Linux 2.6 内核移植(for AT91rm9200)文档

一.U-boot 编译及改动

1. 获得 U-boot
2. 修改 u-boot-1.0.0 部分源代码
3. 修改 U-boot 传递给内核的参数
4. 编译 U-boot

二.获得 linux kernel 源码及相应硬件的 patch

1.下载 linux 2.6.12 内核源代码

2.下载 for at91rm9200 patch

三.编译内核

1. 交叉编译环境的建立
2. 配置内核的详细步骤
3. 修改网卡驱动

四.更新 Ramdisk 及工具集(busybox)

1. 关于 Ramdisk 的介绍
2. 制作 Ramdisk 方法一
3. 制作 Ramdisk 方法二
4. 使用新版本的 busybox

五. 一些有用的网站

一. U-boot 编译及改动

1. 获得 U-boot

U-boot1.0.0，可以从 http://sourceforge.net/projects/u-boot 网站下载。推荐使用更高一级的版本。

将下载到的 u-boot-1.0.0.tar.bz2 在 linux 环境下解压缩。

```
# tar jxvf u-boot-1.0.0.tar.bz2
```

2. 修改 u-boot-1.0.0 部分源代码

a. 更改网口驱动程序。

原版 u-boot 的网口 I/O 配置不符合自产板要求，更换文件

/u-boot/cpu/at91rm9200/at91rm9200_ether.c

可根据网口芯片选型情况，采用 at91rm9200_ether(dvcom).c 或 at91rm9200_ether（lxt971）.c 进行替换。详情见文件内注释。

b. 添加 intel flash 驱动及命令

将 #/u-boot/common/cmd_mem.c 用 cmd_mem(intel).c 文件进行替换。使其具备对 intel flash 的通过”fl”命令，写及删除能力。

若使用 mem29lv160 芯片，可将

/u-boot/board/at91rm9200dk/flash.c

文件用 flash(fuji).c 文件替换。此时用 u-boot 本身具备的命令(”cp”,”erase”等)进行 flash 操作。

c. 加入 xdownload 命令

在/u-boot/common/ 目录下添加 xmodem.c、cmd_xdownload 文件

将/u-boot/common/Makefile 文件第 50 行修改为：

virtex2.o xilinx.o cmd_xdownload.o xmodem.o

d. 加入 ping 命令

将/u-boot/include/configs/at91rm9200dk.h

用 at91rm9200dk(liqi)进行替换，以配置 mac 地址、ip 地址及系统其他相应参数。

e. 加入 linux 参数传递函数及 linux 引导代码

将/u-boot/common/main.c

用 at91rm9200dk(liqi)进行替换，以加入引导代码

f. 根据本板情况划分内存空间

根据本板属性更改/u-boot/include/configs/at91rm9200dk.h 文件

使用 64MB SDRAM 内存配置参考如下：

```
#define COMMAND_LINE "initrd=0x20800000,32M root=/dev/ram
init=/linuxrc console=ttyS0"
#define CONFIG_BOOTARGS "initrd=0x20800000,64M
root=/dev/ram init=/linuxrc console=ttyS0,115200"

#define BOOT_COMMAND "go 20c00000"
#define INITRD_START 0x20400000
#define INITRD_LEN 0x00400000

#define FLASH_UBOOT_START 0x10020000
#define FLASH_KERNEL_START 0x10040000
#define FLASH_RAMDISK_START 0x10140000

#define FLASH_BASEBOOT_LEN 0x00020000
#define FLASH_UBOOT_LEN 0x00020000
#define FLASH_KERNEL_LEN 0x00100000
#define FLASH_RAMDISK_LEN INITRD_LEN

#define RAM_UBOOT_START 0x21f00000
#define RAM_KERNEL_LOAD_START 0x23000000
#define RAM_RAMDISK_START INITRD_START
```

