

|  |
| --- |
| Linux下CAN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，自我操作总结。由于作者水平有限，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)交流和建议，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 | [c:\users\administrator\appdata\roaming\360se6\User Data\temp\1411389502416719.jpg](http://www.goembed.com/index.php/Products/detail/tpid/25) |
| 串口调试器：COM10U | [c:\users\administrator\appdata\roaming\360se6\User Data\temp\1408786855283579.jpg](http://goembed.com/index.php/Products/detail/tpid/16) |

|  |  |
| --- | --- |
| 软件 | 详细介绍链接 |
| Ubuntu版本：12.04 LTS（64bit） | [http://www.ubuntu.org.cn/download/desktop](http://www.ubuntu.org.cn/download/desktop%20) |
| 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

#### eMMC空间分配

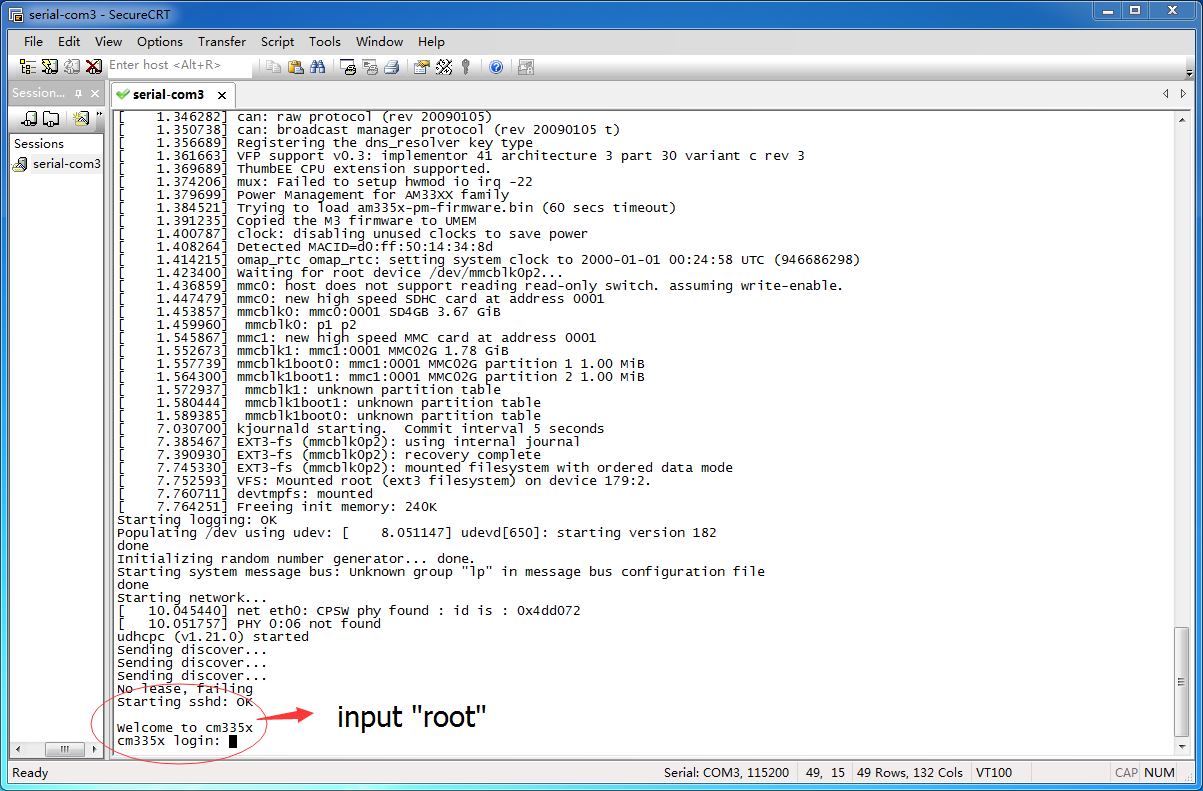
|  |  |  |
| --- | --- | --- |
| Partition | Size | Description |
| BootLoader | **200MB** | **FAT32格式分区** |
| rootfs | **约1500MB** | **EXT3格式分区** |

### 准备工作

1、准备好已经烧好Linux系统的TF卡，且TF卡FAT分区中必须有：MLO、u-boot.img、uEnv.txt、uImage和rootfs.tar.bz2这几个文件，再把卡插到开发板中。

2、连接好USB转TTL串口模块，打开串口调试软件SecureCRT.exe。

3、开发板接上12V电源适配器，开机后串口调试软件打印如下信息：



4、输入root登录Linux系统。

5、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。

### CAN总线-简介

CAN(Controller Aea Netork),中文名称控制器局域网络，通常称为CAN bus，即CAN总线。是由德国BOSCH（博世）公司研究开发的，现已成为ISO国际标准化的串行通信协议，是目前国际上应用最广泛的开放式现场总线之一。

### CAN总线-报文

CAN总线以报文的形式发送数据，每组报文的前十一位字符为标识符（在同一系统中，标识符是唯一的），不包括具体发送数据，是对报文优先级的定义，我们将报文的这种格式称为面向内容的编址方案。

### CAN总线-帧结构

帧可分为数据帧、远程帧、错误帧和过载帧四种类型。CAN总线上传输的大部分都是数据帧，数据帧负责携带数据从发送器到接收器；远程帧是由总线单元出发的，负责请求发送相同的数据帧；错误帧可由任何单元在检测到总线错误时发出；过载帧用于在两数据帧或远程帧中提供延时。

### CAN总线-原理

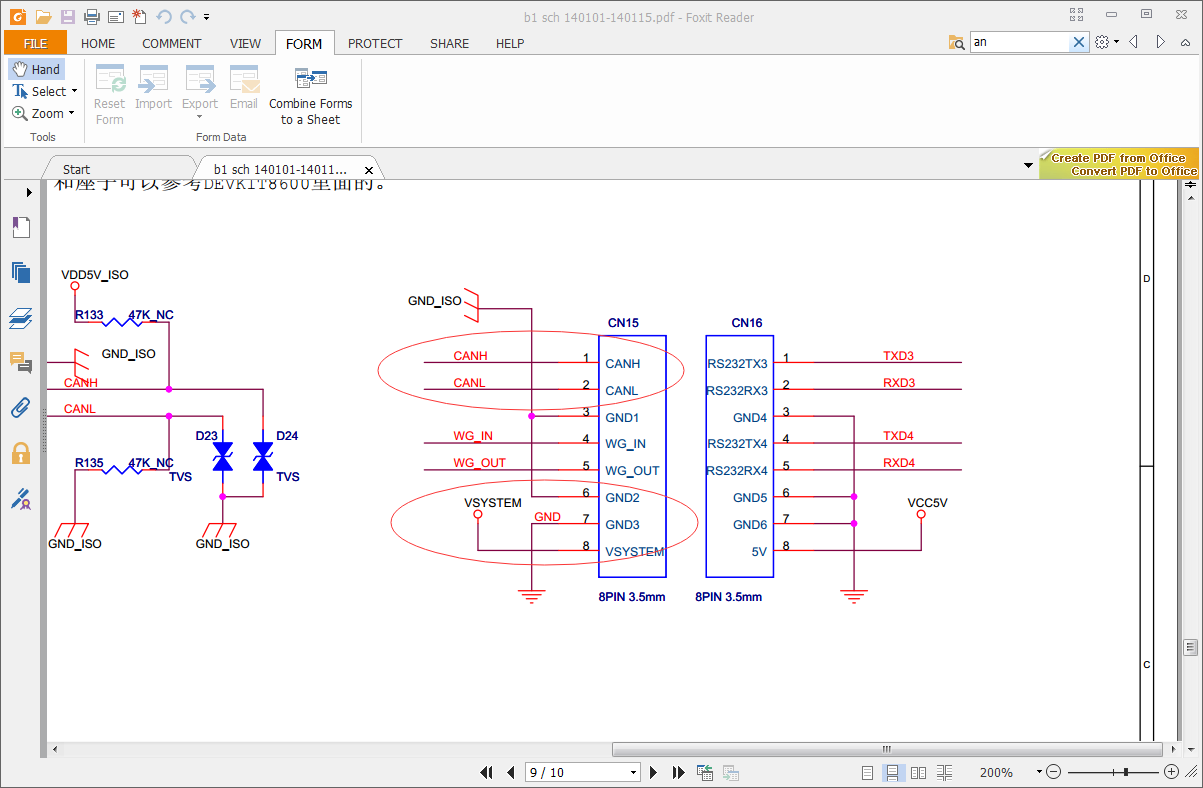
CAN总线以广播的方式从一个节点向另外一个节点发送数据。当一个节点发送数据时，该节点的CPU把将要发送的数据和标识符发送给本节点的CAN芯片，并使其进入准备状态；一旦该CAN芯片收到总线分配，就变为发送报文状态，该CAN芯片将要发送的数据组成规定的报文格式发出。此时，网络中的其他节点都处于接收状态，所有节点都要先对其进行接收，通过检测来判断该报文是否是发给自己的。

