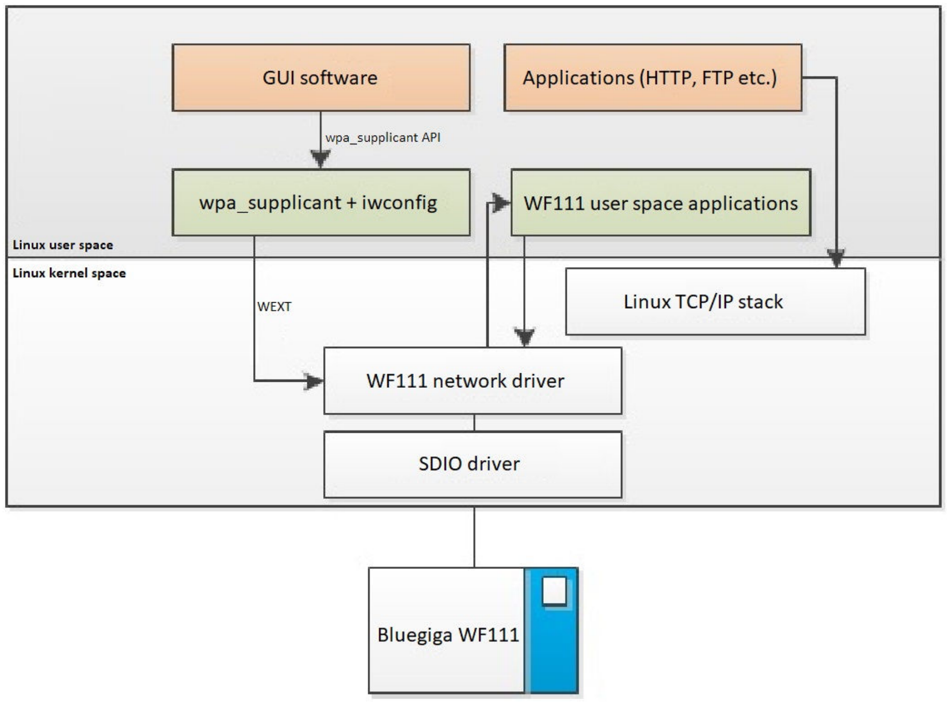
AN996: WF111 Linux Driver Installation

This document describes how to compile and install the Linux drivers for Bluegiga WF111 802.11 b/g/n module and how to verify the driver functionality.

The WF111 Linux driver architecture is briefly described below.

KEY POINTS

- Complete instructions on WF111 Linux driver compilation and installation
- Driver installation verification
- WF111 configuration
- WF111 usage examples for STA and AP modes
- Introduction to troubleshooting



WF111 Linux Driver Architecture

1. Prerequisities

To install the WF111 Linux driver on armv5 or armv7 machines, the driver first need to be cross compiled by a separate build machine which is a Linux machine. The armv5 or armv7 machines are the target machine where the WF111 Linux driver is running as well as the WF111 module is integrated. There are prerequisites for both build and target machine.

1.1 Build Machine Prerequisites

- Install cross compiler on build machine.
- Build Linux kernel source code for the kernel version which is running on the target machine. The Linux kernel source tree need to be prepared by:

```
make modules_prepare ARCH=arm CROSS_COMPILE=/usr/bin/arm-linux-
```

Info

The **CROSS_COMPILE** is the prefix for the ARM tool chain, so for example if the ARM gcc is **"/usr/bin/arm-linux-gcc"**,then the prefix would be **"/usr/bin/arm-linux-"**.

Note

In the configuration file (usually .config) of Linux kernel source code, the following options have to be enabled:

- `CONFIG_WIRELESS_EXT`
- `CONFIG_MODULES`
- `CONFIG_FW_LOADER`

Please note that these options are enabled for the kernel of your target machine.

- In case you get the message "WIRELESS_EXT is missing from the kernel config" during compilation, it means that Wireless-Extensions are not enabled. One way to enable them is by following the procedure below:

```
$ make ARCH=arm menuconfig

=> Networking support
=> Wireless
<> cfg80211 - wireless configuration API
<> Generic IEEE 802.11 Networking Stack (mac80211)

=> Device Drivers
=> Network device support
=> Wireless LAN
[M] IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
```

1.2 Target Machine Prerequisites

- WF111 Linux driver requires kernel version 2.6.24 up to 4.1. Minimum kernel version is 2.6.37 if at91_mci mmc driver is used because of a multiblock transfer bug in the driver code. Your target systems kernel version can be checked by:

```
$ uname -a
Linux apx4devkit 3.2.36 #1 Tue Mar 25 02:48:55 EET 2014 armv5tejl GNU/Linux
```

  For client mode (STA) with WPA/WPA2 support requires:
  - wireless-tools version 28 or newer. This can be checked using the command **iwconfig -v**
  - wpa_supplicant version 0.7.3. The version can be checked using the command **wpa_supplicant -v**
- Userspace must support firmware loading, in particular when using an embedded system with Busybox and its mdev, the option ENABLE_FEATURE_MDEV_LOAD_FIRMWARE must be set. On a non-busybox system this is normally handled by **udev**.
  - Busybox version 1.11 or newer is required if mdev is used as kernel hotplug event handler.

