		文档级别		总页数	
文档名称		编制人		所在部门	
		审核人		编制日期	

基于 OMAP3530 的 linux 内核移植

目　　录

1. 概述

目前开始对 OMAP3530 进行研究，需要使用到操作系统，综合各方面，目前暂时考虑使用 linux 操作系统。为了便于用户使用，TI 官方提供了 linuxPSP（Platform Support Package）和 DVSDK（Digital Video Software Development Kit），内有 linux-2.6 源代码和大量相关驱动程序，不过这些中很多都是基于 OMAP35x DVEVM 板（以下简称 EVM 板）的。本公司将来如果在产品中使用 OMAP3530，其电路原理不可能与 EVM 板完全一致，并且现在还没有 EVM 板的处理器板和电源板的电路原理图，因此需要考虑 linux 内核的移植问题。本文主要讨论 linux 内核的移植过程以及出现的问题。

内核的移植一般都是在一个已经比较完善的版本的基础上进行改动得到的，本文针对的是 linux-2.6.29-rc3，从内核的角度上讲改动量最大的可能是由于硬件不同造成的软件驱动程序上的差别，关于一些应用相关的改动这里不考虑。因此本文中内核的移植主要包括以下两方面：

1) 修改或增加一些配置文件，以便在编译时可以专门针对本公司自己研制的目标板进行；
2) 修改或增加删减一些驱动程序，主要是针对本公司自己研制的目标板与 EVM 板的差别进行。

2. EVM 板简要说明

这里首先简要说明 EVM 板，这个板是由 MISTRAL 公司设计生产的，由于没有处理器板和电源板电路原理图，因此只能大概做一个说明。EVM 板由几个模块组成，包括处理器板、电源板、多媒体子板和主板等（最近好像又推出了一个 WIFI+蓝牙模块）。其中：

1) 处理器板主要包括处理器 OMAP3530CBB 和 POP 封装的 MCP 内存即 NAND+LPDDR 以及一些接口电路；
2) 电源板主要包括 PMU 芯片 TPS65950 及一些外围相关电路；
3) 多媒体子板主要包括视频解码芯片 TVP5146、USB 接口芯片 USB3322 及一些外围相关电路；
4) 主板主要包括 OMAP3530 的外设部分接口电路，包括 LCD 接口、触摸屏接口、串口、键盘、网口、MMC 卡插座、USB 插座、音视频插座、

电源输入插座等。

下图是 EVM 板组成框图。

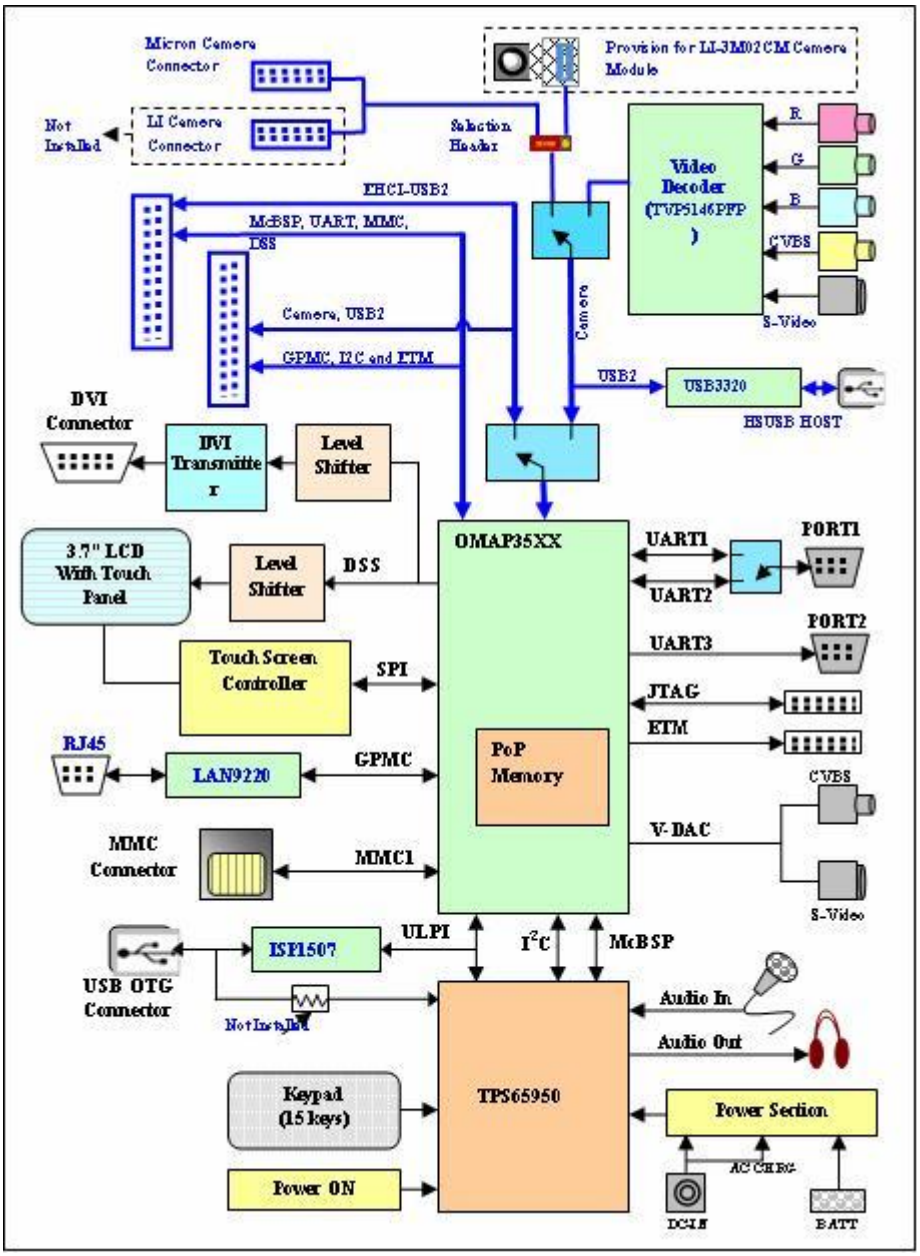


图 1 EVM 板组成框图

下图是处理器板实物图。

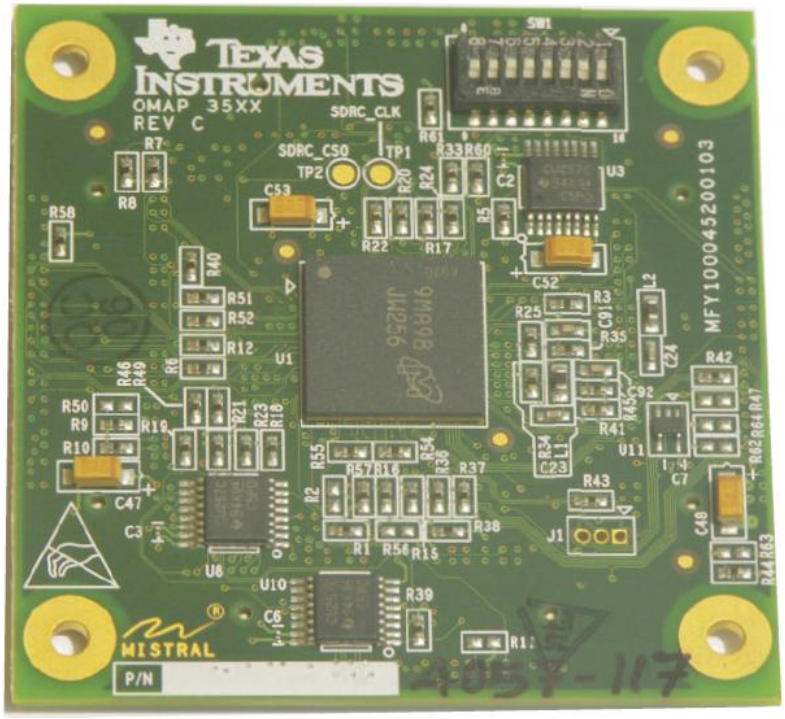


图 2 处理器板实物图

下图为电源板实物图。

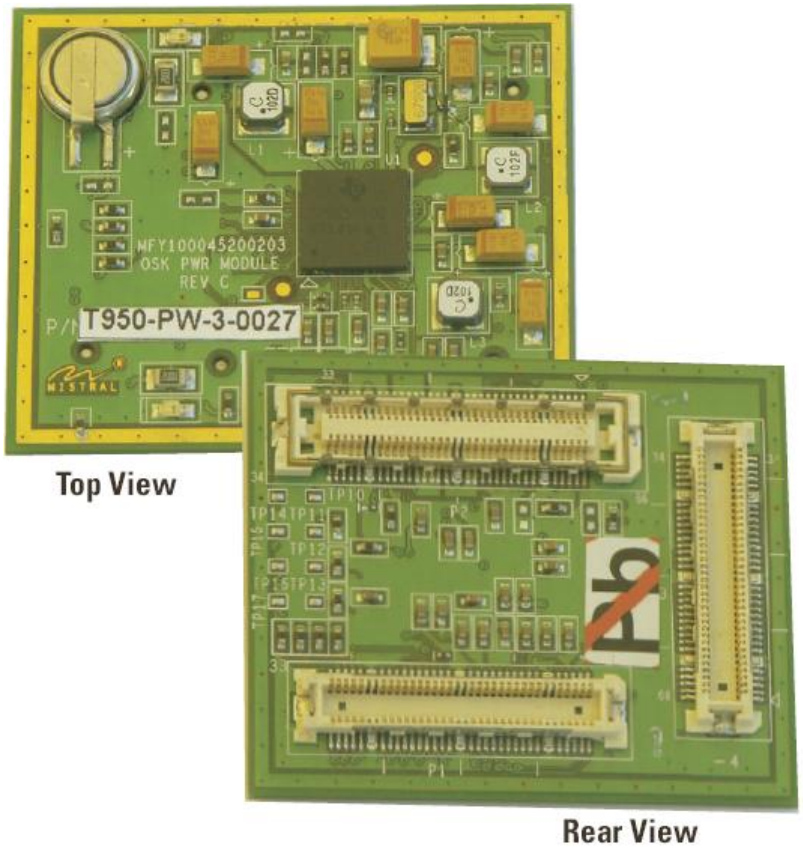


图 3 电源板实物图

下图为多媒体子板实物图。

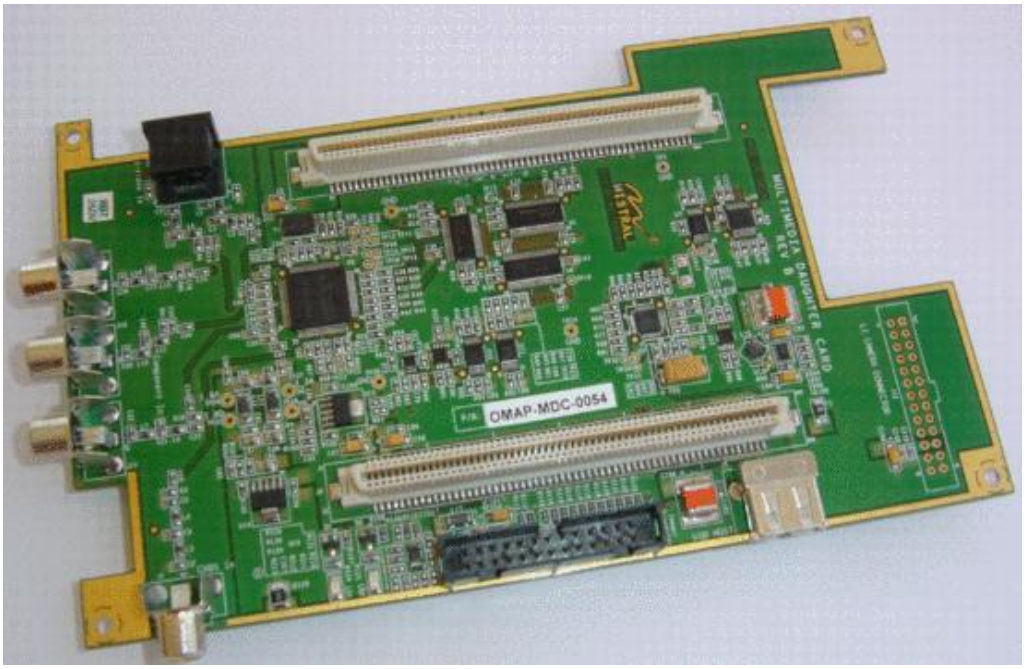


图 4 多媒体子板实物图

下图为主板实物图。

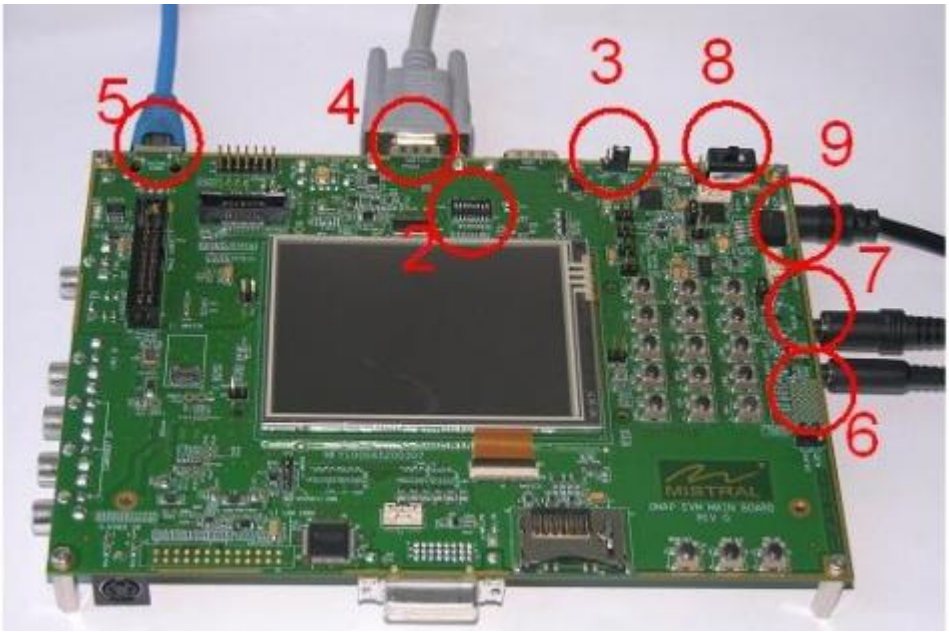


图 5 主板实物图正面

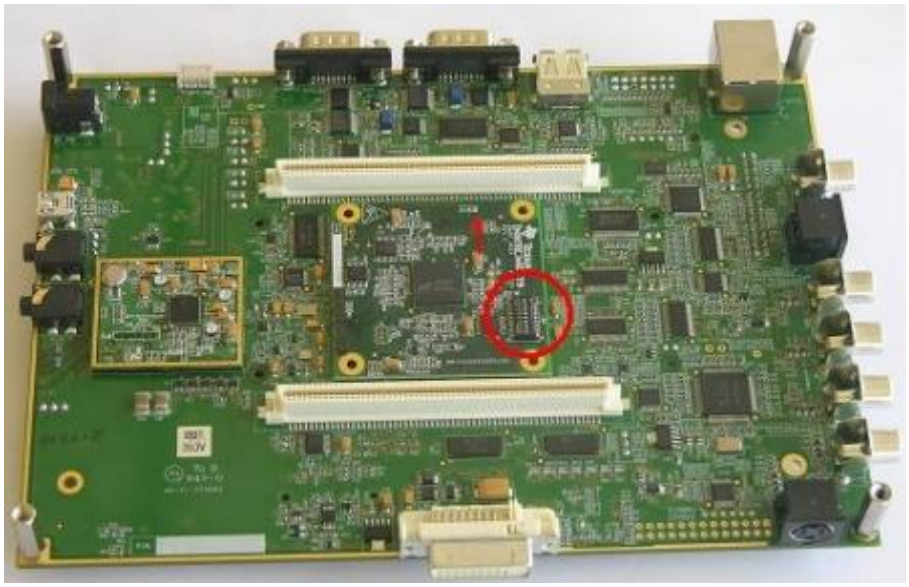


图 6 主板实物图背面

从主板背面可以看到处理器板和电源板。

2.1 可能的区别

下面考虑本公司可能的产品会与 EVM 有怎样的区别。

2.1.1 处理器

首先处理器的封装会不同，EVM 使用的是 OMAP3530CBB，它是 POP 封装；而本公司如果使用的话会是 OMAP3530CUS，它的管脚数少于 CBB 封装芯片，因此有部分功能管脚无法使用，详细见 OMAP3530TRM（SPRUF98D）的表 1-1。参考表 1-1 可以知道 CUS 封装的芯片比 CBB 封装而言主要是减少了一些外设功能管脚，因此只要避免使用这些外设功能管脚，对软件而言二者就没有什么区别了。从 EVM 的使用来看并没有使用这些管脚，因此这方面的问题不大。

2.1.2 存储器

EVM 的处理器是 OMAP3530CBB，它是 POP 封装，因此它的存储器部分（NAND+LPDDR）是安装在处理器上的，而本公司使用的 OMAP3530CUS 不是 POP 封装，因此它的存储器要放在印制板上，不过这些区别主要是对硬件原理图和印制板图影响，对软件基本没有影响。不过 NAND 和 LPDDR 的容量大小会有区别，特别是 LPDDR 的容量大小改变可能对 u-boot 和 linux 内核都有影响。

2.1.3 电源管理芯片

电源管理芯片即 PMU，EVM 使用的是 TPS65950，而本公司如果使用的话

应该是 TPS65930(或 TPS65920),二者区别除了封装外(TPS65950 的间距 0.4mm,太小了,而 TPS65930 为 0.65mm),在使用上来讲主要是 CODEC 部分的差别(另外 TPS65950 还有充电电路而 TPS65930 没有,不过这里暂时不考虑充电电路问题),TPS65930 的 CODEC 部分的外部接口比较少(TPS65920 没有 CODEC 部分),在实际使用时可能满足不了使用要求,需要外加一个 CODEC 芯片,而 TPS65930 的 CODEC 部分实际上就不用了。因此 PMU 上的差别主要就变成了 CODEC 部分的差别(在键盘数量、GPIO 等还有一些差别),在软件上就需要进行修改,这部分的改动可能会比较大一些。

2.1.4 外设

这部分改动可能是最大的,因为本公司如果使用的话 GPIO 会很多,可能要多达 40～50 个,而 EVM 板上 GPIO 使用数量相对较少,因此这部分的改动很大。首先 OMAP3530 要配置出大量 GPIO 管脚,因此管脚复用上就需要改动,同时因为配置出这么多 GPIO,其它外设就受到很大影响(因为所有 GPIO 管脚都是复用的),部分外设就没有办法使用,因此需要确定哪些外设可能用到,哪些外设一定不会用到,对整个系统需求做一个完善的规划,这部分考虑在下一节讨论。

3. 系统需求

因工作暂停,略。

4. CETEK-OMAP3530-Mini 板移植说明

由于本公司的 OMAP3530 开发板还在研究中,目前 OMAP3530 实物开发板只有一块 ICETEK-OMAP3530-Mini 板(以下简称 Mini 板),因此考虑首先将 linux 内核移植到这块板上。

4.1 Mini 板简要说明

Mini 板是国内的瑞泰创新公司设计生产的,它基本是仿制国外的一款 beagelboard,当然它使用的处理器也是 OMAP3530CUS(beagleboard 是 OMAP3530CBB),PMU 也是 TPS65930(beagleboard 是 TPS65950),因此与 beagleboard 的软件上的区别也主要是 CODEC 部分,这两款开发板的外设接口都很少,有很多功能无法使用。不过进行内核移植还是足够的,实际上 linux 内对 beagleboard 的支持是比较好的,因此对 Mini 板的移植相对也比较容易。

