ATWILC1000/ATWILC3000

Wi-Fi®链接控制器 Linux®用户指南

简介

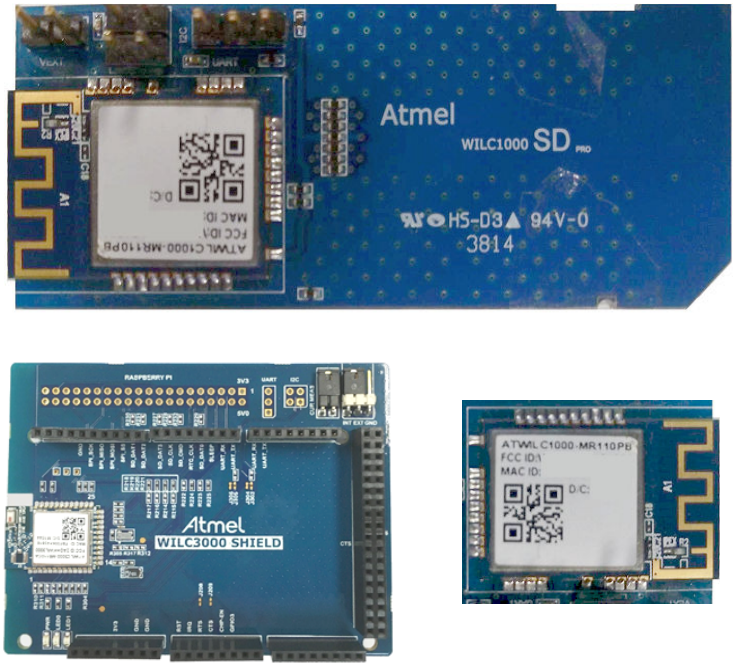
本用户指南介绍如何在 ATWILC1000 SD 卡或 ATWILC3000 Shield 板（安装在运行 Linux®内核 4.9 的 SAMA5D4 Xplained Ultra 上）上运行 Wi-Fi。

注： 除非另外说明，否则所有对 ATWILC 模块的引用均指下列两款器件：

- ATWILC1000
- ATWILC3000

源代码在 GitHub 上维护。要获取最新的源代码，请访问以下网站查看 GitHub Linux for ATWILC：https://github.com/linux4wilc。

图 1. ATWILC1000 SD 卡和 ATWILC3000 Shield 板



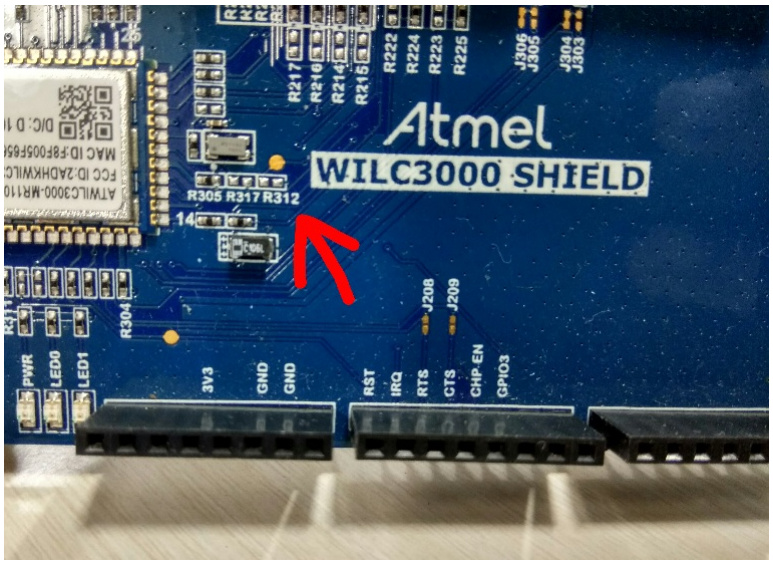
目录

1. 前提条件

编译 Linux 的前提条件是拥有一台运行 Linux 操作系统的主机 PC。硬件要求如下：

- Linux
  - SAMA5D4 Xplained Ultra
  - ATWILC1000 SD Pro 卡
  - ATWILC3000 Shield 板
  - USB 转串口适配器（用于 DEBUG 端口）
- 通用
  - Micro-USB 线缆（Micro-A/Micro-B）

为避免修改内核代码，应在 ATWILC3000 Shield 板上的相应位置（如下图所示）安装一个 120 kΩ 左右的电阻 R312。



2. 为 SAMA5D4 Xplained Ultra 板编译 Linux

本章介绍如何编译用于 ATWILC 器件演示的根文件系统和内核镜像。

本用户指南仅介绍关于 AT91Bootstrap 和 U-Boot 的一般信息，更多详细信息，请参见AT91 Smart Arm®单片机的 Linux 和开源代码相关信息的 U-Boot 部分。

2.1 克隆内核源代码和根文件系统

本演示使用 buildroot 来获取合适的工具链、根文件系统和 Linux 内核。

buildroot 从 linux4wilc github 克隆，地址如下：

```
$ git clone https://github.com/linux4wilc/buildroot4wilc.git
```

克隆的 buildroot 位于当前目录中的以下路径：

```
\buildroot4wilc
```

当前 buildroot4wilc 从 buildroot 的资源库（`git://git.buildroot.net/buildroot`，分支 2017_08）复制，然后通过 WILC 配置文件（`configs/sama5_wilc_defconfig`）以及其他帮助运行 WILC 示例的配置文件进行修改。

2.2 载入 SAMA5D4 配置文件

使用预定义的 `defconfig` 文件来创建所需的`.config` 配置文件。此 `defconfig` 文件位于 buildroot 文件夹 `buildroot4wilc` 下的 `configs` 文件夹中。

对于 SAMA5D4，使用的 defconfig 文件为 `sama5_wilc_defconfig`。

为 SAMA5D4（运行适用于 ATWILC 板的 Linux 内核 4.9）编译根文件系统，应浏览至文件的解压缩目录，然后使用以下命令创建**.config** 文件：

```
$ cd buildroot4wilc
$ make sama5_wilc_defconfig
```

2.3 Buildroot 文件系统和 Linux 内核

使用 `buildroot` 目录下的`$ make` 命令启动编译操作。

此`$ make` 命令会在终端上显示编译状态。
注： 请确保主机 PC 在启动编译操作之前已连接到 Internet，并且未使用任何编译选项。

编译操作完成后，将在 `buildroot/output/images` 目录下生成 `rootfs.ubi` 文件。对于默认编译操作，`rootfs.ubi` 文件中将包含 WILC 模块。

驱动程序源文件位于 `linux-at91` 内核中，地址如下：`https://github.com/linux4wilc/linux-at91/tree/master/drivers/staging/wilc1000`。
注： 由于历史原因，驱动程序的目录名称为 wilc1000，但实际上同时支持 ATWILC1000 和 ATWILC3000。

2.4 单独编译 Linux 内核

Buildroot 根据 GitHub 上的 buildroot 配置文件下载 Linux 内核。下载的内核必须位于 `buildroot4wilc/output/build/linux-xxxx` 路径下，并在 buildroot 编译操作期间自动编译。

但是，如果在编译 buildroot 之后修改了内核，则用户必须重新编译内核。以下是针对工具链和 Arm 架构编译 Linux 内核的步骤：

1. 使用以下命令将目录更改为 Linux 内核源文件夹：

```
$ cd output/build/linux-xx
```

2. 使用以下命令通过 sama5_defconfig defconfig 文件创建内核：

```
$ make ARCH=arm sama5_defconfig
```

3. 使用以下命令通过 menuconfig 工具执行所需更改：

```
$ make ARCH=arm menuconfig
```

4. 使用以下命令针对工具链和 Arm 架构编译 Linux 内核：

```
$ make ARCH=arm CROSS_COMPILE=../../../output/host/opt/ext-toolchain/bin/arm-linux-
gnueabihf-
$ make ARCH=arm CROSS_COMPILE=../../../output/host/opt/ext-toolchain/bin/arm-linux-
gnueabihf- zImage
$ make ARCH=arm CROSS_COMPILE=../../../output/host/opt/ext-toolchain/bin/arm-linux-
gnueabihf- dtbs
```

3. 为 SAMA5D2 Xplained Ultra 板编译 Linux

本章介绍如何编译用于 ATWILC 器件演示的 bootstrap、U-Boot、根文件系统（Root File System，RFS）和内核镜像。

3.1 克隆和编译二进制文件

本节详细介绍如何克隆和编译 AT91Bootstrap、U-Boot、内核和 RFS。

3.1.1 AT91Bootstrap

请按照以下步骤编译 AT91Bootstrap。

1. 从 github 克隆 AT91Bootstrap，地址如下：

```
$ git clone git://github.com/linux4sam/at91bootstrap.git
```

2. 下载 AT91Bootstrap 后，使用以下命令进入克隆的目录：

```
$ cd at91bootstrap/
```

3. 使用以下命令编译 bootstrap：
   假设用户位于 AT91Bootstrap 根目录下，其中有一个包含多个默认配置文件的 `board/sama5d2_xplained` 文件夹。AT91Bootstrap 已完成配置，并通过嵌入式多媒体控制器（embedded Multi-Media Controller，eMMC）载入 U-Boot 二进制文件。

```
$make mrproper
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sama5d2_xplainedemmc_uboot_defconfig
```

   注： 确保将工具链的路径导出至 PATH 环境变量。

   随即将在二进制文件夹下生成 `sama5d2_xplained-sdcardboot-uboot-3.8.12.bin` 二进制文件。

4. 要使引导 ROM 代码识别出 SD 卡或嵌入式多媒体控制器（eMMC）中的有效引导代码，需将 `sama5d2_xplained-sdcardboot-uboot-3.8.12.bin` AT91bootstrap 文件重命名为 `BOOT.bin`。

3.1.2 U-Boot

请按照以下步骤编译 u-boot。

注： 确保已在 Linux 计算机上安装 mkenvimage 工具。

1. 从 github 克隆 u-boot，地址如下：

```
$ git clone git://github.com/linux4sam/u-boot-at91.git
```

2. 下载 AT91Bootstrap 后，使用以下命令进入克隆的目录：

```
$ cd u-boot-at91
```

3. 使用以下命令切换到新分支 `u-boot-2018.07-at91`：

```
$git branch -r
$ git checkout origin/u-boot-2018.07-at91 -b u-boot-2018.07-at91
```

4. 使用以下命令将配置文件（`sama5d2_xplained_mmc_defconfig`）应用于 u-boot：

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sama5d2_xplained_mmc_defconfig
```

5. 打开 `u-boot-at91/include/configs/sama5d2_xplained.h` 文件，然后使用以下命令修改 `FAT_ENV_DEVICE_AND_PART` 和 `CONFIG_BOOTCOMMAND` 的定义：

```
/*bootstrap + u-boot + env in sd card */
#undef FAT_ENV_DEVICE_AND_PART
#undef CONFIG_BOOTCOMMAND
#define FAT_ENV_DEVICE_AND_PART  "0"
#define CONFIG_BOOTCOMMAND "fatload mmc 0:1 0x21000000 at91-sama5d2_xplained.dtb; " \
```

```
        "fatload mmc 0:1 0x22000000 zImage; " \
        "bootz 0x22000000 - 0x21000000"
#undef CONFIG_BOOTARGS
#define CONFIG_BOOTARGS \
        "console=ttyS0,115200 earlyprintk root=/dev/mmcblk0p2 rw rootwait"
```

6. 使用以下命令编译 u-boot 二进制文件：

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

随即将在根文件夹（u-boot-at91 文件夹）下编译 u-boot.bin 输出。
注： 确保交叉编译器工具链位于同一路径下。

7. 在某个目录（例如，包含 u-boot 环境变量的主文件夹）下创建一个文本文件 u-boot-env.txt，然后将以下内容复制到文件中：

```
bootargs=console=ttyS0,115200 root=/dev/mmcblk0p2 rw rootfstype=ext4 rootwait
bootcmd=fatload mmc 0:1 0x21000000 at91-sama5d2_xplained.dtb; fatload mmc 0:1 0x22000000
zImage; bootz 0x22000000 - 0x21000000
bootdelay=1
ethact=gmac0
stderr=serial
stdin=serial
stdout=serial
```

8. 移至主文件夹，然后输入以下命令以生成 uboot.env 文件。

```
$ mkenvimage -s 0x2000 -o uboot.env u-boot-env.txt
```

3.1.3 内核

请按照以下步骤编译内核。

1. 使用以下命令克隆资源库以获取源代码：

```
git clone git://github.com/linux4sam/linux-at91.git
```

2. 要使用其他分支，可通过以下命令列出这些分支并使用其中一个分支：

```
git branch -r
git checkout origin/linux-4.14-at91 -b linux-4.14-at91
```

3. 将 drivers/staging/wilc 中的 ATWILC1000 驱动程序替换为 ATWILC 驱动程序资源库中 driver/wilc 目录下的内容。资源库可从以下地址获取：https://github.com/linux4wilc。
输入以下命令以从 linux4wilc 中获取相应的文件：

```
git clone git://github.com/linux4wilc/driver
```

4. 在 linux-at91/drivers/staging/Makefile 中修改以下行，以便编译时能够找到正确的目录：

```
FROM: obj-$(CONFIG_WILC1000) += wilc1000/
TO: obj-$(CONFIG_WILC) += wilc1000/
```

5. 使用以下命令配置内核：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sama5_defconfig
```

6. 使用 menuconfig 修改默认配置。请按照以下步骤打开 menuconfig：
   - make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
   - 请按照以下步骤选择 ATWILC 驱动程序模块：
     1. 转至 menuconfig。
     2. 导航至 *Device Drivers* > *Staging driver*（设备驱动程序 > 分段驱动程序）。
     3. 按下“y”以包含分段驱动程序。
     4. 根据要求选择 Atmel WILC SDIO 或 Atmel WILC SPI。
     5. 按下“M”以选择 WILC SDIO 或 WILC SPI。
     6. 保存配置。

确保 arch/arm/boot/dts/at91-sama5d2_xplained.dts 文件的 mmc1 节点与以下节点类似。

```
mmc1: mmc@fc000000 {
pinctrl-names = "default";
pinctrl-0 = <&pinctrl_mmc1_clk_cmd_dat0 &pinctrl_mmc1_dat1_3 &pinctrl_mmc1_cd>;
vmmc-supply = <&vcc_mmc1_reg>;
vqmmc-supply = <&vcc_3v3_reg>;
non-removable;
status = "okay";
slot@0 {
   reg = <0>;
   bus-width = <4>;
   };
};
```

7. 使用以下命令编译内核：

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
```

8. 使用以下命令编译.dtb 文件：

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- at91-sama5d2_xplained.dtb
```

9. 使用以下命令编译模块：

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
```

编译成功后，最终的内核镜像将位于 *arch/arm/boot/*目录下，at91-sama5d2_xplained.dtb 文件将位于 arch/arm/boot/dts 文件夹下。

3.1.4 根文件系统

使用 Buildroot 编译 rootfs。有关更多信息，请参见为 SAMA5D2 Xplained Pro 板编译 rootfs。

3.2 为 SAMA5D2_Xplained 创建镜像以使用 eMMC 进行引导

需要将一个可引导镜像写入 SAMA5D2 Xplained 目标板的 eMMC。此镜像必须包含所有编译的镜像（AT91bootstrap、u-boot、内核和 rootfs）（克隆和编译二进制文件）。要创建镜像，请按以下步骤操作。

1. 在主目录下创建一个名为 junk 的目录。使用以下命令创建一个虚拟镜像文件 sdcard.img：

```
$sudo dd if=/dev/zero of=~/junk/sdcard.img bs=2G count=1
$ls -al
```

2. 移至 junk 目录，然后使用以下命令将镜像文件划分为两个分区：

```
$sudo fdisk sdcard.img
Welcome to fdisk(util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x24d68b30.