1.3 Needed Files

There is separate driver package for armv5 and armv7 machines. The driver package wf111-linux-driver-5.2.2-r4_ARCH.tar.gz contains following items:

Note
Word ARCH in the file name is replaced by the architecture of system (for example armv5) and number r4 reports release number.

- WF111 Linux kernel module source
  - Source code for **unifi_sdio.ko**
- WF111 dynamically and statically linked userspace applications for ARM.
  - **unififw** - The script that runs unifi_helper with the appropriate parameters when a WF111 module is detected. This gets automatically called by the kernel module to start the unifi_helper.
  - **unifi_helper** - User-space helper daemon for the WF111 driver, started by the unififw.
  - **unifi_config** - Configuration and status reporting utility, used for example to configure power saving modes.
- WF111 firmware and configuration files
  - **staonly.xbv** - Station firmware executed in the WF111 module.
  - **ap.xbv** - Access Point mode firmware executed in the WF111 module.
  - **ufmib.dat** - Symbolic link to the used firmware file that contains configuration parameters.
  - **mib111_drv.dat** - Firmware file without coex or activity LED support.
  - **mib111_drv_led.dat** - Firmware file without coex but activity LED support enabled. This is default.

2. Compiling, Installing, and Loading WF111 Driver

2.1 Compiling WF111 Driver on the Build Machine

- Based on the architecture of your target machine, download the corresponding version of WF111 linux driver from Bluegiga the website.
- Extract the driver files on your build machine.
- In the directory where you have extracted the driver files, issue the following commands to compile the WF111 driver:

```
tar xzvf wf111-linux-driver_5.2.2-r3_armv5t.tar.gz
cd wf111-linux-driver_5.2.2-r3_armv5t
make install_static ARCH=arm KDIR=path/to/arm_kernel/source CROSS_COMPILE=/usr/bin/arm-linux
```

Info
The **CROSS_COMPILE** is the prefix for the ARM tool chain, so for example if the ARM gcc is "**/usr/bin/arm-linux-gcc**", then the prefix would be "**/usr/bin/arm-linux-**".The **KDIR** is the path to where the Linux kernel source code are located on the build machine. You need to change the "**path/to/arm_kernel/source**" to match the location of ARM Linux kernel source code.

Info
Run "make help" to get more information about build parameters.

After this, an output folder will be generated in the build machine located at: wf111-linux-driver_5.2.2-r3_armv5t/output.

2.2 Overriding the MAC Address

If you want to override the MAC address of WF111 module, you need to write a file called lib/firmware/unifi-sdio/mac.txt, which will be located in the output directory after the previous commands. If there is no need to override the MAC address, the file mac.txt should be deleted if exists.

Sample mac.txt file format
00:11:22:33:44:55

2.3 Installing the Driver Files on Target Machine

Copy the full file system hierarchy of the **output** folder in the build machine and paste it to the target machine. The files on the target machine should be as follows:

- /lib/modules/KERNEL_VERSION/extra/unifi_sdio.ko
- /lib/firmware/unifi-sdio/staonly.xbv
- /lib/firmware/unifi-sdio/ap.xbv
- /lib/firmware/unifi-sdio/mib111_drv_led.dat
- /lib/firmware/unifi-sdio/ufmib.dat -> mib111_drv_led.dat
- /lib/firmware/unifi-sdio/mib111_drv.dat
- /usr/sbin/unififw
- /usr/sbin/unifi_helper
- /usr/sbin/unifi_config

2.4 Loading the Linux Driver on Target Machine

On the target machine, execute the command below to load the WF111 driver into the kernel.

```
depmod -a
modprobe unifi_sdio
```

Info

To increase debug level of the unifi driver use unifi_debug kernel module parameter
modprobe unifi_sdio unifi_debug=3
or change debug level on the fly
echo 3 >/sys/module/unifi_sdio/parameter/unifi_debug
where 3 is debug level from 0 to 9.

3. Verifying Driver Installation

On the target machine, execute the command below to verify that the driver has been correctly installed.

