用户手册

[SBC-EC9100]

历史版本

Rev.	Note	Author
20160907	Initial	Sandy

目录

Release Note

1. 镜像版本

91200003_SBC-EC9100-SDcard_Shipment_Image_REV00.img

91200003_SBC-EC9100-EMMC_Shipment_Image_REV00.img

2. 功能列表

Feature List	SBC-EC9100			
	Schematic Page#	On-Chip Peripherals	On-Board Peripherals	Detail Functions(Needed)
u-boot version	2015.04			Can only boot the kernel
kernel version	3.14.52			Kernel will support all the function below
Filesystem	buildroot			buildroot of embest own
CPU	EC9100-U2	imx6ul		
DDRAM	EC9100-U7	DDR3	MT41K256M16HA-125	Can access read write and run code
MicroSD_(TF)	EC9100-J3	MMC0	Null	Can access read write and boot
eMMC	EC9100-U6	MMC1		Can access read write
UART-0		UART1		System can send and receive data in loopback mode
UART-1		RS485		System can send and receive data in loopback mode
UART-2		UART3		Debug Serial
PMIC	EC9100-U3	I2C0	PZUFE3001	
Ethernet-1	EC9100-U8	RGMII1	KSZ8081RNXIA	
USB-Host	EC9100-J7	USB1		Can recognize U disk by USB host
USB-OTG	EC9100-J8	USB0	Null	Can recognize U disk by USB host
LCD	EC9100-J5	RGB	Null	Can show picture on the screen
Backlight	EC9100-U24	PWM	Null	System can control the LCD backlight
Touchscreen	EC9100-J5	ADC-TSC	Null	
EEPROM	EC9100-U25	I2C0	AT24C256W	Can access read write
CAN-1	EC9100-u14	CAN1	mc33901wef	System can send and receive data in loopback mode
CAN-2	EC9100-U17	CAN0	mc33901wef	System can send and receive data between two board
RS485-1	EC9100-MN	UART1	ADM3485	System can send and receive data

	1			between two board
debian filesystem				
ADC		Null	Null	
CAMERA		CSI&I2C1	Null	
WIFI	EC9100-U21	SDIO2	Null	
SPI		SPI3	Null	

