君正®

T41 WiFi 驱动移植指南

Date：2022-01

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

Trademarks and Permissions

Disclaimer

All the deliverables and data in this folder serve only as a reference for customer development. Please read through this disclaimer carefully before you use the deliverables and data in this folder. You may use the deliverables in this folder or not. However, by using the deliverables and data in this folder, you agree to accept all the content in this disclaimer unconditional and irrevocable. If you do not find the content in this disclaimer reasonable, you shall not use the deliverables and data in this folder.

The deliverables and data in this folder are provided "AS IS" without representations, guarantees or warranties of any kind (either express or implied). To the maximum extent permitted by law, Ingenic Semiconductor Co., Ltd (Ingenic) provides the deliverables and data in this folder without implied representations, guarantees or warranties, including but not limited to implied representations, guarantees and warranties of merchantability, non-infringement, or fitness for a particular purpose. Deviation of the data provided in this folder may exist under different test environments.

Ingenic takes no liability or legal responsibility for any design and development error, incident, negligence, infringement, and loss (including but not limited to any direct, indirect, consequential, or incidental loss) caused by the use of data in this folder. Users shall be responsible for all risks and consequences caused by the use of data in this folder.

北京君正集成电路股份有限公司
地址：北京市海淀区西北旺东路 10 号院东区 14 号楼君正大厦
电话:**(86-10)56345000**
传真:**(86-10)56345001**
Http: //www.ingenic.cn

前言

概述

本文档主要介绍 T41 如何进行 WiFi 移植及移植过程中的问题分析。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
T41	

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修订章节
2022-01	1.0	第一次正式版本发布

目录

1 WiFi 模块的分类

WiFi 是目前使用最多的一种无线网络传输技术之一，广泛应用于各种智能物联网设备中。不同的应用场景对 WiFi 模组的功能要求不同，WiFi 从功能上分为两大类：透传型 WiFi 模组和通用型 WiFi 模组。

透传型 WiFi 模组：通常应用于点到点的数据传输场景，模块内置 WiFi 驱动和协议，用户不需要关心 WiFi 协议是如何实现的，透传型的 WiFi 模块一般用在 MCU 中。

通用型 WiFi 模组：比如笔记本、手机、各种嵌入式设备上的 USB 或 SDIO 接口的 WiFi 模块，WiFi 协议栈和驱动工作在 Windows、Android、Linux 等操作系统中，本文主要介绍通用型 WiFi 模组如何移植到君正平台。

目前君正 T 系列芯片支持的 WiFi 型号如下：

USB 接口	RTL8188FTV、RTL8188EUS、RTL8188ETV、RTL8723BU
	MT7601
	ATBM6022
	RDA5995
SDIO 接口	RTL8189ES、RTL8189ETV、RTL8189FTV
	MARVELL8801
	BCM6212A
	SSV6x5x
	Hi3881V100

注：上述 WiFi 模组为君正客户在实际项目中与君正工程师一起联调的。

2 WiFi 的内核配置

对于 SDIO/USB 接口的 WiFi，首先它是一个 SDIO/USB 设备，然后具备了 WiFi 的功能。所以设备注册的时候是先以 SDIO 卡/USB 设备去注册的，当检测到设备时再驱动设备的 WiFi 功能，之后再通过 SDIO 协议/USB 协议发送命令和数据与 WiFi 模组进行交互。

因此，我们首先需要使能内核的 SDIO/USB 的功能，这样才能识别到 WiFi 设备。

T21/T30/T31/T41 使用的内核版本为 Kernel-3.10.14，T40/T41 使用的内核版本为 Kernel-4.4.94。针对不同的内核版本，内核的配置也有不同。

2.1 Kernel-3.10.14 配置

2.1.1 USB WiFi 内核配置

对于 USB 接口的 WiFi，只要将 DWC2 控制器配置为 Host 模式即可。

```
Device Drivers  --->
[*] USB support  --->
    <*>   DesignWare USB2 DRD Core Support
       Driver Mode (Host Mode Only)  --->
```