1. Run **lsmod** and verify that **unifi_sdio** is installed.
2. Plug the WF111 module in your target machine.
3. Run **dmesg** and verify that the output looks as in the following example. The **UniFi ready** message is the sign that the driver has been correctly installed and the module is successfully initialized. Alternatively the output can be seen from the Linux system log file (syslog)

```
...
unifi0: UniFi f/w protocol version 9.1 (driver 9.1)
unifi0: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_gcc 1089 bfsw@eagle@630492
unifi0: Firmware patch 1410
unifi0: unifi0 is wlan0
unifi0: UniFi ready
```

Note

Always make sure that the protocol version matches the driver version. The first line of the example above shows the correct situation where both versions are 9.1

You can also verify that the wireless interface is enabled by running **iwconfig**.

```
wlan0     IEEE 802.11-bgn  ESSID:off/any
          Access Point: Not-Associated   Bit Rate=0 kb/s
          RTS thr=0 B   Fragment thr:off
          Power Management period:500ms  mode:All packets received
          Link Quality:25/40  Signal level:-51 dBm  Noise level:-76 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Note that unless changed, the WF111 kernel module will call the device **wlanX** where X is the interface number.

At this point, the standard Linux wireless tools on the target machine can be used to manage the new wireless interface. For example you can test that scanning for access points works by using the **iwlist** command, as seen below, or you can try to connect to a open wireless network with the command **iwconfig**.

```
iwlist scan
```

In order to connect to WPA/WPA2 protected networks you would need to use the WPA_Supplicant (iwconfig is only working with open networks). This requires the wpa_supplicant.conf to be configured before launching the related executable. For more information about Linux wireless tools, please refer for example to:

- http://en.wikipedia.org/wiki/Wireless_tools_for_Linux
- http://linux.die.net/man/5/wpa_supplicant.conf

Note

For protection against Key Reinstallation Attacks (KRACK) in 4-way handshake it is required to use at least wpa_supplicant v2.6 with applied patches from https://w1.fi/security/2017-1/

Note

Unifi driver has it's own roaming algorithm so it is needed to disable roaming from wpa_supplicant by adding following lines to wpa_supplicant build time configuration file (.config).

```
CONFIG_NO_ROAMING=y
CFLAGS += -DCONFIG_NO_ROAMING
```

Info
There is an example wpa_supplicant configuration file delivered with driver package at csr/os_linux /wpa_supplicant/wpa_supplicant_sample.conf.

4. Configuration

WF111 configuration can be done with a command line application called unifi_config. It can be used to configure the WF111 or report it's status on the target machine.

unifi_config provides the following status information:

- Version information
- Power-save mode
- Co-existence mode
- ESSID

unifi_config can be used to control the following features:

- Turn the WF111 ON or OFF
- Control power save modes
- Packet filter configuration
- WMM power save configuration

unifi_config can be found under **/usr/sbin**.

Argument	Description
`--help`	Prints usage information
`--dev <filename>`	Device to open. **Default: /dev/unifiudi0**
`--show`	Displays status information
`--wifion`	Turns WF111 ON
`--wifioff`	Turns WF111 OFF

`--powersave none`
`--powersave fast`
`--powersave full`
`--powersave auto`
	Configures the power save mode and allows user to override the automatic power saving.
	
	none: turns off power saving and forces active mode
	fast: puts WF111 into a power save mode where WF111 enters 802.11 power save mode after 2 seconds of 802.11 inactivity
	full: forces WF111 into a full power save mode where WF111 is always in 802.11 power save mode
	auto: restores the 802.11 power save control to the Linux driver
	
	Default: none

`--batteries yes`
`--batteries no`
	This option informs the devices system power state and whether the device runs of batteries or not.

5. AP Mode

This section instructs how to turn the WF111 into Access Point (AP) mode.

This is done by first issuing the following driver specific **iwpriv** command meant to configure the AP parameters:

```
# iwpriv wlanN AP_SET_CFG ASCII_CMD=AP_CFG,SSID="<network name>",SEC="<open/wpa2-psk>",KEY=<PSK>,
CHANNEL=<channel number>,PREAMBLE=<0/1>,MAX_SCB=<N>,END
```

Where

- **wlanN** is the network interface
- **SSID** is the network name (1 to 32 characters)
- **SEC** is the security mode used (open or WPA2-Personal)
- **KEY** is the (AES-CCMP) PSK converted to a string (32 characters)
- **CHANNEL** is the 802.11b/g channel (1-14, but restricted by the regulatory domain)
- **PREAMBLE** is either short (1) or legacy preamble (0)
- **MAX_SCB** is the maximum number of stations allowed to be connected to AP at any time (1 to 8 stations)

Then the AP mode can be started with the following command:

```
# iwpriv wlanN AP_BSS_START
```

At any time the AP mode can be stopped by the command below:

```
# iwpriv wlanN AP_BSS_STOP
```

After starting the AP it is needed to assign an IP address for example by using the ifconfig utility.

In addition, it is possible to also configure the AP's parameters below:

- PHY support (b/g/bg)
- Beacon interval
- DTIM period
- Enable/disable WMM
- Maximum allowed listen interval
- WPA2 related parameters

This can be done by calling **unifi_config** with parameter **--setap_cfg** <config file>

Note
To generate the security key (KEY) use Linux command line tool : **wpa_passphrase <SSID> <passphrase>**. The tool will output the network security key in correct mode.

An example of configuration file for the unifi_config --setap_cfg is given below:

```
##### CSR AP configuration file ##############################################
# Empty lines and lines starting with # are ignored
#
# Operation mode (b = IEEE 802.11b, g = IEEE 802.11g,
# Default: IEEE 802.11b\+ IEEE 802.11g + IEEE 80211n
# bit0=>Reserved. Shall be set to 0.
# bit1=>b
# bit2=>g
# bit3=>n
# Default :0x0E=>bgn &nbsp;
phySupportedBitmap=14
#
# Beacon interval in kus (1.024 ms) (default: 100; range 15..65535)
beaconInterval=100
#
# DTIM (delivery trafic information message) period (range 1..255):
# number of beacons between DTIMs (1 = every beacon includes DTIM element)
# (Default: 3)
dtimPeriod=3
#
# WMM Enable/Disable : Default Enabled
#wmmEnabled=1
#
# Maximum allowed Listen Interval Default: 255
maxListenInterval=255
#
##### IEEE 802.11 AP MAC configuration #######
# Using 9 mcrosecond slot instead of 20 microsecond slot. Note that if Enabled, AP shall use short slot time
only if
# all the station in the BSS support short slot.
# 0 => use 20 microsecond 1=> use 9 microsecond if possible.
#shortSlotTimeEnabled=0
#
# ctsProtectionType indicates use of CTS protection in various conditions
# 0 : Always use CTS protection
# 1 : Never use CTS protection
# 2 : Use CTS protection when an overlapping legacy BSS is detected or if our BSS has non ERP stations.
# 3 : Use CTS only if our BSS has non ERP stations.
#ctsProtectionType=2
#
##### IEEE 802.1x related NME configuration #####################
# Valid only if WPA2 is configured via UI.
# Group key timeout in seconds after which group rekey procedure is triggered.
# Default 0 => No group rekeying.
#groupkeyTimeout=0
#
# Whether to enable group key rekeying every time a station leaves our BSS.
# Default : Disabled.
#strictGtkRekeyEnabled=0
# GroupMasterKey time out in msec.
# Default 0 => No time out.
#gmkTimeout=0
#
# Time in msec for which Wifi Stack waits for an EAPOL response after sending a request.
# Default 100 msec.
#responseTimeout=100
# Number of times an EAPOL Request is retransmitted.
#retransLimit=3
#
##### IEEE 802.11n related configuration ######################################
# Following parameters are relevent only if 11n is enabled.
# Allow reception of receiving frames that are sent using the Space-Time Block Code (STBC)
# Default : 1
rxStbc=1
# Allow RIFS (Reduced InterFrame Space) in the BSS
# Default : 0
rifsModeAllowed=0
```

6. Troubleshooting

Before starting the troubleshooting process, please verify that the pre-requisites are fulfilled, especially the Linux kernel configuration, as described in the Prerequisites section. If you are using a WF111 Evaluation Kit, make sure it has proper connection to the SDIO slot.

Info
There is wf111_supportinfo.sh script in the driver package which may help in troubleshooting.

6.1 MAC Fault

On occasion when the driver is loaded, a MAC fault message will be generated. This is nothing to worry about, the driver will automatically recover from this.

```
unifi0:      543633: MAC fault 0030, arg 0924 (x6)
```

6.2 Block Write Failed

If the EEPROM is faulty, the module's firmware may fail to start. The symptom of this is a quick stream of error messages.

