHCI or UHCI standard. This is to reduce the gate count of the IP.
The USB 2.0 Host PHY consists the I/O pads, USB transceiver (T18U_LS15_AS_USB20HOSTPHY_A), PLL, and the digital part of the chip, UTMI_D.

8.1 Linux USB Sub-system

To understand all the Linux-USB framework, you'll use these resources:

* This source code. This is necessarily an evolving work, and includes kerneldoc that should help you get a current overview. ("make pdfdocs", and then look at "usb.pdf" for host side and "gadget.pdf" for peripheral side.) Also, Documentation/usb has more information.

* The USB 2.0 specification (from www.usb.org), with supplements such as those for USB OTG and the various device classes. The USB specification has a good overview chapter, and USB peripherals conform to the widely known "Chapter 9".

* Chip specifications for USB controllers. Examples include host controllers (on PCs, servers, and more); peripheral controllers (in devices with Linux firmware, like printers or cell phones); and hard-wired peripherals like Ethernet adapters.

* Specifications for other protocols implemented by USB peripheral functions. Some are vendor-specific; others are vendor-neutral but just standardized outside of the www.usb.org team.

Here is a list of what each subdirectory here is, and what is contained in them.

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

core/ - This is for the core USB host code, including the usbfs files and the hub class driver ("khubd").

host/ - This is for USB host controller drivers. This includes UHCI, OHCI, EHCI, and others that might be used with more specialized "embedded" systems.

gadget/ - This is for USB peripheral controller drivers and the various gadget drivers which talk to them.

Individual USB driver directories. A new driver should be added to the first subdirectory in the list below that it fits into.

image/ - This is for still image drivers, like scanners or digital cameras.
../input/ - This is for any driver that uses the input subsystem, like keyboard, mice, touchscreens, tablets, etc.
../media/ - This is for multimedia drivers, like video cameras, radios, and any other drivers that talk to the v4l subsystem.
../net/ - This is for network drivers.
serial/ - This is for USB to serial drivers.
storage/ - This is for USB mass-storage drivers.
class/ - This is for all USB device drivers that do not fit into any of the above categories, and work for a range of USB Class specified devices.
misc/ - This is for all USB device drivers that do not fit into any of the above categories.

8.2 Function Description

None

8.3 API Usage and Example

None

8.4 Related Files

(a) Programming Guide for Linux USB Device Drivers. Add File list and description of the

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

driver.

Programming Guide
for Linux USB Device

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.

9. Streaming Engine

9.1 SONiX Galaxy Streaming Sever

SONiX Galaxy is a full function streaming server for both video and audio in SONiX SN986 Serial SDK. Galaxy receives the live encoded video(from the hardware codec) and audio(from SONIX SN986 Serial middleware)as input, and then streams it through RTP over RTSP over UDP/TCP/HTTP. Users can connect to Galaxy and get video/audio stream through the IE browsers or the VLC Media players on the PC、mobile phone or the mobile pad.

For the detail function list and descriptions, please refer to "*SN986 Serial Galaxy Streaming Server Application Note*".

9.2 ONVIF Framework

For the detail function list and descriptions, please refer to "*SN986 Serial ONVIF Application Note*".

Confidential for Tozar

本資料為松翰科技股份有限公司專有之財產，未經書面同意不准透露、使用，亦不准複印或轉變成任何其他形式使用 The information contained herein is the exclusive property of SONIX and shall not be distributed, reproduced or disclosed in whole or no in part without prior written permission of SONIX.