下图分别为 Mini 板和 beagleboard 的实物图。

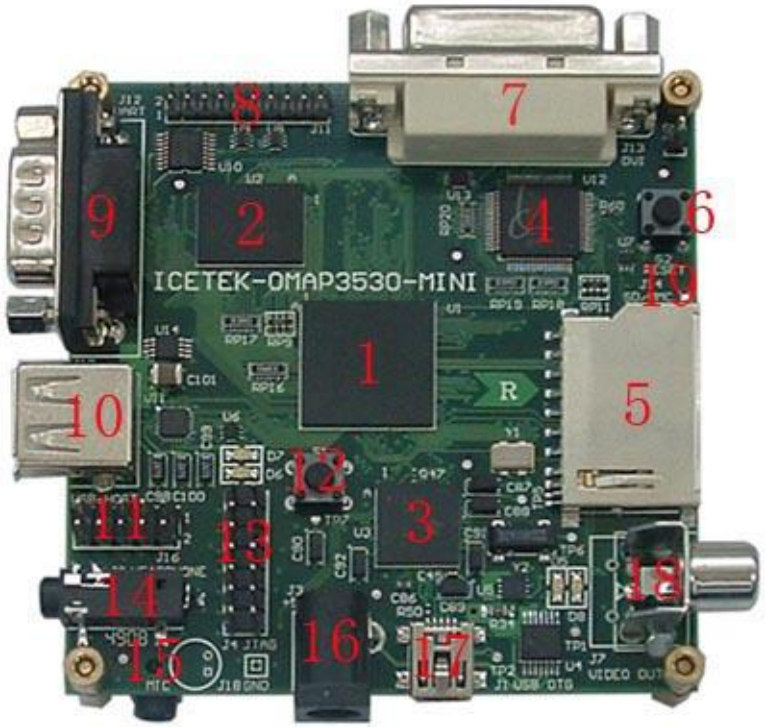


图 8 Mini 板实物图

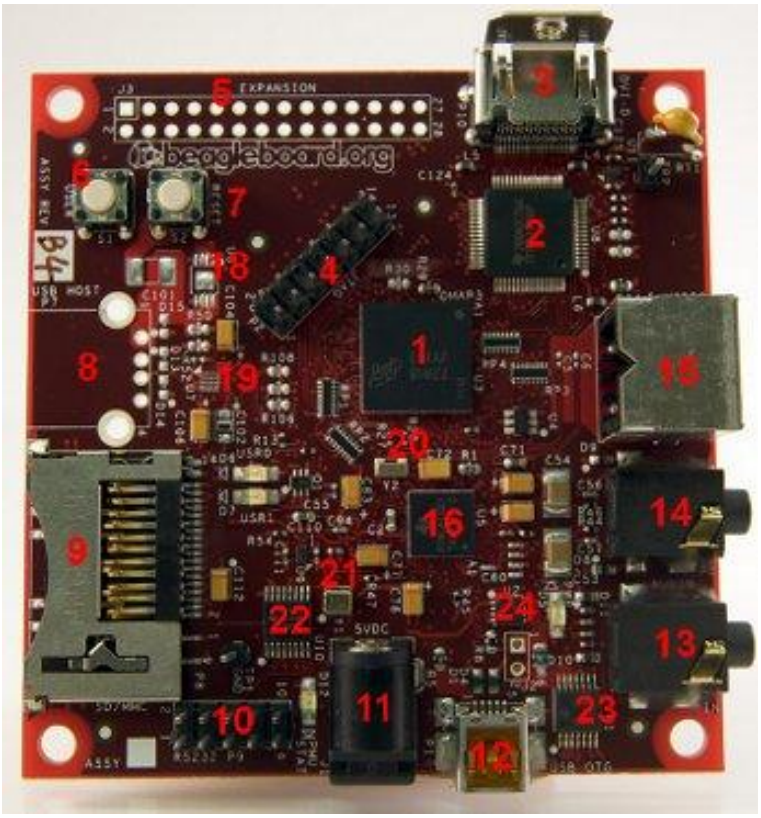


图 9 beagleboard 实物图

4.2 Mini 板 linux 内核移植

下面考虑对 Mini 板进行移植，主要就是参照 beagleboard 相关的地方进行修改。

以下的文件修改和编译是工作在计算机 linux 环境下，本文中 linux 版本为 fedora6，并都是以 root 工作的。

4.2.1 软件安装

首先需要安装 linuxPSP，步骤可以参考网上一篇文章《OMAP35x DVEVM Software Setup》，文章对 PSP 和 DVSDK 的安装有比较详细的说明，并且这些安装都是图形界面方式的，这里不再叙述。本文的 linuxPSP 版本是 02.01.03.11，DVSDK 版本是 3_00_02_44。在 PSP 文件夹内可以得到内核源代码压缩文件 linux-02.01.03.11.tar.gz，解压缩文件就得到源代码，本文将文件解压缩在目录/usr/local/src/linux-02.01.03.11（以下用/linux 表示内核源代码文件目录）内。

4.2.2 下载编译工具链

下载编译的工具链，同样可以参考 OMAP35x DVEVM Software Setup，本文下载的是 arm-2008q1-126-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2。下载后解压缩，本文将文件解压缩在目录/opt/codesourcery/arm-2008q1。这个工具链是在命令行下工作的，是免费的，图形界面的编译工具要收费。

4.2.3 mkimage 文件

在编译内核时需要 u-boot 内的 mkimage 文件将内核映像文件加一个文件头，以便在 u-boot 下加载内核映像文件时 u-boot 可以识别。这个文件需要编译 u-boot 后生成，本文不讨论 u-boot 的移植问题，因此首先使用 Mini 板以前的 u-boot，如果出现问题再讨论。本文的 mkimage 所在目录是/usr/local/src/omap3530Mini。

4.2.4 修改配置文件

首先需要使内核在配置时支持 Mini 板，因此需要修改一些配置文件以便在编译时能识别 Mini 板。

4.2.4.1 mach-types 文件

这个文件在/linux/arch/arm/tools 目录下，从文件名可以看出是用于指示机器类型（即目标板类型），需要手动改动这个文件增加内核对 Mini 板的支持，可以

看到目前版本中已经支持到 2006 个目标板类型（可能中间少几个），在文件第 1997 行加入：

```
omap3_mini        MACH_OMAP3_MINI        OMAP3_MINI        2015
```

2015 这个数字是为了与 Mini 板以前版本保持一致，并且在 u-boot 中也使用了这个数字，因此可以用 Mini 板以前的 u-boot 而不会出现问题。不过因为没有提供给 linux 官网，这个数字可能在不久后在正式的 linux 版本中就会出现，因此这个数字其实可以改大一些，比如 4000 或 5000，这样至少在几年内不会冲突，便于升级。不过这样还要改 u-boot，因此暂时先使用这个数字。

在编译过程中，编译程序会根据这个文件在/linux/include/asm-arm/目录下生成 mach-types.h 文件，并根据这个头文件确定需要编译的目标板类型。

4.2.4.2 omap3_mini_defconfig 文件

这是一个目标板默认配置文件，文件内有大量配置选项，在编译时首先需要编译默认配置文件，在/linux 目录下生成一个.config 文件，还可以通过 make menuconfig 来进行手动配置。在编译内核时根据.config 文件在/linux/include/linux 目录下生成 autoconf.h 文件，并根据这个头文件确定需要编译的部分。每一块目标板都需要这样一个配置文件，目前版本的内核源代码里没有提供 Mini 板的配置文件，因此需要手动加一个，Mini 板与 beagleboard 基本一致，因此可以将 beagleboard 的配置文件改一个名稍加修改就可使用。在/linux/arch/arm/configs 目录下将 omap3_beagle_defconfig 拷贝到其它目录并改名为 omap3_mini_defconfig。然后再拷贝回/linux/arch/arm/configs 目录。

下面修改这个文件，这个文件是执行 make config 然后手动逐项配置的，需要手动改写。

1) 将文件第 213 行用#号注释掉改成（其实这一行也可以不改，不过这样做的话将来的可移植性可能就差一些，因为后续的 beagleboard 可能不一定和现在完全一致）

```
# CONFIG_MACH_OMAP3_BEAGLE is not set
```

2) 在文件第 214 行回车空出一行，键入

```
CONFIG_MACH_OMAP3_MINI=y
```

这样在编译时在 autoconf.h 文件中宏 CONFIG_MACH_OMAP3_MINI=1，将

来在编写只与 Mini 板相关的程序时就可以以这个宏标记。

4.2.4.3 Kconfig 文件

在/linux 源代码的大多数目录中都可以看到包含一个 Kconfig 文件，这个文件在使用 make config 进行配置时提供各种配置选项，如果默认配置后不需要改动，这些文件可以不动（实际上 XXX_defconfig 默认配置文件最开始也是由 make config 得到的）。不过为了能够了解它怎样工作，可以尝试改动一下。经过查找共有 3 个 Kconfig 文件内包含与 beagleboard 有关的选项，如下：

1) /linux/arch/arm/mach-omap2/Kconfig；
2) /linux/drivers/i2c/busses/Kconfig；
3) /linux/sound/soc/omap/Kconfig。

下面分别修改这 3 个文件。

1) /linux/arch/arm/mach-omap2/Kconfig：将第 156 行～158 行的内容拷贝并粘贴到第 160 行并回车，修改粘贴的内容如下：

```
config MACH_OMAP3_MINI
	bool "OMAP3 Mini board"
	depends on ARCH_OMAP3 && ARCH_OMAP34XX
```

2) /linux/drivers/i2c/busses/Kconfig：将第 436 行～439 行的内容拷贝并粘贴到第 441 行并回车，修改粘贴的内容如下：

```
config I2C2_OMAP_MINI
	bool "Enable I2C2 for OMAP3 MiniBoard"
	depends on ARCH_OMAP && MACH_OMAP3_MINI
	select OMAP_MUX
	default n
	help
	  Say Y here if you want to enable I2C bus 2 at OMAP3 based
	  MiniBoard.
	  I2C2 at MiniBoard is connected to expansion connector, i.e. unused
	  if nothing is connected to this connector. As internal OMAP3 pull up
	  resistors are not strong enough, enabled but unused I2C2 bus results
	  in error messages (e.g. I2C timeouts). Enable this only if you have
	  something connected to I2C2 at board's expansion connector and this
	  extension has additional pull up resistors for I2C2 bus.
```

3) /linux/sound/soc/omap/Kconfig：将第 67 行～73 行的内容拷贝并粘贴到第 75 行，修改粘贴的内容如下：

```
config SND_OMAP_SOC_OMAP3_MINI
```

```
	tristate "SoC Audio support for OMAP3 Mini"
	depends on TWL4030_CORE && SND_OMAP_SOC && MACH_OMAP3_MINI
	select SND_OMAP_SOC_MCBSP
	select SND_SOC_TWL4030
	help
	  Say Y if you want to add support for SoC audio on the Miniboard.
```

4.2.4.4 编译配置文件

经过上述改动，配置文件修改基本完成，可以尝试编译配置文件。打开一个终端，进入 linux 源代码目录，注意在编译时首先需要建立编译工具链的路径，键入

```
export PATH=/opt/codesourcery/arm-2008q1/bin:/usr/local/src/omap3530Mini:$PATH
```

其中后面一个路径是给 mkimage 设置的，也可以将上述命令存入一个文件，使用命令 source 执行，这样方便些。

为了编译时减少输入量，稍微修改一下/linux 根目录下的 Makefile 文件，将其第 199 行的

```
CROSS_COMPILE	?= arm-linux-
```

更改为（或者注释掉再重写一行）：

```
CROSS_COMPILE	?= arm-none-linux-gnueabi-
```

然后可以编译配置文件，

```
make distclean
make omap3_mini_defconfig
```

这时可以看到在/linux 目录下生成一个.config 文件，然后可以手动配置，

```
make menuconfig
```

这时可以在系统菜单中看到 Mini 板已被配置上。

4.2.5 修改源代码文件和 Makefile

下面修改源代码文件和 Makefile 文件，首先查找出/linux 目录内所有的与 beagleboard 相关的源代码文件和 Makefile 文件，如下：

1) /linux/arch/arm/plat-omap/include/mach/board-omap3beagle.h；
2) /linux/arch/arm/plat-omap/include/mach/hardware.h；
3) /linux/arch/arm/mach-omap2/board-omap3beagle.c；
4) /linux/drivers/video/omap/lcd_omap3beagle.c；
5) /linux/sound/soc/omap/omap3beagle.c；

6) /linux/arch/arm/mach-omap2/Makefile；

7) /linux/drivers/video/omap/Makefile。

上述有的文件需要修改，有的文件需要重新命名后再修改，下面逐个说明。

4.2.5.1 board-omap3mini.h 文件

将 board-omap3beagle.h 文件拷贝到其它目录，更名为 board-omap3mini.h，再剪切回原目录。下面修改文件内容，比较简单，将文件中出现的 beagle、BEAGLE 或 Beagle 分别用 mini、MINI 或 Mini 替代。

4.2.5.2 hardware.h 文件

这个文件需要改动，拷贝第 346～348 行，在第 350 行粘贴并回车，将这几行出现的 beagle、BEAGLE 或 Beagle 分别用 mini、MINI 或 Mini 替代。

4.2.5.3 board-omap3mini.c 文件

方法同 4.2.5.1，这里不再叙述。

4.2.5.4 lcd_omap3mini.c 文件

方法同 4.2.5.1，这里不再叙述。

4.2.5.5 omap3mini.c 文件

方法同 4.2.5.1，这里不再叙述。

4.2.5.6 Makefile 文件

这个文件需要改动，拷贝第 66～69 行，在第 70 行粘贴并回车，将这几行出现的 beagle、BEAGLE 或 Beagle 分别用 mini、MINI 或 Mini 替代。

4.2.5.7 Makefile 文件

这个文件需要改动，拷贝第 35 行，在第 36 行粘贴并回车，将这行出现的 beagle、BEAGLE 或 Beagle 分别用 mini、MINI 或 Mini 替代。

这样源代码和 Makefile 文件修改完毕，实际上上述配置文件和源文件的改动只是将 beagleboard 重新复制了一遍并改了个名，关键的部分没有做任何改动。

4.2.6 编译目标文件

下面可以编译目标文件，开一个终端，进入 linux 源代码目录，设置好路径，

```
make distclean
make omap3_mini_defconfig
```

make uImage

等待一段时间后，很不幸，报错了，编译没有通过，后使用 beagleboard 的配置进行编译，报的错相同，看来是这个版本专门针对 EVM 板编写了一些程序，但是移植性做的不太好。

4.2.7 修改源代码文件

在上一节中的编译出现了四个错误（这个显示的是针对 beagleboard 编译时出现的错误），如下：

```
  LD      .tmp_vmlinux1
arch/arm/mach-omap2/built-in.o: In function `omap3_beagle_map_io':
/usr/local/src/workspace-omap3/linux-02.01.03.11/arch/arm/mach-omap2/board-omap3beagle.c:450: und
efined reference to `omap2_set_sdram_vram'
arch/arm/mach-omap2/built-in.o: In function `usb_musb_init':
/usr/local/src/workspace-omap3/linux-02.01.03.11/arch/arm/mach-omap2/usb-musb.c:168: undefined re
ference to `get_omap3evm_board_rev'
drivers/built-in.o: In function `twl4030_probe':
/usr/local/src/workspace-omap3/linux-02.01.03.11/drivers/mfd/twl4030-core.c:806: undefined refere
nce to `usb_gpio_settings'
drivers/built-in.o: In function `musb_platform_init':
/usr/local/src/workspace-omap3/linux-02.01.03.11/drivers/usb/musb/omap2430.c:259: undefined refer
ence to `get_omap3evm_board_rev'
make: *** [.tmp_vmlinux1] Error 1
```

图 10 编译错误截图

下面逐个查找问题。

4.2.7.1 board-omap3beagle.c 文件

这个文件出现的问题是 omap2_set_sdram_vram 函数没有定义，这个函数原型定义在/linux/arch/arm/plat-omap/fb-vram.c 内，需定义宏 CONFIG_FB_OMAP2 或 CONFIG_FB_OMAP2_MODULE，beagleboard 都没定义，因此这个函数实际上对 beagleboard 是无效的，因此这句话应注释掉。注释掉这句后再编译就可以看到少了一个错误。

对于文件 board-omap3mini.c 也同样注释掉这句话。

4.2.7.2 usb-musb.c 文件

这个文件出现的问题是 get_omap3evm_board_rev 函数没有定义，这个函数很明显是针对 EVM 板的，对其它目标板没有意义，对于源文件的修改是在原第 167 行前加一句

#ifdef CONFIG_MACH_OMAP3EVM

在原 170 行加上一句

#endif

这样只对 EVM 板才编译这一段程序。

4.2.7.3 twl4030-core.c 文件

这个文件出现的问题是 usb_gpio_settings 函数没有定义，这个函数原型定义在/linux/arch/arm/mach-omap2/board-omap3evm.c 内，很明显是针对 EVM 板的，处理方法同上，在 usb_gpio_settings 函数前后加上同样的语句。

4.2.7.4 omap2430.c 文件

这个问题同 4.2.7.2 的问题相同，处理方法也相同，在文件的原第 257 行和第 264 行加上同样的语句。

改完这四个文件后再编译时可以通过，在/linux/arch/arm/boot/目录下会生成一个 uImage 文件，这个就是内核映像文件。

4.2.8 加载映像文件

下面可以拿 Mini 板实物做一下实验，看新版本的内核是否能够正常工作。加载的方法参见 Mini 板使用手册，将新编译好的内核覆盖 MMC 卡内原来的内核映像文件，其余不动，然后将 MMC 卡插入 Mini 板的 MMC 卡槽，加电，通过串口可以看到内核可以运行，不过似乎缺少模块文件，报了不少错误，这些还需要再进一步研究。

4.2.9 u-boot 问题

上述的加载实验是以 beagleboard 配置编译的，在使用 Mini 板配置编译后生成的映像文件加载失败，调试信息显示目标板的 ID 号不一致，在 u-boot 中的 ID 号是 1546，正好是 beagleboard 对应的 ID 号，这也是为什么在上一节中使用 beagleboard 的配置时可以运行。因此还要检查 u-boot 的问题，以便 Mini 板配置时能够运行。本文的 u-boot 位于/usr/local/src/omap3530Mini/u-boot 目录下，以下就用/u-boot 表示目录。

查找/u-boot/board/omap3530Mini/目录下的 omap3530Mini.c 文件，在第 73 行可以看到：

```
gd->bd->bi_arch_number = MACH_TYPE_OMAP3_BEAGLE; /* board id for Linux */
```

这个值就是传递给 linux 内核的目标板 ID 号，因此原来的 u-boot 中 Mini 板的 ID 号 2015 是假的，实际没用上，将这句话更改为：

```
gd->bd->bi_arch_number = MACH_TYPE_OMAP3_MINI; /* board id for Linux */
```

然后编译 u-boot，生成 u-boot.bin 文件，覆盖到 MMC 卡原来的 u-boot.bin 文件，然后给 Mini 板加电。注意加电时要按住板上的按键 S1，否则处理器会从 NAND 启动，u-boot 还是错的。

加电后可以看到 linux 内核可以正常运行，不过在 u-boot 下似乎 I2C 总线有一些问题，报总线访问超时等现象，后检查发现是编译器版本不同造成的，原来的 u-boot 是在 2007q3 版本下编译的，而报 I2C 总线问题的是在 2008q1 版本下编译的，如果拿 2007q3 版本编译则不会出现这个问题，这可能是 2008q1 内有针对处理器的优化部分，因为 u-boot 下 I2C 总线访问用了一些类似 for 的延时语句，在这上面新版本优化后可能超时时间减小了，不过这里不确定。

4.2.10 模块问题

在上一节说明 linux 运行时报缺少模块文件目录错误，如下，

```
Starting Bluetooth subsystem:modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory

 hcid hid2hci.
Running ntpdate to synchronize clockmodprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or direc
tory

Error : Temporary failure in name resolution
.
/etc/rc5.d/S81cups: line 232: echo_success: not found
cups: started scheduler.
Starting GPE display manager: gpe-dm
modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/2.6.29-rc3-omap3/modules.dep: No such file or directory
```

图 11 模块错误截图

后查找这个目录是应该在文件系统中，参考网上文章《OMAP35x DVEVM Software Setup》，原来是文件系统的模块文件没有更新，在 linux 编译环境下键入：

make modules

make INSTALL_MOD_PATH=<target filesys dir> modules_install

其中目标文件目录 target filesys dir 是根文件系统的目录，对于本文来说因暂时不使用 NFS 系统，这个目录可以随意，编译完可以在目标文件目录下得到一个 lib 目录，再进入两级子目录可以看到目录/2.6.29-rc3-omap3。将 Mini 板上的 MMC 卡取下插入读卡器中，将读卡器插入计算机 USB 口，可以看到有两个分区，进入 ext3 的分区将目录/2.6.29-rc3-omap3 拷贝到 MMC 卡的/lib/modules/目录内，拷贝完毕将 MMC 卡取出再插入 Mini 板上加电。可以看到这次不再报模块错误了。

