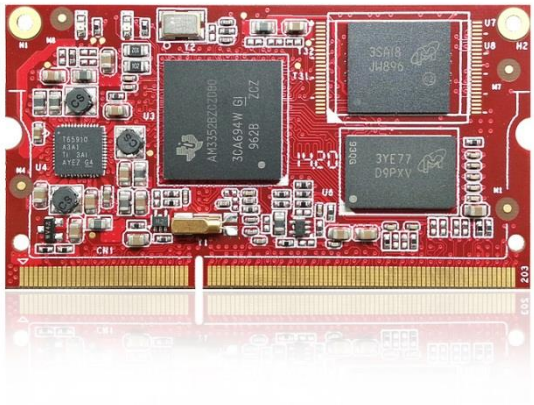
Linux 下按键测试程序开发 v1.0

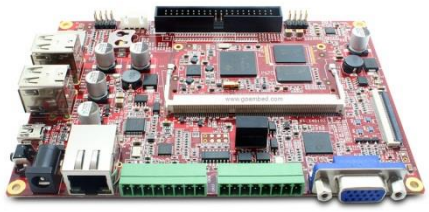
基于 TI AM335x 核心平台



免责声明

本文档是作者对 GOEMBED 产品进行实际操作和测试后，自我心得总结。建议读者具备一定的计算机基础和基本软件操作能力，如在操作过程中，遇到疑问和错误，欢迎加 QQ 群(462424566)交流，或发厂商技术支持邮箱进行咨询: support@goembed.com

操作环境配套说明：

硬件	详细介绍链接
SBC3358-B1A 单板机	
串口调试器：COM10U	

软件	详细介绍链接
Ubuntu 版本：12.04 LTS（64bit）	http://www.ubuntu.org.cn/download/desktop
Linux 版本：3.11.0-15-generic	
gcc 版本：4.6.3	

SBC3358-B1A 单板机软件特性

1、BootLoader 版本：u-boot-2013.01.01

2、内核版本：linux-3.2.0

- LCD 驱动
- LCD 背光驱动
- 电阻式触摸屏驱动
- VGA 驱动
- HSMMC/SD/MMC/SDIO 驱动
- IIC 驱动
- SPI 驱动
- 音频驱动
- DMA 驱动
- RTC 实时时钟驱动
- 电源管理
- USB HOST/DEVICE 驱动
- USB OTG 驱动
- DEBUG 驱动
- 以太网驱动
- TF 卡驱动
- CAN 驱动
- 串口驱动
- WG 驱动

3、交叉工具链：arm-linux-gnueabihf-gcc

SBC3358-B1A 单板机资源分配特性

1、emmc 空间分配

Partition	Size	Description
BootLoader	**200MB**	**FAT32 格式分区**
rootfs	**约 1500MB**	**EXT3 格式分区**

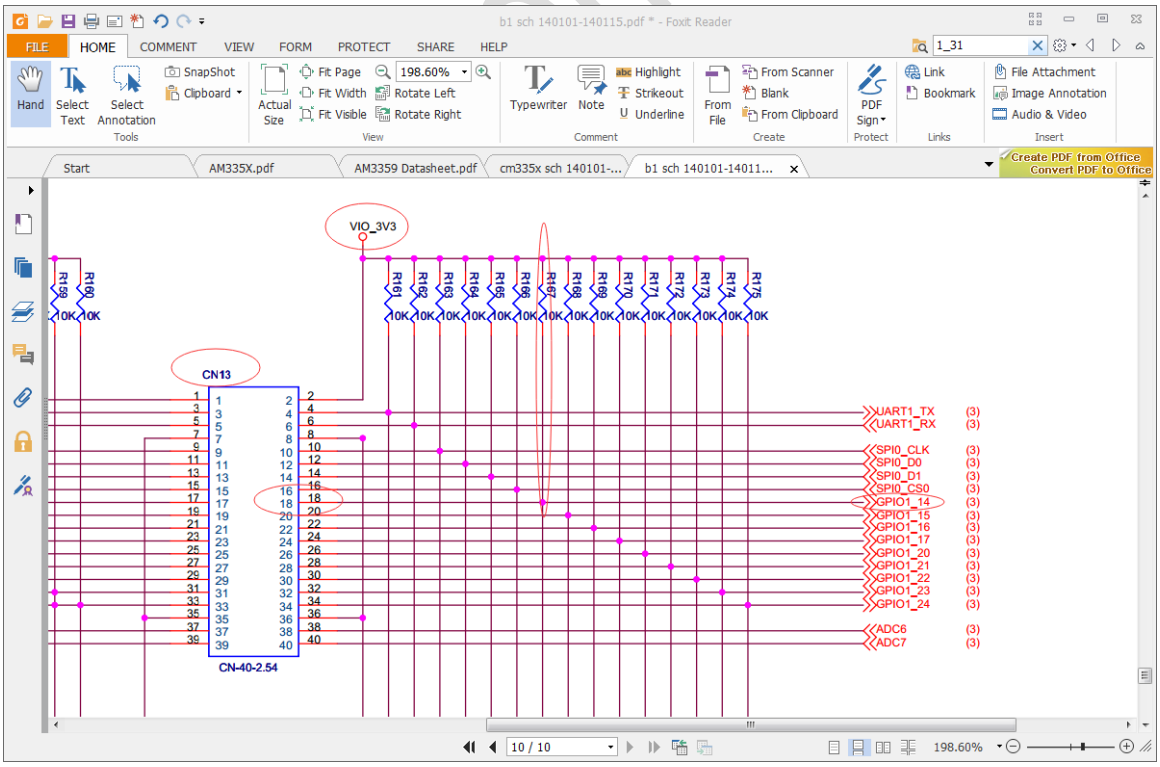
一、准备工作

1、参考《TI AM335x 搭建 Linux 开发环境 v1.0.docx》和《TI AM335x Linux 系统编译 v1.0.docx》把开发环境搭建好。

2、为了方便阅读和修改代码，在这里我使用的是 Source Insight(一个代码编辑工具)对代码进行修改。用户可以直接在终端使用 VI 编辑器编辑代码，结果是一样的，这里是为了阅读方便。

二、分析原理图

首先打开 SBC3358-B1A 的底板原理图，我们发现 SBC3358-B1A 的底板并没有设置普通按键（只有一个 Reset 按键），但是板子还是预置了非常多 GPIO 口的，我们随便选择其中一个来进行按键测试。先看下图：



我们可以发现，GPIO1_14（类似的旁边还有很多，我们取一个做例子）通

过 CN13 的 pin18 连接到核心板上，当 GPIO1_14 引脚为高电平时 pin18 为高，当 GPIO1_14 为低电平时 pin18 也为低，也就是说我们完全可以用 GPIO1_14 来模拟一个按键。实际操作中我们可以使用一根导线接在 pin18，当把 pin18 接地时模拟按键按下，当不接地时相当于未按下。沿着这个思路我们接着往下做。

三、修改代码

1、修改“gpio_keys_pin_mux”

由于 SBC3358-B1A 底板没有自带普通按键，因此出厂镜像也没有添加按键的平台设备，我们需要新增 KEY 的平台设备。该文件是：/home/goembed/335x/work/linjia/linux-3.2.0-psp04.06.00.08.sdk/arch/arm/mach-omap2/board-cm335x.c

打开 board-cm335x.c，找到 gpio_keys_pin_mux 结构体数组变量，增加要测试的 GPIO1_14，

代码为：

```
//GPIO1_14{
{"gpmc_ad14.gpio1_14",    OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
//GPIO1_14}
```

如下图：

```
00641:     {"uart0_rtsn.d_can1_rx", OMAP_MUX_MODE2 | AM33XX_PIN_INPUT_PULLUP},
00642:     {NULL, 0},
00643: };
00644:
00645: static struct pinmux_config d_can_ia_pin_mux[] = {
00646:     {"uart0_rxd.d_can0_tx", OMAP_MUX_MODE2 | AM33XX_PULL_ENBL},
00647:     {"uart0_txd.d_can0_rx", OMAP_MUX_MODE2 | AM33XX_PIN_INPUT_PULLUP},
00648:     {NULL, 0},
00649: };
00650: #else
00651: static struct pinmux_config d_can_pin_mux[] = {
00652:     {"uart1_ctsn.d_can0_tx", OMAP_MUX_MODE2 | AM33XX_PULL_ENBL},
00653:         {"uart1_rtsn.d_can0_rx", OMAP_MUX_MODE2 | AM33XX_PIN_INPUT_PULLUP},
00654:         {NULL, 0},
00655: };
00656: #endif
00657:
00658: /* pinmux for gpio based key */
00659: static struct pinmux_config gpio_keys_pin_mux[] = {
00660:         {"gpmc_csn1.gpio1_30",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
00661:         {"gpmc_csn2.gpio1_31",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
00662:         {"gpmc_ad8.gpio0_22",   OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
00663:
00664:         //GPIO1_14{
00665:
00666:         {"gpmc_ad14.gpio1_14",   OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
00667:
00668:         //GPIO1_14}
00669:
00670:         {NULL, 0},
00671: };
00672:
00673: /* pinmux for led device */
00674: static struct pinmux_config gpio_led_mux[] = {
00675: //        {"gpmc_a10.gpio1_26",  OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
00676:   //      {"gpmc_a11.gpio1_27",  OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
00677:       {"gpmc_be1n.gpio1_28", OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
00678:         {NULL, 0},
00679: };
```