Command (m for help): n
Partion type
 p primary (0 primary, 0 extended, 4 free)
 e extended (container for logical partions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-4194295, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-4194295, default 4194295):+64M

Created a new partition 1 of type 'Linux' and of size 64 MiB.

Command (m for help): t
Selected partion 1
Partition type (type L to list all types): b
Changed type of partition 'Linux' to 'W95 FAT32'.


Command (m for help): n
```

```
Partion type
 p primary (1 primary, 0 extended, 3 free)
 e extended (container for logical partitions)
Select (default p):


Using default response p.
Partition number (2-4, default 2):
First sector (133120-4194295, default 133120):
Last sector, +sectors or +size{K,M,G,T,P} (133120-4194295, default4194295):

Created a new partition 2 of type 'Linux' and of size 2 GiB.
Command (m for help): w
The partition table has been altered.
Syncing disks.
```

随后会在 sdcard.img 文件中创建两个分区。

3. 使用以下命令将两个分区挂载到两个循环设备上：

```
$sudo losetup /dev/loop0 sdcard.img -o 1048576
$sudo losetup /dev/loop1 sdcard.img -o 68157440
```

注： 数字 1048576 和 68157440 分别为两个分区的偏移量。

可使用以下命令验证分区：

```
fdisk -l sdcard.img
Disk linux4sam-yocto-sama5d2_xplained.img: 2 GiB, 2147479552 bytes, 4194296sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x24d68b30

Device                                 Boot   Start   End Sectors  Size  Id  Type
linux4sam-yocto-sama5d2_xplained.img1  2048   133119  131072       64M   b   W95 FAT
linux4sam-yocto-sama5d2_xplained.img2  133120 4194295 4061176      2G    83  Linux
```

第一个分区从“2048”存储单元开始，其物理存储单元为 512 字节 * 2048，即 1048576。

同样，第二个分区从“133120”存储单元开始，其物理存储单元为 512 字节 * 133120，即 68157440。

4. 使用以下命令对挂载到循环设备上的分区进行格式化：

```
$sudo mkfs.vfat /dev/loop0
$sudo mkfs.ext4 /dev/loop1
```

5. 创建两个临时文件夹，然后使用以下命令将两个分区（FAT32 和 EXT4）挂载到文件夹上：

```
$ mkdir emmcmntp1
$ mkdir emmcmntp2
$ sudo mount -o loop,offset=1048576 sdcard.img emmcmntp1
$ sudo mount -o loop,offset=68157440 sdcard.img emmcmntp2
```

6. 在第一个分区（FAT32）中，使用以下命令复制 AT91bootstrap、u-boot、uboot.env、内核和 dtb 文件：

```
$ cd emmcmntp1
$ sudo cp <path>at91bootstrap/binaries/BOOT.bin .
$ sudo cp <path>u-boot-at91/u-boot.bin .
$ sudo cp <path>uboot.env .
$ sudo cp <path>linux-at91/arch/arm/boot/zImage .
$ sudo cp <path>linux-at91/arch/arm/boot/dts/at91-sama5d2_xplained.dtb .
```

7. 在第二个分区（EXT4）中，使用以下命令复制 rootfs：

```
$ cd ../emmcmntp2
$ sudo tar -zxvf <path> <path to the newly build rootfs tar file>
eg: core-image-minimal-sama5d2-xplained-20181114120437.rootfs.tar.gz
```

8. 使用以下命令卸载临时挂载点 emmcmntp1、emmcmntp2 和循环设备：

```
$ cd ..
$ sudo umount emmcmntp1 emmcmntp2
```

```
$ sudo losetup -d/dev/loop0
$ sudo losetup -d/dev/loop1
```

3.3 将演示映像安装到 SAMA5D2 Xplained eMMC 上

前提条件：

- 将 FTDI 线缆连接到调试连接器（J1）。
  注： 不要使用 J14 连接器接收调试消息。
- 通过 SAM-BA 系统内编程器下载适用于 Linux 软件的 SAM-BA® 3.2.1。

1. 在~.bashrc 文件中添加 SAM-BA 的路径。
2. 将 USB 线缆连接到 J23 端口以刷写镜像。
3. 闭合跳线 JP9，按下复位按钮，然后断开跳线。
4. 创建一个 qml 文件 `emmc-usb.qml` 并向其中添加以下内容：

```
import SAMBA 3.2
import SAMBA.Connection.Serial 3.2
import SAMBA.Device.SAMA5D2 3.2

SerialConnection {
//port: "ttyACM0"
//port: "COM85"
//baudRate: 57600

device: SAMA5D2Xplained {
}

onConnectionOpened: {
// initialize SD/MMC applet
initializeApplet("sdmmc")

// write file
applet.write(0, "sdcard.img", false)

// initialize boot config applet
initializeApplet("bootconfig")

// Use BUREG0 as boot configuration word
applet.writeBootCfg(BootCfg.BSCR, BSCR.fromText("VALID,BUREG0"))

// Enable external boot only on SDMMC0
applet.writeBootCfg(BootCfg.BUREG0,
BCW.fromText("EXT_MEM_BOOT,UART1_IOSET1,JTAG_IOSET1," +
"SDMMC0,SDMMC1_DISABLED,NFC_DISABLED," +
"SPI1_DISABLED,SPI0_DISABLED," +
"QSPI1_DISABLED,QSPI0_DISABLED"))
}
}
```

5. 使用以下命令运行.qml 脚本：

```
$sudo su
$ <path>sam-ba -x emmc-usb.qml
```

注： 此过程需要几分钟才能完成。

`sdcard.img` 安装在 SAMA5D2 Xplained eMMC 上。

刷写操作完成后，将通过 J1 端口发送调试消息。

4. 编译系统镜像并刷写到 SAMA5D3 Xplained 板

请按照以下步骤编译系统镜像并刷写到 SAMA5D3 Xplained 板。

1. 访问 https://www.at91.com/linux4sam/bin/view/Linux4SAM/Sama5d3XplainedMainPage 下载默认演示包 `linux4sam-poky-sama5d3_xplained-6.0.zip`。
2. 访问 https://www.kernel.org/下载 Linux 内核 4.4.87。
3. 将 `drivers/staging/wilc1000` 目录下的现有 WILC1000 驱动程序替换为 www.github.com/linux4wilc/上的 ATWILC 驱动程序。
4. 将 `Makefile` 中的 `CONFIG_WILC1000` 宏修改为 `CONFIG_WILC`。此文件位于 `drivers/staging/Makfile` 中。
5. 访问 https://github.com/linux4wilc/firmware 下载固件二进制文件，并用其更新内核目录`/firmware/mchp/`下的现有固件文件。
6. 使用命令 `make ARCH=arm sama5_defconfig` 配置内核。
7. 使用命令 `make ARCH=arm menucofig` 打开 menuconfig。
8. 请按照以下步骤选择 ATWILC 驱动程序模块：
   8.1. 转至 **menuconfig**。
   8.2. 导航至 **Device Drivers > Staging driver**。
   8.3. 根据要求选择 **Atmel WILC SDIO** 或 **Atmel WILC SPI**。

   注： 确保 `arch/arm/boot/dts/at91-sama5d3_xplained.dts` 文件的 `mmc1` 节点与以下节点类似：

```
mmc1: mmc@fc000000 {
pinctrl-names = "default";
pinctrl-0 = <&pinctrl_mmc1_clk_cmd_dat0 &pinctrl_mmc1_dat1_3 &pinctrl_mmc1_cd>;
vmmc-supply = <&vcc_mmc1_reg>;
vqmmc-supply = <&vcc_3v3_reg>;
non-removable;
status = "okay";
slot@0
{ reg = <0>; bus-width = <4>; };
};
```

9. 保存`.config` 文件。
10. 使用以下命令编译内核：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi
```

11. 使用以下命令编译模块：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi modules
```

12. 使用以下命令编译 zImage：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi zImage
```

13. 使用以下命令编译 dtb 文件：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi at91-sama5d2_xplained.dtb
```

    随后会在`/drivers/staging/wilc1000` 下编译 ATWILC 驱动程序模块。
    `wilc.ko`、`wilc-sdio.ko` 和 `wilc-spi.ko` 模块是 ATWILC1000 和 ATWILC3000 的通用模块。
14. 通过 SAM-BA 系统内编程器下载适用于 Linux 软件的 SAM-BA 工具版本 SAM-BA 2.16。
15. 将 SAM-BA 二进制文件的路径导出至 PATH 环境变量。
16. 将 `zImage` 和 `at91-sama5d3_xplained.dtb` 文件复制到演示包 `linux4sam-poky-sama5d3_xplained-5.6` 中。
17. 将演示包中的 `zImage` 文件重命名为 `zImage-sama5d3-xplained.bin`。
18. 将 Micro USB 线缆连接到 ATSAMA5D3 板的 EDBG-USB 连接器（J6）。
19. 将 FTDI 线缆连接到 ATSAMA5D3 板的 DEBUG 连接器（J23）以接收调试消息。

20. 使用 minicom 打开/dev/ttyUSB0。将波特率设置为 115200。
21. 断开跳线 jp5，按下复位按钮，随后会向 minicom 发送日志消息“RomBOOT”。
22. 短接跳线（jp5）并运行 demo_linux_nandflash.sh 脚本以刷写二进制文件。
23. 引导完成后，使用海量存储驱动器将 wilc.ko.wilc-sdio.ko 和 wilc-spi.ko 复制到根文件系统。
24. 访问 http://www.github.com/linux4wilc/复制固件文件，并用其替换现有固件文件。
25. 运行 wilc.ko 和 wilc-sdio.ko 模块。成功后，wlan0 接口便会启动并运行。

5. 编译系统镜像并刷写到 SAMA5D27-SOM1-EK1

本章介绍如何编译组件以在 SAMA5D27-SOM1-EK1 板上运行 Linux。此配置将从 micro-SD 卡槽引导系统，因此可将标准 SD 卡槽用于 ATWILC1000 SDIO 板。

注： 由于复位信号和芯片使能信号不会通过 SDIO 连接器传送到 ATWILC1000 模块，因此在系统复位时，必须对 SDIO 板执行掉电再上电才能将此模块复位。否则，Linux 提示符会反复报错，直到将卡拔出并重新插入为止。通过板上的 NRST 或通过 Linux 重启命令执行复位时可以观察到此行为。

对于演示板而言，变通方法为将卡拔出并重新插入。在客户端应用中，必须通过 I/O 引脚控制 ATWILC1000 的复位信号和芯片使能信号。

如果编译成功，则可在 `arch/arm/boot/`目录下找到最终的镜像。

5.1 编译组件

本节介绍编译 Bootstrap 和 U-Boot 的步骤。

5.1.1 Bootstrap

本节介绍从 git 资源库获取源代码，进行默认配置，根据默认配置自定义 AT91Bootstrap 以及编译 AT91Bootstrap 以生成二进制文件的步骤。

1. 使用以下命令克隆资源库以获取源代码：

```
git clone git://github.com/linux4sam/at91bootstrap.git
cd at91bootstrap/
```

2. 配置 AT91Bootstrap。假设用户位于 AT91Bootstrap 根目录下，其中有一个包含多个默认配置文件的 `board/sama5d27_som1_ek` 文件夹：
   配置 AT91Bootstrap 以从 SD 卡载入 U-boot 二进制文件。

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-ama5d27_som1_eksd_uboot_defconfig
```

   如果配置成功，则可在 AT91Bootstrap 根目录下找到`.config` 文件。

3. 使用 menuconfig 自定义 AT91Bootstrap。
   输入以下命令并选择 SDHC1（而非默认的 SDHC0）作为 SD 卡接口。

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

4. 使用以下命令编译 AT91Bootstrap：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

   编译成功后，可在 `binaries/at91bootstrap.bin` 文件夹中找到最终的`.bin` 镜像。

5.1.2 U-Boot

请按照以下步骤编译 U-boot。

1. 使用以下命令克隆 Linux4sam GitHub U-Boot 资源库，以获取 SAMA5D2 的默认 u-boot 代码：

```
git clone git://github.com/linux4sam/u-boot-at91.git
cd u-boot-at91
```

2. 源代码从通向最新分支的主分支获取。如果用户要使用其他分支，则可以使用以下命令列出这些分支并使用其中一个分支：

```
git branch -r
git checkout origin/u-boot-2018.07-at91 -b u-boot-2018.07-at91
```

3. 编译 u-boot。U-Boot 环境变量可存储在不同介质中，具体由配置文件指定。使用以下命令添加 SD/MMC 卡中的环境变量：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sama5d27_som1_ek_mmc1_defconfi
```

4. 使用以下命令编译 U-boot：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

随后会生成 U-boot 二进制文件 `u-boot.bin`。

5.2 编译内核

请按照以下步骤编译内核。

1. 使用以下命令克隆资源库以获取源代码：

```
git clone git://github.com/linux4sam/linux-at91.git
```

2. 要使用其他分支，可通过以下命令列出这些分支并使用其中一个分支：