4.3 Mini 板 CODEC 驱动程序修改

由于 Mini 板使用的 PMU 是 TPS65930，其 CODEC 部分与 TPS65950 相比有一些区别，因此这一部分考虑修改 CODEC 驱动程序，以使 Mini 板的 CODEC 部分能够正常工作。虽然将来不一定使用 TPS65930 的 CODEC 部分，但也可以熟悉驱动程序的修改过程。

4.3.1 TPS65930 与 TPS65950 的区别

要想对程序进行修改，首先就要清楚需要修改的地方，也就是 TPS65930 和 TPS65950 之间的区别，这里只讨论二者在 CODEC 部分上的区别，其它部分暂时不考虑。区别主要是参考二者的 datasheet 和 technical reference manual，分别是 SWCS037E、SWCU052C、SWCS032C 和 SWCU050D，上述文档都可以在 TI 的网站上下载。

TPS65930 与 TPS65950 相比主要是减少了一些功能，下图的二者的原理框图和接口示意图就可以比较明显的看出二者的区别。

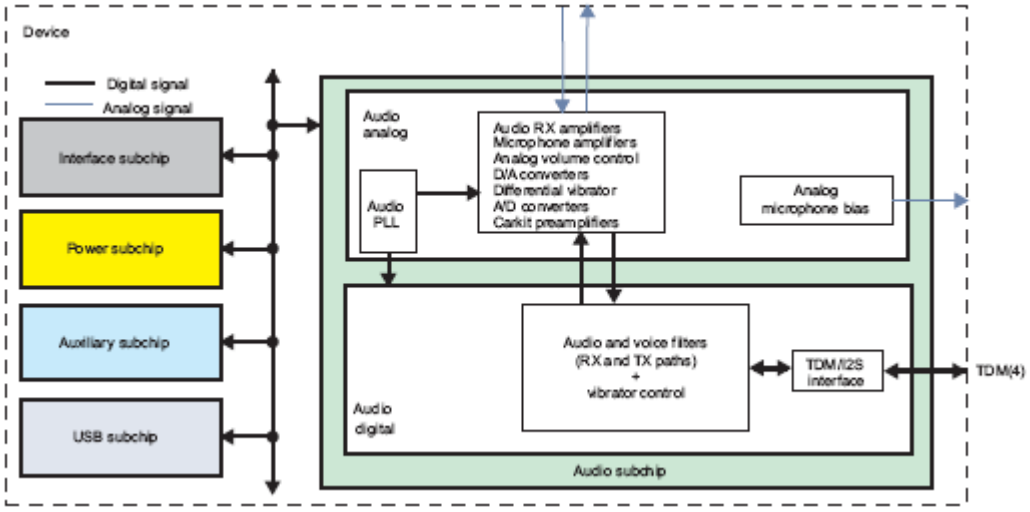


图 12 TPS65930 原理框图 1

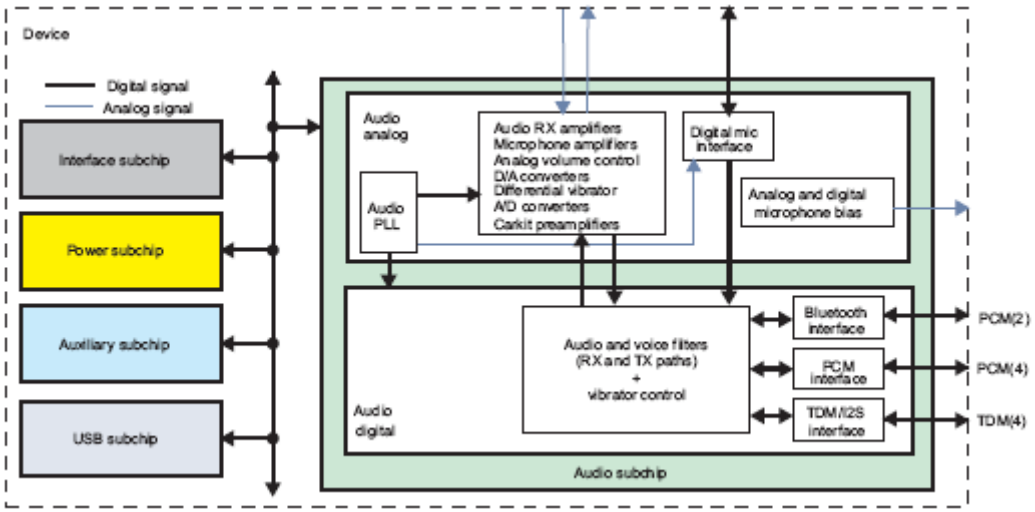


图 13 TPS65950 原理框图 1

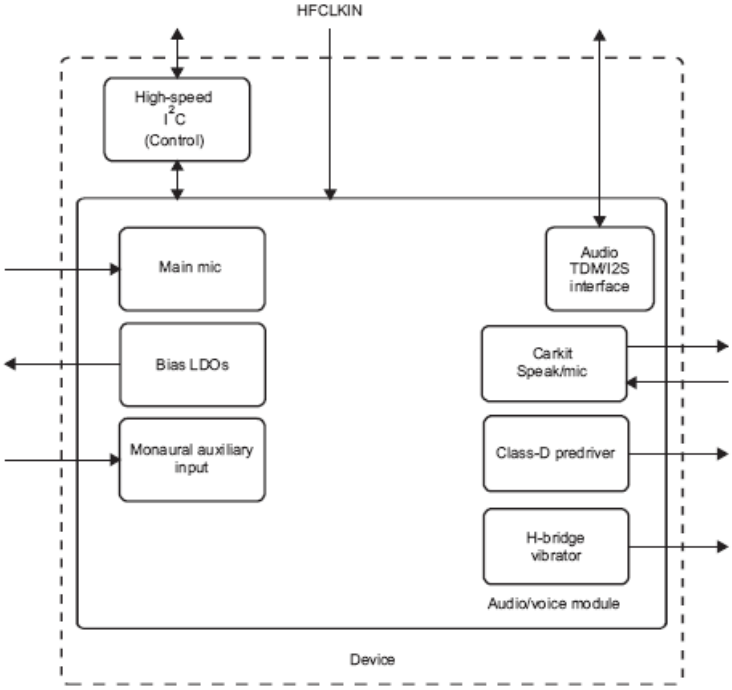


图 14 TPS65930 原理框图 2

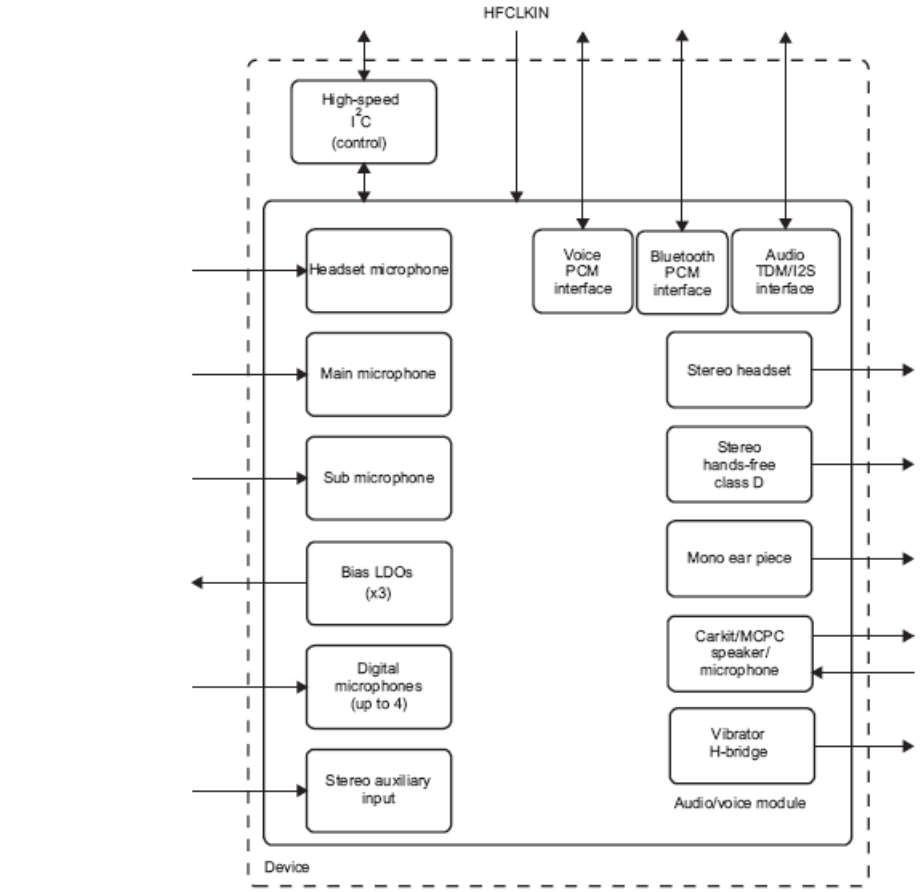


图 15 TPS65950 原理框图 2

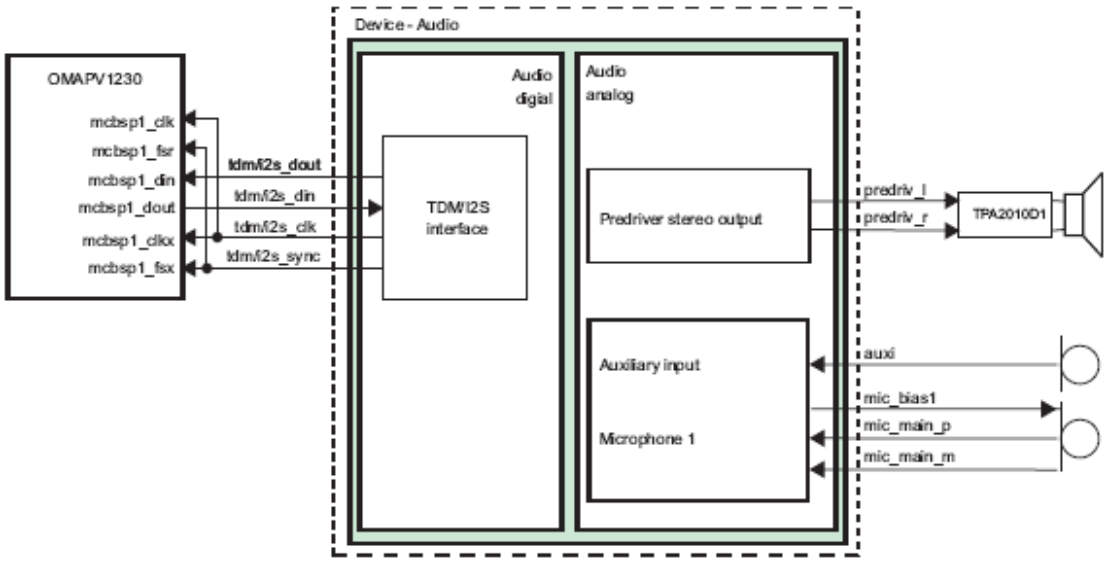


图 16 TPS65930 接口示意图

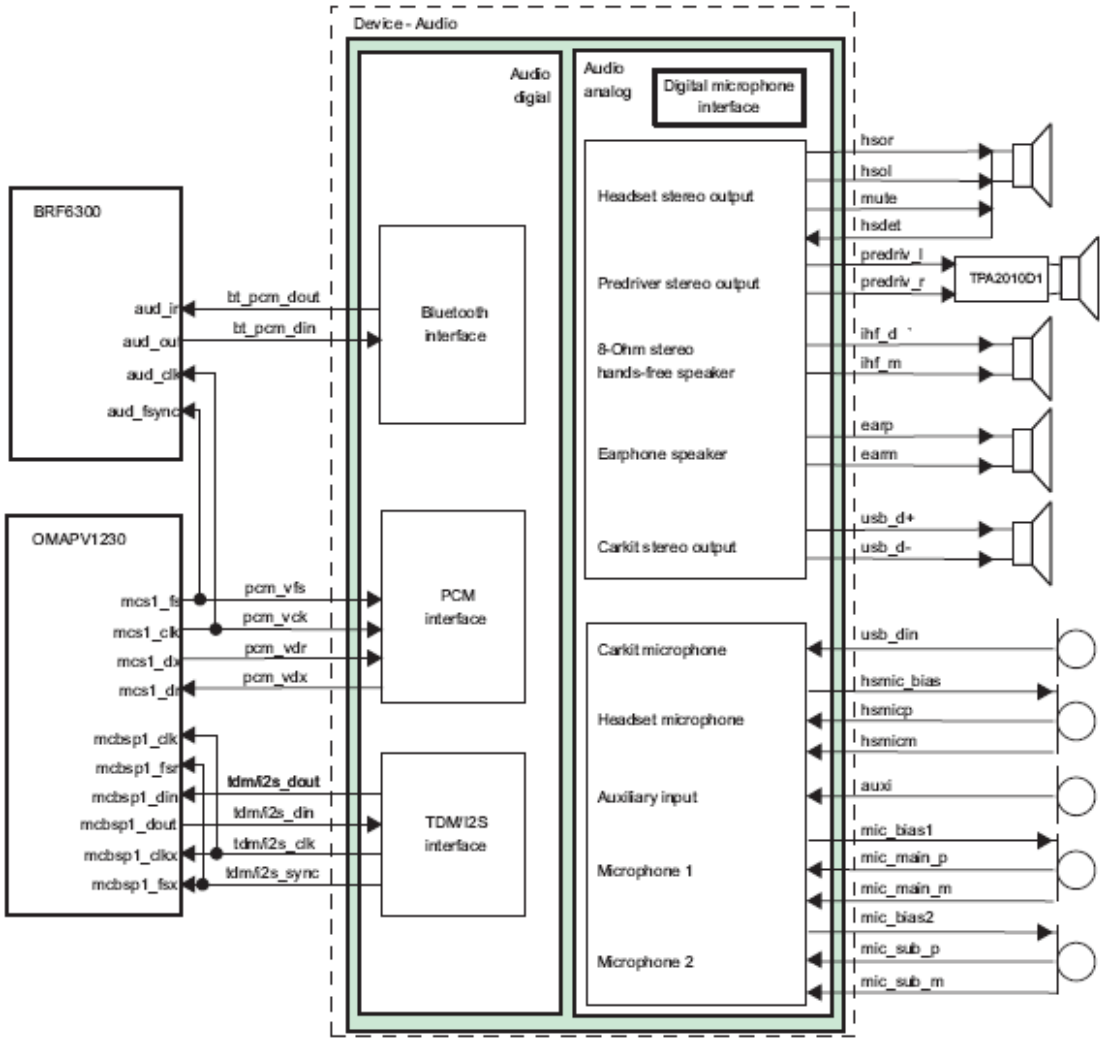


图 17 TPS65950 接口示意图

下面从两个方面说明二者的区别。

4.3.1.1 数字音频接口

可以看到对于数字音频接口部分，TPS65930 只有一个 I2S 接口；而 TPS65950 则有 I2S、Bluetooth PCM 和 voice PCM 接口可以使用，因此它可以接蓝牙芯片等接口。不过对于 beagleboard 和许多 OMAP3530 开发板，实际上只使用了 I2S 接口，而目前 linux 的驱动程序主要也是针对 I2S 接口的，因此对应数字接口部分在软件上基本没有改动。

4.3.1.2 模拟音频接口

这部分的差别比较大，相对于 TPS65930 只有一个模拟麦克风输入、一个辅助输入、一个预驱动放大器立体声输出而言，TPS65950 有大量的输入、输出通道可供选择。因此这方面对应的软件改动会出现，不过这部分的改动对应的只是

寄存器配置上的区别，也就是说 TPS65950 的配置选项有多种，而 TPS65930 只能选其中一种，因此只需要找出这一种配置方式，并通过宏定义出在使用 TPS65930 时的特定部分，应该就可以了。二者的寄存器配置都是通过 I2C 总线进行的，因此修改的关键是清楚寄存器的含义。

4.3.2 TPS65930 音频配置寄存器

这一部分说明 TPS65930 中与音频相关的配置寄存器包括与 TPS65950 之间的一些区别，在了解了这些寄存器后就可以清楚如何配置了。实际上二者的 CODEC 相关寄存器是完全相同的，只不过有一些对于 TPS65930 来说没有用。

寄存器是通过 I2C 总线访问的，因此每个寄存器都是 8 位长度，共 68 个寄存器。下面逐个分析寄存器的内容。

这里需要说明一下，对于 TPS65950 来说有两种工作模式，其中模式 2 中使用到了 PCM 接口，而 TPS65930 没有这种工作模式，因此下面的寄存器说明中不再考虑模式 2 的情况，只针对模式 1 说明。

4.3.2.1 CODEC_MODE 寄存器

这个寄存器是系统模式控制寄存器。

7	6	5	4	3	2	1	0
APLL_RATE				SEL_16K	SPARE	CODECPDZ	OPT_MODE

位	名称	功能说明	类型	复位值
7:4	APLL_RATE	音频模式：选择采样频率 0x0：8kHz 0x1：11.025kHz 0x2：12kHz 0x4：16kHz 0x5：22.05kHz 0x6：24kHz 0x8：32kHz	RW	0x0
		0x9：44.1kHz 0xA：48kHz 0xE：96kHz		
3	SEL_16K	语音模式： 0x0：采样频率 8kHz 0x1：采样频率 16kHz	RW	0x0
2	SPARE	spare 位	RW	0x0
1	CODECPDZ	CODEC 电源控制； 0x0：CODEC 关闭 0x1：CODEC 打开	RW	0x0

0	OPT_MODE	音频和语音模式选择： 0x0：模式 2； 0x1：模式 1，注意 TPS65930 只能工作在模式 1 下，即这一位只能选 1	RW	0x0

这里的音频可能主要指 I2S 接口，语音主要指 PCM 接口，也就是说对于 TPS65930 语音相关的是没有用的。

4.3.2.2 OPTION 寄存器

这个寄存器控制音频/语音数字滤波器电源。

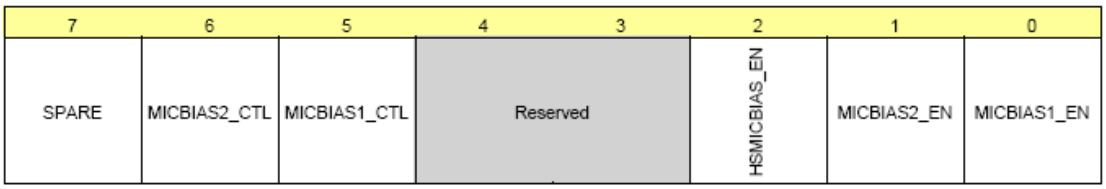
7	6	5	4	3	2	1	0
ARXR2_EN	ARXL2_EN	ARXR1_EN	ARXL1_VRX_EN	ATXR2_VTXR_EN	ATXL2_VTXL_EN	ATXR1_EN	ATXL1_EN

位	名称	功能说明	类型	复位值
7	ARXR2_EN	音频 RX 右 2 使能位： 0x0：禁止 0x1：使能	RW	0x0
6	ARXL2_EN	音频 RX 左 2 使能位： 0x0：禁止 0x1：使能	RW	0x0
5	ARXR1_EN	音频 RX 右 1 使能位： 0x0：禁止 0x1：使能	RW	0x0
4	ARXL1_VRX_EN	音频 RX 左 1 使能位： 0x0：禁止 0x1：使能	RW	0x0
3	ATXR2_VTXR_EN	音频 TX 右 2 使能位： 0x0：禁止 0x1：使能	RW	0x0
2	ATXL2_VTXL_EN	音频 TX 左 2 使能位： 0x0：禁止 0x1：使能	RW	0x0
1	ATXR1_EN	音频 TX 右 1 使能位： 0x0：禁止 0x1：使能	RW	0x0
0	ATXL1_EN	音频 TX 左 1 使能位： 0x0：禁止 0x1：使能	RW	0x0

