NEW RADIO COMMUNICATION FOR NEW ERA

NRC7292 Evaluation Kit User Guide

(Host Mode)

Ultra-low power & Long-range Wi-Fi

Ver 2.5
Jun. 12, 2023

Newracom, Inc.

NRC7292 Evaluation Kit User Guide (Host Mode)
Ultra-low power & Long-range Wi-Fi

Office

NEWRACOM, Inc.
505 Technology Drive, Irvine, CA 92618 USA
http://www.NEWRACOM.com

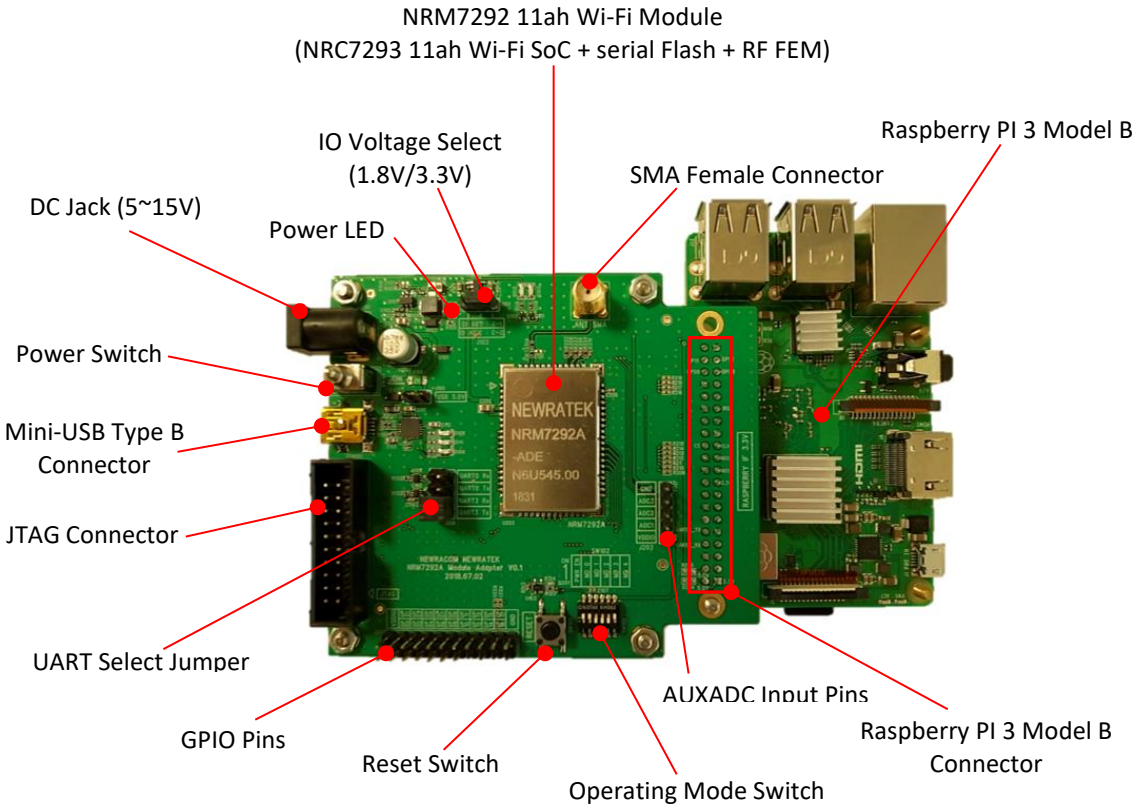
Contents

List of Tables

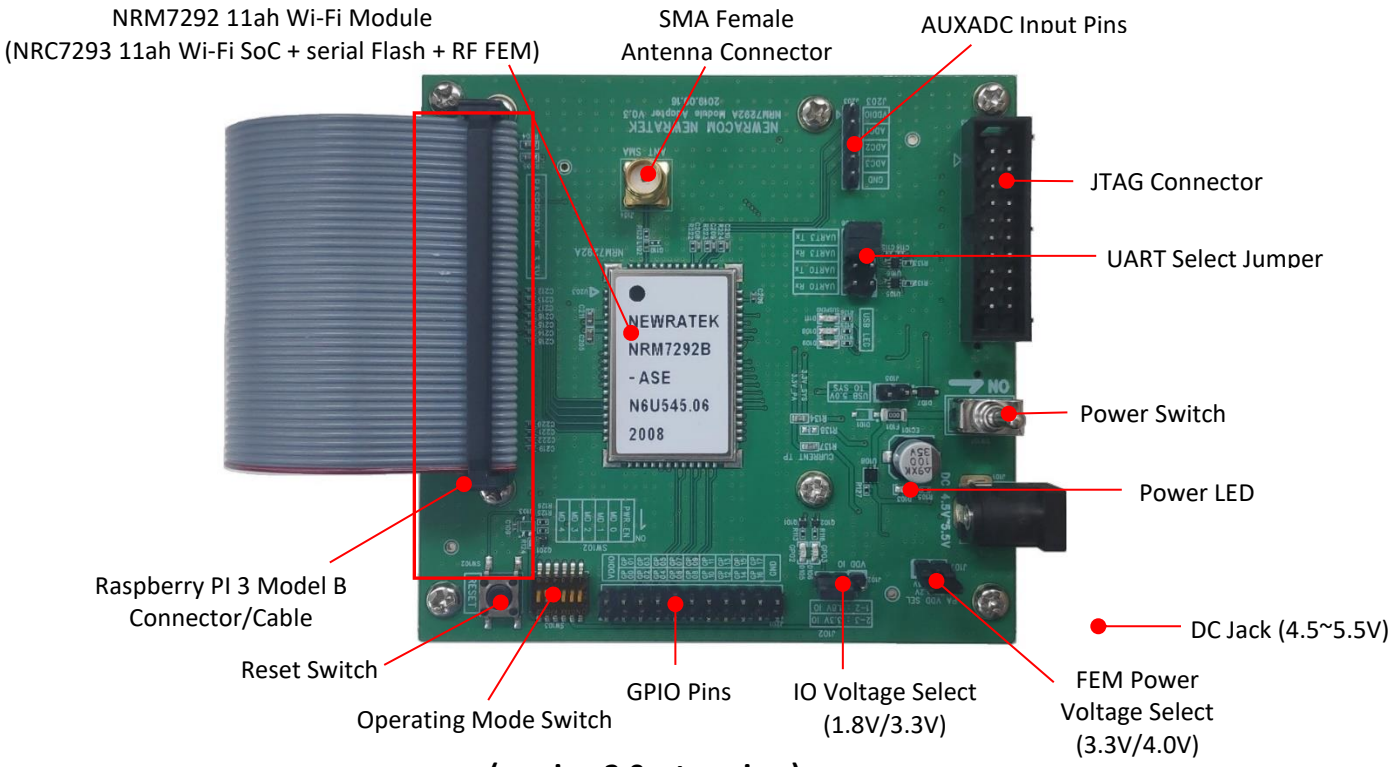
List of Figures

1 Overview

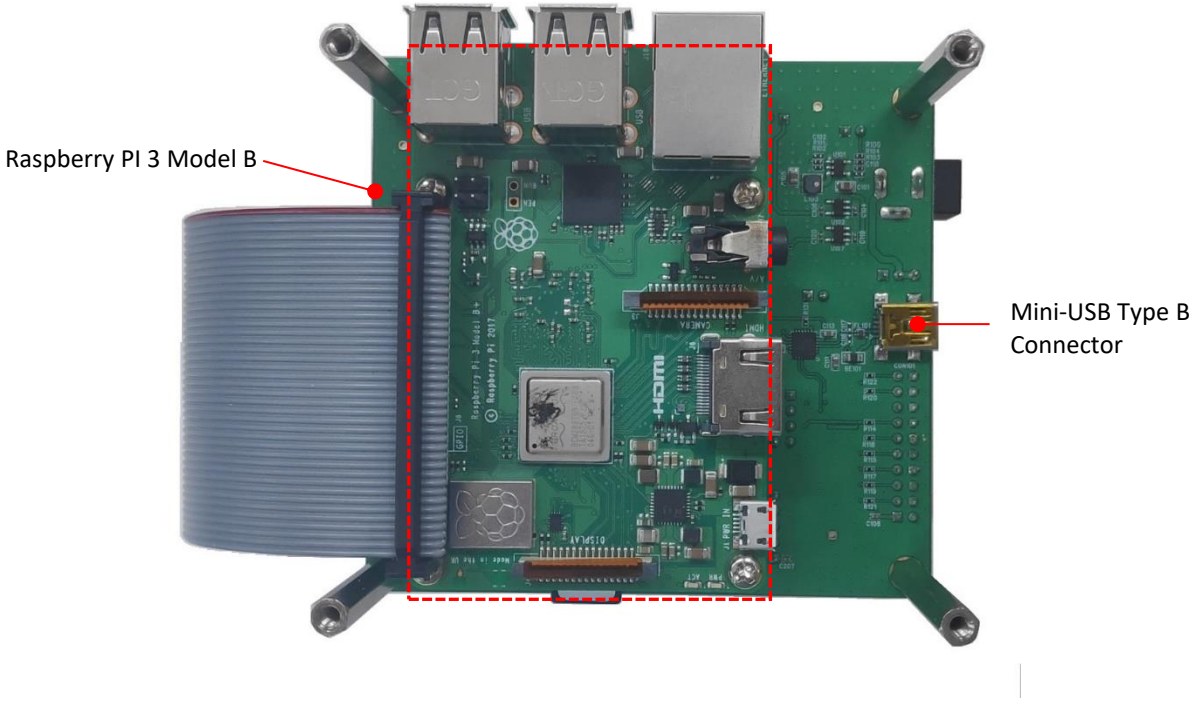
This document introduces NEWRACOM's NRC7292 Evaluation kit (EVK). The EVK kit is used to evaluate the performance of the NRC7292 Wi-Fi module containing NEWRACOM's IEEE 802.11ah Wi-Fi System on Chip (SoC) solution.



(version 1.0)



(version 2.0 – top view)



(version 2.0 – bottom view)

Figure 1.1 NRC7292 evaluation board

Figure 1.2 Host mode configuration

1.1 HW list

As shown in Figure 1.3, NRC7292 EVK consists of three boards.

1.1.1 NRC7292 module board

