

UG452: RS9116N EVK Software User's Guide

Version 1.0

10/21/2020

Table of Contents

1	RS9116 EVK Overview	4
1.1	The RS9116 n-Link®.....	4
2	Evaluation Kit (EVK) Details	5
2.1	Evaluation Kit Contents.....	5
2.2	Software Package.....	5
2.2.1	Related Links	5
3	Hardware (EVB) Details.....	6
4	Driver Software Installation	8
4.1	Overview.....	8
4.2	Host Interface.....	8
4.3	Package Contents.....	8
4.4	Compilation Steps.....	8
4.4.1	Build from Kernel Source.....	9
4.4.2	Build from the Local Path.....	10
4.5	Installation Steps.....	10
4.6	Wi-Fi Only Mode Installation	13
4.6.1	Wi-Fi Station Mode.....	13
4.6.2	Wi-Fi Access Point (AP) Mode.....	19
4.7	Bluetooth Only Modes (BT/BLE) Installation	22
4.7.1	Bluetooth Classic Only Mode.....	22
4.7.2	Bluetooth LE only mode	22
4.7.3	Bluetooth Classic + Bluetooth LE Mode	23
4.8	Installation in Wi-Fi + Bluetooth Classic Coexistence Mode.....	23
4.9	Installation in Wi-Fi + Bluetooth LE Coexistence Mode.....	23
4.10	Installation in Wi-Fi + Bluetooth Classic + Bluetooth LE Coexistence Mode	24
4.11	Uninstalling the Driver	26
5	Checking Throughput.....	27
6	Appendix A: SDIO Connectivity Header Pin Description	30
6.1	SDIO Header Pin Description.....	30
7	Appendix B: Using the Bluetoothctl Application	31
8	Appendix C: Cross-Compiling Open Source Driver for Target Platforms	32
9	Appendix D: Configuration of Kernel SDIO Stack and Bluetooth Stack	33
9.1	SDIO Stack Options.....	33
9.2	Bluetooth Stack Options.....	33
10	Open Source Driver QSG Revision Report.....	34

About this Document

This document covers the RS9116 Module's Evaluation Board (EVB) and its usage for evaluating ultra-low-power, single spatial stream, dual-band 802.11n + BT 5.0 modules in n-Link® mode.

1 RS9116 EVK Overview

Silicon Labs' RS9116 module provides a comprehensive multi-protocol wireless connectivity solution including 802.11 a/b/g/n (2.4 GHz and 5 GHz) and dual-mode Bluetooth 5.0 using RS9116N Open Source Driver in n-Link mode.

1.1 The RS9116 n-Link®

The n-Link™ products provide single-band Wi-Fi (2.4GHz, 802.11bgn 1x1) and dual-band Wi-Fi (2.4/5GHz, 802.11abgn 1x1) connectivity in systems which have 32/64-bit host processor/microcontroller running Linux OS. These products come with a rich source of interfaces allowing maximum flexibility of integration into any host processor/ microcontroller systems. These can be interfaced over interfaces like SDIO, USB bus to host processor/microcontroller where generic TCP/IP network stack and Wireless stacks/profiles are running.

These n-Link products are capable of up to 100 Mbps Wi-Fi application throughputs with multiple operating modes such as Wi-Fi Client and Wi-Fi Access Point.

Solution Highlights

- Co-existence of Wi-Fi and Bluetooth protocols with single radio managed by an internal protocol arbitration manager
- Ultra-low power consumption with multiple power modes to reduce system energy consumption.
- All the products are designed to operate in an industrial (-40°C to +85°C) temperature range.
- Fully integrated and wireless certified modules with multiple sizes as small as 4.63 mm x 7.90 mm.
- Footprint compatible single-band and dual-band modules in a hosted mode for easy migration within the product family
- Leading-edge RF performance proving the best power and performance, showing robustness and reliability in a wide range of applications and power scenarios

2 Evaluation Kit (EVK) Details

2.1 Evaluation Kit Contents

The RS9116 Module Evaluation Kit comes with the following components:

1. RS9116 Module Evaluation Board (IO Base Board)
2. Wireless daughter card
3. Micro A/B-type USB cable
4. SDIO Adaptor Cable
5. Software package



Figure 1: Evaluation Kit Contents

Silicon Labs provides drivers for multiple OS's and MCU platforms for the n-Link® modules and OS-less MCU platforms for WiSeConnect® modules. The software provided in this kit is to enable easy and quick evaluation on a PC.

2.2 Software Package

The Open Source Driver package is delivered as a .zip with a filename in the format: **RS9116.NX0.NL.GNU.LNX.OSD.a.b.c.d.zip**

Where the naming convention is as follows:

NX0 – defines whether the package supports only Wi-Fi (N00) or Bluetooth Classic/Low Energy along with Wi-Fi (NB0).

a.b.c.d– identifies the software package.

2.2.1 Related Links

Refer to the links below for Product datasheets and software packages:

https://www.silabs.com/support/resources.p-wireless_wi-fi_rs9116-wi-fi-transceiver-modules

<https://www.silabs.com/wireless/wi-fi/rs9116-wi-fi-transceiver-modules/device.rs9116n-db00-cc0>

3 Hardware (EVB) Details

This section describes the RS9116 EVB's various components and headers.

The Open Source Driver software for the n-Link® modules supports SDIO and USB interfaces to connect to the Host. The same hardware also supports OneBox-Embedded software for the WiSeConnect® modules that supports UART, SPI, USB, and USB-CDC interfaces to connect to the Host MCU.

As shown in the image below, the RS9116 Evaluation Board (EVB) has three USB connectors for the USB, USB-CDC, and UART connections. The UART signals of the module are converted to USB using an onboard circuit.

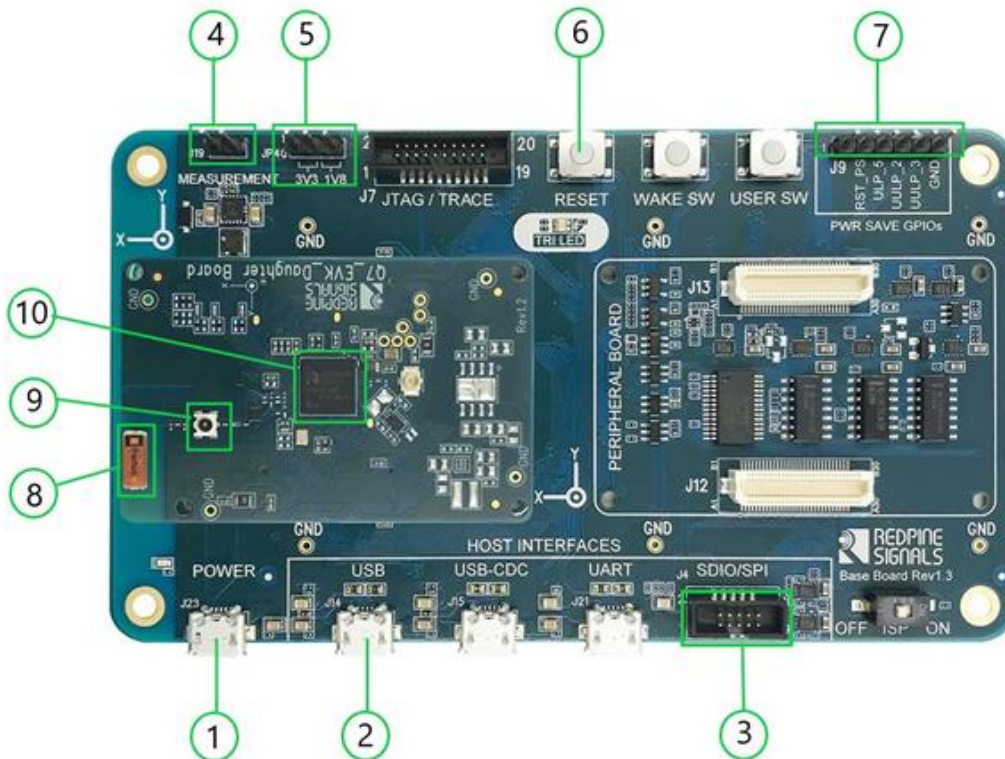


Figure 2: RS9116 EVB

The table below describes the features of the board.