由于CAN 总线是面向内容的编址方案，因此容易构建控制系统对其灵活地进行配置，使其可以在不修改软硬件的情况下向CAN总线中加入新节点。

参考：<http://www.eepw.com.cn/article/269402.htm>

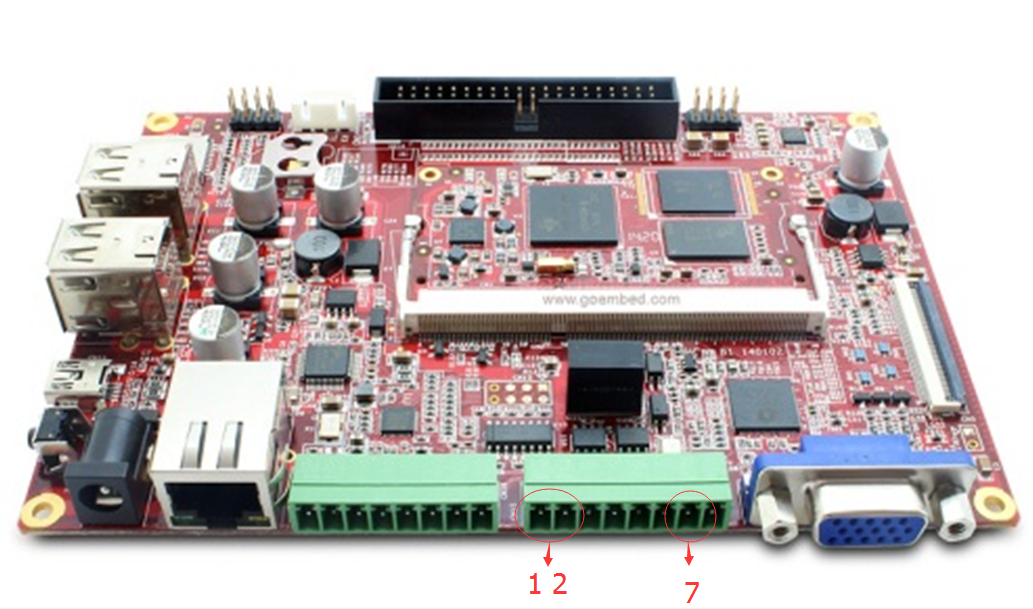
### 接线方法

由于CAN测试是两个板子之间的通信，这里我简单介绍一下接线的方法。我们先来查看一下CAN部分原理图：



在这里我们只需要接三根线，CANH（pin 1）、CANL（pin 2）和GND（pin 7）。

在B1A板子上对应以下引脚：



首先应该先把A（发送端）和B（接收端）开发板的CAN口用线连接起来，连接方法为：CANH对CANH，CANL对CANL，GND对GND。然后分别接上电源和串口调试器，接线方法不再赘述。

### CAN测试流程

#### 1、A（发送）端代码（除头文件包含外）如下：

|  |
| --- |
| A（发送）端代码 |
| #define CAN\_INIT "canconfig can0 bitrate 125000 ctrlmode triple-sampling on"  #define CAN\_EN "canconfig can0 start"  #define CAN\_RECIVE "candump can0 1>/tmp/can"  #define CAN\_SEND "cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88"  #define CAN\_STOP "canconfig can0 stop 2>>/tmp/can"  #define CMD\_GREP "cat /tmp/can | grep \"11 22 33 44 55 66 77 88\" 1>/tmp/can\_info"  #define ERRO\_INFO "/tmp/can\_info"  void \*thread\_function(void \*arg)  {  int num\_read;  printf("time start!\n\r");  sleep(1);  printf("1s\n\r");  sleep(1);  printf("2s\n\r");  sleep(1);  printf("3s\n\r");  sleep(1);  printf("4s\n\r");  sleep(1);  printf("5s\n\r");  system(CAN\_STOP);  pthread\_exit("can stop!");  }  int main(void)  {  char buff[256];  struct stat stat\_buf;  pid\_t pid;  pthread\_t a\_thread;  int res = 0;  system (CAN\_INIT);  system (CAN\_EN);  usleep (200000);  system (CAN\_SEND);  usleep(50000);  res = pthread\_create(&a\_thread, NULL, thread\_function, NULL);  if (res != 0){  perror("Thread creation failed");  exit(1);  }  system (CAN\_RECIVE);  system (CMD\_GREP);  if (stat (ERRO\_INFO, &stat\_buf) < 0)  return -1;  if (stat\_buf.st\_size <= 0)  return -1;  printf("this is success\n");  return 0;  } |

#### 2、B（接收）端代码（除头文件包含外）如下：

|  |
| --- |
| B（接收端代码） |
| #define CAN\_INIT "canconfig can0 bitrate 125000 ctrlmode triple-sampling on"  #define CAN\_EN "canconfig can0 start"  #define CAN\_RECIVE "candump can0"  #define CAN\_SEND "cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88"  #define CAN\_STOP "canconfig can0 stop 2>>/tmp/can"  #define CMD\_GREP "cat /tmp/can | grep \"11 22 33 44 55 66 77 88\" 1>/tmp/can\_info"  #define ERRO\_INFO "/tmp/can\_info"  void \*thread\_function(void \*arg)  {  struct stat stat\_buf;  int num\_read;  printf("time start!\n\r");  sleep(1);  printf("1s\n\r");  sleep(1);  printf("2s\n\r");  sleep(1);  printf("3s\n\r");  sleep(1);  printf("4s\n\r");  sleep(1);  printf("5s\n\r");  printf("time over\n\r");  system(CAN\_STOP);  system (CAN\_INIT);  system (CAN\_EN);  system (CAN\_SEND);  system(CAN\_STOP);  pthread\_exit("can stop!");  }  int main(void)  {  char buff[256];  struct stat stat\_buf;  pid\_t pid;  pthread\_t a\_thread;  int res = 0;  system (CAN\_INIT);  system (CAN\_EN);  usleep (200000);  res = pthread\_create(&a\_thread, NULL, thread\_function, NULL);  if (res != 0)  {  perror("Thread creation failed");  exit(1);  }  system (CAN\_RECIVE);  return 0;  } |

