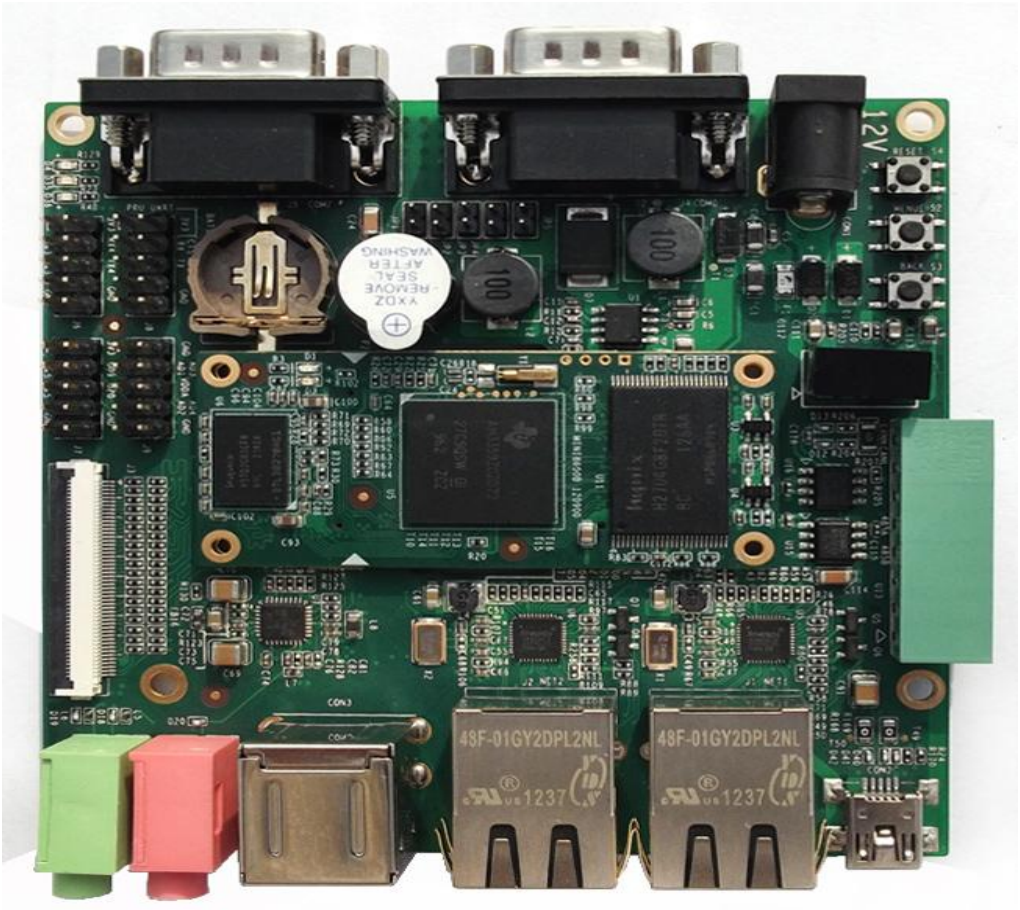
SOM860E

Single Board Computer



User Manual

Version:1.0

2019-07-12

Version History：

Version	Date	Description
1.0	2019-07-12	Initial

Table of Contents

1. Product Overview

1.1 Introduction

SOM-860-E is a high-performance system on module (SOM) based on highly integrated TI AM335X family of products, with a small form-factor just only 60*27mm ultra-small size. The tiny module integrates 2*256MBytes DDR3 SDRAM and 8G bytes eMMC; It uses two 0.4mm pitch 40-pin board-to-board male extension connectors to bring out most hardware peripheral signals and GPIOs from the CPU, It can be widely used in HMI, Medical appliances, Industrial automation.

SBC is a single board computer designed to carry the SOM-860-E core board. The flexible design allows the fast and easy way of realizing and upgrading the controller's capabilities. The SBC extension board has 5 serial ports (including 2 RS232 and 3 TTL), 2 USB Hosts and 1 USB OTG, 2 Ethernet ports, CAN, RS485, LCD, Touch screen, Audio, ADC and more other peripherals. It is a ready-to-run platform to support with Linux 4.1, Android 2.3 and WinCE 7 operating systems.

The attached development documentations include user manual, Schematic, Linux 4.1, source code of BSP, these enable users to shorten the time for development and launch their products to market rapidly.

1.2 Resource Download

You can access to our remote server to download the hardware and software resources through 2 ways below:

① Open web browser (Firefox is recommended), input the following url:

https://47.107.111.205/svn/TI/SOM860E

If the browser reports a warning like "Potential Security Risk Ahead ", please

choose the **Advanced...** option and "**Accept the Risk and Continue**".

Access Authorization:

User Name : guest

Password : guest

② Run svn command under Linux operating system, such as Debian and U buntu:

svn co svn://47.107.111.205/TI/SOM860E

Note：

🕮 If the command not found, please install svn tools : apt-get update && apt-get install -y subversion

The svn program will ask for access authorization:

User Name : guest

Password : guest

1.3 Hardware features

1.3.1 SOM

Electric/Mechanical Parameter

- Working Temperature：0 °C~ 70°C
- Humidity Range: 20% - 90%
- Dimensions: 60mm x 27mm
- Input Voltage: 3.3V

Processor

- 1GHz ARM Cortex™-A8 32-Bit RISC Microprocessor
  - NEON™ SIMD Coprocessor
  - 32KB/32KB of L1 Instruction/Data Cache with Single-Error Detection (parity)
  - 256KB of L2 Cache with Error Correcting Code (ECC)

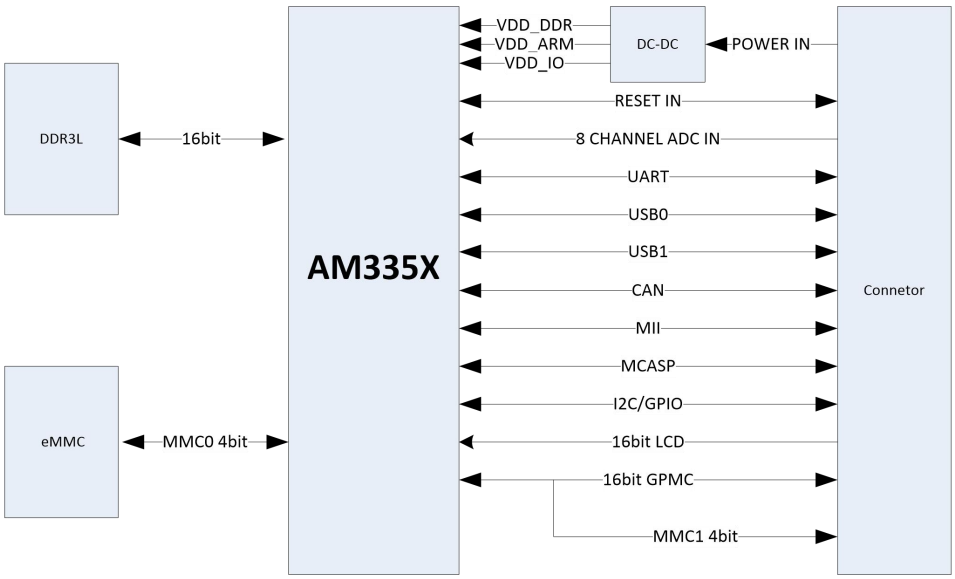
- SGX530 Graphics Engine
- Programmable Real-Time Unit Subsystem

Memory

- 8GByte eMMC
- 2*256MB DDR3 SDRAM

Supported Interfaces

- 24-bit TFT LCD
- 2 USB2.0 Host and 1 USB 2.0 OTG
- 6 UARTs
- 1 SPI
- 2 10/100/1000Mbps Ethernet
- 1 McASP
- 8 channel 12bit ADC Input
- 3 IIC bus
- 1 4-bit SDIO
- GPMC bus



SOM-860-E

1.3.2 SBC

Electric/Mechanical Parameter

- Working Temperature：0 °C~ 70°C
- Humidity Range: 20% - 90%
- Dimensions: 95mm x 95mm
- Input Voltage: 12V

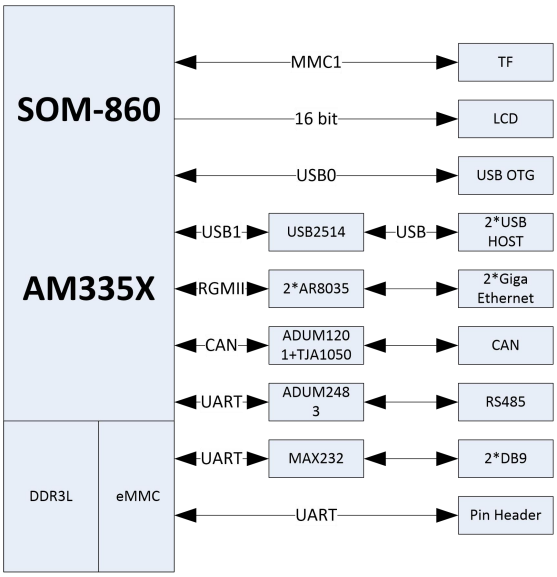
Audio/Video Interface

- 24-bit parallel LCD/4-Line resistive touch panel
- 3.5mm Audio input
- 3.5mm stereo audio output

Supported interfaced

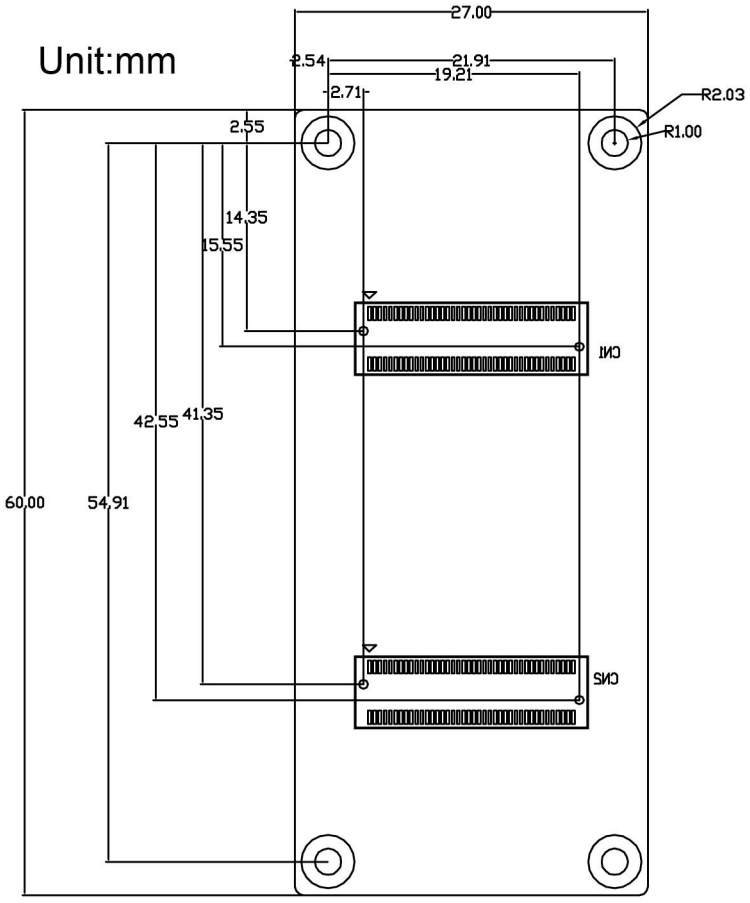
- 2 10/100/1000Mbps Ethernet
- 1 CAN bus
- 1 RS485 bus
- 1 USB OTG
- 2 USB HOST
- 1 TF Card
- UART
  - UART0，RS232，DB9 for debug
  - UART2，RS232，DB9
  - UART3， TTL，Pin Header
  - UART4， TTL，Pin Header
  - UART5， TTL，Pin Header
- GPIO
- 2 user button
- 1 Reset button
- 1 Beep
- 1 power LED

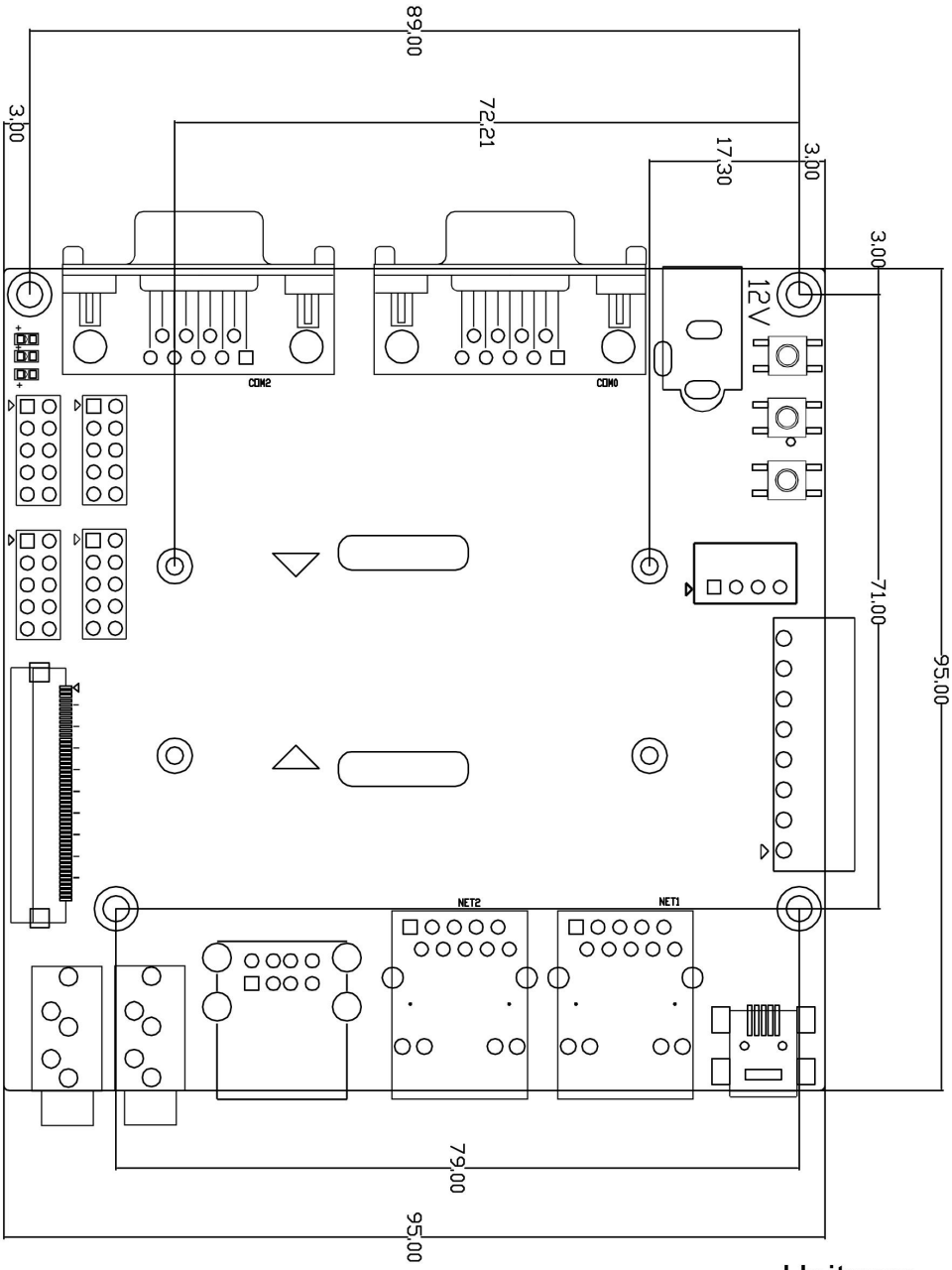
- 2 user LED



SBC-860

1.4 Mechanical Demension





SBC8600B Bird's-eye View

Unit:mm

2. Linux Operation System

This chapter will give you a general map of the Linux software resources contained in the DVD-ROM provided along with the product, as well as detailed introduction to the process of Linux system development, drivers development, system update, functionality tests and application development examples

Note：

🕮 It is recommended referring to Appendix for details of Ubuntu Linux installation and learning about embedded Linux development technology before you get started.