NRC7292 module contains IEEE 802.11ah Wi-Fi SoC solution. It also includes a RF front end module (FEM) to increase transmission power up to +23 dBm. Onboard serial flash memory can be used for over-the-air (OTA) software development and with 32KB cache in the NRC7292 supports the execution in place (XIP) feature.

1.1.2 NRC7292 adapter board

NRC7292 adapter board mainly offers communication interfaces to sensors or an external host. It also supplies the main power of NRC7292 Wi-Fi module.

1.1.3 Host board

NRC7292 module can be used either in standalone or slave to host processor via serial peripheral interface (SPI) or universal asynchronous receive transmitter (UART). Raspberry PI 3 can be one of the hosts used for normal operation, evaluation, and testing. When used in standalone, Raspberry PI3 board is not needed because NRC7292 operates without an additional host processor.

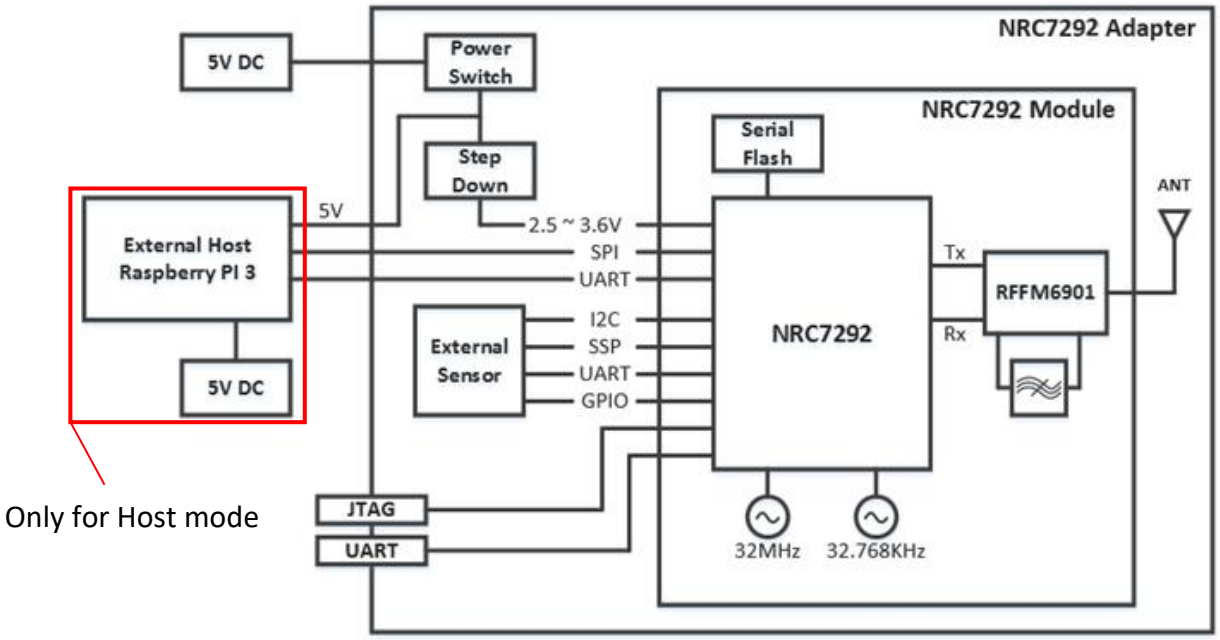
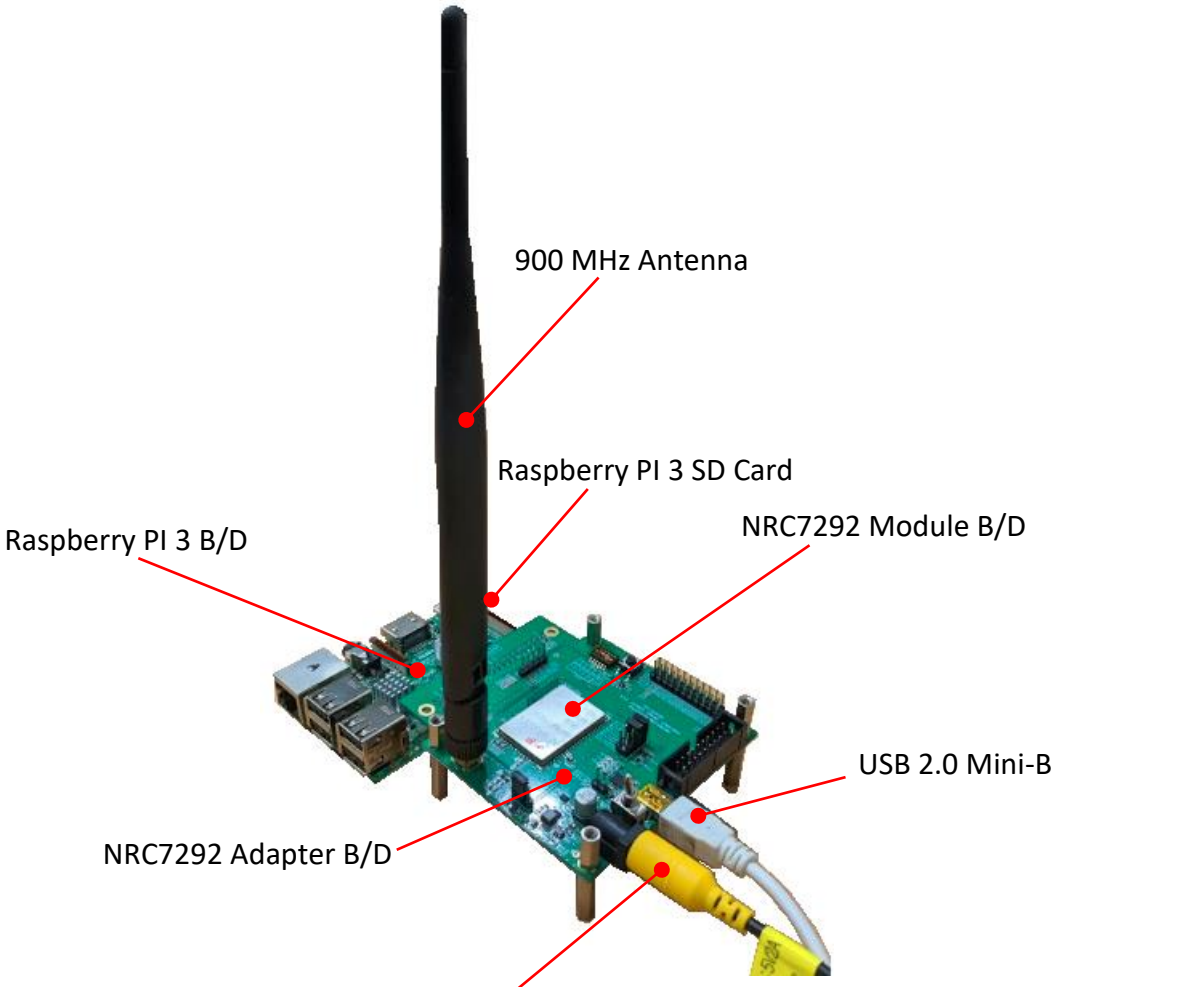