```
git branch -r
git checkout origin/linux-4.14-at91 -b linux-4.14-at91
```

3. 将 `drivers/staging/wilc` 中的 ATWILC1000 驱动程序替换为 ATWILC 驱动程序资源库中 `driver/wilc` 目录下的内容。资源库可从以下地址获取：https://github.com/linux4wilc。
输入以下命令以从 `linux4wilc` 中获取相应的文件：

```
git clone git://github.com/linux4wilc/driver
```

4. 在 `linux-at91/drivers/staging/Makefile` 中修改以下行，以便编译时能够找到正确的目录：

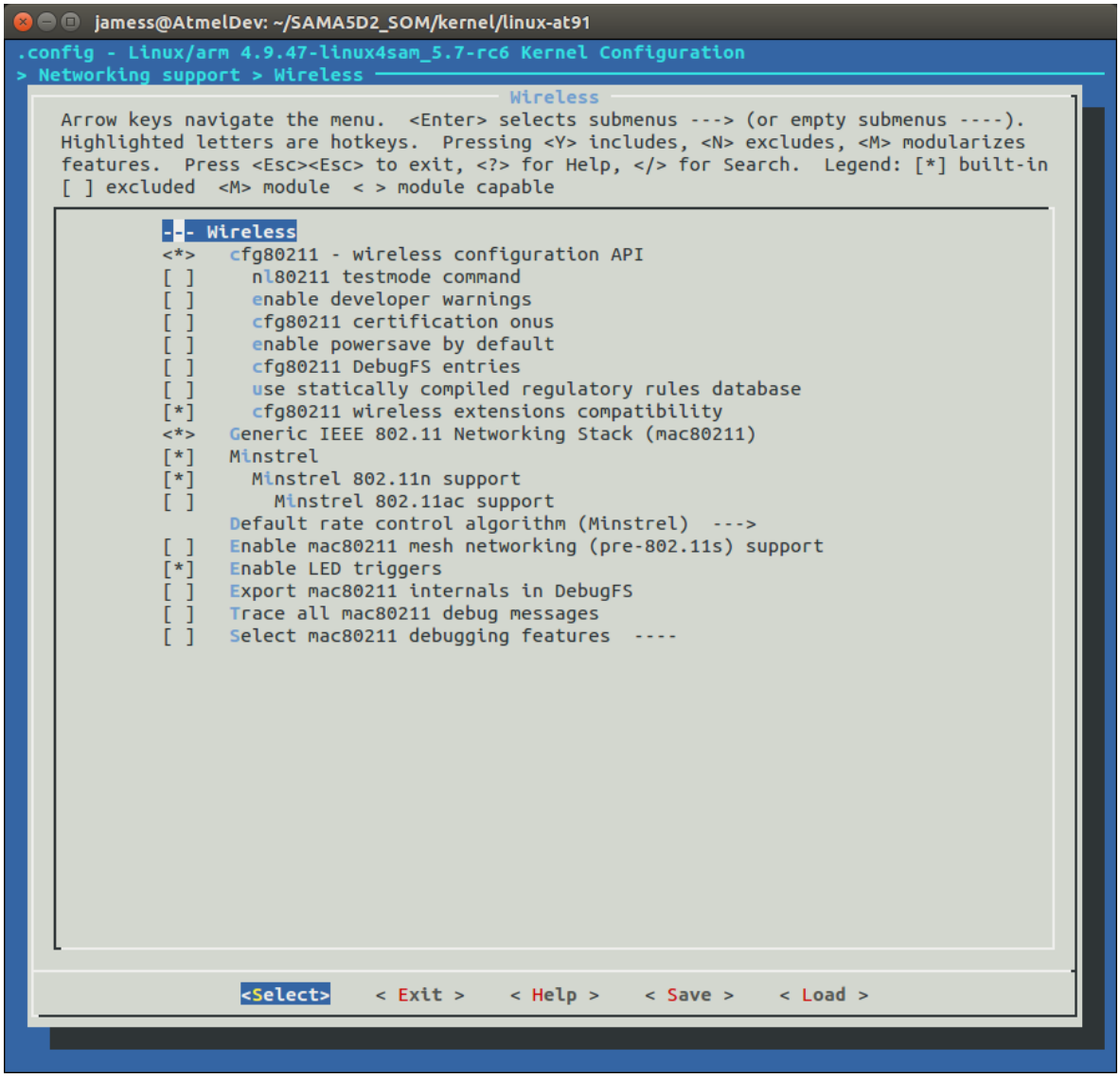
```
FROM: obj-$(CONFIG_WILC1000) += wilc1000/
TO: obj-$(CONFIG_WILC) += wilc1000/
```

5. 使用以下命令配置内核：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sama5_defconfig
```

6. 使用 menuconfig 修改默认配置。请按照以下步骤打开 menuconfig：
   - `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menucofig`
   - 请按照以下步骤选择 ATWILC 驱动程序模块：
     1. 转至 menuconfig。
     2. 导航至 *Device Drivers* > *Staging driver*。
     3. 按下“y”以包含分段驱动程序。
     4. 根据要求选择 Atmel WILC SDIO 或 Atmel WILC SPI。
     5. 按下“M”以选择 WILC SDIO 或 WILC SPI。
     6. 保存配置。
7. 选择 *Networking Support* > *Wireless*（网络支持 > 无线），如下图所示：
注： 一定要严格按照下图完成所有配置。

图 5-1. 网络支持 Menuconfig 窗口



8. 使用以下命令编译内核：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-z Image
```

如果编译成功，则可在 arch/arm/boot/目录下找到最终的镜像。

9. 使用以下命令编译模块：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi modules
```

10. 使用以下命令编译 dtb 文件：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi at91-sama5d2_xplained.dtb
```

6. 将二进制文件和系统镜像更新到目标板

本章介绍如何更新或刷写系统镜像。预编译镜像包括预编译驱动程序和固件二进制文件，可从 GitHub 上获取。

使用 SAM-BA®工具将二进制文件刷写到目标板。
注： 更新系统镜像之前，应确保已在主机计算机上安装 SAM-BA 工具。演示包中的脚本可使用 SAM-BA 2.16 或 3.2.x，具体取决于用户在下文的步骤 5 中选择的下载脚本。

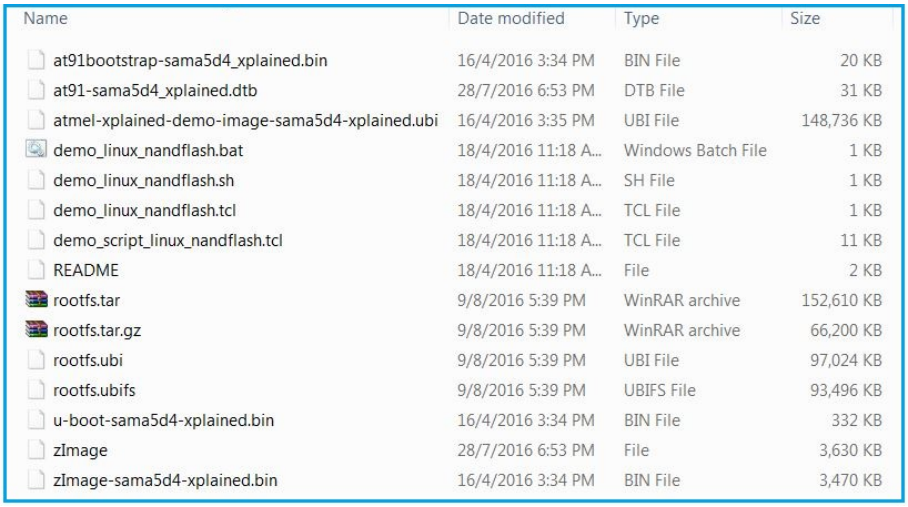
有关更多信息，请参见以下内容：

- 软件工具
- SAMA5D4 Xplained 板
- ATSAMA5D44 微处理器

要启动刷写操作，请按以下步骤操作：

1. 访问 https://github.com/linux4wilc/wilc_demo 下载预编译镜像。
2. 解压缩下载的文件。
3. 按第 2 章为 SAMA5D4 Xplained Ultra 板编译 Linux 所述编译新镜像后，必须将这些文件从 `buildroot\output\images` 目录复制到 `demo_linux_nandflash.tcl` 文件所在的目录。

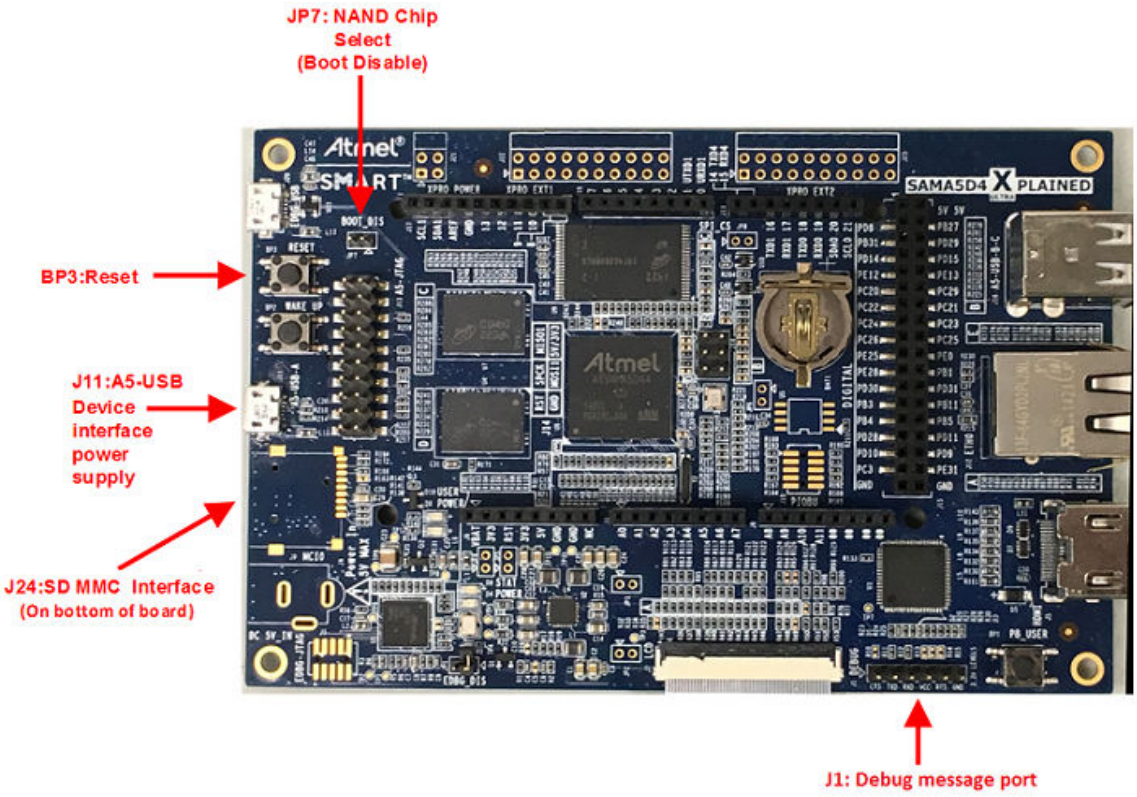
   图 6-1. buildroot\output\images 目录下的文件列表



Name	Date modified	Type	Size
at91bootstrap-sama5d4_xplained.bin	16/4/2016 3:34 PM	BIN File	20 KB
at91-sama5d4_xplained.dtb	28/7/2016 6:53 PM	DTB File	31 KB
atmel-xplained-demo-image-sama5d4-xplained.ubi	16/4/2016 3:35 PM	UBI File	148,736 KB
demo_linux_nandflash.bat	18/4/2016 11:18 A...	Windows Batch File	1 KB
demo_linux_nandflash.sh	18/4/2016 11:18 A...	SH File	1 KB
demo_linux_nandflash.tcl	18/4/2016 11:18 A...	TCL File	1 KB
demo_script_linux_nandflash.tcl	18/4/2016 11:18 A...	TCL File	11 KB
README	18/4/2016 11:18 A...	File	2 KB
rootfs.tar	9/8/2016 5:39 PM	WinRAR archive	152,610 KB
rootfs.tar.gz	9/8/2016 5:39 PM	WinRAR archive	66,200 KB
rootfs.ubi	9/8/2016 5:39 PM	UBI File	97,024 KB
rootfs.ubifs	9/8/2016 5:39 PM	UBIFS File	93,496 KB
u-boot-sama5d4-xplained.bin	16/4/2016 3:34 PM	BIN File	332 KB
zImage	28/7/2016 6:53 PM	File	3,630 KB
zImage-sama5d4-xplained.bin	16/4/2016 3:34 PM	BIN File	3,470 KB

4. 在 JP7 处添加跳线，然后通过 J11 处的 USB 端口连接到主机 PC。确保主机计算机完成 USB 串行端口连接，然后移除 JP7 处的跳线。下图显示了 SAMA5D4 适配器的连接。

图 6-2. SAMA5D4 适配器连接



5. 执行 `demo_linux_nandflash.bat`（针对 Windows®）文件或 `demo_linux_nandflash.sh`（针对 Linux）文件。

   注：

   - 默认情况下，`demo_linux_nandflash.sh` 文件中包含适用于 32 位操作系统的 `sam-ba` 二进制文件。对于 64 位操作系统，需将此文件中的 `sam-ba` 更改为 `sam-ba_64`。
   - 在超级用户模式下执行脚本。如果安装了 sam-ba 3.2，则使用 `demo_linux_nandflash_3_2.bat` 或 `demo_linux_nandflash_3_2.sh` 代替。

   可通过 J1 串行端口查看输出日志。
   通过 COM 端口在 PC 上打开串行终端，配置如下：

   - 115200 波特率
   - 8 位数据
   - 无奇偶校验
   - 1 个停止位
   - 无流控制

6. 如果自动打开日志文件，即表示系统镜像已成功下载到目标板上。此日志文件包含下载过程的所有历史记录。

7. 更新 ATWILC 固件

本章介绍如何更新演示镜像上的 ATWILC 固件或驱动程序。

7.1 ATWILC1000 和 ATWILC3000 驱动程序模块

在系统完成引导后，将 ATWILC 驱动程序模块 `wilc-sdio.ko` 或 `wilc-spi.ko` 添加到`/lib/modules/4.9.xx-XX/kernel/drivers/staging/wilc1000/`目录下，或者复制到文件系统上的任意位置。

7.2 ATWILC1000 和 ATWILC3000 固件二进制文件

1. 将 ATWILC1000 固件 `wilc1000_wifi_firmware.bin` 添加到`/lib/firmware/mchp/`目录下。
2. 将 ATWILC3000 Wi-Fi 固件 `wilc3000_wifi_firmware.bin` 添加到`/lib/firmware/mchp/`目录下。
3. 将 ATWILC3000 Bluetooth®固件 `wilc3000_ble_firmware.bin` 添加到`/lib/firmware/mchp/`目录下。

注： 可访问 https://github.com/linux4wilc/firmware 获取固件。

可采用以下任一传输协议将文件传输到 SAMA5D4 平台：

- 以太网
- ZMODEM

7.2.1 通过以太网添加文件

可使用以下命令通过局域网（Local Area Network，LAN）/广域网（Wide Area Network，WAN）将文件从一台计算机传输到另一台计算机：

