

RS9116 n-Link EVK User's Guide

Version 1.1

14 May 2018

Redpine Signals, Inc.

2107 North First Street, Suite #540, San Jose, California 95131,
United States of America.

Phone: +1-408-748-3385, Fax: +1-408-705-2019

Email: info@redpinesignals.com

Website: www.redpinesignals.com

Disclaimer

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.

Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Copyright © 2018 Redpine Signals, Inc. All rights reserved.

Table of Contents

1	RS9116 EVK Introduction.....	5
2	Getting Started with RS9116 n-Link EVK.....	6
3	Restoring Wand board solo configuration.....	8
3.1	Linux	8
3.2	Windows	8
4	Evaluation of RS9116 EVK by using master script.....	9
5	Compilation of OneBox driver on Linux x86 machine.....	13
6	RS9116 EVK Schematics and BOM	18
7	RS9116 EVK User's Guide Revision History	19

About this Document

This document is a Preliminary version of RS9116N EVK user's guide provided to customers under a Non-Disclosure Agreement (NDA).

1 RS9116 EVK Introduction

This document describes how to bring up the IMX6 wandboard with Redpine's OneBox™ driver that enables RS9116 n-Link module to function in Wi-Fi/BTLE/BT Classic/Zigbee modes.

Redpine's OneBox™ driver can also run on Linux X86 machines. The procedure to compile and run the same is discussed in this document.

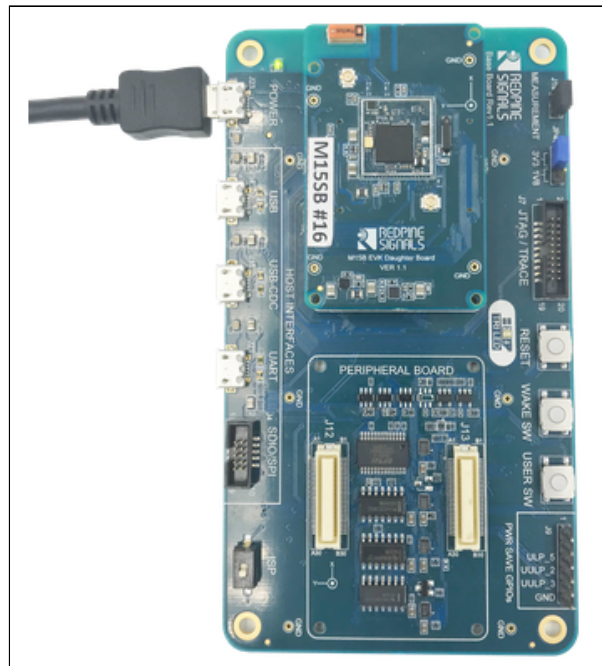


Figure 1: RS9116 EVK Board

2 Getting Started with RS9116 n-Link EVK

Hardware

The items that are required to setup for evaluation are outlined below:

1. RS9116N EVK board
2. A dipole antenna with U.FL connector
3. A USB cable or SDIO cable
4. NXP Wandboard Solo (i.MX6) processor board
5. An Ethernet cable
6. A Wi-Fi access point
7. 5V DC barrel type adapter

Software

1. Minicom(linux X86 machine)

Setup

1. Power up RS9116 EVK board by plugging USB connector to POWER labeled micro USB slot on the board.
2. Connect the Wandboard Solo to RS9116N EVK board using desired interface either using USB or SDIO cable. Power up the board using 5V adapter. Connect DB9 connector to the serial port of PC.