Figure 1.3 Block diagram of NRC7292 evaluation board

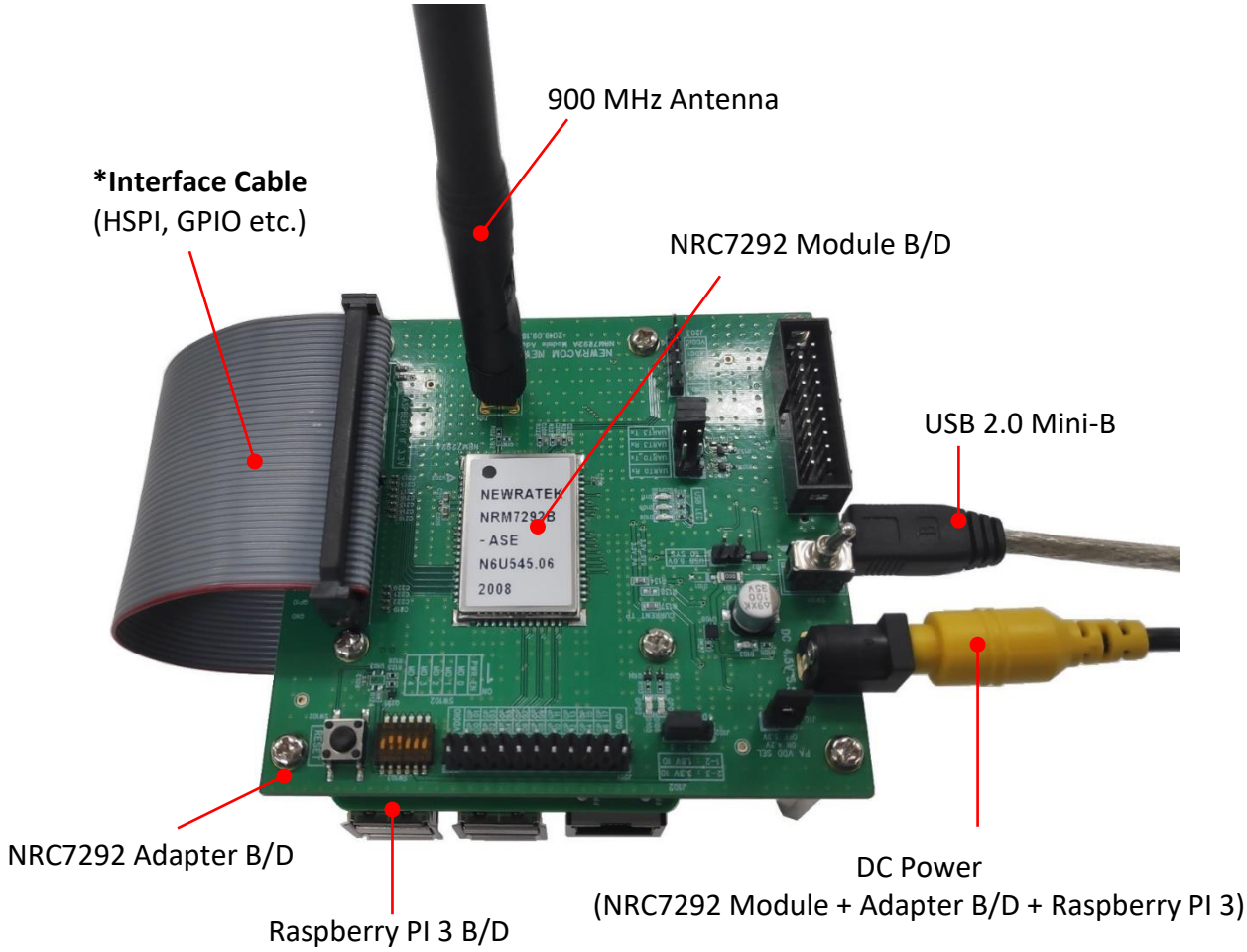
1.2 Kit list

NRC7292 EVK includes:

- NRC7292 Wi-Fi module board
- NRC7292 Adapter board
- Raspberry PI 3 board
- SD card with Linux OS, NRC7292 firmware, Wi-Fi driver, and scripts
- DC 5V (2A) power for EVK (Raspberry PI 3 + Wi-Fi module + Adapter board)
- 900 MHz Antenna



(version 1.0)



(version 2.0)

Figure 1.4 NRC7292 hardware set

***NOTE:**

The interface cable is removable to support the USB interface with the FTDI FT232H USB-SPI bridge cable. For more information, refer to the **AN-7292-008-FT232H_USB_SPI.pdf** file.

2 NRC7292 EVK manipulation

This chapter describes how to manipulate NRC7292 EVK directly or remotely. To directly control NRC7292 EVK, I/O devices such as: keyboard, mouse, monitor, and HDMI cable are needed. Otherwise, users can use terminal programs such as: Tera Term, Mobaxterm, VNC, and etc. to control NRC7292 EVK remotely.

2.1 Direct manipulation

Using I/O devices is the simplest way to control NRC7292 EVK. It includes Raspberry PI3 which supports various I/O, especially useful is the HDMI connection. User can simply display Raspberry Pi (RPi) to a monitor by HDMI cable. Other USB-type keyboard and mouse can be utilized as input devices. However, if these I/O devices are not available, users have to additional options to manipulate NRC7292 EVK.

2.2 Remote manipulation

Raspberry PI3 has an Ethernet port, so users can remotely connect to RPi with terminal programs. There are two ways to obtain an IP address on RPi Ethernet, one way is static IP and the other is dynamic IP (DHCP). The initial (default) setting is set to obtain an IP address from the external DHCP server. Furthermore, the RPi can be operated as a DHCP server by changing the configuration. When enabled, the DHCP server will assign a static IP address (192.168.100.1) to RPi Ethernet and run the DHCP server. The PC will then obtain an IP address from RPi after connecting by Ethernet cable to the RPi. After the PC receives the IP address from RPi, then users can remotely access RPi (192.168.100.1) with terminal programs. Figure 2.1 and Figure 2.2 show Mobaxterm and VNC displays of PC receiving the IP address (192.168.100.10) after connecting to RPi.

