乐鑫信息科技
Espressif Systems

ESP8089 Driver Release Description

ESP8089 driver is used in ESP8089/ESP8289 SoC, SDIO/SPI interface and Android4.1/4.2/4.3/4.4 platform and supports at least Linux kernel 2.6.27-3.10.40.

I、 How to Compile

Compiling can be done independently and can also be built into the kernel.

Compiling Option:

Driver compiling option is in Makefile and esp_config.mk.

Compiling option in Makefile is shared in all platforms. Please do not make changes. Compiling option in esp_config.mk is dependent on the platform and generally needs not to be modified. Some of these macros are described below.

P2P_CONCURRENT: This macro is used to support P2P in the driver. If the kernel version is too low(<2.6.37), the option can be closed in esp_config.mk.

ESP_ACK_INTERRUPT: This macro is used for ack sdio/spi interruption in the driver.The option can be closed in esp_config.mk if there is already ACK action.But please do not make changes.

ESP_USE_SDIO and ESP_USE_SPI: This macro is used to indicate the SDIO/SPI interface. Only one of the two macros must be opened in esp_config.mk.

USE_EXT_GPIO: The option is used to support external GPIO.Open the option in esp_config.mk if the platform requires external GPIO.

EXT_GPIO_OPS: The option is used to register OPS in external GPIO to the kernel in order for other drivers that needs to use external GPIO function. There is no need to call ESP8089 driver function. For more details, refer to external GPIO instructions.

MAC80211_NO_CHANGE: ESP8089 changed the mac802111 function in the old porting document. But this change would be deleted in the new porting document. If there is no change, add the macro in esp_config.mk.

MMC_NO_CHANG: ESP8089 changed the kernel mmc function in the old porting document. But this change would be deleted in the new porting document. If there is no change, add the macro in esp_config.mk.

SELF_MAC:Open the macro, ignore the first three bytes in the MAC address (which normally shows vendor ID), and adopt the user-defined three bytes in the MAC address (the three variables inside the macro SELF_MAC needs to be modified).

DHAS_FW: Open the macro and compile the firmware into the driver. Do not put firmware into file system. The macro in Makefile is recommended to be opened and do not make changes.

ESP_ANDROID_LOGGER: Open the macro and put driver log to logcat.Close the macro, and the driver log appears in UART that can be modified in Makefile. Note:UART printing will influence network performance. Do not transport log into UART, if there is no need to do so.

INIT_DATA_CONF: Open the macro, and driver will import the corresponding directory and init_data.conf. Otherwise, the file will be compiled as head file into the driver. Any change is not recommended as the macro is in Makefile.

HAS_INIT_DATA: Open the macro and esp_init_data.bin will be compiled into the driver as head file（esp_init_data.h）.

ESP_PREALLOC: Open the marco. The allocation of large memory in the driver will use the “pre-alloc” program the to alloc memory rather than call the memory allocation function of kernel such as kzalloc/kmalloc in order to avoid exceptions of due to the low memory of the machine. Machines with large enough memory is recommended not to use the macro. The macro needs to be matched with esp_prealloc program. Refer to the following content for more information about the application method.

-DDEBUGFS_BOOTMODE: Based on debugfs, open CONFIG_DEBUG_FS via linux kernel to support version 4.0 and FCC test APK later. The macro is closed by default when it is released. It is replaced by the following macro -DESP_CLASS for the same function.

-DESP_CLASS: Based on CLASS_ATTR, the macro is used to support version 4.0 and FCC test APK later on. The macro is open by default when it is released. It is used to replace-DDEBUGFS_BOOTMODE and therefore recommended to open. Although the two macros have the same function,the reason why there are two macros is that some clients have special requirements. The two macros can be closed. However, unless there are exceptional cases, one macro, or-DESP_CLASS is recommended to be opened.

(1) compile independently, the driver is put in any directory. The driver directory file xxx.env can be changed.
specify variable TOOLCHAIN_PREFIX，KERNEL_DIR，EAGLE_BASE_DIR
$source xxx.env
$make
If the compiled driver name needs to be changed, change the value of DRIVER_NAME in Makefile.