Option	Feature name	Description
1	Power USB	It is used to supply power to the EVB. While using a USB interface, connecting power USB is optional as power is drawn from the USB itself.
2	USB	It is the port for the USB interface used to communicate with the host.
3	SDIO/SPI	It is the common port for both SDIO and SPI interface, used to communicate with the host. Note: In nLink only, SDIO is supported.
4	Measurement	It is a provision to measure the current consumption of the chip using an ammeter or Digital Multimeter.
5	3.3V/1.8V Voltage selection	It is a provision to select the operating voltage of the chip. The user needs to set this to either 3.3V or 1.8V selection.
6	Reset Switch	Provision to reset chip
7	Power save GPIO's	These are GPIOs that need to be connected appropriately to the host while using GPIO handshake in ULP. Please refer to the TRM for more details.

8	Onboard antenna /Internal Antenna	This is an onboard antenna used for wireless communications
9	External Antenna UFL connector	Provision given to connect external antenna as per the requirement.
10	RS9116 SOC	<p>Chip number will be printed on top of SOC which has below information included in it.</p> <p>M/N:M7DB6</p> <p>RS9116-CC0-2</p> <p>FCC ID:KFS-M7DBN6</p> <p>For details refer to RS9116 Product Brief</p>

Power supply and Interface connectivity:

The board is designed to configure the module to use the interface on which power supply is detected. This is indicated through the LEDs mounted on the board. The SDIO and SPI interfaces require power supply to be provided over the USB connector. Hence, for these interfaces, it is required that the USB connection be provided first followed by the SDIO or SPI connection.

Follow the steps below to use the EVB with different interfaces:

1. USB, UART, USB-CDC Modes
 - a. Connect the Micro A/B-type USB cable between a USB port of a PC/Laptop and the micro-USB port labelled USB, UART or USB-CDC on the EVB.
2. SDIO Mode (For SDIO Header Pin Description, please refer to section in [Appendix A SDIO Connectivity Header Pin Description](#))

Current Measurement:

There is a 2-pin inline jumper available for measuring the current being consumed by the module during different stages of operation. This is labelled as "MEASUREMENT" on the baseboard. The user may connect a power meter or an ammeter to this jumper to measure the current.

Note

Make sure the ISP switch is in OFF state. If it is ON state you will not get the boot loader messages.

Important Note: If the baseboard Rev is 1.1 or below than follow below procedure:

Note

1. For SDIO/SPI, insert the USB into the Power port first before the SDIO/SPI connector is connected to the Host platform
2. For USB and USB-CDC, please connect the USB port to the Host platform first before connecting the USB for the Power port.

4 Driver Software Installation

4.1 Overview

Open Source Driver (OSD) is a SoftMAC driver which interacts with the Linux wireless MAC layer i.e., MAC80211. The driver is a group of simple and efficient kernel modules which currently supports Silicon Labs 9116 chipsets and it can be ported to any embedded platform in-addition to X-86 platform. It supports the following protocols:

- Wi-Fi (Client and Access point modes)
- Bluetooth Classic
- Bluetooth Low Energy

It supports the following combinations :

- WLAN STATION
- WLAN ACCESS POINT
- BT CLASSIC MODE
- BT LE MODE
- WLAN STATION + BT CLASSIC MODE
- WLAN STATION + BT LE MODE
- BT CLASSIC + BT LE MODE
- WLAN STATION + BT CLASSIC MODE + BT LE MODE

4.2 Host Interface

The driver supports two host interfaces USB and SDIO, which can be selected while building the modules.

4.3 Package Contents

The driver package contains the following files/folders:

- Readme.txt
- ReleaseNotes.pdf
- Documents
- Firmware
- rsi (contains driver source code)
- scripts

The drivers can be found under each Product on our website. Please find an example location below:

https://www.silabs.com/support/resources.p-wireless_wi-fi_rs9116-wi-fi-transceiver-modules

4.4 Compilation Steps

This section describes the steps to be followed in order to compile the RSI OSD for different platforms. The steps are outlined below:

1. Extract the package using the following command:


```
# unzip RS9116.NX0.NL.GNU.LNX.OSD.<version>.zip
```

2. Go to the package and copy all the files present in Firmware folder to '/lib/firmware' by following the below command.

```
# cd RS9116.NX0.NL.GNU.LNX.OSD.<version>
# cp Firmware/* /lib/firmware
```

3. To build the driver, there are two ways.
 - a. Build from kernel source
 - b. Build from the local path

4.4.1 Build from Kernel Source

- Copy the driver 'rsi' to <kernel_source_path>/drivers/net/wireless/. (Ex : linux-5.7.0/drivers/net/wireless/rsi)
- Go to rsi directory and Move Makefile to Makefile_local.

```
# mv Makefile Makefile_local
```

- Move Makefile_ker to Makefile.

```
# mv Makefile_ker Makefile
```

- Give 'make menuconfig' from kernel source directory. (Ex : linux-5.7.0/)

```
# make menuconfig
```

- Go to 'Device Drivers->Network device support->Wireless LAN'.
- Select 'Redpine Signals Inc' devices.
- Select the SDIO/USB bus support depending on the requirement. You will see the below screen with all the build options mentioned. Select the required options.

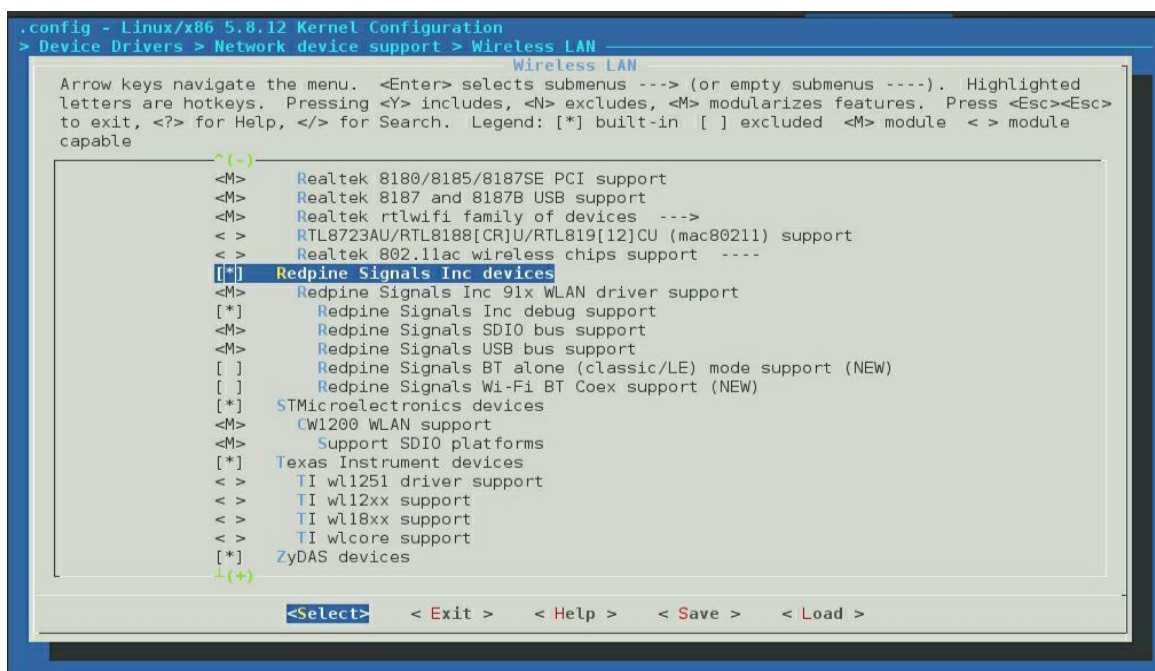


Figure 3: Selection of Required Configuration

- Build driver by using the below commands

```
# make M=drivers/net/wireless/rsi
```

4.4.2 Build from the Local Path

1. Configure build flags in driver source.

```
# cd rsi
```

2. Open Makefile and configure build flags. Below are the flags to be set based on the usage of driver. Selecting the required options shall reduce the binary size which is important for kernel modules particularly on embedded platforms.