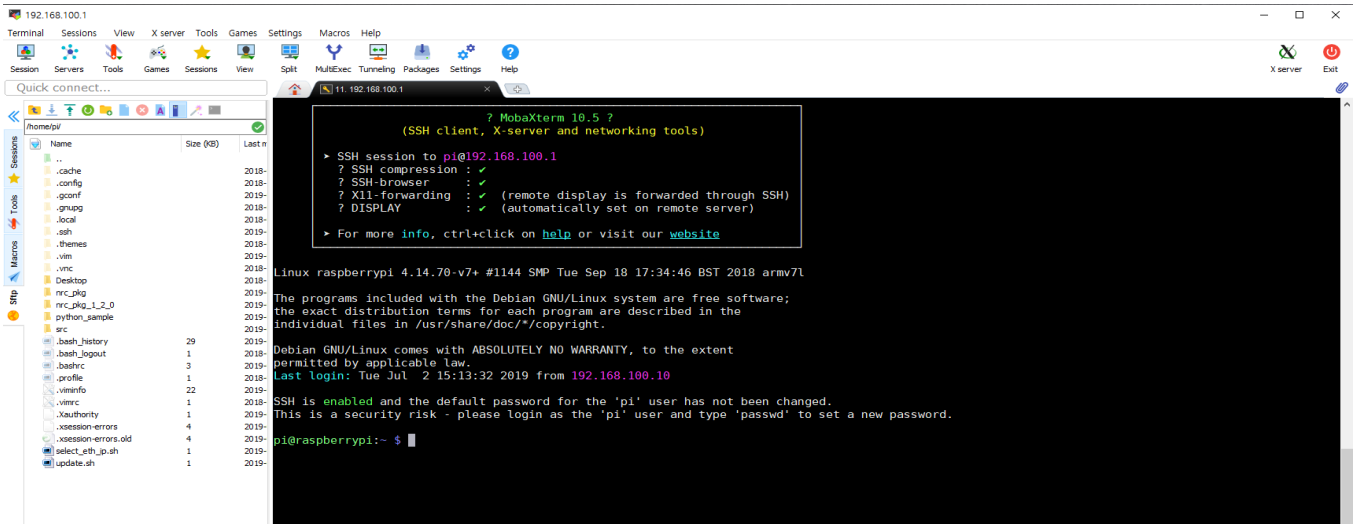


Figure 2.1 Mobaxterm display

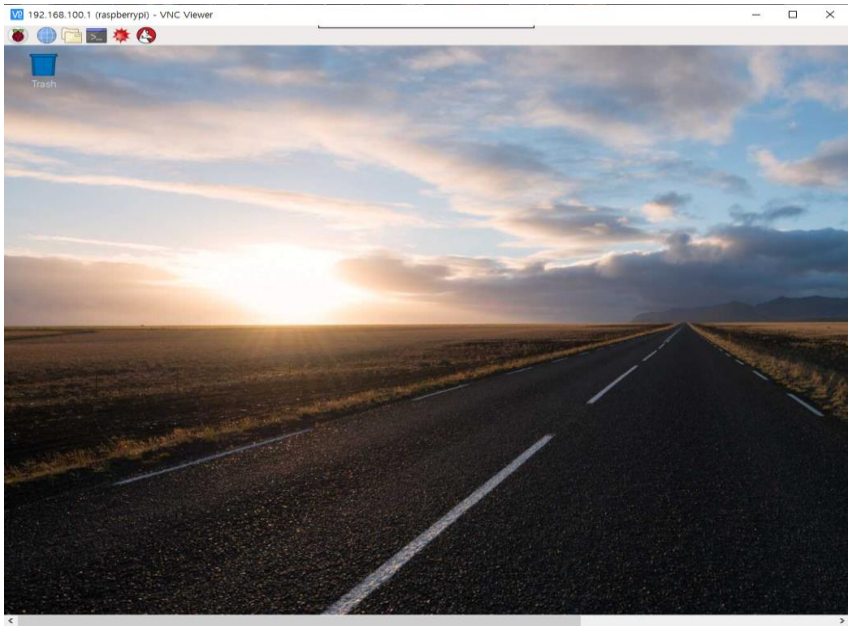


Figure 2.2 VNC viewer display

2.3 IP setting for Ethernet

By default, the RPi's Ethernet port is assigned an IP address through a DHCP server. To set up as an intra network, the Ethernet port can be used as a DHCP client by connecting the Ethernet cable from a switch, hub, or router to the RPi. The DHCP server on an intra network is assigned an Ethernet IP address and remotely accesses the RPi with the same IP address.

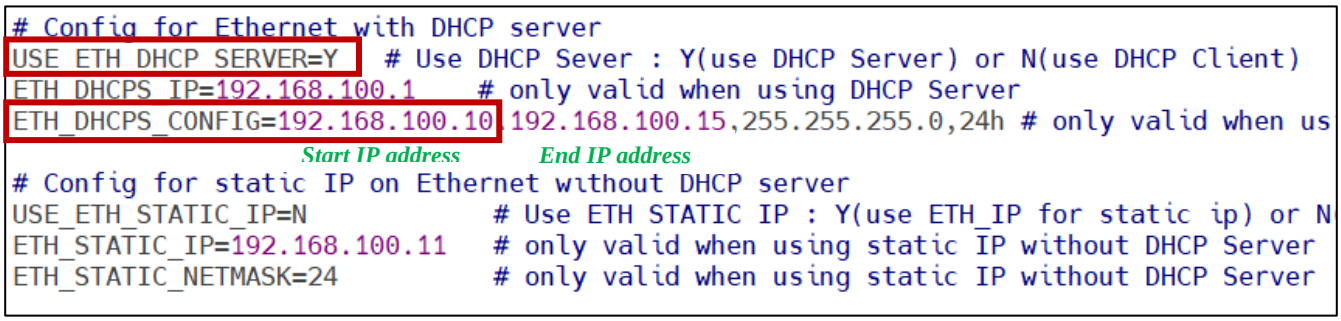
Script file (~/nrc_pkg/script/conf/etc/CONFIG_IP)

```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=N    # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.15,255.255.255.0,24h # only valid when usi

# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=N           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N(
ETH_STATIC_IP=192.168.100.11  # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24         # only valid when using static IP without DHCP Server
```

Figure 2.3 Configuration of IP setting for Ethernet DHCP client

If the RPi Ethernet port works as a DHCP server, change it to “USE_ETH_DHCP_SERVER=Y” and run the script and reboot the RPi. After the reboot, the Ethernet IP is set to 192.168.100.1. One device connected with an Ethernet cable will be assigned an IP of 192.168.100.10. If more than one DHCP client devices are required, change the 'End IP address' parameter of 'ETH_DHCPS_CONFIG' from '192.168.100.10' to the required range such as”
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.15,255.255.255.0,24h”




```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=Y   # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.15,255.255.255.0,24h # only valid when us

# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=N           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N
ETH_STATIC_IP=192.168.100.11  # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24         # only valid when using static IP without DHCP Server
```

Figure 2.4 Configuration of IP setting for Ethernet DHCP server

If the RPi's Ethernet Port is to work with Static IP instead of a DHCP Server, change it to “USE_ETH_DHCP_SERVER=N” and “USE_ETH_STATIC_IP=Y” and run the script and reboot the RPi. After reboot, the RPi's Ethernet IP is set to 192.168.100.11 and the device connected to the Ethernet cable is set to Static IP.

```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=N   # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.15,255.255.255.0,24h # only valid when us

# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=Y           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N
ETH_STATIC_IP=192.168.100.11  # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24         # only valid when using static IP without DHCP Server
```

Figure 2.5 Configuration of IP setting for Ethernet Static IP

3 NRC7292 EVK AP/STA operation

This chapter explains how to start the IEEE 802.11ah AP operation and enable STA to connect to AP.

3.1 Start Script

The “start.py” in ~/nrc_pkg/script folder is the unified script used to initiate AP, STA, Sniffer, MP, MAP, RELAY and STA with Ucode. For each operation, the different arguments are needed.

Three arguments which are *sta_type*, *security_mode*, *country* is necessary for an AP or STA operation.

However, for Sniffer operation, two additional arguments, *channel*, and *sniffer mode*, are needed.