#### 3、A（发送）端流程分析

在main函数中，该测试程序完成CAN的初始化和使能后，发送一个数据帧到B（接收）端，然后开启一个线程进行5秒钟的计时（线程与main函数同时工作且互不影响），开启了线程之后执行system（CAN\_REVICE），由于这是一个阻塞函数所以main函数会阻塞在这里（即使收到数据也会阻塞在这里）。因此就需要有人帮助跳出这个阻塞函数，这时就要用到我们上面开启的线程。该线程的在5秒钟之后便会执行一个system（CAN\_STOP）函数，这个函数用于跳出main函数中的阻塞函数system（CAN\_RECIVE）。无论CAN是否接收到B端回复的数据都会跳出,然后执行线程退出函数。所以，无论CAN有无收到B端返回的数据，5s钟之后都会执行system（CMD\_GREP）和后面的代码。这个函数是判断B端发送回来的数据是否包含“11 22 33 44 55 66 77 88”这一个字符串，如果有包含的话就把它打印到/tmp/can\_info文件中。最后代码判断/tmp/can\_info这个文件，如果这个文件不为空，我们就认为成功接收到B端返回的数据。到此，A端代码分析结束。

#### 4、B（接收）端功能分析

在main函数中，该测试程序完成CAN 的初始化和使能后，开启一个线程（线程与main函数同时工作互不影响）。然后调用system（CAN\_REVICE）处于接收状态并且阻塞在这里。另一边，刚才开启的线程计时5s，时间到了之后无论CAN 有无接收到数据都跳出接收状态。然后反过来发送一个数据到A端。到此，B端代码分析结束。

#### 5、整个流程分析

上面已经很清楚地分析了发送端和接收端自己的工作流程，现在我把两者联合起来再分析一下。大致流程是这样的：首先我们执行B（接收）端的程序，让B端处于接收状态，需要注意的是按上面的设置接收状态只能维持5秒，然后再在5秒内执行A端的测试程序。当执行A端的程序时，A端会发送数据到B端，B端成功接收的话会打印相关信息。B端在5秒钟后会返回一个数据到A端。当A端成功接收到该数据时也会打印相关信息。如此一来，A、B两端都能测试各自的收、发功能。到此，整个流程分析结束。

#### 6、几个要点

要理解代码的流程还需要理解以下几个知识点：

##### （1）、system（）函数

system()会调用fork()产生子进程，由子进程来调用/bin/sh-c string来执行参数string字符串所代表的命令，此命>令执行完后随即返回原调用的进程。

##### （2）、candump can0 1>/tmp/can

这个命令的意思是使CAN处于接收状态，当有接收到数据时将标准正确输出重定向到/tmp/can文件中（可以简单地理解为将数据写入到/tmp/can文件中）

##### （3）、cat /tmp/can | grep \"11 22 33 44 55 66 77 88\" 1>/tmp/can\_info

为了方便测试，我们从A端发送到B端的数据为“11 22 33 44 55 66 77 88”，这个命令的意思是搜索/tmp/can文件，如果该文件中包含有“”中的字符串，则将它重定向到/tmp/can\_info文件中。

##### （4）、stat(ERRO\_INFO,&stat\_buf)

这个函数的作用是判断/tmp/can\_info文件是否为空，如果不为空，说明有接收到数据，且数据中包含“11 22 33 44 55 66 77 88”这个字符串。

理解了以上几个知识点，相信读者能很好地理解该测试程序。

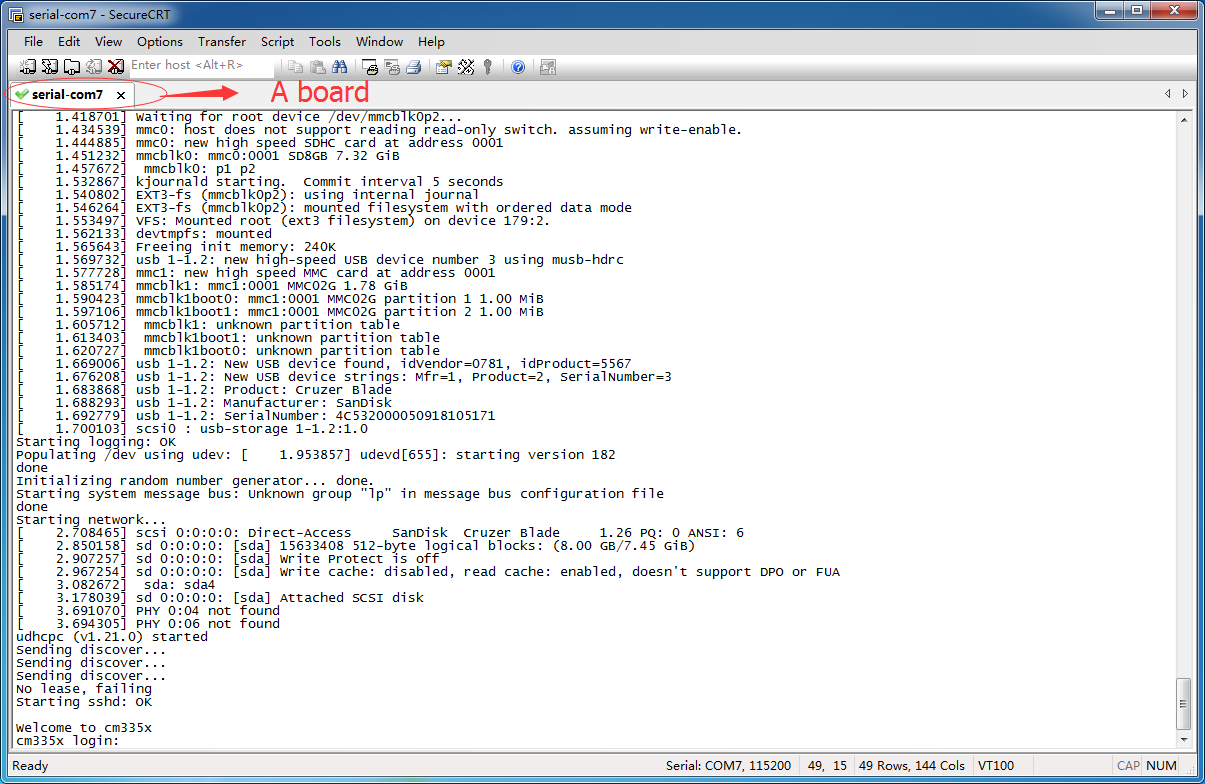
### 代码测试

注：SBC3358-B1A默认先从eMMC启动，如果eMMC中没有镜像，则会从TF卡启动，如果TF卡中也没有镜像，则串口终端会一直打印“CCCC”。当eMMC中有镜像且TF卡也有镜像时，如果这时想从TF卡启动，只要先将板子上的CN17引脚短接再上电即可从TF卡启动。

