经天纬地之奇才，济世匡时之智略

技术性博客

个人资料

lichangc

访问： 50806次

积分： 875

等级： BLOG 3

排名： 千里之外

原创： 31篇 转载： 16篇

译文： 1篇 评论： 19条

AT91SAM926x开发全流程

分类： ARM9--ＡＴ91SAMxxx 2012-04-05 10:54 1638人阅读 评论(0) 收藏 举报

linux flash 编译器 asynchronous windows struct

目录(?) [+]

http://www.mcuzone.com:8080/dokuwiki/doku.php?id=product:sam9_linux#lcd_%E6%97%B6%E5%BA%8F

本页内容发表于MCUZone。

如果你对本页的技术内容有疑问，请到 MCUZone技术论坛 发帖。

如果你认为本页的内容侵害了你的权益，请与hotislandn@hotmail.com;hdapple_2000@hotmail.com 联系，我们会在确认后移除。

MCUZone旗下网站：

http://www.mcuzone.com—MCUZone是微控电子主力站点。

微控电子论坛—微控电子的技术/售后服务论坛。

http://www.atarm.com—ATARM为微控电子的第二品牌，主要进行ATARM的推广和芯片以及周边产品的销售。

SAM926x Linux HowTo

本文的目的在于帮助AT91SAM926x(以下简称为SAM926x)的软件开发工程师从零开始，为SAM926x建立Linux运行环境。

在进行SAM926x Linux开发之前，需要建立开发环境。

本文的行文不拘泥于一种形式，变化颇多，请谅解。

本文的部分内容来源于网络。

本文不会补充Linux的基本应用知识，请自行google。

本文涉及到的源码包都将收录于本站的 开发板配套光盘。

SAM926x Linux的基本组成：

- Bootstrap
- U-boot
- Kernel
- Rootfs

Bootstrap可以在Windows PC上编译，其余的在Linux PC上编译。

各部分组成如下图：

Lattice FPGA中假双口R/ (1790)
AT91SAM926x开发全流 (1637)
AT91SAM9260 SMC外挂 (1600)
SAM-BA v2.6和NandFla (1388)
ARM9启动分析--存储器I (1289)
linux内核编译问题解决方 (1106)

评论排行

HI3518E用J-link烧写裸机 (6)
HI3516A 高清4K支持H2((5)
S7-200西门子PLC自由口 (3)
Lattice FPGA中假双口R/ (2)
linux/videodev.h: No suc (1)
SAM-BA和AT91SAM926 (1)
基于AT91SAM9G10（A (1)
linux内核编译问题解决方 (1)
AT45DB041D存储驱动设 (1)
RT3070 USB WIFI 在连接 (1)

推荐文章

* 程序员的一天
* 如何面试程序员
* 只有专家才能做这事
* 你的企业一定要转型吗？
* Android悬浮框 覆盖与被覆盖
* HTTP POST请求报文格式分析与Java实现文件上传

最新评论

HI3516A 高清4K支持H265/HEV
lpower9: 能发给我一份吗？我的QQ:331234250

HI3516A 高清4K支持H265/HEV
joan2004: 我需要，能给我一份HI3516A的SDK吗？我的QQ147943867

HI3516A 高清4K支持H265/HEV
mobeilangzibai: @lpower9:我有QQ 1272782960

HI3516A 高清4K支持H265/HEV
scsi000: 我需要，请与我联系。QQ： 851800214

HI3516A 高清4K支持H265/HEV
lpower9: 好东西，我有需求，请联系我,

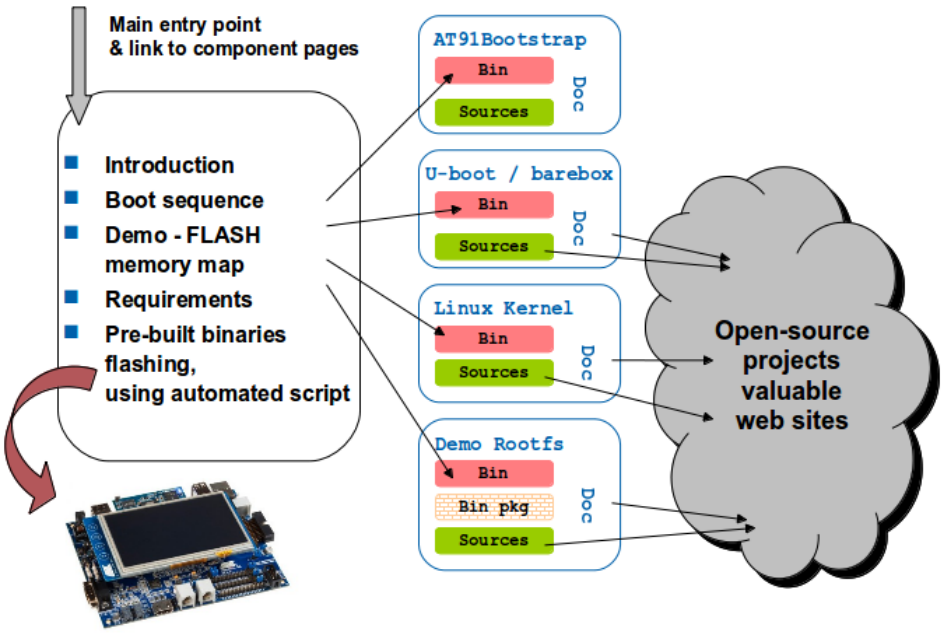
HI3518E用J-link烧写裸板fastboo
lichangc: @wonrowl:不能直接用，必须要修改！

HI3518E用J-link烧写裸板fastboo
wonrowl: @lichangc:谢谢回复,启动选择成spi,启动 ,内存初始化脚本用sdk包里的Hi3518_...