Please refer to other user guide documents for the usage of MP, MAP for Mesh and RELAY.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py
Done.
Done.
Usage:
        start.py [sta_type] [security_mode] [country] [channel] [sniffer_mode]
        start.py [sta_type] [security_mode] [country] [mesh_mode] [mesh_peering] [mesh_ip]
Argument:
        sta_type      [0:STA   |  1:AP  |  2:SNIFFER  | 3:RELAY |  4:MESH | 5:IBSS]
        security_mode [0:Open  |  1:WPA2-PSK  |  2:WPA3-OWE  |  3:WPA3-SAE | 4:WPS-PBC]

        country       [US:USA  |  JP:Japan  |  TW:Taiwan  | EU:EURO | CN:China |

                       AU:Australia  |  NZ:New Zealand  | K1:Korea-USN  | K2:Korea-MIC]

        ------------------------------------------------------------
        channel       [S1G Channel Number]   * Only for Sniffer & AP
        sniffer_mode  [0:Local | 1:Remote]   * Only for Sniffer
        mesh_mode     [0:MPP | 1:MP | 2:MAP] * Only for Mesh
        mesh_peering  [Peer MAC address]     * Only for Mesh
        mesh_ip       [Static IP address]    * Only for Mesh
        ibss_ip       [0:DHCPC or static IP | 1:DHCPS]    * Only for IBSS
Example:
        OPEN mode STA for US               : ./start.py 0 0 US
        Security mode AP for US              : ./start.py 1 1 US
        Local Sniffer mode on CH 40 for Japan  : ./start.py 2 0 JP 40 0
        SAE mode Mesh AP for US              : ./start.py 4 3 US 2
        Mesh Point with static ip            : ./start.py 4 3 US 1 192.168.222.1

        Mesh Point with manual peering       : ./start.py 4 3 US 1 8c:0f:fa:00:29:46

        Mesh Point with manual peering & ip    : ./start.py 4 3 US 1 8c:0f:fa:00:29:46 192.168.222.1

        OPEN mode IBSS for US with DHCP server      : ./start.py 5 0 US 1
        Security mode IBSS for US with DHCPC client : ./start.py 5 1 US 0
        Security mode IBSS for US with static IP    : ./start.py 5 1 US 0 192.168.200.17
Note:
        sniffer_mode should be set as '1' when running sniffer on remote terminal
        MPP, MP mode support only Open, WPA3-SAE security mode
```

Figure 3.1 Usage of start.py script

[NOTE]

Executing "start.py" script overwrites some lane in the dhcpcd.conf (~/nrc_pkg/etc/dhcpcd/) and the dnsmasq.conf (~/nrc_pkg/etc/dnsmasq) files as shown below. To keep the user's configuration, users should leave that lane blanked.

dhcpcd.conf file	**dnsmasq.conf**
Line 59, 62 //IP address for wlan0 Line 65, 68 //IP address for wlan1 Line 71, 72 //IP address for eth0	Line 2 //DHCPS configuration for eth0 Line 3, 4 //DHCPS configuration for wlan0 or wlan1

3.2 AP mode operation

1) Open terminal with SSH

After boot-up of AP board, connect via SSH to the AP by using the terminal emulator like MobaXterm. The ID and PW are as follows:

- ID : pi
- PW : raspberry

2) Run script

To run AP, a user should give "1" as *sta_type* and select one of the security mode and country code.

parameter setting

- *sta_type* : **1 (AP)**
- *security_mode* : available modes are 0(open), 1(WPA2-PSK), 2(WPA3-OWE) and 3(WPA3-SAE)
- *country* : available country codes are US, JP, TW, KR(K0/K1/K2), EU, CN, AU, NZ

Ex) open mode AP to be used in Korea : ./start.py 1 0 K1

3) Check results

Once AP runs by start script, there are several procedures that are executed: (1) clear apps (2) copy firmware (3) load module (4) set configurations (5) start the hostapd (6) set NAT (7) start DNSMASQ, and performed in sequential order. If all procedures are successful, the user can find the wlan0 interface with the IP address created as shown in Figure 3.3.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py 1 0 K1
Done.
Done.
-----------------------------
Model            : 7292
STA Type         : AP
Country          : K1
Security Mode    : OPEN
BD Name          : nrc7292_bd.dat
AMPDU            : AUTO
Download FW      : uni_s1g.bin
MAX TX Power     : 24 dBm
BSS MAX IDLE     : 1800
-----------------------------
NRC AP setting for HaLow...
[*] Set Max CPU Clock on RPi
1400000
1400000
1400000
1400000
Done
[0] Clear
Selected interface 'wlan0'
wlan0: AP-DISABLED
nl80211: deinit ifname=wlan0 disabled_11b_rates=0
wlan0: interface state ENABLED->DISABLED
OK
Selected interface 'wlan0'
Selected interface 'wlan0'
[1] Copy and Set Module Parameters
total 632
drwxr-xr-x 2 pi   pi     4096 Jun 16 10:43 .
drwxr-xr-x 4 pi   pi     4096 Jun 16 10:42 ..
-rwxrwxrwx 1 pi   pi      271 Jun 16 10:42 copy
-rwxr-xr-x 1 pi   pi     2148 Jun 16 10:42 nrc7292_bd.dat
-rwxr-xr-x 1 pi   pi   315260 Jun 16 10:42 nrc7292_cspi.bin
-rwxr-xr-x 1 root root 315260 Jun 16 11:22 uni_s1g.bin
-rwxr-xr-x 1 root root 315260 Jun 16 11:22 /lib/firmware/uni_s1g.bin
====================================
 INTERFACE        : eth0
 ETH STATIC IP    : 10.10.10.81
 NET MASK NUM     : 16
 AP INTERFACE     : wlan0
 AP STATIC IP     : 192.168.200.1
 NET MASK NUM     : 24
====================================
Config for AP is done!
IP and DHCP config done
[2] Set Initial Country
[3] Loading module
sudo insmod /home/pi/nrc_pkg/sw/driver/nrc.ko  hifspeed=20000000 spi_bus_num=0 spi_cs_num=0 spi_gpio_ir
q=5 spi_polling_interval=0 fw_name=uni_s1g.bin bss_max_idle=1800 ndp_preq=1 listen_interval=1000 kr_ban
d=1 bd_name=nrc7292_bd.dat
[4] Set Maximum TX Power
Type     : limit        Tx power : 24
                        OK
[*] Transmission Power Control(TPC) is activated
[5] Set guard interval
OK
[*] Start DHCPCD and DNSMASQ
[*] Self configuration off
```

Figure 3.2 Results of running AP (1/2)

```
[6] Start hostapd on wlan0
[*] configure file copied from: /home/pi/nrc_pkg/script/conf/K1/ap_halow_open.conf
wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED
[7] Start NAT
[8] ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.81  netmask 255.255.0.0  broadcast 10.10.255.255
        inet6 fe80::1bf7:34e9:d731:517d  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:35:f3:79  txqueuelen 1000  (Ethernet)
        RX packets 66436  bytes 62469716 (59.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40935  bytes 3230026 (3.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.179  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::32a7:7d7c:74d5:2ac6  prefixlen 64  scopeid 0x20<link>
        ether 70:88:6b:80:98:98  txqueuelen 1000  (Ethernet)
        RX packets 351016  bytes 45861861 (43.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7865  bytes 576584 (563.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 191  bytes 18994 (18.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 191  bytes 18994 (18.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.1  netmask 255.255.255.0  broadcast 192.168.200.255
        ether 8c:0f:fa:00:2a:d2  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 1218 (1.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

HaLow AP ready
------------------------------------------------------------------
Done.
```

Figure 3.3 Results of running AP (2/2)

3.3 AP mode operation with Self-configuration

We provide 'Self-configuration' feature on AP, which is optional but aims to start AP with optimal channel after CCA scan before starting hostapd. (c.f. without the feature, AP just use static CH in conf. file of hostapd)

AP executes CCA scan on every CH, finds the best channel, and finally starts with the channel once the feature is enabled in the 'start.py'. There are 3 parameters related to the feature and Figure 3.4 shows them.