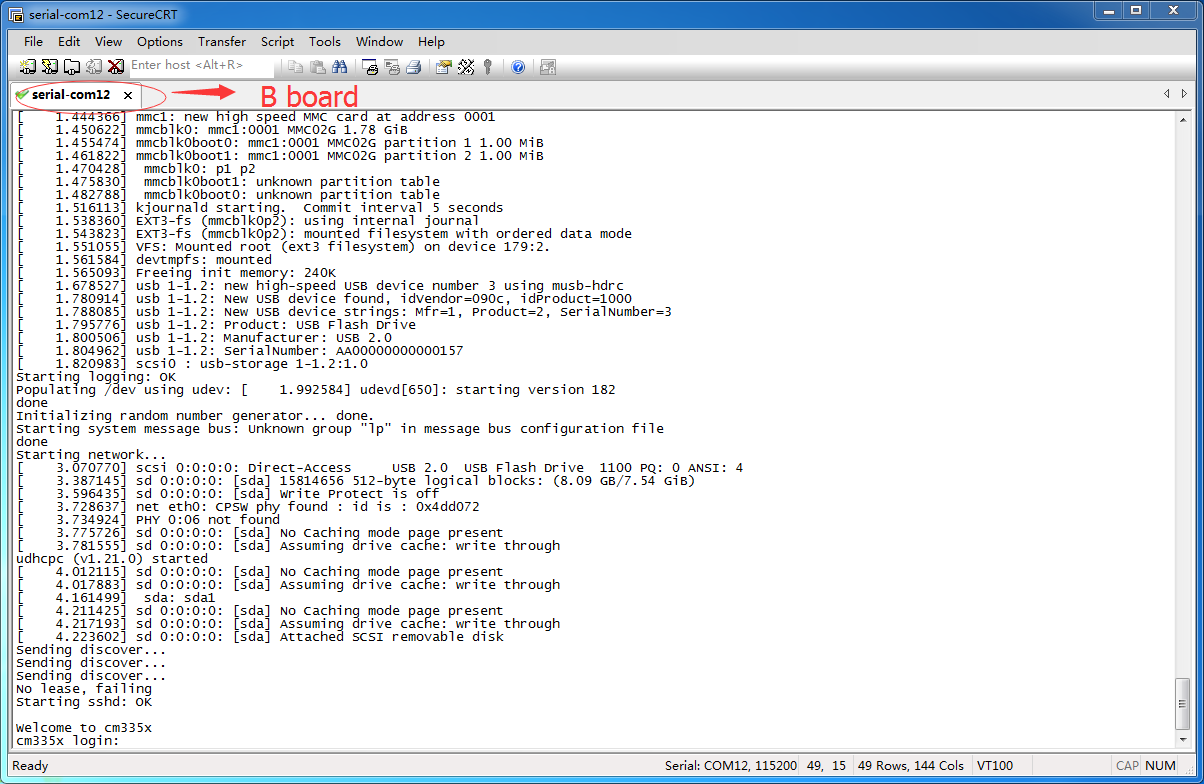
（1）、插上卡，接好串口线，打开串口调试软件。

（2）、软件设置为：波特率115200,8 bit数据位，无校验位，1bit停止位，无流控。

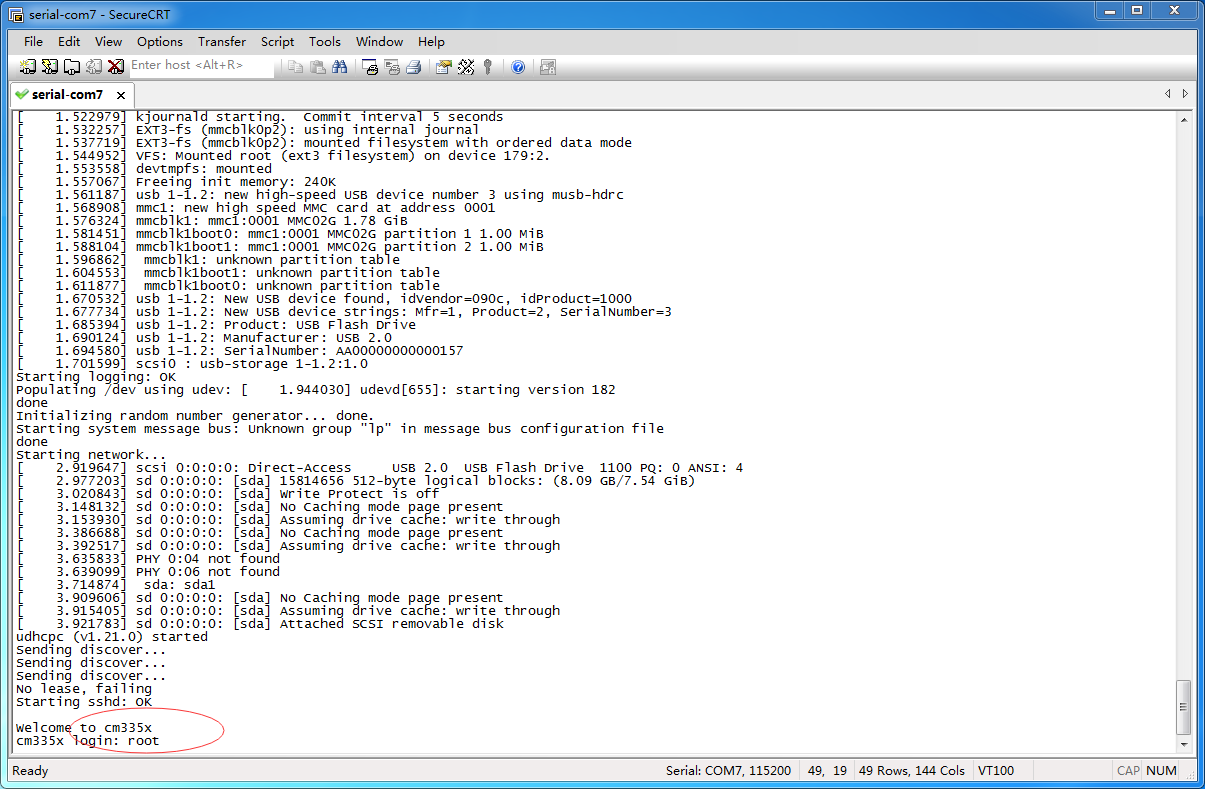
（3）、本次演示A开发板对应com7，如下所示：



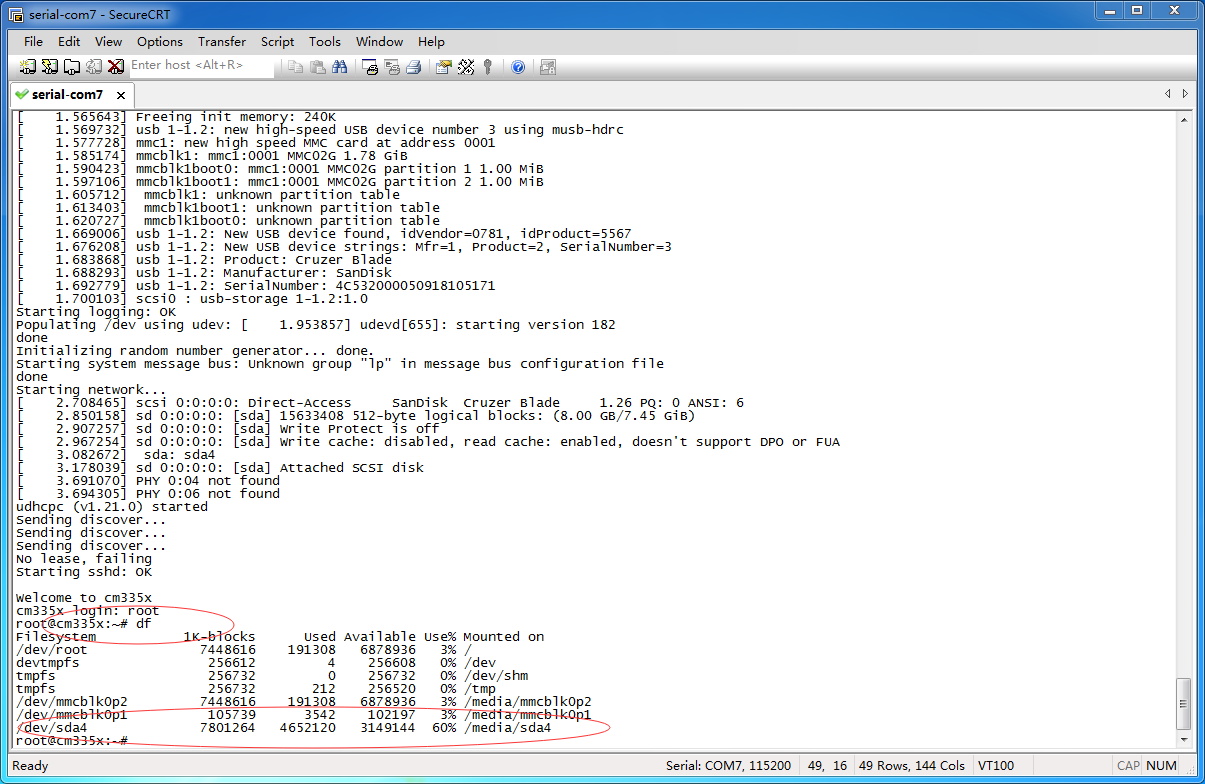