使用 16MB SDRAM 内存配置参考如下：

```
#define COMMAND_LINE "initrd=0x20400000,8M    root=/dev/ram
init=/linuxrc console=ttyS0"
#define    CONFIG_BOOTARGS              "initrd=0x20400000,8M
root=/dev/ram init=/linuxrc console=ttyS0,115200"

#define BOOT_COMMAND "go 20c00000"
#define INITRD_START  0x20400000
#define INITRD_LEN    0x00400000

#define FLASH_UBOOT_START 0x10020000
#define FLASH_KERNEL_START    0x10040000
#define FLASH_RAMDISK_START  0x10140000

#define FLASH_BASEBOOT_LEN   0x00020000
#define FLASH_UBOOT_LEN      0x00020000
#define FLASH_KERNEL_LEN 0x00100000
#define FLASH_RAMDISK_LEN    INITRD_LEN

#define RAM_UBOOT_START      0x20f00000
#define RAM_KERNEL_LOAD_START    0x20c00000
#define RAM_RAMDISK_START    INITRD_START
```

3. 修改 U-boot 传递给内核的参数

在/u-boot/include/configs/at91rm9200dk.h 28 行中修改 U-boot 传递给内核的参数:

```
#define COMMAND_LINE  “initrd=0x20800000,0x1377c4 root=/dev/ram
init=/linuxrc console=ttyS0,115200”
```

注意:在 2.6 内核中, initrd=后的格式为 “start_address,size” 其中 size 为 Ramdisk.gz 大小,否则在启动后导致 shell 启动不了。

4. 编译 U-boot

make distclean

make at91rm9200dk_config

make

生成 u-boot.bin ,并将其写入 flash 相应地址。

二.获得 linux kernel 源码及相应硬件的 patch

1.下载 linux 2.6.12 内核源代码。

在 www.kernel.org 上下载 linux-2.6.12-rc1.tar.gz

2.下载 for at91rm9200 patch。

在 www.maxim.org.za/AT91ARM9200/2.6/ (附录 1)

下载 2.6.12-rc1-at91.patch.gz

三.编译内核

1.交叉编译环境的建立。

参考 linux-2.6.12-rc1/README 中关于编译环境的说明:

```
****************************************************

               COMPILING  the  kernel:

     Make  sure  you  have  gcc  2.95.3  available.

      Gcc  2.91.66  (egcs-1.1.2),  and  gcc  2.7.2.3  are  known  to
   miscompile some  parts  of  the  kernel,  and  are  *no  longer
   supported*.

       Also  remember  to  upgrade  your  binutils  package  (for
      as/ld/nm  and  company)

****************************************************
```

若使用 cross-3.2.tar.gz 作为交叉编译器,在编译内核时出现如下错误:

```
  UPD     include/linux/version.h
  SYMLINK include/asm -> include/asm-arm
  SPLIT   include/linux/autoconf.h -> include/config/*
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
  MKELF   scripts/mod/elfconfig.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTCC  scripts/mod/modpost.o
  HOSTCC  scripts/mod/sumversion.o
  HOSTLD  scripts/mod/modpost
  HOSTCC  scripts/kallsyms
  HOSTCC  scripts/conmakehash
  SYMLINK include/asm-arm/arch -> include/asm-arm/arch-at91rm9200
  CC      arch/arm/kernel/asm-offsets.s
arch/arm/kernel/asm-offsets.c:38:2: #error Your compiler is too buggy; it is kno
wn to miscompile kernels.
arch/arm/kernel/asm-offsets.c:39:2: #error Known good compilers: 2.95.3, 2.95.4,
 2.96, 3.3
make[1]: *** [arch/arm/kernel/asm-offsets.s] Error 1
make: *** [arch/arm/kernel/asm-offsets.s] Error 2
```

若使用华恒提供版本为 2.95.3 的编译器,在编译内核时会出现如下错误:

```
  CC      init/initramfs.o
  CC      init/calibrate.o
  LD      init/built-in.o
  CHK     usr/initramfs_list
  AS      usr/initramfs_data.o
usr/initramfs_data.S: Assembler messages:
usr/initramfs_data.S:29: Error: Unknown pseudo-op:  `.incbin'
make[1]: *** [usr/initramfs_data.o] Error 1
make: *** [usr] Error 2
```