(2) compiling with kernel whole compiling into module (*.ko)
Driver directory is put into drivers/net/wireless/ (Kconfig file is included in driver directory). Change the kernel option of ESP8089 driver into m.
Change Makefile: obj-m := $(DRIVER_NAME).o into
obj-$(CONFIG_ESP8089) := $(DRIVER_NAME).o
Compile the whole kernel.
If the compiled driver name needs to be changed, change the value of DRIVER_NAME in Makefile.

(3) compiling with kernel (built-in)
Similar to kernel compiling into module, but change the kernel option of ESP8089 driver into y.
In addition, close INIT_DATA_CONF.

(4) **esp_prealloc** program compiling and usage
Two compiling methods: **ko and built-in. B**uilt-in is recommended.

Ko: similar to esp8089.ko. But remember to copy Module .symvers generated by compiling to the directory of esp8089. The file is needed in compiling ESP8089 driver. Users need to modify kernel code so that esp_prealloc.ko wiil be called automatically after booting.

built-in: compile with linux kernel so that esp_prealloc will automatically boot when kernel is booting（built-in is highly recommended）.

II、 Driver Modification

For SDIO platform , the file sdio_stub.c might need to be modified as it is mainly used to modify GPIO control option.

For SPI platform, the file spi_stub.c might need to be modified as it is mainly used to modify GPIO control option.

In addition, if the SDIO driver capability platform has been called, please keep up the driver capability after modification.

III、 About Files

Normally, there are three files that need to be put into file system:

Put **esp8089.ko, esp_init_data.bin and init_data.conf** into there own driver directory respectively (such as /system/lib/modules or /system/vendor/modules).

For build-in version, there might not exist file esp_init_data.bin and init_data.conf (because file system might not have been loading when executing driver). Compile in the form of esp_init_data.h and esp_conf.h respectively into the driver.

Note:

If the platform works, init_data.conf cannot be modified, except when parameters need to be added. If the platform is matched with RF performance, please make sure to save esp_init_data.bin and esp_init_data.h provided by the designer. Please make sure to save the files even when updating the driver.

IV、 init_data.conf File description

The conf file is used to adapt to various hardware designs, such as crystals with different frequency, SDIO protocol with different versions, bt-wifi co-exist, dual antenna, external PA and reset pins etc. In the original design when the chip works with conf file and updates the driver, vendors add new conf option to the initial conf file to optimize its performance. An example is given below:

```
crystal_26M_en=0;test_xtal=0;sdio_configure=2;bt_configure=0;bt_protocol=0;dual_ant_co
nfigure=0;test_uart_configure=2;share_xtal=0;gpio_wake=0;no_auto_sleep=0;ext_rst=0;wak
eup_gpio=12;ate_test=0;speed_suspend=1;$
```

A detailed parameter description is given below; the default value shows the parameter value when configuration option is not changed:

1. crystal_26M_en:
   Shows the crystal frequency supported by the SoC. Currently, it supports the following three kinds of crystals:
   0: 40MHz crystal
   1: 26MHz crystal
   2: 24MHz crystal
   Default value: 0
   DESC: After opening wifi search AP, check if the crystal is matched with the value.

2. test_xtal:
   It shows wifi chip output crystal clock to other chips and is used to check the clock pin. It has the following three effective values:
   0: no pin output for clock test
   1: GPIO can test if there is clock output to other chips
   2: U0RXDcan test if there is clock output to other chips
   Default value: 0
   DESC: The value is generally set at 0. It is set at 1 or 2 only when checking clock output.

3. sdio_configure
   Shows the SDIO protocol version of mmc_host. It has the following three effective values:
   0: automatic mapping by pin configuration

1: SDIO V1.1, data sampling when the clock is in positive edge
2: SDIO V2.0, data sampling when the clock is in negative edge

Default value: 0
DESC: If there are SDIO/SPI reading and writing errors, consider modifying the value. For SPI platforms, 1 is recommended.