```
$ scp [path of file to send] root@[receiver's IP]:[target directory]
```

例如，以下命令可使用内部 IP 地址 `192.168.0.11` 将 `wilc1000_wifi_firmware.bin` 文件从二进制文件目录发送到设备的`/lib/firmware/mchp` 目录下。

```
$ scp binary/wilc1000_wifi_firmware.bin root@192.168.0.11:/lib/firmware/mchp
```

7.2.2 通过 ZMODEM 添加文件

也可采用 ZMODEM 文件传输协议传输文件。

在 Teraterm 中，使用以下命令更改目标位置目录：

```
$ cd Target_location
```

使用以下命令执行 ZMODEM 命令：

```
$ rz
```

在 Teraterm 的 `File`（文件）菜单中，选择 `Transfer` > `Send`（传输 > 发送），然后浏览并选择所需文件。

8. 运行 ATWILC

本章介绍如何在 SAMA5D4 Xplained 板或任何类似的 Linux 平台上使用 ATWILC1000 和 ATWILC3000。

8.1 访问控制台

用户可通过板上的串口转 USB 转换器访问串行控制台。评估工具包上的嵌入式调试器（EDBG）芯片实际作为串口转 USB 转换器，并载入可通过 USB-CDC 协议进行通信的固件。

要使能 EDBG，应断开 JP1 并将 USB 线缆连接到电路板（J20 EDBG-USB）。

8.1.1 对于 Microsoft Windows 用户

为 Atmel 和 Segger 工具安装 USB 驱动程序，然后识别已建立的 USB 连接。用户可通过设备管理器中是否出现 EDBG 虚拟 COM 端口来判断。可使用 COMxx 编号配置终端仿真器。

8.1.2 对于 Linux 用户

通过监视 `dmesg` 命令的最后几行来识别 USB 连接。可使用`/dev/ttyACMx` 编号配置终端仿真器。

以下是 USB 调试端口连接：

```
[172677.700868] usb 2-1.4.4: new full-speed USB device number 31 using ehci-pci
[172677.792677] usb 2-1.4.4: not running at top speed; connect to a high speed hub
[172677.793418] usb 2-1.4.4: New USB device found, idVendor=03eb, idProduct=6124
[172677.793424] usb 2-1.4.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[172677.793897] cdc_acm 2-1.4.4:1.0: This device cannot do calls on its own.It is not a modem.
[172677.793924] cdc_acm 2-1.4.4:1.0: ttyACM0: USB ACM device
```

标识符 **idVendor=03eb** 和 **idProduct=6124** 用于指示设备为具有 USB 连接的评估工具包板。

在对终端进行相应的设置（见表 8-1）后，就能使用终端仿真器与 SAMA5D4 适配器通信。

8.1.3 串行通信参数

串行通信参数如下：

表 8-1. 串行端口设置

功能	设置
波特率	115200
数据	8 位
奇偶校验	无
停止位	1 位
流控制	无

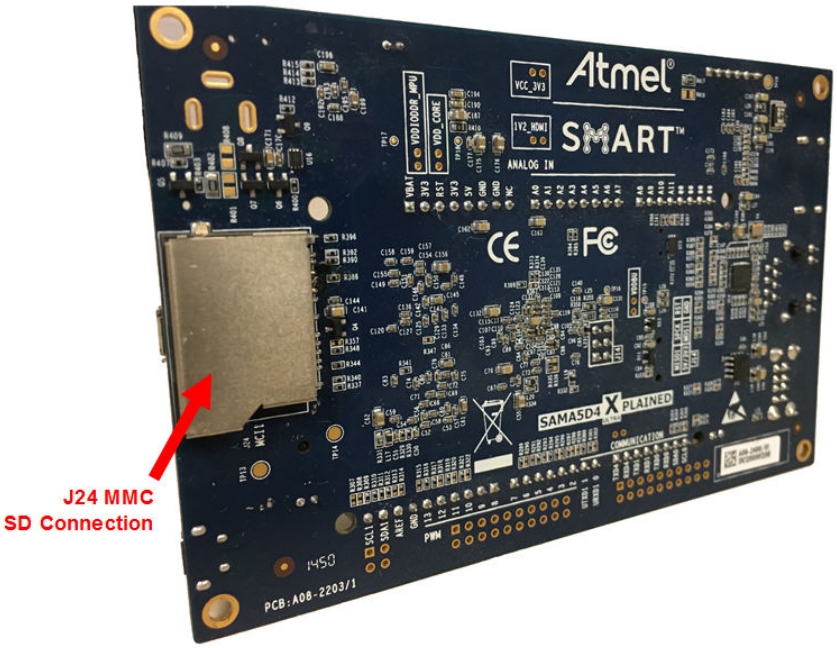
8.2 识别 ATWILC1000

本节介绍 SD Express 板和串行外设接口（Serial Peripheral Interface，SPI）板的连接。

8.2.1 SD Express 板

在执行引导操作之前，应确保 ATWILC1000 SD Express 板已连接到 SAMA5D4 Xplained 板的 SD 插槽（J24）中（见下图）。

图 8-1. SAMA5D4 SD 连接



在引导期间，通过以下行来识别安全数字输入/输出（Secure Digital Input/Output，SDIO）Express 卡。

```
mmc0: new high speed SDIO card at address 0001
```

使用以下命令载入 ATWILC1000 模块 SDIO 驱动程序。

```
Welcome to Buildroot
buildroot login: root
[root@buildroot ~]# insmod wilc.ko
wilc: module is from the staging directory, the quality is unknown, you have been warned.
[root@buildroot ~]# insmod wilc-sdio.ko
wilc_sdio: module is from the staging directory, the quality is unknown, you have been
warned.
linux_sdio_probe init_power =0
wilc_sdio mmc0:0001:1:Driver Initializing success
```

注： 如果在载入模块时收到以下消息，请不要担心：
`wilc: module is from the staging directory, the quality is unknown, you have been warned`

这是内核分段目录下所有驱动程序的默认消息。

8.2.2 串行外设接口板

必须将 ATWILC1000 串行外设接口（SPI）板连接到 J17 处的 SPI1 接口，如下图所示。

图 8-2. SAMA5D4 SPI 连接

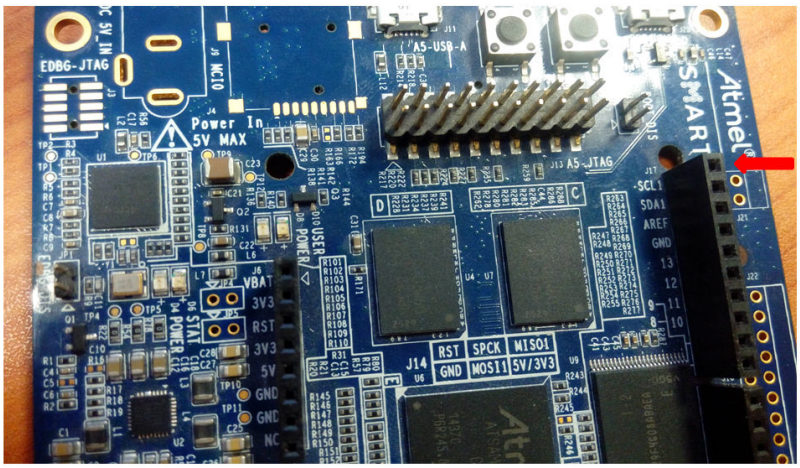


表 8-2. SPI 引脚说明

SPI 引脚	插座 J17 引脚	XPRO EXT1 引脚
MOSI	PIN11	PIN16
CLK	PIN13	PIN18（SPCK）
MISO	PIN12	PIN17
CS	PIN10	PIN15
IRQ	PIN8	PIN9

注： 可将 SPI 卡中的 VEXT 引脚连接到插座 J6 中的 3V3 引脚。或者，也可将 WINC1500/WINC3400 Xplained Pro 板直接连接到 XPRO EXT1 插座（与 J17 一样公开 SPI1 外设）。在这种情况下，必须将 IRQ GPIO 更改为 PB26，即 XPRO EXT1 的引脚 9。

8.3 识别 ATWILC3000

本节介绍 SDIO Shield 板和 SPI Shield 板的连接。

8.3.1 SDIO Shield 板

在执行引导操作之前，应确保 ATWILC3000 Shield 板已连接到 SAMA5D4 Xplained 适配器的 Shield Arduino Shield 堆叠连接器。

使用以下命令载入 Wi-Fi SDIO 驱动程序模块：

```
# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been warned.
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [wilc_wfi_cfg_alloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [wilc_wfi_cfg_alloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
wilc_sdio mmc0:0001:1: WILC got 60 for gpio_reset
wilc_sdio mmc0:0001:1: WILC got 94 for gpio_chip_en
wilc_sdio mmc0:0001:1: WILC got 91 for gpio_irq
wifi_pm : 0
wifi_pm : 1
wilc_sdio mmc0:0001:1: Driver Initializing success
# wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_netdev_cleanup]Unregistering netdev d4643800
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_netdev_cleanup]Freeing Wiphy...
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_free_wiphy]Unregistering wiphy
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_free_wiphy]Freeing wiphy
```

```
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_netdev_cleanup]Freeing netdev...
wilc_sdio mmc0:0001:1 p2p0: INFO [wilc_netdev_cleanup]Unregistering netdev d46ba800
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_netdev_cleanup]Freeing Wiphy...
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_free_wiphy]Unregistering wiphy
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_free_wiphy]Freeing wiphy
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_netdev_cleanup]Freeing netdev...
Module_exit Done.
at_pwr_dev: deinit
at_pwr_dev: unregistered
mmc0: card 0001 removed
mmc0: new high speed SDIO card at address 0001
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [wilc_wfi_cfg_alloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [wilc_wfi_cfg_alloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
wilc_sdio mmc0:0001:1: WILC got 60 for gpio_reset
wilc_sdio mmc0:0001:1: WILC got 94 for gpio_chip_en
wilc_sdio mmc0:0001:1: WILC got 91 for gpio_irq
wilc_sdio mmc0:0001:1: Driver Initializing success
```

注： 如果在载入模块时收到以下消息，请不要担心：

```
wilc: module is from the staging directory, the quality is unknown, you have been warned
```

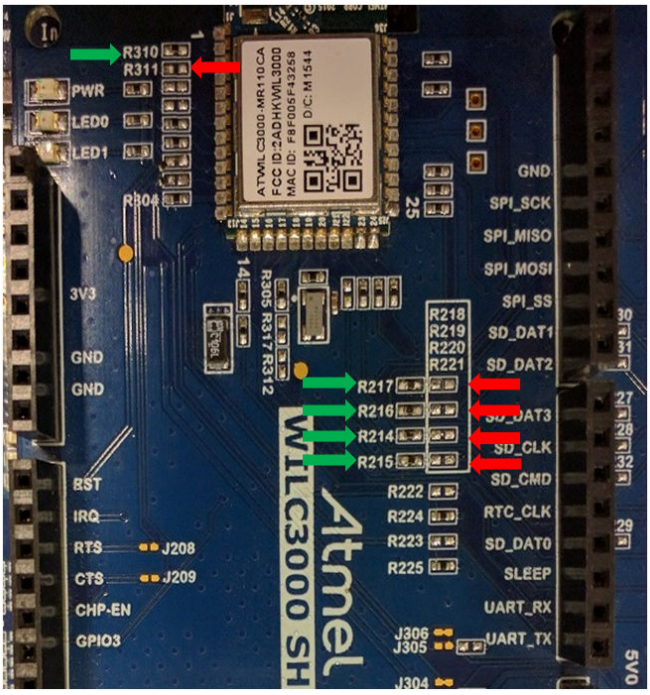
这是内核分段目录下所有驱动程序的默认消息。

8.3.2 串行外设接口 Shield 板

ATWILC3000 Shield 板支持 SDIO 和 SPI 两种工作模块，具体通过安装或移除 0Ω 电阻来进行配置。默认情况下，此板预先配置为 SDIO 模式。

要切换到 SPI 模式，用户必须按下图所示更改电阻。

图 8-3. 配置为 SPI 模式的 ATWILC3000 Shield 板



必须连接用绿色箭头标记的电阻，同时移除用红色箭头标记的电阻。

表 8-3. SPI 电阻配置

要移除的电阻	要连接的电阻
R311	R310
R218	R214
R219	R215
R220	R216
R221	R217

1. 使用以下命令载入 Wi-Fi SDIO 驱动程序模块：