WiFi 模组上电后可以通过 lsusb 命令查看 USB WiFi 模组是否正常识别。

2.1.2 SDIO WiFi 内核配置

T 系列芯片上有两个 msc 接口，但一般情况下 msc0 留给 SD 卡使用，SDIO WiFi 通常使用 msc1 接口，下面就以 msc1 接口为例介绍如何配置内核。

```
Device Drivers  --->
  <*> MMC/SD/SDIO card support  --->
      [*]     MSC for MMC1
          MSC GPIO function pins select (GPIO C, data with 1 bit)
--->
```

T 系列芯片存在多组 GPIO 可以配置成 msc1 功能，在实际项目中根据硬件的设计来配置内核使用哪一组 GPIO 引脚，比如 T31 可以配置为 msc1 的 GPIO 有 PA、PB、PC 三组。

```
( ) GPIO B, Data width 4 bit
( ) GPIO C, Data width 4 bit
(X) GPIO A, Data width 4 bit
```

2.2 Kernel-4.4.94 配置

2.2.3 USB 接口 WiFi 内核配置

对于 USB 接口的 WiFi，只要将 DWC2 控制器配置为 Host 模式即可。

```
Device Drivers  --->
  [*] USB support  --->
```

```
        <*>   Support for Host-side USB
        <*>   DesignWare USB2 DRD Core Support
              DWC2 Mode Selection (Host only mode)  --->
```

WiFi 模组上电后可以通过 lsusb 命令查看 USB WiFi 模组有没有正常识别。

2.2.4 SDIO WiFi 内核配置

T 系列芯片上有两个 msc 接口，但一般情况下 msc0 用作 SD 卡，SDIO WiFi 通常使用 msc1 接口，下面就以 msc1 接口为例介绍如何配置内核。

首先配置内核支持 MSC 功能：

```
  Device Drivers  --->
    <*> MMC/SD/SDIO card support  --->
      <*>   Ingenic(XBurst2)  MMC/SD Card Controller(MSC) supp
ort
```

然后在设备树文件 kernel-4.4.94/arch/mips/boot/dts/ingenic/marmot.dts 中：

```
&msc1 {
    status = "ok";
    pinctrl-names = "default";
    /*mmc-hs200-1_8v;*/
    cap-mmc-highspeed;
    max-frequency = <50000000>;
    bus-width = <4>;
    voltage-ranges = <1800 3300>;
    non-removable;

    ingenic,sdio_clk=<1>;

    /* special property */
    ingenic,wp-gpios = <0>;
    ingenic,rst-gpios = <0>;
    pinctrl-0 = <&msc1_pb>;
};
```

①将 status 配置为 okay，使能 msc1 的功能

② 添加 ingenic,sdio_clk = <1>; 这个属性

③ T41 上有两组 GPIO 可以配制成 msc1 功能，分别为 PB 和 PC，根据具体的硬件设计将 pinctrl-0 配置为 msc1_pb 或 msc1_pc。

3 WiFi 驱动的移植过程

3.1 SDIO WiFi 驱动移植

下面以 RTL8189FTV 为例，讲解一下 SDIO WiFi 的移植过程：

（1）获取 RTL8189FTV 的驱动源码，rtl8189FS_linux_v5.3.16_32695.20190327-ingenic-20190618

（2）移植 SDIO WiFi 驱动

① Makefile 中添加一个 PLATFORM_INGENIC 的宏

```
101 ###################### Platform Related ######################
102 CONFIG_PLATFORM_INGENIC = y
103 CONFIG_PLATFORM_I386_PC = n
104 CONFIG_PLATFORM_ANDROID_X86 = n
105 CONFIG_PLATFORM_ANDROID_INTEL_X86 = n
```

② Makefile 中添加一段关于 SDIO 支持的操作

```
1518 ifeq ($(CONFIG_PLATFORM_INGENIC), y)
1519 EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
1520 EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
1521 ARCH ?= mips
1522 CROSS_COMPILE ?= mips-linux-gnu-
1523 KSRC ?= /home/heng/work/opensource/kernel-4.4.94
1524
1525 ifeq ($(CONFIG_SDIO_HCI), y)
1526 EXTRA_CFLAGS += -DCONFIG_PLATFORM_OPS
1527 _PLATFORM_FILES += platform/platform_ingenic_sdio.o
1528 endif
1529 endif
```