其中，“gpmc_ad14.gpio1_14”是怎么得到的呢？我们可以查看技术手册：

			mmc1_dat5	2	I/O		
			mmc2_dat1	3	I/O		
			eQEP2B_in	4	I		
			pr1_mii0_txd1	5	O		
			pr1_pru0_pru_r30_15	6	O		
			gpio1_13	7	I/O		
W17	V13	GPMC_AD14	gpmc_ad14	0	I/O	L	L
			lcd_data17	1	O		
			mmc1_dat6	2	I/O		
			mmc2_dat2	3	I/O		
			eQEP2_index	4	I/O		
			pr1_mii0_txd0	5	O		
			pr1_pru0_pru_r31_14	6	I		
			gpio1_14	7	I/O		

Copyright © 2011–2012, Texas Instruments Incorporated

Submit Documentation Feedback

Product Folder Link(s): AM3359 AM3358 AM3357 AM3

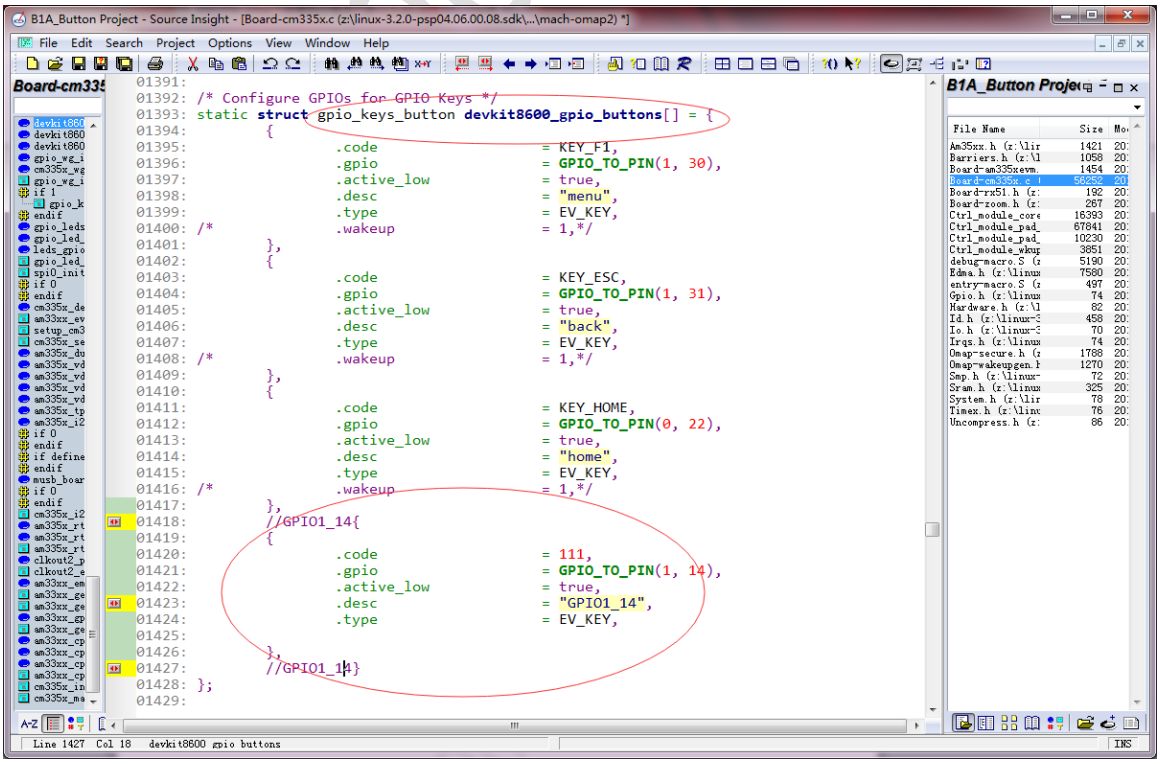
从手册可以看出，由于很多 GPIO 口都有复用功能，“gpmc_ad14.gpio1_14”是指使用 GPIO 功能，也可以看出 MODE 应为 MODE7，也就是“OMAP_MUX_MOD

E7 ”，然后设置为输入引脚“AM33XX_PIN_INPUT”。

2、修改“devkit8600_gpio_buttons”

在“devkit8600_gpio_buttons”结构体数组变量中新增以下代码：

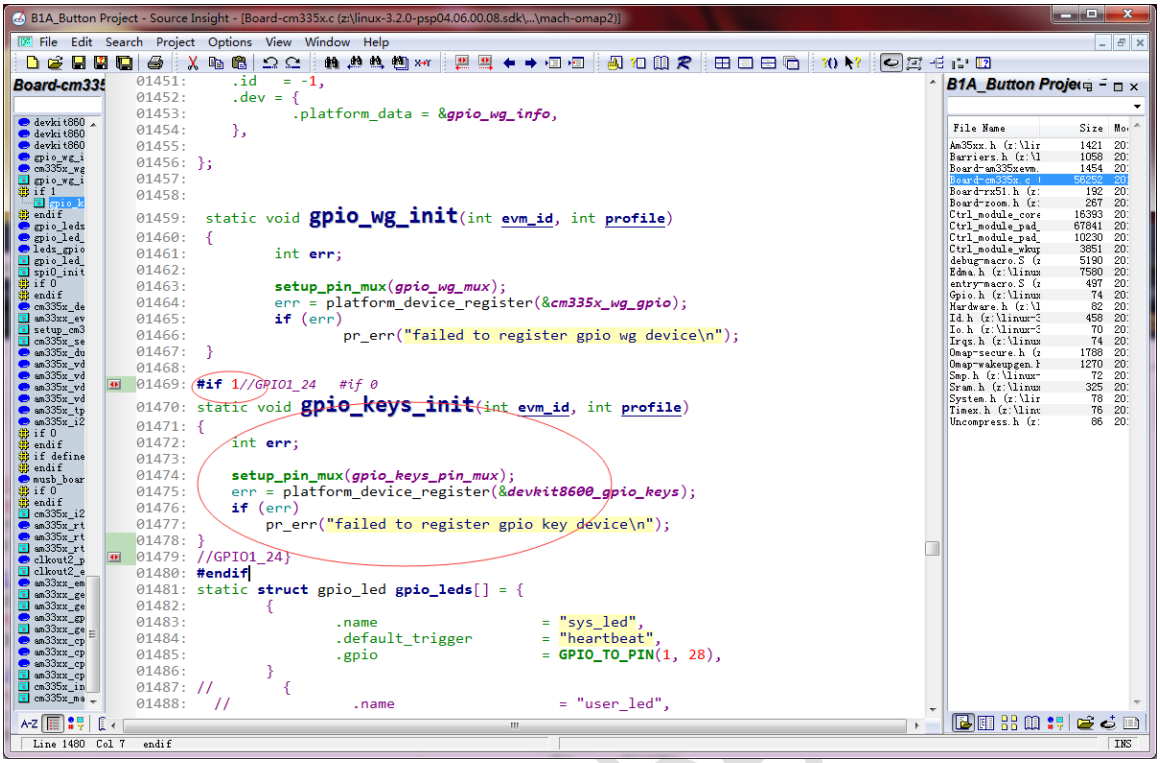
```
//GPIO1_14{
{
    .code           = 111,
    .gpio           = GPIO_TO_PIN(1, 14),
    .active_low       = true,
    .desc           = "GPIO1_14",
    .type           = EV_KEY,
},
//GPIO1_14}
```



其中 code 我们暂取为“111”,gpio 则对应 GPIO1_14 修改为 GPIO_TO_PIN(1, 14)，active_low 为“true”表示低电平有效，desc 暂取为“GPIO1_14”，EV_KEY 表示按键事件。