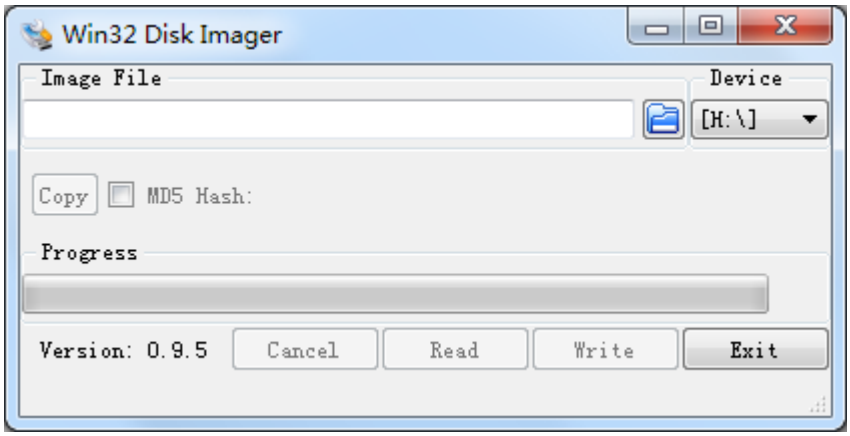
3. 已知问题

1. The highest resolution of Camera is 720*576 resolution.
2. Wi-Fi module is selectable ,but now cannot work;

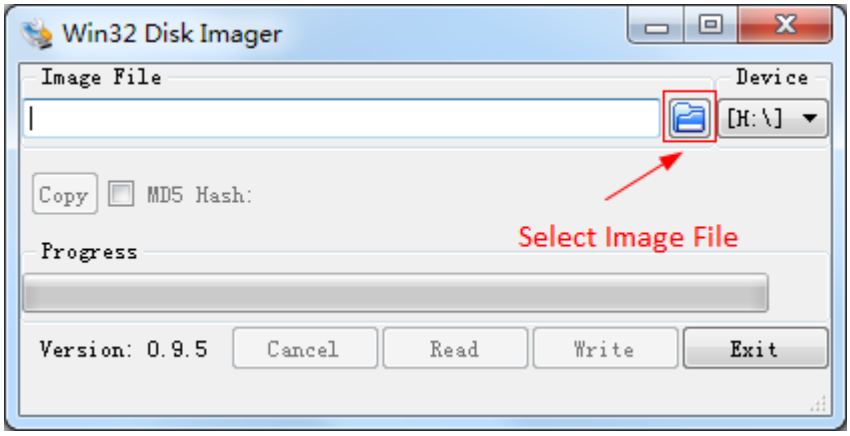
第1章 快速启动

1.1 烧写镜像到 SD 卡

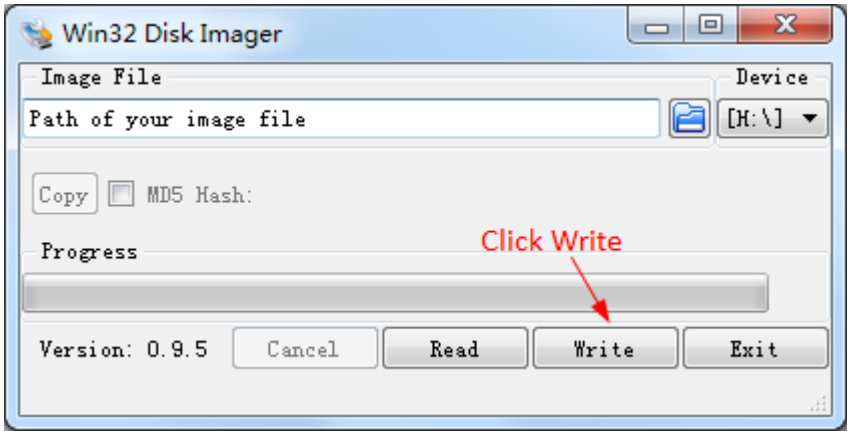
- 首先，你需要准备一张不小于 2G 的 SD 卡
- 然后，你需要从 https://sourceforge.net/projects/win32diskimager/ 下载并安装 Win32 Disk Imager



- 选择需要烧写的镜像文件，91200003_SBC-EC9100-SDcard_Shipment_Image_REV00.img



- 点击“Write”烧写镜像：



1.2 从 SD 卡启动系统

- 在 PC 上安装串口软件（例如 SecureCRT）,选择正确的端口号，波特率 115200，8 位数据位，1 位停止位，无奇偶校验
- 用 USB 转 TTL 模块连接板子上的串口引脚 TX(J10 Pin8)RX(J10 Pin10)和 PC
- 把 SD 卡插入板子的插槽(J3)
- 用 5V,2A 的电源, 给板子供电(J1)
- 拨码开关 SW3 拨至 00，SW4 拨至 0010
- 等系统启动完毕之后, 串口显示如下:

```
[  OK  ] Started System Logging Service.
         Starting Getty on tty1...
[  OK  ] Started Getty on tty1.
         Starting Serial Getty on ttymxc2...
[  OK  ] Started Serial Getty on ttymxc2.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started Embest AutoExec Service.
[  OK  ] Started Login Service.

Debian GNU/Linux 8 embest ttymxc2

embest login:
```

输入登录名：root ， 密码：root 登陆；

```
Debian GNU/Linux 8 embest ttymxc2

embest login: root
Password:
Linux embest 3.14.52 #1 SMP PREEMPT Wed Aug 31 12:08:45 CST 2016 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@embest:~#
```

1.3 从 EMMC 启动系统

将 91200003_SBC-EC9100-EMMC_Shipment_Image_REV00.img 拷贝到 U 盘；

参考 1.2，先从 SD 卡启动系统，再将 U 盘插入 USB 接口（J8）：

在串口终端输入：

```
root@embest:~# ls /dev/sd*
/dev/sda /dev/sda1
root@embest:~# mount /dev/sda1 /mnt/
root@embest:~# dd if=/mnt/91200003_SBC-EC9100-EMMC_Shipment_Image_REV00.img of=/dev/mmcblk1
```

注意：烧写时间较长，请耐心等待…

如果是第一次烧写 EMMC，需要在 eMMC 用户分区使能 boot_config。（参考 3.2.3 相关内容）

后续烧写 EMMC 只需要执行上述命令即可。

烧写结束后掉电，将拨码开关 SW3 拨至 10,SW4 拨至 0110，拔掉 SD 卡，上电；

第2章 功能测试

首先，请参考第一章 1.1，把系统启动起来. 然后跟随下面的指引测试各项功能.