使用版本号为 3.4.1 的 arm-linux-gcc-3.4.1.tar.bz2 做为编译环境,编译成功。

注意:● 文中所提到的交叉编译器在所附文件中有备份。

● 将交叉编译器(**cross compiler**)解压到适当目录下,相应的交叉编译环境就建成了。

● 相应命令为:(注意适当调整路径)

 # mkdir /cross

 # cd /cross

 # tar jxvf arm-linux-gcc-3.4.1.tar.bz2

此时的交叉编译环境的路径为:

/cross/usr/local/arm/3.4.1/bin/arm-linux-gcc

2. 配置内核的详细步骤。

相应文档参考(在所附文件中有备份):

http://www.arm.linux.org.uk/docs/kerncomp.php

cd /src/linux-2.6.12-rc1

zcat /2.6.12-rc1-at91.patch.gz | patch -p1

注意：这时会打印一些信息,查看是否有错误报告,若有错误,从下几个方面查找:patch 版本是否匹配,patch 与 2.6 源代码是否完整。

vi Makefile

查找到如下:

ARCH ?= $(SUBARCH)

CROSS_COMPILE ?=

修改为:

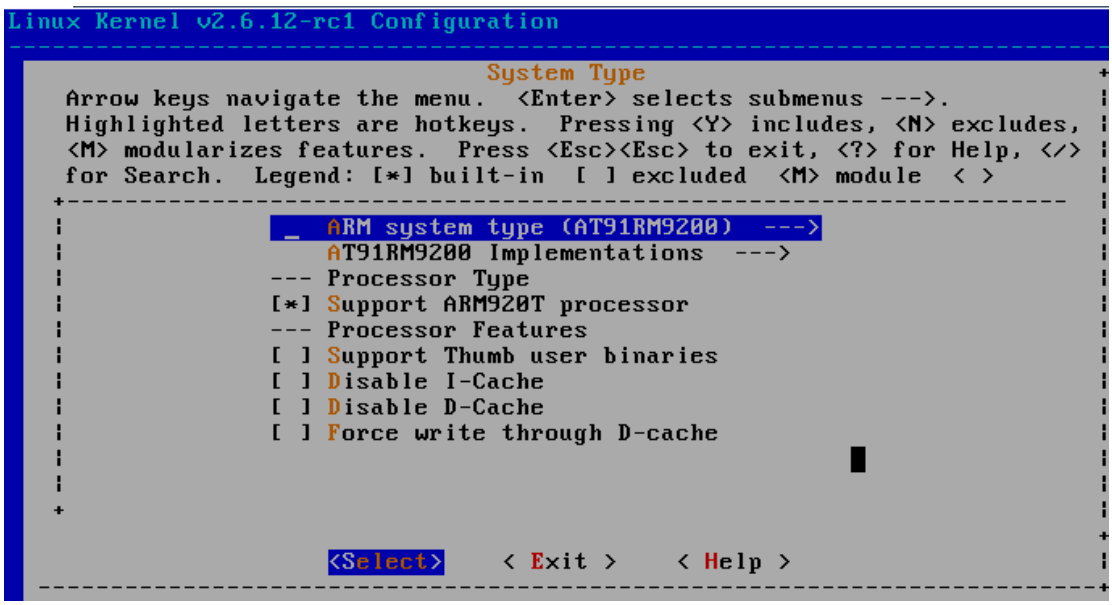
ARCH ?= arm

CROSS_COMPILE ?= 相应交叉编译环境

make at91rm9200dk_defconfig

make menuconfig->system type

显示如下,则配置基本成功。



make menuconfig

进行内核的相应配置。

make

注意：● 在/src/linux-2.6.12-rc1/arch/arm/boot/ 下生成 zImage。

● 编译时间较长,P4 2.8G, 512M 约为 30 分钟(linux 在虚拟机中运行)。

● 将 zImage 写入 flash,启动成功,但网卡 MAC 地址全为 0,网络不通。

3. 修改网卡驱动。

驱动源码在/src/linux-2.6.12-rc1/drivers/net/arm/at91_ether.c

293 行: 在 get_mac_address()中指定 MAC 地址,

static char default_ether_addr [] = {0x00,0x01,0x02,0x03,0x04,0x09};