HI3518E用J-link烧写裸板fastboo
lichangc: @wonrowl:我烧写的是SPI Flash，两个Flash都是可以烧写的。是用JTAG廉价的方式...

HI3518E用J-link烧写裸板fastboo
wonrowl: 楼主,你好,请问下你的烧写是烧写进的nandflash还是spi flash啊,另外是jtag的连接...

HI3518E用J-link烧写裸板fastboo
lichangc: @staacloo:QQ:670733516



基础知识

- AT91SAM926x是什么？

如果你还没有搞懂这个问题，请不要继续往下看。请先参观一下 本站论坛。

- Linux是什么？

简单地说,Linux是一套免费使用和自由传播的类Unix操作系统，它主要用于基于Intel x86系列CPU的计算机上。这个系统是由世界各地的成千上万的程序员设计和实现的。其目的是建立不受任何商品化软件的版权制约的、全世界都能自由使用的 Unix兼容产品。 Linux的出现，最早开始于一位名叫 Linus Torvalds的计算机业余爱好者，当时他是芬兰赫尔辛基大学的学生。他的目的是想设计一个代替 Minix（是由一位名叫 Andrew Tannebaum的计算机教授编写的一个操作系统示教程序）的操作系统，这个操作系统可用于 386、486或奔腾处理器的个人计算机上，并且具有 Unix操作系统的全部功能，因而开始了 Linux雏形的设计。 Linux以它的高效性和灵活性著称。它能够在 PC计算机上实现全部的 Unix特性，具有多任务、多用户的能力。 Linux是在 GNU公共许可权限下免费获得的，是一个符合 POSIX标准的操作系统。 Linux操作系统软件包不仅包括完整的 Linux操作系统，而且还包括了文本编辑器、高级语言编译器等应用软件。它还包括带有多个窗口管理器的 X-Windows图形用户界面，如同我们使用 Windows NT一样，允许我们使用窗口、图标和菜单对系统进行操作。

- Linux与其他操作系统有什么区别?

Linux可以与 MS-DOS、OS/2、Windows等其他操作系统共存于同一台机器上。它们均为操作系统，具有一些共性，但是互相之间各有特色，有所区别。 目前运行在 PC机上的操作系统主要有Microsoft的MS-DOS、Windows、 Windows NT、 IBM的 OS/2等。早期的 PC机用户普遍使用 MS-DOS，因为这种操作系统对机器的硬件配置要求不高，而随着计算机硬件技术的飞速发展，硬件设备价格越来越低，人们可以相对容易地提高计算机的硬件配置，于是开始使用 Windows、Windows NT等具有图形界面的操作系统。 Linux是新近被人们所关注的操作系统，它正在逐渐为PC机的用户所接受。那么，Linux与其他操作系统的主要区别是什么呢？下面从两个方面加以论述。 首先看一下Linux与MS-DOS之间的区别。在同一系统上运行Linux和MS-DOS已很普遍，就发挥处理器功能来说，MS-DOS没有完全实现x86处理器的功能，而Linux完全在处理器保护模式下运行，并且开发了处理器的所有特性。Linux可以直接访问计算机内的所有可用内存，提供完整的Unix接口。而MS-DOS只支持部分Unix的接口。 就使用费用而言，Linux和MS-DOS是两种完全不同的实体。与其他商业操作系统相比，MS-DOS价格比较便宜，而且在PC机用户中有很大的占有率，任何其他 PC机操作系统都很难达到MS-DOS的普及程度，因为其他操作系统的费用对大多数PC机用户来说都是一个不小的负担。Linux是免费的，用户可以从internet上或者其他途径获得它的版本，而且可以任意使用，不用考虑费用问题。 就操作系统的功能来说，MS-DOS是单任务的操作系统，一旦用户运行了一个MS-DOS的应用程序，它就独占了系统的资源，用户不可能再同时运行其他应用程序。而Linux是多任务的操作系统，用户可以同时运行多个应用程序。

- 从哪里可以获得Linux的源代码？

kernel source
linux4sam kernel patch

- 什么是GPL？

通用性公开许可证(General Public License，简称GPL)。GPL同其它的自由软件许可证一样，许可社会公众享有运行、复制软件的自由；发行传播软件的自由；获得软件源码的自由，以及改进软件并将自己作出的改进版本向社会发行传播的自由。GPL还规定，只要这种修改文本的整体或者其某个部分来源于遵循GPL的程序，则该修改文本的整体就必须按照GPL流通，不仅该修改文本的源码必须向社会公开，而且对于这种修改文本的流通不准许附加修改者自己做出的限制。因此，遵循GPL流通的程序不能同非自由的软件合并。GPL所表达的这种流通规则称为copyleft，表示与copyright(版权)的概念“相左”。

- 哪里有GPL的许可证文本？

一般遵循GPL的软件都会在软件包内包含GPL许可证文本。
也可以到下面网站查看 GNU General Public License。

开发环境

开发Linux应用，需要：

- 一台Linux PC(可以是虚拟机，以下简称为Linux开发机)

可以安装主流的Linux发行版本，比如 ubuntu,Fedora等等。
标准的发行版本可能并没有包括所有开发中需要的组件。开发过程中可以根据需要通过网络安装。
如果采用虚拟机方式，可以选择虚拟机软件如 Virtual PC,VMware, VirtualBox等。安装完成后需要”打通”虚拟机与宿主机的共享通道，可以使用Linux的samba或者使用虚拟软件提供的第三方工具。
使用虚拟机，请保证虚拟机硬盘足够大。可以使用8GB硬盘作为第一硬盘，安装系统；而另外再虚拟一块硬盘，用于编译和工作。