2.1 按键测试

将烧入镜像的 Micro SD 卡插入 SBC-EC9100，接上电源，系统启动。

待系统启动后，按 RESET 键可重启系统。

长按 ON/OFF 8 秒以上，系统会挂起，再长按 8 秒，系统会重新启动

也可通过下面的命令测试

```
root@embest:~# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x0 product 0x0 version 0x0
Input device name: "20cc000.snvs-pwrkey"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 116 (KEY_POWER)
Properties:
Testing ... (interrupt to exit)
```

按下按键，会出现下面信息：

```
Event: time 1469459707.798194, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 1469459707.798194, ---------- EV_SYN ----------
Event: time 1469459708.038234, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 1469459708.038234, ---------- EV_SYN ----------
Event: time 1469459710.058188, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 1469459710.058188, ---------- EV_SYN ----------
Event: time 1469459710.238220, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 1469459710.238220, ---------- EV_SYN ----------
```

2.2 RTC 测试

外接电池后,在串口终端输入：

查看当前时间：

```
root@embest:~# date
Sat Jan    1 00:02:07 UTC 2000
```

设置时间 2016 年 3 月 9 日 10 时 46 分：

root@embest:~# date 030910462016

Wed Mar 9 10:46:00 UTC 2016

把系统时钟写入 RTC:

root@embest:~# hwclock –w

读取 RTC:

root@embest:~# hwclock

Wed 09 Mar 2016 10:46:23 AM UTC -0.432561 seconds

可以看到，硬件时钟 RTC 被设置成 2016 年 3 月 9 日，系统时钟被保存到硬件时钟里。重启系统并查看时间：

root@embest:~# date

Wed Mar 9 10:46:45 UTC 2016

RTC 精度测试(可选):

将板子的时间与 PC 的网络时间同步后,断掉板子的电源,仅用外接电池供电,24 或 48 小时,重新加电读取时间,计算板子 RTC 时间与 PC 时间的差值即 RTC 精度.

2.3 EEPROM 测试

在串口终端输入一下命令：

root@embest:~# ./eeprom_test

data will write to EEPROM at 0x400

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f

30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f

40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f

50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f

60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f

70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f

80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f

90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f

a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af

b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf

c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf

d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df

e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef

f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

data read from EEPROM at 0x400

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

写数据与读到的数据相同，测试通过；

2.4 EMMC 测试

在串口终端执行：

root@embest:~# touch emmc_read emmc_write

编辑 emmc_write:

root@embest:~# vi emmc_write

例如写入“emmc write test”

写 emmc 命令:

root@embest:~# dd if=emmc_write of=/dev/mmcblk1

mmcblk1: p1 p2

0+1 records in

0+1 records out

16 bytes (16 B) copied, 0.0396547 s, 0.4 kB/s

读 emmc 命令：

root@embest:~# dd if=/dev/mmcblk1 of=emmc_read bs=1K count=10

10+0 records in

10+0 records out

10240 bytes (10 kB) copied, 0.0179817 s, 569 kB/s

查看 emmc_read:

root@embest:~# cat emmc_read

emmc write test

测试成功；

2.5 GPIO 测试

EC9100 上作为 GPIO 控制的外接管脚有 4 个分别是 GPIO4（J9 8 脚），GPIO5（J9 7 脚），GPIO9（J9 23 脚）和 GPIO132（J10 7 脚），通过以下命令测试 GPIO 功能：