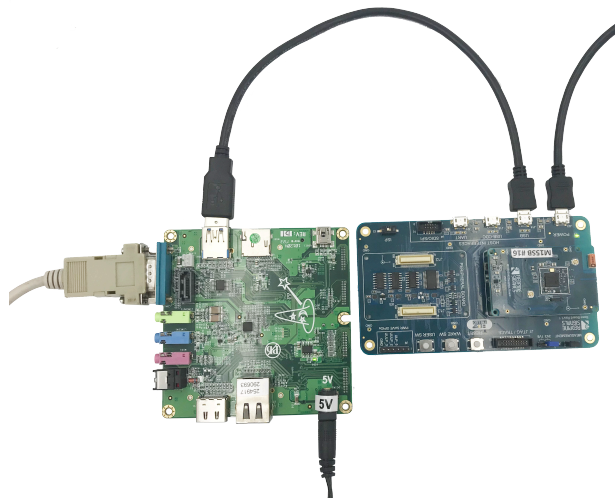


Figure2: Connection of RS9116 EVK to Wandboard Solo using USB Interface

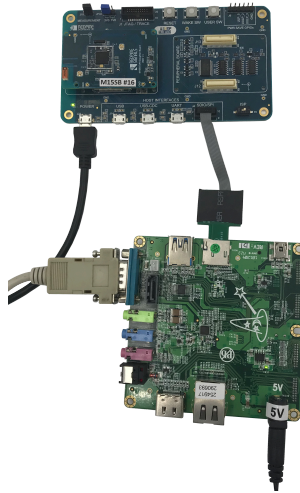


Figure 3: Connection of RS9116 EVK to Wandboard Solo using SDIO Interface

3. The Wandboard Solo is shipped with the Linux operating system and Redpine OneBox drivers. If you have modified the Wandboard Solo configuration, you will need to reconfigure again. Refer [Restoring Wand board solo configuration](#) section for more details.
4. You may run on stand alone Linux X86 PC/Laptop with connections as shown below using USB interface. Refer Compilation of OneBox driver on Linux x86 machine section for more details.

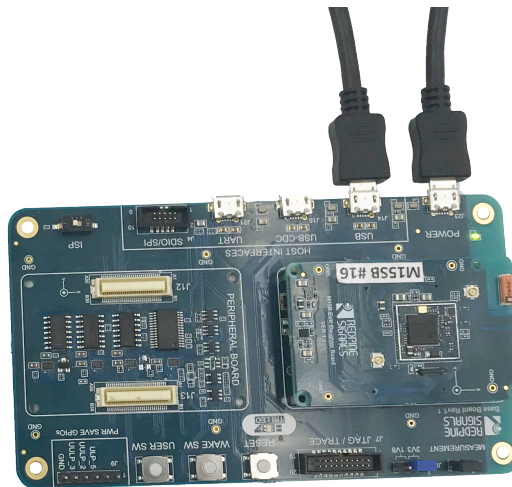


Figure 4: Connection of RS9116 EVK to PC/Laptop using USB Interface

3 Restoring Wand board solo configuration

3.1 Linux

1. Download the Redpine demo image from Redpine Document portal at: TBD
2. Format the SD card

For these instructions we are assuming, DISK=/dev/sdb, lsblk is very useful for determining the device ID. Enter the following command:

```
sudo dd if=~/.redpinedemoimage.img of=<device id>
```

Wait for few minutes without disturbing the setup.

3.2 Windows

One of the best tools to backup and restore SD cards is **Win32 Disk Imager**. Download it from [SourceForge](#) and install the same. Before starting the software, insert your SD card (min. of 8GB of storage) into the SD card reader on your PC. Often, this will open the file explorer showing you the contents of part of your SD card.

Once the Win32 Disk Imager is open, use the blue folder icon to choose the location and the name of the backup you want to take, and then choose the drive letter for your SD card. Please provide the path for the .img file on your hard drive provided by the Redpine.

Click the **Write** button to write it to the SD card.

4 Evaluation of RS9116 EVK by using master script

The Evaluation setup uses a Redpine supplied demo script to enable variety of operating and test modes commonly used in evaluation. The Wand Solo board supplied with the kit is preprogrammed with a Linux operating system and Redpine OneBox™ driver package.

Linux x86 (Standalone): After successful compilation of driver, the user has to navigate to **/host/release/Vx_SmartEvk** folder, where they can find smartEVK script for loading and unloading the driver.