4. bt_configure
It shows wifi and bluetooth co-exist mode with effective value 0-5:
0: no BT chip
1: GPIO0 -> WLAN_ACTIVE/ANT_SEL_WIFI
MTMS -> BT_ACTIVE
MTCK -> BT_PRIORITY
U0RXD -> ANT_SEL_BT
2: There is BT chip but it is not connected to wifi chip
3: GPIO0 -> WLAN_ACTIVE/ANT_SEL_WIFI
MTMS -> BT_PRIORITY
MTCK -> BT_ACTIVE
U0RXD -> ANT_SEL_BT
4: GPIO0 -> ANT_SEL_BT
MTMS -> BT_ACTIVE
MTCK -> BT_PRIORITY
U0RXD -> WLAN_ACTIVE/ANT_SEL_WIFI
5: GPIO0 -> ANT_SEL_BT
MTMS -> BT_PRIORITY
MTCK -> BT_ACTIVE
U0RXD -> WLAN_ACTIVE/ANT_SEL_WIFI
Default value: 0

5. bt_protocol
It shows the protocol for WiFi-BT co-exist. It has the following 0-5 effective values:
0: no co-exist protocol, antenna is used for WiFi
1: no co-exist protocol, antenna is used for BT
2: 2-line protocol, only BT_ACTIVE signal，WiFi and BTuse its respective antenna
3: 3-line protocol, WiFi and BTuse its respective antenna
4: 2-line protocol, only BT_ACTIVE signal，WiFi and BTuse the same antenna
5: 2-line protocol, WiFi and BT use the same antenna
Default value: 0

6. dual_ant_configure
It shows antenna configuration mode. It has the following 3 effective values:
0: normal mode
1: GPIO0+U0RXD switch two antenna
2: GPIO0+U0RXD switch external PA and LNA. GPIO0 at high level and U0RXD at low level shows TX
3: GPIO0+U0RXD switch external PA and LNA. GPIO0 at low level and U0RXDat high level shows TX
Default value: 0

7. test_uart_configure
It shows the pin of UART output.
0: no output
1: GPIO2 output

2: U0TXD output

Default value: 0

8. share_xtal

It shows BT shared crystal clock mode. It has the following 4 effective values:

0: no clock output

1: always output clock(even when the chip sleeps)

2: the value of XPD_DCDC correspond to clock output (even when the chip sleeps)

3: the value of GPIO2 correspond to clock output (even when the chip sleeps)

Default value: 0

9. gpio_wake

It shows that pins can be waken up when the wifi chip sleeps . It has the following 4 effective values:

0: cannot be waken up

1: XPD_DCDC wake up

2: GPIO0 wake up

3: both XPD_DCDC and GPIO0 can wake up

Default value: 0

Note: low level is effective.

10. no_auto_sleep

It shows whether the wifi chip sleeps when connected to AP. It has the following 2 effective values:

0: automatic sleep

1: no sleep but can be used in devices with low power requirement, such as TV box, etc. Increase of electric current will improve the performance.

Default value: 0

11. ext_rst

Reset the chip through external GPIO. It has the following two effective values:

0: not effective

1: Reset the chip through external GPIO(connect to the RESET pin)

Default value: 0

12. wakeup_gpio

Specific GPIO is used to wake up and reset chips. It has the following 0-15 effective values:

0: GPIO0

1: U0TXD

2: GPIO2

3: U0RXD

4: GPIO4

5: GPIO5

6: SD_CLK

7: SD_DATA0

8: SD_DATA1

9: SD_DATA2

10: SD_DATA3

11: SD_CMD

12: MTDI

13: MTCK

14: MTMS
15: MTDO
Default value: 12
Note: ext_rst=0 and low level is effective.

13. ate_test
Shows test mode. Currently, there are two values opened:
0: normal mode
1: used for RF performance test and replacement of the initial eagle_ate.ko
Default value: 0

14. speed_suspend
Used to speed up chip sleep. It has two effective values:
0: speed up chip sleep
1: suspend chip sleep
Default value: 0
Note: For SDIO interface platforms, 1 is recommended whereas 0 must be used for SPI interfaces.

V、 About MAC Address

ESP8089 contains OTP where a MAC address is written. Driver program registers the MAC address to MAC80211 reported by the chip and thus assigns two MAC addresses, one for STA interface (The MAC address is the MAC address in the chip) and another for P2P interface (The MAC address is calculated by the algorithm of MAC address and Supplicant). It is generally not necessary for vendors to know the MAC address.