- a. **KERNELDIR** : Provide the kernel source path here. For example, on X-86 below path is used.

```
KERNELRELEASE=$(Shell uname -r)
KERNELDIR=/lib/modules/$(KERNELRELEASE)/build
```

- b. **CONFIG_RSI_COEX_MODE** : Enable this flag when Wi-Fi and BT co-existence mode is used.
- c. **CONFIG_RSI_DEBUGFS** : It is used by driver to take dynamic configuration from user. Supported debugfs based configurations are listed in the corresponding feature sections in TRM.
- d. **CONFIG_RSI_BT_ALONE** : Enable this flag when only BT EDR/ BT LE only mode is used.

3. Build the driver using make command.

```
# make
```

For embedded platforms, add the target platform and toolchain path as cross compilation option to the **"make"** command.

For example, if the target platform is iMX6 add the kernel path as below :

```
KERNELDIR=home/test/Wand/armv7-multiplatform/KERNEL
```

For example, if the target platform is ARM and tool chain path is ***"/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-multilib-1.0/arm-fsl-linux-gnueabi/bin/arm-none-linux-gnueabi-"***, then the command is issued as:

```
# make ARCH=arm CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-multilib-1.0/arm-fsl-linux-gnueabi/bin/arm-none-linux-gnueabi-
```

On successful compilation, make will generate rsi_91x.ko, rsi_usb.ko and/or rsi_sdio.ko according to the configuration.

4.5 Installation Steps

In order to install the driver, use the following commands:

- 1.) Before installing driver install the dependencies using below commands

```
# modprobe mac80211
# modprobe cfg80211
# modprobe bluetooth
```

- 2.) Insert rsi_91x.ko with the required module params (configuration) as shown below:

```
# insmod rsi_91x.ko dev_oper_mode=<mode> rsi_zone_enabled=<val> ...
```

Module params are used by the driver to take initial configuration required. If not provided, default configuration is used. For most of the applications, default values of these module params will be sufficient. Supported module params with their configurable limits are explained in the TRM in respective feature sections and/or **Appendix B: Initial Configuration Parameters** in TRM.

In the above command example, module param **rsi_zone_enabled** is to enable debug prints in dmesg. Default value of rsi_zone_enabled value is 1, which prints errors (only) in terminal. Please refer **Appendix A: Driver Details** in the OSD TRM, to enable more debug prints.

dev_oper_mode : Device operating mode indicates the possible combination of the wireless protocols that can configured with the device.

Below table provides the operating mode details with its constraints.

S.No	Operating Mode	Protocol Support				Maximum No. of clients for WLAN AP	Maximum No. of BT connections	Maximum No. of BLE Connections
		STA	AP	BT EDR	BT LE			
1	1	√	X	X	X	N/A	1	2 (Can as master for 2 Slaves connections Or Can be as master for one slave connection and can connect to other master as slave)
2	1	X	√	X	X	16 clients		
3	4	X	X	√	X	N/A		
4	5	√	X	√	X	N/A		
5	6	X	√	√	X	16 clients		
6	8	X	X	X	√	N/A		
7	9	√	X	X	√	N/A		
8	12	X	X	√	√	N/A		
9	13	√	X	√	√	N/A		
10	14	X	√	√	√	4 clients		

Note

In the above table AP with other protocols combination (Example dev_oper_mode value 6 and 14) are not supported in this release. Also STA+BT+BLE (Example dev_oper_mode value 13) is not supported in this release.

If any invalid mode is passed to the module, driver returns error and exit. You can check the error message debug logs.

Note

For modes 4 ,8 and 12 build flag CONFIG_RSI_BT_ALONE should be enabled in the driver Makefile.
For modes 5, 9, 6,13 and 14, build flag CONFIG_RSI_COEX_MODE should be enabled in the driver Makefile.

2.) For USB interface, enter the below command:

```
# insmod rsi_usb.ko
```

3.) For SDIO interface, enter the below command:

```
# insmod rsi_sdio.ko sdio_clock=<clk_val>
```

Note: Here "clk_val" is 1 to 50 (in MHz's).

You can install either USB or SDIO or both depending upon the selection of the interface while compiling the driver. After successful installation, if module (EVB) is inserted to the PC/platform a new wireless interface shall be created for WLAN and/or BT/BLE as per the dev_oper_mode selection.

a. If **WLAN** is selected, interface details can be verified using below commands.

Name of the Wi-Fi interface created after successful installation of the driver can be seen using 'ifconfig' command.

```
# ifconfig -a
```

You should expect an output like the sample shown below with all other available interfaces included.

```
wlan0    flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
         inet6 fe80::8da:1aff:fe1e:d1c8 prefixlen 64 scopeid 0x20<link>
         ether 88:da:1a:1e:d1:c8 txqueuelen 1000 (Ethernet)
         RX packets: 3 bytes 372 (372.0 B)
         RX errors 0 dropped 0 overruns 0 frame 0
         TX packets: 6 bytes 696 (696.0 B)
         TX errors 0 dropped 0 overruns 0 collisions:0
```

Also, for **WLAN** there is another utility (iw) with which you can get the interface and physical device details. For example, below command will show the interface status and physical device number.

```
# iw dev <interface_name> info
```

The sample output for this command is shown below.

```
Interface wlan0
- ifindex 5
- wdev 0x100000001
- addr 00:23:a7:b9:ab:44
- type managed
- wiphy 1
- channel 6 (2437 MHz), width: 20 MHz (no HT), center1: 2437 MHz
```

As can be seen, in this case, phy<X> is termed as wiphy 1.

b. If **BT/BLE** is selected, interface details can be verified using below command.

Name of the BT/BLE interface created after successful installation of the driver can be seen using 'hciconfig' command.

```
# hciconfig -a
```

```
hci0: Type: BR/EDR Bus: USB
BD Address: 88:DA:1A:00:00:C2 ACL MTU: 1021:3 SCO MTU: 64:3
DOWN
RX bytes:1006 acl:0 sco:0 events:55 errors:0
TX bytes:0 acl:0 sco:0 commands:55 errors:0
Features: 0xbf 0xfe 0x0d 0xfe 0xdb 0xff 0x5b 0x87
Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
Link policy: RSWITCH SNIFF
Link mode: SLAVE ACCEPT
Name: 'lapt64'
class: 0x0c010c
service Classes: Computer, Laptop
HCI Version: (0x9) Revision: 0x0
Manufacturer: internal use (65535)
```

4.6 Wi-Fi Only Mode Installation

4.6.1 Wi-Fi Station Mode

This section provides steps to configure Wi-Fi station mode using both wpa_supplicant and the Network Manager CLI. Both procedures are given below. User can choose any method.

Before installation, user needs to stop the existing network manager and unblock WLAN from rfkill. Below commands are used to stop the network-manager on different Linux distribution.

1. For ubuntu, we need to use below command.

```
# service network-manager stop
```

2. For fedora, we need to use below command.

```
# service NetworkManager stop
```

3. To stop rfkill blocking WLAN, we need to use below command.

```
# rfkill unblock wlan (or) #rfkill unblock all
```

- For station mode connectivity, ensure that the dev_oper_mode is set in installation as given below and interface is detected after installation (Refer to [Installation Steps](#) section).

```
dev_oper_mode = 1
```

4.6.1.1 Configure Station using WPA_supplicant

1.) Create a **sta_settings.conf** file with below information. Also please fill the information like ssid, psk etc corresponding to the AP you intend to connect in this file. Sample sta_settings.conf file is available within scripts directory of release package with basic configurations required. User may use this file and edit the information as explained below. For the details of all configurations available please refer open source supplicant wpa_supplicant.conf file.