③ 在驱动源码的 platform 目录下创建 platform_ingenic_sdio.c 文件，添加相应的函数。其中与君正平台相关的是 jzmmc_manual_detect(int sdio_index, int on) 函数，该函数用于主动探测 / 移除 SDIO 设备，其中：

(i) sdio_index 参数表示 msc 控制器的编号，T 系列有两个 msc 控制器，对应 0 和 1；

(ii) on 为 1 表示主动探测设备，on 为 0 表示主动移除设备；

(iii) 注意 T2x/T3x 与 T40/T41 添加的头文件不一样

- T2x/T3x 对应 #include <mach/jzmmc.h>
- T40/T41 对应 #include <soc/mmc.h>

特别需要注意的是：该文件中对相关引脚的操作需要跟硬件保持一致。

```
13 #define GPIO_WIFI_WAKEUP        GPIO_PC(17)
14 #define GPIO_WIFI_RST_N         GPIO_PC(16)
15 #define SDIO_WIFI_POWER         GPIO_PC(18)
16 #define WLAN_SDIO_INDEX         1
```

其中 SDIO_WIFI_POWER 是电源引脚，一般通过将 GPIO 配置为 output 模式来控制 WiFi 模组的供电。

上述代码移植结束后，编译代码会生成 8189fs.ko 驱动，然后使用该驱动进行测试。

默认情况下，驱动代码打开了 DEBUG 宏，在调试过程中会产生大量的调试信息，如果想要关闭或减少 DEBUG 信息的输出，可以在 Makefile 中修改如下部分：
CONFIG_RTW_LOG_LEVEL 为 0 ~ 4

```
83 ######################### Debug ##########################
84 CONFIG_RTW_DEBUG = y
85 # default log level is _DRV_INFO_ = 4,
86 # please refer to "How_to_set_driver_debug_log_level.doc" to set the available level.
87 CONFIG_RTW_LOG_LEVEL = 0
```

3.2 SDIO WiFi 功能测试

WIFI 有两种工作模式，AP 模式和 STA 模式，测试在这两种模式下 SDIO WIFI 功能是否正常。

3.2.1 STA 模式网络测试

在该模式下，搭载 WiFi 的 T 系列产品作为无线网络中的一个终端
该模式需要使用的应用程序、配置文件以及动态库如下：
（1）应用程序及配置文件
wpa_supplicant // 负责完成 wifi 认证相关的登录、加密等工作
wpa_supplicant.conf // 配置想要连接的 wifi 热点的名称和密码

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

network={
        ssid="JZ_Guest"
        psk="#wwwingeniccn#"
}
```

udhcpc 用来动态获取 IP 地址、网关、DNS 等信息
default.script 用来自动设置 IP、网关、DNS 等，该文件由 busybox 源码中的 busybox-x.x/examples/udhcp/simple.script 文件重命名而来，使用 udhcpc -h 查看需要将 default.script 放到哪个路径下面。

```
BusyBox v1.22.1 (2020-10-17 11:38:48 CST) multi-call binary.

Usage: udhcpc [-fbqvaRB] [-t N] [-T SEC] [-A SEC/-n]
       [-i IFACE] [-s PROG] [-p PIDFILE]
       [-oC] [-r IP] [-V VENDOR] [-F NAME] [-x OPT:VAL]... [-O OPT]...