- self_config: enable/disable feature (default: disabled)
- perfer_bw: set the specific bandwidth you want to find (1MHz, 2MHz, 4MHz or any (all BW))
- dwell_time: time (in ms unit) to wait on each channel for CCA Scan (default: 100ms)

```
#------------------------------------------------------------------------------#
# Self configuration (AP Only)
#  AP scans the clearest CH and then starts with it
self_config       = 0        # 0 (disable)  or 1 (enable)
prefer_bw         = 0        # 0: no preferred bandwidth, 1: 1M, 2: 2M, 4: 4M
dwell_time        = 100      # max dwell is 1000 (ms), min: 10ms, default: 100ms
#------------------------------------------------------------------------------#
```

Figure 3.4 Parmeter for Self-Configuration

CCA scan results (high percentage means CH is busy) on each channel is shown and best CH is selected accordingly. For this example, best CH is 39 (1MHz 928.5MHz) because CCA percentage is the lowest (0%) among all the channels. If you set prefer_bw = 0, 1M BW CH has higher priority than 2 and 4M BW CH with the same percentage. Basically, the best CH is automatically selected by CCA results, so it takes much more time according to the number of CH and dwell time on each CH.

```
[*] Self configuration start!
country: K1, prefer_bw: 0, dwell_time: 100
Start CCA scan.... It will take up to 2.6 sec to complete
        Frequency       CCA     bandwidth
--      922.0 MHz       0.0%    1M
--      923.0 MHz       0.6%    1M
--      922.5 MHz       0.3%    2M
[Optimal freq.] 922.0 MHz (CCA:0.0%, BW:1M)
[*]ch_num:36
OK
```

Figure 3.5 Results of Self-Configuration on AP

3.4 STA mode operation

1) Open terminal with SSH

After boot-up of STA board, connect via SSH to the STA by using the terminal emulator like MobaXterm. The ID and PW are as follows:

- ID : pi
- PW : raspberry

2) Run script

Running STA is similar to the procedure as AP's except for giving "0" as *sta_type*.

parameter setting

- *sta_type* : **0 (STA)**
- *security_mode* : available modes are 0(open), 1(WPA2-PSK), 2(WPA3-OWE) and 3(WPA3-SAE)
- *country* : available country codes are US, JP, TW, KR(K0/K1/K2), EU, CN, AU, NZ

Ex) security mode STA to be used in Japan : ./start.py 0 1 K1

3) Check results

Once STA runs by start script, there are several procedures that is performed, (1) clear apps (2) copy firmware (3) load module (4) Set configurations (5) start wpa supplicant (6) start scan AP (7) connect to AP (8) start DHCP client are performed sequentially. If all the procedures are successful, STA finally gets an IP address.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py 0 0 K1
Done.
Done.
-----------------------------
Model            : 7292
STA Type         : STA
Country          : K1
Security Mode    : OPEN
BD Name          : nrc7292_bd.dat
AMPDU            : AUTO
CQM              : ON
Download FW      : uni_s1g.bin
MAX TX Power     : 24 dBm
Power Save Type  : Always On
Listen Interval  : 1000
-----------------------------
NRC STA setting for HaLow...
[*] Set Max CPU Clock on RPi
1400000
1400000
1400000
1400000
Done
[0] Clear
Selected interface 'wlan0'
wlan0: AP-DISABLED
nl80211: deinit ifname=wlan0 disabled_11b_rates=0
wlan0: interface state ENABLED->DISABLED
OK
Selected interface 'wlan0'
Selected interface 'wlan0'
[1] Copy and Set Module Parameters
total 632
drwxr-xr-x 2 pi   pi     4096 Jun 16 10:43 .
drwxr-xr-x 4 pi   pi     4096 Jun 16 10:42 ..
-rwxrwxrwx 1 pi   pi      271 Jun 16 10:42 copy
-rwxr-xr-x 1 pi   pi     2148 Jun 16 10:42 nrc7292_bd.dat
-rwxr-xr-x 1 pi   pi   315260 Jun 16 10:42 nrc7292_cspi.bin
-rwxr-xr-x 1 root root 315260 Jun 16 11:29 uni_s1g.bin
-rwxr-xr-x 1 root root 315260 Jun 16 11:29 /lib/firmware/uni_s1g.bin
====================================
 INTERFACE        : eth0
 ETH STATIC IP    : 10.10.10.81
 NET MASK NUM     : 16
 STA INTERFACE    : wlan0
 USE DHCP Client
====================================
Config for STA is done!
IP and DHCP config done
[2] Set Initial Country
[3] Loading module
sudo insmod /home/pi/nrc_pkg/sw/driver/nrc.ko  hifspeed=20000000 spi_bus_num=0 spi_cs_num=0 spi_gpio_ir
q=5 spi_polling_interval=0 fw_name=uni_s1g.bin listen_interval=1000 kr_band=1 bd_name=nrc7292_bd.dat
[4] Set Maximum TX Power
Type     : limit        Tx power : 24
                        OK
[*] Transmission Power Control(TPC) is activated
```

Figure 3.6 Results of running STA (1/2)

```
[5] Set guard interval
OK
[*] Start DHCPCD and DNSMASQ
wpa_supplicant: no process found
[6] Start wpa_supplicant on wlan0
Successfully initialized wpa_supplicant
wlan0: SME: Trying to authenticate with 8c:0f:fa:00:2b:39 (SSID='halow_demo' freq=5180 MHz)
wlan0: Trying to associate with 8c:0f:fa:00:2b:39 (SSID='halow_demo' freq=5180 MHz)
wlan0: Associated with 8c:0f:fa:00:2b:39
wlan0: CTRL-EVENT-CONNECTED - Connection to 8c:0f:fa:00:2b:39 completed [id=0 id_str=]
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
[7] Connect and DHCP
Waiting for IP
        inet 192.168.200.37  netmask 255.255.255.0  broadcast 192.168.200.255
IP assigned. HaLow STA ready
------------------------------------------------------------------
Done.
```

Figure 3.7 Results of running STA (2/2)

3.5 Configure static IP address

The DHCP client on a STA can obtain an IP address after successful connection to the AP. Or the users can assign a static IP address to the STA by using the script file provided with software package. Figure 3.8 presents the script (~/nrc_pkg/script/conf/etc/CONFIG_IP) and indicates the fields to set.

```
# Config for HaLow STA's IP and Defautl GW
USE_HALOW_STA_STATIC_IP=N     # Use STATIC IP :  Y(use Static IP) or N(use Dynamic IP(DHCP))
HALOW_STA_IP=192.168.200.11   # only valid when using static IP
HALOW_STA_NETMASK=24          # only valid when using static IP
HALOW_STA_DEFAULT_GW=192.168.200.1 # only valid when using static IP
```

Figure 3.8 Configure static IP address for STA

In addition, users can configure static IP addresses in the same way as the STA configuration. Figure 3.6 shows the fields to configure.

```
# Config for HaLow AP's IP and DHCPS configuration
HALOW_AP_IP=192.168.200.1
HALOW_AP_NETMASK=24
HALOW_AP_DHCPS_CONFIG=192.168.200.10,192.168.200.50,255.255.255.0,24h
```

Figure 3.9 Configure static IP address for AP

4 NRC7292 EVK performance evaluation

4.1 Performance test

Users can evaluate throughput performance by using the iperf3 tool.

1) Run iperf3 server at AP side

```
pi@raspberrypi:~/nrc_pkg/script $ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.200.37, port 48108
[  5] local 192.168.200.1 port 5201 connected to 192.168.200.37 port 33605
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   320 KBytes  2.62 Mbits/sec  6.193 ms  0/40 (0%)
[  5]   1.00-2.00   sec   352 KBytes  2.88 Mbits/sec  9.022 ms  0/44 (0%)
[  5]   2.00-3.00   sec   352 KBytes  2.88 Mbits/sec  10.556 ms  0/44 (0%)
[  5]   3.00-4.00   sec   368 KBytes  3.01 Mbits/sec  7.511 ms  0/46 (0%)
[  5]   4.00-5.00   sec   376 KBytes  3.08 Mbits/sec  7.605 ms  0/47 (0%)
[  5]   5.00-6.00   sec   368 KBytes  3.01 Mbits/sec  8.038 ms  0/46 (0%)
[  5]   6.00-7.00   sec   368 KBytes  3.01 Mbits/sec  7.942 ms  0/46 (0%)
[  5]   7.00-8.00   sec   400 KBytes  3.28 Mbits/sec  6.621 ms  0/50 (0%)
[  5]   8.00-9.00   sec   384 KBytes  3.15 Mbits/sec  8.250 ms  0/48 (0%)
[  5]   9.00-10.00  sec   360 KBytes  2.95 Mbits/sec  9.083 ms  0/45 (0%)
[  5]  10.00-10.24  sec  96.0 KBytes  3.22 Mbits/sec  9.234 ms  0/12 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-10.24  sec  0.00 Bytes  0.00 bits/sec  9.234 ms  0/468 (0%)
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 4.1 Run iperf3 server

2) Run iperf3 client at STA side (users can use other iperf3 options as well)

```
pi@raspberrypi:~/nrc_pkg/script $ iperf3 -c 192.168.200.1 -u -b 10m -t 10
Connecting to host 192.168.200.1, port 5201
[  4] local 192.168.200.37 port 33605 connected to 192.168.200.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   416 KBytes  3.41 Mbits/sec  52
[  4]   1.00-2.00   sec   360 KBytes  2.95 Mbits/sec  45
[  4]   2.00-3.00   sec   344 KBytes  2.82 Mbits/sec  43
[  4]   3.00-4.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   4.00-5.00   sec   376 KBytes  3.08 Mbits/sec  47
[  4]   5.00-6.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   6.00-7.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   7.00-8.00   sec   392 KBytes  3.21 Mbits/sec  49
[  4]   8.00-9.00   sec   392 KBytes  3.21 Mbits/sec  49
[  4]   9.00-10.00  sec   360 KBytes  2.95 Mbits/sec  45
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  3.66 MBytes  3.07 Mbits/sec  9.234 ms  0/468 (0%)
[  4] Sent 468 datagrams

iperf Done.
```

Figure 4.2 Run iperf3 client

4.2 Enable/Disable A-MPDU

Aggregate MAC protocol data unit (A-MPDU) is a MAC frame with multiple MAC sub-frames and single PHY header. By using A-MPDU, users can improve throughput because overhead is decreased in PHY header and inter-frame space (IFS).