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

Also add network block to the sta_settings.conf file as per the AP security. Example network block for different security modes are listed below.

i. For Open (non-Secure) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=NONE
    priority=3
}
```

ii. For WPA2-PSK (CCMP) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-PSK
    psk=<passphrase specified in the Access Point>
    proto=WPA2
    pairwise=CCMP
    group=CCMP
}
```

The pass phrase can be input either in ASCII or Hexadecimal formats:

ASCII Format: psk="very secret passphrase"

Hexadecimal Format: psk= 06b4be19da289f475aa46a33cb793029d4ab3db7a23ee92382eb0106c7

iii. For WPA3 security mode

To connect in WPA3, we need to compile the latest supplicant with below flags enabled in wpa_supplicant .config file

```
CONFIG_SAE=y
CONFIG_IEEE80211W=y
```

```
pmf=2
network={
    ssid="<SSID of Access Point>"
    key_mgmt=SAE
    psk=<passphrase specified in the Access Point>
    ieee80211w=2
}
```

WPA3 Enterprise security mode is not supported in this release.

iv. For WPA2-EAP TLS (Enterprise mode) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=TLS
    anonymous_identity="tlsuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    ca_cert="/etc/certs/wifiuser.pem"
    client_cert="/etc/certs/wifiuser.pem"
    private_key_passwd=<private key password>
    private_key="/etc/certs/wifiuser.key"
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

In EAP-TLS user has to copy client certificates in a path and the path need to be configured in network block as given above.

v. For WPA2-EAP PEAP (Enterprise mode) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=PEAP
    anonymous_identity="peapuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

vi. For WPA2-EAP TTLS (Enterprise mode) mode:

```
network={
    ssid="<SSID of Access Point>"
    key_mgmt=WPA-EAP
    eap=TTLS
    anonymous_identity="ttlsuser"
    identity="test"
    password=<passphrase specified in the Access Point>
    pairwise=CCMP TKIP
    group=CCMP TKIP
    proto=WPA2 WPA
    priority=20
}
```

To connect to an Access Point whose SSID is not broadcast (Hidden), add the following line to the network block.

```
scan_ssid=1
```

For example:

```
network={
    ssid="<SSID of Access Point>"
    scan_ssid=1
    key_mgmt=NONE
}
```

2.) Start the supplicant using below command:

```
# wpa_supplicant -i <interface_name> -D nl80211 -c sta_settings.conf -dt > supp.log &
```

- "-i" option specifies the Wi-Fi interface name
- <interface_name> - This name as listed in iw dev output
- "-D" specifies the driver interface to be used. In open source driver it is nl80211.
- "-c" specifies the supplicant configuration file.
- "-d" specifies the log level of supplicant. You can append more d's to it for more detailed logs.

3.) To check the scan results please use below command.

```
# wpa_cli -i <interface_name> -p scan_results
```

For example, above command will give scan results output as follows.

bssid	/ frequency	/ signal level	/ flags	/ ssid
50:d4:f7:1e:5a:40	2457	-21	[WPA2-PSK-CCMP] [ESS]	TP_LINK
04:79:70:72:03:e7	2412	-31	[ESS]	honor_9i

4.) To check whether the connection is successful or not use below command

```
# iwconfig <interface_name>
```

For example, if connection is successful we will see below output

```
wlan0      IEEE 802.11bgn  ESSID:"Range"  Nickname:""
Mode:Managed  Frequency:2.412 GHz  Access Point: 38:A4:ED:DE:BB:06
Bit Rate:39 Mb/s   Tx-Power=16 dBm
Retry short limit:7   RTS thr:2353 B   Fragment thr:2352 B
Encryption key:off
Power Management:off
Link Quality=80/80  Signal level=-28 dBm  Noise level:0 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

If the connection is successful, then the connected Access point SSID along with the MAC address is displayed as shown above. If it is not connected to an Access point, a message **"Not Associated"** is displayed as shown below.


```
wlan0      IEEE 802.11  ESSID:off/any
           Mode:Managed  Access Point: Not-Associated  Tx-Power=0 dBm
           Retry short limit:7   RTS thr:off   Fragment thr:off
           Encryption key:off
           Power Management:off
```

5.) IP for the device can be set in two ways either get IP dynamically from AP or set static IP. To obtain dynamic IP from AP, use below commands.

```
# dhclient < interface_name > -r
# dhclient < interface_name > -v
```

To set static IP to STA use below command.

```
# ifconfig <interface_name> <IP_address>
```

6.) To check whether IP is assigned or not use below command.

```
# ifconfig <interface_name>
```

Output:

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.114  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::224:d7ff:fe56:54dc  prefixlen 64  scopeid 0x20<link>
        ether 00:24:d7:56:54:dc  txqueuelen 1000  (Ethernet)
        RX packets 31160  bytes 31082515 (29.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23356  bytes 3367496 (3.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4.6.1.2 Configure Station using the Network-Manager CLI (nmcli)

Below are the specific commands that can be used for connection using the Network Manager CLI(nmcli):

1.Check the network manager status (started or stopped) with below command.

For fedora,

```
# service NetworkManager status
```

For ubuntu,

```
# service network-manager status
```

2. If the network manager is inactive or not started, start it with the below command.

For fedora,

```
# service NetworkManager start
```

For ubuntu,

```
# service network-manager start
```

3.To view the currently available network connections, enter the following on command prompt:

```
# nmcli con show
```








sample output:

NAME	UUID	TYPE	DEVICE
eth0	96a5deb0-5eb0-41e1-a7ed-38fea413f9c8	802-3-ethernet	eth0
wlan0	91451385-4eb8-4080-8b82	802-11-wireless	wlan0

4. To view the list of access points, issue the below command:

```
# nmcli dev wifi list
```

Sample output is shown below.

*	SSID	MODE	CHAN	RATE	SIGNAL	BARS	SECURITY
	ASUS_5G	Infra	36	54 Mbit/s	100		WPA2
	ASUS	Infra	11	54 Mbit/s	100		WPA2
	test123	Infra	8	54 Mbit/s	32		WPA1 WPA2
	cisco	Infra	1	54 Mbit/s	30		WPA1 WPA2
	test	Infra	13	54 Mbit/s	25		---
	Dlink	Infra	1	54 Mbit/s	0		WPA2
	TP-LINK_E11946	Infra	7	54 Mbit/s	83		WPA1 WPA2

5. For connecting to an AP with WPA/WPA2 security, issue the below command:

```
# nmcli dev wifi connect ASUS password 12345678 <interface_name>
```

Here, ASUS is the AP's SSID and password is 12345678.

6. For connecting to an AP without security, issue the below command:

```
# nmcli dev wifi connect test <interface_name>
```

'test' is the SSID .

7. To know the status of the devices and the connections, issue the below command:

```
# nmcli dev status
```

Sample output:

DEVICE	TYPE	STATE	CONNECTION
wlan0	Wi-Fi	connected	my-ssid
eth0	ethernet	unavailable	--

As can be seen, the STATE corresponding to wlan0 interface shows connected.

8. To Enable (to make active) a connection on interface, using nmcli, issue the below command. connection_name can be obtained from above command.

```
# nmcli con up id <connection_name>
```

9. To Disable an interface using nmcli, issue the below command:

```
# nmcli dev disconnect <interface_name>
```

4.6.2 Wi-Fi Access Point (AP) Mode

This section provides steps to configure Wi-Fi AP mode using hostapd application.

- For AP mode connectivity, ensure that the dev_oper_mode is set in installation as given below and interface is detected after installation (Refer to [Installation Steps](#)).

```
dev_oper_mode = 1
```

Follow below steps for running AP mode with hostapd application.

- Before running hostapd make sure wpa_supplicant is not running in the background. If it is running kill wpa_supplicant using below command

```
# killall wpa_supplicant
```

- Install hostapd.

```
# apt-get install hostapd
```

- Configure Hostapd

Create a hostapd configuration file (for eg: ap.conf) and add below:

- Create **ap.conf** file with below information. Sample .conf files (ap_open.conf, ap_wpa.conf) are available within scripts directory of release package with basic configurations required. User may use this file and edit the information as explained below. For the details of all configurations available please refer open source hostapd **hostapd.conf** file.

- Set interface name:

```
interface=<interface_name>
Ex: interface=wlan0
```

- Set driver name:

```
driver=nl80211
```

- Set country name code in ISO/IEC 3166-1 format. This is used to set regulatory domain. Set as needed to indicate country in which device is operating. This can limit available channels and transmit power. For example, IN for India, UK for United Kingdom, US for the United States of America.

```
country_code=IN
```

- e. Set your SSID: Here, we have set 'Test_AP' as ssid for example.

```
ssid=Test_AP
```

- f. Set operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g)

```
hw_mode=g
```

- g. Set channel number

```
channel=6
```

User may select required channel of operation or he may opt for Auto Channel Selection (ACS). In case of ACS, user has to follow below procedure.

Compile hostapd with below flag set in its .config file.

CONFIG_ACS=y

Also user has to add below configurations to hostapd.conf file.

channel=0

acs_num_scans=5 (Default Value, user may select)

- h. Set Beacon Interval:

```
beacon_int=100
```

User may select required beacon interval within the range of 56-1000 ms. For value less than 56ms or more than 1000ms driver will return an error.

The default value is 100ms.

- i. Set wpa mode to 2:

```
wpa=2
```

- j. Set your passphrase (Wi-Fi password):

```
wpa_passphrase=MyWiFiPassword
```

- k. Set key and auth options for WPA2:
Set the key management algorithm as shown.