例：测试 GPIO4 引脚：

root@embest:~# echo 4 > /sys/class/gpio/export

root@embest:~# echo out > /sys/class/gpio/gpio4/direction

输入以下命令，再用万用表测试相应管脚的电压：

root@embest:~# echo 1 > /sys/class/gpio/gpio4/value

此时管脚输出应该为高电平

root@embest:~# echo 0 > /sys/class/gpio/gpio4/value

此时管脚输出应该为低电平

把以上命令中的 4 改成 5， 9，132 即可测试 GPIO5（J9 7 脚），GPIO9（J9 23 脚）和 GPIO132（J10 7 脚）的功能。

2.6 LCD 测试

连接 LCD 模块到 J5

4.3 寸屏：

打开 SD 卡中 uEnv.Txt 文件，修改 fdtfile= embest_fsl_sbc_ec9100_4.3inch.dtb

重新启动系统；

7 寸屏：

打开 SD 卡中 uEnv.Txt 文件，修改 fdtfile= embest_fsl_sbc_ec9100_7inch.dtb

重新启动系统；

背光调节测试，执行如下命令：

root@embest:~# echo 0 > /sys/class/backlight/backlight.9/brightness

依次输入 0-7 后，可以看到屏幕亮度的变化；

2.7 触摸屏测试

连接显示屏到 J5，首先确认 fuse 已经写入。通过如下命令读出 fuse 值,只需要把 20 和 21bit 置为 1 即可,如(0x003000xx).

root@embest:~# cat /sys/fsl_otp/HW_OCOTP_CFG2

设置方法：(xx 替换成之前 cat 读到的数值)

root@embest:~# echo 0x003000xx > /sys/fsl_otp/HW_OCOTP_CFG2

注意：读出来的值可能每块板不一样，没有影响。

输入以下命令后，校准及相关命令能才正常使用：

root@embest:~# export TSLIB_TSDEVICE=/dev/input/event1

在串口终端输入以下命令执行触摸屏校准程序，

root@embest:~# ts_calibrate

按照屏幕上提示，点击 “+” 图标 5 次完成校准

2.8 串口测试

开发板上有 2 个串口，其中 UART-3 为 debug 接口；短接 J203 将 iomux 芯片选择为 UART 和 CAN 模式；

2.8.1 回环测试：

UART2,短接 J9 第 20，22 号接口：

root@embest:~# ./uart_test -d /dev/ttymxc1 -b 115200

/dev/ttymxc1 SEND: 1234567890

/dev/ttymxc1 RECV 10 total

/dev/ttymxc1 RECV: 1234567890

2.8.2 板间测试：

两块 ec9100,RX（J9，22 号引脚）接另一块的 TX（J9，20 号引脚）,TX 接另一块的 RX；

两块板子对应的终端分别执行：

root@embest:~# ./uart_test -d /dev/ttymxc1 -b 115200

两块板子都有发送接收到数据，串口终端上打印信息如下：

/dev/ttymxc1 SEND: 1234567890

/dev/ttymxc1 RECV 10 total

/dev/ttymxc1 RECV: 1234567890

……

注意：Ctrl+C 中断串口测试

2.9 RS485 测试

需要两块 ec9100,对接两块板子上 J9 的 39，40 号引脚；

在 A 板上执行

root@embest:~# ./uart_test2 /dev/ttymxc0 9600 0 100

在 B 板上执行

root@embest:~# ./uart_test2 /dev/ttymxc0 9600 1

如果测试成功，在 B 板上会接收到如下信息。

*****data length = 31 ******

41 54 31 32 33 34 35 36 37 38 39 58 59 5a 61 62 63 64 65 64 66 68 69 6a 6b 6c 6d 6e 0d 00 fb

******* receive data 135667 pkts...0 bytes.....

交换在 A,B 板上运行的指令，测试现象应该一样

2.10 CAN 测试

将 CAN0 和 CAN1 连接，即将 J9 的 33 脚与 34 脚连接，35 脚与 36 脚连接。

测试步骤：

1. 打开 CAN0 CAN1

root@embest:~# ip link set can0 type can bitrate 50000 triple-sampling on

root@embest:~# ip link set can1 type can bitrate 50000 triple-sampling on

root@embest:~# ip link set can0 up

flexcan 2090000.can can0: writing ctrl=0x27292085

root@embest:~# ip link set can1 up

flexcan 2094000.can can1: writing ctrl=0x27292085

2. 发送与接收：

CAN1 接收 ， CAN0 往 CAN1 发数据，显示如下表示测试收发成功。

root@embest:~# candump can1&

[4] 589

root@embest:~# cansend can0 123#01020304050607

can1 123 [7] 01 02 03 04 05 06 07

用查看命令可以看到 CAN0 Tx 增加了 1 个 packet，7 个 bytes。CAN1 RX 增加了 1 个 packet，7 个 bytes

root@embest:~# ip -d -s link show can0

2: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 576 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 10