2.1 Software Resources

The DVR-ROM provided along with the board contains demos, application examples, Linux source code and tools, helping you develop Linux applications and systems easily and quickly.

2.1.1 Locations of Resources

You can find software resources such as programs and codes contained in the DVD-ROM according to the information showed in the table below;

Categories	Location
Applications	CD\Source\App\uart.tar.xz
Source Code	CD\Source\linux-am335x.tar.xz
	CD\Image\rootfs.tar.xz
	CD\Source\u-boot-am335x.tar.xz
Tools	CD\Tools
Precompiled Images	CD\Image

2.1.2 BSP

The following table lists types and formats of the files contained in BSP;

Names		Note	Formats
Bootloader	SPL	MMC/SD	Source Code
		FAT	Source Code
	u-boot	MMC/SD	Source Code
		FAT	Source Code
		NET	Source Code
Kernel	linux-4.1	Support ROM/CRAM/EXT4 /FAT/NFS/JFFS2/UBIFS various of file system	Source Code
Device Driver	Serial	Serials driver	Source Code
	RTC	Hardware RTC driver	Source Code
	NET	10/100M/1000M Ethernet driver	Source Code
	CAN	CAN bus driver	Source Code
	SPI	SPI driver	Source Code
	I2C	I2C driver	Source Code
	LCD	TFT LCD driver	Source Code
	Touch Screen	4-line Resistive touch panel driver	Source Code
	ADC	4 channels ADC input	Source Code
	MMC/SD	MMC/SD controller driver	Source Code
	USB OTG	USB OTG driver	Source Code
	Audio	Audio driver (supports audio recording and playback)	Source Code
	Keypad	GPIO keypad driver	Source Code
	LED	LED driver	Source Code
	UVC Camera	USB camera driver	Source Code
	VGA	VGA8000-A module driver	Source Code
	WIFI	USB WIFI dongle driver	Source Code
	Bluetooth	USB Bluetooth driver	Source Code
Ramdisk	BusyBox	Simplified rootfs based on busybox	Image
Rootfs	Debian 8	Simplified, No GUI	Image

2.2 Structure of Embedded Linux System

SOM-860-E has a Linux-4.1 system in on-board eMMC by default. This system supports 4.3-inch touch-screen and consists of spl（MLO）, U-boot, kernel and rootfs. The following figure shows the structure of embedded Linux system.：

1) SPL is the first level bootstrap program. After the system starts up, the ROM inside the CPU will copy spl to internal RAM and perform its routine work. Its main function is to initialize the CPU, copy u-boot into the memory and let u-boot lead the booting process；

2) U-boot is the second-level boot loader. It is used to interact with users, update images and boot the kernel；

3) The kernel used in this document is Linux 4.1 and has customizable based on the hardware；

4) Ramdisk, based on busybox, mainly used for system updating；

5) Rootfs adopts open-source system Debian8 with EXT4 format.

2.3 Building Development Environment

Before the software development, users have to establish a Linux cross development environment on PC. This section will take Ubuntu operating system as an example to introduce how to establish a cross development environment.

It is strongly recommended to install necessary software packages for a newly installed Ubuntu through the following commands.

- **sudo apt-get update; sudo apt-get install -y xz-utils ncurses-dev autoconf libtool automake texinfo bison flex libc6:i386 libncurses5:i386 libstdc++6:i386**

Note：

- Each instruction has been put a bullets "•" before it to prevent confusion caused by the long instructions that occupy more than one line in the context.
- Please note the SPACES within each instruction; Missing of any SPACE will cause failure when executing instructions.

2.3.1 Installing Cross Compilation Tools

Insert the DVD-ROM in your PC's DVD-ROM drive, Ubuntu system will automatically mount it under /media/cdrom. Execute the following instructions in the terminal window of Ubuntu to uncompress cross-compiling tools from /media/cdrom/linux/tools to $HOME；

- **mkdir $HOME/tools**
- **cd /media/cdrom/Tools**
- **tar -xvf gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz -C $HOME/tools**

2.3.2 Adding environment variables

After all above tools are installed, it is necessary to use the following commands to add them in the temporary environment variables:

- **export PATH=$HOME/tools/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin: $HOME/tools:$PATH**
- **export ARCH=arm**
- **export CROSS_COMPILE=arm-linux-**

Note：

- The instructions can be added in the .bashrc file located at the user directory, so that the addition of environment variables will be loaded automatically when the system is booting up;
- If you want to check the path, please use the instruction **echo $PATH**

2.4 Preparing the Source Code

This section will introduce two ways to get the source code：

2.4.1 Getting the Source Code from DVD

Execute the following instructions to uncompress source code from linux/source of the DVD-ROM to Ubuntu system：

- **mkdir $HOME/work**
- **cd $HOME/work**
- **tar -xvf /media/cdrom/Source/u-boot-am335x.tar.xz**
- **tar -xvf /media/cdrom/Source/linux-am335x.tar.xz**
- **mkdir rootfs**
- **sudo tar -xvf /media/cdrom/Image/rootfs.tar.xz -C rootfs**

When the above steps are completed, the directories u-boot-2015.07, Linux-4.1 and rootfs will be created under current directory.

Note：

Please make sure the uncompressed source code is saved under the directory specified in the above instructions, or errors might occur in compilation process.

2.4.2 Getting the source code on the Internet

Download source code and Image in the Internet by the following commands：

- **$ cd ~**
- **$ svn co svn://198.56.248.66/TI/SOM860E**

> **Note:**
> User is “guest” and password is also “guest”.

2.5 Compilation

1） Compiling Bootloader

SOM-860-E can boot up from TF card and eMMC，if the jumper JP5 is disconnect, the board will boot from eMMC, otherwise it will boot from TF card.

Please execute the following instructions to compile boot loader：

- **cd u-boot-am335x**
- **make distclean**
- **make am335x_som860e_defconfig**
- **make**

After all the instructions are executed, you can find booting images named MLO and u-boot.img under current directory.

2） Compiling Kernel

Execute the following instructions to compile kernel：

- **cd linux-am335x**
- **make distclean**
- **make am335x_som860e_defconfig**
- **make zImage am335x-som860e.dtb**

After all the instructions are executed, you can find a kernel image named zImage under arch/arm/boot and am335x-som860e.dtb under arch/arm/boot/dts

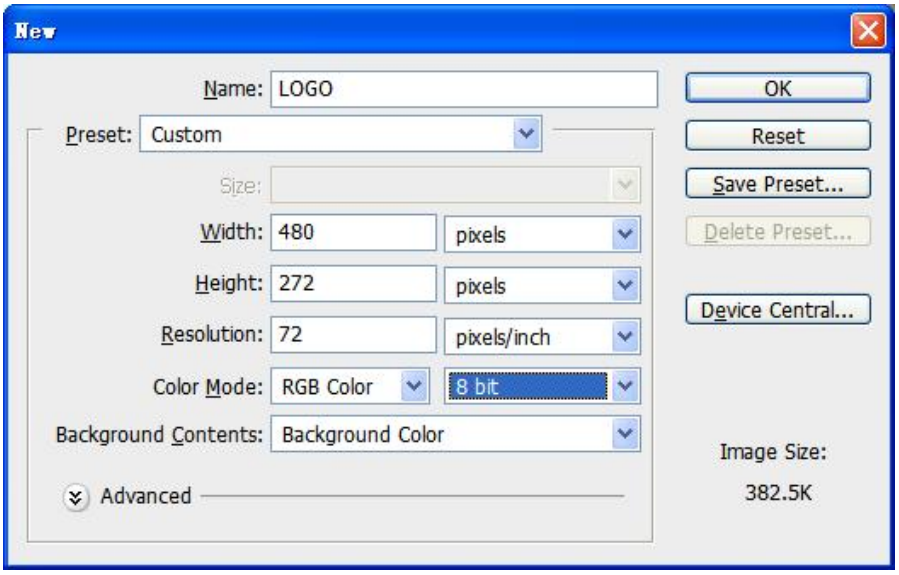
2.6 Linux System Customization

In order to satisfy different requirements of customers, designers commonly need to make some custom modification based on the default configuration of Linux kernel. This chapter will introduce the process of system customization by showing you some examples.

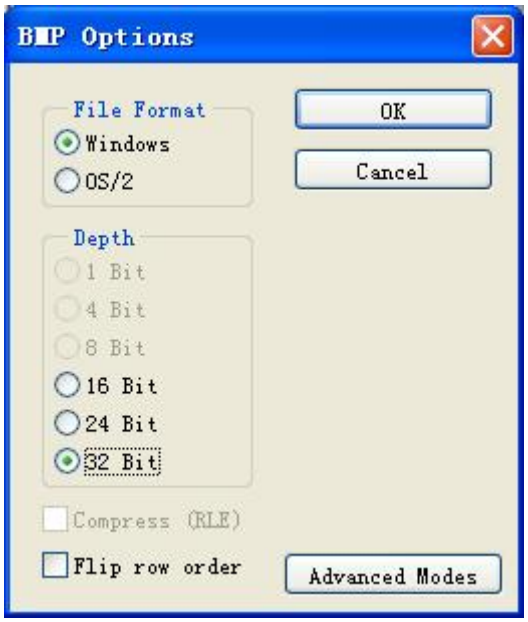
2.6.1 Generating U-Boot LOGO

How to Generate an U-Boot LOGO? take photoshop as example.

- Create a new image



- Save the image as logo.bmp with the setting dialog as below：



2.6.2 Setting Configuration Menu

A default configuration file is provided in the factory kernel source codes:

linux-am335x/arch/arm/configs/am335x_som860e_defconfig

Please execute the following instructions to enter the configuration menu:

- **cd linux-am335x**
- **make am335x_som860e_defconfig**
- **make menuconfig**

Note:

If an error occurs when command 'make menuconfig' is executed, you might need to install 'ncurse' in the Ubuntu system, 'ncurses' is a character graphic library required to generate configuration menu. Please enter the following instruction to install the library:

sudo apt-get install ncurses-dev

2.6.3 Menu Options

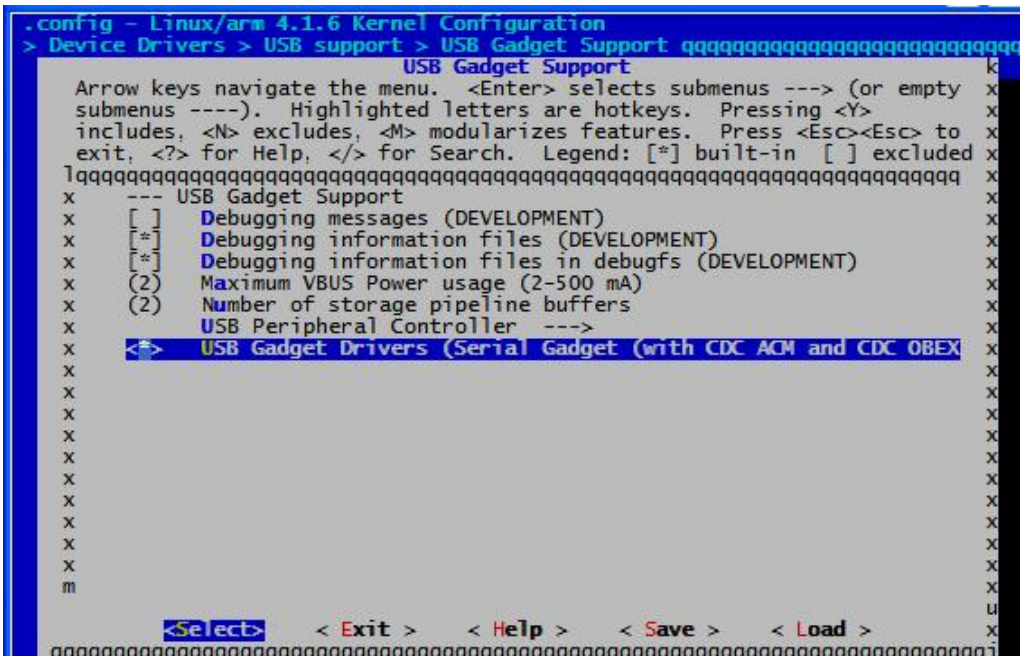
Configure options according to customization requirements after entering configuration menu, for example, access **Device Drivers** > **USB support** > **USB Gadget Support** > **USB Gadget Drivers** as shown below:

-> Device Drivers

-> USB support

-> USB Gadget Support

-> USB Gadget Drivers



Set USB Gadget Drivers（Serial Gadget.....） to **<*>**, and then exit and save changes.