```
# modprobe wilc-spi
wilc_spi: module is from the staging directory, the quality is unknown, you have been
warned.
WILC_SPI spi32765.0: spiModalias: wilc_spi, spiMax-Speed: 48000000
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
WILC_SPI spi32765.0: WILC got 60 for gpio_reset
WILC_SPI spi32765.0: WILC got 94 for gpio_chip_en
WILC_SPI spi32765.0: WILC got 91 for gpio_irq
wifi_pm : 0
wifi_pm : 1
WILC_SPI spi32765.0: WILC SPI probe success
# ifconfig wlan0 up
WILC_SPI spi32765.0 wlan0: INFO [wilc_mac_open]MAC OPEN[d477d800] wlan0
WILC POWER UP
WILC_SPI spi32765.0 wlan0: INFO [wilc_init_host_int]Host[d477d800][d477cc00]
WILC_SPI spi32765.0 wlan0: INFO [wilc_mac_open]*** re-init ***
WILC_SPI spi32765.0 wlan0: INFO [wlan_init_locks]Initializing Locks ...
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_init]Initializing WILC_Wlan ...
WILC_SPI spi32765.0 wlan0: INFO [init_chip]Bootrom sts = c
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_initialize]WILC Initialization done
WILC_SPI spi32765.0 wlan0: INFO [init_irq]IRQ request succeeded IRQ-NUM= 137 on GPIO: 91
WILC_SPI spi32765.0 wlan0: INFO [wlan_initialize_threads]Initializing Threads ...
WILC_SPI spi32765.0 wlan0: INFO [wlan_initialize_threads]Creating kthread for
transmission
WILC_SPI spi32765.0 wlan0: INFO [wlan_initialize_threads]Creating kthread for Debugging
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_get_firmware]Detect chip WILC3000
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_get_firmware]loading firmware mchp/
wilc3000_wifi_firmware.bin
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_get_firmware]WLAN firmware: mchp/
wilc3000_wifi_firmware.bin
WILC_SPI spi32765.0 wlan0: INFO [wilc_firmware_download]Downloading Firmware ...
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_firmware_download]Downloading firmware size =
137172
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_firmware_download]Offset = 120228
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_firmware_download]Offset = 137172
WILC_SPI spi32765.0 wlan0: INFO [wilc_firmware_download]Download Succeeded
WILC_SPI spi32765.0 wlan0: INFO [linux_wlan_start_firmware]Starting Firmware ...
WILC_SPI spi32765.0 wlan0: INFO [linux_wlan_start_firmware]Waiting for Firmware to get
ready ...
WILC_SPI spi32765.0 wlan0: INFO [linux_wlan_start_firmware]Firmware successfully started
WILC_SPI spi32765.0 wlan0: INFO [wilc_wlan_initialize]WILC Firmware Ver =
WILC_WIFI_FW_REL_15_00_RC4 Build: 9153
[root@buildroot ~]#
```

8.4 修改配置文件

要使用 Wi-Fi 模块，用户必须在预编译的镜像上载入一组默认配置文件。这些文件可根据下文所述的要求进行修改。

8.4.1 Wi-Fi 保护接入客户端

Wi-Fi 保护接入（Wi-Fi Protected Access，WPA）客户端的参考配置文件位于/etc/目录下。站点模式和接入点模式的配置文件均位于演示预编译镜像中。

8.4.1.1 站点模式

站点模式的配置文件 wilc_wpa_supplicant.conf 包含以下行。

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
```

8.4.1.2 开放式安全接入点模式

开放式安全接入点（Access Point，AP）模式的配置文件 wilc_hostapd_open.conf 包含以下行。

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=7
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
```

8.4.1.3 WEP 安全接入点模式

支持有线等效加密（Wired Equivalent Privacy，WEP）安全标准的 AP 模式配置文件 wilc_hostapd_wep.conf 包含以下行。

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=7
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
ieee80211n=1
auth_algs=1

#########    WEP  ############
wep_default_key=0
wep_key0=1234567890
wep_key1="vwxyz"
wep_key2=0102030405060708090a0b0c0d
wep_key3=".2.4.6.8.0.23"
wep_key_len_broadcast=5
wep_key_len_unicast=5
wep_rekey_period=300
```

8.4.1.4 WPA 安全模式

支持 WPA 安全标准的 AP 模式配置文件 wilc_hostapd_wpa.conf 包含以下行。

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=7
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
ieee80211n=1
auth_algs=1
```

```
#########    WPA/WPA2  ############
wpa=3
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
```

8.4.2 动态主机配置协议

动态主机配置协议（Dynamic Host Configuration Protocol，DHCP）服务器的参考配置文件位于/etc/dhcp/dhcpd.conf 文件中。

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;

option subnet-mask 255.255.255.0;
option domain-name-servers 168.126.63.1, 164.124.101.2; # DNS Server IP
option domain-name “sample.example”;                     # domain name

subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.100 192.168.0.110;                    # range ip
   option broadcast-address 192.168.0.255;
   option routers 192.168.0.1;                           # gateway ip
}
Log-facility local7;
```

注： 必须根据测试环境修改每个值。

dhcpd.conf 文件的位置应与 test -f /etc/dhcp/dhcpd.conf || exit 0 下的/etc/init.d/S80dhcp-server 中定义的位置相匹配：

8.4.3 radvd

对于 IPv6，需要使用 radvd 配置文件。演示镜像上的参考文件位于/etc/radvd.conf 目录下。

```
interface wlan0
{
    AdvSendAdvert on;
    prefix 2001:db8:0:2::/64
    {
    };
};
```

8.5 以站点模式运行 ATWILC

下面举例说明如何以站点模式运行 ATWILC 器件以及连接到 AP。

1. 使用以下命令初始化 ATWILC1000 和 ATWILC3000 驱动程序模块：

```
Welcome to Buildroot
buildroot login: root
root@buildroot ~]# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been
warned.
linux_sdio_probe init_power =0
wilc_sdio mmc0:0001:1: Driver Initializing success
```

2. 使用以下命令启动 WPA 客户端服务并执行 wpa_supplicant：

```
# wpa_supplicant -iwlan0 -Dnl80211 -c /etc/wilc_wpa_supplicant.conf &
# Successfully initialized wpa_supplicant
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mgmt_frame_register]Frame registering Frame
Type: d0: Boolean: 1
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mgmt_frame_register]Return since mac is closed
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]MAC OPEN[d464f800] wlan0
WILC POWER UP
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_init_host_int]Host[d464f800][d463b000]
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]*** re-init ***
```

```
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_init_locks]Initializing Locks ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_init]Initializing WILC_Wlan
wilc_sdio mmc0:0001:1: SDIO speed: 50000000
wilc_sdio mmc0:0001:1: chipid 001003a0
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_initialize]WILC Initialization done
wilc_sdio mmc0:0001:1 wlan0: INFO [init_irq]IRQ request succeeded IRQ-NUM= 137 on GPIO:
91
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Initializing Threads ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Creating kthread for
transmission
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Creating kthread for Debugging
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]Detect chip WILC1000
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]loading firmware mchp/
wilc1000_wifi_firmware.bin
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]WLAN firmware: mchp/
wilc1000_wifi_firmware.bin
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_firmware_download]Downloading Firmware ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Downloading firmware size
= 134964
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Offset = 119660
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Offset = 134964
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_firmware_download]Download Succeeded
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Starting Firmware ...
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Waiting for FW to get
ready ...
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Firmware successfully
started
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_initialize]WILC Firmware Ver =
WILC_WIFI_FW_REL_15_01_RC3 Build: 9792
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_init_test_config]Start configuring Firmware
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]Mac address: fa:f0:05:f1:3d:64
```

3. 连接到接入点：

3.1. 要连接到非安全 AP：
应使用以下命令扫描 AP 并与之连接。

```
# wpa_cli -p/var/run/wpa_supplicant ap_scan 1
# wpa_cli -p/var/run/wpa_supplicant add_network
# wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP"'
# wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE
# wpa_cli -p/var/run/wpa_supplicant select_network 0
```

注： 将 **User_AP** 更改为所需 AP 的服务集标识符（Service Set Identifier，SSID）。

3.2. 要连接到 WPA 安全接入点：
应使用以下命令扫描受 WPA 或 WPA2 和临时密钥完整性协议（Temporal Key Integrity Protocol，TKIP）或高级加密标准（Advanced Encryption Standard，AES）保护的 AP，并与其连接。

```
# wpa_cli -p/var/run/wpa_supplicant ap_scan 1
# wpa_cli -p/var/run/wpa_supplicant add_network
# wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP"'
# wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt WPA-PSK
# wpa_cli -p/var/run/wpa_supplicant set_network 0 psk '"12345678"'
# wpa_cli -p/var/run/wpa_supplicant select_network 0
```

注： 将 **User_AP** 和 **12345678** 分别更改为所需 AP 的 SSID 和密码。

3.3. 要连接到 WEP 安全接入点：
应使用以下命令扫描受 WEP 共享密钥保护的 AP，并与其连接。

```
#wpa_cli  -p/var/run/wpa_supplicant ap_scan 1
#wpa_cli  -p/var/run/wpa_supplicant add_network
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 ssid  ‘“User_AP”’
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE
#wpa_cli -iwlan0 -p/var/run/wpa_supplicant set_network 0 wep_key0 1234567890
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 auth_alg SHARED
#wpa_cli  -p/var/run/wpa_supplicant select_network 0
```

注： 将 **User_AP** 和 **12345** 分别更改为所需 AP 的服务集标识符（SSID）和 ASCII（即十六进制数）。

3.4. 使用以下命令连接到 WPS 安全接入点触发 WPS 按钮模式：

```
wpa_cli wps_pbc
```

（或）使用以下命令通过 PIN 码方法连接：

```
sudo wpa_cli wps_pin any <the pin>
```

4. 运行 DHCP 服务。
如果可从 AP 自动分配 IP 地址，应使用以下命令启动 DHCP 客户端：

```
#dhcpcd wlan0 &
```

注： 如果 AP 不支持 DHCP 服务，应使用 `ifconfig wlan0 xxx,xxx.xxx.xxx` 命令手动设置静态 IP 地址值。

5. 使用以下命令检查并验证连接状态：

```
# wpa_cli status

bssid=88:9b:39:f3:d0:4d
ssid=User_AP
id=0
mode=station
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.43.2
address=00:80:c2:b3:d7:4d
```

用户可使用 Linux 中的 `wpa_cli save` 命令来保存和使用网络信息，以便自动连接到网络。

8.6 以 AP 模式运行 ATWILC

本节介绍如何将设备连接到 ATWILC1000 接入点。

1. 使用以下命令初始化 ATWILC1000 或 ATWILC3000 驱动程序模块：

```
[root@buildroot ~]# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been
warned.
linux_sdio_probe init_power =0
wilc_sdio mmc0:0001:1: Driver Initializing success
```

2. 使用以下命令将 hostapd 作为用户配置运行：

```
# hostapd /etc/wilc_hostapd_open.conf -B &
# Configuration file: /etc/wilc_hostapd_open.conf
wilc_sdio mmc0:0001:1 wlan0: INFO [change_virtual_intf]In Change virtual interface
function
wilc_sdio mmc0:0001:1 wlan0: INFO [change_virtual_intf]Wireless interface name =wlan0
wilc_sdio mmc0:0001:1 wlan0: INFO [change_virtual_intf]Changing virtual interface,
enable scan
wilc_sdio mmc0:0001:1 wlan0: INFO [change_virtual_intf]Interface type = NL80211_IFTYPE_AP
wilc_sdio mmc0:0001:1 wlan0: INFO [add_virtual_intf]Adding monitor interface[d4789800]
wilc_sdio mmc0:0001:1 wlan0: INFO [add_virtual_intf]Initializing mon ifc virtual device
driver
wilc_sdio mmc0:0001:1 wlan0: INFO [add_virtual_intf]Adding monitor interface[d4789800]
wilc_sdio mmc0:0001:1 wlan0: INFO [add_virtual_intf]Setting monitor flag in private
structure
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]MAC OPEN[d4789800] wlan0
WILC POWER UP
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_init_host_int]Host[d4789800][d45dd000]
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]*** re-init ***
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_init_locks]Initializing Locks ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_init]Initializing WILC_Wlan
wilc_sdio mmc0:0001:1: SDIO speed: 50000000
wilc_sdio mmc0:0001:1: chipid 001003a0
```

```
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_initialize]WILC Initialization done
wilc_sdio mmc0:0001:1 wlan0: INFO [init_irq]IRQ request succeeded IRQ-NUM= 137 on GPIO:
91
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Initializing Threads ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Creating kthread for
transmission
wilc_sdio mmc0:0001:1 wlan0: INFO [wlan_initialize_threads]Creating kthread for Debugging
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]Detect chip WILC1000
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]loading firmware mchp/
wilc1000_wifi_firmware.bin
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_get_firmware]WLAN firmware: mchp/
wilc1000_wifi_firmware.bin
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_firmware_download]Downloading Firmware ...
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Downloading firmware size
= 134964
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Offset = 119660
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_firmware_download]Offset = 134964
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_firmware_download]Download Succeeded
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Starting Firmware ...
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Waiting for FW to get
ready ...
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_start_firmware]Firmware successfully
started
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_initialize]WILC Firmware Ver =
WILC_WIFI_FW_REL_15_01_RC3 Build: 9792
wilc_sdio mmc0:0001:1 wlan0: INFO [linux_wlan_init_test_config]Start configuring Firmware
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_mac_open]Mac address: fa:f0:05:f1:3d:64

wilc_sdio mmc0:0001:1 wlan0: INFO [del_station]Deleting station
wilc_sdio mmc0:0001:1 wlan0: INFO [del_station]All associated stations
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_del_allstation]NO ASSOCIATED STAS
Using interface wlan0 with hwaddr fa:f0:05:f1:3d:64 and ssid "wilc1000_SoftAP"
wilc_sdio mmc0:0001:1 wlan0: INFO [start_ap]Starting ap
wilc_sdio mmc0:0001:1 wlan0: INFO [start_ap]Interval= 100
 DTIM period= 2
 Head length= 66 Tail length= 9
wilc_sdio mmc0:0001:1 wlan0: INFO [set_channel]Setting channel 7 with frequency 2442
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_set_bssid]set bssid on[d4789800]
wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_wlan_set_bssid]set bssid [fa][f0][5]
wilc_sdio mmc0:0001:1 wlan0: INFO [change_bss]Changing Bss parametrs
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

注： 关于未加密 AP 设置，请参见 `wilc_hostapd_open.conf` 文件；关于 WEP AP 设置，请参见 `wilc_hostapd_wep.conf` 文件；关于 WPA/WPA2 AP 设置，请参见 `wilc_hostapd_wpa.conf` 文件。

3. 运行 DHCP 服务器，为客户端分配 IP。使用`#ifconfig wlan0 192.168.0.1` 命令设置网关的 IP 地址。
注： 网关 IP 地址在 `dhcpd.conf` 文件中定义。

   使用`#/etc/init.d/S80dhcp-server start` 命令启动 DHCP 服务器。

   用户现在即可将 PC 或智能手机连接到 ATWILC1000 接入点。

   要将 AP 配置为 WPS 模式，可首先按照 WPA/WPA2 设置的步骤操作，然后使用以下命令配置为按钮模式：