（1） As the address segment 18:fe:34 is about to be running out, ESP8089 will adopt new address segment （ac:d0:74）. Please make sure to update driver to V1.9.1 （01302015） or newer when vendors use the latest ESP8089.

（2） ESP8089 will support customization of MAC address. Customized MAC address will be defined. (driver v1.9 or newerversion)

The first three bytes (which shows vendor ID) will be determined by a special ID:
ID=0:
The first three bytes are 0x18, 0xfe and 0x34;
ID=1:
The first three bytes are0xac, 0xd0, 0x74;
ID=255 (hexadecimal number 0xff):
It is a customized chip. The first three bytes are defined by clients.

The code below shows the corresponding relationship:

In esp_sip.c of ESP8089 driver, there are following lines of codes:

```
struct esp_mac_prefix esp_mac_prefix_table[] = {
    {0, {0x18, 0xfe, 0x34}},
    {1, {0xac, 0xd0, 0x74}},
   {255, {0x18, 0xfe, 0x34}},      //This line shows the customized MAC address
};
```

0\1\255 in the first column is the ID stated above. If vendors requires customized MAC address, please adopt customized MAC address(for ID>255, must customized by us; for average,) and change the codes in **blue** （0x18, 0xfe, 0x34） into the first three bytes of self-defined MAC

address. Please do not make any changes to the other two lines.

For clients who has special requirement on customized MAC address, please contact us for more information about customized MAC address.

（3） ESP8089 driver offers following solutions for vendors with special requirements (Vendors who have some products that are not customized want to change the first three bytes of the MAC address): (driver v1.9 or newer version)

In esp_sip.c of ESP8089 driver, there are following lines of codes:

```
#ifdef SELF_MAC
        epub->mac_addr[0] = 0xff;
        epub->mac_addr[1] = 0xff;
        epub->mac_addr[2] = 0xff;
#endif
```

The first three bytes in blue are generally vendor ID. Normally, the macro of is closed. Please open the macro in Makefile, i.e., and change the blue 0xff in above lines.

Note: Do not use this method . For vendor who do not customize the MAC address, the MAC address in the chip is recommended. Please do not make any changes to the above codes in order to avoid MAC address invalidity and conflicts.

Changing MAC address can be summarized into following three point:

(1) not change MAC address:

Command the mass production tool to achieve MAC address. For example, in Android platform, netcfg command will list all the interfaces with MAC address inside.

(2) only change the first three bytes of MAC address(vendor ID):

If the MAC address needs to be adopted by the vendors, change the first three bytes to change the MAC address registered into the system. It is very simple. Just change the first three bytes of epub->mac_addr in the above code. However, it can not ensure the continuity of MAC address. You can also get the MAC address via method (1).

(3) completely change MAC address

If vendors require to change the complete MAC address, there should be matching with external storage devices, such as flash. For example, vendors can write the required MAC address into flash during mass production. When ESP8089 is loading the driver, append new codes to the SELF_MAC in order to read MAC address in the flash. Assign values to epub->mac_addr[0~5] and open -DSELF_MAC in Makefile. In this way, support from is a must. But a full control on MAC address is realized.

Note: Please adopt the practically-applied MAC address. Otherwise, illegal MAC address issues (broadcast/multi-cast MAC address) might arise. In that case , the chip will not work.

VI、 About the configuration of strengthening SoftAP performance(Please make sure to read carefully, if you have special requirement on SoftAp)

（1） ANDROID platform

Please find the directory of android/system/netd/ and open SoftapController.cpp, find the following codes:

```
asprintf(&wbuf, "interface=%s\ndriver=nl80211\nctrl_interface="
"/data/misc/wifi/hostapd\nssid=
%s\nchannel=6\nhw_mode=g\nieee80211n=1\n",
iface, ssid);
```

append ”dtim_period=1\n” to the printing tail, i.e.

```
asprintf(&wbuf,"interface=%s\ndriver=nl80211\nctrl_interface="
"/data/misc/wifi/hostapd\nssid=
%s\nchannel=6\nhw_mode=g\nieee80211n=1\ndtim_period=1\n",
iface, ssid);
```

（2）simple linux platform（on the premise of using hostapd）

Find hostapd.conf，the file is generally in the directory of /etc

Open the file and append to the tail

dtim_period=1

V1.9.2-06292015

1、fix that some APs which have bugs themselves may cause something wrong on connection in encryption mode.