修改后源代码如下:（原始文件与修改后文件均在所附文件中有备份）

```
**************************************
static void get_mac_address(struct net_device *dev) {
    AT91PS_EMAC regs = (AT91PS_EMAC) dev->base_addr;
    char addr[6];
        /*add by vcom*/
     static char default_ether_addr [] =
{0x00,0x01,0x02,0x03,0x04,0x09};
     /*end add by vcom*/
    unsigned int hi, lo;
    /* Check if bootloader set address in Specific-Address 1 */
    hi = regs->EMAC_SA1H;
    lo = regs->EMAC_SA1L;
    memcpy(dev->dev_addr,&default_zhong_ether_addr,6);
}
**************************************
```

在/src/linux-2.6.12-rc1/include/asm-arm/arch-at91rm9200/pio.h

75 行:修改为如下所示:

```
**************************************
/*
 * Configure Ethernet for MII mode.
 */
/*static inline void AT91_CfgPIO_EMAC_MII(void) {
    AT91_SYS->PIOA_PDR = AT91C_PA16_EMDIO | AT91C_PA15_EMDC |
AT91C_PA14_ERXER | AT91C_PA13_ERX1
        | AT91C_PA12_ERX0 | AT91C_PA11_ECRS_ECRSDV | AT91C_PA10_ETX1
        | AT91C_PA9_ETX0 | AT91C_PA8_ETXEN | AT91C_PA7_ETXCK_EREFCK;
    AT91_SYS->PIOB_PDR = AT91C_PB19_ERXCK | AT91C_PB18_ECOL |
AT91C_PB17_ERXDV
```

```
            | AT91C_PB16_ERX3 | AT91C_PB15_ERX2 | AT91C_PB14_ETXER |
AT91C_PB13_ETX3
            | AT91C_PB12_ETX2;
        AT91_SYS->PIOB_BSR = AT91C_PB19_ERXCK | AT91C_PB18_ECOL |
AT91C_PB17_ERXDV
            | AT91C_PB16_ERX3 | AT91C_PB15_ERX2 | AT91C_PB14_ETXER |
AT91C_PB13_ETX3
            | AT91C_PB12_ETX2;
    }*/

    /*liqi add here*/
    static inline void AT91_CfgPIO_EMAC_MII(void) {
        AT91_SYS->PIOA_PDR |= AT91C_PA16_EMDIO | AT91C_PA15_EMDC |
AT91C_PA14_ERXER | AT91C_PA13_ERX1 | AT91C_PA12_ERX0 |
AT91C_PA11_ECRS_ECRSDV | AT91C_PA7_ETXCK_EREFCK;

        AT91_SYS->PIOD_PDR |= AT91C_PD0_ETX0 | AT91C_PD1_ETX1 |
AT91C_PD4_ETXEN |AT91C_PD2_ETX2 | AT91C_PD3_ETX3 | AT91C_PD5_ETXER ;

        AT91_SYS->PIOB_PDR |= AT91C_PB25_EF100 | AT91C_PB19_ERXCK |
AT91C_PB18_ECOL | AT91C_PB17_ERXDV | AT91C_PB16_ERX3 | AT91C_PB15_ERX2;

        AT91_SYS->PIOD_BSR |= AT91C_PD2_ETX2 | AT91C_PD3_ETX3 |
AT91C_PD5_ETXER ;

        AT91_SYS->PIOB_BSR |= AT91C_PB25_EF100 | AT91C_PB19_ERXCK |
AT91C_PB18_ECOL | AT91C_PB17_ERXDV | AT91C_PB16_ERX3 | AT91C_PB15_ERX2 ;
    }
    /*liqi add end*/
    ************************************
```

修改后MII接口的网络也可正常使用。此时,该驱动可以支持MII接口与RMII接口网络，(在配置内核时选择适当的网络设备配置) 。重新编译内核,将内核写入flash,重启板子,网络可用。

四.更新Ramdisk及工具集(busybox)