```
hostapd_cli wps_pbc
```

   （或）使用以下命令配置为 PIN 码模式：

```
hostapd_cli wps_pin any <pin>
```

8.7 以 P2P 模式运行 ATWILC

P2P 组包括两个设备：一个设备充当 P2P 组所有者（Group Owner，GO），而另一个设备充当 P2P 客户端。ATWILC 器件同时支持 P2P 组所有者和 P2P 客户端模式。以下是在 ATWILC 上测试 P2P 模式的步骤。

可在两种情形下测试 P2P 模式，具体如下所述：

情形 1——WILC 器件充当组所有者，手机充当 P2P 客户端

将 WILC 器件配置为组所有者：

1. 使用以下命令载入两个 WILC 模块：

```
modprobe wilc-sdio
echo <mode> > /sys/wilc/p2p_mode
```

其中，mode = 1 表示 P2P 组所有者，mode = 0 表示 P2P 客户端。

2. 使用以下命令启动 WPA 客户端服务并打开 P2P 设备：

```
wpa_supplicant -Dnl80211 -ip2p0 -c/etc/wilc_p2p_supplicant.conf &
```

3. 使用以下命令配置 P2P 组所有者的 IP 地址并启动 DHCP 服务器：

```
ifconfig p2p0 192.168.0.1
/etc/init.d/S80dhcp-server start
```

4. 在终端上，使用以下命令进入 wpa_cli 交互模式：

```
wpa_cli -ip2p0
```

5. 使用以下命令在一段指定时间内扫描附近的 P2P 设备：

```
p2p_find <scan_duration_in_seconds>
```

6. 扫描完成后，使用以下命令列出可用的 P2P 对等设备：

```
p2p_peers
```

此命令将列出 P2P 对等设备的 BSSID。

7. 使用以下命令通过 P2P 对等设备的 BSSID 连接到 P2P 客户端：

```
p2p_connect <MAC_ADDRESS> pbc
```

将手机配置为 P2P 客户端：

在手机的 Wi-Fi 设置菜单中，进入 Wi-Fi Direct®模式并执行以下操作以建立连接。

- 从 WILC 触发连接：
  1. 在 WILC 上输入无超时值的 `p2p_find` 命令。
     手机上随即会显示 P2P 对等设备的 SSID。
  2. 在 WILC 上输入上文所示的 `p2p_connect` 命令。手机上随即会弹出一个窗口。
  3. 单击 Accept（接受）按钮或提示以完成连接。
- 从手机触发连接：
  1. 单击手机上显示的 SSID 并发送 P2P 邀请。
  2. 在 WILC 上输入 `p2p_connect <MAC_ADDRESS> pbc` 命令以组成 P2P 组。

情形 2——WILC 器件充当 P2P 客户端，手机充当组所有者

将 WILC 器件配置为 P2P 客户端：

1. 使用以下命令载入两个 WILC 模块：

```
modprobe wilc-sdio
```

2. 使用以下命令启动 WPA 客户端服务并打开 P2P 设备：

```
wpa_supplicant -Dnl80211 -ip2p0 -c/etc/wilc_p2p_supplicant.conf &
```

3. 在终端上，使用以下命令进入 wpa_cli 交互模式：

```
wpa_cli -ip2p0
```

4. 使用以下命令在一段指定时间内扫描附近的 P2P 设备：

```
p2p_find <scan_duration_in_seconds>
```

5. 扫描完成后，使用以下命令列出可用的 P2P 对等设备：

```
p2p_peers
```

此命令将列出 P2P 对等设备的 BSSID。

6. 使用以下命令通过 P2P 对等设备的 BSSID 连接到 P2P 组所有者：

```
p2p_connect <MAC_ADDRESS> pbc go_intent=1
```

7. 按下 Ctrl+C 退出交互模式。
8. 在 WILC 上运行 DHCP 客户端以获取 IP 地址。

```
dhcpcd p2p0 &
```

将手机配置为组所有者：

在手机的 Wi-Fi 设置菜单中，进入 Wi-Fi Direct 模式并执行以下操作以建立连接。

- 从 WILC 触发连接：
  1. 在 WILC 上输入无超时值的 p2p_find 命令。
     手机上随即会显示 P2P 对等设备的 SSID。
  2. 在 WILC 上输入上文所示的 p2p_connect 命令。手机上随即会弹出一个窗口。
  3. 单击 Accept 按钮或提示以完成连接。
- 从手机触发连接：
  1. 单击手机上显示的 SSID 并发送 P2P 邀请。
  2. 在 WILC 上输入 p2p_connect <MAC_ADDRESS> pbc 命令以组成 P2P 组。

8.8 支持同时执行的模式

ATWILC 器件支持同时执行以下模式。

- STA - STA（见*以站点模式运行 ATWILC* 一节）
- STA - P2P 客户端（见*以站点模式运行 ATWILC* 一节和*将 WILC 器件配置为 P2P 客户端*部分）
- STA - P2P GO（见*以站点模式运行 ATWILC* 一节和*将 WILC 器件配置为组所有者*部分）
- AP - P2P 客户端（见*以 AP 模式运行 ATWILC* 一节和*将 WILC 器件配置为 P2P 客户端*部分）
- STA - AP（见*同时以站点模式和 AP 模式运行 ATWILC 器件*一节）
  注： 同时使用 Wlan0 和 p2p0 接口运行 ATWILC 器件。

8.8.1 同时以站点模式和 AP 模式运行 ATWILC 器件

本节介绍同时以站点（STA）和 AP 模式运行 ATWILC 器件的配置步骤。

1. 使用以下命令初始化 ATWILC1000 和 ATWILC3000 驱动程序模块：

```
Welcome to Buildroot
buildroot login: root
[root@buildroot ~]# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been
warned.
linux_sdio_probe init_power =0
wilc_sdio mmc0:0001:1: Driver Initializing success
```

2. 使用以下命令启动 WPA 客户端服务并执行 wpa_supplicant：

```
# wpa_supplicant -Dnl80211 -iwlan0 -c/etc/wilc_wpa_supplicant.conf &
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control dev
wilc_sdio mmc0:0001:1 wlan0: Detect chip WILC3000
wilc_sdio mmc0:0001:1 wlan0: loading firmware wilc3000_wifi_firmware.bin
wilc_gnrl_async_info_received
wilc_sdio mmc0:0001:1 wlan0: WILC Firmware Ver = WILC_WIFI_FW_REL_15_00 Build: 8719
```

3. 使用以下命令连接到接入点：

```
#wpa_cli  -p/var/run/wpa_supplicant ap_scan 1
#wpa_cli  -p/var/run/wpa_supplicant add_network
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 ssid  '"User_AP"'
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE
```

```
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 psk  ‘” 12345” ’
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
#wpa_cli  -p/var/run/wpa_supplicant set_network 0 auth_alg SHARED
#wpa_cli  -p/var/run/wpa_supplicant select_network 0
```

4. 运行 DHCP 服务。
   如果可从 AP 自动分配 IP 地址，应使用以下命令启动 DHCP 客户端：

```
#dhcpcd wlan0 &
```

5. 使用以下命令对 **User AP** 执行 ping 操作以检查连接：

```
# ping 192.168.0.1
```

6. 将 hostapd 作为用户配置运行。

```
# hostapd /etc/wilc_hostapd_open.conf -B &

Configuration file: /etc/wilc_hostapd_open.conf
rfkill: Cannot open RFKILL control device
wilc_sdio mmc0:0001:1 wlan0: Detect chip WILC3000
wilc_sdio mmc0:0001:1 wlan0: loading firmware wilc3000_wifi_firmware.bin
wilc_gnrl_async_info_received
wilc_sdio mmc0:0001:1 wlan0: WILC Firmware Ver = WILC_WIFI_FW_REL_15_00 Build: 8719
Using interface wlan0 with hwaddr fa:f0:05:f6:56:6a and ssid "wilc_SoftAP"
wilc_gnrl_async_info_received
wilc_sdio mmc0:0001:1 wlan0: there is no current Connect Request
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

7. 运行 DHCP 服务器，为客户端分配 IP。
   - 设置 AP 的 IP；`#ifconfig p2p0 192.168.0.1`
   - 启动 DHCP 服务器；`#/etc/init.d/S80dhcp-server start`
     用户现在即可将 PC 或智能手机连接到 ATWILC1000 AP。

8.9 节能

8.9.1 Wi-Fi 节能

Wi-Fi 节能状态可通过内核或命令行来控制。要更改默认节能状态，可通过定义 `CONFIG_CFG80211_DEFAULT_PS` 在 WLAN 接口初始化时使能节能，也可通过取消定义在初始化时禁止节能。要在 WLAN 接口初始化后手动控制节能，应使用 iw 工具。

```
$ iw dev wlan0 set power_save on
```

注： 对于 AP 和 P2P 模式，默认情况下禁止节能模式。

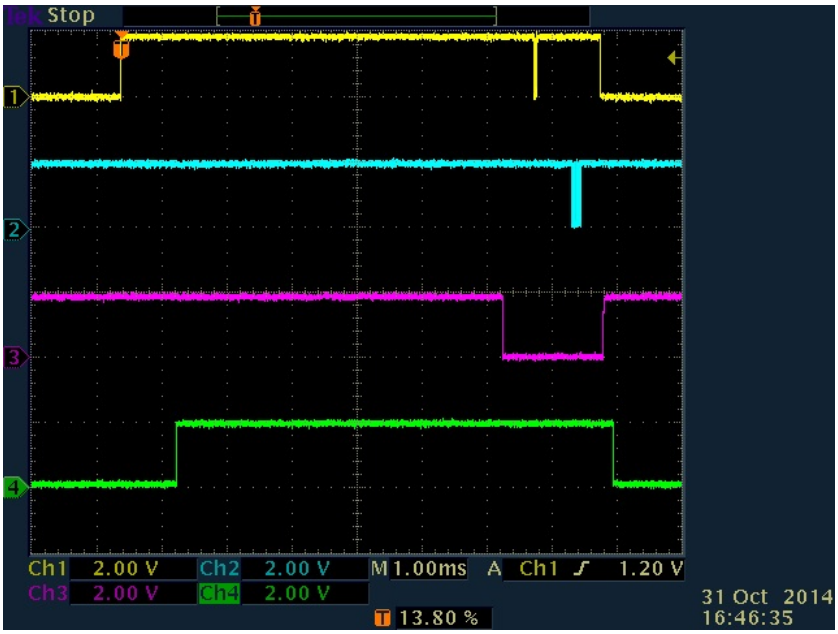
8.9.2 BLE 节能

要使用 BLE 节能模式，应使能 UART 流控制，以阻止主机向处于休眠模式的 ATWILC3000 BLE 控制器发送新命令。

这可以通过以下方式实现：使用更新 UART 参数供应商特定的 HCI 命令在 ATWILC3000 上使能流控制，然后更新主机 UART 配置以使能流控制。此外，主机应用程序应允许 ATWILC3000 BLE 控制器进入节能模式，方法是将主机的 UART Tx 线设置为低电平，以进入断路模式。在启动任何 HCI 通信之前，应用程序应使主机的 UART 退出断路模式，然后继续将 HCI 命令发送到 ATWILC3000。

当 ATWILC3000 处于节能模式时，它会将 UART RTS 线设置为高电平，以阻止主机继续发送任何 HCI 命令。一旦主机 UART Tx 线恢复高电平，ATWILC3000 便会退出节能模式，但不会立即完全进入工作状态。在 ATWILC3000 顺利启动并准备好接收更多 HCI 命令之后，会将 UART RTS 线设置为低电平，随后主机便可发送更多 HCI 命令。

该过程如下图所示：



1. 黄色：UART Rx（ATWILC3000 视角） 2. 蓝色：UART Tx 3. 紫色：UART RTS 4. 绿色：ATWILC3000 就绪

要控制断路模式，可按如下所示使用 IOCTL：

```
int main(int argc,char *argv[])
{
    int fd, serial;

    fd = open("/dev/ttyS1", O_RDWR);
    if(atoi(argv[1])==1) {
        printf("assert on %d\n",fd);
        ioctl(fd, TIOCCBRK, 0);
    } else if(atoi(argv[1])==0) {
        printf("deassert on %d\n",fd);
        ioctl(fd, TIOCSBRK, 0);
    }
     close(fd);
}
```

此类应用程序的示例位于 etc/uart_brk_ioctl 下的参考镜像中。要使能节能模式，可使用以下命令：

```
# modprobe wilc-sdio.ko
# echo BT_POWER_UP > /dev/wilc_bt
# echo BT_DOWNLOAD_FW > /dev/wilc_bt
# hciattach ttyS1 any 115200 noflow
# hciconfig hci0 up
# hcitool cmd 0x3F 0x0053 00 C2 01 00 01
# stty -F /dev/ttyS1 crtscts
# /etc/etc/uart_brk_ioctl
```

要禁止断路模式并唤醒 ATWILC3000，可使用以下命令：

```
# /etc/etc/uart_brk_ioctl
```

8.10 天线切换

ATWILC 器件支持天线分集，即通过外部天线开关将两个天线连接到芯片。

用户使用两个输入信号来控制天线开关，以此选择工作天线。用户可使用以下两种配置来控制 GPIO：

1. 双 GPIO——使用 ATWILC 器件的两个不同 GPIO 分别控制天线开关的两条控制线。
2. 单 GPIO——使用 ATWILC 器件的单个 GPIO 来控制开关的其中一条控制线，同时将该 GPIO 取反后连接到另一条控制线。此配置需要外接逆变器。天线选择算法会评估每秒平均 RSSI，并依此确定是否需要切换天线。

平均 RSSI 基于接收每个数据包时读取的 RSSI 来计算。如果平均 RSSI 低于阈值，则切换到另一个天线并将新阈值设置为停用天线的平均 RSSI。为避免发生不必要的切换，应仅在 RSSI 低于-30 dBm 时切换天线，同时留出 1 dBm 的裕量以避免滞后。

可使用 Sysfs 条目配置天线分集模式下的 ATWILC 器件驱动程序，以及用于在运行时控制天线开关的 GPIO。

8.10.1 天线开关 GPIO 控制

可按如下所示使用 Sysfs 条目/sys/wilc/ant_swtch_mode 来配置用于控制天线开关的 GPIO：

```
# echo mode > /sys/wilc/ant_swtch_mode
```

其中，mode = 1 表示单天线，mode = 2 表示双天线，mode = 0 表示禁止天线分集。

对于 WILC1000，有效 GPIO 包括 0、1、3、4 和 6；对于 WILC3000，有效 GPIO 包括 0、3、4、6、17、18、19 和 20。

8.10.2 GPIO

要配置连接到天线开关的 GPIO，可按如下所示使用 sysfs 条目/sys/wilc/antenna1 和/sys/wilc/antenna2。

```
# echo GPIO_NUM > /sys/wilc/antenna1 ( for single antenna switch)
# echo GPIO_NUM > /sys/wilc/antenna2 ( for dual antenna switch)
```

