SMC-90M 802.11 b/g/n Wifi Module

SMC-90M is an 802.11b/g/n Wireless USB interface LGA module that enables devices with high performance wireless connectivity.

FEATURES

- PHY data rate up to 144.4 Mbps using 20MHz bandwidth,

SPECIFICATIONS

Standards	IEEE 802.11 b/g/n
Chipset	AP6212
Frequency Band	2.4 - 2.483 GHz
Encryption	64/128 bit WEP, WPA,WPA2, IEEE 802.1x
Mode	**2.4GHz Band:** 802.11n (HT20, MCS7) 802.11g (54Mbps) 802.11b
Receive Sensitivity	**2.4GHz:** 802.11b: -80 dBm max. @ 11Mbps 802.11g: -65 dBm max. @ 54Mbps 802.11n (HT20): -64 dBm max. @ 144.4Mbps
DC Voltage	3.3V (Typical)
Host Interface	USB 2.0
Antenna	Onboard antenna (1T1R)

Host Connector	Wifi Module
Temperature	Operating: 0 ~ 70 Celsius Storage: -20 ~ 70 Celsius
Humidity	Storage: 10 ~ 80% (Non Condensing)
Dimensions (L x W x H)	38.5 x 23.0 x 2.85 mm

***Specifications are subject to change without further notice.**

FEDERAL COMMUNICATIONS COMMISSION INTERFERENCE STATEMENT

This equipment has been tested and found to comply with the limit s for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment of f and on, the user is encouraged to try to correct the interference by one or more of the following measures:

-- Reorient or relocate the receiving antenna.
-- Increase the separation between the equipment and receiver.
-- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
-- Consult the dealer or an experienced radio/TV technician for help.

CAUTION:
Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.
This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Safety warning notice:

RF exposure warning: End user to keep at least 20 cm distance from the antennas of the device.

OEM INTEGRATION INSTRUCTIONS

This device is intended only for OEM integrators under the following conditions:
The antenna must be installed such that 20 cm is maintained between the antenna and users, and the transmitter module may not be co-located with any other transmitter or antenna. The module shall not be used with any other antenna than the certified integral on-board PCB antenna.

As long those conditions above are met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed (for example, digital device emissions, PC peripheral requirements, etc.).

Validity of using the module certification:
In the event that these conditions cannot be met (for example certain computer configurations or co-location with another transmitter), then the FCC authorization for this module in combination with the host equipment is no longer considered valid and the FCC ID of the module cannot be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate FCC authorization.

End product labeling:
This transmitter module is authorized only for use in device where the antenna may be installed such that 20 cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following: “Contains FCC ID: 2ABDZSMC90”.

Information that must be placed in the end user manual:
The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user's manual of the end product which integrates this module. The end user manual shall include all required regulatory information/warning as show in this manual.

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 rtl8188C_8192C_8192D_usb_linux_v3.3.0_2920.20111123.tar.gz:

```
root@driver/# tar zxvf rtl8188C_8192C_8192D_usb_linux_v3.3.0_2920.20111123.tar.gz
```

3. Selecting Chip Type with make_drv Script (for compound release)

Our driver source release has two types: 1) single release, which can build out driver only for single chip type, and 2) compound release, which can build out drivers for multiple chip types separately.

For compound release driver, you will see make_drv script after you decompress the driver tar ball located in driver folder. Before compiling driver source, executing the make_drv to select the target chip type to compile. For example:

```
root@rtl8188C_8192C_8192D_usb_linux_v3.3.0_2920.20111123# ./make_drv
Please select chip type(1/2):
1) RTL8192cu
2) RTL8192du
#? 1
You have selected RTL8192cu
```

4. Compilation Settings in Makefile

4.1. 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
```

4.2. Platform Setting Section in Detail

l 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**

Different module name is assigned to drivers for different chips:

Chip type	Default module name
RTL8192CU-series	8192cu
RTL8192CE-series	8192ce
RTL8192DU-series	8192du
RTL8192DE-series	8192de
RTL8723AS-series	8723as
RTL8723AU-series	8723au
RTL8189ES-series	8189es
RTL8188EU-series	8188eu

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
```

4.3. Other Compilation Settings

We still have some compilation settings could be applied. For settings and further information about power saving mode, please refer to:

document/HowTo_enable_the_power_saving_functionality.pdf.

If you know what the macro means in the autoconf file, you could modify the

configuration by yourself. See the following table for the autoconf file you should modify for a specific chip type:

Chip type	Autoconf file to modify
RTL8192CU-series	autoconf_rtl8192c_usb_linux.h
RTL8192CE-series	autoconf_rtl8192c_pci_linux.h
RTL8192DU-series	autoconf_rtl8192d_usb_linux.h
RTL8192DE-series	autoconf_rtl8192d_pci_linux.h
RTL8723AS-series	autoconf_rtl8723a_sdio_linux.h
RTL8723AU-series	autoconf_rtl8723a_usb_linux.h
RTL8189ES-series	autoconf_rtl8189e_sdio_linux.h
RTL8188EU-series	autoconf_rtl8188e_usb_linux.h

5. 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.

For compound release driver source, make_drv should be execute to select chip type for the driver source. Please refer to:

"3. Selecting Chip Type with make_drv Script (for compound release)".

For different chip types, we have different suggestions for <compile_flag> and <folder_name> to use for the integration process:

Chip type	<compile_flag>	<folder_name>
RTL8192CU-series	CONFIG_RTL8192CU	rtl8192cu
RTL8192CE-series	CONFIG_RTL8192CE	rtl8192du
RTL8192DU-series	CONFIG_RTL8192DU	rtl8192du
RTL8192DE-series	CONFIG_RTL8192DE	rtl8192de
RTL8723AS-series	CONFIG_RTL8723AS	rtl8723as
RTL8723AU-series	CONFIG_RTL8723AU	rtl8723au
RTL8189ES-series	CONFIG_RTL8189ES	rtl8189es
RTL8188EU-series	CONFIG_RTL8188EU	rtl8188eu

Assuming the driver source is for RTL8192CU-series, to integrate driver source into kernel building system, go through the following steps:

1). Copy the driver source folder into drivers/net/wireless/ and rename it as <folder_name>, rtl8192cu.

2). Add the following line into drivers/net/wireless/Makefile, CONFIG_RTL8192CU is for <compile_flag>, rtl8192cu is for <folder_name>:

```
obj-$(CONFIG_RTL8192CU)    += rtl8192cu/
```

3). Add the following line into drivers/net/wireless/Kconfig, rtl8192cu is for <folder_name>:

```
source "drivers/net/wireless/rtl8192cu/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.

6. Compiling Driver

6.1. 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@rtl8188C_8192C_8192D_usb_linux_v3.3.0_2920.20111123# ./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 "4.2. Platform Setting Section in Detail".

6.2. Compiling Driver under Kernel Tree

For compiling driver under kernel tree, please refer to:

"5. Integrating Driver Source into Linux Kernel Tree".

7. Driver Installation

If you have compiled Wi-Fi driver as kernel module and produced a .ko file such as 8192cu.ko, you should insert driver module with 'insmod' command:

```
root@rtl8188C_8192C_8192D_usb_linux_v3.3.0_2920.20111123# insmod 8192cu.ko
```

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