内核启动正常后,若使用2.4版本的ramdisk(busybox-1.00-pre10),在使用shell命令时会出现如下错误或告警:

```
~ #
~ # lsmod
Module                  Size  Used by
lsmod: QM_MODULES: Function not implemented

~ #
```

或出现:命令版本与内核版本不匹配等,此时需更新 ramdisk。

1. 关于 RAMDISK 的介绍。

把 ramdisk.image.gz 解压后 mount -o loop 到一个目录上(mount -o loop ramdisk.img /mnt)，这样就可以看到 ramdisk 里面的文件系统内容，这时再把你的新编译的 busybox 的可执行文件复制到这个目录的 bin 目录下面覆盖原来的文件，重新建立链接，umount 这个目录，再 gzip 压缩，这样你所作的改动就被带到这个新生成的 ramdisk.image.gz 文件里面了，然后你烧写这个文件就可以看到新的文件系统了。

简单的步骤:

```
# gunzip ramdisk.image.gz
# mount -o loop ramdisk.image /mnt
# cp -f busybox /mnt/bin
# umount /mnt
# gzip ramdisk.image
```

注意:要自己调整目录路径

2.制作 Ramdisk 方法一。

创建一个简单的基于 Ext2fs 的 Ramdisk

```
# mke2fs -vm0 /dev/ram 4096
# mount -t ext2 /dev/ram /mnt
# cd /mnt
# cp /bin, /sbin, /etc, /dev ... files in mnt
# cd ../
# umount /mnt
# dd if=/dev/ram bs=1k count=4096 of=ramdisk.img
```

说明:

● mke2fs 是用于在任何设备上创建 ext2 文件系统的实用程序 — 它创建超级块、索引节点以及索引节点表等等。

●在上面的用法中，/dev/ram 是上面构建有 4096 个块的 ext2 文件系统的设备。然后，将这个设备（/dev/ram）挂装在名为 /mnt 的临时目录上并且复制所有必需的文件。一旦复制完这些文件，就卸装这个文件系统并且设备（/dev/ram）的内容被转储到一个文件（ext2ramdisk）中，它就是所需的 Ramdisk（Ext2 文件系统）。

●上面的顺序创建了一个 4 MB 的 Ramdisk，并用必需的文件实用程序来填充它。 一些要包含在 Ramdisk 中的重要目录是：

/bin — 保存大多数像 init、busybox、shell、文件管理实用程序等二进制文件。

/dev — 包含用在设备中的所有设备节点

/etc — 包含系统的所有配置文件

/lib — 包含所有必需的库，如 libc、libdl 等

3.制作 ramdisk 的方法二。

（1）建立 loop 设备的临时挂接点和一个大小为 6M(大小可以调节)的临时文件，并将其清零：

```
# mkdir /mnt/loop 2>/dev/null
# dd if=/dev/zero of=/tmp/loop_tmp bs=1k count=6144 >/dev/null
```

（2）将 loop 设备与临时文件联系起来

```
# losetup /dev/loop0 /tmp/loop_tmp
```

（3）Linux 内核识别两种可以直接拷贝到 RAMDISK 的文件系统，它们是 minix 和 ext2，ext2 性能更好：

```
# mke2fs -m 0 /dev/loop0 2>/dev/null
```

mke2fs 将会自动判断设备容量的大小并相应地配置自身， -m 0 参数防止它给 root 保留空，这样会腾出更多地有用空间。

（4）接着把虚拟盘挂在节点/mnt 上：

```
# mount /dev/loop0 /mnt/loop -t ext2
```

（5）将制作好的 root 文件系统拷贝到所挂的节点上，卸下挂接点，删除建

立的挂接点。

```
# cp -a /ramdisk/* /mnt/loop
# umount /mnt/loop
```

现在制作的/tmp/loop_tmp 就是一个 6M 的 ramdisk 文件

4.使用新版本的 busybox。

（1）在 busybox.net 上下载新版本的 busybox-1.00.tar.gz

（2）解压,make menuconfig->Build Options 选定

[*]Do you want to Build Busybox with a Cross Compiler?