2.6.4 Compiling Kernel

Please execute the following instructions to recompile kernel:

- **make am335x-som860e.dtb zImage**

After all the instructions are executed, you can get the new Image **arch/arm/boot/zImage.**

2.6.5 Testing Serial Gadget

Connect mini-USB port and PC USB port, the Gadget Serial v2.4 hardware device will be found in Windows Device Manager.

linux-cdc-acm.inf is in the directory linux-am335x/Documentation/usb/linux-cdc-acm.inf

Run serial tool (such as Hyper Terminal) on PC, open COM port specific to Gadget Serial device. Then access /dev/ttyGS0, it can communicate with PC terminal

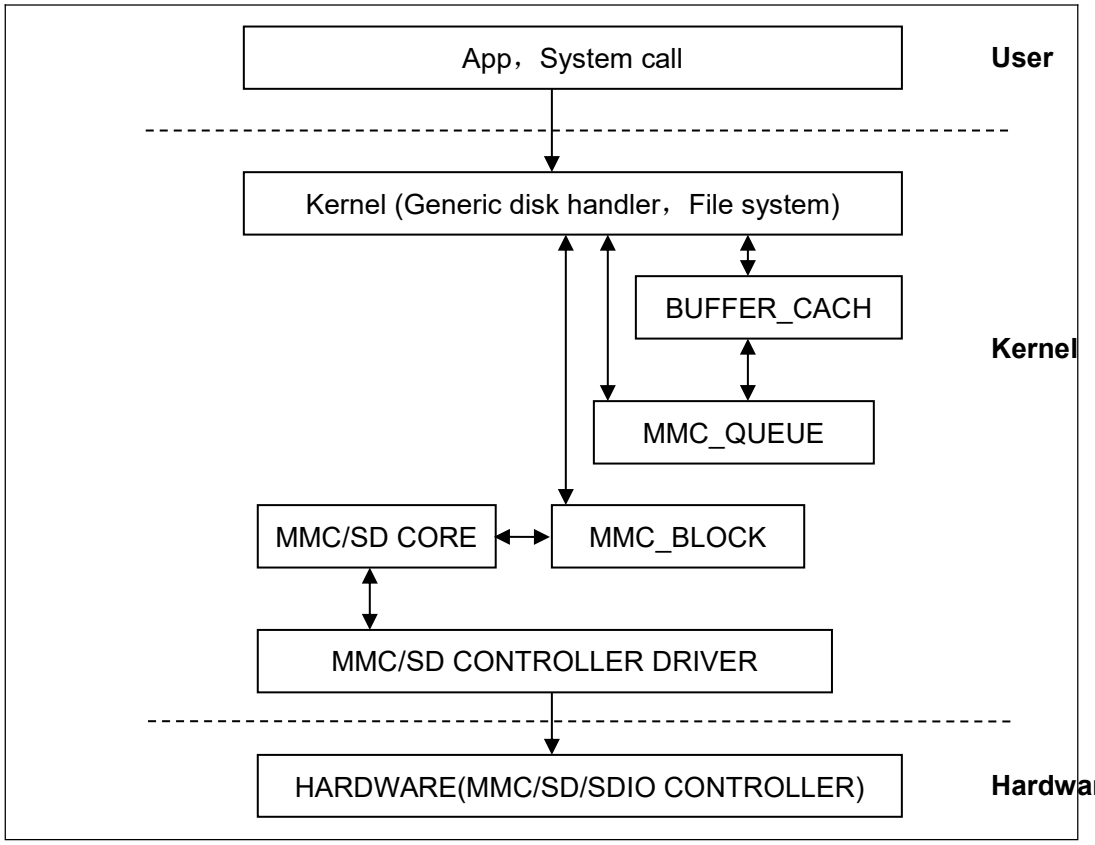
2.7 Introduction to Drivers

2.7.1 The table below shows the access path to find all the drivers :

Category	Name	Description	Location
Bootloader	SPL	MMC/SD	drivers/mmc/omap_hsmmc.c
		FAT	fs/
	u-boot	MMC/SD	drivers/mmc/omap_hsmmc.c
		FAT	fs/
		NET	drivers/net/cpsw.c
Kernel	Linux-4.1	ROM/CRAM/EXT4 /FAT/NFS/JFFS2/UBIFS filesystem	fs/
Devices	Serial	Serial driver	drivers/tty/serial/8250/8250_omap.c
	RTC	Hardware RTC driver	drivers/rtc/rtc-omap.c
	NET	10/100M/1000M Ethernet driver	drivers/net/ethernet/ti/ti_cpsw.c
	CAN	CAN bus driver	drivers/net/can/c_can/c_can_platform.c
	SPI	SPI driver	drivers/spi/spi-omap2-mcspi.c
	LCD	TFT LCD driver	drivers/gpu/drm/tilcdc/tilcdc_drv.c drivers/video/of_display_timing.c
	Touch Screen	4-line resistive touch panel driver	drivers/input/touchscreen/ti_tscadc.c
	ADC	4 channels ADC input	drivers/iio/adc/ti_am335x_adc.c
	MMC/SD	MMC/SD controller dirver	drivers/mmc/host/omap_hsmmc.c
	USB	USB controller dirver	drivers/usb/musb/musb_am335x.c
	Audio	Audio driver (supporting recording/playback)	sound/soc/codecs/sgtl5000.c
	Keypad	GPIO keypad driver	drivers/input/keyboard/gpio_keys.c
	LED	LED driver	drivers/leds/leds-gpio.c
	UVC Camera	USB camera driver	drivers/media/usb/uvc
	VGA	VGA8000-A module driver	drivers/video/of_display_timing.c
	WIFI	USB wifi dongle driver	drivers/net/wireless/rt2x00

	Bluetooth	USB bluetooth driver	drivers/bluetooth/btusb.c

2.7.2 SD/MMC



SD/MMC drivers in Linux are mainly consisted of SD/MMC core, mmc_block, mmc_queue and SD/MMC driver：

1） SD/MMC core realizes the codes unrelated to structure in the SD/MMC card operation;

2） mmc_block realizes driver structure when SD/MMC card is used as a block device;

3） mmc_queue realizes management of request queue;

4） SD/MMC driver realizes specific controller driver.

Drivers and relevant documents:

linux-am335x/drivers/mmc/

linux-am335x/drivers/mmc/host/omap_hsmmc.c

2.7.3 LCDC

The LCD controller (LCDC) of AM335x is the latest version integrated in OMAP-L138 SoC which has differences as follows comparing with OMAP-L138.

Different interrupt configuration and status register

2048*2048 Higher display resolution of up to 2048*2048

24-bit active TFT grating per pixel

So Linux LCD driver can be used to improve the LCD_VERSION2 code. By reading PID register, the update of LCDC version can be found.

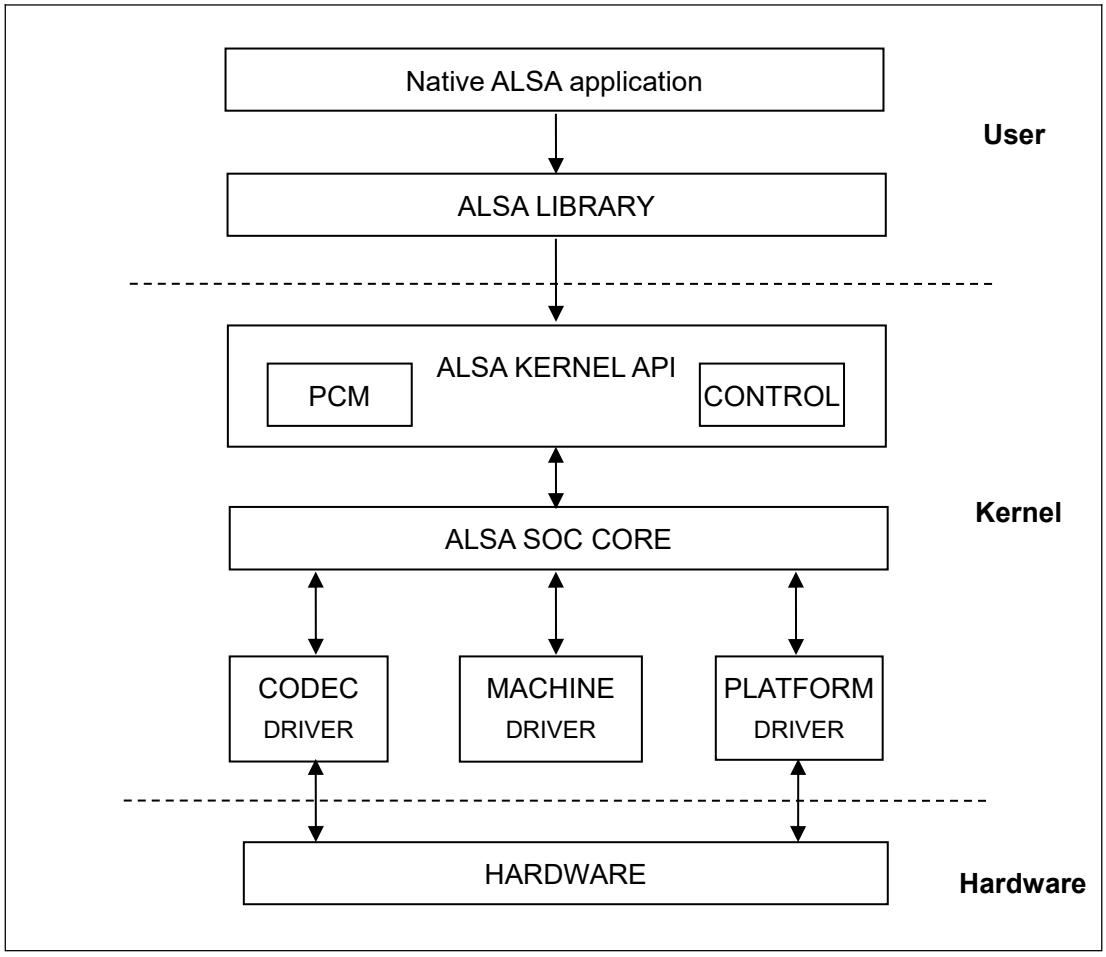
Drivers and relevant documents:

linux-am335x/drivers/video/

linux-am335x/drivers/gpu/drm/tilcdc/tilcdc_panel.c

linux-am335x/drivers/video/of_display_timing.c

2.7.4 Audio In/Out



ASoC embedded audio system basically consists of three components:

1) Codec driver: The codec driver is platform independent and contains audio controls, audio interface capabilities, codec dapm definition and codec IO functions.

2) Platform driver: It contains the audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM) of that platform.

3) Machine driver: The machine driver handles any machine specific controls and audio events i.e. turning on an amp at start of playback.

Drivers and relevant documents:

linux-am335x/sound/soc/

linux-am335x/sound/soc/davinci/davinci-evm.c

linux-am335x/sound/soc/codecs/sgtl5000.c

2.8 Driver development

2.8.1 GPIO_keys driver

1）Device Definition

linux-am335x/arch/arm/boot/dts/am335x-som860e.dts

Define gpio0.20 as “menu” key, return value as “KEY_F1”, triggered by low level; gpio2.1 as “back” key, return value as “KEY_ESC”, triggered by low level.

```
    gpio_keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&button_pins>;

        key@0 {
            label = "MENU";
            linux,code = <KEY_F1>;
            gpios = <&gpio0 20 GPIO_ACTIVE_LOW>;
            gpio-key,wakeup;
        };

        key@1 {
            label = "BACK";
            linux,code = <KEY_ESC>;
            gpios = <&gpio2 1 GPIO_ACTIVE_LOW>;
            gpio-key,wakeup;
        };
    };
```

2）GPIO pinmuxConfiguration

Define the GPIO0.20 and GPIO2.1 as MODE7 (GPIO mode) and AM33XX_PIN_INPUT (configuration input).

linux-am335x/arch/arm/boot/dts/am335x-som860e.dts

```
    button_pins: pinmux_button_pins {
        pinctrl-single,pins = <
            0x1B4 (PIN_INPUT_PULLUP | MUX_MODE7)     /* xdma_event_intr1.gpi
o0_20 */
```

```
                0x08C  (PIN_INPUT_PULLUP | MUX_MODE7)      /*  gpmc_clk.gpio2_1  */
        >;
    };
```