本次演示B开发板对应com12，如下所示：



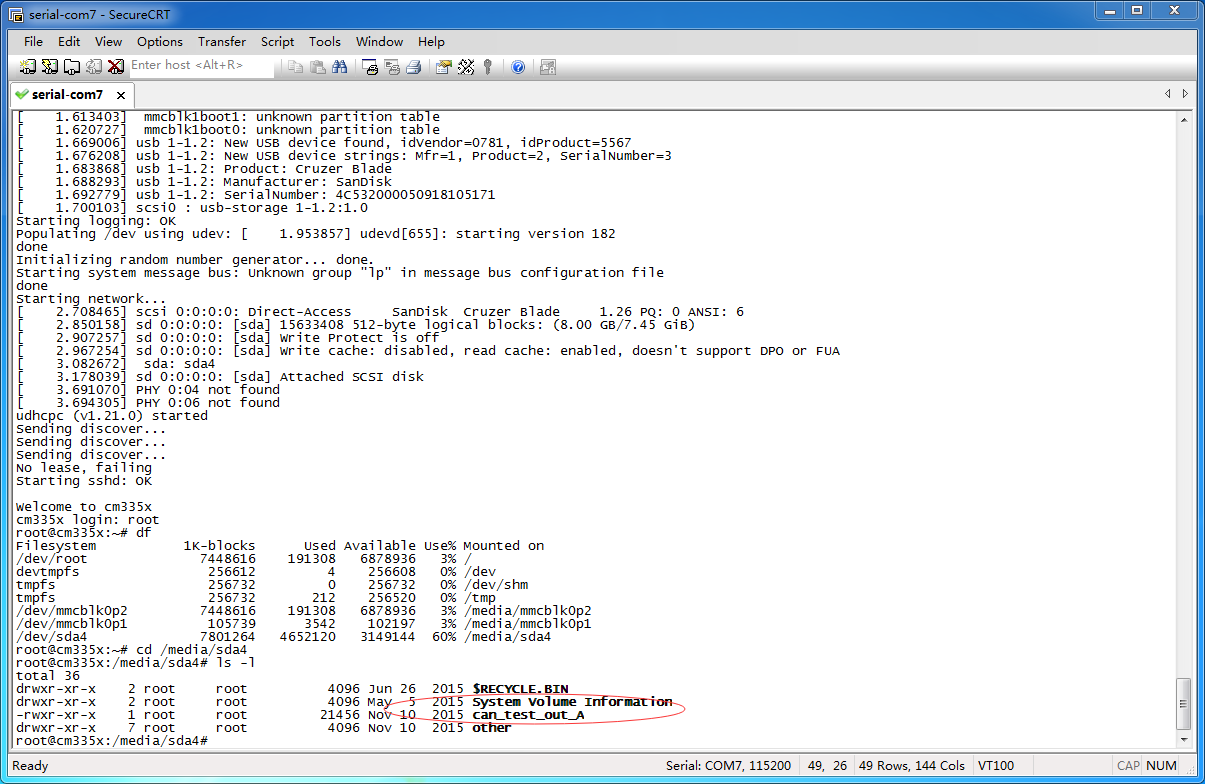
（4）、根据第六点将两个开发板接好线，上电开机后在终端输入“root”登录：



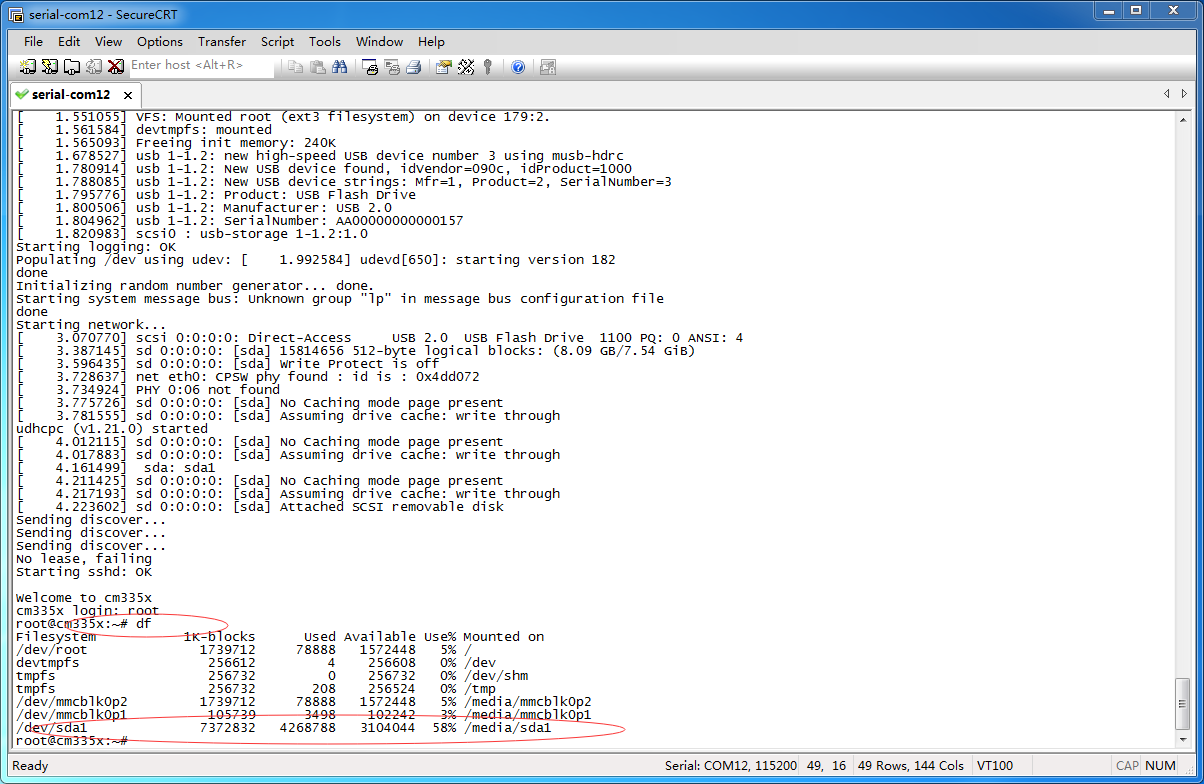
（3）、将can\_test\_out\_A放在U盘中并插入A开发板，找到can\_test\_out\_A这个文件。可先查看该优盘挂载的位置：