link/can promiscuity 0

can <TRIPLE-SAMPLING> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0

bitrate 50000 sample-point 0.866

tq 1333 prop-seg 6 phase-seg1 6 phase-seg2 2 sjw 1

flexcan: tseg1 4..16 tseg2 2..8 sjw 1..4 brp 1..256 brp-inc 1

clock 30000000

re-started bus-errors arbit-lost error-warn error-pass bus-off

0 0 0 0 0 0

RX: bytes packets errors dropped overrun mcast

0 0 0 0 0 0

TX: bytes packets errors dropped carrier collsns

7 1 0 0 0 0

注意：

两个 can 端口必须设置成相同的波特率。

2.11 网络测试

在串口终端中输入以下命令：

设置 IP 地址：

root@embest:~# :~# ifconfig eth0 192.168.2.64

网络测试：

root@embest:~# ping 192.168.2.1

2.12 USB 测试

2.12.1 Host 测试

2.12.1.1 U 盘

将 U 盘插入 USB host 接口（J8），串口显示磁盘信息:

usb 2-1: USB disconnect, device number 3

usb 2-1: new high-speed USB device number 4 using ci_hdrc

usb-storage 2-1:1.0: USB Mass Storage device detected

scsi0 : usb-storage 2-1:1.0

scsi 0:0:0:0: Direct-Access Generic Flash Disk 8.07 PQ: 0 ANSI: 4

sd 0:0:0:0: [sda] 7823360 512-byte logical blocks: (4.00 GB/3.73 GiB)

sd 0:0:0:0: [sda] Write Protect is off

sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA

sda: sda1

sd 0:0:0:0: [sda] Attached SCSI removable disk

串口终端输入如下命令：

root@embest:~# ls /dev/sd*

/dev/sda /dev/sda1

/dev 下存在 sdx 的设备节点；

2.12.1.2 USB 键盘

将 usb 键盘接入到 usb host 接口（J8）,会出现打印,

usb 2-1: new low-speed USB device number 3 using ci_hdrc

input: SIGMACH1P USB Keykoard as

/devices/soc0/soc.0/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb2/2-1/2-1:1.0/0003:1C4F:0002.0003/input/input4

hid-generic 0003:1C4F:0002.0003: input: USB HID v1.10 Keyboard [SIGMACH1P USB Keykoard] on

usb-ci_hdrc.1-1/input0

input: SIGMACH1P USB Keykoard as /devices/soc0/soc.0/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb2/2-1/2-1:1.1/0003:1C4F:0002.0004/input/input5

hid-generic 0003:1C4F:0002.0004: input: USB HID v1.10 Device [SIGMACH1P USB Keykoard] on usb-ci_hdrc.1-1/input1

运行 evtest 命令，测试/dev/input/event2:

root@embest:~# evtest /dev/input/event2

Input driver version is 1.0.1

Input device ID: bus 0x3 vendor 0x1c4f product 0x2 version 0x110

Input device name: "SIGMACH1P USB Keykoard"

Supported events:

  Event type 0 (EV_SYN)

  Event type 1 (EV_KEY)

    Event code 1 (KEY_ESC)

    Event code 2 (KEY_1)

    Event code 3 (KEY_2)

    Event code 192 (KEY_F22)

    Event code 193 (KEY_F23)

    Event code 194 (KEY_F24)

    Event code 240 (KEY_UNKNOWN)

  Event type 4 (EV_MSC)

    Event code 4 (MSC_SCAN)

  Event type 17 (EV_LED)

    Event code 0 (LED_NUML)

    Event code 1 (LED_CAPSL)

    Event code 2 (LED_SCROLLL)

Key repeat handling:

  Repeat type 20 (EV_REP)

    Repeat code 0 (REP_DELAY)

      Value   250

    Repeat code 1 (REP_PERIOD)

      Value   33

Properties:

Testing ... (interrupt to exit)

按下键盘按键,对应的按键的数据会被打印出来:

Event: time 73542.642111, type 4 (EV_MSC), code 4 (MSC_SCAN), value 70017

Event: time 73542.642111, type 1 (EV_KEY), code 20 (KEY_T), value 1

Event: time 73542.642111, ---------- EV_SYN ----------

Event: time 73542.762091, type 4 (EV_MSC), code 4 (MSC_SCAN), value 70017

Event: time 73542.762091, type 1 (EV_KEY), code 20 (KEY_T), value 0

Event: time 73542.762091, ---------- EV_SYN ----------

Event: time 73544.202085, type 4 (EV_MSC), code 4 (MSC_SCAN), value 7000a

Event: time 73544.202085, type 1 (EV_KEY), code 34 (KEY_G), value 1

Event: time 73544.202085, ---------- EV_SYN ----------

Event: time 73544.290084, type 4 (EV_MSC), code 4 (MSC_SCAN), value 7000a

Event: time 73544.290084, type 1 (EV_KEY), code 34 (KEY_G), value 0

Event: time 73544.290084, ---------- EV_SYN ----------

2.12.2 OTG 测试

通过转接线连接设备(u 盘或键盘)到 J7,测试方法及现象同 Host 测试，但是 USB 编号不同

2.12.2.1 U 盘

ci_hdrc ci_hdrc.0: timeout waiting for 00000800 in 12

ci_hdrc ci_hdrc.0: EHCI Host Controller

ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus number 1

ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00

hub 1-0:1.0: USB hub found

hub 1-0:1.0: 1 port detected

usb 1-1: new high-speed USB device number 2 using ci_hdrc

usb-storage 1-1:1.0: USB Mass Storage device detected

scsi1 : usb-storage 1-1:1.0

scsi 1:0:0:0: Direct-Access Generic Flash Disk 8.07 PQ: 0 ANSI: 4

sd 1:0:0:0: [sda] 7823360 512-byte logical blocks: (4.00 GB/3.73 GiB)

sd 1:0:0:0: [sda] Write Protect is off

sd 1:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA

sda: sda1

sd 1:0:0:0: [sda] Attached SCSI removable disk

2.12.2.2 USB 键盘

usb 1-1: new low-speed USB device number 4 using ci_hdrc

input: SIGMACH1P USB Keykoard as

/devices/soc0/soc.0/2100000.aips-bus/2184000.usb/ci_hdrc.0/usb1/1-1/1-1:1.0/0003:1C4F:0002.0003/input/input4

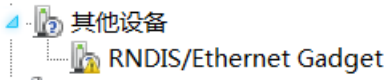
hid-generic 0003:1C4F:0002.0003: input: USB HID v1.10 Keyboard [SIGMACH1P USB Keykoard] on

usb-ci_hdrc.0-1/input0

input: SIGMACH1P USB Keykoard as
/devices/soc0/soc.0/2100000.aips-bus/2184000.usb/ci_hdrc.0/usb1/1-1/1-1:1.1/0003:1C4F:0002.0004/input/input5
hid-generic 0003:1C4F:0002.0004: input: USB HID v1.10 Device [SIGMACH1P USB Keykoard] on usb-ci_hdrc.0-1/input1

2.12.2.3 从设备

连接 J7 到 PC 端，打开 windows 设备管理器，识别到如下设备：



2.13 Camera 测试

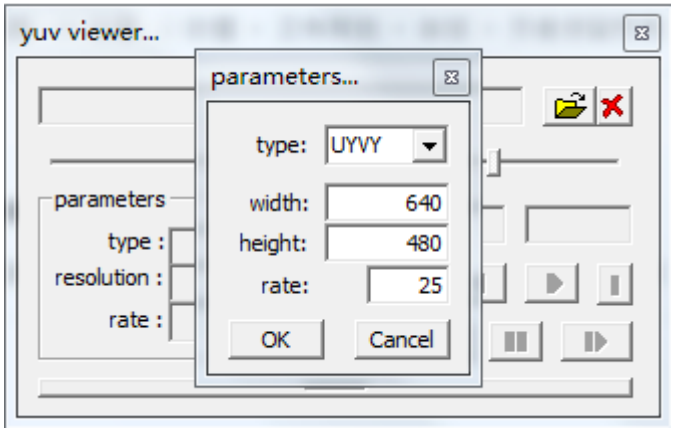
2.13.1 录制视频

将 Camera 模块连接到 J6。串口终端输入如下命令：

root@embest:~# ./mxc_v4l2_capture -iw 640 -ih 480 -ow 640 -oh 480 -c 25 -f UYVY /boot/firmware/test.yuv
root@embest:~# sync

摄像头录制一段分辨率 640*480，帧率 25 的视频，并在 SD 卡目录下会生成 test.yuv 文件。

连接 SD 卡到电脑，用 Pyuv.exe 打开。参数设置如下：



注意：Tool 目录下提供了 Pyuv.exe.