```
wpa_key_mgmt=WPA-PSK
```

Set cipher suites i.e., encryption algorithms:

TKIP stands for Temporal Key Integrity Protocol and CCMP is AES in Counter mode with CBC-MAC .

```
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Shared Key Authentication:

```
auth_algs=1
```

Save and close the file.

- I. Start the hostapd application:

```
# hostapd ap.conf -dddt > log_file &
```

- m. To check if AP mode is successfully started or not use below command:

```
# iw dev
```

For example If the AP is successfully started ,expect the below sample output i.e with SSID and channel information:

```
phy#10
Interface wlan0
  ifindex 13
  wdev 0xa00000001
  addr 88:da:1a:78:06:e4
  ssid rsi_ap_wpa
  type AP
  channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz
```

And If AP failed to start, output doesnot show SSID and channel information as shown in below sample output :

```
phy#10
Interface wlan0
  ifindex 13
  wdev 0xa00000001
  addr 88:da:1a:78:06:e4
  type managed
```

- n. Run dhcp server script (In scripts folder) to assign IPs to client.

```
# sh dhcp_server.sh <interface_name>
```

dhcp_server.sh script uses dhcpd.conf file for required configurations. User may modify this file as per the requirement.

In the scripts folder, several hostapd config files are provided to start the AP in various modes like open (ap_open.conf), WPA/2-PSK (ap_wpa.conf). User could use these conf files instead of creating new ones.

For other configurations of hostapd (ex. ACL Policy, Keep Alive) for AP mode please refer **Appendix C: Hostapd usage guidelines** in Open Source Driver TRM.

Please refer to [Checking Throughput](#) section for measuring Wi-Fi performance through UDP/TCP protocols using iperf application.

4.7 Bluetooth Only Modes (BT/BLE) Installation

4.7.1 Bluetooth Classic Only Mode

This section provides steps to configure BT classic mode. BT classic mode supports only one connection.

- For BT classic usage, ensure that the dev_oper_mode is set in installation as given below and BT interface is detected after installation (Refer [Installation Steps](#) section).

```
dev_oper_mode = 4
```

- Bring hci interface up with below command.

```
# hciconfig -a hci0 up
```

- After the device is up, we can pair it with the other devices using the Bluetooth Manager application or bluetoothctl application. The files can also be sent and received using Bluetooth Manager. Instead of Bluetooth Manager, the device can be configured using "hcidtool" or "hciconfig" as given below.
 - The procedure for using bluetoothctl is explained in [Appendix B Using the Bluetoothctl application](#)
 - The procedure for using Bluetooth Manager is explained in the section **Appendix H: Using the Bluetooth Manager** in OSD TRM.
- To start BT-Classic scanning, we need to give below command.

```
# hcidtool -i hci0 scan
```

Once BT-classic connectivity completed, we can use l2test application for connectivity status as given below.

```
# l2test -i hci0 -r (on RSI BT device side)
```

```
# l2test -i hci0 -s < BD address of RSI BT devices>(on third party BT device side)
```

4.7.2 Bluetooth LE only mode

This section provides steps to configure Bluetooth LE mode. Bluetooth LE mode supports maximum of 2 connections, i.e. Can be as master for 2 Slaves connections (Or) Can be as master for one slave connection and can connect to other master as slave.

- For Bluetooth LE usage, ensure that the dev_oper_mode is set in installation as given below and BLE interface is detected after installation (Refer to [Installation Steps](#)).

```
dev_oper_mode = 8
```

- Bring hci interface up with below command.

```
# hciconfig -a hci0 up
```

- After the device is up, we can Advertise, Scan and Connect with other BLE devices. The device can be configured using hcidtool or hciconfig.
- Advertise, Scan, Connect Commands
 - Enable Advertise

```
# hciconfig -a <hciX> leadv
```

b. Disable Advertise

```
# hciconfig -a <hciX> noleadv
```

c. Initiate Scan - Below command displays the scan responses and advertising information.

```
# hcitool -i <hciX> lescan
```

d. Master Mode Connected State, Ensure that the remote device is in Advertise mode and then issue the command below.

```
# hcitool -i <hciX> lecc <remote_MAC_Addr>
```

The "remote_MAC_Addr" parameter above is the MAC address of the remote device, e.g., 00:23:AC:01:02:03.

e. Slave Mode Connected State, ensure that our device is in Advertise mode and then issue command below.

```
# hcitool -i <hciX> lecc <device_MAC_Addr>
```

4.7.3 Bluetooth Classic + Bluetooth LE Mode

Ensure that the dev_oper_mode is set as below

- dev_oper_mode = 12

Once the installation completed and interface of Bluetooth is detected follow below procedure to install BT and BLE.

1. For Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode installation](#).
2. Also for Bluetooth classic protocol, follow the instructions given in [Bluetooth classic only mode installation](#).

4.8 Installation in Wi-Fi + Bluetooth Classic Coexistence Mode

Ensure that the dev_oper_mode is set as below

- dev_oper_mode = 5 (Wi-Fi STA + BT)
- dev_oper_mode = 6 (Wi-Fi AP + BT)

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BT.

1. For Wi-Fi, follow the instructions given in [Wi-Fi Station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.
2. Also for Bluetooth classic protocol, follow the instructions given in [Bluetooth classic only mode installation](#).

Note

Wi-Fi AP with BT combination is not supported in this release.

4.9 Installation in Wi-Fi + Bluetooth LE Coexistence Mode

Ensure that the dev_oper_mode is set as below.

- `dev_oper_mode = 9` (Wi-Fi STA + BLE)

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BLE.

1. For Wi-Fi, follow the instructions given in [Wi-Fi station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.
2. Also for Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode installation](#).

Note

Wi-Fi AP with BT LE combination is not supported in this release.

4.10 Installation in Wi-Fi + Bluetooth Classic + Bluetooth LE Coexistence Mode

Ensure that the `dev_oper_mode` is set as below

- `dev_oper_mode = 13` (Wi-Fi STA + BT + BLE)
- `dev_oper_mode = 14` (Wi-Fi AP + BT + BLE)

Note

Wi-Fi AP with BT + BT LE combination and STA+BT+BLE combinations (i.e. `dev_oper_mode` value 13 & 14) are not supported in this release.

Once the installation completed and interface of Wi-Fi and Bluetooth are detected follow below procedure to install Wi-Fi and BLE.

1. For Wi-Fi, follow the instructions given in [Wi-Fi station mode](#) or [Wi-Fi Access Point \(AP\) mode](#) in alone mode for Wi-Fi installation.
 2. Also for Bluetooth LE protocol, follow the instructions given in [Bluetooth LE only mode installation](#).
 3. Also for Bluetooth classic protocol, follow the instructions given in [Bluetooth classic only mode installation](#).
- In coex modes, to know the device type for BT i.e., device is supporting LE or BR/EDR use below command.

```
# hciconfig -a hci<x> features
```

Example Output 1:

For LE Opermode i.e `dev_oper_mode = 8`


```

hci1:    Type: BR/EDR  Bus: USB
BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 251:5  SCO MTU: 0:0
Features page 0: 0xbf 0xfe 0x0d 0xbe 0xfb 0xff 0x41 0x85
          <3-slot packets> <5-slot packets> <encryption> <slot offset>
          <timing accuracy> <role switch> <sniff mode> <RSSI>
          <channel quality> <SCO link> <HV2 packets> <HV3 packets>
          <u-law log> <A-law log> <CVSD> <power control>
          <transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
          <enhanced iscan> <interlaced iscan> <interlaced pscan>
          <extended SCO> <EV4 packets> <EV5 packets> <AFH cap. slave>
          <AFH class. slave> <BR/EDR not supp.> <LE support>
          <3-slot EDR ACL> <5-slot EDR ACL> <sniff subrating>
          <pause encryption> <AFH cap. master> <AFH class. master>
          <EDR eSCO 2 Mbps> <EDR eSCO 3 Mbps> <3-slot EDR eSCO>
          <extended inquiry> <non-flush flag> <LSTO> <EPC>
          <extended features>
Features page 1: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