4.3.2.3 MICBIAS_CTL 寄存器

这个寄存器控制麦克风偏置电压和模拟麦克风放大器电源，只有在

CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



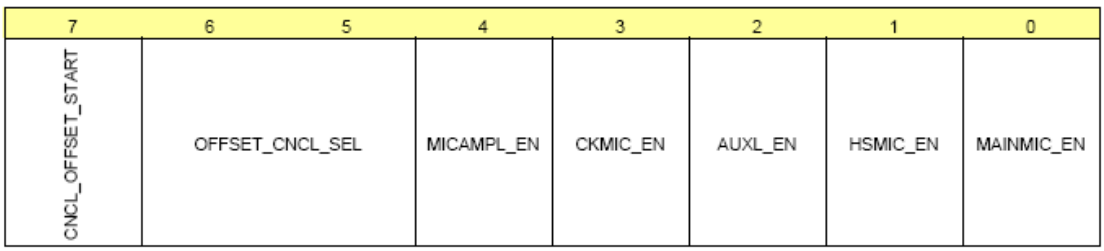
7	6	5	4	3	2	1	0
SPARE	MICBIAS2_CTL	MICBIAS1_CTL	Reserved		HSMICBIAS_EN	MICBIAS2_EN	MICBIAS1_EN

位	名称	功能说明	类型	复位值
7	SPARE	spare 位	RW	0x0
6	MICBIAS2_CTL	麦克风偏置 2 控制位： 0x0：模拟麦克风偏置 0x1：数字麦克风偏置	RW	0x0
5	MICBIAS1_CTL	麦克风偏置 1 控制位： 0x0：模拟麦克风偏置 0x1：数字麦克风偏置	RW	0x0
4:3	保留位		RW	0x0
2	HSMICBIAS_EN	耳机麦克风偏置电源控制位： 0x0：禁止 0x1：使能	RW	0x0
1	MICBIAS2_EN	副麦克风偏置电源控制位： 0x0：禁止 0x1：使能	RW	0x0
0	MICBIAS1_EN	主麦克风偏置电源控制位： 0x0：禁止 0x1：使能	RW	0x0

这里需要说明一下，对于 TPS65930 来说没有数字麦克风，因此有关麦克风的选项中只能选模拟麦克风。

4.3.2.4 ANAMICL 寄存器

这个寄存器控制左模拟麦克风和偏移消除，左模拟麦克风包括主麦克风、耳机麦克风、carkit 麦克风和辅助左麦克风，它们可以分别连接到 A/D 的输入（但只能有一个使能位置 1）。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



7	6	5	4	3	2	1	0
CNCL_OFFSET_START	OFFSET_CNCL_SEL		MICAMPL_EN	CKMIC_EN	AUXL_EN	HSMIC_EN	MAINMIC_EN

位	名称	功能说明	类型	复位值

7	CNCL_OFFSET_START	RX 路径偏移消除控制位。置 1 时发起偏移消除，当偏移消除计算完毕后，这一位自动清 0 0x0：应用模式 0x1：发起偏移消除	RW	0x0
6:5	OFFSET_CNCL_SEL	选择偏移消除路径控制位： 0x0：音频 RX 左 1 和右 1 0x1：音频 RX 左 2 和右 2 0x2：语音 RX 0x3：全部选择	RW	0x0
4	MICAMPL_EN	左麦克风放大器电源控制位 0x0：禁止 0x1：使能	RW	0x0
3	CKMIC_EN	carkit 麦克风输入控制位 0x0：禁止 0x1：使能	RW	0x0
2	AUXL_EN	AUXL 输入控制位 0x0：禁止 0x1：使能	RW	0x0
1	HSMIC_EN	耳机麦克风输入控制位 0x0：禁止 0x1：使能	RW	0x0
0	MAINMIC_EN	主麦克风输入控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.5 ANAMICR 寄存器

这个寄存器控制右模拟麦克风，右模拟麦克风包括副麦克风辅助右麦克风，它们可以分别连接到 A/D 的输入(但只能有一个使能位置 1)。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。

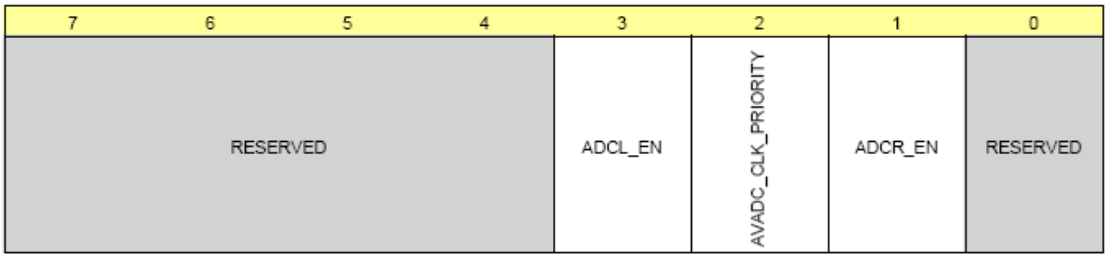
7	6	5	4	3	2	1	0
RESERVED			MICAMPR_EN	RESERVED	AUXR_EN	RESERVED	SUBMIC_EN

位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4	MICAMPR_EN	右麦克风放大器电源控制位 0x0：禁止 0x1：使能	RW	0x0
3	保留位		RW	0x0
2	AUXR_EN	AUXR 输入控制位 0x0：禁止 0x1：使能	RW	0x0
1	保留位		RW	0x0
0	SUBMIC_EN	副麦克风输入控制位	RW	0x0

		0x0：禁止 0x1：使能		

4.3.2.6 AVADC_CTL 寄存器

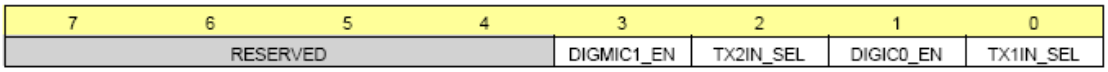
这个寄存器控制音频/语音的 A/D 转换器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7:4	保留位		RW	0x0
3	ADCL_EN	左 A/D 转换器电源控制位 0x0：禁止 0x1：使能	RW	0x0
2	AVADC_CLK_ PRIORITY	仅在模式 2 下这一位有效，不考虑，默认即可	RW	0x0
1	ADCR_EN	右 A/D 转换器电源控制位 0x0：禁止 0x1：使能	RW	0x0
0	保留位		RW	0x0

4.3.2.7 ADCMICSEL 寄存器

这个寄存器控制数字音频/语音 TX 滤波器的输入选择。

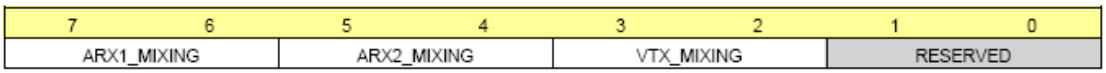


位	名称	功能说明	类型	复位值
7:4	保留位		RW	0x0
3	DIGMIC1_EN	数字麦克风 1 接口电源控制 0x0：禁止 0x1：使能	RW	0x0
2	TX2IN_SEL	音频模式下 TX2 的输入选择位 0x0：ADC 输入 0x1：数字麦克风 1	RW	0x0
1	DIGMIC0_EN	数字麦克风 0 接口电源控制 0x0：禁止 0x1：使能	RW	0x0
0	TX1IN_SEL	音频模式下 TX1 的输入选择位 0x0：ADC 输入	RW	0x0

		0x1：数字麦克风 0		

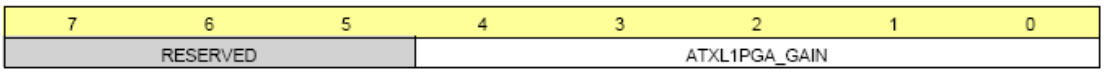
4.3.2.8 DIGMIXING 寄存器

这个寄存器控制音频/语音数字混频。对于模式 1 这个寄存器必须为 0x0。



4.3.2.9 ATXL1PGA 寄存器

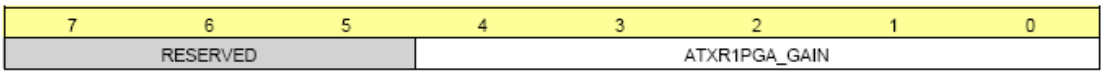
这个寄存器控制音频 TXL1 的增益。



位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4:0	ATXL1PGA_GAIN	TXL1 数字增益控制 0x0：0dB 0x1：1dB 0x2：2dB … 0x1F：31dB	RW	0x0F

4.3.2.10 ATXR1PGA 寄存器

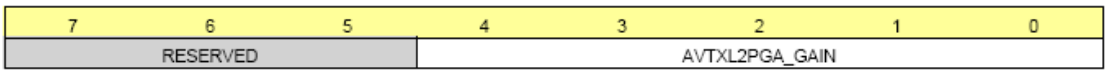
这个寄存器控制音频 TXR1 的增益。



位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4:0	ATXR1PGA_GAIN	TXR1 数字增益控制 0x0：0dB 0x1：1dB 0x2：2dB … 0x1F：31dB	RW	0x0F

4.3.2.11 AVTXL2PGA 寄存器

这个寄存器控制音频/语音 TXL2 的增益。

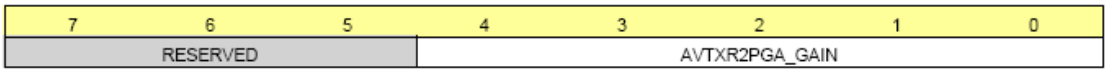


位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4:0	AVTXL2PGA_ GAIN	TXL2 数字增益控制 0x0：0dB 0x1：1dB	RW	0x0F

		0x2：2dB … 0x1F：31dB		

4.3.2.12 AVTXR2PGA 寄存器

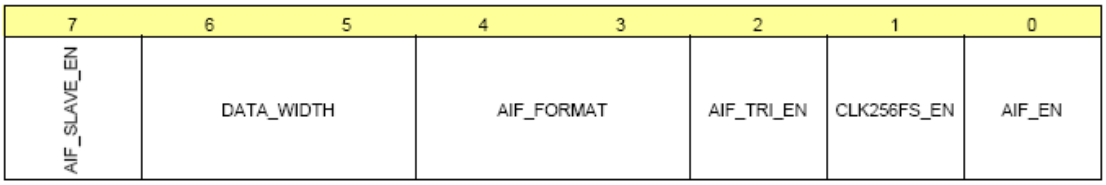
这个寄存器控制音频/语音 TXR2 的增益。



位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4:0	AVTXR2PGA_ GAIN	TXR2 数字增益控制 0x0：0dB 0x1：1dB 0x2：2dB … 0x1F：31dB	RW	0x0F

4.3.2.13 AUDIO_IF 寄存器

这个寄存器控制音频接口模式。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7	AIF_SLAVE_EN	主从模式选择位 0x0：TPS65930 为主模式 0x1：TPS65930 为从模式	RW	0x0
6:5	DATA_WIDTH	音频接口的字长和采样长度控制位 0x0：16 位采样长度，16 位字长 0x1：保留 0x2：32 位采样长度，16 位字长 0x3：32 位采样长度，24 位字长	RW	0x0
4:3	AIF_FORMAT	音频接口数据格式选择位 0x0：codec 模式，即 I2S 0x1：左调整模式 0x2：右调整模式 0x3：TDM 模式	RW	0x0
2	AIF_TRI_EN	音频接口输出管脚高阻选择位 0x0：应用模式 0x1：高阻	RW	0x0

1	CLK256FS_EN	256FS 时钟输出使能控制位 0x0：禁止 0x1：使能	RW	0x0
0	AIF_EN	音频串行接口模式控制位 0x0：关闭模式，DIN/DOUT/SYN/CLK=L，这里有点疑问 DOUT 为何也是低 0x1：应用模式	RW	0x0

4.3.2.14 VOICE_IF 寄存器

这个寄存器控制语音接口模式。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
VIF_SLAVE_EN	VIF_DIN_EN	VIF_DOUT_EN	VIF_SWAP	VIF_FORMAT	VIF_TRI_EN	VIF_SUB_EN	VIF_SUB_EN

4.3.2.15 ARXR1PGA 寄存器

这个寄存器控制音频 RXR1 增益。

7	6	5	4	3	2	1	0
ARXR1PGA_CGAIN		ARXR1PGA_FGAIN					

位	名称	功能说明	类型	复位值
7:6	ARXR1PGA_ CGAIN	增益粗调控制位 0x0：0dB 0x1：6dB 0x2：2dB 0x3：12dB	RW	0x0
5:0	ARXR1PGA_ FGAIN	增益精调控制位 0x0：静音 0x1：-62dB 0x2：-61dB … 0x3F：0dB	RW	0x3F

4.3.2.16 ARXL1PGA 寄存器

这个寄存器控制音频 RXL1 增益。

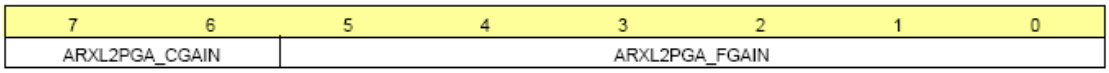
7	6	5	4	3	2	1	0
ARXL1PGA_CGAIN		ARXL1PGA_FGAIN					

位	名称	功能说明	类型	复位值
7:6	ARXL1PGA_	增益粗调控制位	RW	0x0

	CGAIN	0x0：0dB 0x1：6dB 0x2：2dB 0x3：12dB		
5:0	ARXL1PGA_ FGAIN	增益精调控制位 0x0：静音 0x1：-62dB 0x2：-61dB … 0x3F：0dB	RW	0x3F

4.3.2.17 ARXR2PGA 寄存器

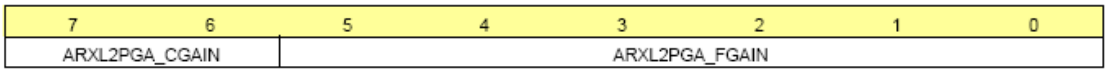
这个寄存器控制音频 RXR2 增益。



位	名称	功能说明	类型	复位值
7:6	ARXR2PGA_ CGAIN	增益粗调控制位 0x0：0dB 0x1：6dB 0x2：2dB 0x3：12dB	RW	0x0
5:0	ARXR2PGA_ FGAIN	增益精调控制位 0x0：静音 0x1：-62dB 0x2：-61dB … 0x3F：0dB	RW	0x3F

4.3.2.18 ARXL2PGA 寄存器

这个寄存器控制音频 RXL2 增益。

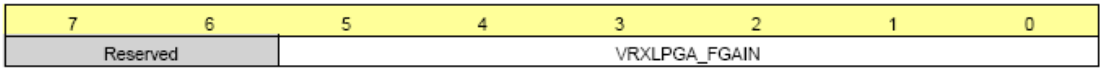


位	名称	功能说明	类型	复位值
7:6	ARXL2PGA_ CGAIN	增益粗调控制位 0x0：0dB 0x1：6dB 0x2：2dB 0x3：12dB	RW	0x0
5:0	ARXL2PGA_ FGAIN	增益精调控制位 0x0：静音 0x1：-62dB 0x2：-61dB …	RW	0x3F

		0x3F：0dB		

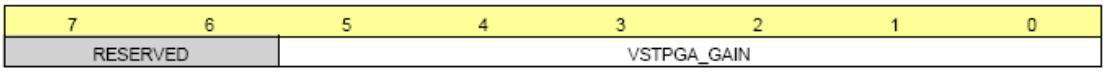
4.3.2.19 VRXPGA 寄存器

这个寄存器控制语音下行线路增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。



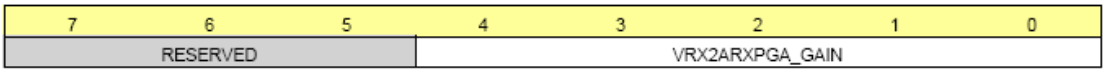
4.3.2.20 VSTPGA 寄存器

这个寄存器控制语音侧音增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。



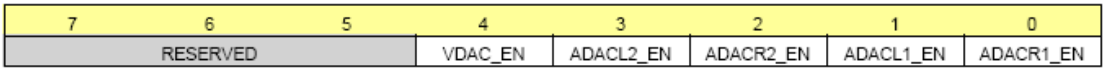
4.3.2.21 VRX2ARXPGA 寄存器

这个寄存器控制语音 RX 到音频 RX 的 PGA 增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。



4.3.2.22 AVDAC_CTL 寄存器

这个寄存器控制音频 DAC 和语音 DAC 电源。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。

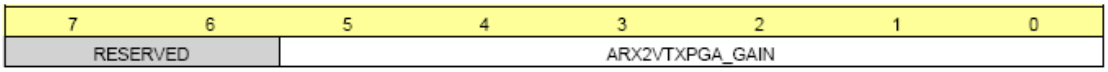


位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4	VDAC_EN	语音 DAC 电源控制位 0x0：禁止 0x1：使能	RW	0x0
3	ADACL2_EN	音频 DACL2 电源控制位 0x0：禁止 0x1：使能	RW	0x0
2	ADACR2_EN	音频 DACR2 电源控制位 0x0：禁止 0x1：使能	RW	0x0
1	ADACL1_EN	音频 DACL1 电源控制位 0x0：禁止 0x1：使能	RW	0x0
0	ADACR1_EN	音频 DACR1 电源控制位	RW	0x0

		0x0：禁止 0x1：使能		

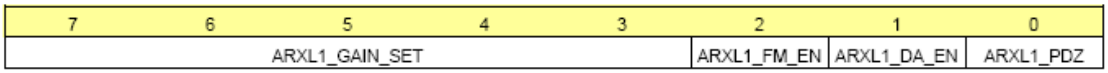
4.3.2.23 ARX2VTXPGA 寄存器

这个寄存器控制音频 RX 到语音 TX 的 PGA 增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。



4.3.2.24 ARXL1_APGA_CTL 寄存器

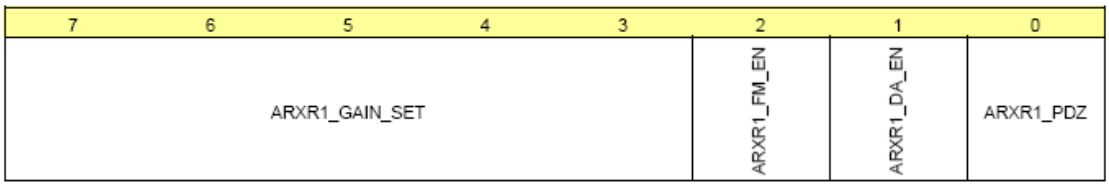
这个寄存器控制音频 RXL1 的模拟 PGA。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7:3	ARXL1_GAIN _SET	模拟可编程增益控制位，需要静音可以通过禁止所有输入通道实现 0x0：12dB 0x1：10dB 0x2：8dB … 0x0F：-18dB	RW	0x06
2	ARXL1_FM_EN	FM 回路使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	ARXL1_DA_EN	数字-模拟路径使能控制位 0x0：禁止 0x1：使能	RW	1
0	ARXL1_PDZ	模拟 PGA 电源控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.25 ARXR1_APGA_CTL 寄存器

这个寄存器控制音频 RXR1 的模拟 PGA。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值

7:3	ARXR1_GAIN_SET	模拟可编程增益控制位，需要静音可以通过禁止所有输入通道实现 0x0：12dB 0x1：10dB 0x2：8dB … 0x12：-24dB	RW	0x06
2	ARXR1_FM_EN	FM 回路使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	ARXR1_DA_EN	数字-模拟路径使能控制位 0x0：禁止 0x1：使能	RW	1
0	ARXR1_PDZ	模拟 PGA 电源控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.26 ARXL2_APGA_CTL 寄存器

这个寄存器控制音频 RXL2 的模拟 PGA。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。

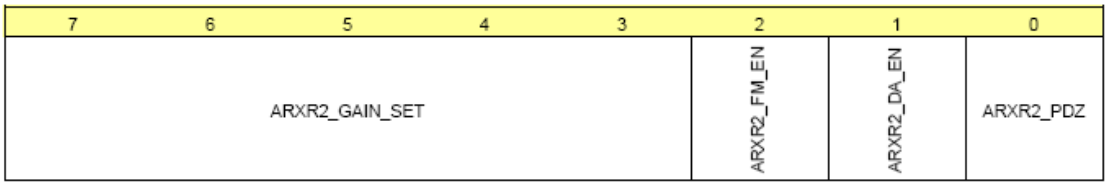
7	6	5	4	3	2	1	0
ARXL2_GAIN_SET					ARXL2_FM_EN	ARXL2_DA_EN	ARXL2_PDZ

位	名称	功能说明	类型	复位值
7:3	ARXL2_GAIN_SET	模拟可编程增益控制位，需要静音可以通过禁止所有输入通道实现 0x0：12dB 0x1：10dB 0x2：8dB … 0x12：-24dB	RW	0x06
2	ARXL2_FM_EN	FM 回路使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	ARXL2_DA_EN	数字-模拟路径使能控制位 0x0：禁止 0x1：使能	RW	1
0	ARXL2_PDZ	模拟 PGA 电源控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.27 ARXR2_APGA_CTL 寄存器