2、fix RSSI display.

3、improve the stability of throughput.

4、 add a function that esp_fcc_tool can modify the tx power when send modulation and non-modulation packet. (This function depend on esp_fcc_tool of version v1.3)

V1.9.1-01302015

1、fix The problem of Softap sudden disconnect in some case

2、 strengthen the performance of esp_prealloc and its compatibility with mono-nuclear CPU (which should be matched with version esp_prealloc V2.4)

3、improve the using method of FCC test APK. There is no need to consider whether the driver is in built-in version. (should be supported by APK4.0 or above version. please refer to the last two macro descriptions in I when compiling the driver)

4、Strengthen the check to STA (GC) in SOFTAP(GO), and avoid “abnormal” disconnect to STA (or GC).

5、 fix the support for new mac address. (V1.9 can support new MAC address. However, when the new MAC address is not included in the mapping list, V1.9 will continue to adopt the old MAC address)

6、strengthen the smooth algorithm of RSSI fluctuation.

7、fix some parameter and increase the compatibility.

8、update the version of esp_prealloc to V2.4.

V1.9-11272014

1、Strengthen the support to the new MAC address and MAC address customization version. (It is of great importance to refer to 5 for details about MAC address)

2、Strengthen the code robustness.

3、Strengthen the driver uniformity of simple linux and android.

4、 Strengthen the pre-allocation memory in SPI mode (should be matched with esp_prealloc V2.3 or above version).

5、Strengthen the compatibility with non-standard AP.

6、Strengthen the stability of P2P mode.

V1.8-09292014

1、Fix the BT-WiFi co-exist module (WiFi might not work if there is no common protocol.)

2、Strengthen the compatibility with kernel 3.10

3、 Strengthen the performance of Soft AP and P2P GO mode. （For high requirements on Soft AP performance, please refer to directory 6）

V1.7-09042014

1、Add support to LINUX KERNEL to 3.10.

2、Strengthen the optimization of low memory (no more than 513MB) devices(which should be matched with esp_prealloc)

3、Strengthen the compatibility with AP.

4、Strengthen the robustness of external GPIO underlying code.

V1.6-08152014

1、Strengthen fault tolerant mechanism and avoid KERNEL crash due to small probability events.

2、Strengthen the bogus interrupt of filter SDIO and avoid the disturbance of I/O by SDIO bogus

interrupt.
3、Further decrease power.

V1.5-07242014
1、 Solve the small probability problem that some clients cannot open WiFi.
2、 Add SDIO test code in order to support new version APP.
3、 Add test code. There is no need to compile new driver for RF performance text.
4、 Solve the problem of too large current for WiFi during system hibernation and accelerate hibernation.
5、 Strengthen code robustness of external GPIO interface function.
6、 Fix bugs that SOFT AP cannot work in the 11b mode.
7、 Optimize BT WiFi co-exist mode and support high power PA.
8、 Increase its operation capability when working in extreme conditions(low/high temperature)
9、 add init_data.conf file description to the document which is no longer provided independently.

V1.4-06172016 (important updates)
1、Solve the problem of fluctuating signals in wifi status bar.
2、 Solve the WiFi on/off test problem due to the match between ESP8089 and PMU. Please make sure to use the new version.
3、 Solve the problem that might lead to the overlap of P2P MAC address when some clients claim to use their own MAC address.
4、Solve the problem that CPU cannot be waken up after external GPIO interrupt.

V1.3-05212014
1、 Support Linux 2.6.27kernel.
2、 Fix system crash caused by WiFi on/off in Android 4.4.
3、 Increase the down/up rate in SoftAP and P2P modes.
4、 Support BK3515D BT WiFi co-exist protocol and change the config description file.
5、 Fix some platform that cannot connect to WiFi after waking up.

V1.2-04282014
1、 Fix some platforms where RF performance app is not available with driver code.
2、 Fix kernel error led by external GPIO on some platforms.

V1.1-04172014
1、 Unified driver that supports SDIO/SPI.
2、 The driver supports RF performance test app.
3、 Solve the problem of kernel crash when play multiple videos.