3、把 gpio_keys_init()函数启用：（原本是#if 0）

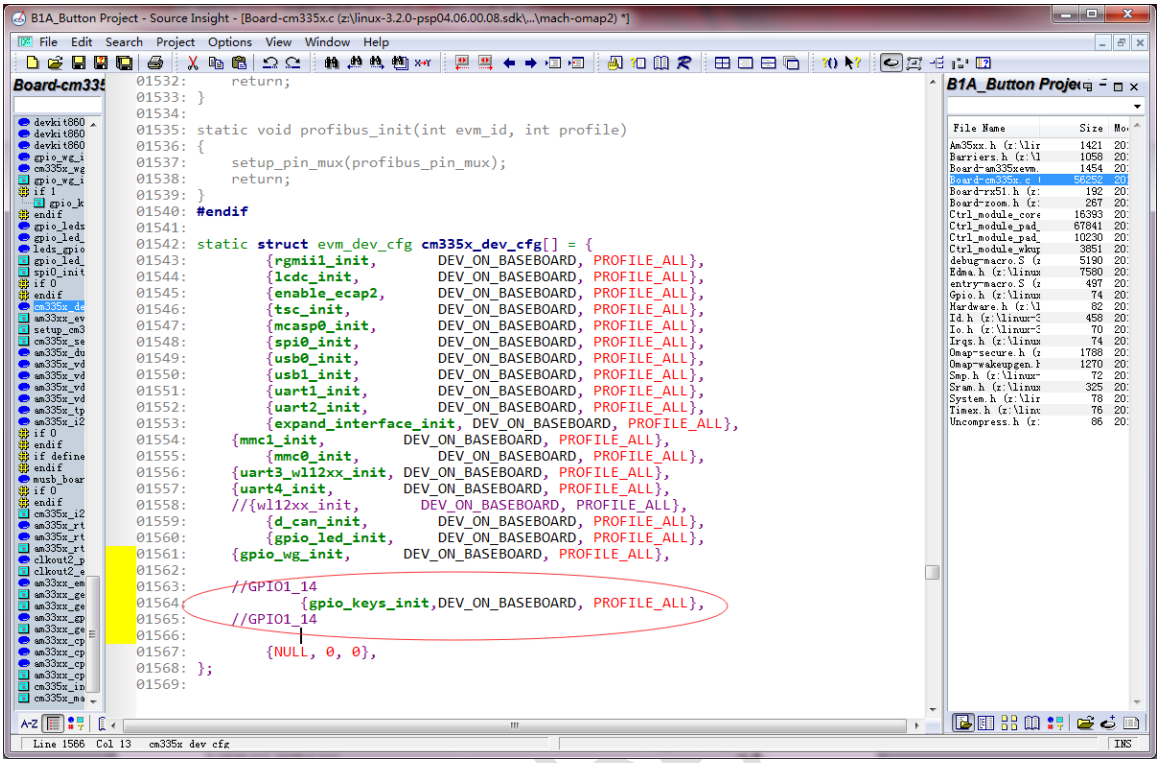
```
#if 1  //GPIO1_14   #if 0
static void gpio_keys_init(int evm_id, int profile)
{
   int err;
   setup_pin_mux(gpio_keys_pin_mux);
   err = platform_device_register(&devkit8600_gpio_keys);
   if (err)
      pr_err("failed to register gpio key device\n");
}
//gpio1_14}
```



gpio_keys_init()函数最大的最用是注册 KEY 的平台驱动。我们上面定义的 devkit8600_gpio_buttons 在这里被调用，因此 gpio_keys_init()这个函数后面必须调用。

4、在 cm335x_dev_cfg 中加代码：

```
//GPIO1_14
    {gpio_keys_init,DEV_ON_BASEBOARD, PROFILE_ALL},
//GPIO1_14
```



到此，内核代码修改完毕，我们需要重新编译内核并烧写到启动卡。

四、编译新的内核

参考《TI AM335x Linux 系统编译 v1.0.docx》重新编译内核镜像，这里只给出最重要的命令：

- export PATH=$HOME/i686-arago-linux/usr/bin/:$PATH
- make O=am335x CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm distclean
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- cm335x_tisdk_defconfig
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- uImage

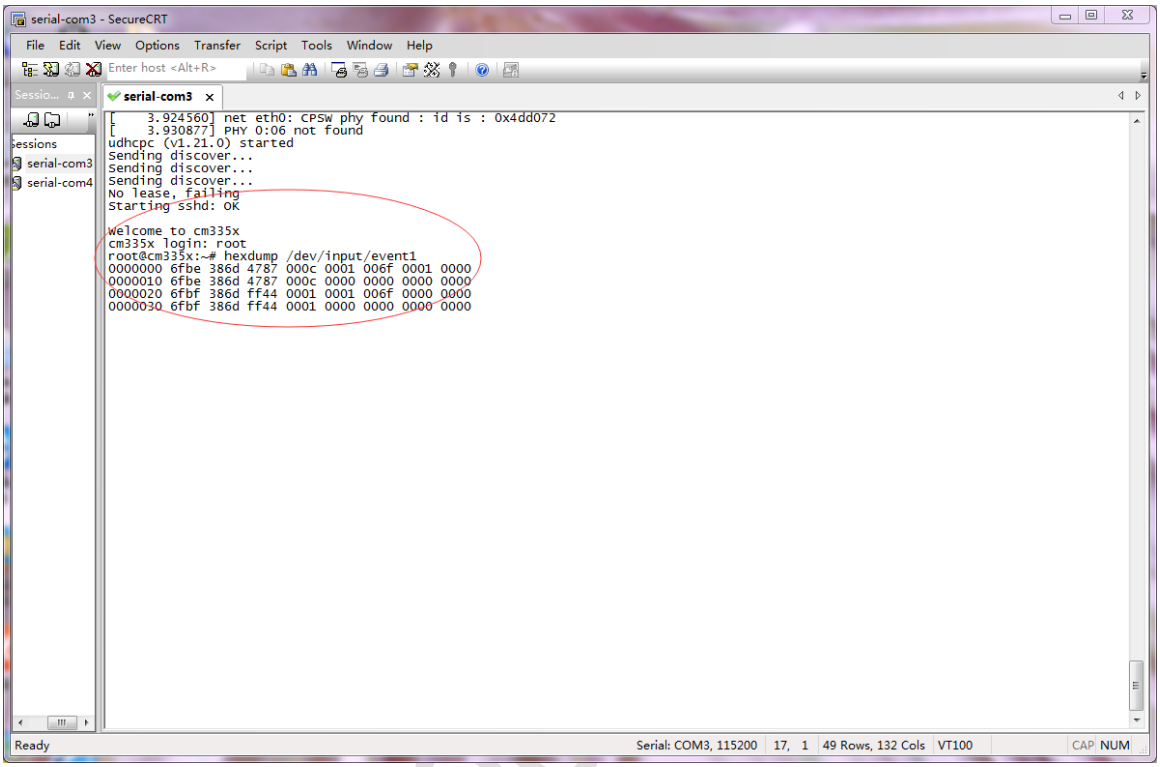
接下来用生成的新的 uImage 替换原来的 uImage，然后重启系统。

五、测试新的内核

SecureCRT.exe 中输入“root”登录，然后输入：

```
hexdump /dev/input/event1
```

当用导线将 GPIO1_14 接地时可以看到如下打印信息（由于按键抖动，终端会打印出许多信息，后面由测试程序进行消抖）：



能看到以上信息说明新增的 KEY 没问题，接下来编写测试程序即可。

六、编写按键测试程序

在这里我写了一个测试程序，功能如下：当按键按下后，判断是长按（>1s）还是短按（<=1s），并且能把 0.05s 内产生的按键抖动过滤。

程序如下：

```
#include <stdint.h>
#include <linux/version.h>
#include <linux/input.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
#include <unistd.h>
int main(int argc, char **argv)
{
    int fd_anjian;
    struct input_event t;

    unsigned int backlight_value=0x40;
    int ret=0;
    fd_anjian = open("/dev/input/event1",O_RDONLY);
    if(fd_anjian <=0 )
    {
        printf("error to open event0\r\n");
        return 0;
    }
double shijiancha=0;
double  starttime;
double  endtime;
    while(1)
        {
        //printf("\n\r");
        if(read(fd_anjian,&t,sizeof(t)) == sizeof(struct input_event))
        {
            if(t.type == EV_KEY){

                endtime=(t.time.tv_sec+t.time.tv_usec/1000000.0);
                shijiancha=0;
                if(t.code == 111 && (t.value==1) ){
                starttime=(t.time.tv_sec+(t.time.tv_usec/1000000.0));
                //printf("starttimr:%f\n\r",starttime);
                //printf("t.time.tv_usec:%lf\n\r",t.time.tv_usec/1000000.0);
                //printf("now
//time:%f\n\r",((t.time.tv_usec/1000000.0)+t.time.tv_sec));
                shijiancha=0;
                    }
                if(t.code == 111 &&!t.value  ){
                shijiancha=0;
                    endtime=(t.time.tv_sec+(t.time.tv_usec/1000000.0));
                //printf("endtimr:%lf\n\r",endtime);
                }
                shijiancha=endtime-starttime;
                //printf("shijiancha:%lf\n\r",shijiancha);
```

```
                if(shijiancha>1)printf("KEY:%d long %lf
s\n\r",t.code,shijiancha);
                if(shijiancha>=0.05&&shijiancha<=1)printf("KEY:%d short %lf
s\n\r",t.code,shijiancha);
                shijiancha=0;
            }
        }
    }
    close(fd_anjian);
    return 0;
}
```