       -i,--interface IFACE    Interface to use (default eth0)
       -s,--script PROG        Run PROG at DHCP events (default /usr/share/udhcpc/default.script)
```

（2）使用到的动态库
libcrypto.so.1.0.0 // 加密相关
libiw.so.29 // iw 相关的命令会使用到该库，如 iwlist、iwconfig
libnl.so.1 // hostpad 需要用到的库
libssl.so.1.0.0 // openssl 需要使用到的库
将动态库放到系统相应的 lib 目录下面。
（3）测试流程
① 加载 8189fs.ko，加载成功之后会生成 wlan0 网卡
② 启动网卡 ifconfig wlan0 up

```
[root@Ingenic-uc1_1:wifi_tools]# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 44:01:BB:4C:C0:C2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

③ 将上面的 4 个动态库文件放到/system/lib 目录下

④ ./wpa_supplicant -i wlan0 -Dnl80211 -c wpa_supplicant.conf &

⑤ udhcpc -i wlan0 // 动态获取 IP 地址

```
[root@Ingenic-uc1_1:wifi_tools]# udhcpc -i wlan0
udhcpc (v1.22.1) started
Setting IP address 0.0.0.0 on wlan0
Sending discover...
Sending select for 11.4.30.219...
Lease of 11.4.30.219 obtained, lease time 86400
Setting IP address 11.4.30.219 on wlan0
Deleting routers
route: SIOCDELRT: No such process
Adding router 11.4.30.1
Recreating /etc/resolv.conf
 Adding DNS server 202.102.213.68
 Adding DNS server 114.114.114.114
```

⑥ ping www.baidu.com // 网络正常

```
[root@Ingenic-uc1_1:wifi_tools]# ping www.baidu.com
PING www.baidu.com (14.215.177.38): 56 data bytes
64 bytes from 14.215.177.38: seq=0 ttl=53 time=53.065 ms
64 bytes from 14.215.177.38: seq=1 ttl=53 time=523.027 ms
64 bytes from 14.215.177.38: seq=3 ttl=53 time=757.225 ms
64 bytes from 14.215.177.38: seq=4 ttl=53 time=65.250 ms
64 bytes from 14.215.177.38: seq=5 ttl=53 time=30.266 ms
64 bytes from 14.215.177.38: seq=6 ttl=53 time=186.635 ms
64 bytes from 14.215.177.38: seq=7 ttl=53 time=103.790 ms
64 bytes from 14.215.177.38: seq=8 ttl=53 time=342.833 ms
64 bytes from 14.215.177.38: seq=9 ttl=53 time=130.757 ms
64 bytes from 14.215.177.38: seq=10 ttl=53 time=77.834 ms
64 bytes from 14.215.177.38: seq=11 ttl=53 time=43.304 ms
```

3.2.2 AP 模式网络测试

AP 模式即无线接入点，是一个无线网络的创建者，是网络的中心节点。

（1）应用程序及配置文件

hostapd // 将 wifi 网卡切换为 master 模式，模拟 AP 功能作为 AP 的认证服务器，负责控制管理 stations 的接入和认证

hostapd.conf // hostapd 的配置文件，生成的 wifi 热点的名称、密码、加密方式等

```
interface=wlan0
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
ssid=TEST_Wifi
hw_mode=g
```

```
channel=1
beacon_int=100
driver=nl80211
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
own_ip_addr=127.0.0.1
wpa=2
wpa_passphrase=12345678
rsn_pairwise=TKIP CCMP
```

udhcpd　　// dhcp 的 server，给 dhcp 的 client 分配 IP、网关等
udhcpd.conf　// udhcpd 的配置文件，负责 IP、网关、DNS 等的分配

```
start           192.168.1.20    #default: 192.168.0.20
end             192.168.1.254   #default: 192.168.0.254
interface       wlan0
```

（2）使用到的动态库

libcrypto.so.1.0.0　//加密相关
libnl.so.1　　// hostpad 需要用到的库
libssl.so.1.0.0　　//openssl 需要使用到的库

（3）测试流程

① 加载 8189fs.ko，加载成功之后会生成 wlan0 网卡
② 启动网卡并配置 IP 地址，ifconfig wlan0 192.168.1.10 up

```
[root@Ingenic-uc1_1:wifi_tools]# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 44:01:BB:4C:C0:C2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

③ 将上面的 3 个动态库文件放到/system/lib 目录下
④ ./hostapd -B hostapd.conf　// 创建热点
⑤ udhcpd udhcpd.conf　　// 配置 dhcp 的 server
⑥ 使用手机连接前面生成的热点 TEST_wifi，然后 ping 192.168.1.20