```

Example Output 2:

For Classic (BT BR/EDR) Only i.e dev_oper_mode = 4

```

hci1:    Type: BR/EDR  Bus: USB
BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 1021:3  SCO MTU: 64:3
Features page 0: 0xbf 0xfe 0x0d 0xfe 0x9b 0xff 0x59 0x87
          <3-slot packets> <5-slot packets> <encryption> <slot offset>
          <timing accuracy> <role switch> <sniff mode> <RSSI>
          <channel quality> <SCO link> <HV2 packets> <HV3 packets>
          <u-law log> <A-law log> <CVSD> <power control>
          <transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
          <enhanced iscan> <interlaced iscan> <interlaced pscan>
          <inquiry with RSSI> <extended SCO> <EV4 packets> <EV5 packets>
          <AFH cap. slave> <AFH class. slave> <3-slot EDR ACL>
          <5-slot EDR ACL> <sniff subrating> <pause encryption>
          <AFH cap. master> <AFH class. master> <EDR eSCO 2 Mbps>
          <EDR eSCO 3 Mbps> <3-slot EDR eSCO> <extended inquiry>
          <simple pairing> <encapsulated PDU> <non-flush flag> <LSTO>
          <inquiry TX power> <EPC> <extended features>
Features page 1: 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Features page 1: 0x30 0x00 0x00 0x00 0x00 0x00 0x00 0x00

```

Example Output 3:

For the Classic and LE i.e dev_oper_mode = 12

```
hci1:      Type: BR/EDR  Bus: USB
          BD Address: 88:DA:1A:16:E4:4F  ACL MTU: 1021:3  SCO MTU: 64:3
          Features page 0: 0xbf 0xfe 0x0d 0xfe 0xdb 0xff 0x5b 0x87
                        <3-slot packets> <5-slot packets> <encryption> <slot offset>
                        <timing accuracy> <role switch> <sniff mode> <RSSI>
                        <channel quality> <SCO link> <HV2 packets> <HV3 packets>
                        <u-law log> <A-law log> <CVSD> <power control>
                        <transparent SCO> <EDR ACL 2 Mbps> <EDR ACL 3 Mbps>
                        <enhanced iscan> <interlaced iscan> <interlaced pscan>
                        <inquiry with RSSI> <extended SCO> <EV4 packets> <EV5 packets>
                        <AFH cap. slave> <AFH class. slave> <LE support>
                        <3-slot EDR ACL> <5-slot EDR ACL> <sniff subrating>
                        <pause encryption> <AFH cap. master> <AFH class. master>
                        <EDR eSCO 2 Mbps> <EDR eSCO 3 Mbps> <3-slot EDR eSCO>
                        <extended inquiry> <LE and BR/EDR> <simple pairing>
                        <encapsulated PDU> <non-flush flag> <LSTO> <inquiry TX power>
                        <EPC> <extended features>
          Features page 1: 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x00
          Features page 2: 0x30 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

4.11 Uninstalling the Driver

To uninstall the driver follow the below procedure.

If wpa supplicant method is used to connect in STA mode, kill wpa supplicant using below command.

```
# killall wpa_supplicant
```

To kill hostapd application, use the below command.

```
# killall hostapd
```

To remove driver use below commands.

```
# rmmod rsi_usb
# rmmod rsi_sdio
# rmmod rsi_91x
```

After uninstalling the driver, the created wireless interface disappears.

5 Checking Throughput

You can measure Wi-Fi performance through UDP/TCP protocols by using the iperf application. If you wish to evaluate the throughputs in Wi-Fi Client mode, you will need to connect a second PC/Laptop to the Access Point. Download and install the iperf application from <https://iperf.fr> on the second PC/Laptop.

Please note the following points for this evaluation:

1. To evaluate the module for throughput performance, it's recommended that you connect the second PC/Laptop to the Access Point over Ethernet as shown below.

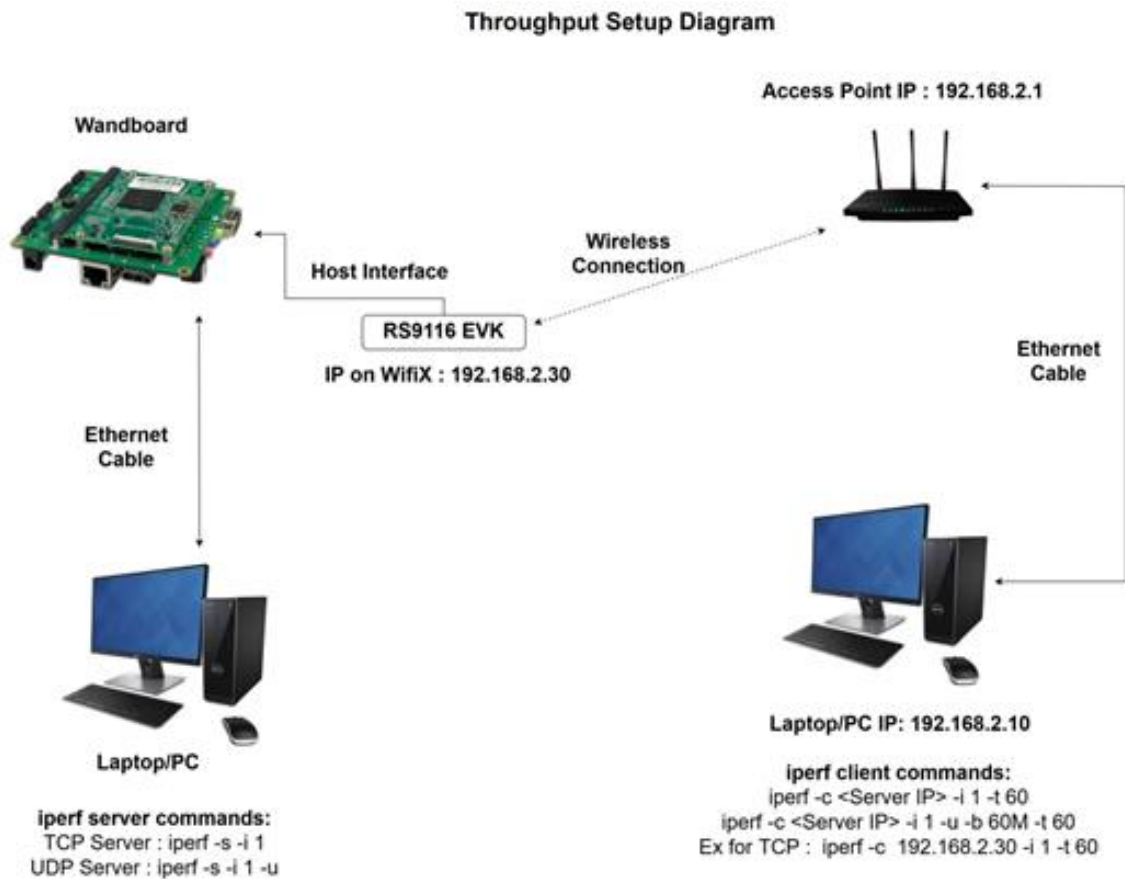


Figure 4: Throughput Measurement Setup Diagram

2. It is recommended that you choose a Wi-Fi channel which has less interference, preferably in the 5GHz band if the EVB is for a Dual band module, to observe optimal throughput of the module.
3. Wireless throughput vary with the environment of the test setup – distance, obstacles, type of obstacles, etc.
4. The throughput is also dependent on whether all the Wi-Fi components in the test support 40MHz and 20MHz or only 20MHz bandwidths.
5. To check Transmit throughput, first run the command below on the 2nd PC/Laptop:

```
# iperf -s -u -i 1
```

This command starts an iperf Server which is listening for data. It prints the received throughput at an interval of 1 second. The <-u>option in the above command is for UDP traffic. If it is not mentioned, then the traffic is TCP.

6. Next, run the command below on the PC/Laptop connected to the EVB.