在这里我简单解释一下这个程序的流程：首先我们打开KEY对应的设备文件，也就是“/dev/input/event1”，然后进入循环等待按键按下，当有按键按下时我们先判断是不是键值为“111”，是的话记录下当前系统的时间，然后判断按键是否弹起，如果弹起则记录下弹起时的系统时间，我们用弹起时的系统时间减去按下时的系统时间得到时间差，然后判断该时间差，如果是小于0.05s的我们认为是抖动，如果是大于等于0.05秒且小于等于1s的我们认为是短按，如果是大于1s的我们认为是长按（是“长”是“短”用户可自定义，这里只是举个例子）。

注：应用程序编译要采用arm-2009q1交叉编译。（后续补上）

➢ 命令：

```
arm-none-linux-gnueabi-gcc -o GPIO1_14 GPIO1_14.c
```

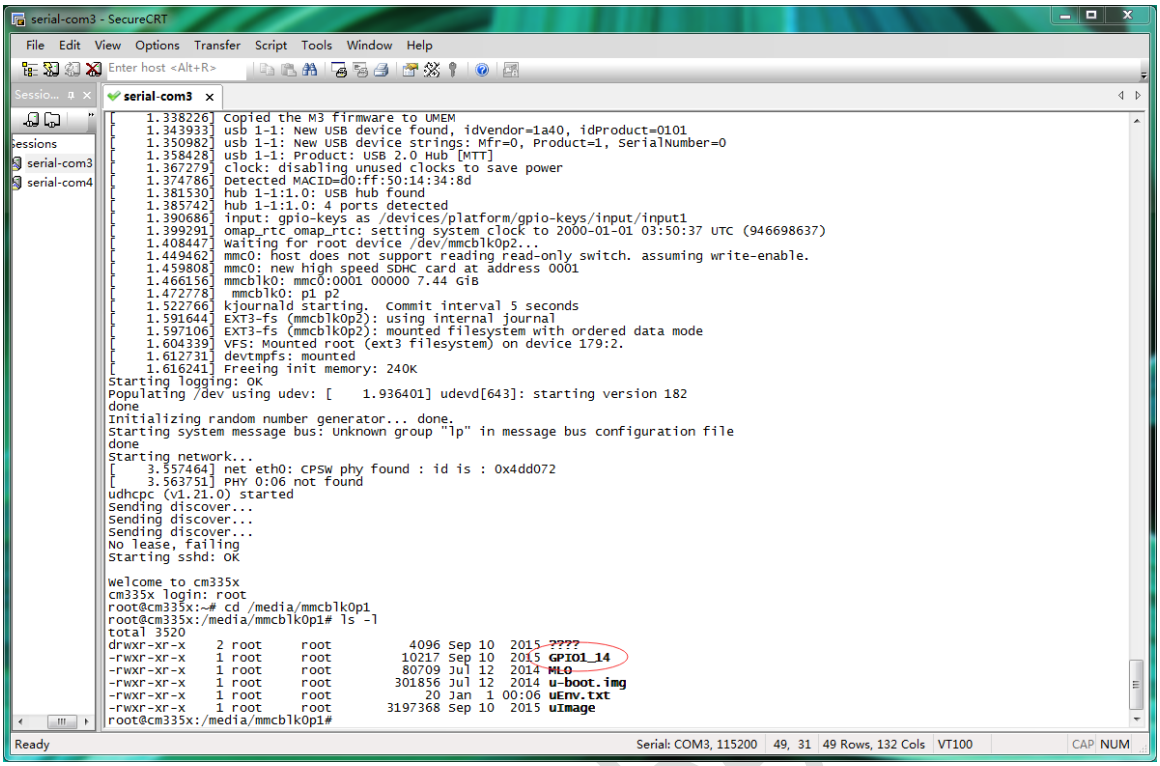
编译完成后生成可执行文件GPIO1_14,如下图：

```
lin@goembed-ubuntu: ~/project/335x/button_test
lin@goembed-ubuntu:~/project/335x/button_test$ arm-none-linux-gnueabi-gcc -o GPIO1_14 GPIO1_14.c
lin@goembed-ubuntu:~/project/335x/button_test$ ls -l
total 16
-rwxrwxr-x 1 lin lin 10217 Sep 10 08:19 GPIO1_14
-rwx------ 1 lin lin  1649 Sep 10 03:54 GPIO1_14.c
lin@goembed-ubuntu:~/project/335x/button_test$
```