- 可用的网络

基于两个原因：
1. 开发初期使用NFS会比较方便。
2. Linux开发机安装组件。

- 开发机上所需要的基本程序开发工具

比如gcc(GCC-HOWTO)等。

- 目标板所用的交叉编译器

所谓交叉编译器，就是指在开发机上运行，编译结果在另外架构的平台上运行。
这里的交叉编译器指的就是ARM交叉编译器。编译器在Linux x86上将源代码编译成ARM体系结构适用的可执行文件。
Linux下可以使用的ARM交叉编译器很多，经过测试，本站推荐使用arm-none-linux-gnueabi，在下载页面选择GNU/Linux对应的软件包。
下载后在Linux开发机上解压缩，并将工具链bin文件夹(arm-none-linux-gnueabi-gcc所在文件夹)添加到系统PATH,即可完成安装。

- 目标烧写(编程)工具

有了编译的输出，还需要编程工具将其固化到SAM926x的板子上运行。这个过程就是编程的过程。
由于SAM926x提供了SAM-BA工具，所以最基本的方式就是通过USB口利用SAM-BA软件下载。 SAM-BA软件使用教程。
如果对编程速度有高的要求，可以选择第三方编程器。

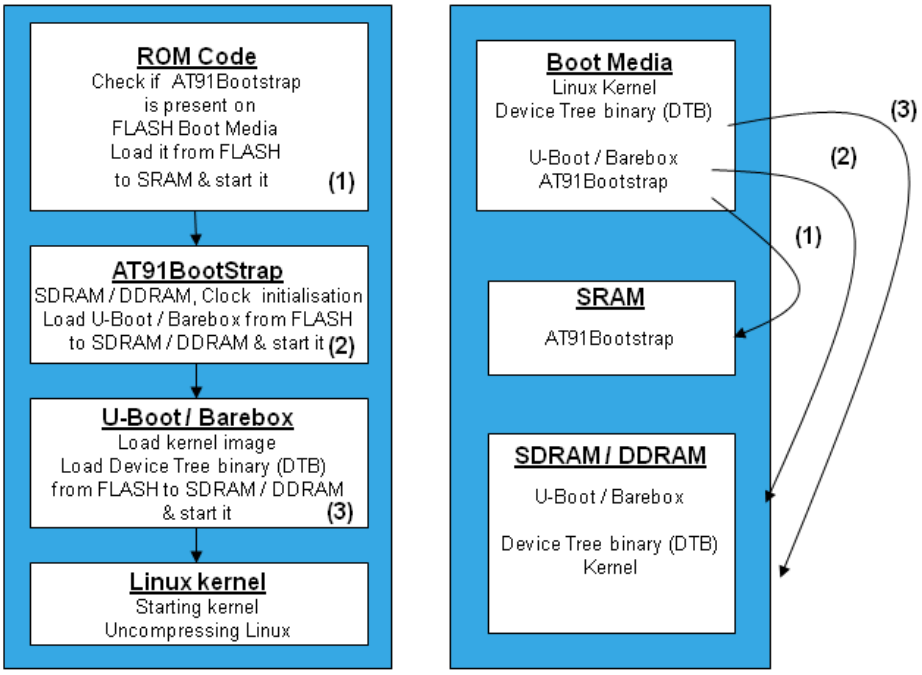
- 硬件仿真器

可以用于对bootstrap与u-boot进行调试，方便定位可能的问题。
Linux下可以使用软件的方法debug，比如打log，或者使用 GDB。

SAM926x Linux 基础

启动过程

本文主要讲述从Data Flash上启动的过程，NAND flash启动，NOR Flash启动请参考相关文档。



1. 处理器复位，根据BMS引脚的设置，选择从内部ROM启动(NOR Flash启动方式不在本文的讨论范围)。
2. ROM中的代码开始运行，初始化处理器和必要的外设，比如DBGU，USB device port。然后开始从Data Flash或NAND的0地址检索合法(?具体参考数据手册的boot program的章节)的启动程序，也就是Bootstrap。
3. 如果合理的Bootstrap存在，ROMBOOT将其复制到内部SRAM并跳转到Bootstrap运行。如果没有，检测其它支持启动的存储介质,如果都没有,则等待DBGU或者USB口的连接，这部分内容请参考SAM-BA手册。
4. Bootstrap将初始化一些设备，主要是Data Flash或NAND与SDRAM，然后从Data Flash或NAND的特定位置(在Bootstrap源代码中指定)将U-boot复制到SDRAM的指定位置，然后跳转到U-boot开始位置运行。
5. U-boot根据环境变量(bootcmd)加载Linux Kernel的image。加载完成后跳转到Kernel运行，并传递启动参数(bootargs)。
6. Linux Kernel开始运行，加载相关驱动，并加载rootfs。其中的细节，可以参考Linux书籍。

Data Flash Boot Mem Map

- ! Internal Flash 在sam926x上是ROM
- ! Data Flash的分区在u-boot下指定，上图显示的是新分区，和现在的例子不同
- ! NAND FLASH(128MB)平均分为两个分区，分区0作为rootfs(JFFS2格式)，分区1由用户定义

NAND Flash Boot Mem Map

- 图中所示为页面大小为2048字节的NAND,一个block(NAND擦除的最小单位)为128KB,也就是0x20000
- 图上为ATMEL官方分区,kernel image从2MB开始,到4MB结束,!大小不能超过2MB
- 本站例子中的kernel image从0xA0000开始,结束地址相同,也是4MB,使得kernel image可以超过3MB

Bootstrap

Bootstrap用于加载一段程序到SDRAM运行，本文关注于加载u-boot的应用。

Bootstrap可以称作(程序员可见的)一级boot，实际上芯片内部还有个BOOTROM，也就是SAM-BA的启动程序。关于芯片内部BOOTROM程序的运行流程和详细描述，请参考数据手册的相关章节(Boot Program)。

Bootstrap的代码由芯片上的BOOTROM根据一定的规则(0x14处的数据由特殊含义，这也是烧写一级boot时需要选择send boot file的原因)加载到内部SRAM运行。