```
UniFi SDIO Driver: 5.0.1 Sep  2 2011 13:25:15 CSR SME with WEXT support Kernel 3.1.0
UniFi: Using native Linux MMC driver for SDIO.
sdio bus_id:      mmc0:0001:1 - UniFi card 0x0 inserted
unifi0: Initialising UniFi, attempt 1
unifi0: Chip ID 0x07  Function 1  Block Size 512  Name UniFi-4(UF60xx)
unifi0: Chip Version 0x3A22
unifi0: Calling CsrSdioHardReset
unifi0: Falling back to software hard reset
unifi0: Chip ID 0x07  Function 1  Block Size 512  Name UniFi-4(UF60xx)
unifi0: MAILBOX2 non-zero after reset (mbox2 = ffff)
unifi0: unifi_dl_patch c2c4de98 0100060e
unifi0: Block write failed
unifi0: CMD53 failed writing 2042 bytes to handle 1
unifi0: Failed to copy block of 2042 bytes to UniFi
unifi0: Patch failed after 0 bytes
unifi0: Failed to patch image
unifi0: Failed to patch firmware
unifi0: Failed to establish communication with UniFi
unifi0: Failed to start host protocol.
unifi0: Failed to initialise UniFi chip.
unifi0: Initialising UniFi, attempt 2
unifi0: Chip ID 0x07  Function 1  Block Size 64  Name UniFi-4(UF60xx)
unifi0: Chip Version 0x3A22
unifi0: Calling CsrSdioHardReset
unifi0: Falling back to software hard reset
unifi0: Chip ID 0x07  Function 1  Block Size 64  Name UniFi-4(UF60xx)
unifi0: MAILBOX2 non-zero after reset (mbox2 = ffff)
unifi0: unifi_dl_patch c2c4de98 0100060e
unifi0: Block write failed
unifi0: CMD53 failed writing 2042 bytes to handle 1
unifi0: Failed to copy block of 2042 bytes to UniFi
unifi0: Patch failed after 0 bytes
unifi0: Failed to patch image
unifi0: Failed to patch firmware
```

6.3 Hotplug/Firmware Loader Faults

If you don't have working hotplug system (kernel options enabled, mdev/udev) you will get something like below. There is a long delay after first line.

```
unifi mmc0:0001:1: firmware: requesting unifi-sdio-0/staonly.xbv
unifi0: SDIO block size 64 requires 8 padding chunks
unifi0: UniFi f/w protocol version 8.0 (driver 9.1)
unifi0: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_gcc 1089 bfsw@eagle@630492
unifi0: UniFi f/w protocol major version (8) is different from driver (v9.1)
unifi0: Failed to establish communication with UniFi
unifi0: Failed to start host protocol.
unifi0: Failed to initialise UniFi chip.
unifi0: Initialising UniFi, attempt 2
unifi0: Chip ID 0x07  Function 1  Block Size 64  Name UniFi-4(UF60xx)
unifi0: Chip Version 0x3A22
unifi0: Calling CsrSdioHardReset
unifi0: Falling back to software hard reset
unifi0: Chip ID 0x07  Function 1  Block Size 64  Name UniFi-4(UF60xx)
```

6.4 No "unifi0: ..." Messages Appear in System Log File

When the driver is loaded using modprobe the following is output to the system log:

```
UniFi SDIO Driver: 5.1.0 May 13 2013 15:25:41
CSR SME with WEXT support
Split patch support
Kernel 3.1.10
UniFi: Using native Linux MMC driver for SDIO.
```

However nothing else gets printed, i.e. the unifi0 driver doesn't detect the card.

When the WF111 evaluation kit is not detected in the SDIO, this typically means that it is not getting power or card detect system of the SDIO driver does not detect WF111. Please check the jumper configuration and card detect signals.

6.5 Driver Goes to Sleep and Never Wakes Up

It is likely that MMC_AT91 driver is being used, but this is deprecated in latest kernel. MMC_AT91 has also multiblock transfer issue which is fixed in kernel version 2.6.37
See more about SDIO multiblock transfer fix at
http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=commit;h=a2255ff45143001fecbc5e5a4b58fcb999d393ae
Please, use MMC_ATMELMCI mmc driver instead.

This symptom also has been known to occur on a Freescale i.MX35 running Kernel 2.6.31, using the Freescale **mx_sdhci** driver. The problem in this case is that the SDIO interrupts were not indicated to the SDHC driver correctly. Disabling the host capability **MMC_CAP_SDIO_IRQ** solved the issue.

```
unifi: <= CsrSdioInterruptEnable
unifi0: New state=0
unifi0: bh_thread goes to sleep.
unifi0: bh_thread calls unifi_bh().
unifi0: bh_thread goes to sleep.
```

6.6 Invalid or Corrupted MIB File (syslog)

In case the MIB file is corrupted, the following output is expected:

```
Mar  2 08:53:48 apx4devkit daemon.notice unififw: Synergy Wifi Example Application starting...
Mar  2 08:53:48 apx4devkit daemon.notice unififw: 2013/03/02-08:53:48.670 CRT CsrWifiSme      .MIBFSM    |MIB
configuration header not 'UDMI' in file 1
Mar  2 08:53:48 apx4devkit daemon.notice unififw: 2013/03/02-08:53:48.673 ERR CsrWifiSme      .UNIFI     |Ini-
tialisation of MIB failed (MibStatus_InvalidParameters)
Mar  2 08:53:48 apx4devkit user.notice kernel: unifi0: SME client close (unifi0)
```

6.7 Missing Firmware File or Invalid Firmware Version (dmesg)

If you don't have working hotplug system (kernel options enabled, mdev/udev) you will get something like below. There is a long delay after first line.