七、测试按键

将生成的可执行文件 GPIO1_14 拷贝到 TF 卡中，启动系统后找到该文件：

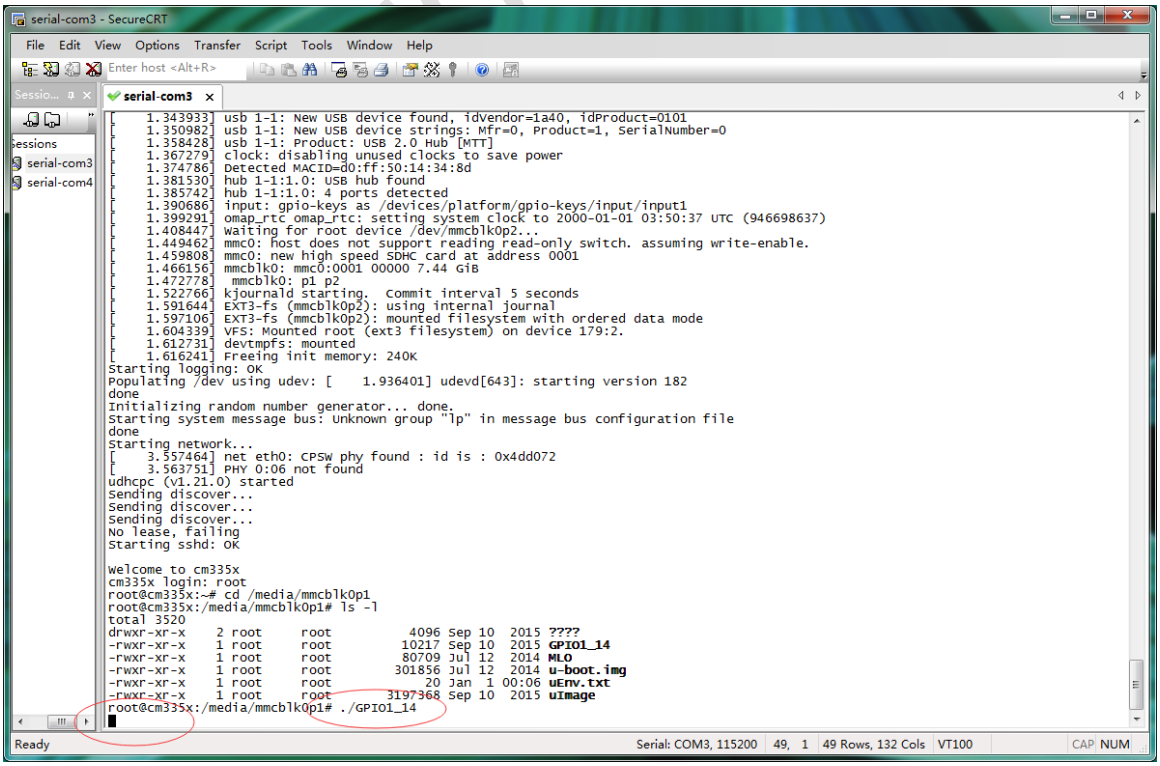
（该文件的位置根据用户启动系统的实际情况决定，在这里不一一赘述）



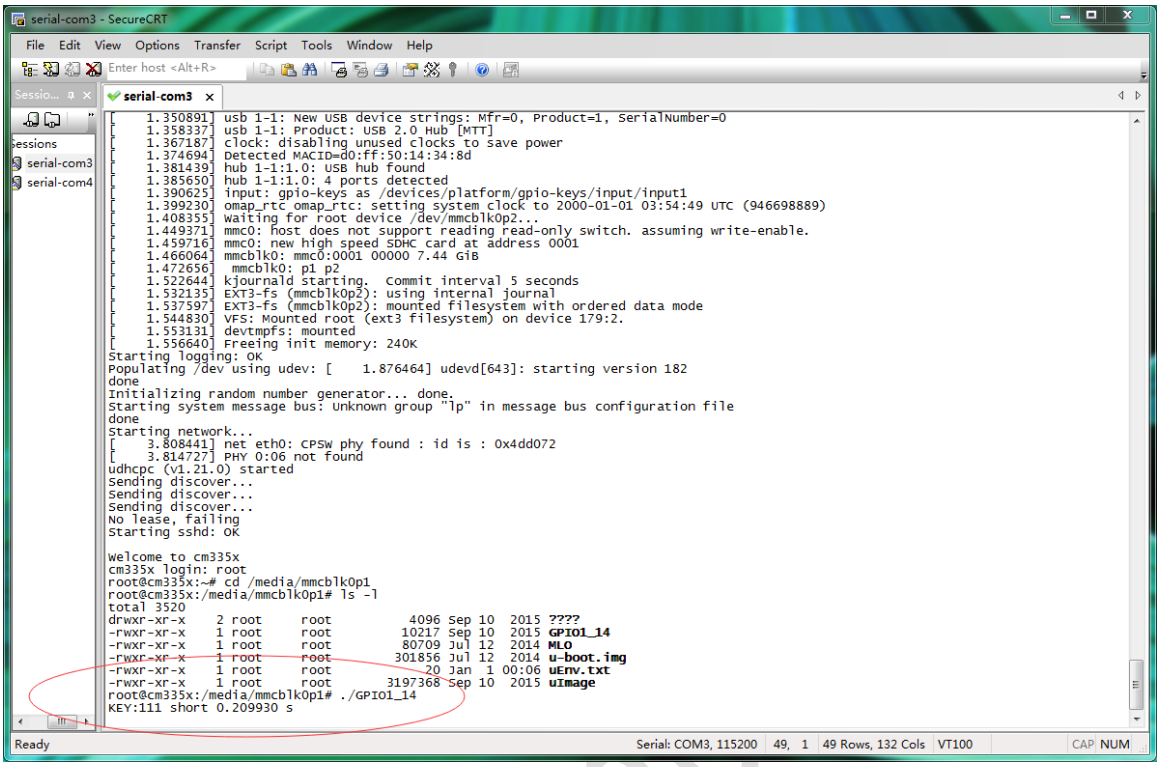
➢ 命令：

./GPIO1_14

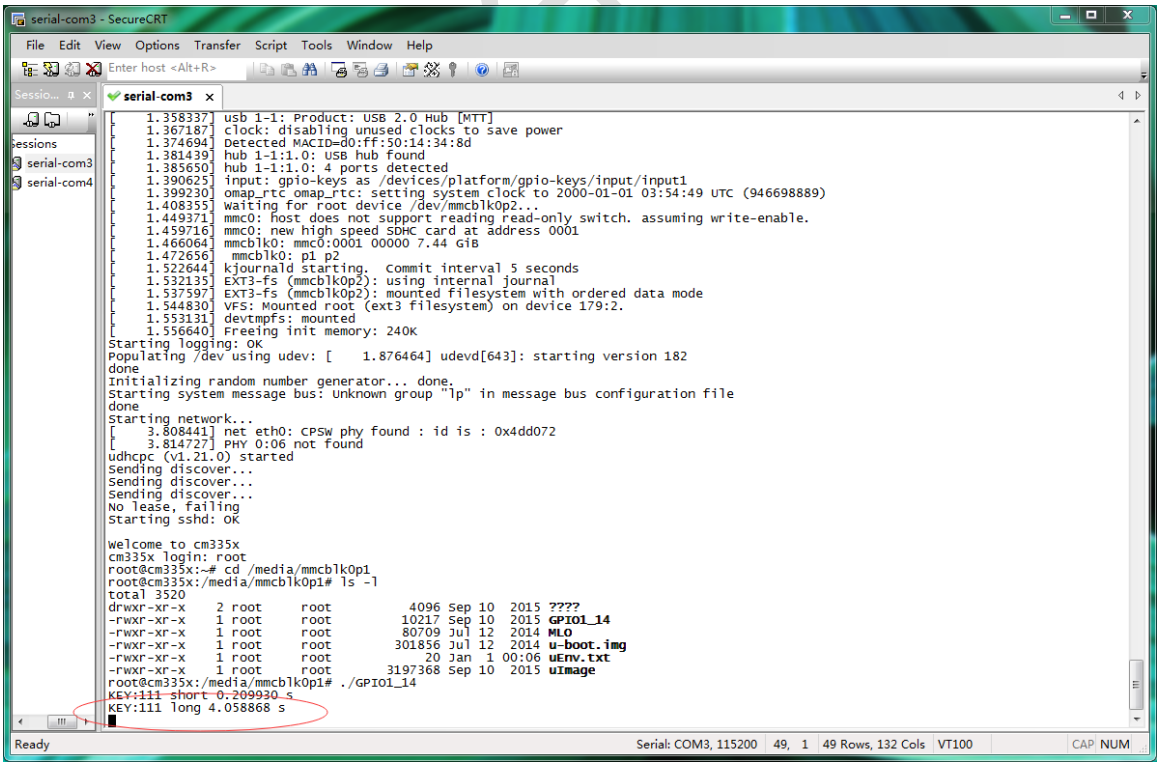
输入以上命令执行该应用程序（成功执行黑色光标在闪）：



短按时如下：



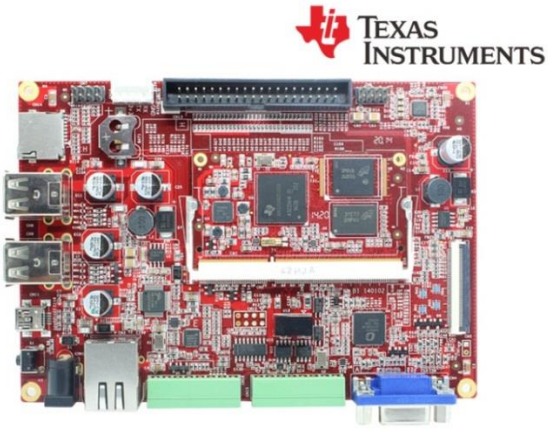
长按时如下



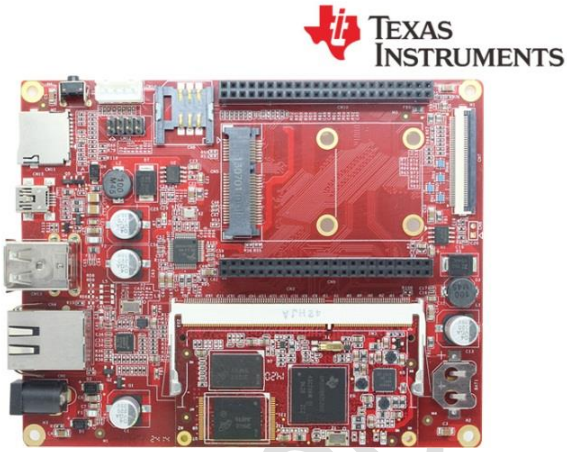
通过以上截图可以发现，按键的抖动被很好地过滤。该测试程序可以精确

判断长按和短按。

附 相关 GOEMBED 产品介绍



SBC335x – B1A



SBC335x – B2A

The single board computer SBC335x-B1A/B2A which has an expansion board to carry the CM335X is one of our design of the base plate . The flexible design allows the fast and easy way of realizing and upgrading the controller's capabilities. In additional to those features offered by CM335X.