3）Driver Design

linux-am335x/drivers/input/keyboard/gpio_keys.c

a) Call platform_driver_register to register gpio_keys driver

```
static struct platform_driver gpio_keys_device_driver = {
        .probe          = gpio_keys_probe,
        .remove         = gpio_keys_remove,
        .driver         = {
                .name   = "gpio-keys",
                .pm     = &gpio_keys_pm_ops,
                .of_match_table = gpio_keys_of_match,
        }
};

static int __init gpio_keys_init(void)
{
        return platform_driver_register(&gpio_keys_device_driver);
}

static void __exit gpio_keys_exit(void)
{
        platform_driver_unregister(&gpio_keys_device_driver);
}

late_initcall(gpio_keys_init);
module_exit(gpio_keys_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Phil Blundell <pb@handhelds.org>");
MODULE_DESCRIPTION("Keyboard driver for GPIOs");
MODULE_ALIAS("platform:gpio-keys");
```

b) Call input_register_device to register input driver

```
static int __devinit gpio_keys_probe(struct platform_device *pdev)
{
        if (!pdata) {
                pdata = gpio_keys_get_devtree_pdata(dev);
                if (IS_ERR(pdata))
```

```
                return PTR_ERR(pdata);
        }
...
        input = devm_input_allocate_device(dev);
...
        for (i = 0; i < pdata->nbuttons; i++) {
                const struct gpio_keys_button *button = &pdata->buttons[i];
                struct gpio_button_data *bdata = &ddata->data[i];

                error = gpio_keys_setup_key(pdev, input, bdata, button);

                error = gpio_keys_setup_key(pdev, bdata, button);
                if (error)
                        return error;

                if (button->wakeup)
                        wakeup = 1;
        }
        error = sysfs_create_group(&pdev->dev.kobj, &gpio_keys_attr_group);
        if (error) {
                dev_err(dev, "Unable to export keys/switches, error: %d\n",
                        error);
                goto fail2;
        }

        error = input_register_device(input);
        if (error) {
                dev_err(dev, "Unable to register input device, error: %d\n",
                        error);
                goto err_remove_group;
        }
...
```

c) Apply for gpio and define the gpio as input, and register gpio interrupt.

```
static int gpio_keys_setup_key(struct platform_device *pdev,
                               struct input_dev *input,
                               struct gpio_button_data *bdata,
                               const struct gpio_keys_button *button)
{
        const char *desc = button->desc ? button->desc : "gpio_keys";
        struct device *dev = &pdev->dev;
        irq_handler_t isr;
        unsigned long irqflags;
```

```
int irq;
int error;

bdata->input = input;
bdata->button = button;
spin_lock_init(&bdata->lock);

if (gpio_is_valid(button->gpio)) {

        error = devm_gpio_request_one(&pdev->dev, button->gpio,
                                      GPIOF_IN, desc);
        if (error < 0) {
                dev_err(dev, "Failed to request GPIO %d, error %d\n",
                        button->gpio, error);
                return error;
        }

        if (button->debounce_interval) {
                error = gpio_set_debounce(button->gpio,
                                button->debounce_interval * 1000);
                /* use timer if gpiolib doesn't provide debounce */
                if (error < 0)
                        bdata->software_debounce =
                                        button->debounce_interval;
        }

        if (button->irq) {
                bdata->irq = button->irq;
        } else {
                irq = gpio_to_irq(button->gpio);
                if (irq < 0) {
                        error = irq;
                        dev_err(dev,
                                "Unable to get irq number for GPIO %d, error %d\n",
                                button->gpio, error);
                        return error;
                }
                bdata->irq = irq;
        }

        INIT_DELAYED_WORK(&bdata->work, gpio_keys_gpio_work_func);

        isr = gpio_keys_gpio_isr;
```

```
        irqflags = IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING;

} else {
        if (!button->irq) {
                dev_err(dev, "No IRQ specified\n");
                return -EINVAL;
        }
        bdata->irq = button->irq;

        if (button->type && button->type != EV_KEY) {
                dev_err(dev, "Only EV_KEY allowed for IRQ buttons.\n");
                return -EINVAL;
        }

        bdata->release_delay = button->debounce_interval;
        setup_timer(&bdata->release_timer,
                    gpio_keys_irq_timer, (unsigned long)bdata);

        isr = gpio_keys_irq_isr;
        irqflags = 0;
}

input_set_capability(input, button->type ?: EV_KEY, button->code);

/*
 * Install custom action to cancel release timer and
 * workqueue item.
 */
error = devm_add_action(&pdev->dev, gpio_keys_quiesce_key, bdata);
if (error) {
        dev_err(&pdev->dev,
                "failed to register quiesce action, error: %d\n",
                error);
        return error;
}

/*
 * If platform has specified that the button can be disabled,
 * we don't want it to share the interrupt line.
 */
if (!button->can_disable)
        irqflags |= IRQF_SHARED;
```

```
        error = devm_request_any_context_irq(&pdev->dev, bdata->irq,
                                             isr, irqflags, desc, bdata);
        if (error < 0) {
                dev_err(dev, "Unable to claim irq %d; error %d\n",
                        bdata->irq, error);
                return error;
        }

        return 0;
}
```

d) Interrupt processing

When button is pressed, an interrupt is generated and key value is reported.

```
static irqreturn_t gpio_keys_gpio_isr(int irq, void *dev_id)
{
           ...
           mod_delayed_work(system_wq,
                            &bdata->work,
                            msecs_to_jiffies(bdata->software_debounce));
           ...
}

static void gpio_keys_gpio_work_func(struct work_struct *work)
{
          ...
        gpio_keys_gpio_report_event(bdata);
          ...
}
static void gpio_keys_gpio_report_event(struct gpio_button_data *bdata)
{
        const struct gpio_keys_button *button = bdata->button;
        struct input_dev *input = bdata->input;
        unsigned int type = button->type ?: EV_KEY;
        int state = (gpio_get_value_cansleep(button->gpio) ? 1 : 0) ^ button->active_low;

        if (type == EV_ABS) {
                if (state)
                        input_event(input, type, button->code, button->value);
        } else {
                input_event(input, type, button->code, !!state);
        }
        input_sync(input);
}
```

2.8.2 GPIO_leds Driver

1）Device Definition

linux-am335x/arch/arm/boot/dts/am335x-som860e.dts

Configure GPIO2.5 as system indicator, lighting up by outputting high level.

```
leds {
        compatible = "gpio-leds";
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&user_leds_default>;
        pinctrl-1 = <&user_leds_sleep>;

        led@0 {
                label = "sys";
                gpios = <&gpio2 5 GPIO_ACTIVE_HIGH>;
                linux,default-trigger = "heartbeat";
                default-state = "off";
        };
};
```

2）GPIO pinmux Configuration

linux-am335x/arch/arm/boot/dts/am335x-som860e.dts

Config GPIO2.5as MODE7(gpiomode),AM33XX_PIN_OUTPUT（output mode）

```
user_leds_default: pinmux_user_leds_default {
        pinctrl-single,pins = <
                /* leds */
                0x09c       (PIN_OUTPUT_PULLUP       |       MUX_MODE7)                   /*
gpmc_be0n_cle.gpio2_5 */
};
```

3）Driver Design

linux-am335x/drivers/leds/leds-gpio.c

a) Call platform_driver_register to register gpio_leds driver

```
static struct platform_driver gpio_led_driver = {
        .probe          = gpio_led_probe,
        .remove         = gpio_led_remove,
        .driver         = {
                .name     = "leds-gpio",
                .of_match_table = of_gpio_leds_match,
                .pm = &gpio_led_pm_ops,
```

```
    },
};

module_platform_driver(gpio_led_driver);

MODULE_AUTHOR("Raphael Assenat <raph@8d.com>, Trent Piepho
<tpiepho@freescale.com>");
MODULE_DESCRIPTION("GPIO LED driver");
MODULE_LICENSE("GPL");
MODULE_ALIAS("platform:leds-gpio");
```

b) Apply for gpio and call led_classdev_register to led_classdev drivr.

```
static int gpio_led_probe(struct platform_device *pdev)
{
…
    if (pdata && pdata->num_leds) {
        priv = devm_kzalloc(&pdev->dev,
                sizeof_gpio_leds_priv(pdata->num_leds),
                    GFP_KERNEL);
        if (!priv)
            return -ENOMEM;

        priv->num_leds = pdata->num_leds;
        for (i = 0; i < priv->num_leds; i++) {
            ret = create_gpio_led(&pdata->leds[i],
                        &priv->leds[i],
                        &pdev->dev, pdata->gpio_blink_set);
            if (ret < 0) {
                /* On failure: unwind the led creations */
                for (i = i - 1; i >= 0; i--)
                    delete_gpio_led(&priv->leds[i]);
                return ret;
            }
        }
    } else {
        priv = gpio_leds_create(pdev);
        if (IS_ERR(priv))
            return PTR_ERR(priv);
    }

    platform_set_drvdata(pdev, priv);

    return 0;
```

```
}

static int create_gpio_led(const struct gpio_led *template,
        struct gpio_led_data *led_dat, struct device *parent,
        int (*blink_set)(struct gpio_desc *, int, unsigned long *,
                        unsigned long *))
{
…
                ret = devm_gpio_request_one(parent, template->gpio, flags, template->name);
…
                ret = gpiod_direction_output(led_dat->gpiod, state);
…
                return led_classdev_register(parent, &led_dat->cdev);
}
```

c) Users may access the file named brightness under /sys/class/leds/xxx/brightness, and call gpio_led_set to configure LED status

```
static void gpio_led_set(struct led_classdev *led_cdev,
        enum led_brightness value)
{
…
                        gpiod_set_value(led_dat->gpiod, level);
}
```

2.9 System Update

SOM-860-E can boot up from both TF card and eMMC, this section briefly introduce the process of system update on TF card and eMMC.

2.9.1 Update of TF Card System Image

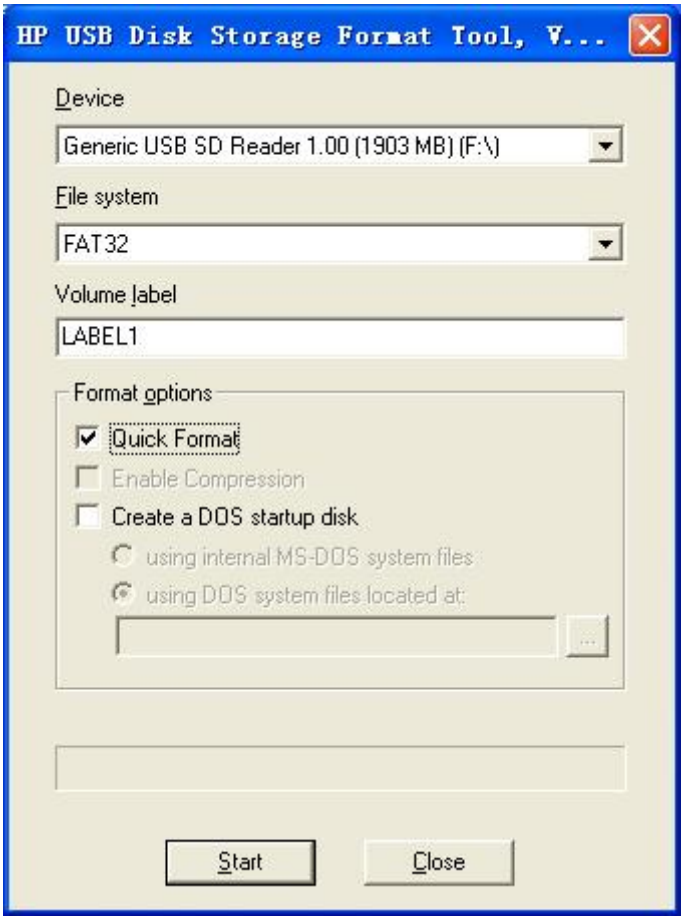
1） Formatting TF Card

HP USB Disk Storage Format Tool 2.0.6 is recommended as the formatting tool.

CD\Tools\SDFormatter_HP.zip。

a) Insert TF card into a card reader and then insert the reader into your PC

b) Open HP USB Disk Storage Format Tool to show the following window:



c) Select “FAT32” file system

d) Click “Start”

e) When formatting is complete, click “OK”

Note：

- It is not recommended to use other versions of HP USB Disk Storage Format Tool.
- HP USB Disk Storage Format Tool will erase the partitions of TF card.

2）Updating Image

Copy all the files under **CD/Image** to TF card，then insert TF card to card cage and short out the jumper JP5, then power on the board. The information in HyperTerminal window is shown below:

Note:

- The default display definition is 800*600. If you are working with the LCDs of other size, please enter u-boot when the board is booting up to configure the display mode, and then type **boot** to continue boot-up process.
- **Make sure JP5 is close,** make CPU boot from TF card.

```
U-Boot SPL 2015.07-g4e2a180 (Jul 11 2019 - 15:17:51)
reading args
spl_load_image_fat_os: error reading image args, err - -1
reading u-boot.img
reading u-boot.img
U-Boot 2015.07-g4e2a180 (Jul 11 2019 - 15:17:51 +0800)
          Watchdog enabled
I2C:     ready
DRAM:    512 MiB
NAND:    0 MiB
MMC:     OMAP SD/MMC: 0, OMAP SD/MMC: 1
reading uboot.env
** Unable to read "uboot.env" from mmc0:1 **
Using default environment
reading logo.bmp
FAT: Misaligned buffer address (8000a002)
Net:     <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:    1
switch to partitions #0, OK
mmc1 is current device
SD/MMC found on device 1
reading boot.scr
** Unable to read file boot.scr **
reading uEnv.txt
300 bytes read in 5 ms (58.6 KiB/s)
Loaded env from uEnv.txt
Importing environment from mmc1 ...
Running uenvcmd ...
reading /am335x-som860e.dtb
38105 bytes read in 13 ms (2.8 MiB/s)
reading /zImage
4111720 bytes read in 326 ms (12 MiB/s)
reading ramdisk.img
14143658 bytes read in 1100 ms (12.3 MiB/s)
```

```
Booting from ramdisk ...
Kernel image @ 0x82000000 [ 0x000000 - 0x3ebd68 ]
## Loading init Ramdisk from Legacy Image at 88080000 ...
   Image Name:
   Created:          2019-06-13     6:38:03 UTC
   Image Type:     ARM Linux RAMDisk Image (gzip compressed)
   Data Size:       14143594 Bytes = 13.5 MiB
   Load Address: 81600000
   Entry Point:    81600000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Loading Ramdisk to 8f282000, end 8ffff06a ... OK
   Loading Device Tree to 8f275000, end 8f2814d8 ... OK
Starting kernel ...
[      0.000000] Booting Linux on physical CPU 0x0
[      0.000000] Initializing cgroup subsys cpuset
[      0.000000] Initializing cgroup subsys cpu
[      0.000000] Initializing cgroup subsys cpuacct
[        0.000000] Linux version 4.1.6 (chengpg@Ubuntu18) (gcc version 4.9.2 20140904
(prerelease) (crosstool-NG linaro-1.13.1-4.9-2014.09 - Linaro GCC 4.9-2014.09) ) #1
PREEMPT Thu Jul 11 15:20:30 CST 2019
[      0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c5387d
[      0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[      0.000000] Machine model: TI AM335x SOM860E
[      0.000000] cma: Reserved 24 MiB at 0x9e800000
[      0.000000] Memory policy: Data cache writeback
[      0.000000] CPU: All CPU(s) started in SVC mode.
[      0.000000] AM335X ES2.1 (sgx neon )
......
[      2.709173] RAMDISK: gzip image found at block 0
[      5.827792] EXT4-fs (ram0): mounting ext2 file system using the ext4 subsystem
[      5.837166] EXT4-fs (ram0): mounted filesystem without journal. Opts: (null)
[      5.844376] VFS: Mounted root (ext2 filesystem) on device 1:0.
[      5.851233] devtmpfs: mounted
[      5.855765] Freeing unused kernel memory: 740K (c0a59000 - c0b12000)
[      5.911728] EXT4-fs (ram0): re-mounted. Opts: (null)
Starting logging: OK
Populating /dev using udev: [      6.106777] udevd[97]: starting version 2.1.1
[      6.121516] random: udevd urandom read with 18 bits of entropy available
[      7.269066] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may
be corrupt. Please run fsck.
[      7.370398] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Some data may
```

```
be corrupt. Please run fsck.
done
Starting watchdog...
Initializing random number generator... done.
Starting system message bus: done
Starting network...
[      8.080994] net eth0: initializing cpsw version 1.12 (0)
[      8.167483] net eth0: phy found : id is : 0x4dd072
ssh-keygen: generating new host keys: ED25519
Starting sshd: OK
```

The above information indicates a successful boot-up of Linux from TF card.