这个寄存器控制音频 RXR2 的模拟 PGA。只有在 CODECPDZ 位置 1 后这个

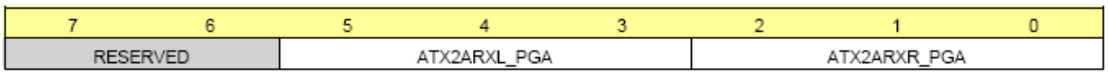
寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7:3	ARXR2_GAIN_SET	模拟可编程增益控制位，需要静音可以通过禁止所有输入通道实现 0x0：12dB 0x1：10dB 0x2：8dB … 0x12：-24dB	RW	0x06
2	ARXR2_FM_EN	FM 回路使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	ARXR2_DA_EN	数字-模拟路径使能控制位 0x0：禁止 0x1：使能	RW	1
0	ARXR2_PDZ	模拟 PGA 电源控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.28 ATX2ARXPGA 寄存器

这个寄存器控制音频 TX 到音频 RX 的 PGA 的数字增益。



位	名称	功能说明	类型	复位值
7:6	保留位		RW	0x00
5:3	ATX2ARXL_PGA	回路 TX 到 RX 数字增益控制位(左) 0x0：静音 0x1：-24dB 0x2：-24dB 0x3：-24dB 0x4：-18dB 0x5：-12dB 0x6：-6dB 0x7：0dB	RW	0x0
2:0	ATX2ARXR_PGA	回路 TX 到 RX 数字增益控制位(右) 0x0：静音 0x1：-24dB 0x2：-24dB	RW	0x0

		0x3：-24dB 0x4：-18dB 0x5：-12dB 0x6：-6dB 0x7：0dB		

4.3.2.29 BT_IF 寄存器

这个寄存器控制蓝牙接口模式。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
SPARE	BT_DIN_EN	BT_DOUT_EN	BT_SWAP	RESERVED	BT_TRI_EN	RESERVED	BT_EN

4.3.2.30 BTPGA 寄存器

这个寄存器控制蓝牙路径数字增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
BTTXPGA_GAI				BTRXPGA_GAIN			

4.3.2.31 BTSTPGA 寄存器

这个寄存器控制蓝牙侧音增益。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
RESERVED		BTXTPGA_GAIN					

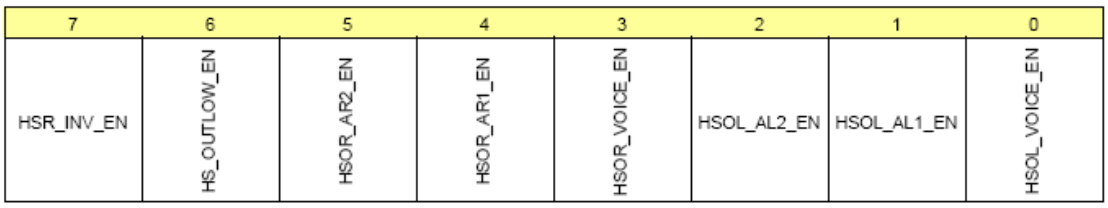
4.3.2.32 EAR_CTL 寄存器

这个寄存器控制听筒放大器。TPS65930 没有这个接口 ，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
EAR_OUTLOW_EN	SPARE	EAR_GAIN		EAR_AR1_EN	EAR_AL2_EN	EAR_AL1_EN	EAR_VOICE_EN

4.3.2.33 HS_SEL 寄存器

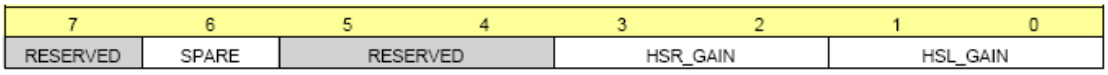
这个寄存器控制耳机放大器。TPS65930 没有这个接口 ，这里不考虑，取其默认值即可。



7	6	5	4	3	2	1	0
HSR_INV_EN	HS_OUTLOW_EN	HSOR_AR2_EN	HSOR_AR1_EN	HSOR_VOICE_EN	HSOL_AL2_EN	HSOL_AL1_EN	HSOL_VOICE_EN

4.3.2.34 HS_GAIN_SET 寄存器

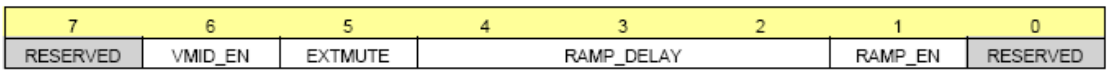
这个寄存器控制耳机放大器增益。TPS65930 没有这个接口 ， 这里不考虑，取其默认值即可。



7	6	5	4	3	2	1	0
RESERVED	SPARE	RESERVED		HSR_GAIN		HSL_GAIN	

4.3.2.35 HS_POPN_SET 寄存器

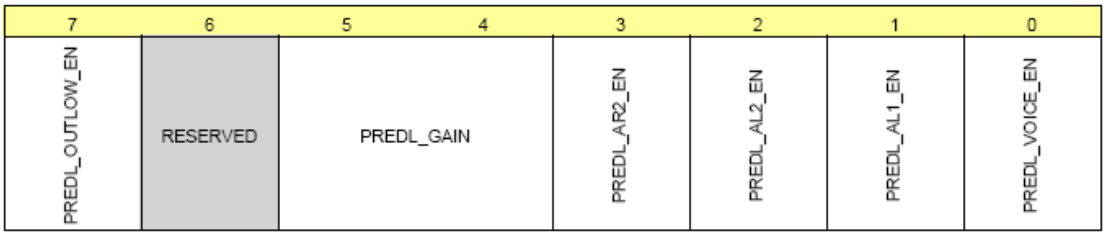
这个寄存器控制耳机放大器破音衰减。TPS65930 没有这个接口 ，这里不考虑，取其默认值即可。



7	6	5	4	3	2	1	0
RESERVED	VMID_EN	EXTMUTE	RAMP_DELAY			RAMP_EN	RESERVED

4.3.2.36 PREDL_CTL 寄存器

这个寄存器控制左预驱动放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



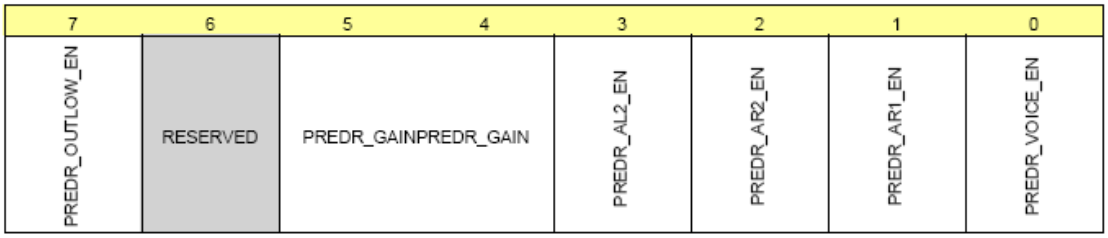
7	6	5	4	3	2	1	0
PREDL_OUTLOW_EN	RESERVED	PREDL_GAIN		PREDL_AR2_EN	PREDL_AL2_EN	PREDL_AL1_EN	PREDL_VOICE_EN

位	名称	功能说明	类型	复位值
7	PREDL_OUTLOW_EN	输出低电平控制位 0x0：禁止 0x1：使能(输出低电平)	RW	0x0
6	保留位		RW	0x0
5:4	PREDL_GAIN	放大器增益控制位，静音控制可以通过禁止所有输入通道实现 0x0：停机 0x1：6dB 0x2： 0dB 0x3：-6dB	RW	0x0
3	PREDL_AR2_EN	音频 R2 使能控制位 0x0：禁止 0x1：使能	RW	0x0

2	PREDL_AL2_EN	音频 L2 使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	PREDL_AL1_EN	音频 L1 使能控制位 0x0：禁止 0x1：使能	RW	0x0
0	PREDL_VOICE _EN	语音使能控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.37 PREDR_CTL 寄存器

这个寄存器控制右预驱动放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7	PREDR_OUTLOW _EN	输出低电平控制位 0x0：禁止 0x1：使能(输出低电平)	RW	0x0
6	保留位		RW	0x0
5:4	PREDR_GAIN	放大器增益控制位，静音控制可以通过禁止所有输入通道实现 0x0：停机 0x1：6dB 0x2： 0dB 0x3：-6dB	RW	0x0
3	PREDR_AL2_EN	音频 L2 使能控制位 0x0：禁止 0x1：使能	RW	0x0
2	PREDR_AR2_EN	音频 R2 使能控制位 0x0：禁止 0x1：使能	RW	0x0
1	PREDR_AR1_EN	音频 R1 使能控制位 0x0：禁止 0x1：使能	RW	0x0
0	PREDR_VOICE _EN	语音使能控制位 0x0：禁止 0x1：使能	RW	0x0

4.3.2.38 PRECKL_CTL 寄存器

这个寄存器控制 carkit 左预驱动放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。这个接口一般不会用，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
RESERVED	PRECKL_EN	PRECKL_GAIN		RESERVED	PRECKL_AL2_EN	PRECKL_AL1_EN	PRECKL_VOICE_EN

4.3.2.39 PRECKR_CTL 寄存器

这个寄存器控制 carkit 右预驱动放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。这个接口一般不会用，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
RESERVED	PRECKR_EN	PRECKR_GAIN		RESERVED	PRECKR_AR2_EN	PRECKR_AR1_EN	PRECKR_VOICE_EN

4.3.2.40 HFL_CTL 寄存器

这个寄存器控制免提左 D 类放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。TPS65930 没有这个接口 ，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
RESERVED		HFL_REF_EN	HFL_RAMP_EN	HFL_LOOP_EN	HFL_HB_EN	HFL_INPUT_SEL	

4.3.2.41 HFR_CTL 寄存器

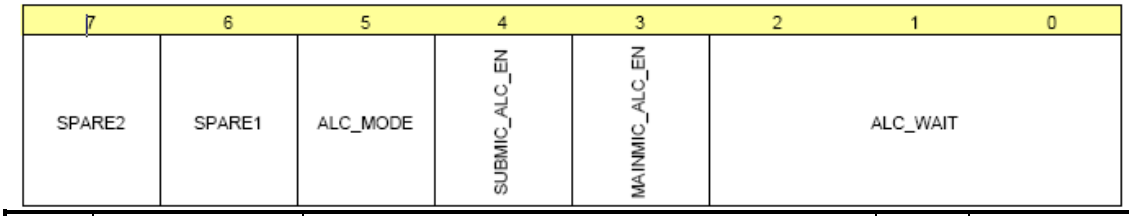
这个寄存器控制免提右 D 类放大器。只有在 CODECPDZ 位置 1 后这个寄存器的设置内容才有效。TPS65930 没有这个接口 ，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
RESERVED		HFR_REF_EN	HFR_RAMP_EN	HFR_LOOP_EN	HFR_HB_EN	HFR_INPUT_SEL	

4.3.2.42 ALC_CTL 寄存器

这个寄存器控制 ALC（自动电平控制）。只有在 CODECPDZ 位置 1 后这个

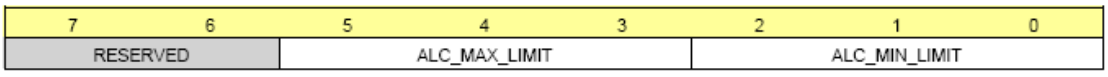
寄存器的设置内容才有效。



位	名称	功能说明	类型	复位值
7	SPARE2	spare 位	RW	0x0
6	SPARE1	spare 位	RW	0x0
5	ALC_MODE	ALC 模式控制位 0x0：正常 0x1：互锁	RW	0x0
4	SUBMIC_ALC _EN	副麦克风 ACL 电源控制位 0x0：关闭 0x1：打开		
3	MAINMIC_ALC _EN	主麦克风 ACL 电源控制位 0x0：关闭 0x1：打开	RW	0x0
2:0	ALC_WAIT	等待时间控制位 0x0：1×WaitToRelease 0x1：2×WaitToRelease 0x2：4×WaitToRelease 0x3：8×WaitToRelease … 0x7：128×WaitToRelease	RW	0x5

4.3.2.43 ALC_SET1 寄存器

这个寄存器控制 ALC 高低电平门限。

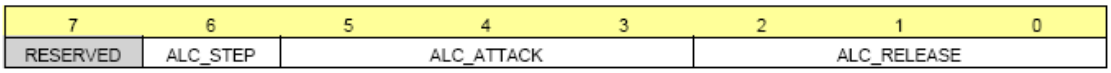


位	名称	功能说明	类型	复位值
7:6	保留位		RW	0x0
5:3	ALC_MAX_LIMIT	ALC 高门限控制位 0x0：-9dB 0x1：-12dB 0x2：-15dB … 0x6：-27dB 0x7：-27dB	RW	0x0
2:0	ALC_MIN_LIMIT	ALC 低门限控制位 0x0：-12dB 0x1：-15dB 0x2：-18dB	RW	0x0

		… 0x6：-30dB 0x7：-30dB		

4.3.2.44 ALC_SET2 寄存器

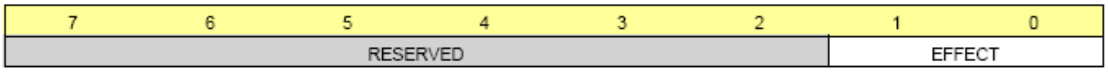
这个寄存器控制 ALC 高低电平门限。



位	名称	功能说明	类型	复位值
7	保留位		RW	0x0
6	ALC_STEP	ALC 步进 0x0：1dB 0x1：2dB	RW	0x0
5:3	ALC_ATTACK	ALC 开始时间控制位，ALC 开始时间定义两个步进递减增益的最小时间 0x0：6.25us 0x1：12.5 us 0x2：25 us … 0x7：800 us	RW	0x0
2:0	ALC_RELEASE	ALC 释放时间控制位，ALC 释放时间定义两个步进递增增益的最小时间 0x0：1.6ms 0x1：3.2 ms 0x2：6.4 ms 0x3：12.8 ms 0x4：25.6 ms 0x5：51.2 ms 0x6：1.6 ms 0x7：1.6 ms	RW	0x0

4.3.2.45 BOOST_CTL 寄存器

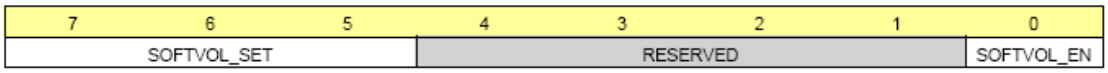
这个寄存器控制均衡器。



位	名称	功能说明	类型	复位值
7:2	保留位		RW	0x0
1:0	EFFECT	重音效果可以加重低频部分，用于补偿高通滤波器 0x0：无效果 0x1：重音 1 模式 0x2：重音 2 模式 0x3：重音 3 模式	RW	0x0

4.3.2.46 SOFTVOL_CTL 寄存器

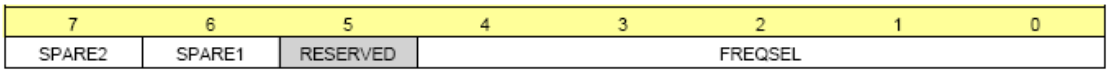
这个寄存器控制软件音量。



位	名称	功能说明	类型	复位值
7:5	SOFTVOL_SET	软件音量扫描时间控制位 0x0：1024×0.8/Fs(Fs-采样频率) 0x1：512×0.8/Fs 0x2：256×0.8/Fs 0x3：128×0.8/Fs 0x4：64×0.8/Fs 0x5：16×0.8/Fs 0x6：4×0.8/Fs 0x7：1×0.8/Fs	RW	0x0
4:1	保留位		RW	0x0
0	SOFTVOL_EN	软件音量模式控制位 0x0：旁路模式 0x1：应用模式	RW	0x0

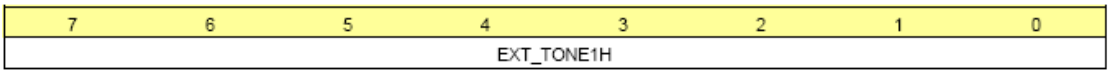
4.3.2.47 DTMF_FREQSEL 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



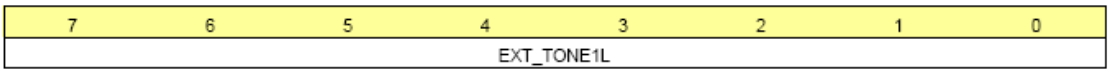
4.3.2.48 DTMF_TONEXT1H 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



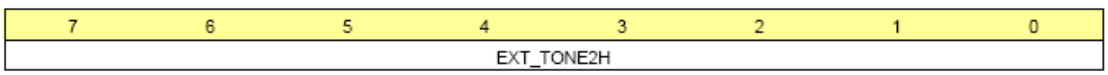
4.3.2.49 DTMF_TONEXT1L 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



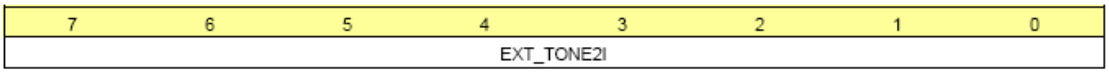
4.3.2.50 DTMF_TONEXT2H 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



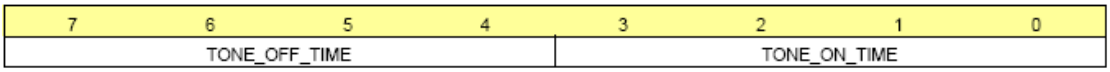
4.3.2.51 DTMF_TONEXT2L 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



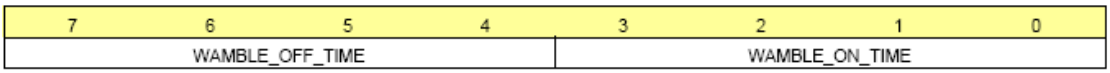
4.3.2.52 DTMF_TONOFF 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



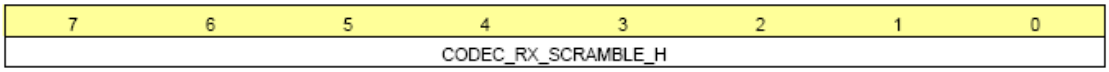
4.3.2.53 DTMF_WANONOFF 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



4.3.2.54 CODEC_RX_SCRAMBLE_H 寄存器

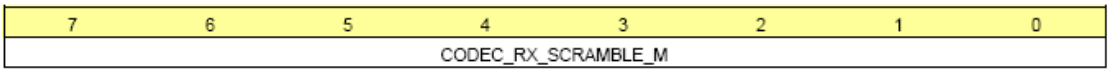
这个寄存器控制 DAC 杂散高 8 位。



位	名称	功能说明	类型	复位值
7:0	CODEC_RX_SCRAMBLE_H	音频 RX 杂散高 8 位	RW	0x0

4.3.2.55 CODEC_RX_SCRAMBLE_M 寄存器

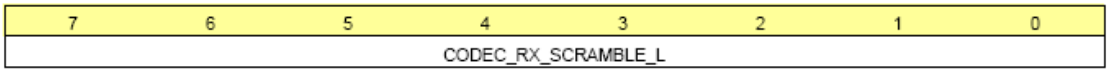
这个寄存器控制 DAC 杂散中 8 位。



位	名称	功能说明	类型	复位值
7:0	CODEC_RX_SCRAMBLE_M	音频 RX 杂散中 8 位	RW	0x0

4.3.2.56 CODEC_RX_SCRAMBLE_L 寄存器

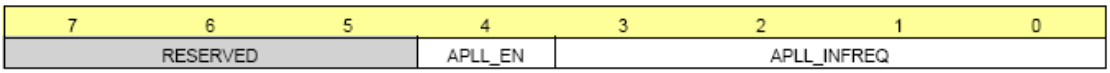
这个寄存器控制 DAC 杂散低 8 位。



位	名称	功能说明	类型	复位值
7:0	CODEC_RX_SCRAMBLE_L	音频 RX 杂散低 8 位	RW	0x0

4.3.2.57 APLL_CTL 寄存器

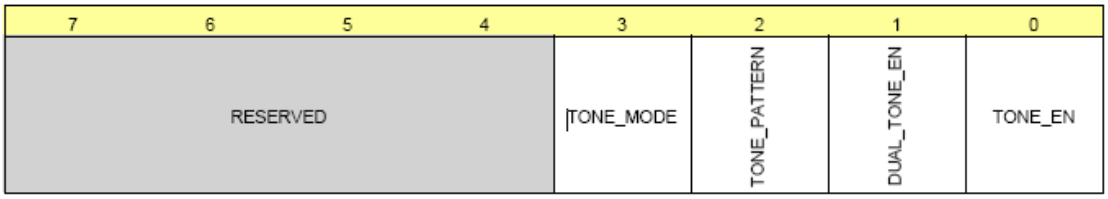
这个寄存器控制音频 PLL。



位	名称	功能说明	类型	复位值
7:5	保留位		RW	0x0
4	APLL_EN	PLL 电源控制位 0x0：禁止 0x1：使能	RW	0x0
3:0	APLL_INFREQ	输入频率选择位 0x0：保留 0x1：保留 0x2：保留 0x3：保留 0x4：保留 0x5：19.2Hz 0x6：26Hz 0x7：保留 0x0F：38.4Hz	RW	0x6