Bootstrap的代码很短，主要原因在于SAM926x内部SRAM的限制。其主要工作就是初始化SDRAM和相关存储器(Data Flash, NAND),然后加载u-boot到SDRAM指定位置并开始运行u-boot。

下载**Bootstrap**代码

ATMEL bootstrap源代码

下载后展开压缩包。

Bootstrap源代码结构

Bootstrap源代码由共用的硬件驱动，比如SDRAM，NAND，DATA Flash等，库文件，头文件等组成。

新的Bootstrap代码中包含了一个PDF格式的说明文件，包含在doc文件夹下。

主要的地方在board文件夹下，也就是不同板子的项目所在。根据板子的启动配置不同，同一个板子的文件夹下有不同的目标配置，比如dataflash与nandflash。进入不同的目标配置文件就可以看到该目标配置的配置文件(*.h)及Makefile。

Bootstrap目标配置文件

以SAM9261 Data flash bootstrap为例说明配置文件的作用：

Bootstrap-v1.11\board\at91sam9261ek\dataflash\at91sam9261ek.h:

```
#ifndef _AT91SAM9261EK_H
#define _AT91SAM9261EK_H

/* ********************************************************************** */
/* PMC Settings                                                           */
/*                                                                        */
/* The main oscillator is enabled as soon as possible in the c_startup */
/* and MCK is switched on the main oscillator.                            */
/* PLL initialization is done later in the hw_init() function          */
/* ********************************************************************** */
#define MCK_100

#ifdef MCK_100

#define MASTER_CLOCK            (198656000/2)
#define PLL_LOCK_TIMEOUT        1000000

#define PLLA_SETTINGS    0x2060BF09
#define PLLB_SETTINGS    0x10483F0E

/* AC characteristics */
/* DLYBS = tCSS= 250ns min and DLYBCT = tCSH = 250ns */
#define DATAFLASH_TCSS          (0x1a << 16)     /* 250ns min (tCSS) <=> 12/48000000 = 250ns */
#define DATAFLASH_TCHS          (0x1 << 24)      /* 250ns min (tCSH) <=> (64*1+SCBR)/(2*48000000)
*/

#else   /* 133 MHz */

#define MASTER_CLOCK            (250000000/2)
#define PLL_LOCK_TIMEOUT        1000000

#define PLLA_SETTINGS    0x20D8BF10
#define PLLB_SETTINGS    0x10483F0E
/*
#define MASTER_CLOCK            (266000000/2)
#define PLL_LOCK_TIMEOUT        1000000

#define PLLA_SETTINGS    0x2064BF07
#define PLLB_SETTINGS    0x10483F0E
*/
#define DATAFLASH_TCSS          (0xf << 16)      /* 250ns min (tCSS) <=> 12/48000000 = 250ns */
#define DATAFLASH_TCHS          (0x2 << 24)      /* 250ns min (tCSH) <=> (64*1+SCBR)/(2*48000000)
*/
```

```
#endif

/* Switch MCK on PLLA output PCK = PLLA = 2 * MCK */
#define MCKR_SETTINGS           (AT91C_PMC_PRES_CLK | AT91C_PMC_MDIV_2)
#define MCKR_CSS_SETTINGS       (AT91C_PMC_CSS_PLLA_CLK | MCKR_SETTINGS)

/* ********************************************************************** */
/* DataFlash Settings                                                     */
/*                                                                        */
/* ********************************************************************** */
#define AT91C_BASE_SPI  AT91C_BASE_SPI0
#define AT91C_ID_SPI    AT91C_ID_SPI0

/* SPI CLOCK */
#define AT91C_SPI_CLK           33000000

#define DF_CS_SETTINGS          (AT91C_SPI_NCPHA | (AT91C_SPI_DLYBS & DATAFLASH_TCSS) |
(AT91C_SPI_DLYBCT & DATAFLASH_TCHS) | ((MASTER_CLOCK / AT91C_SPI_CLK) << 8))

/* ********************************************************************** */
/* BootStrap Settings                                                     */
/*                                                                        */
/* ********************************************************************** */
#define AT91C_SPI_PCS_DATAFLASH         AT91C_SPI_PCS0_DATAFLASH        /* Boot on SPI NCS0 */

#define IMG_ADDRESS             0x8400                  /* Image Address in DataFlash */
#define IMG_SIZE                0x33900                 /* Image Size in DataFlash    */

#define MACH_TYPE       0x350       /* AT91SAM9261-EK */
#define JUMP_ADDR               0x23F00000              /* Final Jump Address         */

/* ********************************************************************** */
/* Application Settings                                                   */
/* ********************************************************************** */
#undef CFG_DEBUG
#define CFG_DATAFLASH
#define CFG_HW_INIT
#define CFG_SDRAM

#endif  /* _AT91SAM9261EK_H */
```

- 根据data flash配置，设置时序及data flash片选
- 设置被加载的代码(u-boot, for Linux)的加载源地址(data flash中)，及加载目标地址(SDRAM中)，以及代码加载长度

加载的起始地址必须与分区表一致，代码长度也必须适合编译输出的u-boot。

- 定义为Dataflash启动，不使用DEBUG的输出

下载**Windows**下的编译器**arm-elf-gcc**

WinARM:The gnu-toolchain and several tools and samples for ARM controller/processors for MS-Windows-Platforms

GNU ARM™ toolchain for CygWin, Linux and MacOS

YAGARTO GNU ARM toolchain

Sourcery G++ Lite ARM EABI

选一即可。

直接安装即可，不同的软件可能需要不同的配置，比如添加路径到PATH。可以参考软件的release notes。

下面的编译过程以GNU ARM + cygwin环境为例。

检验编译器安装：

$ which arm-elf-gcc

/cygdrive/d/_dsetup/arm/gnuarm/bin/arm-elf-gcc

$ arm-elf-gcc -v

Using built-in specs.