2.9.2 Updating eMMC

eMMC boot-up image update is accomplished with the use of ramdisk， whether eMMC has data or not, Ramdisk can be used to make eMMC image.

1） Preparation

a) Format the TF card as FAT or FAT32 file system by using HP USB Disk Storage Format Tool 2.0.6;

b) Copy files **MLO, u-boot.img, am335x-som860e.dtb, zImage ， ramdisk.img, rootfs.tar.xz, logo.bmp, uEnv.txt, uEnv_eMMC.txt** into TF card;

2） Update

a) Insert TF card which contains the system images into the development board and shourt the jumper **JP5**, and then power up. The update process will auto start.

```
U-Boot SPL 2015.07-gba89fddf4-dirty (Apr 09 2019 - 14:04:26)
reading args
spl_load_image_fat_os: error reading image args, err - -1
reading u-boot.img
reading u-boot.img
U-Boot 2015.07-gba89fddf4-dirty (Apr 09 2019 - 14:04:26 +0800)
  Watchdog enabled
I2C:     ready
DRAM:    512 MiB
NAND:    0 MiB
```

```
MMC:     OMAP SD/MMC: 0, OMAP SD/MMC: 1
** No partition table - mmc 0 **
Using default environment
reading logo.bmp
FAT: Misaligned buffer address (8000a002)
Net:     <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:  1
switch to partitions #0, OK
mmc1 is current device
SD/MMC found on device 1
reading boot.scr
** Unable to read file boot.scr **
reading uEnv.txt
300 bytes read in 4 ms (73.2 KiB/s)
Loaded env from uEnv.txt
Importing environment from mmc1 ...
Running uenvcmd ...
reading /am335x-som860e.dtb
37689 bytes read in 11 ms (3.3 MiB/s)
reading /zImage
4048952 bytes read in 318 ms (12.1 MiB/s)
reading ramdisk.img
14143658 bytes read in 1101 ms (12.3 MiB/s)
Booting from ramdisk ...
Kernel image @ 0x82000000 [ 0x000000 - 0x3dc838 ]
## Loading init Ramdisk from Legacy Image at 88080000 ...
```

```
……
Starting logging: OK
Populating /dev using udev: [       6.061087] udevd[96]: starting version 2.1.1
[       6.067224] random: udevd urandom read with 27 bits of entropy available
done
Starting watchdog...
Initializing random number generator... done.
Starting system message bus: done
Starting network...
[       7.910771] net eth0: initializing cpsw version 1.12 (0)
[       7.997811] net eth0: phy found : id is : 0x4dd072
ssh-keygen: generating new host keys: ED25519
Starting sshd: OK
running system update...
```

```
UPDATE FLAG    : TRUE
==============eMMC UPDATE============
Warning: disk /dev/mmcblk0 will be formatted !
... ...
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): [      13.017622] cfg80211: Exceeded CRDA call max attempts. Not calling CRDA
done
Writing superblocks and filesystem accounting information: done

[     21.038910] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
Info: Copy /media/mmcblk1p1/MLO to /tmp/p1
Info: Copy /media/mmcblk1p1/u-boot.img to /tmp/p1
Info: Copy /media/mmcblk1p1/logo.bmp to /tmp/p1
Info: Copy /media/mmcblk1p1/uEnv_eMMC.txt to /tmp/p1/uEnv.txt
Info: Copy /media/mmcblk1p1/am335x-som860e.dtb to /tmp/p1
Info: Copy /media/mmcblk1p1/zImage to /tmp/p1
Thu Jun 13 14:56:00 UTC 2019
Info: Copy /media/mmcblk1p1/rootfs.tar.xz to /tmp/p2
/media/mmcblk1p1/rootfs.tar.xz (1/1)
   100 %            46.9 MiB / 157.9 MiB = 0.297     2.2 MiB/s           1:13
UPDATE   : COMPLETED
```

Fast flashing LED on the board indicates that the update is running;

When the LEDs on the board blink as heart frequency and buzzer alarmed, the update has been finished;

Please remove TF card and the JP5 jumper cap, and then reboot the board, Linux will boot up from eMMC now.

3） U-boot configuration

The default display definition is 800*600. If you are working with the LCDs of other size, please enter u-boot when the board is booting up to configure the display mode, and then type **boot** to continue boot-up process

2.10 Display Mode Configuration

System supports a wide range of display mode. Users can change the display mode by modifying the U-Boot configure parameters.

1） Edit the display configuration library file: **include/configs/am335x_evm.h**：

```
#if defined(CONFIG_BOARD_SOM860E)
# define LCDMODE                "dispmode=800x600\0"
```

Please change the display mode parameter according to the LCD you are using，the supported display definition includes 480x272, 640x480, 800x480, 800x600 and 1024x768;

2） Re-compile u-boot and get new image.

2.11 Test and Demonstration

This section will carry out some tests on the devices.

2.11.1 LED Test

The D35 LED on the board is the system indicator.

The following operations are accomplished in HyperTerminal:

1) Controlling system indicator:

- root@arm:~# **echo none > /sys/class/leds/sys/trigger**
- root@arm:~# **echo 1 > /sys/class/leds/sys/brightness**
- root@arm:~# **echo 0 > /sys/class/leds/sys/brightness**

Restoring to heart blink function:

- root@arm:~# **echo heartbeat > /sys/class/leds/sys/trigger**

2.11.2 KEYPAD Test

The board has two user custom keys, BACK and MENU. You can test them by executing the following instructions:

```
root@arm:~# evtest /dev/input/event1
Input driver verevdev: (EVIOCGBIT): Suspicious buffer size 511
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 1 (Esc)
    Event code 59 (F1)
Testing ... (interrupt to exit)
Event: time 1233046135.256046, type 1 (Key), code 1 (Esc), value 1
Event: time 1233046135.256053, ---------- Report Sync ------------
Event: time 1233046135.426967, type 1 (Key), code 1 (Esc), value 0
Event: time 1233046135.426970, ---------- Report Sync ------------
Event: time 1233046136.373255, type 1 (Key), code 59 (F1), value 1
Event: time 1233046136.373260, ---------- Report Sync ------------
Event: time 1233046136.548841, type 1 (Key), code 59 (F1), value 0
Event: time 1233046136.548844, ---------- Report Sync ------------
```

Note：

🕮 Press Ctrl+C to quit the test. These combined keys can be used to quit any following test.

2.11.3 Touch Screen Test

1) Execute the following instruction to test touch-screen:

- root@arm:~# **ts_calibrate**

The information on LCD will guide you to click the icon "+" for 5 times to complete the calibration.

2) Calibration is completed, enter the following commands for Touch Panel Test:

- root@arm:~# **ts_test**

Select drawing dots or drawing lines from the prompt information to start testing.

2.11.4 Backlight Test

The backlight brightness has a range from 0 to 8, in which 8 means highest

brightness, 0 means lowest.

1) Execute the following instruction to view the default backlight brightness:

- root@arm:~# **cat /sys/class/backlight/backlight/brightness**

2) Execute the following instructions to set backlight brightness to 0 and check if the value is set:

- root@arm:~# **echo 0 > /sys/class/backlight/backlight/brightness**

Now backlight is turned off and the LCD shows a black screen.

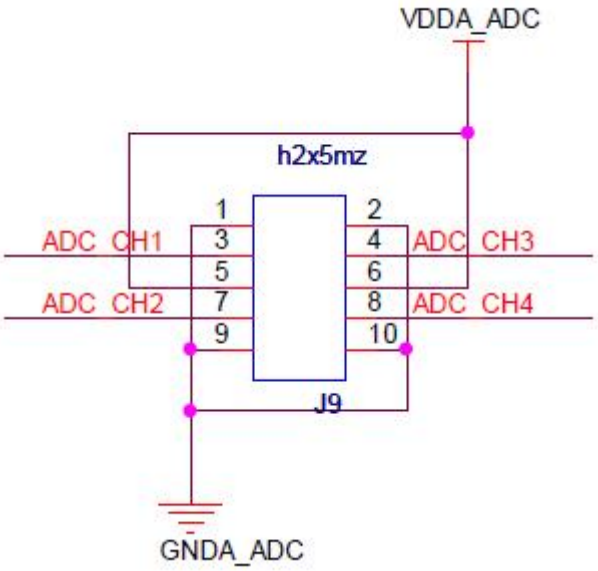
3) Execute the following instructions to set brightness to level 8:

- root@arm:~# **echo 8 > /sys/class/backlight/backlight/brightness**

The screen is turned on.

2.11.5 ADC Test

SOM-860-E has four channels ADC to measure external analog signal voltage.



ADC Pin	System node
ADC_CH1	in_voltage4_raw
ADC_CH2	in_voltage5_raw
ADC_CH3	in_voltage6_raw
ADC_CH4	in_voltage7_raw

Connect the signal that to be measured to a ADC pin, for example ADC_CH2, execute the following instruction to get ADC value:

- root@arm:~# **cat /sys/bus/iio/devices/iio:device0/in_voltage5_raw**

Follow below formula to get adc input signal voltage:

Voltage Input = VDDA_ADC * (value read) / 4096

VDDA_ADC=1.8V。

2.11.6 RTC Test

There is a hardware RTC which can store and recover system time; please follow the steps listed below to test RTC：

1) Set system time to 2019-06-30 12:10

```
root@arm:~# date -s "2019-06-30 12:10"
Sun Jun 30 12:10:00 UTC 2019
```

2) Write system time into RTC

- root@arm:~# **hwclock -w**

3) Read RTC

```
root@arm:~# hwclock
Sun Jun 30 12:10:21 2019    0.000000 seconds
```

4) Reboot the system and execute the following instructions to recover system clock

```
root@arm:~# hwclock -s
root@arm:~# date
Sun Jun 30 12:11:00 UTC 2019
```

The above information indicates that system clock has been recovered with hardware clock.

Note：

📖 There is no CR1220 backup battery installed by default, you need to purchase it separately.

2.11.7 TF Card Test

1） Insert a TF card, system will mount the filesystem under /media/ automatically; please execute the following instructions to view the device name of TF card in the system:

```
root@arm:~# df -h
Filesystem          Size    Used Avail Use% Mounted on
......
/dev/mmcblk0p1    121M    4.6M    116M     4% /media/mmcblk0p1
/dev/mmcblk1p1    3.7G     34M    3.6G     1% /media/mmcblk1p1
```

2） View the contents of the TF card:

```
root@arm:~# ls /media/mmcblk1p1
MLO                          ramdisk.img                  uEnv.txt
logo.bmp                     u-boot.img
```

3） Unmount the TF card manually:

- root@arm:~# **umount /media/mmcblk1p1**

4） mount the TF card manually and view the contents:

```
root@arm:~# mount -t vfat /dev/mmcblk1p1 /mnt
root@arm:~# df -h
Filesystem                   Size         Used Available Use% Mounted on
......
/dev/mmcblk1p1               3.7G      34M         3.6G     1% /mnt
root@arm:~# ls /mnt
MLO                          ramdisk.img                  uEnv.txt
logo.bmp                     u-boot.img
```

2.11.8 USB Device Test

The test of USB device is accomplished by creating a network communication between miniUSB interface on the board and USB interface on PC.

1) After system boots up, connect SBC to your PC through a Mini B-to-USB A cable, and then you need to install Linux USB Ethernet driver ;

2) Set IP address of the USB interface;

```
root@arm:~#    ifconfig usb0 192.168.1.115
root@arm:~#    ifconfig
lo            Link encap:Local Loopback
              inet addr:127.0.0.1    Mask:255.0.0.0
              UP LOOPBACK RUNNING    MTU:16436    Metric:1
              RX packets:26 errors:0 dropped:0 overruns:0 frame:0
              TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:2316 (2.2 KiB)    TX bytes:2316 (2.2 KiB)

usb0          Link encap:Ethernet    HWaddr 5E:C5:F6:D4:2B:91
              inet addr:192.168.1.115    Bcast:192.168.1.255    Mask:255.255.255.0
              UP BROADCAST RUNNING MULTICAST    MTU:1500    Metric:1
              RX packets:253 errors:0 dropped:0 overruns:0 frame:0
              TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:35277 (34.4 KiB)    TX bytes:10152 (9.9 KiB)
```

3) Right-click **My Network Places** on the desktop of your PC and select **Properties** to open **Network Connections** window; you can find a new **Local Area Connection** in the window as shown below;

4) Right-click the icon of new **Local Area Connection** and select **Properties**, and then double-click **Internet Protocol (TCP/IP)** to open the window, set IP address of computer as 192.168.1.15.

5) Use ping command in HyperTerminal window to test if the network works properly:

```
root@arm:~#    ping 192.168.1.15
PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms
```

6) The above information indicates the network has been created successfully.

Note：

🕮 Please ensure the IP addresses used above and the Ethernet IP address of the board are set in different segments.