```
[root@Ingenic-uc1_1:wifi_tools]# ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20): 56 data bytes
64 bytes from 192.168.1.20: seq=0 ttl=64 time=108.097 ms
64 bytes from 192.168.1.20: seq=1 ttl=64 time=23.143 ms
64 bytes from 192.168.1.20: seq=2 ttl=64 time=48.437 ms
64 bytes from 192.168.1.20: seq=3 ttl=64 time=72.478 ms
64 bytes from 192.168.1.20: seq=4 ttl=64 time=111.937 ms
```

3.3 USB WiFi 驱动移植

USB WIFI 和 SDIO WIFI 只是底层的接口不同，在测试 STA 模式和 AP 模式时，使用的工具和方法都是相同的，因此这一部分只说明如何移植驱动，不再讲解具体的测试过程。

下面以 rtl8188ftv 为例讲解一下驱动的移植过程：

（1）获取 rtl8188ftv 的驱动源码，rtl8188FU_linux_v5.2.11.1_22663.20170607

（2）修改 Makefile 添加君正平台

```
93 ###################### Platform Related ######################
94 CONFIG_PLATFORM_INGENIC = y
95 CONFIG_PLATFORM_NVT9851X = n
```

（3）Makefile 添加一段关于君正支持的宏

```
1062 ifeq ($(CONFIG_PLATFORM_INGENIC), y)
1063 EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
1064 EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
1065 ARCH ?= mips
1066 CROSS_COMPILE ?= mips-linux-gnu-
1067 KSRC ?= /home/heng/work/opensource/kernel-4.4.94
1068 endif
```

移植之后编译代码就会生成 8188fu.ko 驱动文件。

3.4 USB wifi 功能测试

具体的网络功能测试和 SDIO WiFi 相同，这里不再赘述。

SDK 中已经包含了使用 720 编译器编译生成的 STA 模式和 AP 模式的相关程序，以及 lib 库。在 prebuilt/tools_t41/bin/uclibc/wifi 和 prebuilt/tools_t41/bin/glibc/wifi 目录下。

```
ddyang@ingenic:~/work/prebuilt/tools_t41/bin/glibc/wifi$ ls
hostapd  iwconfig  iwgetid  iwlist  iwpriv  lib  wpa_cli  wpa_supplicant
ddyang@ingenic:~/work/prebuilt/tools_t41/bin/glibc/wifi$ ls lib/
libcrypto.so.1.0.0  libiw.so.29  libnl.so.1  libssl.so.1.0.0
```