Target: arm-elf

Configured with: ../gcc-4.3.2/configure –target=arm-elf –prefix=/c/gnuarm –enable-interwork –enable-multilib –with-

float=soft –with-newlib –with-headers=../newlib-1.16.0/newlib/libc/include –enable-languages=c,c++

Thread model: single

gcc version 4.3.2 (GCC)

编译Bootstrap

Bootstrap for SAM9261 Dataflash

在\Bootstrap-v1.11\board\at91sam9261ek\dataflash文件下输入：

```
$ make
```

输出：

rm -f *.o *.bin *.elf *.map

arm-elf-gcc -g -mcpu=arm9 -c -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash -DTOP_OF_MEM=0x328000 ../../../crt0_gnu.S -o crt0_gnu.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../board/at91sam9261ek/at91sam9261ek.c -o at91sam9261ek.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../main.c -o main.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../driver/gpio.c -o gpio.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../driver/pmc.c -o pmc.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../driver/debug.c-o debug.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../driver/sdramc.c -o sdramc.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../driver/dataflash.c -o dataflash.o

arm-elf-gcc -g -mcpu=arm9 -c -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash -DTOP_OF_MEM=0x328000 ../../../lib/_udivsi3.S -o _udivsi3.o

arm-elf-gcc -g -mcpu=arm9 -c -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash -DTOP_OF_MEM=0x328000 ../../../lib/_umodsi3.S -o _umodsi3.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../lib/div0.c -o div0.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../lib/udiv.c -o udiv.o

arm-elf-gcc -c -g -mcpu=arm9 -Os -Wall -DAT91SAM9261 -I./../../../board/at91sam9261ek/dataflash ../../../lib/string.c -o string.o

arm-elf-gcc -nostartfiles -nostdlib -Wl,-Map=dataflash_at91sam9261ek.map,–cref -T ../../../elf32-littlearm.lds -Ttext 0x300000 -n -o dataflash_at91sam9261ek.elf crt0_gnu.o at91sam9261ek.o main.o gpio.o pmc.o debug.o sdramc.o dataflash.o _udivsi3.o _umodsi3.o div0.o udiv.o string.o

arm-elf-objcopy –strip-debug –strip-unneeded dataflash_at91sam9261ek.elf -O binary dataflash_at91sam9261ek.bin

生成的dataflash_at91sam9261ek.bin就是Bootstrap的启动文件，需要通过SAM-BA 以send boot file的方式烧写到data flash。

Bootstrap for SAM9263 Dataflash

由于官方默认时钟频率为16MHz，而板子上使用的是18MHz，因此需要修改源代码：Bootstrap-v1.11\board\at91sam9263ek\dataflash\at91sam9263ek.h:

```
#define    CRYSTAL_16_36766MHZ  1
```

修改为

```
#define    CRYSTAL_18_432MHZ    1
```

修改代码后使用make即可编译生成板子适用的data flash bootstrap的代码。

Bootstrap for SAM9263 Nand

由于官方默认时钟频率为16MHz，而板子上使用的是18MHz，因此需要修改源代码：Bootstrap-v1.11\board\at91sam9263ek\nandflash\at91sam9263ek.h:

```
#define    CRYSTAL_16_36766MHZ  1
```

修改为

```
#define    CRYSTAL_18_432MHZ    1
```

修改代码后使用make即可编译生成板子适用的nand bootstrap的代码。

U-boot

U-boot是一个庞大的公开源码的软件。支持一系列的ARM体系(但不仅限于ARM体系结构)，包含常见的外设的驱动，是一个功能强大的板极支持包。

U-BOOT是由PPCBOOT发展起来的，是PowerPC、ARM9、Xscale、X86等系统通用的Boot方案。u-boot是一个open source的bootloader。

为什么需要u-boot？显然可以将Linux直接烧入flash，仅使用简单的引导装载程序（bootloader）加载。但是从软件升级的角度，程序的健壮程度，支持的外设等等来说，一款强大的bootloader能极大提高程序开发效率：

1. U-boot可以在开发初期测试大部分外设
2. U-boot可以使用网络，USB等方式加载kernel image，不需要每次都烧写，既节省时间，又避免flalsh/NAND的损坏
3. 丰富的FLASH/NAND编程功能的支持，可以用于代码的烧写

下载U-boot源代码

- U-boot源代码

该页面提供了一个FTP链接，直接从其FTP下载。

本文使用的U-boot版本为2008.10。

对于下载下来的u-boot-2008.10.tar.bz2文件，使用下面命令解压：

tar jxvf u-boot-2008.10.tar.bz2

U-boot代码简介

U-boot的代码比较多，但是一般开发过程中关注board与inculde/configs即可。

board目录包含了各种板子的绝大部分板级支持。

inculde/configs中以板子命名的头文件定义了该板子的各种配置，比如对网络的支持，JFFS2，USB的支持等。修改头文件就可以改变板子的配置。

- 读9261的U-boot代码获得的函数调用关系
- 使用AXD在9261上调试u-boot

更改U-boot源代码

1. 修改Makefile以指定工具链

u-boot-2008.10/Makefile:

默认使用arm-linux-工具链：

```
ifeq ($(ARCH),arm)
CROSS_COMPILE = arm-linux-
endif
```

修改为上面安装的arm-none-linux-gnueabi-工具链：

```
ifeq ($(ARCH),arm)
CROSS_COMPILE = arm-none-linux-gnueabi-
endif
```