```
# iperf -c <IP address of 2nd PC> -u -i 1 -b 100M -t 120
```

This command starts an iperf Client which starts transmitting data to the Server. The <-u> and <-b 100m> options in the above command are for UDP traffic. If they are not mentioned, then the traffic is TCP.

7. To check Receive throughput, run the Server command first on the PC/Laptop connected to the EVB, followed by the client command on the 2nd PC/Laptop.

Example of running throughput:

TCP Server:

```
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 51568
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0- 1.0 sec  5.19 MBytes 43.5 Mbits/sec
[ 4] 1.0- 2.0 sec  5.63 MBytes 47.2 Mbits/sec
[ 4] 2.0- 3.0 sec  5.77 MBytes 48.4 Mbits/sec
[ 4] 3.0- 4.0 sec  5.94 MBytes 49.8 Mbits/sec
[ 4] 4.0- 5.0 sec  5.92 MBytes 49.6 Mbits/sec
[ 4] 5.0- 6.0 sec  5.88 MBytes 49.3 Mbits/sec
[ 4] 6.0- 7.0 sec  5.93 MBytes 49.7 Mbits/sec
[ 4] 7.0- 8.0 sec  5.86 MBytes 49.1 Mbits/sec
[ 4] 8.0- 9.0 sec  5.92 MBytes 49.6 Mbits/sec
[ 4] 9.0-10.0 sec  5.83 MBytes 48.9 Mbits/sec
[ 4] 0.0-10.2 sec  59.2 MBytes 48.6 Mbits/sec
```

TCP Client:

```
-----
Client connecting to 192.168.0.2, TCP port 5001
TCP window size: 43.8 KByte (default)
-----
[ 3] local 192.168.0.14 port 45964 connected with 192.168.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec  5.25 MBytes 44.0 Mbits/sec
[ 3] 1.0- 2.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 2.0- 3.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 3.0- 4.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 4.0- 5.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 5.0- 6.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 6.0- 7.0 sec  5.00 MBytes 41.9 Mbits/sec
[ 3] 7.0- 8.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 8.0- 9.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 9.0-10.0 sec  5.12 MBytes 43.0 Mbits/sec
[ 3] 0.0-10.0 sec  50.9 MBytes 42.5 Mbits/sec
```

UDP Server:

```
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 3] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 36565
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams
[ 3] 0.0- 1.0 sec    6.80 MBytes 57.1 Mbits/sec 0.043 ms    0/ 4852 (0%)
[ 3] 1.0- 2.0 sec    6.48 MBytes 54.4 Mbits/sec 0.050 ms    0/ 4622 (0%)
[ 3] 2.0- 3.0 sec    6.60 MBytes 55.3 Mbits/sec 0.083 ms    0/ 4706 (0%)
[ 3] 3.0- 4.0 sec    6.62 MBytes 55.6 Mbits/sec 0.049 ms    0/ 4724 (0%)
[ 3] 4.0- 5.0 sec    6.87 MBytes 57.6 Mbits/sec 0.043 ms    0/ 4900 (0%)
[ 3] 5.0- 6.0 sec    6.50 MBytes 54.6 Mbits/sec 0.043 ms    0/ 4640 (0%)
[ 3] 6.0- 7.0 sec    6.86 MBytes 57.6 Mbits/sec 0.066 ms    0/ 4896 (0%)
[ 3] 7.0- 8.0 sec    6.62 MBytes 55.5 Mbits/sec 0.060 ms   108/ 4831 (2.2%)
[ 3] 8.0- 9.0 sec    6.83 MBytes 57.3 Mbits/sec 0.085 ms   223/ 5095 (4.4%)
[ 3] 9.0-10.0 sec    6.84 MBytes 57.4 Mbits/sec 0.053 ms   233/ 5113 (4.6%)
[ 3] 0.0-10.5 sec   70.6 MBytes 56.3 Mbits/sec 0.063 ms  672/51020 (1.3%)
```

UDP Client:

```
-----
Client connecting to 192.168.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 3] local 192.168.0.126 port 50346 connected with 192.168.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec    6.32 MBytes 53.0 Mbits/sec
[ 3] 1.0- 2.0 sec    6.23 MBytes 52.2 Mbits/sec
[ 3] 2.0- 3.0 sec    6.28 MBytes 52.7 Mbits/sec
[ 3] 3.0- 4.0 sec    6.34 MBytes 53.2 Mbits/sec
[ 3] 4.0- 5.0 sec    6.30 MBytes 52.9 Mbits/sec
[ 3] 5.0- 6.0 sec    6.37 MBytes 53.4 Mbits/sec
[ 3] 6.0- 7.0 sec    6.01 MBytes 50.4 Mbits/sec
[ 3] 7.0- 8.0 sec    6.00 MBytes 50.3 Mbits/sec
[ 3] 8.0- 9.0 sec    5.94 MBytes 49.9 Mbits/sec
[ 3] 9.0-10.0 sec    6.26 MBytes 52.5 Mbits/sec
[ 3] 0.0-10.0 sec    62.1 MBytes 52.1 Mbits/sec
[ 3] Sent 44263 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec    62.1 MBytes 51.8 Mbits/sec 0.186 ms    0/44262 (0%)
[ 3] 0.0-10.0 sec    1 datagrams received out-of-order
```

6 Appendix A: SDIO Connectivity Header Pin Description

The following table describes the pins of SDIO header.

SDIO/SPI HEADERS

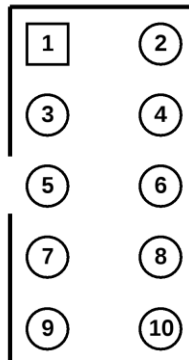


Figure 5: Header PIN Orientation

6.1 SDIO Header Pin Description

The following table describes the pins of SDIO header.

Pin #	Pin Name	Direction	Description
1	SDIO_DATA3	Input/Output	Data3 of SDIO interface.
2	SDIO_CMD	Input/Output	SDIO Mode: SDIO interface command signal.
3	GND	-	Ground
4	VDD	-	Supply voltage.
5	SDIO_CLK	Input	This signal is SDIO clock.
6	GND	-	Ground
7	SDIO_DATA0	Input/Output	Data0 of SDIO interface.
8	SDIO_DATA1	Input/Output	Data1 of SDIO interface.
9	SDIO_DATA2	Input/Output	Data2 of SDIO interface.
10	NC	-	No Connect

Table 2: SDIO Header Pins

Reset - When the EVB is powered through the USB on "Power" port or through the power on any interface (UART, USB, USB-CDC, SDIO/SPI) than it gets the Power on Reset.

To control the reset there are two methods:

1. Reset Button on baseboard.
2. Host can control the Reset by controlling the pin #2 (RST_PS) on header J9 via GPIO.

Note: Signal Integrity Guidelines for SPI/SDIO interface: Glitches in the SPI/SDIO clock can potentially take the SPI/SDIO interface out of synchronization. The quality and integrity of the clock line needs to be maintained. In case a cable is used for board to board connection, the following steps are recommended (please note that this is not an exhaustive list of guidelines and depending on individual cases additional steps may be needed.):

1. Minimize the length of the SPI/SDIO bus cable to as small as possible, preferably to within an inch or two.
2. Increase the number of ground connections between the EVB and the Host processor PCB.

7 Appendix B: Using the Bluetoothctl Application

1. Once Bluetooth modules have been installed as mentioned in [Installation Steps](#), check if the Bluetooth "hci" interface has been created or not with the command below :

```
# hciconfig -a
```

2. Sample output to the above command is as follows:

```
hci0:  Type: BR/EDR  Bus: USB
      BD Address: 88:DA:1A:00:00:C2  ACL MTU: 1021:3  SCO MTU: 64:3
      UP RUNNING
      RX bytes:1006 acl:0 sco:0 events:55 errors:0
      TX bytes:0 acl:0 sco:0 commands:55 errors:0
      Features: 0xbf 0xfe 0x0d 0xfe 0xdb 0xff 0x5b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH SNIFF
      Link mode: SLAVE ACCEPT
      Name: 'lapt64'
      class: 0x0c010c
      service Classes: Computer, Laptop
      HCI Version: (0x9) Revision: 0x0
      Manufacturer: internal use (65535)
```

3. Use the following commands to pair / trust / connect to another Bluetooth device.
 - a. bluetoothctl –a (upon executing this command user will enter into a Bluetooth mode console, and user needs to enter the following commands)
 - b. power on
 - c. agent on
 - d. scan on (when this command is entered, you will see the scan list of other Bluetooth devices on the console)
 - e. scan off
 - f. pair <Target BD address> (it will ask for a key confirmation)
 - g. trust <Target BD address>
 - h. Connect <Target <BD address>
 - i. After successful connection, user can see the connection status on the command line as well as using the command "hcitool –l hciX con", here X can be 0,1,2,3..
4. Now, user can do data transfer by transferring a file to the target device and run some tests like I2test, I2ping etc.

8 Appendix C: Cross-Compiling Open Source Driver for Target Platforms

Please find the below steps to cross compile the Redpine driver for target platforms:

1. Go to source directory. Open Makefile and edit these changes in the Makefile.
2. Comment the actual KERNELDIR using '#' before the KERNELDIR. Add the Linux source path as the default kernel directory.