IMX6 Wand Solo: Open minicom or any serial terminal with 115200 bps as selected baud rate. Navigate to **/host/release/Vx_SmartEvk** folder.

Rest of the procedure is common for both Embedded and linux standalone platforms.

In order to work with SmartEVK master scripts, follow the steps mentioned below:

1. Configure the settings in **config.py** file as per the requirement. Please find example of configuration name of SSID and security for the STA and API.
2. If user wish to run NL80211 and Hostapd mode, they need to stop the network manager and disable rkill by using following commands:
 - a. **"service NetworkManager stop"**
 - b. **"rfkill unblock all"**
3. Run the script named **"rsi_master.py"** and select the user input as per the selection of specific protocol.
4. Please follow the steps below in order to create a connection between STA and AP in OPEN mode.

BSD Example

```
[root@cpu312 V3_SmartEvk]# python rsi_master.py
```

Mode:

0. Quit
1. Standalone mode
2. coex mode

Select the option : 1

Standalone mode:

0. Quit
1. STA
2. AP
3. BT
4. BLE

Select the option : 1

Driver interface:

0. Quit
1. nl80211
2. bsd

Select the option : 2

Config mode:

- 0. Quit
- 1. open_mode
- 2. ccmp_mode

Select the option : 1

Once the user is done with the selections, following prints will be displayed on the host:

Please wait;driver will remove and load again

Driver has removed successfully!

Please wait driver is loading

coex mode has set successfully!

insert_driver

Driver has inserted successfully!

Protocol has enabled successfully!

vap has created successfully for STA mode!

sta_config file has updated for STA mode!

supplicant is running for to configure the STA mode!

Checking wpa_supplicant status!

Collecting IP from DHCP server!

192.168.1.21

Example for evaluation of NL80211 driver interface

[root@cpu312 V7_SmartEVK]# python rsi_master.py

Mode:

- 0. Quit
- 1. Standalone mode
- 2. coex mode

Select an option : 2

Coex mode:

- 0. Quit
- 1. STA+AP
- 2. STA+BT
- 3. AP+BT
- 4. STA+BLE
- 5. BT+BLE
- 6. STA+BT+BLE

7. AP+BT+BLE

Select an option : 1

Driver interface:

- 0. Quit
- 1. nl80211
- 2. bsd

Select an option : 1

STA Config mode:

- 0. Quit
- 1. open_mode
- 2. ccmp_mode

Select an option : 1

AP Config mode:

- 0. Quit
- 1. host_apd_open_mode
- 2. host_apd_ccmp_mode

Select an option : 1

Please wait;driver will remove and load again

Driver has removed successfully!

Please wait driver is loading

coex mode has set successfully!

insert_driver

Driver has inserted successfully!

Protocol has enabled successfully!

vap has cerated successfully for STA mode!

sta_config file has updated for STA mode!

wlp0s26u1u4

supplicant is running for to configure the STA mode!

Checking wpa_supplicant status!

Collecting IP from DHCP server!

192.168.43.17

create vap for AP

ap_config file has updated for Host APD mode!

hostapd is running for to configure the AP mode!

DHCP server has started

```
[root@cpu312 V7_SmartEVK]# ps -a
PID TTY      TIME CMD
1966 pts/0    00:00:00 su
1969 pts/0    00:00:01 bash
5411 pts/0    00:00:00 wpa_supplicant
5550 pts/0    00:00:00 hostapd
5573 pts/0    00:00:00 ps
[root@cpu312 V7_SmartEVK]# iwconfig
rpine0  no wireless extensions.
lo      no wireless extensions.
wlp0s26u1u4 IEEE 802.11bgn ESSID:"sample" Nickname:""
        Mode:Managed Frequency:2.417 GHz Access Point: 66:DB:43:2F:1C:AD
        Bit Rate:65 Mb/s Tx-Power=7 dBm Sensitivity=1/0
        RTS thr:off Fragment thr:off
        Encryption key:off
        Power Management:off
        Link Quality=80/80 Signal level=-24 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
em1     no wireless extensions.
wifi1   IEEE 802.11bgn ESSID:"REDPINE_AP_CCMP" Nickname:""
        Mode:Master Frequency:2.417 GHz Access Point: 00:23:A7:DD:F3:AD
        Bit Rate:1 Mb/s Tx-Power=20 dBm Sensitivity=1/0
        RTS thr:off Fragment thr:off
        Encryption key:off
        Power Management:off
        Link Quality=80/80 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
[root@cpu312 V7_SmartEVK]#z
```