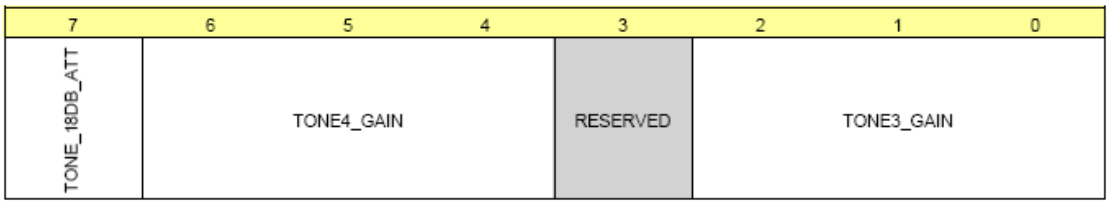
4.3.2.58 DTMF_CTL 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



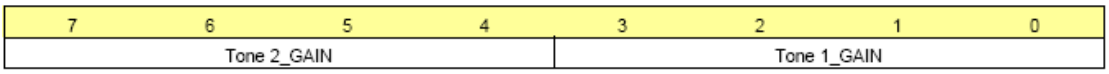
4.3.2.59 DTMF_PGA_CTL2 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



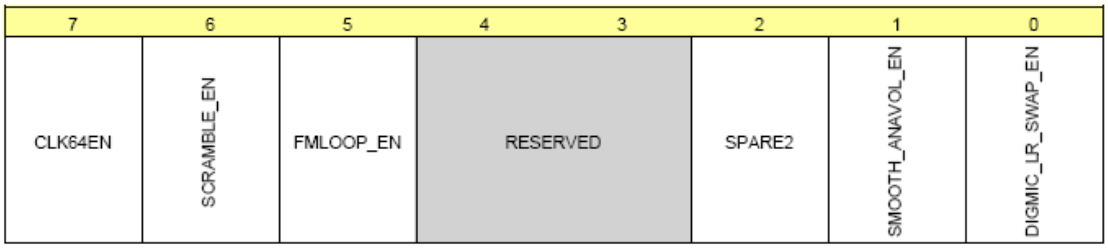
4.3.2.60 DTMF_PGA_CTL1 寄存器

这个寄存器与 DTMF 相关，暂时不考虑。



4.3.2.61 MISC_SET_1 寄存器

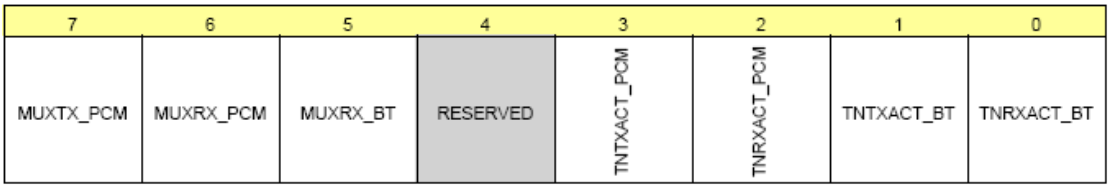
这个寄存器控制各种选项。



位	名称	功能说明	类型	复位值
7	CLK64_EN	64KHz 时钟使能控制位 0x0：禁止 0x1：使能	RW	0x0
6	SCRAMBLE_EN	杂散功能使能控制位 0x0：禁止 0x1：使能	RW	0x0
5	FMLOOP_EN	模拟 FM 收音机回路使能控制位 0x0：禁止 0x1：使能	RW	0x0
4:3	保留位		RW	0x0
2	SPARE2	spare2 位	RW	0x0
1	SMOOTH_ANAVOL_EN	当模拟音量控制改变时平滑使能控制位 0x0：旁路模式，增益立即改变 0x1：模拟增益改变时破音消减，增益改变在信号过零或 25ms 延时后进行	RW	0x0
0	DIGMIC_LR_SWAP_EN	数字混频，左右声道交换使能位 0x0：左右声道不交换 0x1：左右声道交换	RW	0x0

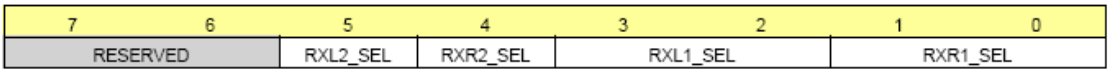
4.3.2.62 PCMBTMUX 寄存器

这个寄存器控制 PCM/BT 复用。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。



4.3.2.63 RX_PATH_SEL 寄存器

这个寄存器控制接收路径选择。



位	名称	功能说明	类型	复位值
7:6	保留位		RW	0x0
5	RXL2_SEL	输入 RXL2 PGA 选择控制位	RW	0x0

		0x0：SDRL2 到 RXL2 PGA 0x1:SDRM2(混合 SDRL2 和 SDRR2)到 RXL2 PGA		
4	RXR2_SEL	输入 RXR2 PGA 选择控制位 0x0：SDRR2 到 RXR2 PGA 0x1:SDRM2(混合 SDRL2 和 SDRR2)到 RXR2 PGA	RW	0x0
3:2	RXL1_SEL	输入 RXL1 PGA 选择控制位 0x0：SDRL1 到 RXL1 PGA 0x1:SDRM1(混合 SDRL1 和 SDRR1)到 RXL1 PGA 0x2：SDRL2 到 RXL1 PGA 0x3:SDRM2(混合 SDRL2 和 SDRR2)到 RXL1 PGA	RW	0x0
1:0	RXR1_SEL	输入 RXR1 PGA 选择控制位 0x0：SDRR1 到 RXR1 PGA 0x1:SDRM1(混合 SDRL1 和 SDRR1)到 RXR1 PGA 0x2：SDRR2 到 RXR1 PGA 0x3:SDRM2(混合 SDRL2 和 SDRR2)到 RXR1 PGA	RW	0x0

4.3.2.64 VDL_APGA_CTL 寄存器

这个寄存器控制语音下行模拟 PGA。这个寄存器只与语音也就是 PCM 有关，这里不考虑，取其默认值即可。

7	6	5	4	3	2	1	0
VDL_GAIN_SET					VDL_FM_EN	VDL_DA_EN	VDL_PDZ

4.3.2.65 VIBRA_CTL 寄存器

这个寄存器控制震动电机 H 桥。

7	6	5	4	3	2	1	0
Reserved	SPARE	VIBRA_DIR_SEL	VIBRA_SEL	VIBRA_AUDIO_SEL		VIBRA_DIR	VIBRA_EN

位	名称	功能说明	类型	复位值
7	保留位		RW	0x0
6	SPARE	spare 位	RW	0x0
5	VIBRA_DIR_SEL	H 桥方向选择位(仅在音频数据驱动 H 桥时有效) 0x0：VIBRA_DIR 决定方向 0x1：音频数据最高位决定方向	RW	0x0

4	VIBRA_SEL	H 桥驱动选择位 0x0：本地震动电机驱动器 0x1：音频数据	RW	0x0
3:2	VIBRA_AUDIO_SEL	音频通道选择控制位 0x0：左 1 0x1：右 1 0x2：左 2 0x3：右 2	RW	0x0
1	VIBRA_DIR	H 桥方向控制位 0x0：正极 0x1：负极	RW	0x0
0	VIBRA_EN	H 桥电源控制位 0x0：H 桥关闭 0x1：H 桥打开	RW	0x0

4.3.2.66 VIBRA_SET 寄存器

这个寄存器控制震动电机 H 桥 PWM 发生器。

7	6	5	4	3	2	1	0
VIBRA_AVG_VAL							

位	名称	功能说明	类型	复位值
7:0	VIBRA_AVG_VAL	设置 H 桥 PWM 发生器开启值，平均值是本寄存器值取反，最小值为 0x1，0x0 不允许	RW	0x00

4.3.2.67 ANAMIC_GAIN 寄存器

这个寄存器控制麦克风放大器增益。

7	6	5	4	3	2	1	0
RESERVED		MICAMPR_GAIN			MICAMPL_GAIN		

位	名称	功能说明	类型	复位值
7:6	保留位		RW	0x0
5:3	MICAMPR_GAIN	如果 ALC 禁止：右麦克风放大器增益设置 如果副麦克风 ALC 使能：右麦克风放大器增益设置最大值 0x0：0dB 0x1：6dB 0x2：12dB 0x3：18dB 0x4：24dB 0x5：30dB	RW	0x0
2:0	MICAMPL_GAIN	如果 ALC 禁止：左麦克风放大器增益设置 如果主麦克风 ALC 使能：左麦克风放大器增益设置最大值 0x0：0dB 0x1：6dB	RW	0x0

		0x2：12dB 0x3：18dB 0x4：24dB 0x5：30dB		

4.3.2.68 MISC_SET_2 寄存器

这个寄存器控制各种选项。

7	6	5	4	3	2	1	0
SPARE2	VTX_3RD_HPF_BYP	ATX_HPF_BYP	VRX_3RD_HPF_BYP	ARX_HPF_BYP	VTX_HPF_BYP	VRX_HPF_BYP	SPARE1

位	名称	功能说明	类型	复位值
7	SPARE2	spare2 位	RW	0x0
6	VTX_3RD_HPF_BYP	语音 TX 三阶高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式	RW	0x0
5	ATX_HPF_BYP	音频 TX 高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式	RW	0x0
4	VRX_3RD_HPF_BYP	语音 RX 三阶高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式	RW	0x0
3	ARX_HPF_BYP	音频 RX 高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式	RW	0x0
2	VTX_HPF_BYP	语音 TX 高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式		
1	VRX_HPF_BYP	语音 RX 高通滤波器模式控制位 0x0：应用模式 0x1：旁路模式	RW	0x0
0	SPARE1	spare1 位	RW	0x0

4.3.3 Mini 板与 beagleboard 在 CODEC 接口上的区别

下面考虑 Mini 与 beagleboard 在 CODEC 接口上的区别，以确定需要软件上更改哪一部分。通过电路原理图可知，对于 Mini 板，模拟输入接口是主麦克风 MIC1，模拟输出是预驱动放大器立体声输出 PreDriver；对于 beagleboard，模拟输入是辅助输入左右声道 AUX，模拟输出是耳机立体声输出 HS。因此软件上需要更改相应的配置部分。实际上 beagleboard 与 EVM 板在 CODEC 接口上是一致

的。

4.3.4 修改配置文件

在前面修改 Makefile 的时候可以发现，在这个版本下 beagleboard 的 CODEC 部分是没有编译进去，这里需要在配置文件中加入音频相关的宏定义，将 Mini 板的 CODEC 部分编译进去。

这里需要注意的是在内核源代码中看不到 TPS65950 的内容，主要是用 TWL4030 表示的，二者管脚和寄存器兼容，可以认为完全相同。

4.3.4.1 omap3_mini_defconfig 文件

由于 TPS65930 是不同于其它芯片，因此可以在默认配置文件中加入这个芯片的宏定义，便于在修改软件代码时用到。

修改文件的方法和前述相同，主要是在第 765 行回车空出一行，键入

```
CONFIG_TPS65930=y
```

同时，因为这个版本的 beagleboard 没有配置音频相关选项，因此还要将音频有关的选项加上，这部分主要参考 EVM 板和 timll 的 devkit8000 的配置文件，

将原第 193 行改为

```
CONFIG_OMAP_MCBSP=y
```

将第 847 行更改为

```
CONFIG_SOUND=y
```

并在后续行键入

```
CONFIG_SND=y
CONFIG_SND_TIMER=y
CONFIG_SND_PCM=y
CONFIG_SND_HWDEP=y
CONFIG_SND_RAWMIDI=y
CONFIG_SND_DYNAMIC_MINORS=y
CONFIG_SND_SUPPORT_OLD_API=y
CONFIG_SND_VERBOSE_PROCFS=y
CONFIG_SND_DRIVERS=y
CONFIG_SND_ARM=y
CONFIG_SND_SPI=y
CONFIG_SND_USB=y
CONFIG_SND_USB_AUDIO=y
CONFIG_SND_SOC=y
CONFIG_SND_OMAP_SOC=y
```

```
CONFIG_SND_OMAP_SOC_MCBSP=y
CONFIG_SND_OMAP_SOC_OMAP3_MINI=y
CONFIG_SND_SOC_I2C_AND_SPI=y
CONFIG_SND_SOC_TWL4030=y
```

这些配置项可能有一些是没有用的。

4.3.4.2 Kconfig 文件

为了可以在手动配置时可选 TPS65930 选项，在/linux/arch/arm/mach-omap2/目录下的 Kconfig 需要进行修改，在第 164 行回车空出一行，键入

```
menu    "TPS65930 selection for OMAP3 Mini"
config TPS65930
        bool "PMU for OMAP3 Mini"
        depends on I2C=y && ARCH_OMAP34XX
endmenu
```

这样在执行 make menuconfig 手动配置文件时就可以在 Mini 板下的一条菜单中看到已经配置了 TPS65930。

4.3.5 修改源代码文件和 Makefile

原代码文件的修改主要是在/linux/sound 目录下进行的，主要是对应器件 TWL4030 的相关文件，TPS65930 相当于它的一个子集，因此可以在此基础上稍加改动。

4.3.5.1 omap3mini.c 文件

在对比了 beagleboard 和 EVM 板的 CODEC 相关的代码文件后，发现 EVM 板的代码在 beagleboard 的基础上作了一些小改动，因此考虑用 EVM 板的程序代替 beagleboard 的程序。因此将/linux/sound/soc/omap/目录下的 omap3evm.c 文件修改成 omap3mini.c 文件，修改方法同前。

4.3.5.2 twl4030.c 文件

这个文件在/linux/sound/soc/codecs 目录下，就是 TWL4030 即 TPS65950 的配置驱动程序文件，通过修改这个文件使系统能够支持 TPS65930。

实际上在 CODEC 部分由于 TPS65930 的功能相对于 TPS65950 来讲少了很多，这个文件的改动量是不小的，不过由于将来不会使用 TPS65930 作为产品上的 CODEC，所以也没必要过于细致，这里只考虑初始化配置的相关改动，其它如 PCM 接口部分、多余的模拟输入输出接口部分，这里暂时先不考虑。实际上

在真正的手机产品中，也不是都用 TWL4030 或 TWL5030（TWL4030 的升级版）作为 CODEC，也有一些还是单独使用了一个 CODEC，如 Palm Pre 和 Motorola 的 Droid 使用 TWL5030 作为 CODEC，Nokia 的 N900 使用一片 TI 单独的 CODEC 芯片 TAIC34I。

拷贝文件第 40 行～119 行，也就是数组 twl4030_reg 的内容，这个是在初始化时 CODEC 的默认配置。在第 40 行回车空出一行键入

```
#ifndef CONFIG_TPS65930
```

然后在第 121 行键入

```
#else
```

在第 122 行粘贴拷贝内容，同时修改如下

```
/*
 * tps65930 register cache & default register settings
 */
static const u8 twl4030_reg[TWL4030_CACHEREGNUM] = {
	0x00, /* this register not used		*/
	0x91, /* REG_CODEC_MODE		(0x1)	*/
	0xc1, /* REG_OPTION		(0x2)	*/
	0x00, /* REG_UNKNOWN		(0x3)	*/
	0x01, /* REG_MICBIAS_CTL	(0x4)	*/
	0x31, /* REG_ANAMICL		(0x5)	*/
	0x00, /* REG_ANAMICR		(0x6)	*/
	0x08, /* REG_AVADC_CTL		(0x7)	*/
	0x01, /* REG_ADCMICSEL		(0x8)	*/
	0x00, /* REG_DIGMIXING		(0x9)	*/
	0x0c, /* REG_ATXL1PGA		(0xA)	*/
	0x0c, /* REG_ATXR1PGA		(0xB)	*/
	0x00, /* REG_AVTXL2PGA		(0xC)	*/
	0x00, /* REG_AVTXR2PGA		(0xD)	*/
	0x01, /* REG_AUDIO_IF		(0xE)	*/
	0x00, /* REG_VOICE_IF		(0xF)	*/
	0x00, /* REG_ARXR1PGA		(0x10)	*/
	0x00, /* REG_ARXL1PGA		(0x11)	*/
	0x6c, /* REG_ARXR2PGA		(0x12)	*/
	0x6c, /* REG_ARXL2PGA		(0x13)	*/
	0x00, /* REG_VRXPGA		(0x14)	*/
	0x00, /* REG_VSTPGA		(0x15)	*/
	0x00, /* REG_VRX2ARXPGA		(0x16)	*/
	0x0c, /* REG_AVDAC_CTL		(0x17)	*/
	0x00, /* REG_ARX2VTXPGA		(0x18)	*/
```

```
    0x00, /* REG_ARXL1_APGA_CTL    (0x19)    */
    0x00, /* REG_ARXR1_APGA_CTL    (0x1A)    */
    0x4b, /* REG_ARXL2_APGA_CTL    (0x1B)    */
    0x4b, /* REG_ARXR2_APGA_CTL    (0x1C)    */
    0x00, /* REG_ATX2ARXPGA        (0x1D)    */
    0x00, /* REG_BT_IF         (0x1E)    */
    0x00, /* REG_BTPGA         (0x1F)    */
    0x00, /* REG_BTSTPGA           (0x20)    */
    0x00, /* REG_EAR_CTL           (0x21)    */
    0x00, /* REG_HS_SEL        (0x22)    */
    0x00, /* REG_HS_GAIN_SET       (0x23)    */
    0x00, /* REG_HS_POPN_SET       (0x24)    */
    0x24, /* REG_PREDL_CTL         (0x25)    */
    0x24, /* REG_PREDR_CTL         (0x26)    */
    0x00, /* REG_PRECKL_CTL            (0x27)    */
    0x00, /* REG_PRECKR_CTL            (0x28)    */
    0x00, /* REG_HFL_CTL           (0x29)    */
    0x00, /* REG_HFR_CTL           (0x2A)    */
    0x00, /* REG_ALC_CTL           (0x2B)    */
    0x00, /* REG_ALC_SET1          (0x2C)    */
    0x00, /* REG_ALC_SET2          (0x2D)    */
    0x00, /* REG_BOOST_CTL         (0x2E)    */
    0x00, /* REG_SOFTVOL_CTL       (0x2F)    */
    0x00, /* REG_DTMF_FREQSEL  (0x30)    */
    0x00, /* REG_DTMF_TONEXT1H     (0x31)    */
    0x00, /* REG_DTMF_TONEXT1L     (0x32)    */
    0x00, /* REG_DTMF_TONEXT2H     (0x33)    */
    0x00, /* REG_DTMF_TONEXT2L     (0x34)    */
    0x00, /* REG_DTMF_TONOFF   (0x35)    */
    0x00, /* REG_DTMF_WANONOFF     (0x36)    */
    0x00, /* REG_I2S_RX_SCRAMBLE_H     (0x37)    */
    0x00, /* REG_I2S_RX_SCRAMBLE_M     (0x38)    */
    0x00, /* REG_I2S_RX_SCRAMBLE_L     (0x39)    */
    0x16, /* REG_APLL_CTL          (0x3A)    */
    0x00, /* REG_DTMF_CTL          (0x3B)    */
    0x00, /* REG_DTMF_PGA_CTL2     (0x3C)    */
    0x00, /* REG_DTMF_PGA_CTL1     (0x3D)    */
    0x00, /* REG_MISC_SET_1        (0x3E)    */
    0x00, /* REG_PCMBTMUX          (0x3F)    */
    0x00, /* not used          (0x40)    */
    0x00, /* not used          (0x41)    */
    0x00, /* not used          (0x42)    */
    0x00, /* REG_RX_PATH_SEL       (0x43)    */
    0x00, /* REG_VDL_APGA_CTL  (0x44)    */
```

```
        0x00, /* REG_VIBRA_CTL        (0x45)    */
        0x00, /* REG_VIBRA_SET        (0x46)    */
        0x00, /* REG_VIBRA_PWM_SET    (0x47)    */
        0x00, /* REG_ANAMIC_GAIN      (0x48)    */
        0x00, /* REG_MISC_SET_2       (0x49)    */
        0x00, /* REG_SW_SHADOW        (0x4A)    - Shadow, non HW register */
};
```