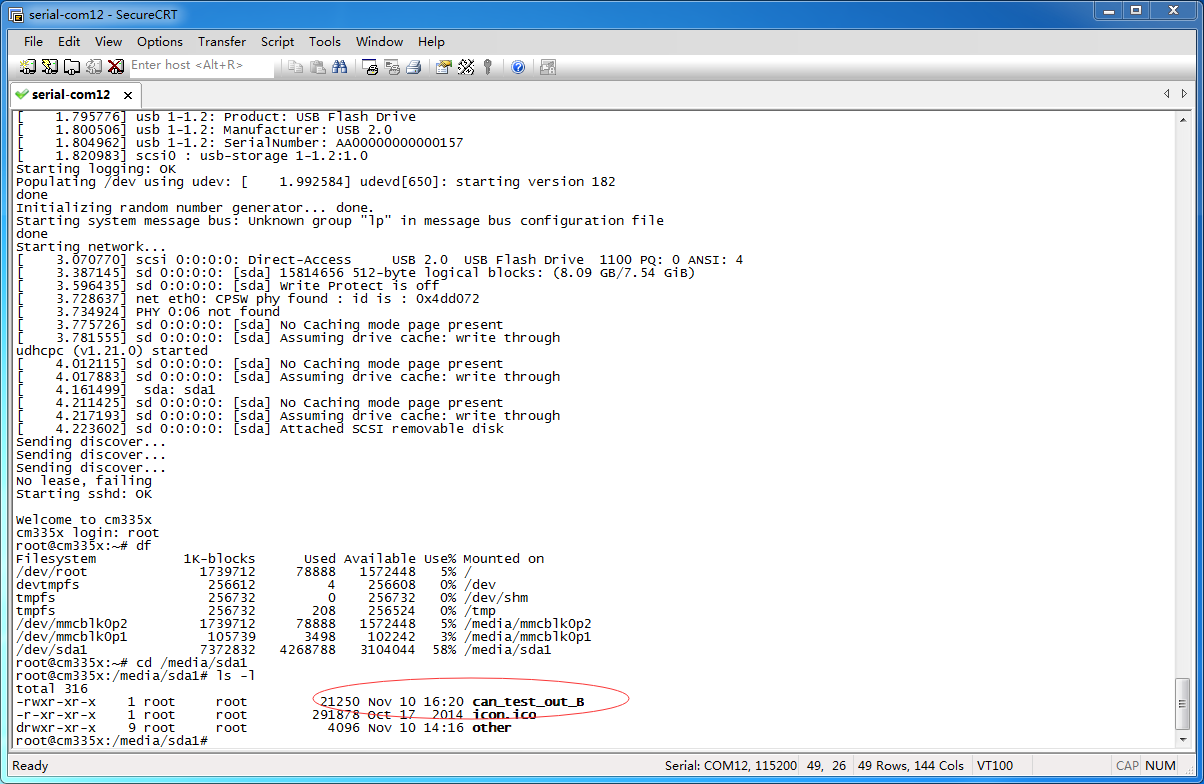
找到该测试文件：



将can\_test\_out\_B放在U盘中并插入B开发板，找到can\_test\_out\_B这个文件。可先查看该优盘挂载的位置：

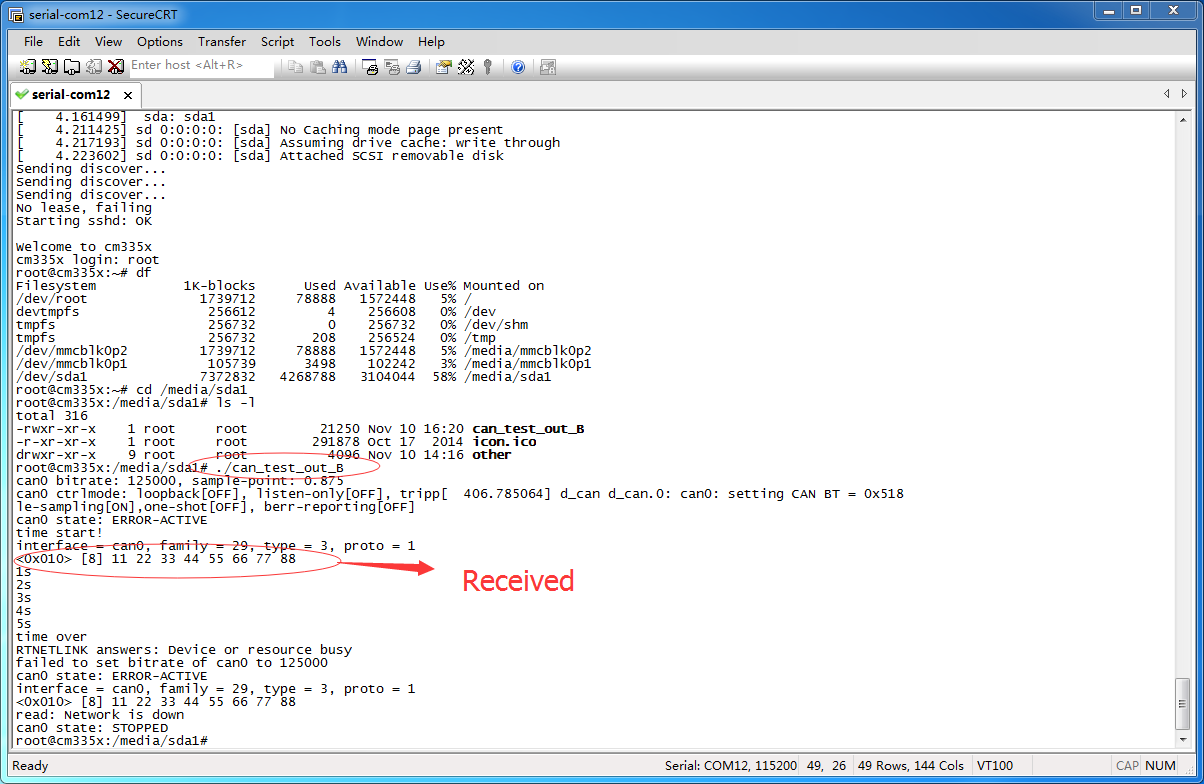


找到该测试文件：



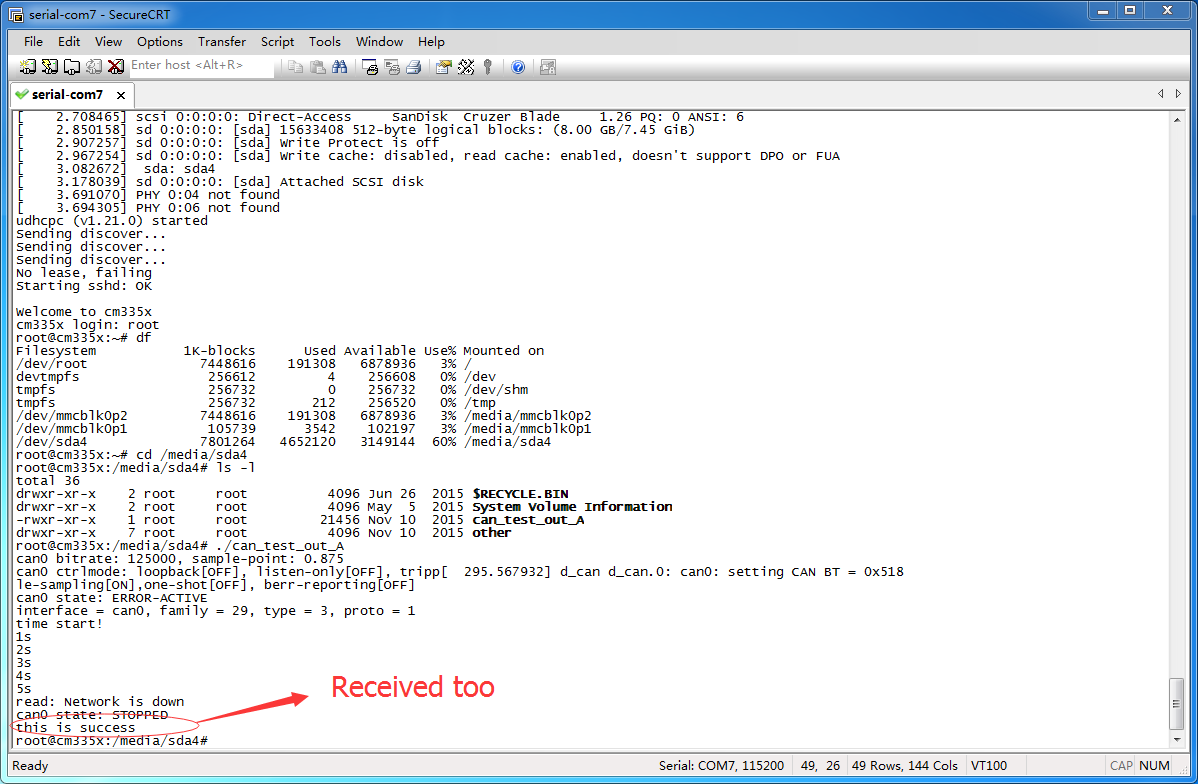
（5）、先在B终端输入（先开启接收端程序）：

./can\_test\_out\_B



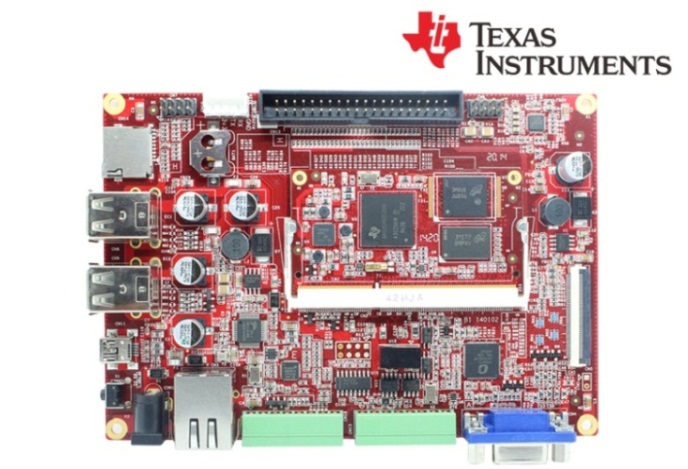
5秒之内在A终端输入（再开启发送端程序）

./can\_test\_out\_A

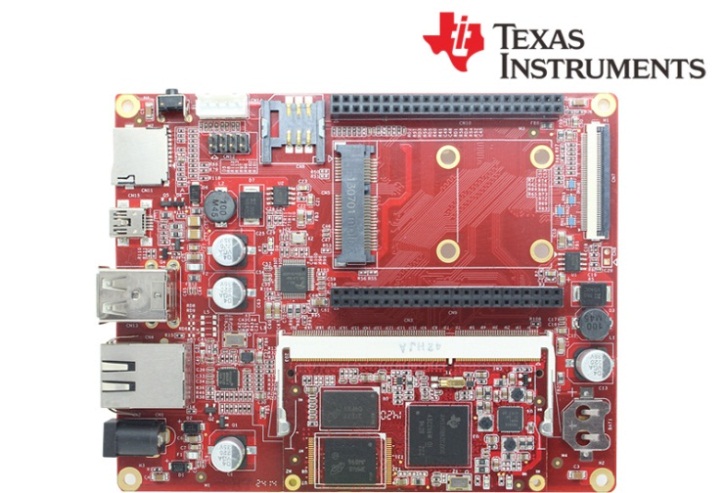


执行A测试程序时，B端立马接收到A发送过来的数据包。B端计时5秒钟之后也把一个数据包发送到A端，A端接收到之后打印“this is success”。程序执行过程完全按照我们的设计。A、B终端打印的信息为上图所示，从打印信息来看，A、B两个板子均接收到对方发来的数据包，测试成功。测试完成后程序自动退出。

到这里，Linux下CAN测试程序开发教程编写测试完成。

附相关GOEMBED产品介绍[](http://www.goembed.com/index.php/Products/detail/tpid/25)

*SBC335x – B1A*

[](http://www.goembed.com/index.php/Products/detail/tpid/12)

*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.

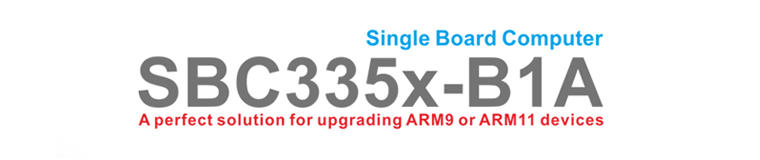
The B1A features 4 serial ports (including 2 RS232 and 2 TTL), 4 USB Host and 1 USB OTG, 1 Ethernet ports, CAN, RS485, Wiegand, VGA, LCD, Touch screen, Audio, ADC and more other peripherals.

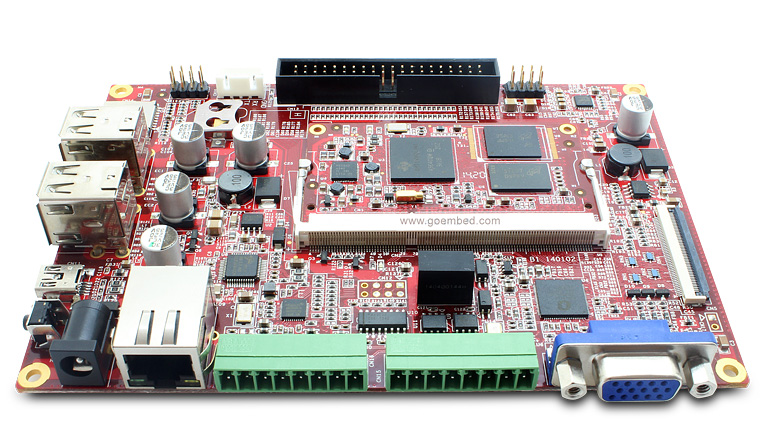
The B2A features 4 USB Host and 1 USB OTG, 1 Ethernet ports, LCD, Touch screen,RTC, and more other peripherals.