Similarly,

For other protocols, follow the same steps as explained in above example cases.

Run the python script named **clean_up.py** for deleting python compiled files(*.pyc).

Also, run python script named **rsi_remove_driver.py** for unloading the driver.

5 Compilation of OneBox driver on Linux x86 machine

The OneBox-Mobile Software is delivered as a tarball with a filename in the format:

RS9116.NXX.NL.GEN.LNX.x.y.z.tgz, where the naming convention is as follows:

NXX – defines whether the package supports only Wi-Fi (N00) or Bluetooth Classic/Low Energy along with Wi-Fi (NB0) or ZigBee along with Wi-Fi (N0Z) or Bluetooth Classic/Low Energy and ZigBee along with Wi-Fi (NBZ).

x.y.z – identifies the software package.

Redpine driver comes in proprietary and open source form. The Linux driver package contains the following files/folders:

- Readme.txt
- Releasenotes.txt
- Documents
- Binary_files (optional)
- source (optional)

This section explains the steps to be followed in order to compile the OneBox-Mobile Linux software for different platforms. The steps are outlined below:

1. Save the required configuration of the driver using the **menuconfig utility**.

Following are the options available in menuconfig:

- Host Interface: SDIO or USB.
- Operating system: Linux or Android
- NL80211 support
- Hostapd Support
- WIFI
- BLUETOOTH
- ZIGBEE

2. To open menuconfig utility, enter the below command:

```
make menuconfig
```

The following images depicts the menuconfig utility options.