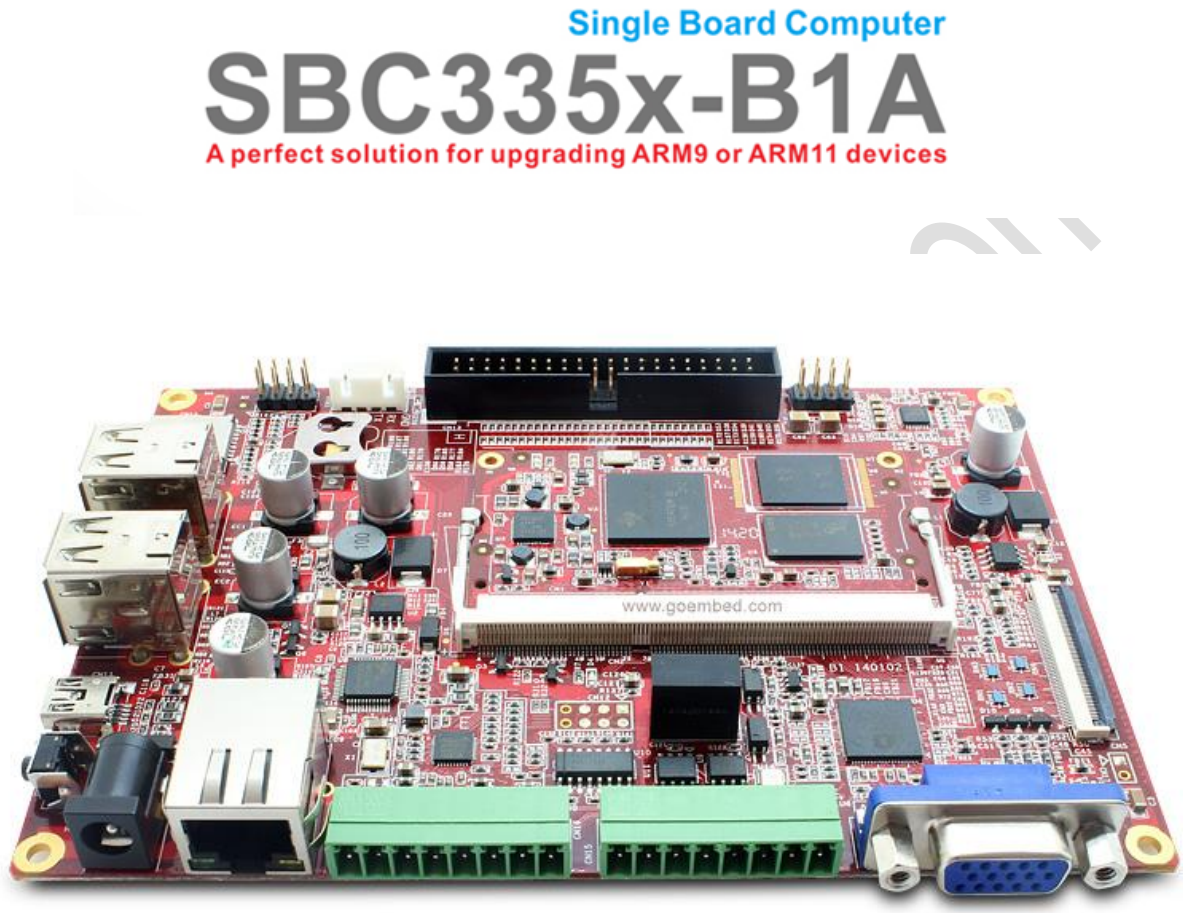
The B1A features 4 serial ports (including 2 RS232 and 2 TTL), 4 USB Host and 1 USB OTG, 1 Ethernet ports, CAN, RS485, Wiegand, VGA, LCD, Touch screen, Audio, ADC and more other peripherals.

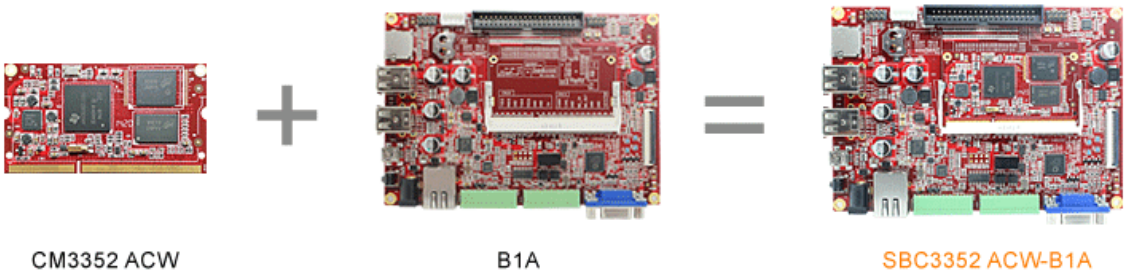
The B2A features 4 USB Host and 1 USB OTG, 1 Ethernet ports, LCD, Touch screen,RTC, and more other peripherals.

The SBC board targets a wide range of applications, including: HMIs, Digital Signage, POS, Data Terminal, Medical Devices, Navigation, Industrial Automation, Entertainment system, Thin Clients, Robotics, Game Console and much more.

The SBC335x-B1A/B2A are ready-to-run platform to support Linux 3.x, Android 4.x and WinCE 7.0/6.0 operating systems.

If you want to support other Operating System, For more information to contact us.





SBC335x-B1A boards Description of part code:

Series	B1	B1	B1	B1
Part Code	SBC3352 ACW-B1A	SBC3352 BCW-B1A	SBC3358 ACW-B1A	SBC3358 BCW-B1A
Order Code	-	-	-	-
Core Module	CM3352 ACW -M51E20/08	CM3352 BCW -M51E40/08	CM3358 ACW -M51E20/10	CM3358 BCW -M51E40/10
CPU Type	ARM Cortex™-A8			
CPU Cores	1x			
CPU Clock	800MHz	800MHz	1.0GHz	1.0GHz
RAM DDR3	Micron 512MB@16bit*1			
eMMC Flash	2GB@8bit*1	4GB@8bit*1	2GB@8bit*1	4GB@8bit*1
PMU	TI TPS65910A3			
Supply Voltage	DC 9-14V			
Optimal Input	DC 12V,1.5A			
Size(L*W)	146 x 102 mm			
Temperature	0° to 70° C			
Support OS	Linux 3.x/ Android 4.x/ Ubuntu/ Angstrom/ Debian/ QT/ WinCE 6.0/7.0			
Inventory status	In Stock	Out of Stock Contact us	In Stock	Out of Stock Contact us
Minimum Availability	2022			

SBC335x-B1A Block Diagram

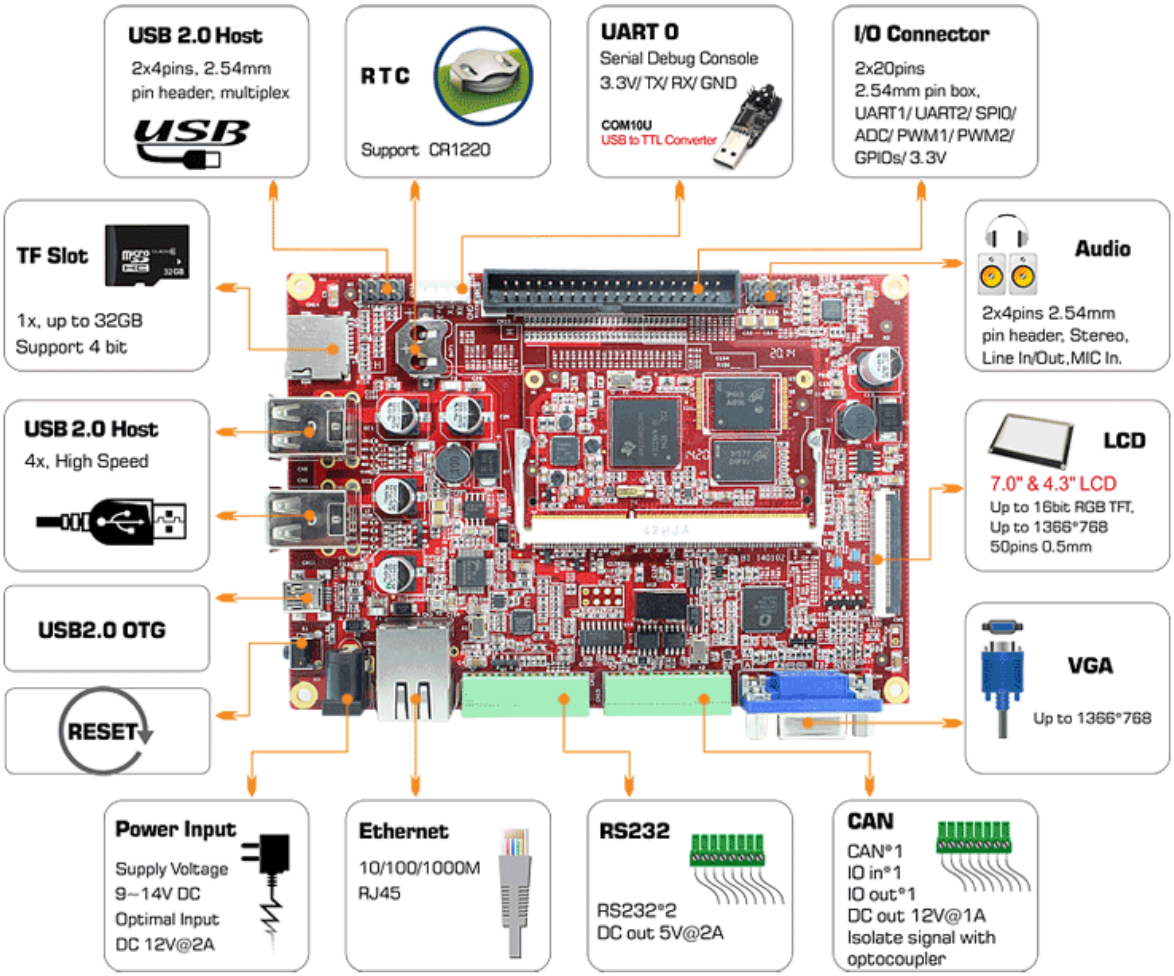


Figure 1 B1 Block Diagram

SBC335x-B2A

Single Board Computer

CM3358 ACW + B2A = SBC3358 ACW-B2A

SBC335x-B2A boards Description of part code:

Series	B2A	B2A	B2A	B2A
Part Code	SBC3352 ACW-B2A	SBC3352 BCW-B2A	SBC3358 ACW-B2A	SBC3358 BCW-B2A
Order Code	-	-	-	-
Core Module	CM3352 ACW -M51E20/08	CM3352 BCW -M51E40/08	CM3358 ACW -M51E20/10	CM3358 BCW -M51E40/10
CPU Type	ARM Cortex™-A8			
CPU Cores	1x			
CPU Clock	800MHz	800MHz	1.0GHz	1.0GHz
RAM DDR3	Micron 512MB@16bit*1			
eMMC Flash	2GB@8bit*1	4GB@8bit*1	2GB@8bit*1	4GB@8bit*1
PMU	TI TPS65910A3			
Supply Voltage	DC 9-14V			
Optimal Input	DC 12V,1.5A			
Size(L*W)	130 x 103.5 mm			
Temperature	0° to 70° C			
Support OS	Linux 3.x/ Android 4.x/ Ubuntu/ Angstrom/ Debian/ QT/ WinCE 6.0/7.0			
Inventory status	In Stock	Out of Stock Contact us	In Stock	Out of Stock Contact us
Minimum Availability	2022			

SBC335x-B2A Block Diagram

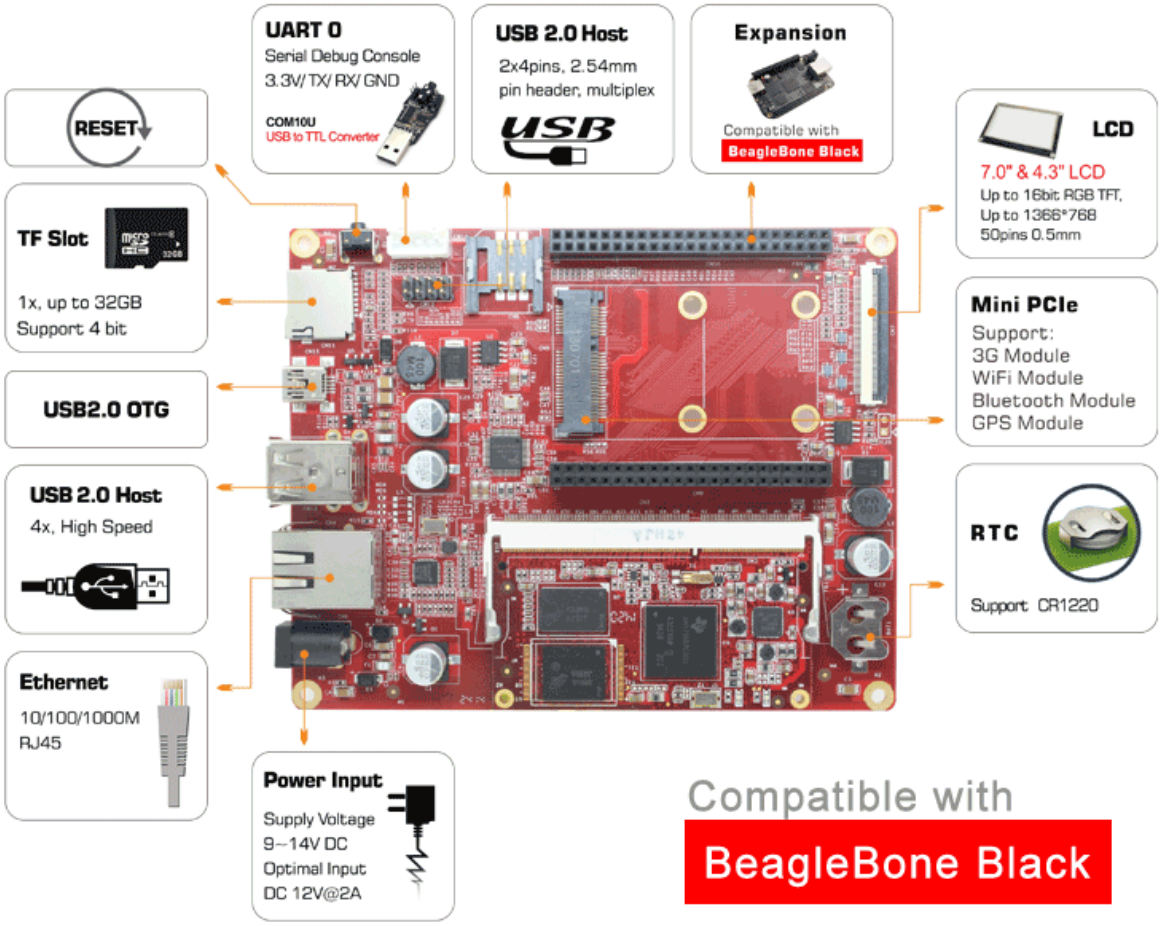


Figure 1 B2A Block Diagram

ABOUT GOEMBED

GOEMBED team with experienced embedded engineers who have been engaged in ARM hardware and software design for 10+ years.

Our products include single board computers and CPU core modules based on TI ® Sitara and Freescale ® i.MX Applications Processors based on ARM® Cores. Supported by Linux / Android / Debian / Ubuntu / QT / Angstrom / WinCE 7.0 & 6.0 / uCOS. We can redesign carrier boards and SBC as your idea quickly.

GOEMBED focus on Embedded Board Solutions, provide a complete new board for your specified requirement or even a turnkey solution to accelerate your new products to market.

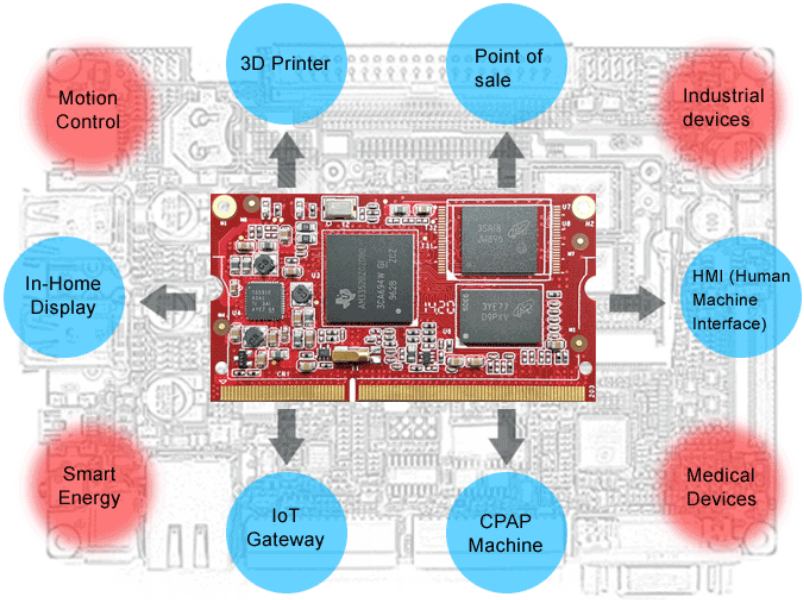
We are your trust worthy partner on ARM embedded design services and solutions.

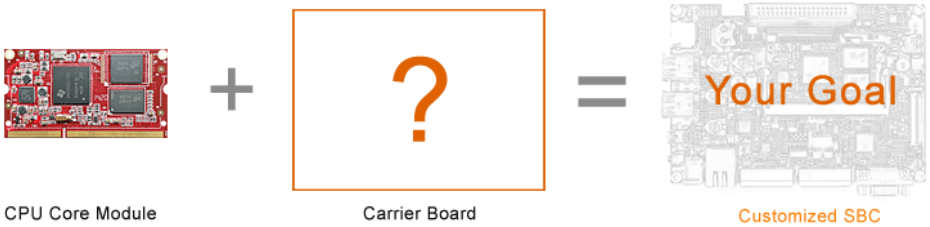
More Carrier Boards

Customized based on your needs!