其中，GPIO_NUM 是适用于天线分集的有效 GPIO。

ATWILC1000 的有效 GPIO 包括 0、1、4 和 6。

ATWILC3000 的有效 GPIO 包括 3、4、17、18、19 和 20。

8.10.3 天线选择

可使用 iw 工具选择工作天线：既可采用固定手动模式（天线 1 或天线 2），也可根据天线性能自动切换，具体如下：

- 使用以下命令设置天线 1：

  ```
  iw phy phy3 set antenna 1 1
  ```

- 使用以下命令设置天线 2：

  ```
  iw phy phy3 set antenna 2 2
  ```

- 使用以下命令使能自动切换：

  ```
  iw phy phy3 set antenna 3 3
  ```

  注： 由于 WILC 公开两个 phy 设备，因此这两个器件均可用于设置天线选择，不过它们会应用同一种选择。此外，设置天线选择之前，应配置天线开关控制 GPIO。

在手动模式下，应根据下表设置 GPIO。

表 8-4. 单天线模式

选择的天线	GPIO1 值
天线 1	1
天线 2	0

表 8-5. 双天线模式

选择的天线	GPIO1 值	GPIO2 值
天线 1	1	0
天线 2	0	1

8.11 调试日志

ATWILC 驱动程序从 Linux 继承调试日志级别。要更改系统的调试级别，应使用以下方法之一：

```
#echo "7" > /proc/sys/kernel/printk
```

其中，“7”是所需的最高日志级别

或

```
# dmesg -n 7
```

要更改编译内核时的默认级别，应更改 kernel_src/include/linux/printk.h 中的以下行

```
#define CONSOLE_LOGLEVEL_DEFAULT 7
```

ATWILC 驱动程序还可使用 debugfs 来允许用户控制为哪些代码区域使能或禁止日志。

在执行此更改之前，用户必须先挂载 debugfs：

```
# mount -t debugfs nodev /sys/kernel/debug
```

随即会回显一个数字，此数字表示用户想要使能日志的区域所对应的位域。位域的定义如下：

```
BIT 0: GENERIC
BIT 1: HOSTAPD
BIT 2: HOSTINF
BIT 3: CORECONFIG
BIT 4: CFG80211
BIT 5: INT
BIT 6: TX
BIT 7: RX
BIT 8: TCP
BIT 9: INIT
BIT 10: PWRDEV
```

8.12 监视器模式

可使用以下命令在 Linux 上使能监视器模式：

```
# modeprobe wilc-sdio.ko
# ifconfig wlan0 up
# iw dev wlan0 set type monitor
# iw dev wlan0 set freq <freq> // eg. 2437 for channel 6
```

随后可使用捕捉工具转储接口上收到的数据包。下面以使用 tcpdump 为例：

```
# tcpdump -i wlan0 -n -w packets_dump.cap
```

注： 要使用 tcpdump，必须先在 *Target Packages* > *Network*（目标包 > 网络）下 buildroot 的 menuconfig 中将其使能。

8.13 其他 Linux 主题

本节介绍有关 Linux 主题的其他信息。

8.13.1 主机暂停/恢复机制

一旦暂停，Linux 版本 4.9 便会与接入点断开连接。要在暂停后保持连接，应修改 Linux 代码，即从\net\wireless\sysfs.c 文件中删除以下代码。

```
//Prevent disconnecting from connected AP's on suspension
        //if (!rdev->wiphy.wowlan_config)
            //cfg80211_leave_all(rdev);
```

以下为\net\wireless\sysfs.c 文件的示例：

```
static int wiphy_suspend(struct device *dev, pm_message_t state)
{
    struct cfg80211_registered_device *rdev = dev_to_rdev(dev);
    int ret = 0;

    rdev->suspend_at = get_seconds();
rtnl_lock();
    if (rdev->wiphy.registered) {
        //Prevent disconnecting from connected AP's on suspension
        //if (!rdev->wiphy.wowlan_config)
            //cfg80211_leave_all(rdev);
        if (rdev->ops->suspend)
            ret = rdev_suspend(rdev, rdev->wiphy.wowlan_config);
        if (ret == 1) {
            /* Driver refuse to configure wowlan */
            cfg80211_leave_all(rdev);
            ret = rdev_suspend(rdev, NULL);
        }
    }
    rtnl_unlock();
return ret;
}
```

用户可使用/sys/power/state 路径中的 mem 字符串将 Linux 配置为暂停模式。有关更多信息，请参见 https://www.kernel.org/doc/Documentation/power/interface.txt。

控制器会在无线 LAN 触发唤醒（可使用 iw 工具配置）时唤醒主机，随后在主机板（与 ATWILC 器件板的 IRQ 引脚相连）上的专用唤醒通用输入/输出（General Purpose Input/output，GPIO）引脚上发出唤醒信号。

ATWILC 只支持在无线唤醒（Wake on Wireless，WoW）模式下将一组允许的唤醒触发信号配置为 ANY。如果用户设置了触发信号，当主机从连接的接入点收到任何类型的数据包时，会唤醒 ATWILC 器件。如果用户未设置触发信号，则控制器不会唤醒主机。

要将主机唤醒触发信号配置为 ANY，应使用以下 any 命令参数：

```
#iw phy0 wowlan enable any
```

其中，phy0 为无线硬件接口名称，any 为所需的触发信号。

要禁止所有触发信号，应使用 disable 参数，如以下命令所示：

```
#iw phy0 wowlan disable
```

要显示已配置的触发信号，应使用 show 参数，如以下命令所示：

```
#iw phy0 wowlan show
```

要将主机配置为暂停模式，应使用以下命令：

```
#echo mem > /sys/power/state
```

8.13.2 设置发送功率

用户可使用 iw 工具通过以下命令行参数来控制 ATWILC1000 或 ATWILC3000 的发送功率。

```
$ iw dev wlan0 set txpower fixed x
```

其中，x 是所需发送级别。

支持的发送级别包括 0、3、6、9、12、15 和 18。

注： 如果输入发送功率值不在上述支持的发送级别之列，则 x 值会自动设为其中最接近的较大值。

8.13.3 扫描

要扫描可用的 AP，应使用$ wpa_cli scan 命令。

8.13.4 获取扫描结果

要获取已识别的 AP 及 bssid、频率、接收信号强度指示符（Received Signal Strength Indicator，RSSI）、加密和服务集标识符（SSID）等相关属性的列表，应使用以下命令：

```
$ wpa_cli scan_results
Selected interface 'wlan0'
bssid / frequency / signal level / flags / ssid
02:1a:11:f5:56:81       2437    -54     [ESS]   AndroidAP
68:7f:74:c7:4e:d9       2462    -54     [WPA2-PSK-CCMP][WPS][ESS]       IOT_58
d8:fe:e3:03:4e:30       2422    -54     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    dlink-
enterprise
00:0c:43:44:0a:b4       2437    -51     [ESS]   RT2880_AP
```

8.13.5 保存网络信息

为避免重启后丢失网络信息，应使用$ wpa_cli save_config 命令。

8.13.6 载入网络信息

要在重启后获取保存的网络信息，应使用$ wpa_cli list_networks 命令。

8.13.7 获取当前网络信息

要获取网络的连接接口信息（包括 RSSI、信道和加密等），应使用以下命令：

```
$ iwconfig wlan0
DBG [WILC_WFI_get_tx_power: 3418]Got tx power 18
wlan0     IEEE 802.11bgn  ESSID:"AndroidAP"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 02:1A:11:F5:56:81
          Bit Rate=0 kb/s   Tx-Power=18 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=49/70  Signal level=-61 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

8.13.8 更改监管域设置

内核的中央监管域代理（Central Regulatory Domain Agent，CRDA）充当监管内核通信和合规性的 udev 帮助程序。参考平台上默认使能 CRDA。要为其他平台使能 CRDA，必须使用 menuconfig 在 buildroot 的目标包中选择 CRDA：

Target Packages > Networking applications > crda（目标包 > 网络应用程序 > crda）

CRDA 将数据库（指定每个国家/地区使用的信道）与“世界监管域”限制配合使用。此数据库在 wireless-regdb 包内的 db.txt 文件中定义。“世界监管域”会根据国家/地区设置某些限制，令设备在此限制下运行，即使用户在没有这些限制的国家/地区使用设备，限制依然有效。更多详细信息，请参见 https://wireless.wiki.kernel.org/en/developers/Regulatory/CRDA#Using_iw_to_change_regulatory_domains。

Linux 允许按照世界监管限制（包括 US FCC）更改监管域。为此，设备应严格遵守其设定的监管域，国家/地区代码选择会进一步强化这些监管限制。这符合 FCC 第 15 部分国家代码选择知识库（出版号 594280）中的规定。例如，如果设备被设计为在美国（允许使用信道 1-11，2.4 GHz 频段）使用，当用户将设备带到日本（允许使用信道 1-14）并将监管域更改为 JP 时，将不能使用信道 12、13 或 14（CCK）。但是，如果设备被设计为在日本使用，当用户将设备带到美国并选择 US 作为监管域时，也会禁止信道 12-14。

默认数据库将信道 12-14 限制为仅监听；因此，应将这些信道用于 AP 模式。例如，必须移除标志 NO-IR。

1. 这是世界监管域国家/地区 00：（2402-2472 @ 40），（20）
2. 信道 12-13。（2457-2482 @ 20），（20），AUTO-BW
3. 信道 14。只有 JP 允许使用此信道，且仅限 802.11b（2474-2494 @ 20），（20），NO-OFDM
4. 信道 36-48（5170-5250 @ 80），（20），NO-IR，AUTO-BW

5. 信道 52-64（5250-5330 @ 80），（20），NO-IR，DFS，AUTO-BW
6. 信道 100-144（5490-5730 @ 160），（20），NO-IR，DFS
7. 信道 149-165（5735-5835 @ 80），（20），NO-IR
8. IEEE 802.11ad（60 GHz），信道 1..3（57240-63720 @ 2160），（0）

生成新的监管数据库二进制文件

监管域数据库二进制文件需经过数字签名来保证完整性；因此，要生成新的数据库二进制文件，还必须在编译 CRDA 时使用密钥并将其复制到目标。要创建新的监管文件，请按以下步骤操作：

1. 打开已编译的 buildroot。
2. 转至 `output/build/wireless-regdb-2017.03.07/`。
3. 更改 `db.tx`。
4. 使用 `make` 命令编译 regdb。
   随即会生成一个新的公钥，可用于生成新的 `regulartory.bin` 文件以及对此文件进行签名。用户必须安装 m2crypto Python®包才能编译 `regdb # sudo apt-get install python-m2crypto`。
5. 将文件复制到 `output/build/crda-3.18/pubkeys/`。
6. 修改 wireless-regdb 包以将新密钥安装到目标，具体操作如下：
   - 转至 `wireless-regdb.mk`。
   - 编辑 `WIRELESS_REGDB_INSTALL_TARGET_CMDS` 以将新密钥复制到目标文件夹。
7. 删除`.stamp_target_installed`、`.stamp_built from output/build/crda-3.18` 和 `wireless-regdb-2017.03.07`，针对 crda 和 wireless-regdb 强制执行重新编译并安装到目标的操作。
8. 重新编译 buildroot。

要验证此过程，应使用 regdbdump 确保新的 `regulatory.bin` 可接受验证。

```
# regdbdump /usr/lib/crda/regulatory.bin
```

8.13.9 获取当前监管域

要获取已识别的 AP 及 bssid、频率、RSSI、加密和 SSID 等相关属性的列表，应使用以下命令：

```
$ iw reg get
country EG: DFS-UNSET
        (2402 - 2482 @ 40), (N/A, 20)
        (5170 - 5250 @ 80), (N/A, 20)
        (5250 - 5330 @ 80), (N/A, 20), DFSiwconfig wlan0
```

8.13.10 设置当前监管域

要获取已识别的 AP 及 bssid、频率、RSSI、加密和 SSID 等相关属性的列表，应使用以下命令：

```
$ iw reg set US
cfg80211: Calling CRDA for country: US
[root@buildroot ~]# cfg80211: Regulatory domain changed to country: US
cfg80211:  DFS Master region: unset
cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp), (dfs_cac_time)
cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 3000 mBm), (N/A)
cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz), (N/A, 1700 mBm), (N/A)
cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz), (N/A, 2300 mBm), (0 s)
cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 3000 mBm), (N/A)
cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 4000 mBm), (N/A)
```

要更改 Linux 在启动时使用的默认监管域，用户必须使用 vi 工具编辑先前启动 `wpa_cli` 时已传送的配置文件。配置如下：

```
$ vi /etc/wilc_wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
country=US

network={
        ssid="AndroidAP"
```

```
        key_mgmt=NONE
}
```

8.14 以蓝牙模式运行 ATWILC3000

使用以下命令在载入 wilc-sdio.ko 模块后使用 BLE。

WILC3000 初始化时会在/dev/wilc_bt 下创建一个节点，此节点可用于写入以下命令：

- BT_POWER_UP
- BT_DOWNLOAD_FW
- BT_FW_CHIP_WAKEUP
- BT_FW_CHIP_ALLOW_SLEEP
- BT_POWER_DOWN

8.14.1 BT_POWER_UP

以下命令用于为芯片上电，并指示 BT 需要芯片上电。

```
$ echo BT_POWER_UP > /dev/wilc_bt
```

8.14.2 BT_DOWNLOAD_FW

以下命令使用 SDIO 下载 BT 固件。

```
$ echo BT_DOWNLOAD_FW > /dev/wilc_bt
```

8.14.3 BT_FW_CHIP_WAKEUP

以下命令用于阻止芯片休眠。

```
$ echo BT_FW_CHIP_WAKEUP > /dev/wilc_bt
```

此命令在使用通用异步收发器（Universal Asynchronous Receiver/Transmitter，UART）下载固件之前使用。否则，当协议栈下载 BT 固件时，芯片可能会进入休眠模式。

8.14.4 BT_FW_CHIP_ALLOW_SLEEP

以下命令指定 at_pwr_dev 模块不需要芯片唤醒。用户必须在使用 UART 下载和启动 BT 固件之后使用此命令，从而允许 BT 和 Wi-Fi 固件做出休眠或唤醒决策。