2.13.2 拍照

连接显示屏和摄像头，然后在串口终端中输入如下命令

root@embest:~# ./v4l2_capture_jpeg img1.jpg

在显示屏模块上显示拍摄的图片。

第3章 系统编译

3.1 配置编译环境

将文件夹（解压 rar 文件）拷贝到 Linux 环境下的$HOME 目录下，编译工具 fsl-linaro-toolchain-master.tar.gz 在$HOME/S5_Tool 目录下，用如下命令解压：

$tar –xzvf fsl-linaro-toolchain-master.tar.gz

导入环境变量：

$export CROSS_COMPILE=$HOME/S5_Tool/fsl-linaro-toolchain-master/bin/arm-fsl-linux-gnueabi-

$export ARCH=arm

3.2 编译 UBOOT

3.2.1 获取 uboot 源码

Uboot 源码在$HOME/S4_Sourcecode 目录下，解压 u-boot*.tar.gz :

$ cd $HOME/S4_Sourcecode

$ tar -xzvf u-boot*.tar.gz

3.2.2 编译并烧写镜像到 SD 卡

$ cd $HOME/S4_Sourcecode/u-boot

$ make distclean

$make embest_fsl_ec9100_sdcard_defconfig

$make

编译完成后在$HOME/S4_Sourcecode/u-boot 目录下生成 u-boot.imx，将 u-boot.imx 导入到 SD 卡中：(sdx 用 Linux 系统下 SD 卡的实际编号代替)

$ sudo dd if=u-boot.imx of=/dev/sdx bs=512 seek=2 conv=fsync

3.2.3 编译并烧写镜像到 EMMC

$ cd $HOME/S4_Sourcecode/u-boot

$ make distclean

$make embest_fsl_ec9100_emmc_defconfig

$make

编译完成后在$HOME/S4_Sourcecode/u-boot 目录下生成 u-boot.imx，将 u-boot.imx 拷贝到 SD 卡中：

参考 1.2，先从 SD 卡启动系统

启动系统后，在串口终端中执行下面的命令操作 EMMC 实现烧写过程：

1. 对 EMMC 分区

$ sudo dd if=/dev/zero of=/dev/mmcblk1 bs=1K count=1

$ echo -e " o\nn\np\n1\n20480\n+64M\na\nt\nc\nn\np\n2\n151552\n\nw\n " | fdisk /dev/mmcblk1

$ sudo mkfs.vfat /dev/mmcblk1p1

$ sudo mkfs.ext4 /dev/mmcblk1p2

$ sudo fdisk /dev/mmcblk1 –l

2. 将刚刚拷贝的 u-boot.imx(for emmc)烧写到 emmc 中

$ sudo dd if=u-boot.imx of=/dev/mmcblk1 bs=512 seek=2 conv=fsync

3. 在 eMMC 用户分区使能 boot_config：

$ sudo echo 56 > /sys/block/mmcblk1/device/boot_config

4. 检查是否使能成功：

$ sudo cat /sys/block/mmcblk1/device/boot_info

若使能成功，会打印以下信息：

boot_partition:0x78;

BOOT_ACK:1 - Boot acknowledge sent during boot operation

BOOT_PARTITION-ENABLE: 7 - User area enabled for boot

这样 U-boot 就被烧写到 EMMC 中了。

3.3 Kernel

3.3.1 获取内核源码

内核源码存在$HOME/S4_Sourcecode 目录下,解压 linux*.tar.gz

$ tar -zxvf linux*.tar.gz

3.3.2 编译并烧写镜像到 SD 卡

$ cd $HOME/S4_Sourcecode/linux

$ make distclean

$ make embest_fsl_sbc_ec9100_defconfig

$ make

编译完成后在

- $HOME/S4_Sourcecode/ linux/arch/arm/boot 目录下生成 zImage
- $HOME/S4_Sourcecode/linux/arch/arm/boot/dts 目录下生成
    1. embest_fsl_sbc_ec9100_4.3inch.dtb
    2. embest_fsl_sbc_ec9100_7inch.dtb

（dtb 文件分别对应 4.3 寸屏和 7 寸屏）（参考 LCD 测试）

将文件拷贝到 SD 卡中。