2.11.9 USB Host Tes

1) Insert a flash drive to the USB interface, system will mount the filesystem under **/media/usbhd-sda1**:

```
root@arm:~# df -h
Filesystem                Size      Used Available Use% Mounted on
......
/dev/sda1                 3.7G     74M      3.6G   2% /media/usbhd-sda1
root@arm:~# ls /media/usbhd-sda1
MLO       u-boot.img    zImage
```

2) Unmount flash drive manually:

- root@arm:~# **cd**
- root@arm:~# **umount /media/usbhd-sda1**

3) Check if the flash drive has been removed sucessfully，input the command ‘df’, can’t find the directory **/media/usbhd-sda1/.**

```
root@arm:~# df
Filesystem           Size  Used Avail Use% Mounted on
ubi0:rootfs          206M   55M  146M  28% /
devtmpfs             237M     0  237M   0% /dev
tmpfs                249M     0  249M   0% /dev/shm
tmpfs                249M  6.6M  243M   3% /run
tmpfs                5.0M     0  5.0M   0% /run/lock
tmpfs                249M     0  249M   0% /sys/fs/cgroup
```

4) Mount the flash drive manually:

```
root@arm:~# mount   -t   vfat   /dev/sda1   /mnt/
root@arm:~# df -h
Filesystem                Size      Used Available Use% Mounted on
......
/dev/sda1                 3.7G     74M      3.6G   2%   /mnt
```

2.11.10 AUDIO Test

The filesystem is integrated with alsa-utils audio recording and playback tool. You can test it by following the steps below:

1) **Audio Recording:**

Insert a microphone into the 3.5mm audio input interface, then execute the following insteuction to start audio recording:

```
root@arm:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
   Transformation table:
      0 <- 0*0.5 + 1*0.5
Its setup is:
   stream          : CAPTURE
   access          : RW_INTERLEAVED
   format          : S16_LE
   subformat       : STD
   channels        : 1
   rate            : 44100
   exact rate      : 44100 (44100/1)
   msbits          : 16
   buffer_size     : 22052
   period_size     : 5513
   period_time     : 125011
   tstamp_mode     : NONE
   period_step     : 1
   avail_min       : 5513
   period_event : 0
   start_threshold    : 1
   stop_threshold     : 22052
   silence_threshold: 0
   silence_size : 0
   boundary        : 1445199872
Slave: Hardware PCM card 0 'AM335x-SGTL5000' device 0 subdevice 0.....
```

2) **Audio Playing:**

Adjust Audio Volume:

```
root@arm:~# amixer set Headphone 100
```

Where the volume parameter 1(100 eg) give value in range [0..127].

Insert an earphone into the 3.5mm audio output interface, and then execute the following instruction to play the recorded audio.

```
root@arm:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
   Transformation table:
      0 <- 0
      1 <- 0
Its setup is:
   stream          : PLAYBACK
   access          : RW_INTERLEAVED
   format          : S16_LE
   subformat       : STD
   channels        : 1
   rate            : 44100
   exact rate      : 44100 (44100/1)
   msbits          : 16
   buffer_size     : 22052
   period_size     : 5513
   period_time     : 125011
   tstamp_mode     : NONE
   period_step     : 1
   avail_min       : 5513
   period_event : 0
   start_threshold    : 22052
   stop_threshold     : 22052
   silence_threshold: 0
   silence_size : 0
   boundary        : 1445199872
Slave: Hardware PCM card 0 'AM335x-SGTL5000' device 0 subdevice 0.....
```

2.11.11 VIDEO Test

The debian8 already supports mplayer, it can play avi, mp4 etc. video files.

- root@arm:**~# mplayer -vo fbdev2 movie.avi**

Note:

The video dimension can’t exceed the size of LCD, or play fail.

2.11.12 Network Test

There are two Ethernet interfaces, NET1 (J1) and NET2 (J2) ; Corresponding device nodes are eth0 and eth1. Please use two network cables connect the interfaces to a network and ensure that the IP addresses of the interfaces are set in different network segments.

Note:

The IP addresses of the two network interfaces need to be set in different network segments, or the testing would be failed.

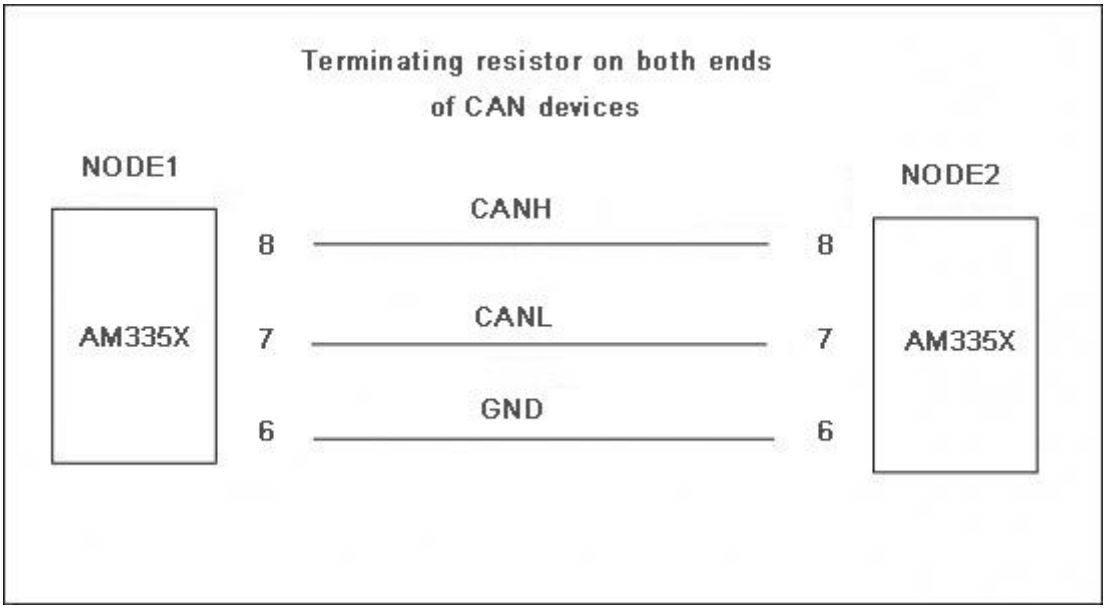
```
root@arm:~# ifconfig eth0 192.192.192.200
root@arm:~# ifconfig
eth0      Link encap:Ethernet  HWaddr D4:94:A1:8D:EB:25
          inet addr:192.192.192.200 Bcast:192.192.192.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:137 errors:0 dropped:4 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13792 (13.4 KiB)  TX bytes:0 (0.0 B)
          Interrupt:40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
root@arm:~# ping 192.192.192.170
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms
root@arm:~# ifconfig eth1 192.168.168.116
root@arm:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:17:EA:96:34:D5
          net addr:192.168.168.116  Bcast:192.168.168.255 Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
```

```
                RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                collisions:0 txqueuelen:1000
                RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
      Lo        Link encap:Local Loopback
                inet addr:127.0.0.1  Mask:255.0.0.0
                UP LOOPBACK RUNNING  MTU:16436  Metric:1
                RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                collisions:0 txqueuelen:0
                RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
root@arm:~# ping 192.168.168.121
PING 192.168.168.121 (192.168.168.121): 56 data bytes
64 bytes from 192.168.168.121: seq=0 ttl=64 time=7.969 ms
64 bytes from 192.168.168.121: seq=1 ttl=64 time=0.319 ms
```

The above information indicates the network is working properly.

2.11.13 CAN Test

There is only one CAN bus. Please connect the CAN interfaces on your SBC8600B and another CAN device according to the board schematic and the figure shown below:



Follow the steps listed below to complete CAN test:

1) Set the communication bit rate to 125Kbps for both CAN nodes, and enable CAN devices:

Making sure the device is closed, otherwise an error will occur.

- root@arm:~# **ifconfig can0 down**

Set the communication bit rate to 125Kbps:

- root@arm:~# **/bin/ip link set can0 type can bitrate 125000**

Essential step to set recovery time:

- root@arm:~# **/bin/ip link set can0 type can restart-ms 100**

Open CAN device:

- root@arm:~# **ifconfig can0 up**

2) Transmit data on the one device by typing the following instructions.

- root@arm:~# **cansend can0 "5A1#1122334455667788"**

Note:

- The instruction sends data only once. Type it again to send another date package.
- The receiving device needs to remain in receiving status so that the received information can be shown in the terminal window.

3) Receiving data package:

- root@arm:~# **candump can0**

The terminal window will print the information of the received data package.

4) Stop the CAN device:

- root@arm:~# **ifconfig can0 down**

You can run the test with different baudrates. The following table contains several baudrates which have been proved. However, you can also try other baudrates if you want:

25Kbps（250000）

50Kbps（50000）

125Kbps（125000）

500Kbps（500000）

650Kbps（650000）

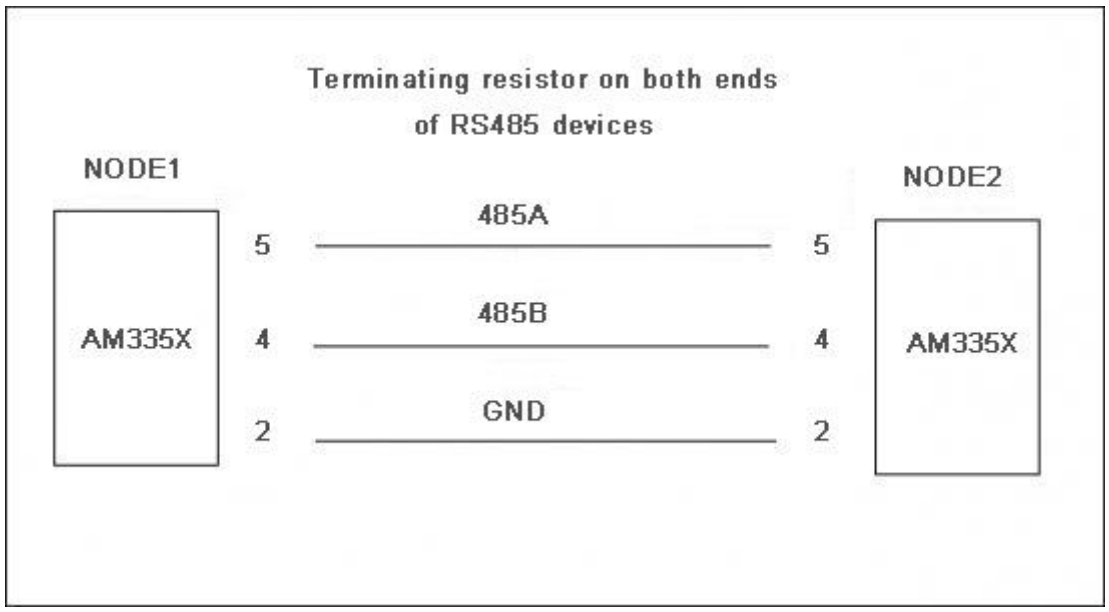
1Mbps（1000000 ）

Note:

- You need to disable CAN bus BEFORE you can set a new baud rate. The two end devices of CAN communication must set identical baud rate.
- For source code, please refer to Open-Source software can-utils.

2.11.14 RS485 Test

Please connect RS485 of SBC with another RS485 node according to the following figure shown.



RS485 interface works under half-duplex mode, which means each of two ends can only send or receive data at a time. Copy the app file under CD\Source\App\uart\ into TF card, and then insert the card and execute the following instructions;

```
root@arm:~# /test/app/uart -d /dev/ttyS1 -b 115200
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 SEND: 1234567890
```

```
/dev/ttyS1 RECV 10 total
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
```

2.11.15 Serial Interface Test

Short the pins RX3V3 and TX3V3 of J5 on the board and copy the file uart_test under linux\example\uart_test to TF card, and then instert it on the board. Execute the following instructions in the terminal window:

- root@arm:~# **/embest/uart -d /dev/ttyS2 -b 115200**

The following information in the terminal window indicates a successful testing.

```
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
```

The same testing method can be applied on UART3, UART4 and UART5 of J6 and J7.

2.11.16 Buzzer Test

1） Enable the buzze;

- root@arm:~# **echo 1 > /sys/class/leds/buzzer/brightness**

2） Disable the buzzer;

- root@arm:~# **echo 0 > /sys/class/leds/buzzer/brightness**

2.11.17 Suspend & Resume Test

- root@arm:~# **echo mem > /sys/power/state**

```
[ 4351.715262] PM: Syncing filesystems ... done.
[ 4351.729920] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 4351.738777] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[ 4351.747471] Suspending console(s) (use no_console_suspend to debug)
[Debug Uart Input, Touch Screen or Button S2]
[ 4352.047646] PM: suspend of devices complete after 293.057 msecs
[ 4352.049277] PM: late suspend of devices complete after 1.602 msecs
[ 4352.051137] PM: noirq suspend of devices complete after 1.833 msecs
[ 4352.051147] PM: Successfully put all powerdomains to target state
[ 4352.051147] PM: Wakeup source UART
[ 4352.068706] PM: noirq resume of devices complete after 17.431 msecs
[ 4352.070122] PM: early resume of devices complete after 1.071 msecs
[ 4352.070950] net eth0: initializing cpsw version 1.12 (0)
[ 4352.075081] sgtl5000 0-000a: Failed to get mclock: -2
[ 4352.145051] net eth0: phy found : id is : 0x4dd072
[ 4352.147181] net eth1: initializing cpsw version 1.12 (0)
[ 4352.150627] sgtl5000 0-000a: Failed to get mclock: -2
[ 4352.225065] net eth1: phy found : id is : 0x4dd072
[ 4352.344861] PM: resume of devices complete after 274.700 msecs
[ 4352.422824] Restarting tasks ...
[ 4352.426558] usb 2-1: USB disconnect, device number 2
[ 4352.454743] done
```

2.11.18 Unique ID

- root@arm:~# **cat /sys/devices/soc0/soc_id**

```
A90BC968F798A90BC968F998
```

or

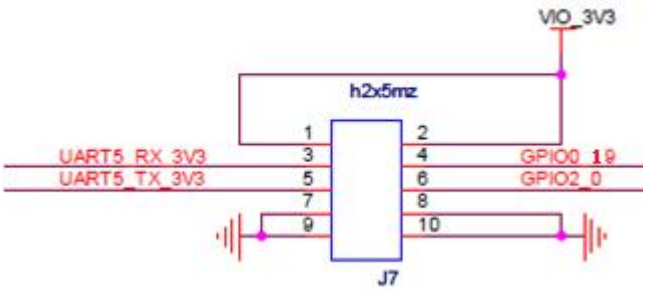
- root@arm:~# **cat /proc/cpuinfo**

```
processor       : 0
model name      : ARMv7 Processor rev 2 (v7l)
BogoMIPS        : 498.89
Features        : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
```

```
CPU variant     : 0x3
CPU part        : 0xc08
CPU revision    : 2

Hardware        : Generic AM33XX (Flattened Device Tree)
Revision        : 0000
Serial          : A90BC968F798A90BC968F998
```

2.11.19 GPIO Test

Below steps show to test GPIO0_19 and GPIO2_0 of J7.