```
ddyang@ingenic:~/work/prebuilt/tools_t41/bin/uclibc/wifi$ ls
hostapd  iwconfig  iwgetid  iwlist  iwpriv  lib  wpa_cli  wpa_supplicant
ddyang@ingenic:~/work/prebuilt/tools_t41/bin/uclibc/wifi$ ls lib/
libcrypto.so.1.0.0  libiw.so.29  libnl.so.1  libssl.so.1.0.0
```

4 WiFi 常见问题分析

4.1 SDIO 对应的 GPIO Function 配置异常

引发该问题的原因大概有两点：

A：内核本身配置错误，这个要参考硬件电路设计看具体使用的是哪一组 GPIO

B：其他的驱动修改了 SDIO 对应的 GPIO Function 功能，比如我们一个客户在运行 IPC 程序时，SDIO WIFI 网络通信就异常了，最后排查原因是 sensor 的驱动修改了 SDIO GPIO 对应的 function 配置。

下面介绍一下如何排查这种错误：

这里以 T31 的 PA 口为例，PA 口的 PA08、PA09、PA10、PA11、PA16、PA17 六个引脚配置为 FUNCTION3 即为 msc1 功能。

Table 21-2 GPIO Port A summary

PAD_ID	POWER DOMAIN	PULL_RST	SMT_RST	DS_RST	FUNCTION0	FUNCTION1	FUNCTION2	FUNCTION3
PA08	0	HIZ	0	2pf	sd2(io-0)	dvp_d8_i(i-0)	uart2_cts_i(i-1)	msc1_d0(io-1)
PA09	0	HIZ	0	2pf	sd3(io-0)	dvp_d9_i(i-0)	uart2_rts_o(o)	msc1_d1(io-1)
PA10	0	HIZ	0	2pf	sd4(io-0)	dvp_d10_i(i-0)	uart2_txd_o(o)	msc1_d2(io-1)
PA11	0	PU	0	2pf	sd5(io-0)	dvp_d11_i(i-0)	uart2_rxd_i(i-1)	msc1_d3(io-1)
PA12	0	PU	0	2pf	sd6(io-0)	smb0_sda(io-1)	(i-0)	(i-0)
PA13	0	PU	0	2pf	sd7(io-0)	smb0_sck(io-1)	(i-0)	(i-0)
PA14	0	HIZ	0	2pf	sa0_o(o)	dvp_pclk_i(i-0)	pwm0_o(o)	(i-0)
PA15	0	HIZ	0	2pf	sa1_o(o)	dvp_mclk_o(o)	(i-0)	(i-0)
PA16	0	PU	0	2pf	cs2_o(o)	dvp_hsync_i(i-0)	smb1_sda(io-1)	msc1_clk_o(o)
PA17	0	PU	0	2pf	rd_o(o)	dvp_vsync_i(i-0)	smb1_sck(io-1)	msc1_cmd(io-0)

T31 的各组 GPIO 的基地址如下：

Name	Base	Description
PA_BASE	0x10010000	Address base of GPIO Port A
PB_BASE	0x10011000	Address base of GPIO Port B
PC_BASE	0x10012000	Address base of GPIO Port C

我们只需要查询四个寄存器就可以确定 GPIO 的功能了，分别是：

PORTx Interrupt Registers	Offset 0x10
PORTx Mask Registers	Offset 0x20
PORTx PAT1/Direction Registers	Offset 0x30
PORTx PAT0/Data Registers	Offset 0x40

我们使用下面的命令来查询 PA8、PA9、PA10、PA11、PA16、PA17 的功能。

```
echo $((((`devmem 0x10010010`) >> 8) & 0x30f));\
echo $((((`devmem 0x10010020`) >> 8) & 0x30f));\
echo $((((`devmem 0x10010030`) >> 8) & 0x30f));\
echo $((((`devmem 0x10010040`) >> 8) & 0x30f))
```

首先对读取到的寄存器值右移 8 位，因为是从 PA08 开始的；然后再跟 0x30f 进行&操作。0x30f 对应 PA08、PA09、PA10、PA11、PA16、PA17 这 6 位，如果是其它的 GPIO 引脚，修改这里即可。

```
[root@Ingenic-uc1_1:~]# echo $((((`devmem 0x10010010`) >> 8) & 0x30f));\
> echo $((((`devmem 0x10010020`) >> 8) & 0x30f));\
> echo $((((`devmem 0x10010030`) >> 8) & 0x30f));\
> echo $((((`devmem 0x10010040`) >> 8) & 0x30f))
0
0
783
783
```

783即为11 0000 1111

可以看到这六个引脚对应的 4 个寄存器的值为 0 0 1 1，查询下表可以看出当前这六个 GPIO 引脚配置为 FUNCTION3 模式，即 msc1 功能。如果查询出来的结果不正确，就要检查是哪里出现了问题。

INT	MASK	PAT1	PAT0	Port Description
1	0	0	0	Port is low level triggered interrupt input.
1	0	0	1	Port is high level triggered interrupt input.
1	0	1	0	Port is fall edge triggered interrupt input.
1	0	1	1	Port is rise edge triggered interrupt input.
1	1	0	0	Port is low level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	0	1	Port is high level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	0	Port is fall edge triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	1	Port is rise edge triggered interrupt input. Interrupt is masked. Flag is recorded.
0	0	0	0	Port is pin of device 0.
0	0	0	1	Port is pin of device 1.
0	0	1	0	Port is pin of device 2.
0	0	1	1	Port is pin of device 3.
0	1	0	0	Port is GPIO output 0.
0	1	0	1	Port is GPIO output 1.
0	1	1	?	Port is GPIO input.