```
$ echo BT_FW_CHIP_ALLOW_SLEEP > /dev/wilc_bt
```

8.14.5 BT_POWER_DOWN

以下命令用于在不使用 BT 时使芯片掉电。

```
$ echo BT_POWER_DOWN > /dev/wilc_bt
```

如果 Wi-Fi 处于工作状态，则无法使用 BT_POWER_DOWN 命令使芯片掉电。但是，如果用户以正确的顺序使用 BT_POWER_UP 和 BT_POWER_DOWN，便可以成功为芯片上电和掉电。

8.14.6 连接蓝牙的 UART

ATWILC3000 蓝牙驱动程序提供 UART 接口，并通过电传打字机（TTY）设备连接。它连接到 BlueZ 协议栈。

以下命令用于连接设备。确保目标平台上具有/dev/ttyS1 文件夹。用户必须将蓝牙固件波特率设置为 115200，并应使能无流控制。

```
$ hciattach ttyS1 any 115200 noflow
```

确保已创建主机控制接口（Host Control Interface，HCI）。

```
$ hciconfig -a
hci0:     Type: BR/EDR  Bus: UART
```

```
    BD Address: AB:89:67:45:23:01  ACL MTU: 1021:9  SCO MTU: 255:4
    DOWN
    RX bytes:574 acl:0 sco:0 events:27 errors:0
    TX bytes:411 acl:0 sco:0 commands:27 errors:0
    Features: 0xff 0xff 0xcd 0xfe 0xdb 0xff 0x7b 0x87
    Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
    Link policy: RSWITCH HOLD SNIFF PARK
    Link mode: SLAVE ACCEPT
```

8.14.7 使能蓝牙接口

使用以下命令使能 ATWILC3000 蓝牙 HCI 接口。

```
$ hciconfig hci0 up
```

8.14.8 运行 bluetoothd（蓝牙守护进程）

用户必须按如下所示为 bluetoothd 创建符号链接：

```
$ ln -svf /usr/libexec/bluetooth/bluetoothd /usr/sbin
```

使用$ bluetoothd -n &命令在后台启动蓝牙守护进程。

8.14.9 扫描设备

用户可使用$ scan on 命令扫描附近的网络。扫描完成后，此命令将显示一个包含蓝牙地址（BD_ADDR）和名称的网络列表。

使用$ bluetoothctl 命令启动 **bluetoothctl**，可用于扫描和连接。

以下是启动扫描时的示例：

```
$ scan on
Scanning ...
    60:6C:66:A4:29:63    D247-PC
    60:03:08:89:93:E7    damiank-mbp1
    E0:06:E6:BE:A8:FA    APDN194
    78:DD:08:B2:91:C9    ALEX-PC
```

8.14.10 连接设备

建议使用 DBUS 接口来连接扫描期间发现的设备。

可使用 connect 命令来连接具有指定蓝牙地址的设备。

例如，要连接到蓝牙地址 00:02:3C:3A:95:6F，应使用以下命令：

```
$ connect 00:02:3C:3A:95:6F
```

8.14.11 BlueZ 5.28 及以下版本的 BLE 外设模式示例

可使用 BlueZ 通过低功耗通告命令（leadv）实现以 BLE 外设模式运行。此外，还可使用蓝牙守护进程（bluetoothd）通过以下命令提供时间配置文件：

```
[root@buildroot ~]# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been warned.
linux_sdio_probe init_power =0
wilc_sdio mmc0:0001:1: Driver Initializing success
[root@buildroot ~]# mmc0: card 0001 removed
mmc0: new high speed SDIO card at address 0001
linux_sdio_probe init_power =1
wilc_sdio mmc0:0001:1: Driver Initializing success
# echo BT_SDIO_INIT > /dev/wilc_bt
[root@buildroot ~]# echo BT_POWER_UP > /dev/wilc_bt
[root@buildroot ~]# echo BT_FW_CHIP_WAKEUP > /dev/wilc_bt
[root@buildroot ~]# echo BT_DOWNLOAD_FW > /dev/wilc_bt
[root@buildroot ~]# echo BT_FW_CHIP_ALLOW_SLEEP > /dev/wilc_bt
[root@buildroot ~]# hciattach ttyS1 any 115200 noflow
atmel_usart fc010000.serial: using dma0chan10 for rx DMA transfers
atmel_usart fc010000.serial: using dma0chan11 for tx DMA transfers
```

```
Device setup complete
[root@buildroot ~]# hciconfig hci0 up
[root@buildroot ~]# g_serial gadget: high-speed config #2: CDC ACM config
ln -svf /usr/libexec/bluetooth/bluetoothd /usr/sbin
'/usr/sbin/bluetoothd' -> '/usr/libexec/bluetooth/bluetoothd'
[root@buildroot ~]# bluetoothd -p time -n &
[1] 845
[root@buildroot ~]# bluetoothd[845]: Bluetooth daemon 5.21
bluetoothd[845]: Starting SDP server
bluetoothd[845]: Ignoring (cli) hostname
bluetoothd[845]: Ignoring (cli) wiimote
bluetoothd[845]: Ignoring (cli) autopair
bluetoothd[845]: Ignoring (cli) policy
bluetoothd[845]: Ignoring (cli) neard
bluetoothd[845]: Ignoring (cli) sap
bluetoothd[845]: Ignoring (cli) a2dp
bluetoothd[845]: Ignoring (cli) avrcp
bluetoothd[845]: Ignoring (cli) network
bluetoothd[845]: Ignoring (cli) input
bluetoothd[845]: Ignoring (cli) hog
bluetoothd[845]: Ignoring (cli) health
bluetoothd[845]: Ignoring (cli) gatt
bluetoothd[845]: Ignoring (cli) scanparam
bluetoothd[845]: Ignoring (cli) deviceinfo
bluetoothd[845]: Ignoring (cli) alert
bluetoothd[845]: Ignoring (cli) proximity
bluetoothd[845]: Ignoring (cli) thermometer
bluetoothd[845]: Ignoring (cli) heartrate
bluetoothd[845]: Ignoring (cli) cyclingspeed
bluetoothd[845]: Failed to open RFKILL control device
bluetoothd[845]: Bluetooth management interface 1.14 initialized

[root@buildroot ~]# hciconfig -a
hci0: Type: BR/EDR Bus: UART
BD Address: F8:F0:05:F7:36:9E ACL MTU: 1021:9 SCO MTU: 255:4
UP RUNNING PSCAN
RX bytes:1257 acl:0 sco:0 events:67 errors:0
TX bytes:1381 acl:0 sco:0 commands:67 errors:0
Features: 0xff 0xff 0xcd 0xfe 0xdb 0xff 0x7b 0x87
Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
Link policy: RSWITCH HOLD SNIFF PARK
Link mode: SLAVE ACCEPT
Name: 'BlueZ 5.21'
Class: 0x000000
Service Classes: Unspecified
Device Class: Miscellaneous,
HCI Version: 4.0 (0x6) Revision: 0x709
LMP Version: 4.0 (0x6) Subversion: 0x709
Manufacturer: Atmel Corporation (19)
[root@buildroot ~]# hciconfig hci0 leadv
```

8.14.12 BlueZ 5.29 及以上版本的 BLE 外设模式示例

从 blueZ 5.29 版本开始，不再支持使用 bluetoothd 提供时间配置文件。替代方案是使用在编译 blueZ 包时自动编译的 btgatt-server 示例。但请注意，buildroot 在默认情况下不会将此示例安装到目标，应使用 scp 或 rz 手动将其传输到主机。

要自动安装此示例，应按以下步骤修改 buildroot 系统中 blueZ 的.mk 文件：

1. 编译文件 buildroot/package/bluez5_utils/bluez5_utils.mk。
2. 在文件末尾$(eval $(autotools-package))之前添加以下内容：

```
define BLUEZ5_UTILS_INSTALL_GATTEXAMPLE
        $(INSTALL) -D -m 0755 $(@D)/tools/btgatt-server $(TARGET_DIR)/usr/bin/btgatt-
server
endef
BLUEZ5_UTILS_POST_INSTALL_TARGET_HOOKS += BLUEZ5_UTILS_INSTALL_GATTEXAMPLE
```

要运行此示例，应使用以下命令：

```
# modprobe wilc-sdio
wilc_sdio: module is from the staging directory, the quality is unknown, you have been warned.
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
```

```
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
wilc_sdio mmc0:0001:1: WILC got 60 for gpio_reset
wilc_sdio mmc0:0001:1: WILC got 94 for gpio_chip_en
wilc_sdio mmc0:0001:1: WILC got 91 for gpio_irq
wifi_pm : 0
wifi_pm : 1
wilc_sdio mmc0:0001:1: Driver Initializing success
# wilc_sdio mmc0:0001:1 wlan0: INFO [wilc_netdev_cleanup]Unregistering netdev d4782000
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_netdev_cleanup]Freeing Wiphy...
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_free_wiphy]Unregistering wiphy
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_free_wiphy]Freeing wiphy
wilc_sdio mmc0:0001:1 wlan0 (unregistered): INFO [wilc_netdev_cleanup]Freeing netdev...
wilc_sdio mmc0:0001:1 p2p0: INFO [wilc_netdev_cleanup]Unregistering netdev d477b000
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_netdev_cleanup]Freeing Wiphy...
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_free_wiphy]Unregistering wiphy
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_free_wiphy]Freeing wiphy
wilc_sdio mmc0:0001:1 p2p0 (unregistered): INFO [wilc_netdev_cleanup]Freeing netdev...
Module_exit Done.
at_pwr_dev: deinit
at_pwr_dev: unregistered
mmc0: card 0001 removed
mmc0: new high speed SDIO card at address 0001
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Registering wifi device
(unnamed net_device) (uninitialized): INFO [WILC_WFI_CfgAlloc]Allocating wireless device
(unnamed net_device) (uninitialized): INFO [wilc_create_wiphy]Successful Registering
wilc_sdio mmc0:0001:1: WILC got 60 for gpio_reset
wilc_sdio mmc0:0001:1: WILC got 94 for gpio_chip_en
wilc_sdio mmc0:0001:1: WILC got 91 for gpio_irq
wilc_sdio mmc0:0001:1: Driver Initializing success

# echo BT_POWER_UP > /dev/wilc_bt
at_pwr_dev: open()
AT PWR: bt_power_up
wilc_sdio mmc0:0001:1: SDIO speed: 50000000
wilc_sdio mmc0:0001:1: chipid 003000d0
WILC POWER UP
at_pwr_dev: close()
#
# echo BT_FW_CHIPaWt_pUwr_dev: open()
 > /at_pwwrc_dtev: close()
#
# echo BT_DOWNLOAD_FW > /dev/wilc_bt
at_pwr_dev: open()
AT PWR: bt_download_fw
Bluetooth firmware: mchp/wilc3000_ble_firmware.bin
Downloading BT firmware size = 58276 ...
Starting BT firmware
BT Start Succeeded
at_pwr_dev: close()
#
# echo BT_FW_CHIP_ALLOW_SLEEP   > /dev/wilc_bt
at_pwr_dev: open()
at_pwr_dev: close()
#
# hciattach ttyS1 any 115200 noflow
atmel_usart fc010000.serial: using dma0chan10 for rx DMA transfers
atmel_usart fc010000.serial: using dma0chan11 for tx DMA transfers
Device setup complete
#
# hciconfig hci0 up
#
# hciconfig hci0 leadv
#
# btgatt-server -i hci0 -s low -t public -r -v
Started listening on ATT channel.Waiting for connections
Connect from 49:0D:EA:C2:98:66
NET: Registered protocol family 38
Running GATT server
[GATT server]# att: > 0a 10 00                                          ...
[GATT server]# att: ATT PDU received: 0x0a
```

```
[GATT server]# server: Read Req - handle: 0x0010
[GATT server]# att: ATT op 0x0b
[GATT server]# att: < 0b 01                                              ..
[GATT server]#
```

8.14.13 设置 Wi-Fi MAC 地址

ATWILC 具有一个非易失性存储器，用于保存每个 Wi-Fi 接口的惟一 MAC 地址。

如果 ATWILC 的非易失性存储器中没有 MAC 地址，则主机必须在接口初始化时分配惟一的 MAC 地址。

使用以下 Linux 命令设置 MAC 地址：

```
ifconfig wlan0 up
ifconfig wlan0 hw ether fa:f0:05:f6:53:88
```

（或）

如果 iproute2 实用程序可用，可使用以下命令：

```
ifconfig wlan0 up
ip link set wlan0 address fa:f0:05:f6:53:88
```

用户也可对 p2p0 接口使用上述命令。

9. 文档版本历史

版本	日期	章节	说明
C	2019 年 2 月	• 为 SAMA5D2 Xplained Ultra 板编译 Linux • 编译系统映像并刷写到 SAMA5D3 Xplained 板 • 编译系统映像并刷写到 SAMA5D27-SOM1-EK1 板 • 串行外设接口板 • 监视器模式 • 更改监管域设置 • 设置 Wi-Fi MAC 地址	• 增加了新的章节 • 增加了新的章节 • 增加了新的章节 • 增加了有关 SPI 引脚与 XPRO EXT1 引脚关系的详细信息 • 增加了新的章节 • 增加了新的章节 • 增加了新的章节
B	2018 年 6 月	文档	• 更新了为 SAMA5D4 Xplained Ultra 板编译 Linux 的步骤 • 更新了 ATWILC 固件的更新步骤 • 增加了有关节能、天线切换和调试日志的信息 • 增加了有关 BlueZ 5.28 及以下版本的 BLE 外设模式以及 BlueZ 5.29 及以上版本的 BLE 外设模式的详细信息
A	2017 年 8 月	文档	初始版本

Microchip 网站

Microchip 网站（http://www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 http://www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担

保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 均为 Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

ISBN：978-1-5224-5013-9

AMBA、Arm、Arm7、Arm7TDMI、Arm9、Arm11、Artisan、big.LITTLE、Cordio、CoreLink、CoreSight、Cortex、DesignStart、DynamIQ、Jazelle、Keil、Mali、Mbed、Mbed Enabled、NEON、POP、RealView、SecurCore、Socrates、Thumb、TrustZone、ULINK、ULINK2、ULINK-ME、ULINK-PLUS、ULINKpro、µVision 和 Versatile 均为 Arm Limited（或其子公司）在美国和/或其他国家/地区的商标或注册商标。

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 http://www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： http://www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4450-2828 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820