这里改动的主要是将耳机输出 HS 改为预驱动输出 PreDriver，同时原来的配置里没有将模拟输入打开，这里将 MIC1 通道打开，可能会稍微增加一些功耗。

在随后一行键入

```
#endif
```

另外将函数 static int twl4030_init(struct snd_soc_device *socdev)中的

```
hs_pop = twl4030_read_reg_cache(codec, TWL4030_REG_HS_POPN_SET);
hs_pop &= ~TWL4030_RAMP_DELAY;
hs_pop |= (setup->ramp_delay_value << 2);
twl4030_write_reg_cache(codec, TWL4030_REG_HS_POPN_SET, hs_pop);
```

这四句前后分别加入

```
#ifndef CONFIG_TPS65930
```

和

```
#endif
```

4.3.5.3 Makefile 文件

在/linux/sound/soc/omap/目录下的 Makefile 文件需要修改，在文件的第 16 和 24 行分别键入

```
snd-soc-omap3mini-objs := omap3mini.o
```

和

```
obj-$(CONFIG_SND_OMAP_SOC_OMAP3_MINI) += snd-soc-omap3mini.o
```

重新编译程序生成 uImage 映像文件，替换 MMC 卡中的映像文件，给 Mini 板加电，在内核启动过程中通过串口可以看到 ALSA（Advanced Linux Sound Architecture）驱动已经安装，ALSA 设备也已安装。不过这个 CODEC 的驱动很不完善，有很多功能可能无法实现，而且这里也没有试是否能出声。

Mini 板 CODEC 驱动程序修改暂时先到这里。

5. SBC8100 开发系统移植说明

SBC8100 开发系统是由国内的 Timll 公司生产的 OMAP3530 开发板，其外

设比较丰富，应该与本公司产品比较接近，下面考虑将将 linux 移植到这块板上。

5.1 SBC8100 开发系统简要说明

SBC8100 开发系统在硬件上主要由三部分组成：Mini8100 核心板、SBC8100 扩展板和 LCD 显示模块。下图为 SBC8100 开发系统组成框图。

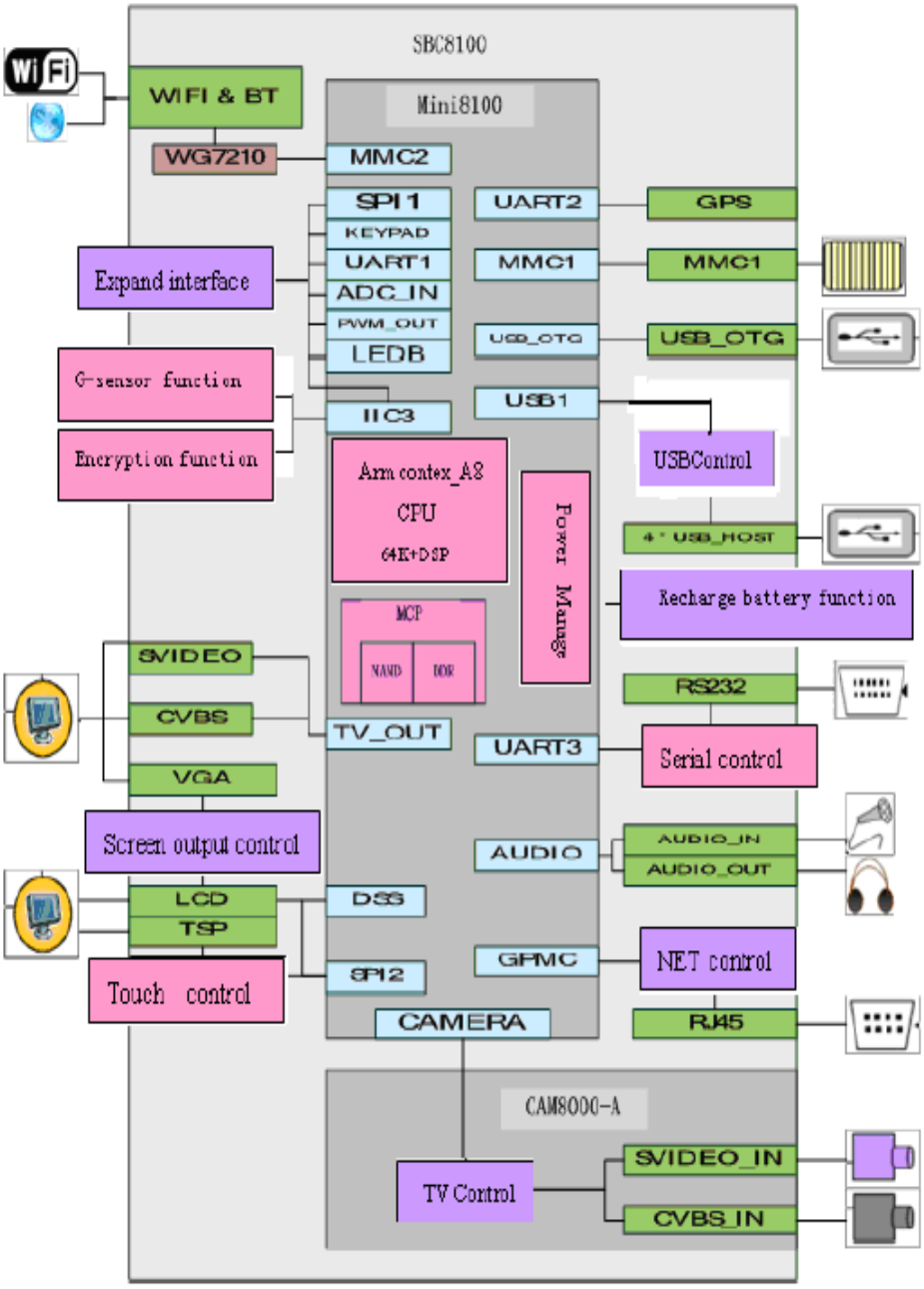


图 18 SBC8100 开发系统组成框图

下图为 Mini8100 核心板实物图，存储器部分 MCP 为 NAND FLASH 128MB 和 LPDDR SDRAM 128MB。

图 19 Mini8100 核心板实物图

下图为 SBC8100 扩展板实物图，图上可以看见 Mini8100 核心板。

图 20 SBC8100 扩展板实物图

下图为 LCD 显示模块实物图。显示模块可安装在 SBC8100 扩展板上，本文使用的是 4.3 吋屏，型号为 CHILIN 的 LR043JC211。

图 21 LCD 显示模块实物图

5.2 SBC8100 开发系统 linux 内核移植

这里与 Mini 板的移植方式大致相同，这里列出需要增加或修改的文件，

1) /linux/arch/arm/tools/mach-types；
2) /linux/arch/arm/configs/omap3_sbc8100_defconfig；
3) /linux/arch/arm/plat-omap/include/mach/board-omap3sbc8100.h；
4) /linux/arch/arm/plat-omap/include/mach/hardware.h；
5) /linux/arch/arm/mach-omap2/Kconfig；
6) /linux/arch/arm/mach-omap2/Makefile；
7) /linux/arch/arm/mach-omap2/board-omap3sbc8100.c；
8) /linux/drivers/video/omap/Makefile；
9) /linux/drivers/video/omap/lcd_omap3sbc8100.c；
10) /linux/drivers/video/omap2/Kconfig；
11) /linux/drivers/video/omap2/Makefile；
12) /linux/drivers/video/omap2/panel-lr043jc211.c；
13) /linux/drivers/net/dm9000.c；
14) /linux/sound/soc/omap/Kconfig；
15) /linux/sound/soc/omap/Makefile；
16) /linux/sound/soc/omap/omap3sbc8100.c；

Kconfig 文件和 Makefile 文件的改动与 Mini 板类似，这里就不再说明了。omap3_sbc8100_defconfig 文件暂时也不做说明。/sound 内文件的改动与 Mini 板基本一样（因这部分电路二者一致），只是改个名字，这里也不再说明。几个源代码文件的修改参考了网上同样是 timll 的 DEVKIT8000 的 patch 文件。DEVKIT8000 同样是基于 OMAP3530CUS 的开发板，它的功能与 SBC8100 相比减少了一些如 WIFI-BT 模块、GPS 模块等，但其它外设与 DEVKIT8000 基本一致（但有差别如 SBC8100 是 VGA 接口而 DEVKIT8000 是 DVI 接口，GPIO 上也有区别），因此可作为参考。对于显示部分，这里只考虑 4.3 吋屏时的情况，其余不考虑。

5.2.1 mach-types 文件

在文件最后一行加入

```
omap3_sbc8100          MACH_OMAP3_SBC8100          OMAP3_SBC8100          2726
```

这个数字是在网上看到似乎 SBC8100 板申请了正式编号是这个数字。这个值同样要在其 u-boot 的头文件内定义并编译。

5.2.2 board-omap3sbc8100.c

这里主要参考 DEVKIT8000 的文件，但做了一些改动，将少量没用的或错误的代码删除或修改。

5.2.3 lcd_omap3sbc8100.c

这里主要参考 DEVKIT8000 的文件，但做了一些改动。这里使用的是 4.3 吋的屏，因此只考虑了 4.3 吋屏的情况，没有考虑其它屏或 VGA 接口的情况。

5.2.4 panel-lr043jc211.c

这个是显示屏的驱动函数，主要参考了 OMAP3530EVM 板的 panel-sharp-ls037v7dw01.c 函数，只是配置和时序参数上有少量修改。

5.2.5 dm9000.c

这里主要参考 DEVKIT8000 的文件，但做了少量改动，主要是将宏 CONFIG_MACH_OMAP3_SBC8100 加入，以使改动仅针对 SBC8100 板，这样不影响其它板工作。

5.3 运行

修改完上面的各文件后，对系统进行编译，在编译过程中 USB 相关的部分出现了错误，因这里暂时先不使用 USB，因此将相关的配置选项去掉了，再次编译通过。在加载内核前首先还要将对应的 u-boot 文件修改，以使二者的板号能对应上，这里对 u-boot 的修改不再说明，与 Mini 板类似。修改编译完 u-boot 后，更新 MMC 卡内的 u-boot.bin 文件。按住 SBC8100 板的启动选择按钮后，给板加电，可以通过串口看到系统通过 MMC 卡加载 bootloader，进入 u-boot 后按任意键停止其运行，这里考虑使用 tftp 将内核文件加载到 SDRAM 中，避免每次写 MMC 卡或 FLASH。运行 tftp 需要先在 u-boot 下设置好网络，本文是这样的，

```
SBC8100 # printenv
bootdelay=3
baudrate=115200
bootfile="uImage"
splashimage=80000000
```

```
lcdtype=4.3inch_LCD
ethaddr=00:0e:99:01:83:70
bootcmd=mmcinit;fatload mmc 0 0x80300000 uImage; fatload mmc 0 0x81600000 ramdisk.gz; bootm 0x80300000
filesize=B00000
fileaddr=80000000
gatewayip=192.168.0.2
netmask=255.255.255.0
ipaddr=192.168.0.27
serverip=192.168.0.26
bootargs=console=ttyS2,115200n8 ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs
```

tftp 的设置这里不再说明，将 uImage 文件下下载到 SDRAM 中（下载时可能需要关闭防火墙），启动。

```
SBC8100 # tftpboot 0x80000000 uImage
dm9000 i/o: 0x2c000000, id: 0x90000a46
MAC: 00:0e:99:01:83:70
operating at 100M full duplex mode
TFTP from server 192.168.0.26; our IP address is 192.168.0.27
Filename 'uImage'.
Load address: 0x80000000
Loading: #################################################################
         #################################################################
         ##########################
done
Bytes transferred = 2268248 (229c58 hex)
SBC8100 # bootm 0x80000000
```

内核开始运行，中间似乎报了一些问题，这以后再说，但运行到最后停了，发现原来没有文件系统，于是安装根文件系统，这里使用原厂提供的根文件系统。文件系统映像烧在 NAND FLASH 内，操作方法见 SBC8100 的用户手册。烧完再次启动系统并加载内核运行，这次可以进入根文件系统，小企鹅看起来似乎也正常，屏底色为全黑，电流大约为 420mA（电压 5.4V，这里使用直流稳压电源，可显示电流，因担心系统不正常烧片子，这里电压稍高可能是因为自制的电源线较细长造成压降）。在串口没有输入一定时间后，屏会关闭以节省功耗（可能处理器内部也会有部分模块停止工作），此时电流大约为 220mA。

5.3.1 触摸屏测试

这里测试触摸屏时出现了很奇怪的现象，显示 Segmentation fault，考虑是没有设置环境变量，设置如下。

```
export TSLIB_CONFFILE=/etc/ts.conf
export TSLIB_CALIBFILE=/etc/pointercal
export TSLIB_PLUGINDIR=/lib/ts
export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export LD_LIBRARY_PATH=/lib
```

还是不行，后来考虑重新编译一次触摸屏的测试程序。网上搜索原来使用的是触摸屏的一个库 tslib，下载了一个比较新的版本 tslib-1.4（似乎还有个版本号 kergoth-tslib-1.0-0，这二者差别不大，似乎 kergoth-tslib-1.0-0 更新），根据网上的一些说明开始编译，使用的交叉编译器与内核编译的是一样的，在编译过程中出现一个问题就是说 rpl_malloc 没定义，后参照网上文章《Getting Started with Qt》的说明更改 config.h 文件，编译通过。将编译好的文件（这里的 ts.conf 文件似乎还要加入一行 module_raw input）逐一替换原来根文件系统中触摸屏的相关文件，重新制作根文件系统并烧到 NAND FLASH 内。启动内核，按上面的操作发现还是不行，又在网上找，后来发现设置环境变量后需要先键入 ts_print_raw 和 ts_print，等待一段时间后按 CTL+c 退出，然后再键入 ts_calibrate 和 ts_test，就可以工作了。校准和测试基本正常。因此可能不需要更新根文件系统 tslib 也能工作，不过这里没有再试。

这里还有问题，实际上内核启动时环境变量都已经设置好了（在/etc/profile 内），因此在原来版本（即 2.6.22）不需要重新设置环境变量就可测试（原来版本似乎也不需要输入 ts_print_raw 和 ts_print），可是这里如果不重新设置就会出问题，这里原因不清楚。并且这里输入 ts_print_raw 和 ts_print 时还是经常出现 Segmentation fault，几个命令来回输入几次就可用，原因不明。

在这里还试验了一下屏幕旋转的功能，这个版本的 linux 提供屏幕旋转的功能，只需在 u-boot 下的设置中加入 omapfb.rotate=1 omapfb.rotate_type=1 即可，在本文中为

```
setenv bootargs console=ttyS2,115200n8 ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs omapfb.rotate=1 omapfb.rotate_type=1
```

启动内核后可以看到小企鹅变成竖着显示。

5.4 问题查找

在内核启动过程中出现了一些问题提示，下面考虑逐个解决这些问题（这里

暂时不考虑 USB 相关的问题，因为 USB 过于复杂）。

1) Ignoring unrecognised tag 0x54410008；

这个应该与 u-boot 有关，在 u-boot 启动内核时的参数传递中有一项是 ATAG_VIDEOLFB，应该是与显示有关的选项，后查网上似乎这一项对于 ARM 很少用，已经删掉了。因此这个提示不影响系统工作，不过看着不舒服。去掉有两种方式，一种是在 u-boot 中不传递 ATAG_VIDEOLFB 参数；另一种是在内核的 setup.c 内对传递过来的 ATAG_VIDEOLFB 参数进行处理。这里用比较简单的方法，即在 u-boot 中稍微修改。在 sbc8100.h 内定义一个宏

```
#define CONFIG_VIDEOLFB_NO_SETUP
```

在/lib_arm/bootm.c 的函数 static void setup_videolfb_tag (gd_t *gd)内加上宏定义即可（即函数内的部分不运行），即

```
#ifndef CONFIG_VIDEOLFB_NO_SETUP
```

和

```
#endif
```

编译 u-boot 后写入 MMC 卡，重新启动即可，这个提示再不会出现。

2) dpll3_m2_clk rate change failed: -22；

这个问题是在配置 SDRAM 参数时出现的，在/arch/arm/mach-omap2/io.c 的函数_omap2_init_reprogram_sdrc 内，这个函数重新设置了一次 dpll3_m2_clk（也就是 SDRAM 的速率，本文这里是 266000000 Hz，即 266MHz，因为是 LPDDR SDRAM，因此其时钟是 133 MHz），但是在其内部函数

```
v = clk_set_rate(dpll3_m2_ck, clk_get_rate(dpll3_m2_ck));
```

出现了问题，参数没有重新设置。这里对查找问题过程不再详细说明，只说明出问题的顺序向后依次为

/arch/arm/plat-omap/clock.c 内函数 clk_set_rate 内的

```
ret = arch_clock->clk_set_rate(clk, rate);
```

/arch/arm/mach-omap2/clock.c 内函数 omap2_clk_set_rate 内的

```
ret = clk->set_rate(clk, rate);
```

/arch/arm/mach-omap2/clock34xx.c 内函数 omap3_core_dpll_m2_set_rate 内的

```
sp = omap2_sdrc_get_params(sdrcrate);
if (!sp)
```

```
        return -EINVAL;
```

在前面的计算中 sdrcrate 得到的值为 133000000，本文中使用的结构体 sp 对应的是 MICRON 的 SDRAM，它对应的结构体定义在/arch/arm/mach-omap2/sdram-micron-mt46h32m32lf-6.h 内的 mt46h32m32lf6_sdrc_params，这个结构体在目标板初始化 IRQ 的函数里的 omap2_init_common_hw 函数用到了，在这个结构体内有一个选项是 133333333 而不是 133000000，因此函数找不到对应参数的结构体，所以返回错误。

这里如果不考虑后续移植等问题，一个比较简单的办法就是在

```
sp = omap2_sdrc_get_params(sdrcrate);
```

前加入一句，

```
if (sdrcrate == 133000000)
        sdrcrate = 133333333;
```

也可更简单直接将头文件里的 133333333 改成 133000000，更改后不会再出现此问题。

这个问题主要是 SDRAM 时钟问题，对于目前的 OMAP3530 的开发板，其 SDRAM 的时钟基本都可以调整到 166M，SBC8100 也可以（已验证过），不过需要对 x-loader 或 u-boot 的代码进行相应的改动。

3) Driver 'sd' needs updating - please use bus_type methods；

这个似乎是提示信息，表示驱动程序可能有点老，这个应不影响工作。

4) clock: dpll5_ck failed transition to 'locked'；

这个是在对时钟 dpll5 配置时出的问题，在/arch/arm/mach-omap2/clock.c 内函数 omap2_clk_disable_unused 内的

```
omap2_clk_enable(clk);
```

出现问题，这个问题似乎是程序内有 bug，在后续版本中不再出现，这里对这个问题不再研究。

5) omap-dss DISPC error: Requested pixel clock not possible with the current OMAP2_DSS_MIN_FCK_PER_PCK setting. Turning the constraint off.；

这个原因是在 omap3_sbc8100_defconfig 文件中设置了一个配置项 CONFIG_OMAP2_DSS_MIN_FCK_PER_PCK=4，这是参照 EVM 板的，其含义是配置 FCK/PCK 的比例。FCK 最大频率 173MHz，如果这样配置即 PCK 最大不能超过 43 MHz。错误信息表示这个 PCK 太大，然后取消了这个限制项。将这

个配置项的值改成 2 即可。

6) Disabling Power mgmt : /etc/init.d/rcS: line 82: cannot create /sys/power/cpuidle_deepest_state: nonexistent directory；

