# Quick Start Guide for Driver Compilation and Installation

## Contents



## Introduction

In this document, we introduce two ways to compile and install our Wi-Fi driver: 1) Using install.sh script for PC-Linux and 2) Step by step manually. The former targets for end users who are not familiar with Linux system, while the later for engineers who want to port our Wi-Fi driver onto different platforms.

## 1. Using install.sh Script for PC-Linux

For driver compilation and installation in PC-Linux, we provide an install.sh script to do the duties automatically. If you want to use our Wi-Fi solutions to access network on PC-Linux, you can just run install.sh script and then control Wi-Fi with utilities such as Network Manager. For further information about Wi-Fi station mode, please refer to:

document/Quick_Start_Guide_for_Station_Mode.pdf.

If you want to apply our Wi-Fi solutions on other embedded platforms, you should read and check the following paragraphs.

## 2. Decompress the driver source tar ball

The driver source tar ball is located in the driver folder of our software package. For example, to decompress rtl8712_8188_8191_8192SU_usb_linux_v2.6.6.0.20120405.tar.gz:

```
root@driver/# tar zxvf rtl8712_8188_8191_8192SU_usb_linux_v2.6.6.0.20120405.tar.gz
```

## 3. Compilation Settings in Makefile

### 3.1. Adding or Selecting Target Platform

The default target platform is PC-Linux, if you do not want to compile driver for other platforms you can skip this section.

To add or select target platform for compilation, we provide two sections in Makefile: 1) platform selection section and 2) platform setting section. First, you should look at the platform selection section of Makefile:

```
CONFIG_PLATFORM_I386_PC          =    y
CONFIG_PLATFORM_ANDROID_X86      =    n
CONFIG_PLATFORM_ARM_S3C2K4       =    n
CONFIG_PLATFORM_ARM_PXA2XX       =    n
CONFIG_PLATFORM_ARM_S3C6K4       =    n
CONFIG_PLATFORM_MIPS_RMI         =    n
CONFIG_PLATFORM_RTD2880B         =    n
CONFIG_PLATFORM_MIPS_AR9132      =    n
CONFIG_PLATFORM_MT53XX           =    n
CONFIG_PLATFORM_RTK_DMP          =    n
```

The platform selection section consists of entries with ‘CONFIG_PLATFORM_’ prefix. Only one entry is allowed to be set with value ‘y’ and others with ‘n’. The ‘CONFIG_PLATFORM_I386_PC’ is selected by default.

We can select an existing entry or add a new entry for your target platform. For example, to add and select a new entry, ‘CONFIG_PLATFORM_NEW’:

```
CONFIG_PLATFORM_I386_PC          =    n
CONFIG_PLATFORM_NEW              =    y
```

Second, you should create and/or modify the corresponding entry inside platform setting section. For example, adding the following entry in platform setting section for ‘CONFIG_PLATFORM_NEW’ we just add:

```
ifeq ($(CONFIG_PLATFORM_NEW), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
ARCH := arm
CROSS_COMPILE := /opt/ new/toolchain/arm-eabi-4.4.3/bin/arm-eabi-
KSRC := /opt /new/kernel
endif
```

### 3.2. Platform Setting Section in Detail

- **EXTRA_CFLAGS**

The EXTRA_CFLAGS is usually used to carry some additional settings at compilation time through macro definitions.

| Macro | Effect |
|---|---|
| CONFIG_BIG_ENDIAN | Define some internal data structure as big endian. |
| CONFIG_LITTLE_ENDIAN | Define some internal data structure as little endian. |
| CONFIG_MINIMAL_MEMORY_USAGE | For better performance in powerful platform, we allocate large physical continuous memory as TX/RX IO buffers. In some embedded platform, there is chance to fail to allocate memory. Define this macro to prevent this situation. |
| CONFIG_PLATFORM_ANDROID | Older Android kernel do not has CONFIG_ANDROID defined. Define this macro to force the Android corresponding code inside our driver to be compiled. For newer Android kernel, it has no need to define this macro, otherwise, warning message about redefinition will show up |

- **ARCH**

The ARCH is used to specify the architecture of the target platform CPU, such as: arm, mips, i386, etc.

- **CROSS_COMPILE**

The CROSS_COMPILE is used to specify the toolchain prefix used for driver compilation.

- **KSRC**

The KSRC is used to specify the path of kernel source used for driver compilation

- **MODULE_NAME**

Our default module name for RTL8191SU-series is 8712u.

If you want to change the module name, you can set value of MODULE_NAME here. For example, setting module name as ‘wlan’:

```
ifeq ($(CONFIG_PLATFORM_NEW), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
ARCH := arm
CROSS_COMPILE := /opt/ new/toolchain/arm-eabi-4.4.3/bin/arm-eabi-
KSRC := /opt /new/kernel
MODULE_NAME := wlan
endif
```

### 3.3. Other Compilation Settings

We have some compilation settings that could be applied.

If you know what the macro means in the autoconf file, you could modify the configuration by yourself. Please consult us if have no idea what some macro means.

#### 3.3.1 Configure Wireless API

We provide two Linux wireless API: WEXT, CFG80211, and our default setting is CFG80211 with compile flag #define CONFIG_IOCTL_CFG80211 1. If you want to change to WEXT, please mark this flag.

## 4. Integrating Driver Source into Linux Kernel Tree

This paragraph is for integrating our driver source into Linux kernel tree and building system. If you have no need to do this, simply skip this paragraph.

We suggest the name with compile flag is CONFIG_RTL8712U and folder name is 8712u, then go through the following steps:

1). Copy the driver source folder into drivers/net/wireless/ and rename it as <folder_name>, 8712u.

2). Add the following line into drivers/net/wireless/Makefile, CONFIG_RTL8712U is for <compile_flag>, 8712u is for <folder_name>:

```
obj-$(CONFIG_RTL8712U)    += 8712u/
```

3). Add the following line into drivers/net/wireless/Kconfig, 8712u is for <folder_name>:

```
source "drivers/net/wireless/8712u/Kconfig"
```

4). Config kernel, for example, with ‘make menuconfig’ command to select ‘y’ or ‘m’ for our driver.

5). Now, you can build kernel with ‘make’ command.

## 5. Compiling Driver

### 5.1. Compiling Driver in Driver Source Folder

For compiling driver in the original driver source folder, simply cd into the driver source folder and start build driver with ‘make’ command.

```
root@rtl8712_8188_8191_8192SU_usb_linux_v2.6.6.0.20120405# ./make
```

If everything goes well, it will produce a *MODULE_NAME*.ko file. The MODULE_NAME is specified in Makefile. Please refer to:

“MODULE_NAME” in “3.2. Platform Setting Section in Detail”.

### 5.2. Compiling Driver under Kernel Tree

For compiling driver under kernel tree, please refer to:

“4. Integrating Driver Source into Linux Kernel Tree”.

## 6. Driver Installation

If you have compiled Wi-Fi driver as kernel module and produced a .ko file such as 8712u.ko, you should insert driver module with ‘insmod’ command:

```
root@ rtl8712_8188_8191_8192SU_usb_linux_v2.6.6.0.20120405# insmod 8712u.ko
```

As for driver compiled in kernel, it has no need to do ‘insmod’ command.