1. 针对本站的SAM926x开发板，必须修改源代码的一些相关配置部分，比如时钟频率(SAM9263),LCD时序参数(SAM9261,SAM9263)。

SAM9261

需要修改LCD时序参数，以适合本站的竖屏。

u-boot-2008.10\board\atmel\at91sam9261ek\at91sam9261ek.c:修改结构体：

```
vidinfo_t panel_info = {
        vl_col:         240,
        vl_row:         320,
        vl_clk:         4965000,
        vl_sync:        ATMEL_LCDC_INVLINE_INVERTED |
                        ATMEL_LCDC_INVFRAME_INVERTED,
        vl_bpix:        3,
        vl_tft:         1,
        vl_hsync_len:   5,
        vl_left_margin: 1,
        vl_right_margin:33,
        vl_vsync_len:   1,
        vl_upper_margin:1,
        vl_lower_margin:0,
        mmio:           AT91SAM9261_LCDC_BASE,
};
```

为

```
vidinfo_t panel_info = {
        vl_col:         240,
        vl_row:         320,
        vl_clk:         4965000,
        vl_sync:        ATMEL_LCDC_INVLINE_INVERTED |
                        ATMEL_LCDC_INVFRAME_INVERTED,
        vl_bpix:        3,
        vl_tft:         1,
        vl_hsync_len:   96,
        vl_left_margin: 48,
        vl_right_margin:16,
        vl_vsync_len:   2,
        vl_upper_margin:31,
        vl_lower_margin:12,
        mmio:           AT91SAM9261_LCDC_BASE,
};
```

SAM9263

- 修改LCD时序参数：

u-boot-2008.10\board\atmel\at91sam9263ek\at91sam9263ek.c:

```
vidinfo_t panel_info = {
        vl_col:         240,
        vl_row:         320,
        vl_clk:         4965000,
        vl_sync:        ATMEL_LCDC_INVLINE_INVERTED |
                        ATMEL_LCDC_INVFRAME_INVERTED,
```

```
        vl_bpix:        3,
        vl_tft:         1,
        vl_hsync_len:   5,
        vl_left_margin: 1,
        vl_right_margin:33,
        vl_vsync_len:   1,
        vl_upper_margin:1,
        vl_lower_margin:0,
        mmio:           AT91SAM9263_LCDC_BASE,
};
```

修改为：

```
vidinfo_t panel_info = {
        vl_col:         240,
        vl_row:         320,
        vl_clk:         4965000,
        vl_sync:        ATMEL_LCDC_INVLINE_INVERTED |
                        ATMEL_LCDC_INVFRAME_INVERTED,
        vl_bpix:        3,
        vl_tft:         1,
        vl_hsync_len:   96,
        vl_left_margin: 48,
        vl_right_margin:16,
        vl_vsync_len:   2,
        vl_upper_margin:31,
        vl_lower_margin:12,
        mmio:           AT91SAM9263_LCDC_BASE,
};
```

- 时钟频率

u-boot-2008.10\include\configs\at91sam9263ek.h:
从16MHz

```
/* ARM asynchronous clock */
#define AT91_CPU_NAME           "AT91SAM9263"
#define AT91_MAIN_CLOCK         199919000       /* from 16.367 MHz crystal */
#define AT91_MASTER_CLOCK       99959500        /* peripheral = main / 2 */
#define CFG_HZ                  1000000         /* 1us resolution */

#define AT91_SLOW_CLOCK         32768   /* slow clock */
```

修改为18MHz：

```
/* ARM asynchronous clock */
#define AT91_CPU_NAME           "AT91SAM9263"
#define AT91_MAIN_CLOCK         (198656000)     /* from 18.432 MHz crystal */
#define AT91_MASTER_CLOCK       (198656000/2)   /* peripheral = main / 2 */
#define CFG_HZ                  1000000         /* 1us resolution */

#define AT91_SLOW_CLOCK         32768   /* slow clock */
```

编译U-boot

在编译前运行下面命令：

```
# make distclean
```

清除以前所有编译结果。

SAM9261

```
# make at91sam9261ek_config
# make
```

SAM9263

```
# make at91sam9263ek_config
# make
```

编译输出

- u-boot-2008.10/u-boot

u-boot编译输出的ELF文件，可以用于调试。

- u-boot-2008.10/u-boot.bin

u-boot.bin就是u-boot的烧写文件，按照分区表(!必须与Bootstrap的一致)，将其烧写，并由Bootstrap加载。
!更改配置编译后需要注意u-boot.bin的大小，保证Bootstrap会将全部的u-boot.bin复制到SDRAM并运行。

- u-boot-2008.10/tools/mkimage

用于将Linux kernel编译输出的zImage转换为U-boot所支持的uImage。
将此文件安装到Linux PC，如将此文件复制到/usr/local/sbin，并将/usr/local/sbin添加到系统路径(一般Ubuntu默认系统路径就包含/usr/local/sbin，可以使用echo $PATH检查)。
mkimage使用详解

U-boot环境变量

U-Boot通过环境变量（env）为用户提供一定程度的可配置性，这些环境变量包括串口终端所使用的波特率（baudrate）、启动操作系统内核的参数（bootargs）、本地IP地址（ipaddr）、网卡MAC地址（ethaddr）等等.
环境变量可以固化到非易失性存储介质中，使用printenv / saveenv命令来查看和修改。

Kernel

现在的Linux Kernel源码包很大，包含了各种体系架构，板级支持包，设备驱动。
编译Kernel源代码需要一些必要的基础支持，推荐大家google。
下面的描述以 Linux-2.6.27为例。

下载Kernel源代码

- linux-2.6.27.tar.bz2
- 2.6.27-at91.patch.gz
- 2.6.27-at91-exp.3.patch.gz

展开内核源码包:

```
# tar jxvf linux-2.6.27.tar.bz2
```

将两个补钉文件直接复制到展开的内核源代码目录下，并在内核目录下运行系列命令按顺序打补钉：

```
# zcat 2.6.27-at91.patch.gz | patch -p1
# zcat 2.6.27-at91-exp.3.patch.gz | patch -p1
```