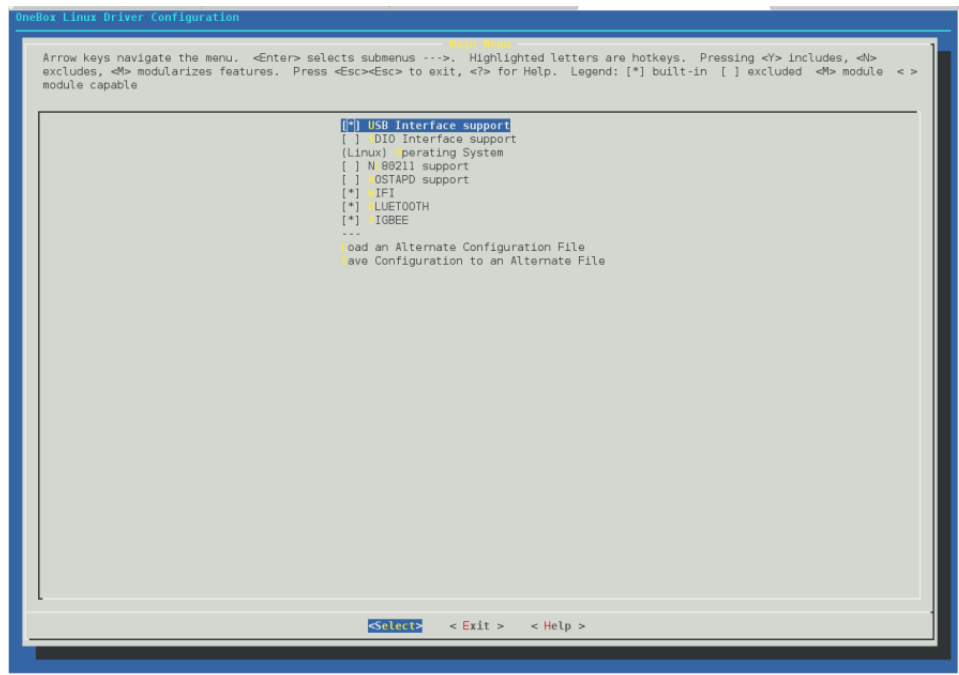


Figure 5: Main Page of menuconfig

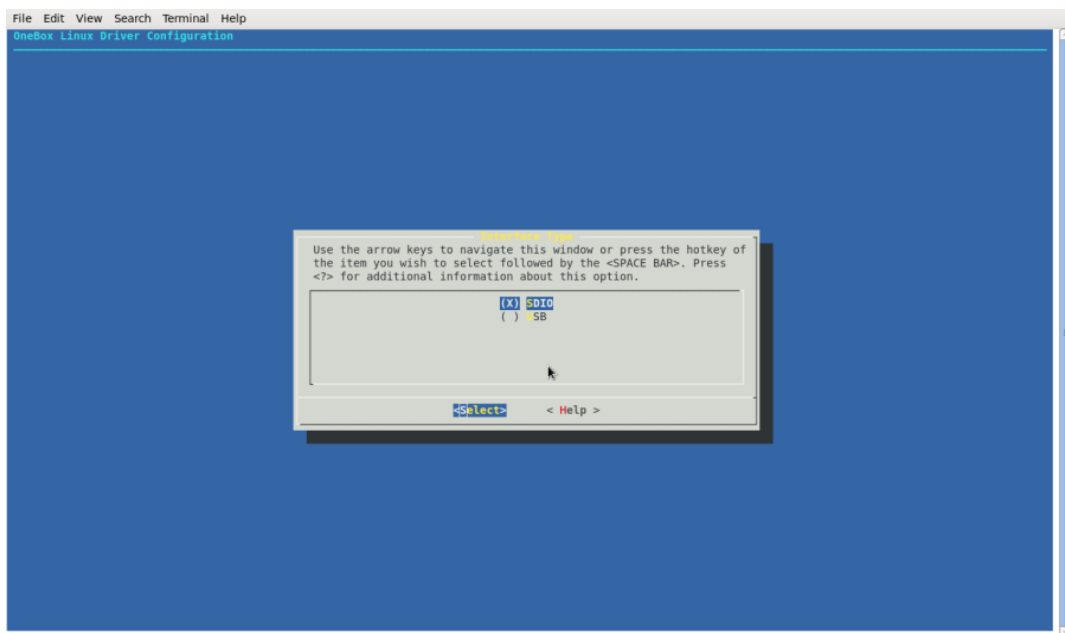


Figure 6: Selecting Host Interface

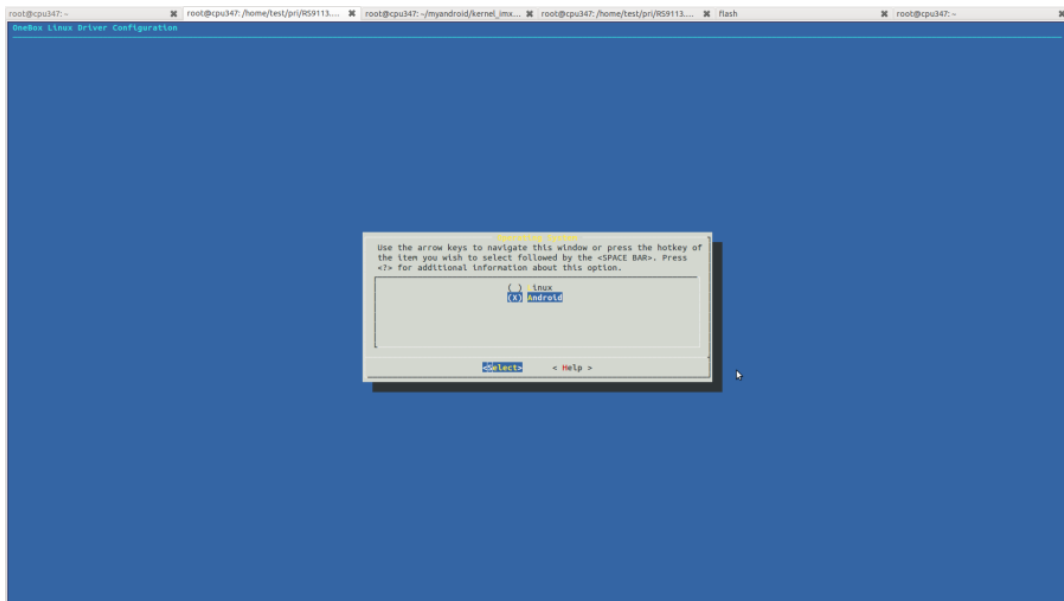


Figure 7: Selecting Operating System

By default, the driver package includes "BSD" support. In case the user needs "NL80211" support for Access point and Station modes, select the **menuconfig** accordingly. For NL80211, the "**Hostapd**" application is used as a configuration utility.

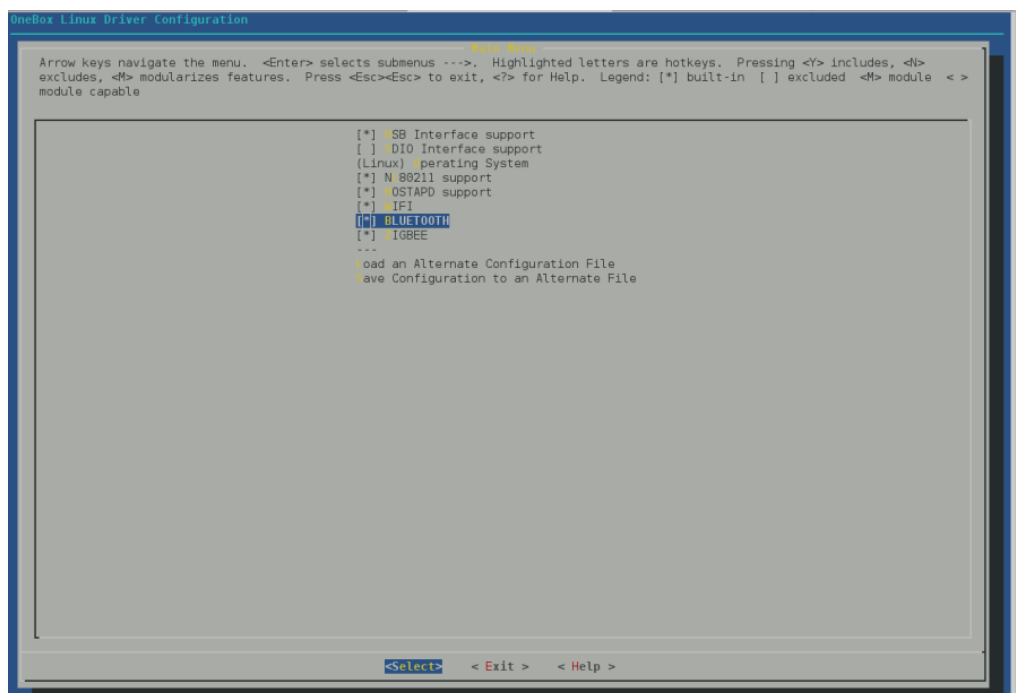


Figure 8: Selection of NL80211 and Hostapd Support



If NL80211 support is enabled in the driver, make sure the following modules are loaded in the kernel prior running the driver in order to avoid module dependencies.

- modprobe cfg80211
- modprobe bluetooth

By default the configuration is enabled with Wi-Fi, Bluetooth and ZigBee. If the user wants to compile the driver for a particular protocol, he can disable the unwanted protocols in **Menuconfig utility**. In case of coex mode, the Wi-Fi must always be enabled in conjunction with BT / ZigBee protocols, even if Wi-Fi will not be used.