这个应该是电源管理部分的程序可能有 bug，因为电源管理部分过于复杂，并且还在不断完善中，这里暂时先不考虑。

7) 有时出现 Spurious irq 95: 0xffffffdf, please flush posted write for irq 29；

这个似乎是程序内部有 bug，这里暂时先不查找。

8) 有时显示屏关闭一段时间后显示不出来，过一段时间后又能够显示。有时是显示屏关闭一段时间再打开后显示内容消失。

这个问题可能与内核的 PM 即电源管理部分有关，如果在串口输入

echo 1 > /sys/power/sleep_while_idle

命令，这个命令应该是禁止系统进入 idle 状态，因此小企鹅一直显示，这时显示内容不会消失，但这里无法完全确定。

9) 有时启动过程中出现 Kernel panic - not syncing: Attempted to kill init!，内核无法启动。

这个现象不常出现，这里暂时先不查找。

5.5 新版本移植

为了保持内核的版本功能更完善，考虑移植目前 TI 的 linuxPSP 的最新版本即 03.00.01.06，这个版本基于 2.6.32。同时使用较新版本的编译器，在网上下载，名称为 arm-2009q1-203-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2。这里不再说明移植过程，与前面的基本类似。只不过要修改的文件比上一版少几个，并且似乎 kconfig 的语法也有改变，子菜单方式有改变。移植后这里 USB host 的功能似乎正常，如果拿一个 U 盘插入 USB 口，系统能够认出 U 盘并可读取其中的内容。不过这里移植时显示部分出现的问题比较多，下面对影响比较大的问题说明。

1) 显示驱动不正常，串口显示

omapdss DSI error: can't get VDDS_DSI regulator
omapdss CORE error: Failed to initialize DSI

这是在使用 OMAP2 显示驱动时需要给相应电源管脚加电，这里的问题是由于 OMAP3530 封装不同及原理图不同造成的，EVM 使用的是 CBB 封装和

TPS65950，这里使用的是 CUS 封装和 TPS65930，通过查看新版 beagleboard 原理图（C4 或以上）可以看出在较新版本的 OMAP3530 中似乎对 VDDS_DSI 和 VDDS_SDI 管脚有单独的定义（在 sprs507f 中还没有单独定义，它们都归类在 vdds 管脚中），其中 VDDS_DSI 似乎在 DPI 显示时也是需要的。在新版 beagleboard（EVM 板应该是一样的）中这些管脚是用 TPS65950 的 vpll2 单独供电的，这个电源可控。但 TPS65930 没有 vpll2，并且在现有的 SBC8100 中这些已连接到 vio 上（基本上国内的 CUS 封装的 OMAP 开发板应该都是这么接的），因此需要配置一个 regulator 给显示驱动，在这里就使用 vio。不过查找源代码发现 vio 在这一版本中没有定义（注释掉了），因此需要在文件/driver/regulator/twl-regulator.c 中加入一些更改，同时在 board-omap3sbc8100.c 加入 vio 相关的结构体。

2) 显示屏不刷新，仅刷了一小块就停了。

经较长时间查找，发现犯了一个比较愚蠢的错误，在移植显示屏驱动部分 panel-lr043jc211.c 时，照抄了一个现成的显示屏的驱动然后改个名并改写时序参数部分结构体，不过忘了改写配置参数，本文的显示屏配置应为

```
dssdev->panel.config = OMAP_DSS_LCD_TFT | OMAP_DSS_LCD_IVS |
      OMAP_DSS_LCD_IHS | OMAP_DSS_LCD_IPC;
```

而原来的屏为

```
dssdev->panel.config = OMAP_DSS_LCD_TFT | OMAP_DSS_LCD_IVS |
      OMAP_DSS_LCD_IHS | OMAP_DSS_LCD_IEO;
```

这一点每个屏是不一样的，因此最好应定义一个与屏的名字相关的宏，便于移植。将此部分更改后显示屏能够显示。

5.5.1 根文件改写

这里考虑使用 PSP 中的根文件系统，同时加入 tslib，验证触摸屏功能。最开始使用的是 root-base.jffs2 文件，报了与图 11 相同的模块错误，因此还需要编译模块文件。

首先将 PSP 的根文件系统解压缩，该文件在 AM35x-OMAP35x-PSP-SDK-03.00.01.06/images/omap3530 内，名称为 nfs.tar.gz。本文解压缩到/tftpboot/rootfs 内。

5.5.1.1 编译模块文件

按照与前文相同步骤编译模块文件，生成一个/2.6.32-omap3（这个-omap3

是本人在 Makefile 里手欠加的，如果不加则为/2.6.32，最好不加，因为可能影响到其它如 DVSDK 或 Graphics SDK 的设置）的目录，将这个目录放入/rootfs 的/lib/modules 目录内，可以看到这个目录内原来的为 2.6.29 版本，可以把原来那个目录删除掉。

5.5.1.2 编译 tslib

这里简要说明编译 tslib 的过程。

首先从网上下载 tslib 压缩文件并解压缩，本文解压缩到/usr/local/src/tslib-1.0-0 内，确认已建立与编译器的路径后，在命令行键入

```
./autogen.sh
./configure --prefix=/lib/ts/ --host=arm-none-linux-gnueabi
```

这里的 prefix 是安装后的位置，可以随意。

这时编辑/tslib-1.0-0 内的 config.h 文件，大约在文件的最后将这句注释掉。

```
#define malloc rpl_malloc --> /* #define malloc rpl_malloc */
```

保存文件，然后在命令行分别键入

```
make
make install
```

如果没有问题，这时会在系统的/lib 目录下生成一个/ts 目录，将这个目录内的各文件放入/rootfs 的各目录下（参照原来 SBC8100 的根文件的位置）。

5.5.1.3 使用 jffs2 文件

原来的 SBC8100 根文件是 UBI 格式，在启动过程中出现一些错误，因不想考虑文件格式的影响，因此这里还是使用 EVM 的文件格式 jffs2。从网上下载一个 mkfs.jffs2 文件（因本文使用的 fedora6 本身不带这个文件），键入

```
mkfs.jffs2 -p -l -e 128 -n -v -r /tftpboot/rootfs/ -o filesys.jffs2
```

会生成一个 filesys.jffs2 文件，按照 SBC8100 使用手册的说明将根文件写入 NAND FLASH 中，这里与手册不同的是由于 board-omap3sbc8100.c 中 NAND FLASH 的配置稍有区别，因此写入的位置也有所区别。

在启动 u-boot 后，下载内核并启动，在根文件中运行到 thttpd 时停止了，这可能与网络配置有关，为了避免频繁擦除 NAND FLASH，这里考虑使用 NFS 根文件系统。

5.5.1.4 使用 NFS 文件

这里对主机的 NFS 服务器配置不再说明。在 u-boot 中首先更改启动配置参数，本文为，

setenv bootargs console=ttyS2,115200n8 mem=99M vram=12M omapfb.vram=0:12M noinitrd ip=192.168.0.27:192.168.0.26:192.168.0.254:255.255.255.0:omap3evm:eth0:off root=/dev/nfs rw nfsroot=192.168.0.26:/tftpboot/rootfs,nolock

这里应一句写完，这里没有安装 dhcp 服务器。这里的 99M 似乎是留一些给 DVSDK 用，这个留在后面研究。

下载内核并启动后，在运行根文件系统脚本时出现如下错误并在后面停住，

/etc/rcS.d/S03udev: line 50: can't create /tmp/uname: Read-only file system

后在网上查找，发现是主机的 NFS 服务器的配置问题。对于本文使用的 fedora6，是在/etc/exports 文件内改为

/tftpboot/rootfs *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)

以前是

/tftpboot/rootfs 192.168.x.* (rw,sync,no_root_squash)

这个文件在使用 OMAP5912OSK 时是有效的，改写文件后重新启动 NFS，再次下载内核并启动后不会出现上述问题并能够进入 login 界面。不过有时会出现一些错误信息。

这里还有一个问题就是对于 SBC8100 来说在根文件系统的/etc 目录下的 inittab 文件需要改动，将里面的 ttyS0 改成 ttyS2，即将调试串口改成 UART3（OMAP3530EVM 用的是 UART1），否则在 Starting thttpd 之后串口就不会再显示了（这个本人折腾半天，还以为是 thttpd 有问题，前面使用 jffs2 应该也是同样的问题）。

如果出现 Warning: unable to open an initial console.的问题，需要在根文件目录的/dev 目录内增加内容，在命令行中进入/dev 目录，然后键入

mknod –m 660 console c 5 1
mknod –m 660 null c 1 3

这里再次测试 tslib 时能够工作，似乎也不会出现 Segmentation fault 字样。不过屏幕上的一个小光标似乎去不掉。此时电流大约 580mA（电压 5.4V）。这里电流比上一个版本大不少（约 150mA）可能是因为 USB HOST 部分工作起来了。

如果在启动配置参数中加入 mpurate=600，则启动后可以看到重新配置了系统

时钟，ARM 时钟为 600M，DSP 时钟为 430M。此时电流大约 620mA（电压 5.4V）。

如果在启动配置参数中加入 mpurate=250，则启动后可以看到重新配置了系统时钟，ARM 时钟为 250M，DSP 时钟为 180M。此时电流大约 540mA（电压 5.4V）。

5.5.2 Graphics SDK 测试

下面考虑使用 TI 的 Graphics SDK。

5.5.2.1 配置 Graphics SDK

这里主要参照网上文章《OMAP35x Graphics SDK Getting Started Guide（SPRUFZ8）》和《AM35x and OMAP35x Rebuilding the Software》以及《Executing the Graphics SDK Demos》的一些说明。

从 TI 网站下载（需要注册）OMAP35x_Graphics_SDK_setuplinux_3_01_00_06.bin 文件。在命令行键入

```
chmod 777 OMAP35x_Graphics_SDK_setuplinux_3_01_00_06.bin
./OMAP35x_Graphics_SDK_setuplinux_3_01_00_06.bin
```

安装为图形界面，比较简单，本文安装在/opt/OMAP35x_Graphics_SDK_3_01_00_06 下（大约 1.2GB）。

安装完毕后进入 SDK 目录，修改 Rules.make，这个需要根据编译器、内核、内核模块和 NFS 根文件系统的位置手动改写。

5.5.2.2 编译并 build

进入 Graphics SDK 工作目录，本文就是/opt/OMAP35x_Graphics_SDK_3_01_00_06，建立编译器路径，键入

```
PATH=$PATH:/opt/codesourcery/arm-2009q1
export PATH
```

也可编辑一个文件用 source 命令或直接编辑一个脚本。然后键入

```
export ARCH=arm
```

然后编译，对于本文，这里先考虑使用 debug 版（SBC8100 的 OMAP 为 ES3.1 版本），

```
make all OMAPES=3.x BUILD=debug
```

然后安装到 NFS 文件系统中，

```
make install OMAPES=3.x
```

如果要重新编译 SGX 模块文件，需要（如果要重新编译的话似乎要放在前

面），

```
make buildkernel OMAPES=3.x BUILD=debug
```

SDK 里有 debug 和 release 两个版本，应根据需要编译。

如果正常，编译可以顺利完成。

5.5.2.3 运行

启动 u-boot 并下载内核，启动内核，可以看到在初始化过程中有 SGX 用户库文件的初始化，这中间似乎报了一些错，但能够进入 login 界面。

进入/usr/local/bin 目录，可以看到一些可执行文件，按照《Executing the Graphics SDK Demos》的说法，似乎运行 OpenGLES1.x 的 demo 时需要改日期，键入，

```
date -s 1970.01.01-01:01:01
```

然后可以运行程序，如

```
./gles1test1 –t
```

可以看到屏幕开始运行 demo，运行时开始似乎也报 errot，不过也能运行。不过不知是显示屏的原因还是其它，屏的色彩不是太好，而且最开始那个光标还在。并且这里有一些测试程序报错无法运行，似乎是 framebuffer 的设置有问题，可能还有其它问题，可能与内核有关，这里暂时先不考虑。

还可以进入/opt/gfxsdkdemos 目录下，再进入 ogles 或 ogles2，运行测试程序。不过这里在运行./OGLESChameleonMan 或./OGLES2Shaders 等测试程序时无法按 Ctrl+c 退出，只能断电，不过运行时屏幕显示似乎正常。运行./OGLESChameleonMan 时电流大约 640mA（电压 5.4V），可见显示加速器 SGX530 的功耗还可以，不过其频率可能只有 110MHz，在新版本中会是 200MHz，功耗可能会大一些。

5.5.3 DVSDK 使用

这部分考虑使用 DVSDK 的功能。

5.5.3.1 安装 DVSDK

首先需要安装 DVSDK，本文的版本为 dvsdk_3_01_00_10_Setup.bin，首先将 DVSDK 相关的文件下载。然后开始安装，这里主要参照 OMAP35x DVEVM Software Setup。

```
chmod +x *.bin
./dvsdk_3_01_00_10_Setup.bin
```

安装是图形界面，本文安装在/opt 内，在选择安装目录时就选择/opt，程序会安装在/opt/dvsdk_3_01_00_10 目录内。然后安装 Codec Server，

```
./cs1omap3530_setupLinux_1_01_00-prebuilt-dvsdk3.01.00.10.bin
```

这里在选择安装目录时选择/opt/dvsdk_3_01_00_10，程序会安装在/opt/dvsdk_3_01_00_10/cs1omap3530_1_01_00 目录内（不过这里有个问题，就是在安装时它的 uninstall 把原来的 DVSDK 的给替换了，造成 DVSDK 没有卸载程序，卸载时只能手动删除，但如果不安装在 DVSDK 目录内，它的一些路径设置还要改动），然后安装 Code Generation Tools，

```
./ti_cgt_c6000_6.1.12_setup_linux_x86.bin
```

这里在选择安装目录时选择/opt/dvsdk_3_01_00_10/C6000CGT6.1.12，程序会安装在/opt/dvsdk_3_01_00_10/ C6000CGT6.1.12 目录内，安装完毕。

然后在参照的文档里说明需要建立路径，不过这似乎应在编译时设置。对于本文就是，

```
export C6X_C_DIR=/opt/dvsdk_3_01_00_10/C6000CGT6.1.12/include:/opt/dvsdk_3_01_00_10/C6000CGT6.1.12/lib
```

如果需要安装 demo 文件，将 data_dvsdk_3_01_00_10.tar.gz 拷贝到 DVSDK 安装目录的/clips 目录内并解压缩（如果不安装 demo 文件，编译时可能会报错，可能需要修改相应的 Makefile 文件，demo 文件比较大，有几百兆，因此根文件系统目录应预留足够的空间，主要是 demo 文件占地方，Graphics SDK 的库文件也比较大，对于本文在编译完 Graphics SDK 和 DVSDK 后，/tftpboot/rootfs 目录大约为 650MB，而编译前仅为 25MB），

```
cd /opt/dvsdk_3_01_00_10/clips
tar -xvzf data_dvsdk_3_01_00_10.tar.gz
```

5.5.3.2 编译 DVSDK

要编译 DVSDK，首先需要修改/opt/dvsdk_3_01_00_10 内的 Rules.make 文件，这个文件设置了各方面的工作路径，因此不能错。这里需要注意的主要是几个可能需要更改的路径，包括 DVSDK_INSTALL_DIR、XDC_INSTALL_DIR、CODEC_INSTALL_DIR、UBOOT_INSTALL_DIR、LINUXKERNEL_INSTALL_DIR、CSTOOL_DIR、EXEC_DIR 等，这里简要说明。

1) DVSDK_INSTALL_DIR：DVSDK 的安装目录，这个目录名可能与文件中不同；
2) CODEC_INSTALL_DIR：Codec Server 的安装目录，这个目录名可能与文件中不同；
3) XDC_INSTALL_DIR：Code Generation Tools 的安装目录，这个目录名可能与文件中不同；
4) UBOOT_INSTALL_DIR：u-boot 的安装目录，这里不清楚 u-boot 在 DVSDK 中有什么作用，可能是 mkimage，这里暂时使用的是 SBC8100 自带的 u-boot。
5) LINUXKERNEL_INSTALL_DIR：内核源代码的目录；
6) CSTOOL_DIR：编译器的目录；
7) EXEC_DIR：NFS 根文件系统目录。

修改完 Rules.make 文件并保存，然后修改目录下的 Makefie，在 80、81 行分别修改内核和 u-boot 的配置文件名，对于本文就是 SBC8100 的配置文件，这两个配置文件应该是在编译 u-boot 和内核时用的，如果已经编译好，这里就不用了。

路径最好一次设置好，否则可能编译时报错，还无法恢复，还要卸载 DVSDK 再重装一次（本人就来回安装删除了好几遍），这里不太清楚原因。

建立编译器路径（还是 2009q1 版本的路径）后键入，

```
make check
make info
```

一般不会有什么问题，但如果路径设置有误，可能会有警告。

```
make clean
make all
make install
```

这里的 clean 或 all 包括 cmem、sdma、lpm、dmammapk、edmak、irqk、dmai、demos、dvtb、examples，但不包括 codecs、dsplink、ce_examples、psp_examples、linux 等。实际上按照 DVSDK 的设置，似乎连 u-boot、内核、DVSDK 等都能一块编译了，如果使用 everything 的话，但最好不要使用 everything，u-boot 和内核最好单独编译。

安装完在 NFS 根文件系统目录的/opt 目录内会有一个/dvsdk 的目录。

还可以直接编译 DSP 端 server，

```
make codecs_clean
make codecs
make install
```

这里都能够编译成功。不过这里没有编译 dsplink。

5.5.3.3 运行

启动 u-boot 并下载内核后，启动内核，进入 login 界面。

```
cd /opt/dvsdk/omap3530
./loadmodules.sh
```

可以看到 CMEMK module、DSPLINK Module 和 SDMAK module 似乎都正常。不过在运行 demo 测试程序时无法运行，报如下等问题，原因不清楚，可能还是因为 SBC8100 的外设与 OMAP3530EVM 有不同造成有些地方的设置不同，但这里暂时没有找到。

```
Error: Failed to create display device
Error: Failed to create Cpu Object
```

下面考虑花一点时间检查这个问题。可以在串口中使用 CE_DEBUG（2 或 3）来查看信息，如

```
CE_DEBUG=2 ./decode -v /opt/dvsdk/omap3530/data/videos/davincieffect_ntsc_1.264 -O lcd
```

可以看到大量信息，出问题的地方在

```
ti.sdo.dmai - [Display] Unknown video standard 0
ti.sdo.dmai - [Display] Could not allocate video display buffers
```

似乎是说设备不正常，这里有一个原因是没有设置输出设备，即在./decode 后需要加 -y 1，并且可能分辨率也不对，查找后发现在 /dvsdk_3_01_00_10/dmai_2_05_00_12/packages/ti/sdo/dmai/VideoStd.h 内对于显示设备的分辨率有唯一的定义，其中 VideoStd_VGA_WIDTH 和 VideoStd_VGA_HEIGHT 的定义是 640 和 480，如果不同可能会报故障。这里考虑将这两个值修改一下，本文的屏为 480*272。这里还有一个问题就是 OMSP3530EVM 板的屏是 480*640，因此使用横屏显示时在 u-boot 中的参数传递中需要设置 omapfb.rotate=1 omapfb.vrfb=y，而这里应该不需要。

此外在 /dvsdk_3_01_00_10/dmai_2_05_00_12/packages/ti/sdo/dmai/linux/Display.c 内似乎也需要修改显示分辨率，这里将文件中的 640 都改写成 272。重新按照上述编译，

```
make clean
make all
make install
```

这里 codecs 不需重新编译安装，因为只需要编译 dmai 的部分，重新启动内核，加载 DVSDK 模块后，

```
./loadmodules.sh
./decode -y 1 -v /opt/dvsdk/omap3530/data/videos/davincieffect_ntsc_1.264 -O lcd
```

在屏幕上开始显示动画，画面还比较精致也比较连续，串口会不停显示 ARM 和 DSP 的负载及帧数，可以看到 DSP 的负载还是较重的（不过本文的 DSP 时钟应为 360MHz，没有全速运行，最新版的 OMAP3530 的 DSP 可到 520 MHz）。此时电流大约 680mA（电压 5.4V），可见 DSP 运行时的功耗也不小，全速运行时功耗会更大。

不过这里很多时候不能成功，经常报错，可能是在./loadmodules.sh 内的内存池的分配有问题，也可能是其它，由于这只是 demo 程序，实际意义不是很大，因此这里不再考虑其它 demo 程序，需要在具体应用中使用时再详细研究。

基于 OMAP3530 的 linux 内核及部分功能模块的初步移植工作完成。