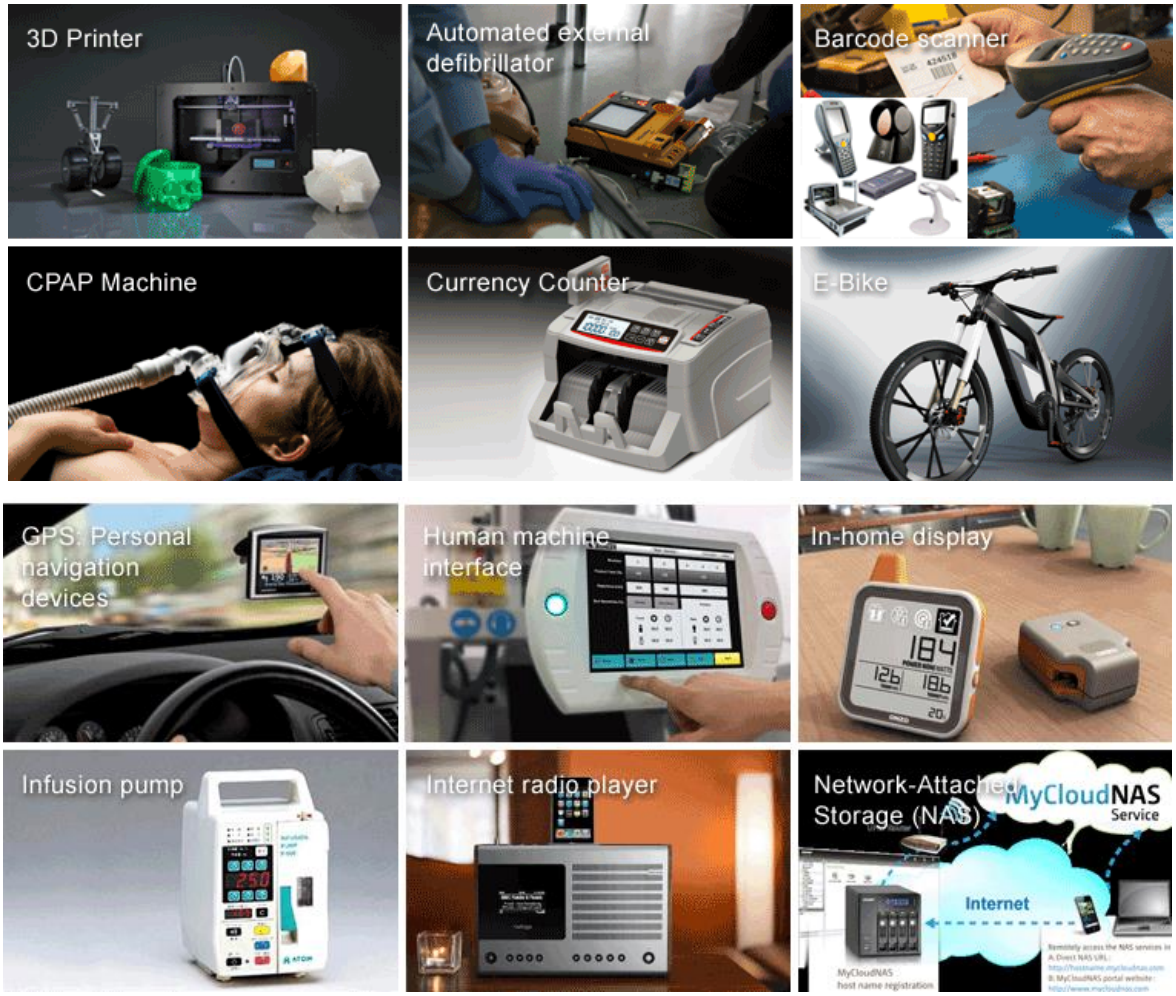
ODM / OEM Services

Bring your new products to market quickly



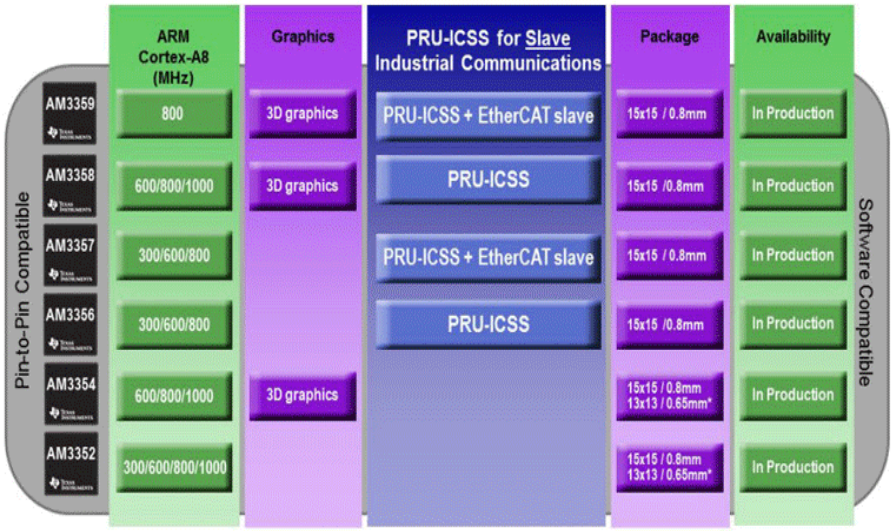


Related end equipment



Learn more applications please click http://www.ti.com/lsds/ti/apps/appshomepage.page

AM335x - A scalable platform with 6 pin-to-pin compatible devices



✓PRU-ICSS is used for slave industrial communication protocols such as Profibus, Profinet, Powerlink & Ethernet/IP

Package	15x15mm (ZCZ)	13x13mm (ZCE)
ARM speed	Up to 1000 MHz	Up to 600 MHz
USB 2.0 OTG + PHY	x2	x1
EMAC	2-port switch	Single port

TI Sitara ARM Cortex-A8 AM335x processors information (Content from TI):

AM335x Cortex™-A8 based processors

Benefits
- High performance Cortex-A8 at ARM9/11 prices
- Rich peripheral integration reduces system complexity and cost

Sample Applications
- Industrial / Home Automation
- Portable Navigation Devices
- Robotics
- Consumer electronics
- Smart Appliances
- Low power instrumentation
- Wireless Accessories
- Networking

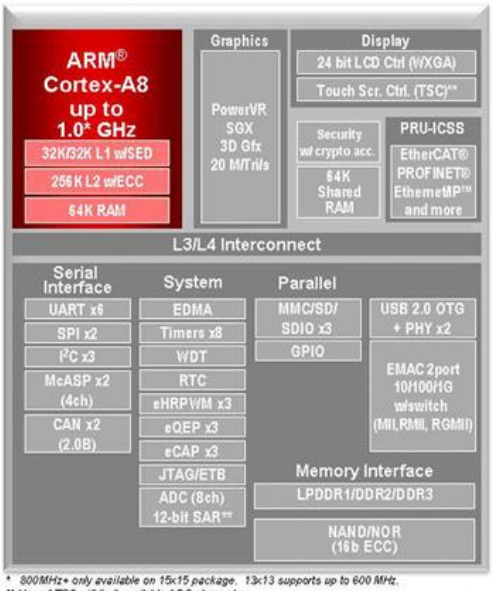
Software and development tools
- Free Linux and Android support packages direct from TI
- StarterWare enables quick and simple programming and migration among TI embedded processors
- WinCE and RTOS (QNX, Wind River, Mentor, etc.) from partners
- Full featured and low cost development board options

Power Estimates
- Total Power: 600mW-1000mW
- Standby Power: ~25mW
- Deep Sleep Power: ~5-7mW

Schedule and packaging
- Status: In production
- Dev. Tools: Available today
- Docs: Available today
- Packaging: 13x13, 0.65mm via channel array
  15x15, 0.8mm

More Information
- www.ti.com/am335x

Availability of some features, derivatives, or packages may be delayed from initial silicon availability
Peripheral limitations may apply among different packages
Some features may require third party support
All speeds shown are for commercial temperature range only



* 800MHz+ only available on 15x15 package. 13x13 supports up to 600 MHz.
** Use of TSC will limit available ADC channels.
SED: single error detection/parity

1

Texas Instruments