修改Kernel源代码

SAM9261

- 修改NAND分区

linux-2.6.27\arch\arm\mach-at91\board-sam9261ek.c:

```
/*
 * NAND flash
 */
static struct mtd_partition __initdata ek_nand_partition[] = {
        {
```

```
                .name   = "Bootstrap",
                .offset = 0,
                .size   = SZ_4M,
        },
        {
                .name   = "Partition 1",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = 60 * SZ_1M,
        },
        {
                .name   = "Partition 2",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = MTDPART_SIZ_FULL,
        },
};
```

本站的板子将128MB NAND平均分为两个分区，分区0作为rootfs，分区1未使用:
如果分区不同，在使用本站的一些例子时会出现VFS mount fail错误。

```
/*
 * NAND flash
 */
static struct mtd_partition __initdata ek_nand_partition[] = {
        {
                .name   = "Partition 1",
                .offset = 0,
                .size   = 64 * SZ_1M,
        },
        {
                .name   = "Partition 2",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = MTDPART_SIZ_FULL,
        },
};
```

- 修改LCD时序

linux-2.6.27\arch\arm\mach-at91\board-sam9261ek.c:

```
/* TFT */
static struct fb_videomode at91_tft_vga_modes[] = {
        {
                .name           = "TX09D50VM1CCA @ 60",
                .refresh        = 60,
                .xres           = 240,          .yres           = 320,
                .pixclock       = KHZ2PICOS(4965),

                .left_margin    = 1,            .right_margin   = 33,
                .upper_margin   = 1,            .lower_margin   = 0,
                .hsync_len      = 5,            .vsync_len      = 1,

                .sync           = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
                .vmode          = FB_VMODE_NONINTERLACED,
        }
};
```

修改为本站的竖屏时序：

```
/* TFT */
static struct fb_videomode at91_tft_vga_modes[] = {
        {
                .name           = "TX09D50VM1CCA @ 60",
                .refresh        = 60,
                .xres           = 240,          .yres           = 320,
                .pixclock       = KHZ2PICOS(4965),

                .left_margin    = 48,           .right_margin   = 16,
                .upper_margin   = 31,           .lower_margin   = 12,
                .hsync_len      = 96,           .vsync_len      = 2,

                .sync           = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
                .vmode          = FB_VMODE_NONINTERLACED,
        }
};
```

- 添加ttyS*

linux-2.6.27\arch\arm\mach-at91\board-sam9261ek.c:

```
static void __init ek_map_io(void)
{
        /* Initialize processor: 18.432 MHz crystal */
        at91sam9261_initialize(18432000);

        /* Setup the LEDs */
        at91_init_leds(AT91_PIN_PA13, AT91_PIN_PA14);

        /* DGBU on ttyS0. (Rx & Tx only) */
        at91_register_uart(0, 0, 0);

        /* set serial console to ttyS0 (ie, DBGU) */
        at91_set_serial_console(0);
}
```

注册USART0到ttyS1, USART1到ttyS2：

```
static void __init ek_map_io(void)
{
        /* Initialize processor: 18.432 MHz crystal */
        at91sam9261_initialize(18432000);

        /* Setup the LEDs */
        at91_init_leds(AT91_PIN_PA13, AT91_PIN_PA14);

        /* DGBU on ttyS0. (Rx & Tx only) */
        at91_register_uart(0, 0, 0);

        /* USART0 on ttyS1. (Rx & Tx only) */
        at91_register_uart(AT91SAM9261_ID_US0, 1, 0);

        /* USART1 on ttyS2. (Rx & Tx only) */
        at91_register_uart(AT91SAM9261_ID_US1, 2, 0);

        /* set serial console to ttyS0 (ie, DBGU) */
        at91_set_serial_console(0);
```

```
}
```

:!:USART2的TXD与RXD所在引脚被NAND占用，因此不可使用。

添加后Linux启动时的信息提示:

```
atmel_usart.0: ttyS0 at MMIO 0xfefff200 (irq = 1) is a ATMEL_SERIAL
atmel_usart.1: ttyS1 at MMIO 0xfffb0000 (irq = 6) is a ATMEL_SERIAL
atmel_usart.2: ttyS2 at MMIO 0xfffb4000 (irq = 7) is a ATMEL_SERIAL
```

SAM9263

- 修改分区

对于data flash启动方式，修改为两个分区:

如果分区不同，在使用本站的一些例子时会出现VFS mount fail错误。

linux-2.6.27\arch\arm\mach-at91\board-sam9263ek.c:

```
/*
 * NAND flash
 */
static struct mtd_partition __initdata ek_nand_partition[] = {
        {
                .name   = "Bootstrap",
                .offset = 0,
                .size   = SZ_4M,
        },
        {
                .name   = "Partition 1",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = 60 * SZ_1M,
        },
        {
                .name   = "Partition 2",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = MTDPART_SIZ_FULL,
        },
};
```

修改为:

```
/*
 * NAND flash
 */
static struct mtd_partition __initdata ek_nand_partition[] = {
        {
                .name   = "Partition 1",
                .offset = 0,
                .size   = 64 * SZ_1M,
        },
        {
                .name   = "Partition 2",
                .offset = MTDPART_OFS_NXTBLK,
                .size   = MTDPART_SIZ_FULL,
        },
};
```

- 修改LCD时序

linux-2.6.27\arch\arm\mach-at91\board-sam9263ek.c:

```
/*
 * LCD Controller
 */
#if defined(CONFIG_FB_ATMEL) || defined(CONFIG_FB_ATMEL_MODULE)
static struct fb_videomode at91_tft_vga_modes[] = {
        {
                .name           = "TX09D50VM1CCA @ 60",
                .refresh        = 60,
                .xres           = 240,          .yres           = 320,
                .pixclock       = KHZ2PICOS(4965),

                .left_margin    = 1,            .right_margin   = 33,
                .upper_margin   = 1,            .lower_margin   = 0,
                .hsy            5,              .vsync_len      = 1,

                .syn            FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
                .vmo            FB_VMODE_NONINTERLACED,
        },
};
```