```
unifi0: Chip ID 0x07  Function 1  Block Size 512  Name UniFi-4(UF60xx)
unifi0: Chip Version 0x3A22
unifi0: Calling CsrSdioHardReset
unifi0: Falling back to software hard reset
unifi0: Chip ID 0x07  Function 1  Block Size 512  Name UniFi-4(UF60xx)
unifi0: SDIO block size 64 requires 8 padding chunks
unifi0: UniFi f/w protocol version 8.0 (driver 9.1)
unifi0: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_gcc 1089 bfsw@eagle@630492
unifi0: UniFi f/w protocol major version (8) is different from driver (v9.1)
unifi0: Failed to establish communication with UniFi
unifi0: Failed to start host protocol.
unifi0: Failed to initialise UniFi chip.
```

6.8 Internal Reference Clock Not available

In cases when internal reference clock for some reason is not available, it is not possible to use the module even though the "UniFi ready" message is printed.

```
mmc1: new high speed SDIO card at address 0001
sdio bus_id: mmc1:0001:1 - UniFi card 0x1 inserted
SDIO: Skip power on; card is already powered.
unifi1: Initialising UniFi, attempt 1
unifi1: Chip ID 0x07 Function 1 Block Size 512 Name UniFi-4(UF60xx)
unifi1: Chip Version 0x3A22
unifi1: Calling CsrSdioHardReset
unifi1: Falling back to software hard reset
unifi1: Chip ID 0x07 Function 1 Block Size 512 Name UniFi-4(UF60xx)
unifi1: unifi_dl_patch c3bb9ebc 0100060e
unifi1: SDIO block size 64 requires 8 padding chunks
unifi1: UniFi f/w protocol version 9.1 (driver 9.1)
unifi1: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_gcc 1089 bfsw@eagle@630492
unifi1: 1047564: PHY fault 001f, arg 6590 (x1)
unifi1: Firmware patch 1247
unifi1: unifi1 is wlan0
unifi1: UniFi ready
unifi1: Failed to read from UniFi (addr 0x8644) after 3 tries
unifi1: Failed to read to-host sig written count
unifi1: Error occured reading to-host signals
unifi1: unifi_bh: state=0 A, clock=50000kHz, interrupt=0 host=0, power_save=enabled
unifi1: Failed to read valid chip version sr=4 (0x0000 want 0x3a22) try 0
unifi1: Try function enable
unifi1: Read chip version 0x3a22 after 1 retries
unifi1: Last UniFi PHY PANIC 800e arg 0002
unifi1: handle_bh_error: fatal error is reported to the SME.
unifi1: SME client close (unifi1)
unifi1: Message for the SME dropped, SME has gone away
unifi1: sme_complete_request: request not pending CsrWifiSmeWifiOffCfmHandler (s:2)
unifi1: Initialising UniFi, attempt 1
unifi1: Chip ID 0x07 Function 1 Block Size 64 Name UniFi-4(UF60xx)
unifi1: Chip Version 0x3A22
unifi1: Calling CsrSdioHardReset
unifi1: Falling back to software hard reset
unifi1: Chip ID 0x07 Function 1 Block Size 64 Name UniFi-4(UF60xx)
unifi1: Mini-coredump capture at t=625039104
unifi1: unifi_dl_patch c3bb9ebc 0100060e
unifi1: SDIO block size 64 requires 8 padding chunks
unifi1: UniFi f/w protocol version 9.1 (driver 9.1)
unifi1: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_gcc 1089 bfsw@eagle@630492
unifi1: 79991: PHY fault 0001, arg 800e (x1)
unifi1: 79994: PHY fault 0094, arg 0002 (x1)
unifi1: 1049793: PHY fault 001f, arg 6590 (x1)
unifi1: Firmware patch 1247
unifi1: UniFi ready
unifi1: Failed to read from UniFi (addr 0x8644) after 3 tries
unifi1: Failed to read to-host sig written count
unifi1: Error occured reading to-host signals
unifi1: unifi_bh: state=0 A, clock=50000kHz, interrupt=0 host=0, power_save=enabled
unifi1: Failed to read valid chip version sr=4 (0x0000 want 0x3a22) try 0
unifi1: Try function enable
```

6.9 Debugging for More Info

By using the module with a higher debug level, much more information can be obtained.
First remove the WF111 module from the SDIO bus.
To enable more verbose debugging, issue commands:

```
sudo rmmod unifi_sdio
sudo modprobe unifi_sdio unifi_debug=3
```

After issuing the above command re-insert WF111 back to the SDIO bus. An example trace of WF111 initialization log is below:

```
unifi1: UDI 0 (0xc3b55fa8) registered. configuration = 0x0
unifi1: Netdev c3b54000 client (id:0 s:0xC000) is registered
unifi: uf_netdev_event: ignore e=5, ptr=c3807c00, priv=c3807f40 lo
unifi: uf_netdev_event: ignore e=1, ptr=c3807c00, priv=c3807f40 lo
unifi: uf_netdev_event: ignore e=5, ptr=c3920800, priv=c3920b40 eth0
unifi: uf_netdev_event: ignore e=1, ptr=c3920800, priv=c3920b40 eth0
unifi1: Allocate buffers for 5 core dumps
unifi1: Core dump configured (5 dumps max)
unifi1: CsrWifiRouterTransportInit:
unifi1: run UniFi helper app...
unifi1: starting /usr/sbin/unififw
unifi1: running /usr/sbin/unififw 1 2
unifi1: UDI 1 (0xc3b56004) registered. configuration = 0x22
unifi1: SME client (id:1 s:0xC100) is registered
unifi1: UNIFI_BUILD_TYPE userspace=AP
unifi1: CsrWifiRouterCtrlWifiOnReqHandler(0x0003)
SDIO: Skip power on; card is already powered.
unifi1: Initialising UniFi, attempt 1
unifi1: Resetting UniFi
unifi1: Chip ID 0x07 Function 1 Block Size 512 Name UniFi-4(UF60xx)
unifi1: Block mode SDIO
unifi1: Chip Version 0x3A22
unifi1: Calling CsrSdioHardReset
unifi1: Falling back to software hard reset
unifi1: Chip ID 0x07 Function 1 Block Size 512 Name UniFi-4(UF60xx)
unifi1: Hard reset (IO_ENABLE)
unifi1: waiting for disable to complete, attempt 0
unifi1: Disable complete (function 1 is disabled) in ~ 0 msecs
unifi1: waiting for reset to complete, attempt 0
unifi1: Reset complete (function 1 is disabled) in ~ 0 msecs
unifi: Set SDIO function block size to 64
unifi1: unifi_configure_low_power_mode: new mode = disabled, wake_host = FALSE
unifi1: unifi_run_bh: discard message.
unifi1: waiting for MAILBOX1 to be non-zero...
unifi1: MAILBOX1 ready (0x0100) in 0 millisecs
unifi1: MAILBOX1 value=0x0100
```

Alternatively it is possible to change debug level of the driver using sysfs interface:

```
echo 3 >/sys/module/unifi_sdio/parameter/unifi_debug
```

6.10 Other Hints

Problems related to communication over the SDIO interface are the most common, so the following aspects should be checked carefully:

- How are the SDIO data lines wired? (Short wires and proper grounding should be used: remember that signal frequency is high)
- Are SDIO interrupts and SDIO multiblock tranfer supported by the SDIO host driver?
- Does decreasing the bus frequency to 400 kHz make the module initialize appropriately? (modprobe unifi_sdio sdio_clock=400)
- Does switching the bus width into 1 bit mode make the module initialize appropriately? (modprobe unifi_sdio buswidth=1)
- Does testing the latest mainline kernel make the module initialize appropriately (It might have improvements in SDIO drivers)
- Use atmel_mci driver instead of at91_mci with Atmel processors (at91_mci is known to have issues in 2.6.36 and earlier)

In case of high packet loss or low bit rate when using an external antenna:

- Check RF signal path
- Check antenna connection
- Test another antenna
- Try another Wi-Fi AP

For further help, please contact Silicon Labs' technical support and provide the following documentation or debug information depending on the actual problem being experienced:

- Design material, such as schematics (especially WF111 parts, powering and data lines between CPU and module)
- Photo of PCB (the quality should be good enough to see small RF components and SDIO signal wires of the module)
- Information about CPU model, SDIO controller and driver
- Outputs of dmesg and syslog after increasing debug level (modprobe unifi_sdio unifi_debug=3 or unifi_debug=9)
- Output of wf111_supportinfo.sh shell script which is delivered with the driver package at scripts folder. It is recommended to forward the output of supportinfo script to the log file with following syntax: "./wf111_supportinfo.sh > /tmp/supportinfo.log".
- Signal quality, signal level, noise level (iwlist scan and/or iwconfig)

7. Revision History

Revision 2.3

June, 2018

- Updated to WF111 Linux driver v5.2.2 r4.
- Added information about KRACK protection.
- Updated document format.

Revision 2.2

January, 2017

- Updated to WF111 Linux driver v5.2.2 r3.
- Removed coex MIB files.

Revision 2.1

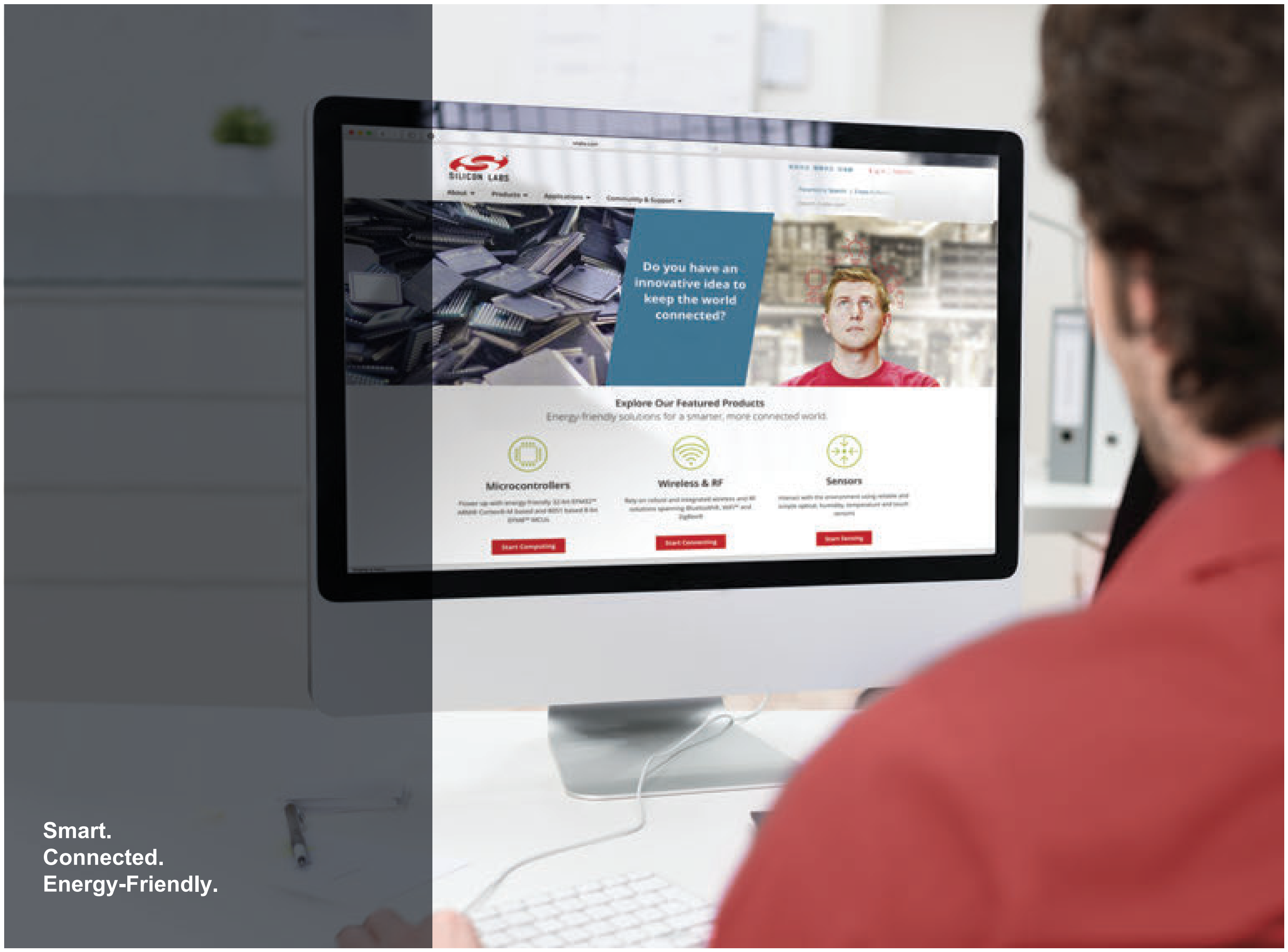
- Updated to WF111 Linux driver v5.2.2 r2.

Revision 2.0

- Updated to WF111 Linux driver v5.2.2.

Revision 1.6

- Improved driver installation instructions.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com