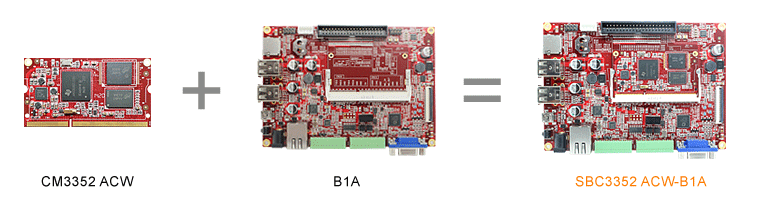
The SBC board targets a wide range of applications, including: HMIs, Digital Signage, POS, Data Terminal, Medical Devices, Navigation, Industrial Automation, Entertainment system, Thin Clients, Robotics, Game Console and much more.

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](http://goembed.com/index.php/Products/detail/tpid/14)  [-M51E20/08](http://goembed.com/index.php/Products/detail/tpid/14) | [CM3352 BCW](http://goembed.com/index.php/Products/detail/tpid/29)  [-M51E40/08](http://goembed.com/index.php/Products/detail/tpid/29) | [CM3358 ACW](http://www.goembed.com/index.php/Products/detail/tpid/44)  [-M51E20/10](http://www.goembed.com/index.php/Products/detail/tpid/44) | [CM3358 BCW](http://goembed.com/index.php/Products/detail/tpid/22)  [-M51E40/10](http://goembed.com/index.php/Products/detail/tpid/22) |
| **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**](mailto:%20sales@goembed.com) | In Stock | **Out of Stock**  [**Contact us**](mailto:%20sales@goembed.com) |
| **Minimum Availability** | 2022 | | | |