修改为：

```
/*
 * LCD Controller
 */
#if defined(CONFIG_FB_ATMEL) || defined(CONFIG_FB_ATMEL_MODULE)
static struct fb_videomode at91_tft_vga_modes[] = {
        {
                .name           = "TX09D50VM1CCA @ 60",
                .refresh        = 60,
                .xres           = 240,          .yres           = 320,
                .pixclock       = KHZ2PICOS(4965),

                .left_margin    = 16,           .right_margin   = 12,
                .upper_margin   = 1,            .lower_margin   = 0,
                .hsync_len      = 96,           .vsync_len      = 2,

                .sync           = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
                .vmode          = FB_VMODE_NONINTERLACED,
        },
};
```


资源： 91

- 修改时钟频率

linux-2.6.27\arch\arm\mach-at91\board-sam9263ek.c:
从16MHz

```
static void __init ek_map_io(void)
{
        /* Initialize processor: 16.367 MHz crystal */
        at91sam9263_initialize(16367660);

        /* DGBU on ttyS0. (Rx & Tx only) */
        at91_register_uart(0, 0, 0);

        /* USART0 on ttyS1. (Rx, Tx, RTS, CTS) */
        at91_register_uart(AT91SAM9263_ID_US0, 1, ATMEL_UART_CTS | ATMEL_UART_RTS);
```

```
        /* set serial console to ttyS0 (ie, DBGU) */
        at91_set_serial_console(0);
}
```

修改为18MHz：

```
static void __init ek_map_io(void)
{
        /* Initialize processor: 16.367 MHz crystal */
        at91sam9263_initialize(18432000); // 16367660

        /* DGBU on ttyS0. (Rx & Tx only) */
        at91_register_uart(0, 0, 0);

        /* USART0 on ttyS1. (Rx, Tx, RTS, CTS) */
        at91_register_uart(AT91SAM9263_ID_US0, 1, ATMEL_UART_CTS | ATMEL_UART_RTS);

        /* set serial console to ttyS0 (ie, DBGU) */
        at91_set_serial_console(0);
}
```

配置Kernel源代码

Linux内核配置选项详解

SAM9261

at91sam9261ek_defconfig供参考下载 at91sam9261ek_defconfig，并将其复制到内核源代码根目录下并改名为.config,覆盖原来的配置文件。

下载保存时注意文件类型，该文件时一个Linux文本文件(LF)，而不是DOS格式文本文件(CRLF)。

```
# cp at91sam9261ek_defconfig .config
```

运行kernel配置工具，比如make config(最基本的界面), make menuconfig(基于 ncurses 的界面), make gconfig(基于 gtk+ 的图形界面), make xconfig(基于 qt 的图形界面)。注意，各种图形界面依赖于开发机(Linux PC)上所安装的图形库，如果图形库不全，将不能启动。

在命令行启动配置时，不要忘记加上ARCH类型。比如：

```
make xconfig ARCH=arm
```

通过图形界面配置比较方便，可以根据需要对内核的部分进行调整：

- 使用touchscreen需要使能SPI，不选MMC/SD支持
- 使用SD需要不选SPI，选择MMC/SD

配置完成，保存退出。运行下面命令编译内核，工具链使用arm-none-linux-gnueabi-：

```
# make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

视Linux PC的配置不同，完成编译的时间也不同。

SAM9263

at91sam9263ek_defconfig供参考

使用kernel image

内核编译完成后会在 linux-2.6.27/arch/arm/boot 下生成zImage，这个就是Linux Kernel编译的输出文件。

下面需要使用编译U-boot生成mkimage将zImage转换为uImage：

```
# mkimage -A arm -O linux -T kernel -C none -a 0x20008000 -e 0x20008000 -n 'Linux-2.6.27' -d
./zImage ./uImage27.bin
```

运行该命令即可生成U-boot可以加载的烧写文件。将uImage27.bin烧写到分区指定地址，通过设置U-boot下的环境变量加载。

Rootfs

busybox 源代码下载

926x开发板文档——使用busybox制作根文件系统

开发资料

LCD 时序

LCD类型	LCD尺寸	xres	yres	pixclock	left_margin	right_margin	upper_margin	lower_margin	hsync_
QVGA(240x320)	3.5”竖	240	320	4965KHz	48	16	12	31	96
480x272	4.3”宽	480	272	9000KHz	2	2	2	2	41
VGA(640×480)	5.6”横	640	480	25175KHz	19	45	12	31	96
WVGA(800x480)	5.0”横	800	480	33000KHz	40	40	29	13	48

相关链接

- 在Windows PC上建立Linux开发环境(ubuntu, VirtualBox)
- 带有官方性质的SAM9 Linux专业网站
- ubuntu 8.10 on VirtualPC
- Linux代码完全注释(赵炯)
- tar how to
- tar用法实例
- 本站FTP上的SAM926x Linux资源(地址: ftp://www.mcuzone.com 用户名: mcuzone 密码: mcuzone 端口: 21)
- U-boot usage
- 为9263编译Linux
- MCUzone提供的原创Application Note
- Linux 下串口编程入门
- Serial Programming HOWTO, 经典的Linux下串口编程教程

参考书籍

- 嵌入式Linux系统开发技术详解--基于ARM
- 嵌入式Linux应用程序开发详解
- 嵌入式Linux应用开发完全手册

上一篇 Nandflash与SAMBA GUI的兼容问题

下一篇 AT91sam9260ek修改nandflash大小调试笔记