Users can enable/disable A-MPDU by changing the *ampdu_enable* value in the 'start.py' script as shown in Figure 4.3. Setting "0" disables A-MPDU, setting "1" enables manual A-MPDU and setting "2" enables auto A-MPDU.

When AMPDU is enabled, maximum numbers of aggregation (i.e. maximum numbers of MPDU per AMPDU) is decided by NDP block ACK type. In case of using 2M NDP Block ACK, maximum 16 MPDUs can be aggregated and maximum 8 MPDUs while using 1M NDP Block ACK. Which one is used can be chosen by a parameter “ndp_ack_1m” in start.py (default: 2M NDP Block ACK on 2M or 4M Bandwidth CH, 1M NDP Block ACK on 1M Bandwidth CH) Please note that only 1M NDP Block ACK can be used on 1MHz Bandwidth channel regardless of the setting of ndp_ack_1m.

Please note that ndp_ack_1m should be set on AP and STA should follow AP’s setting.

```
#------------------------------------------------------------------------------#
# AMPDU (Aggregated MPDU)
#  Enable AMPDU for full channel utilization and throughput enhancement (Default: auto)
#  disable(0): AMPDU is deactivated
#   manual(1): AMPDU is activated and BA(Block ACK) session is made by manual
#     auto(2): AMPDU is activated and BA(Block ACK) session is made automatically
ampdu_enable      = 2        # 0 (disable) 1 (manual) 2 (auto)
#------------------------------------------------------------------------------#
```

Figure 4.3 Enable/disable A-MPDU

4.3 Enable/Disable NDP Probe Request

A probe request frame may be transmitted by the STA at the 802.11 baseline to collect information required for connection operation.
The PHY header uses a robust coding scheme to improve the transmission reliability of NDP probe requests, and the NDP probing procedure is used to reduce energy consumption during the scan procedure.
Users can enable/disable NDP Probe Request by changing the ndp_preq value of the 'start.py' script as shown in Figure 4.4. If "1" is set to ndp_preq, NDP Probe Request is activated. If it is set to "0", Legacy Probe Request is used. To activate the NDP Probe Request, "scan_ssid=1" must be included in the wpa_supplicant configure file.

```
#------------------------------------------------------------------------------#
# NDP Probe Request
#  For STA, "scan_ssid=1" in wpa_supplicant's conf should be set to use
ndp_preq          = 0        # 0 (Legacy Probe Req) 1 (NDP Probe Req)
#------------------------------------------------------------------------------#
```

Figure 4.4 Enable/disable NDP Probe Request

4.4 Enable/Disable power save

As shown in Figure 4.5, the user can enable/disable the power saving option by changing the power_save value in the 'start.py' script, and 4 options can be selected, and the default is 0 (disable).

Setting to "0" disables the power saving function. Values "2" and "3" enable deep sleep, which turns off almost all power, and can save more power than modem sleep.

A value of "2" activates deep sleep mode (TIM) and periodically checks the beacon during power save, A value of "3" activates the deep sleep mode (nonTIM), and it is not awakened by the AP's beacon, but is awakened by the station's Tx or an external interrupt.

```
#------------------------------------------------------------------------------#
# Power Save (STA Only)
#  3-types PS: (0)Always on (2)Deep_Sleep(TIM) (3)Deep_Sleep(nonTIM)
#     TIM Mode : check beacons during PS to receive BU from AP
#  nonTIM Mode : Not check beacons during PS (just wake up by TX or EXT INT)
power_save        = 0        # STA (power save type 0~3)
ps_timeout        = '3s'     # STA (timeout before going to sleep) (min:1000ms)
sleep_duration    = '3s'     # STA (sleep duration only for nonTIM deepsleep) (min:1000ms)
listen_interval   = 1000     # STA (listen interval in BI unit) (max:65535)
                             # Listen Interval time should be less than bss_max_idle time to avoid asso
ciation reject
#------------------------------------------------------------------------------#
```

Figure 4.5 Enable/disable power save

4.5 Enable/Disable BSS MAX IDLE element

A user can enable/disable the option to download bss_max_idle with value by changing the bss_max_idle_enable in the 'start.py' script.

The bss_max_idle is used to check whether the STA is deactivated, and if the keep alive packet or any data is not received from the STA within the bss_max_idle value, the AP determines that the STA is no longer connected. If the AP is determined that the STA cannot be reached, the STA information is cleared through disassoc frame / deauthentication frame. The default value is 1800 which means about 1843.2 seconds and can be set up to 65535. The default 'bss_max_idle_enable' is 1.

```
#-------------------------------------------------------------------------------#
# BSS MAX IDLE PERIOD (aka. keep alive) (AP Only)
#  STA should follow (i.e STA should send any frame before period),if enabled on AP
#  Period is in unit of 1000TU(1024ms, 1TU=1024us)
#  Note: if disabled, AP removes STAs' info only with explicit disconnection like deauth
bss_max_idle_enable = 1      # 0 (disable) or 1 (enable)
bss_max_idle        = 1800   # time interval (e.g. 1800: 1843.2 sec) (1 ~ 65535)
#-------------------------------------------------------------------------------#
```

Figure 4.6 Enable/disable BSS MAX IDLE

5 Internet connection

If NAT configuration is complete with the start script in chapter 0 and AP is connected to the Internet through its Ethernet interface, then the STA connected to the AP can access the Internet via AP. After boot-up, AP can obtain IP address from DHCP server.

On successful Wi-Fi connection, users can verify the reachability to the internet by using a ping program.

```
pi@raspberrypi:~/nrc_pkg/script $ ping google.com -c 3
PING google.com (172.217.24.206) 56(84) bytes of data.
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=1 ttl=50 time=37.9 ms
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=2 ttl=50 time=37.3 ms
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=3 ttl=50 time=38.7 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 37.380/38.016/38.716/0.547 ms
```

Figure 5.1 Result of ping

Once the internet connection is confirmed by STA, users can surf the web and play YouTube.