并修改交叉编译环境

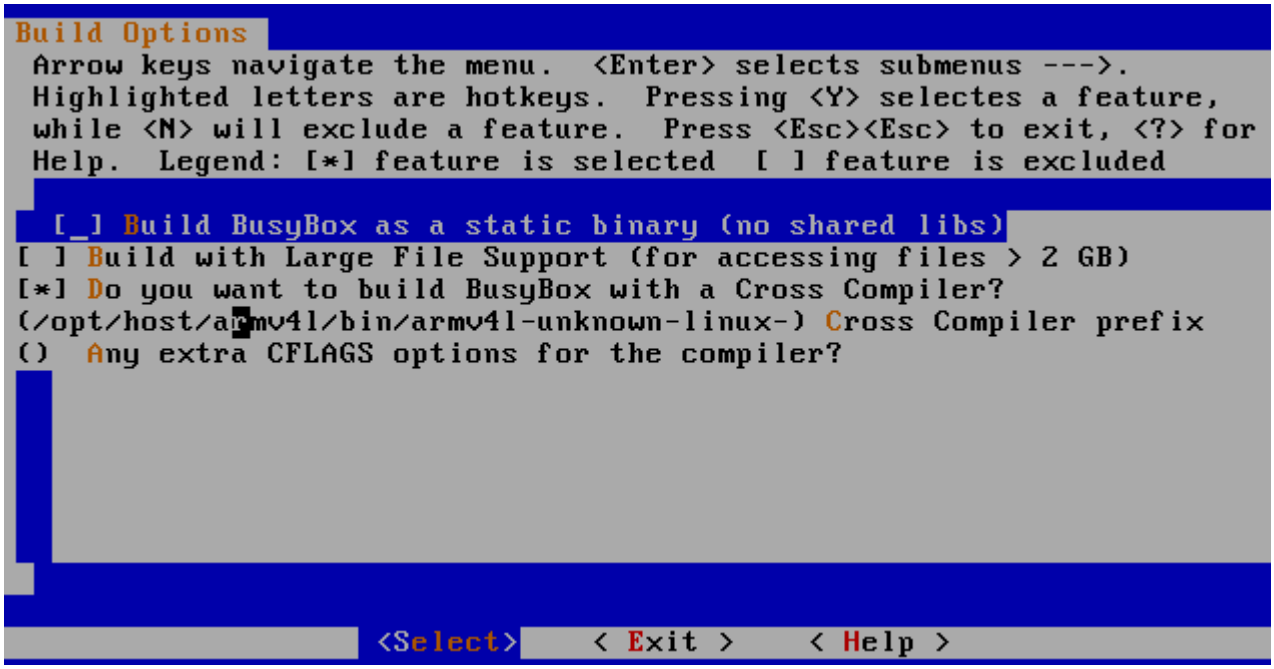
注意：
- 新版的 busybox 使用的编译器为附带的 2.95.2 的编译器。
- make menuconfig->Coreutils->[]usleep
  取消选择,否则编译会出错,跳过 usleep。
- 修改 src/busybox/shell/ash.c 中约 6479 行：
  /* ofd=fd=open(_PATCH_TTY,O_RDWD);*/
  修改为：
  ofd=fd=open(“/dev/ttyS0”,O_RDWD);
  否则可能会出现不能使用 Ctrl+C 的情况
- 调试 busybox 时可采用逐步增删的方法,最终形成所需的 busybox。

（3）make

生成 busybox 及 busybox.links

替换 Ramdisk 中的 busybox,并重新建立链接,例如：

```
# cd /bin
# ln -s /bin/busybox insmod
```



五.一些有用的网站

- www.kernel.org

  Linux 内核源码下载，内核开发相关新闻。

- www.arm.linux.org.uk

  arm linux 开发者一定要去的地方，arm linux 开发的最新进展，新的驱动，还有相应的 patch 下载,还可在 mail list 中找到你所需要的东西。

- www.linuxforum.net

  关于 linux 开发的论坛，很多最新的资料在上面。

- www.busybox.net

  busybox 源码下载，busybox 相关新闻，mail list。

注意：宿主机开发环境为 RedHat 9.0