1） Export GPIO directory, once only

Export GPIO0_19, will create directory /sys/class/gpio/gpio0_19

Execute the following instructions to install a client;

- root@arm:~# **echo $((32 * 0 + 19)) > /sys/class/gpio/export**

The directory /sys/class/gpio/gpio19 will be created.

Export GPIO2_0, will create directory /sys/class/gpio/gpio2_0;

- root@arm:~# **echo $((32 * 2 + 0)) > /sys/class/gpio/export**

The directory /sys/class/gpio/gpio64 will be created

2） Set GPIO0_19 to output mode;

- root@arm:~# **echo out > /sys/class/gpio/gpio19/direction**

Output high level;

- root@arm:~# **echo 1 > /sys/class/gpio/gpio19/value**

Output low level;

- root@arm:~# **echo 0 > /sys/class/gpio/gpio19/value**

3） Set GPIO0_19 to input mode;

- root@arm:~# **echo in > /sys/class/gpio/gpio19/direction**

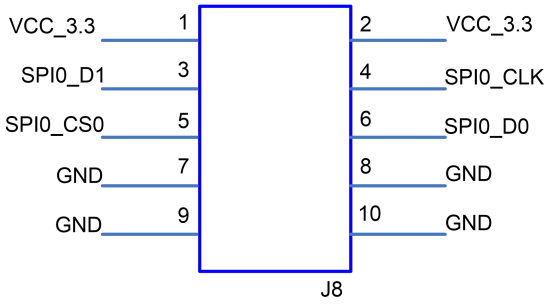
Read input level of GPIO0_19;

- root@arm:~# **cat /sys/class/gpio/gpio19/value**

Note:

🕮 To test GPIO2_0, replace gpio19 in the path above with gpio64.

2.11.20 SPI Test



Connect pin3 and pin6 of J8, execute command;

- root@arm:~# **/test/app/spidev_test -D /dev/spidev1.0 -s 1000000 -p "Hello SPI" -v**

```
spi mode: 0x0
bits per word: 8
max speed: 1000000 Hz (1000 KHz)
TX | 48 65 6C 6C 6F 20 53 50 49 __ __ __ __ __ __ __ __ __ ______ __ __  | Hello.SPI
RX | 48 65 6C 6C 6F 20 53 50 49 __ __ __ __ __ __ __ __ __ __ __ __ __  | Hello.SPI
```

It shows string sent is "Hello SPI" and the one recieved is the same string.

Note:

🕮 SPI0_D0 and SPI0_D1 can exchange function, decided by dt, configured as **ti,pindir-d0-out-d1-in** as default. So SPI0_D0 is SPI_MOSI and SPI0_D1 is SPI_MISO.

2.11.21 Camera Test

CAM8200-U

1） Connect camera to the USB port

```
[  507.281230] usb 2-1.2: New USB device found, idVendor=090c, idProduct=f37d
[  507.288207] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[  507.295565] usb 2-1.2: Product: SMI
[  507.301263] usb 2-1.2: Manufacturer: SMI
[  507.315001] uvcvideo: Found UVC 1.00 device SMI (090c:f37d)
[  507.367637] input: SMI as /devices/platform/ocp/47400000.usb/47401c00.usb/musb-hdrc.1.auto/usb2/2-1/2-1.2/2-1.2:1.0/input/input2
```

2） Display

- root@arm:~# **/test/app/luvc_test -c -S -f mjpg /dev/video0**

Note:

- **7.0 inch LCD** is recommended.
- The camera resolution should be set to **640*480.**

2.11.22 WIFI Test

Wi-Pi: RT5370

1） Connect WIFI module

```
[  139.165325] usb 2-1.2: New USB device found, idVendor=148f, idProduct=5370
[  139.172303] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=
3
[  139.181356] usb 2-1.2: Product: 802.11 n WLAN
[  139.185776] usb 2-1.2: Manufacturer: Ralink
[  139.191570] usb 2-1.2: SerialNumber: 1.0
[  139.297591] usb 2-1.2: reset high-speed USB device number 4 using musb-hdrc
[  139.428426] ieee80211 phy0: rt2x00_set_rt: Info - RT chipset 5390, rev 0502 dete
cted
[  139.452619] ieee80211 phy0: rt2x00_set_rf: Info - RF chipset 5370 detected
[  141.405892] ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading firmware fi
le 'rt2870.bin'
[  141.424760] ieee80211 phy0: rt2x00lib_request_firmware: Info - Firmware detected
- version: 0.29
```

2） Scan WIFI AP

- root@arm:~# **iwlist wlan0 scan**

```
          Cell 35 - Address: F0:3E:90:61:E1:78
                    Channel:7
                    Frequency:2.442 GHz (Channel 7)
                    Quality=45/70  Signal level=-65 dBm
                    Encryption key:on
                    ESSID:"EMBEST_WIFI"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
                              36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=000000063352af2f
                    Extra: Last beacon: 720ms ago
                    IE: Unknown: 000B454D424553545F57494649
                    IE: Unknown: 010482848B96
                    IE: Unknown: 030107
                    IE: Unknown: 0706434E20010D14
                    IE: Unknown: 2A0100
                    IE: Unknown: 32080C1218243048606C
                    IE: Unknown: DD1E00904C33AD0103FFFF000000000000000000
00100000000040646E70D00
                    IE: Unknown: 2D1AAD0103FFFF00000000000000000000001000000
000040646E70D00
                    IE: Unknown: DD1A00904C340700000000000000000000000000
```

```
000000000000000
                    IE: Unknown: 3D160700000000000000000000000000000000000000
000000
                    IE: Unknown: 7F0400000000
                    IE: Unknown: DD180050F2020101890003A4000027A4000042435
E0062322F00
                    IE: Unknown: DD080013920100018515
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : 802.1x
```

3） Register ESSID and password

- root@arm:~# **wpa_passphrase MY_WIFI MYPASSWD >>**

  /etc/wpa_supplicant/wpa_supplicant.conf

4） Restart wifi service

- root@arm:~# **systemctl restart wpa_supplicant.service**

Wait for a while. If everything is OK, the board can connect to the AP.

5） Inquire IP address

- root@arm:~# **ifconfig wlan0**

```
wlan0     Link encap:Ethernet  HWaddr 40:A5:EF:05:5A:B9
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3503 (3.4 KiB)  TX bytes:1736 (1.6 KiB)
```

2.11.23 Bluetooth Test

CSR4.0 Bluetooth

1） Connect BT module

```
[ 2456.167582] usb 2-1.2: new full-speed USB device number 5 using musb-hdrc
[ 2456.306046] usb 2-1.2: New USB device found, idVendor=0a12, idProduct=0001
[ 2456.313035] usb 2-1.2: New USB device strings: Mfr=0, Product=2, SerialNumber=0
[ 2456.322207] usb 2-1.2: Product: CSR8510 A10
[ 2456.529269] Bluetooth: hci0 hardware error 0x35
```

2） Scan other BT device

- root@arm:~# **hciconfig hci0 up; hcitool scan**

```
        28:B2:BD:79:93:F1       GBRLT-XXXXX
        0C:30:21:D3:5A:9B       BLUE-XXXXX
```

3） Ping test

- root@arm:~# **l2ping 0C:30:21:D3:5A:9B**

```
Ping: 0C:30:21:D3:5A:9B from 00:1A:7D:DA:71:13 (data size 44) ...
44 bytes from 0C:30:21:D3:5A:9B id 0 time 9.98ms
44 bytes from 0C:30:21:D3:5A:9B id 1 time 93.48ms
44 bytes from 0C:30:21:D3:5A:9B id 2 time 225.96ms
44 bytes from 0C:30:21:D3:5A:9B id 4 time 90.96ms
4 sent, 4 received, 0% loss
```

2.11.24 Debian Configuration

Version: ARM Debian 8

1) Static Network Configuration

Configration File	**/etc/network/interfaces**
Static	iface eth0 inet static address 192.168.1.210 netmask 255.255.255.0

config of static IP manually:

- root@arm:~# **ifconfig eth0 192.168.1.210**

Note:

📖 DHCP should be disabled: **systemctl stop embest-network.service; systemctl disable embest-network.service.**

2) Dynamic Network Configuration

Disable the static configuration in **/etc/network/interfaces**;

Enable DHCP service:

- root@arm:~# **systemctl restart system-network.service;**
- root@arm:~# **systemctl enable system-network.service;**

Then re-plug net cable on port eth0 and eth1, the device can aquire IP distributed by the DHCP server.

Note:

📖 The default severce of DHCP is monitoring all three ports, including eth0, eth1 and wlan0, if you only want to monitor one or two of them, you can edit **/etc/SystemNetworkDHCP.sh**.

3) Time Zone Configration

System use UTC time by default. Follow below steps to change Timezone to 'Shanghai';

- root@arm:~# **echo "Asia/Shanghai" > /etc/timezone**
- root@arm:~# **ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime**

4) Auto starting application

Follow below steps to make the application **/etc/EmbestUser.sh** auto starting and restart system when all of them finished;

- root@arm:~# **vi /lib/systemd/system/user.service**

```
[Unit]
Description=User Program
After=network.target

[Service]
ExecStart=/etc/User.sh

[Install]
WantedBy=multi-user.target
```

- root@arm:~# **systemctl enable user.service; sync**

```
Created symlink /etc/systemd/system/multi-user.target.wants/user.service → /lib/systemd/system/user.service.
```

5) Disable LCD cursor blinking

LCD cursor will not blink by default. To enable the blinking, follow below instruction;

- root@arm:~# **echo 1 > /sys/class/graphics/fbcon/cursor_blink**

Forbid cursor from blinking

- root@arm:~# **echo 0 > /sys/class/graphics/fbcon/cursor_blink**

2.12 Development of Application

This section will introduce the common process of development applications through several examples.

2.12.1 Development of LED Application

1) Compose source code led_acc.c to instruct the two LEDs to blink in the mode of accumulator.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/ioctl.h>
#include <fcntl.h>

#define LED1 "/sys/class/leds/sys_led/brightness"
#define LED2 "/sys/class/leds/user_led/brightness"

int main(int argc, char *argv[])
{
        int f_led1, f_led2;
        unsigned char i = 0;
        unsigned char dat1, dat2;
        if((f_led1 = open(LED1, O_RDWR)) < 0){
                printf("error in open %s",LED1);
                return -1;
```

```
        }
        if((f_led2 = open(LED2, O_RDWR)) < 0){
                printf("error in open %s",LED2);
                return -1;
        }
        for(;;){
                i++;
                dat1 = i&0x1 ? '1':'0';
                dat2 = (i&0x2)>>1 ? '1':'0';
                write(f_led1, &dat1, sizeof(dat1));
                write(f_led2, &dat2, sizeof(dat2));
                usleep(300000);
        }
}
```

2) Execute the following instruction in Ubuntu Linux system to implement cross compilation;

- **arm-linux-gcc led_demo.c -o led_demo**

3) Download the compiled files to SBC8600B and enter the directory where the file led_acc is saved, and then execute the following instruction to run LED application.

- **./led_demo &**

2.12.2 Development of CAN Applicatio

1） Defining Data to Be Sent;

The syntax for CAN frame definition is “<can_id>#{R|data}”; CAN_ID could be a 3-bit (standard frame) or 8-bit (extended frame) hexadecimal format; Data could be a 0 to 8-bit hexadecimal format (can be separated with separators “.”).

Often used syntax :

```
char *cmd_str = "123#1122334455667788";

char *cmd_str = "123#11.22.33.44.55.66.77.88";

char *cmd_str = "12345678#112233";
```

2） Creating Socket:

Before CAN network is ready to function, you need to create a socket first. SocketCAN introduces a new protocol family, so it is necessary to include PF_CAN as a parameter when calling the function socket(). Currently there are two CAN protocols available, one is Raw Socket protocol, the other is BCM (Broadcast Manager). For example:

```
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
```

After successfully creating a socket, you typically nened to use the function bind() to create a binding between the socket and a CAN interface. After binding (CAN_RAW) or connecting (CAN_BCM) socket, you can use read()/write() on the socket. The basic CAN frame and socket address struct are defined in can.h under /include/linux/. For example:

```
/**
 * struct can_frame - basic CAN frame structure
 * @can_id:   the CAN ID of the frame and CAN_*_FLAG flags, see above.
 * @can_dlc: the data length field of the CAN frame
 * @data:      the CAN frame payload.
 */
struct can_frame {
        canid_t can_id;   /* 32 bit CAN_ID + EFF/RTR/ERR flags */
        __u8      can_dlc; /* data length code: 0 .. 8 */
        __u8      data[8] __attribute__((aligned(8)));
};
```

The valid data of struct is included in data[] array with 64-bit byte alignment, so users can easily transmit their custom struct and union with data[]. There is no default byte sequence on CAN bus. Calling read() over the socket CAN_RAW would return a can_frame struct to user space. sockaddr_can struct includes an index which is bound to specific interfaces. For example:

```
/**
 * struct sockaddr_can - the sockaddr structure for CAN sockets
 * @can_family:   address family number AF_CAN.
 * @can_ifindex: CAN network interface index.
 * @can_addr:      protocol specific address information
 */
struct sockaddr_can {
        sa_family_t can_family;
```

```
        int         can_ifindex;
        union {
                /* transport protocol class address information (e.g. ISOTP) */
                struct { canid_t rx_id, tx_id; } tp;
                /* reserved for future CAN protocols address information */
        } can_addr;
};
```

3） Specify Interface Index and create bindin

In order to band sockets with all the CAN interfaces, the function ioctl() needs to be called when specifying an interface index; Interface index has to 0 so that sockets can receive CAN frames on all the enabled CAN interface. For example:

```
int s;
struct sockaddr_can addr;
struct ifreq ifr;

s = socket(PF_CAN, SOCK_RAW, CAN_RAW);

strcpy(ifr.ifr_name, "can0" );
ioctl(s, SIOCGIFINDEX, &ifr);

addr.can_family = AF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
bind(s, (struct sockaddr *)&addr, sizeof(addr));
```

4） Read CAN frames from Sockets

```
struct can_frame frame;

nbytes = read(s, &frame, sizeof(struct can_frame));

if (nbytes < 0) {
perror("can raw socket read");
return 1;
}

/* paranoid check ... */
if (nbytes < sizeof(struct can_frame)) {
fprintf(stderr, "read: incomplete CAN frame\n");
return 1;
```

```
}
```

5） Write CAN frames on Socket

```
nbytes = write(s, &frame, sizeof(struct can_frame));
```

6） Enable CAN Interface

```
int can_do_start(const char *name)
{
        return set_link(name, IF_UP, NULL);
}
```

7） Disable CAN interface

```
int can_do_stop(const char *name)
{
        return set_link(name, IF_DOWN, NULL);
}
```

8） CAN Demostration Code

CAN applications are saved under can_test directory which contains three source code files:

lib.c defines character conversion function;

```
int parse_canframe(char *cs, struct can_frame *cf);
/*Transfers a valid ASCII string decribing a CAN frame into struct can_frame*/
```

libsocketcan.c defines CAN interface function;

```
int can_do_start(const char *name) /*start the can interface*/
int can_do_stop(const char *name) /*stop the can interface*/
```

can_test.c, The following tables provide part of source code;

```
#define MAX_CANFRAME          "12345678#01.23.45.67.89.AB.CD.EF"
#define MAX_LONG_CANFRAME "12345678      [8] 10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101010     '........'"
static int s = -1;
```

```
char buf[sizeof(MAX_LONG_CANFRAME)+1]="";
char *cmd_str = "111#1122334455667788";// Define the message to be sent

int main(void)
{
	struct sockaddr_can addr;
	static struct ifreq ifr;
	const char* name = argv[1];

	if ((argc < 2) || !strcmp(argv[1], "--help"))
		help();

	if (argc < 3)
		cmd_show_interface(name);

	cmd_stop(argc, argv, name);    // can stop

	while (argc-- > 0) {
		if (!strcmp(argv[0], "bitrate"))
			cmd_bitrate(argc, argv, name);
		if (!strcmp(argv[0], "ctrlmode"))
			cmd_ctrlmode(argc, argv, name);
		if (!strcmp(argv[0], "start"))
			cmd_start(argc, argv, name);
		if (!strcmp(argv[0], "stop"))
			cmd_stop(argc, argv, name);
		argv++;
	}
	cmd_start(argc, argv, name); // can start
	if (s != -1) {
		return 0;
	}
	s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
	if(s < 0) {
		perror("socket");
		return 1;
	}
	memset(&ifr.ifr_name, 0, sizeof(ifr.ifr_name));
	strcpy(ifr.ifr_name, "can0");
	if(ioctl(s, SIOCGIFINDEX, &ifr) < 0) {
		perror("SIOCGIFINDEX");
		exit(1);
	}
```

```
	addr.can_family = AF_CAN;
	addr.can_ifindex = ifr.ifr_ifindex;

	if(bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
		perror("bind");
		return 1;
	}

	funct_select();
}
//Send message:
int can_send(char *buf)
{
	int nbytes;
	struct can_frame frame;

	if(parse_canframe(buf, &frame)) {
		fprintf(stderr, "\nWrong CAN-frame format!\n\n");
		fprintf(stderr, "Try: <can_id>#{R|data}\n");
		fprintf(stderr, "can_id can have 3 (SFF) or 8 (EFF) hex chars\n");
		fprintf(stderr, "data has 0 to 8 hex-values that can (optionally)");
		fprintf(stderr, " be seperated by '.'\n\n");
		fprintf(stderr, "e.g. 5A1#11.2233.44556677.88 / 123#DEADBEEF / ");
		fprintf(stderr, "5AA# /\n      1F334455#1122334455667788 / 123#R ");
		fprintf(stderr, "for remote transmission request.\n\n");
		return 1;
	}

	if((nbytes = write(s, &frame, sizeof(frame)))
	     != sizeof(frame)) {
		perror("write");
		return 1;
	}

	return 0;
}
// Receive message:
int can_recv(char *buf)
{
	int ret, nbytes;
	fd_set rdfs;
	struct can_frame frame;
```

```
        FD_ZERO(&rdfs);
        FD_SET(s, &rdfs);

        if((ret = select(s+1, &rdfs, NULL, NULL, NULL)) < 0) {
                perror("select");
                exit(1);
        }
        if(FD_ISSET(s, &rdfs)) {
                nbytes = read(s, &frame, sizeof(struct can_frame));
                if(nbytes < 0) {
                        perror("read");
                        return 1;
                }
                if(nbytes < sizeof(struct can_frame)) {
                        fprintf(stderr, "read: incomplete CAN frame\n");
                        return 1;
                }
                sprint_long_canframe(buf, &frame, 0);
                printf("%s\n", buf);
        }
}
```

9） Execute the following instructions in Ubuntu system to extact can_test.tar.bz2 from **Source\App\can_test.tar.bz2** to **\work** and then compile the file;

- **cd $HOME/work**
- **tar xvf /media/cdrom/Source/App/can_test.tar.bz2**
- **cd can_test**
- **make**

10） Download the compiled file, and then enter the directory where can_test is saved and execute the following instruction to run CAN application.

```
root@arm:~# ./can_test can0 bitrate 125000 ctrlmode triple-sampling on
can0 state: STOPPED
can0 bitrate: 125000, sample-point: 0.875
can0 ctrlmode: loopback[OFF], listen-only[OFF], tripple-sampling[ON],one-shot[OFF],
bd_can d_can: can0: setting CAN BT = 0x518
err-reporting[OFF]
```

```
can0 state: ERROR-ACTIVE


Select 1 : Send a message
Select 2 : Receive messages
>
```

Type **1** at the transmitting end and press **Enter** key to send data;

```
Select 1 : Send a message
Select 2 : Receive messages
>  1
Information is sent.....
Select 3 : Stop Send
>
```

Type **2** at the receiving end and press **Enter** key to receive data;

```
Select 1 : Send a message
Select 2 : Receive messages
>  2
111   [8] 11 22 33 44 55 66 77 88
111   [8] 11 22 33 44 55 66 77 88
111   [8] 11 22 33 44 55 66 77 88
```

2.12.3 Development of Serial Interface Application

The following table lists the header files required for operations on serial interface.

Header File	Description
#include <stdio.h>	Standard input/output definition
#include <stdlib.h>	Standard function library definition
#include <unistd.h>	UNIX standard function definition
#include <sys/types.h>	
#include <sys/stat.h>	
#include <fcntl.h>	File control definition
#include <termios.h>	PPSIX terminal control definition

1） Opening Serial Interface

The file for serial interfaces under Linux is saved in /dev; By using standard “Open" function, serial interface can be opened; For example:

```
int fd;
fd = open( "/dev/ttyS0", O_RDWR); /* open serial interface by write/read
method */
if (-1 == fd){
perror("error warning");/* can not open serial interface 1*/
}
```

2） Configurate Serial Interface；

Configuration of serial interfaces includes seting up baudrates, check bits, stop bits and values of struct member; For example:

```
struct    termios opt;
tcgetattr(fd, &opt);
cfsetispeed(&opt,B115200);        /*Set to 115200bps*/
cfsetospeed(&opt,B115200);
tcsetattr(fd,TCANOW,&opt);
```

No Check	**OddCheck**
8 bit	7bit
Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS8;	Option.c_cflag |= ~PARENB; Option.c_cflag &= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;
Even	**Space Check**
7 bit	7 bit
Option.c_cflag &= ~PARENB; Option.c_cflag |= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;	Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= &~CSIZE; Option.c_cflag |= CS8;

```
struct termio
{
        unsigned short    c_iflag;          /* input mode flag */
        unsigned short    c_oflag;          /* output mode flag */
        unsigned short    c_cflag;          /* control mode flag */
```

```
    unsigned short   c_lflag;        /* local mode flags */
    unsigned char    c_line;         /* line discipline */
    unsigned char    c_cc[NCC];      /* control characters */
};
```

3） Configurate Check Function;

```
/**
*@brief     set serial interface data bits, stop bits and check bits
*@param    fd         type    int    opened file handle of serial interface
*@param    databits    type    int data bits      value is 7 or 8
*@param    stopbits    type    int stop bits      value is 1 or 2
*@param    parity    type    int    check    value is N,E,O,,S
*/
int set_Parity(int fd,int databits,int stopbits,int parity)
{
    struct termios options;
    if (tcgetattr( fd,&options)   !=   0) {
        perror("SetupSerial 1");
        return(FALSE);
    }
    options.c_cflag &= ~CSIZE;
    switch (databits) /* set data bits */
    {
    case 7:
        options.c_cflag |= CS7;
        break;
    case 8:
        options.c_cflag |= CS8;
        break;
    default:
        fprintf(stderr,"Unsupported data size\n"); return (FALSE);
    }
   switch (parity)
   {
    case 'n':
    case 'N':
        options.c_cflag &= ~PARENB;     /* Clear parity enable */
        options.c_iflag &= ~INPCK;        /* Enable parity checking */
        break;
    case 'o':
    case 'O':
```

```
                options.c_cflag |= (PARODD | PARENB); /* set to odd check */
                options.c_iflag |= INPCK;                    /* Disnable parity
checking */
                break;
        case 'e':
        case 'E':
                options.c_cflag |= PARENB;        /* Enable parity */
                options.c_cflag &= ~PARODD;     /* change to even check */
                options.c_iflag |= INPCK;           /* Disnable parity checking */
                break;
        case 'S':
        case 's':    /*as no parity*/
               options.c_cflag &= ~PARENB;
                options.c_cflag &= ~CSTOPB;break;
        default:
                fprintf(stderr,"Unsupported parity\n");
                return (FALSE);
        }
       /* set stop bits */
       switch (stopbits)
       {
        case 1:
                options.c_cflag &= ~CSTOPB;
                break;
        case 2:
                options.c_cflag |= CSTOPB;
             break;
        default:
                 fprintf(stderr,"Unsupported stop bits\n");
                 return (FALSE);
}
/* Set input parity option */
if (parity != 'n')
        options.c_iflag |= INPCK;
tcflush(fd,TCIFLUSH);
options.c_cc[VTIME] = 150; /* set timeout to 15 seconds*/
options.c_cc[VMIN] = 0; /* Update the options and do it NOW */
if (tcsetattr(fd,TCSANOW,&options) != 0)
{
        perror("SetupSerial 3");
        return (FALSE);
}
return (TRUE);
```

```
}
```

Note:

🕮 If it is not terminal development and serial interfaces are only used to transmit data, but not to process, Raw Mode can be used to realize communication. For example:

```
options.c_lflag    &= ~(ICANON | ECHO | ECHOE | ISIG);    /*Input*/
options.c_oflag    &= ~OPOST;     /*Output*/
```

4） Write/Read Serial Interafc;

After configuration is done, serial interface can be written and read as files; The read function can be used to read serial interfaces; If serial interfaces are working under Raw Mode, the characters length returned by the function is the character length received by serial interfaces; Asynchronous read can be realized by using the function fcntl or select; The following table contains the source code example of write/read operations:

Write on Serial Interface:

```
char    buffer[1024];int       Length;int       nByte;    nByte = write(fd, buffer ,Length)
```

Read on Serial Interface

```
char    buff[1024];int       Len;int    readByte = read(fd,buff,Len);
```

5） Close serial interface;

```
close(fd);
```

6） The following table contains main source code involved in programs for serial interface;

```
int main(int argc, char *argv[])
{
        int    fd, next_option, havearg = 0;
        char *device;
        int i=0,j=0;
        int nread;                /* Read the counts of data */
        char buff[512];           /* Recvice data buffer */
```

```
        pid_t pid;
        char *xmit = "1234567890"; /* Default send data */
        int speed ;
        const char *const short_options = "hd:s:b:";

        const struct option long_options[] = {
                { "help",     0, NULL, 'h'},
                { "device", 1, NULL, 'd'},
                { "string", 1, NULL, 's'},
                { "baudrate", 1, NULL, 'b'},
                { NULL,         0, NULL, 0    }
        };

        program_name = argv[0];
        do {
                next_option = getopt_long (argc, argv, short_options,
long_options, NULL);
                switch (next_option) {
                        case 'h':
                                print_usage (stdout, 0);
                        case 'd':
                                device = optarg;
                                havearg = 1;
                                break;
                        case 'b':
                                speed = atoi(optarg);
                                break;
                        case 's':
                                xmit = optarg;
                                havearg = 1;
                                break;
                        case -1:
                                if (havearg)    break;
                        case '?':
                                print_usage (stderr, 1);
                        default:
                                abort ();
                }
        }while(next_option != -1);

        sleep(1);
        fd = OpenDev(device);
        if (fd > 0) {
```

```
            set_speed(fd, speed);
        } else {
            fprintf(stderr, "Error opening %s: %s\n", device, strerror(errno));
            exit(1);
        }
        if (set_Parity(fd,8,1,'N')== FALSE) {
            fprintf(stderr, "Set Parity Error\n");
            close(fd);
            exit(1);
        }
        pid = fork();
        if (pid < 0) {
            fprintf(stderr, "Error in fork!\n");
        } else if (pid == 0){
            while(1) {
                printf("%s SEND: %s\n",device, xmit);
                write(fd, xmit, strlen(xmit));          // cyclic write
                sleep(1);
                i++;
            }
            exit(0);
        } else {
            while(1) {
                nread = read(fd, buff, sizeof(buff));// cyclic read
                if (nread > 0) {
                    buff[nread] = '\0';
                    printf("%s RECV %d total\n", device, nread);
                    printf("%s RECV: %s\n", device, buff);
                }
            }
        }
        close(fd);
        exit(0);
}
```

7） Execute the following instructions in Ubuntu system to extact uart.tar.bz2 from **Source\App\uart.tar.bz2** to **\work** and then compile the file

- **cd $HOME/work**
- **tar -xvf /media/cdrom/Source/App/uart.tar.xz**

- **cd uart**
- **make**