For example, if the user wants to compile the driver only for Bluetooth, the Wi-Fi and Bluetooth mode must be enabled. Refer to the following images of Menuconfig utility for more information:

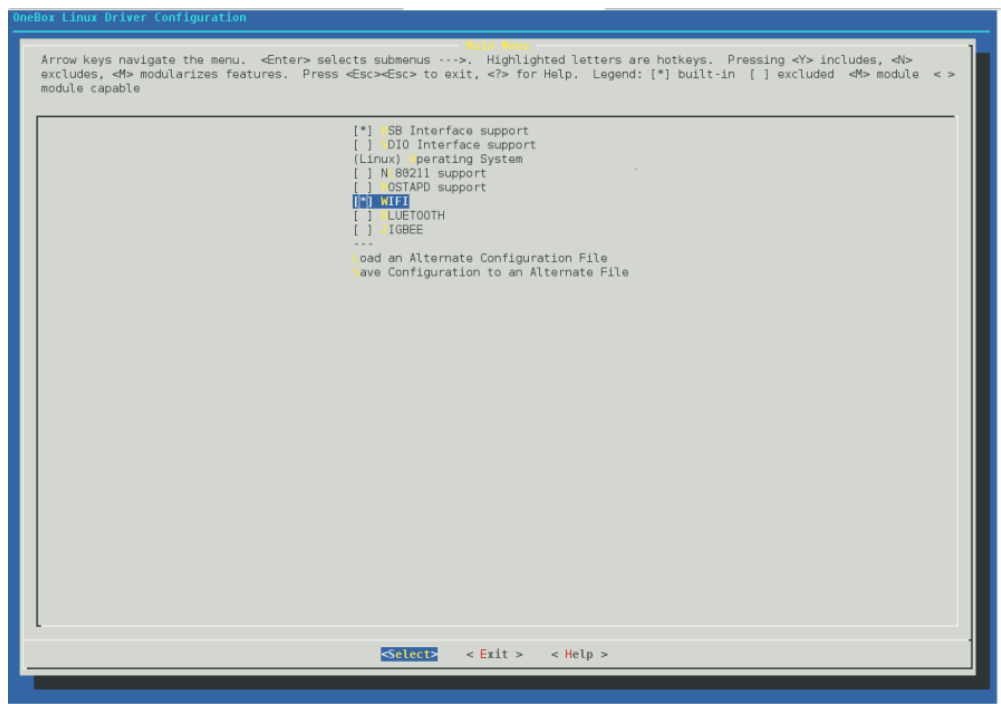


Figure 9: Selection of WIFI Only Mode

3. After selecting the configuration, exit the menuconfig and save the configuration. Refer to the below image.



Figure 10: Save the changes before exiting

4. Now to compile the driver, enter the below command:

```
make
```

The code is compiled and the binaries are generated in the **source/host/release** folder. For embedded platforms, modify the path assigned to the **"DEF_KERNEL_DIR"** variable in the Makefile:

```
cd RS9116.NXX.NL.GEN.LNX.x.y.z/source/host
```

```
vim Makefile
```

The **DEF_KERNEL_DIR** variable has to be assigned along with the compiled kernel path. For an x86 based Linux platform, the path is usually **"/lib/modules/<kernel_version>/build"** and this is the path assigned in the **Makefile** provided in the package.

Example:

```
DEF_KERNEL_DIR:= /lib/modules/3.4.66/build
```

Next, use the **"make"** command to start compiling the driver. 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 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-
```



Before installing the Onebox RS9116 Driver modules, make sure that the RSI opensource modules are uninstalled. This has been taken care in the **onebox_insert.sh** script.

6 RS9116 EVK Schematics and BOM

Refer to Schematics and BoM files in Redpine document portal at the below paths.

Name	Location on Redpine Document Portal
Schematics	/RS9116N n-Link/EVK/
BoM	/RS9116N n-Link/EVK/

7 RS9116 EVK User's Guide Revision History

Revision No.	Version No.	Date	Changes
1	v1.0	March	Preliminary version