**SBC335x-B1A Block Diagram**

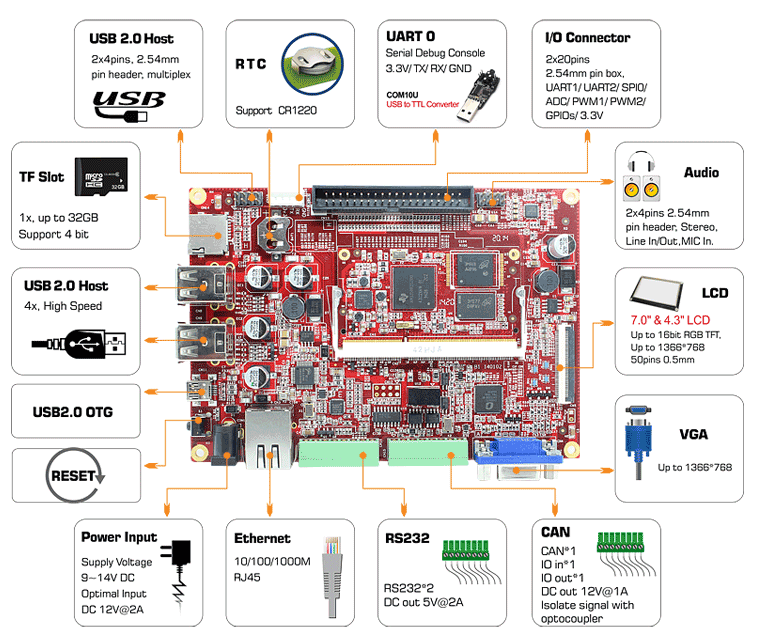
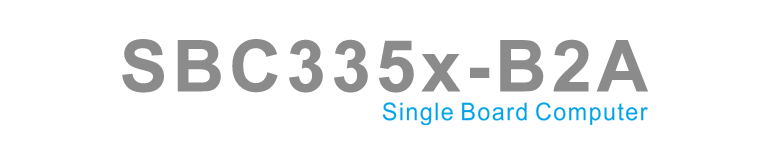
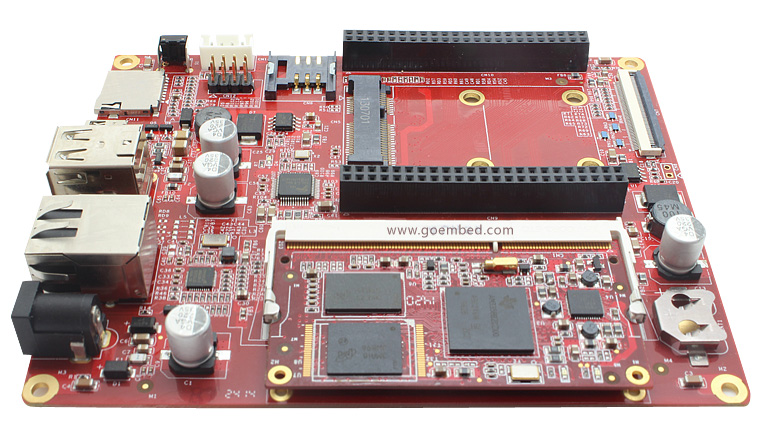
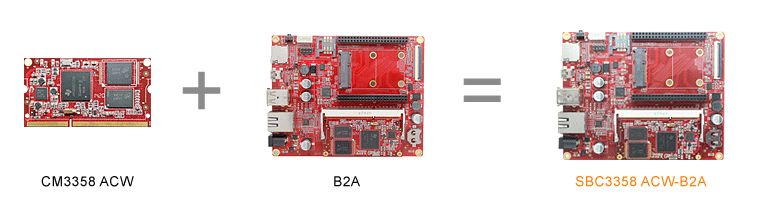


Figure 1 B1 Block Diagram







**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](http://goembed.com/index.php/Products/detail/tpid/14)  [-M51E20/08](http://goembed.com/index.php/Products/detail/tpid/14) | [CM3352 BCW](http://goembed.com/index.php/Products/detail/tpid/29)  [-M51E40/08](http://goembed.com/index.php/Products/detail/tpid/29) | [CM3358 ACW](http://www.goembed.com/index.php/Products/detail/tpid/44)  [-M51E20/10](http://www.goembed.com/index.php/Products/detail/tpid/44) | [CM3358 BCW](http://goembed.com/index.php/Products/detail/tpid/22)  [-M51E40/10](http://goembed.com/index.php/Products/detail/tpid/22) |
| **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**](mailto:%20sales@goembed.com) | In Stock | **Out of Stock**  [**Contact us**](mailto:%20sales@goembed.com) |
| **Minimum Availability** | 2022 | | | |

**SBC335x-B2A Block Diagram**

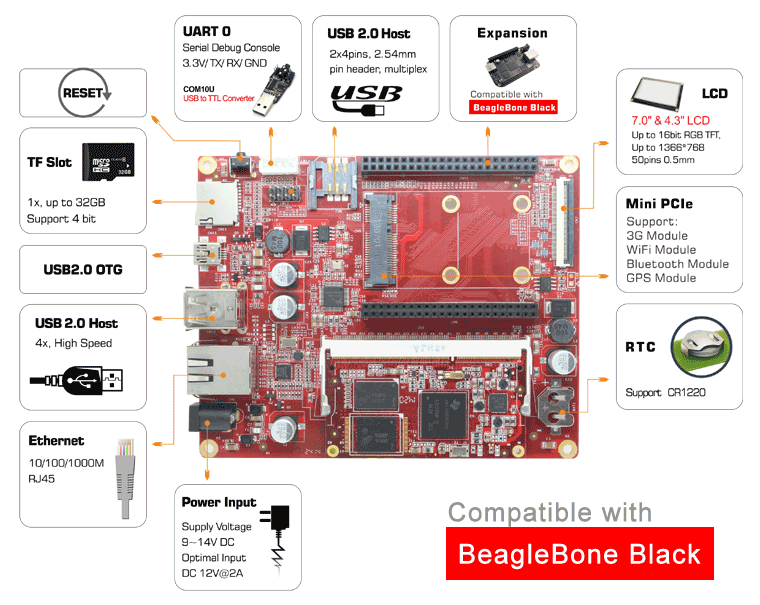


Figure 1 B2A Block Diagram

**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.

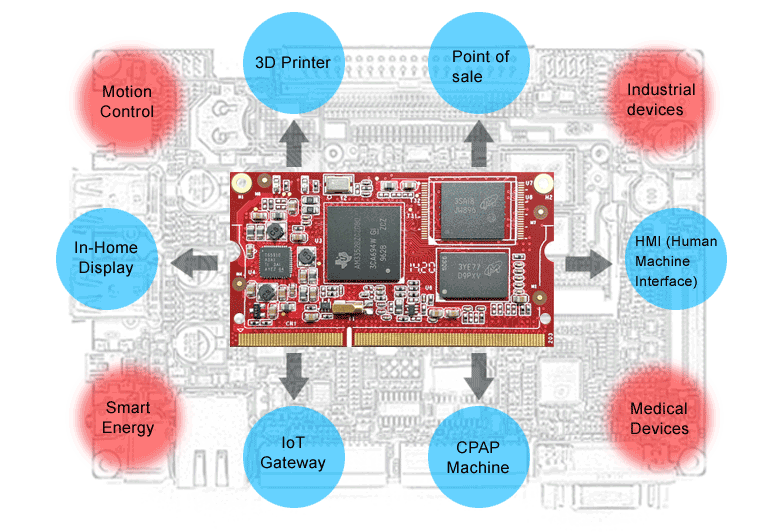
We are your trust worthy partner on ARM embedded design services and solutions.

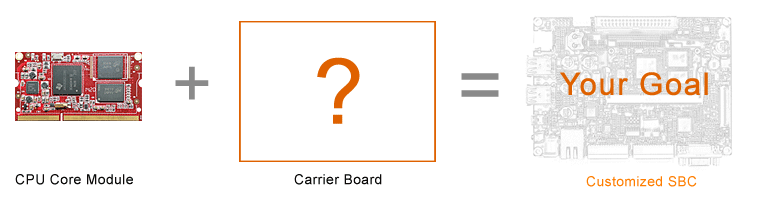
**More Carrier Boards**

Customized based on your needs!

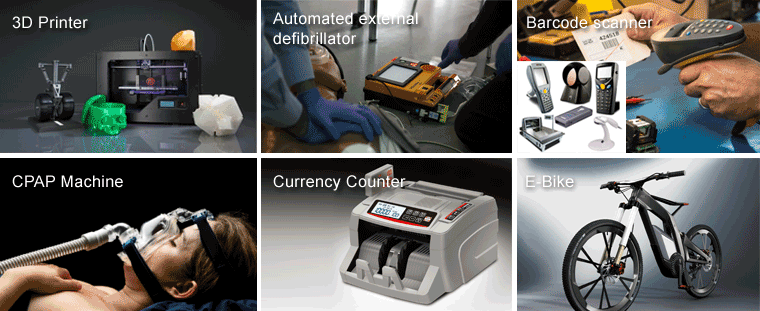
**ODM / OEM Services**

Bring your new products to market quickly





**Related end equipment**





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