```
# KERNELDIR := /lib/modules/$(shell uname -r)/build  
# KERNELDIR:=/path/to/target/platform's Linux Kernel/
```

Example:

```
KERNELDIR:=/work/WandBoardBuild/wandboard-sdk-20140519/linux-3.0.101-4.1.0-wand/
```

3. Finally go to source directory and "make" using tool chain.
"make ARCH=arm CROSS_COMPILE= /path/to/toolchain/bin/arm-none-linux-gnueabi-"
Example :

```
# make ARCH=arm CROSS_COMPILE=/work/WandBoardBuild/arm-2009q1/bin/arm-none-linux-gnueabi-
```

4. After successful compilation copy the **.ko files** to target platform.

9 Appendix D: Configuration of Kernel SDIO Stack and Bluetooth Stack

To ensure that the OSD software works on kernel, some configuration changes might be needed. These are explained in this section. Super user permissions are needed to make these changes.

9.1 SDIO Stack Options

If SDIO is the interface to the Host processor, it has to be ensured that the SDIO stack related modules are compiled in the kernel. If the SDIO stack modules are not present, follow the steps below in order to enable SDIO support in the kernel.

1. Navigate to the Linux kernel source folder. This is usually in `/usr/src/kernels/Linux-<kernel-version>`
2. Execute the **'make menuconfig'** command in order to open the Kernel Configuration menu.
3. Scroll down to the **"Device Drivers --->"** option and hit Enter.
4. In the new menu, scroll down to the **"MMC/SD/SDIO card support --->"** option and press **'M'** to modularize the **"MMC/SD/SDIO card support"** feature and hit Enter.
5. In the new menu, press **'M'** to modularize the following options:
 - o MMC block device driver
 - o Secure Digital Host Controller Interface support
 - o SDHCI support on PCI bus
6. Hit the Tab key to select Exit and hit Enter. Repeat this till you are asked whether you want to save the configuration.
7. Select "Yes" and hit Enter. If the above options are already selected, the menuconfig screen will exit immediately.

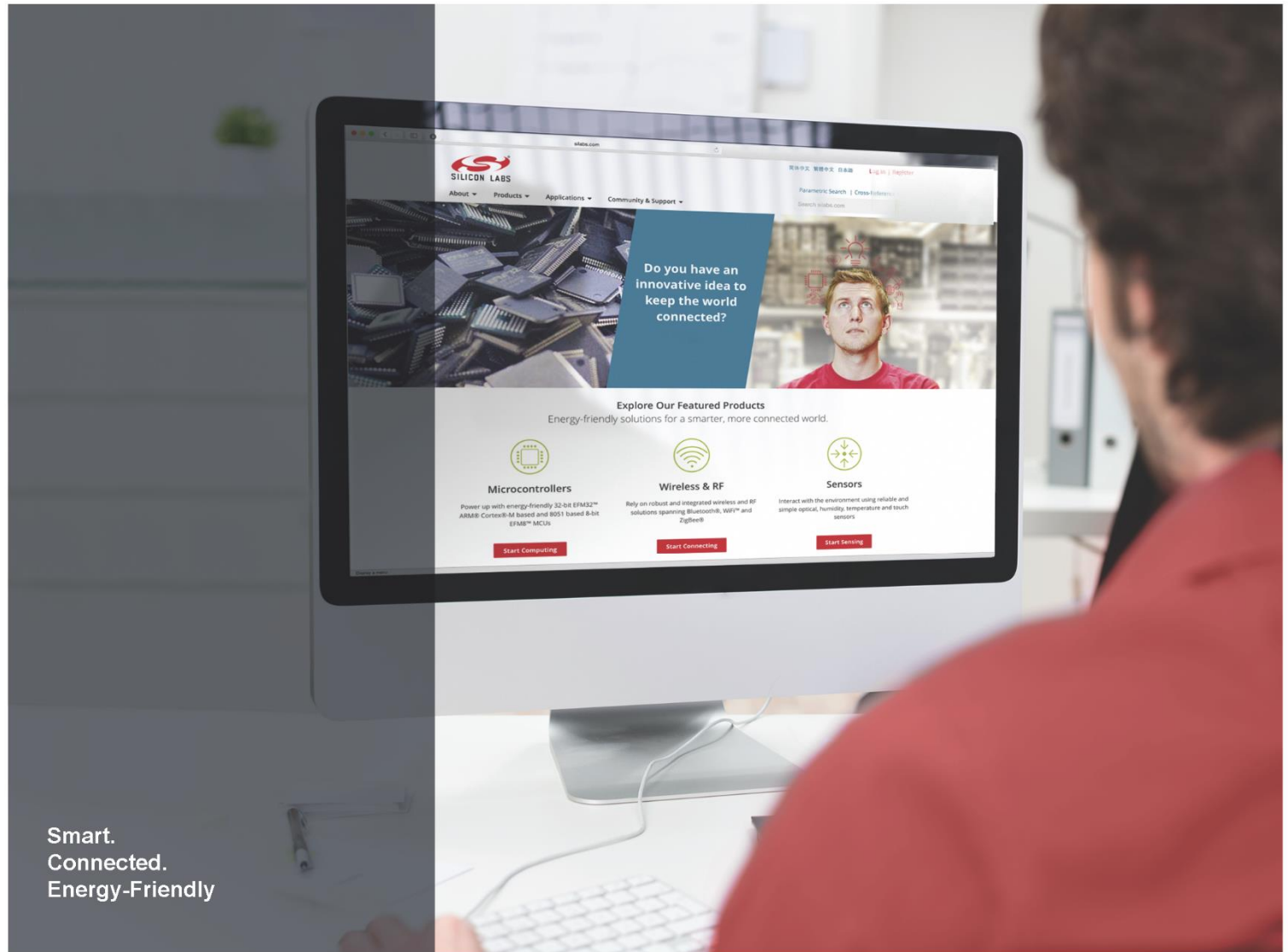
9.2 Bluetooth Stack Options

If Bluetooth is required, it has to be ensured that the Bluetooth modules are compiled in the kernel. If the Bluetooth modules are not present, follow the steps below to enable Bluetooth support in the kernel.

- Navigate to the Linux kernel source folder. This is usually in `/usr/src/kernels/linux-<kernel-version>`
- Execute the **'make menuconfig'** command in order to open the Kernel Configuration menu.
- Scroll down to **"Networking support --->"** and hit Enter.
- In the new menu, scroll down to the **"Bluetooth subsystem support --->"** option and press **'M'** to modularize the **"Bluetooth subsystem support"** feature and hit Enter.
- In the new menu, press **'M'** to modularize the following options:
 - a. RFCOMM Protocol support (enable the "RFCOMM TTY support" feature under this).
 - b. BNEP Protocol support (enable the "Multicast filter support" and "Broadcast filter support" features under this).
 - c. CMTP Protocol support
 - d. HIDP Protocol support
- Hit the Tab key to select Exit and hit Enter. Repeat this till you are asked whether you want to save the configuration.
- Select "Yes" and hit Enter. If the above options are already selected, the menuconfig screen will exit immediately.

10 Open Source Driver QSG Revision Report

Revision No.	Version No.	Date	Changes
1	1.0	September 2020	Preliminary version



Smart.
Connected.
Energy-Friendly



Products
www.silabs.com/products



Quality
www.silabs.com/quality



Support and Community
community.silabs.com

Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



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

<http://www.silabs.com>