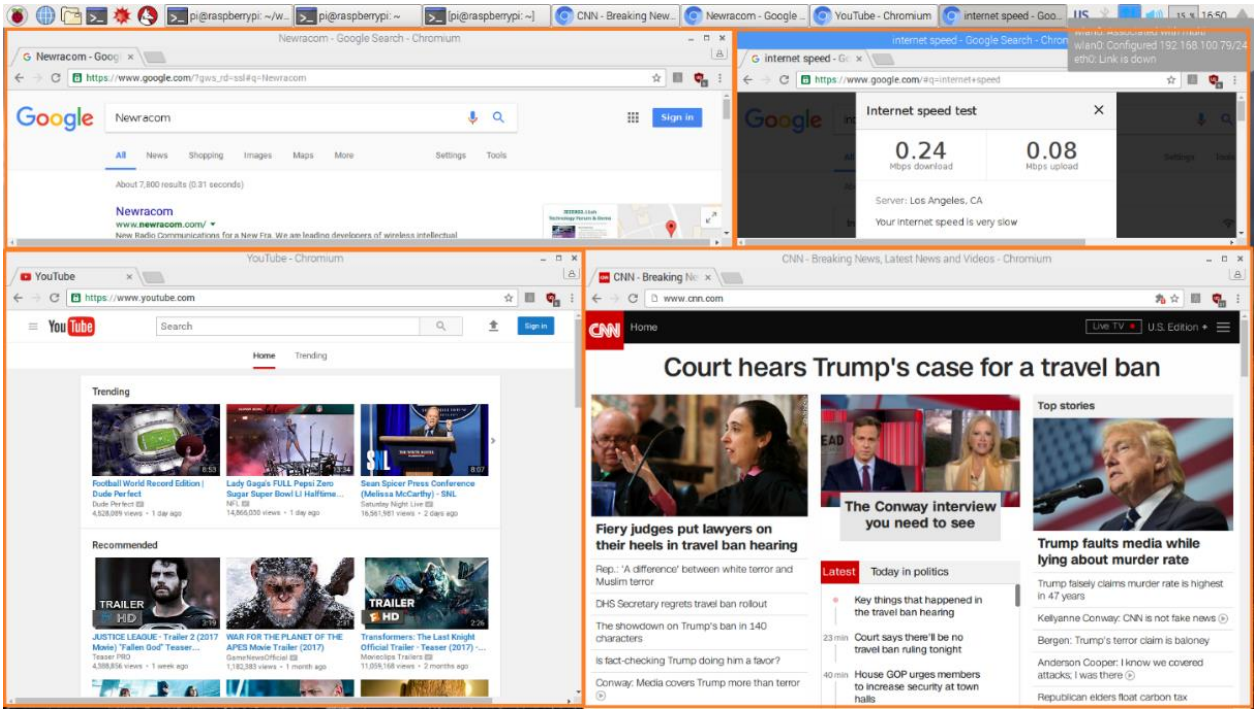


Figure 5.2 Internal connection

6 Change configuration

NRC7292 supports 1/2/4 MHz channel bandwidth (only 1 MHz for JP and 1/2 MHz for EU). The default channel bandwidth and frequency are different for every country (e.g. 2 MHz BW at 909 MHz for US, 1MHz BW at 917 MHz for JP, 2 MHz BW at 843.5 MHz for TW. Users can change the frequency band and channel bandwidth by editing only the frequency band index of the configuration file of AP named COUNTRY/*ap_halow_open.conf* (Users must attempt to change the channel on the AP).

```
ctrl_interface=/var/run/hostapd
country_code=US
interface=wlan0
#ssid=halow_demo
ssid=halow
hw_mode=a

#1MHz
#909.5MHz
#channel=37
#915.5MHz
#channel=43
#922.5MHz
#channel=150

#2MHz
#909MHz
#channel=153
#915MHz
#channel=156
#921MHz
#channel=159
#925MHz
channel=161

#4MHz
#910MHz
#channel=162
#914MHz
#channel=163
#922MHz
#channel=165

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

```
ctrl_interface=/var/run/hostapd
country_code=JP
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#917MHz
channel=36
#918MHz
#channel=37
#919MHz
#channel=38
#920MHz
#channel=39
#921MHz
#channel=40
#922MHz
#channel=41
#923MHz
#channel=42
#924MHz
#channel=43
#925MHz
#channel=44
#926MHz
#channel=45
#927MHz
#channel=46

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

```
ctrl_interface=/var/run/hostapd
country_code=TW
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#839MHz
#channel=36
#843MHz
#channel=40
#850MHz
#channel=47

#2MHz
#839.5MHz
#channel=149
#843.5MHz
channel=151
#845.5MHz
#channel=152
#849.5MHz
#channel=154

#4MHz
#840.5MHz
#channel=155
#844.5Hz
#channel=156
#848.5MHz
#channel=157

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

Figure 6.1 Contents of ap_halow_open.conf file (US/JP/TW)

```
ctrl_interface=/var/run/hostapd
country_code=KR
interface=wlan0
ssid=halow_demo
hw_mode=a

#tx_queue_data2_aifs=3
#tx_queue_data2_cwmin=15
#tx_queue_data2_cwmax=63
#tx_queue_data2_burst=30

#1MHz
#918MHz
#channel=36
#919MHz
#channel=37
#920MHz
#channel=38
#921MHz
#channel=39
#922MHz
#channel=40
#923MHz
#channel=41
#################
#942.8MHz
#channel=46
#943.8MHz
#channel=47
#944.8MHz
#channel=48
#945.8MHz
#channel=149

#2MHz
#918.5MHz
#channel=42
#920.5MHz
#channel=43
#922.5MHz
channel=44
#################
#943.3MHz
#channel=150
#945.3MHz
#channel=151

#4MHz
#921.5MHz
#channel=45
#################
#944.3MHz
#channel=152
```

```
ctrl_interface=/var/run/hostapd
country_code=DE
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#863.5MHz
#channel=36
#864.5MHz
#channel=37
#865.5MHz
#channel=38
#866.5MHz
#channel=39
#867.5MHz
#channel=40

#2MHz
#864MHz
channel=41
#866MHz
#channel=42
```

```
ctrl_interface=/var/run/hostapd
country_code=CN
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#755.5MHz
#channel=36
#759.5MHz
#channel=40
#763.5MHz
#channel=44
#767.5MHz
#channel=48
#770.5MHz
#channel=151
#779.5MHz
#channel=152
#780.5MHz
#channel=153
#781.5MHz
#channel=154
#782.5MHz
#channel=155
#783.5MHz
#channel=156
#784.5MHz
#channel=157
#785.5MHz
#channel=158
#786.5MHz
#channel=159

#2MHz
#780MHz
channel=160
#782MHz
#channel=161
#784MHz
#channel=162
#786MHz
#channel=163

#4MHz
#781MHz
#channel=164
#785MHz
#channel=165
```

Figure 6.2 Contents of ap_halow_open.conf file (KR/EU/CN)

For the latest S1G channel information, please refer to “NRC7292 EVK User Guide (S1G Channel)”.

Note that when changing the AP channel, when using Sub 1 GHz of the Available frequency band index (1~13), the AP configuration file such as ap_halow_open.conf must be modified. The 'hw_mode=a' in the 7th line of the configure file for AP must be changed to 'hw_mode=g', and it is only applicable to US Channel.

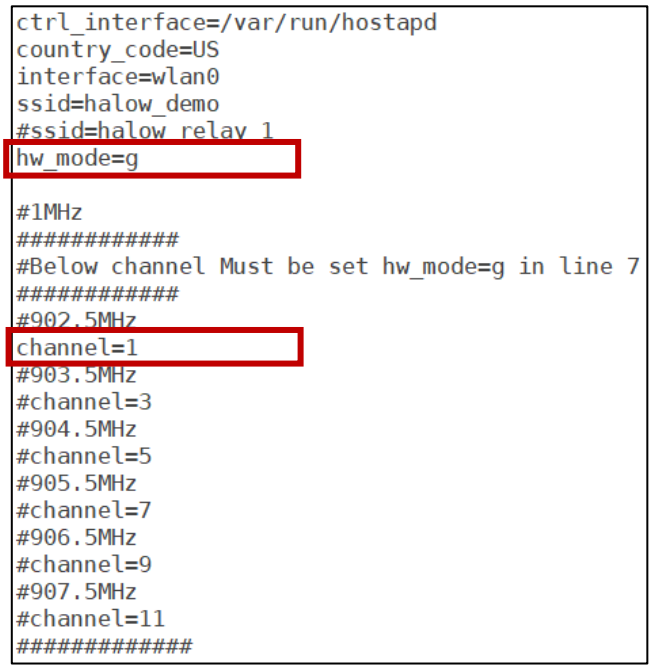


Figure 6.3 hw_mode of ap_halow_open.conf file (US)

Table 6.1 US Channels for hw_mode=g

Available frequency band index	Bandwidth (MHz)	Sub 1 GHz frequency (MHz)
1	1	902.5
3	1	903.5
5	1	904.5
7	1	905.5
9	1	906.5
11	1	907.5
2	2	903
6	2	905
10	2	907
8	2	906

7 NRC7292 EVK software

Table 7.1 and Table 7.2 show AP and STA's file directory on the Raspberry PI3 host. NRC7292 EVK contains some scripts to start AP/STA operation, a configuration file for operation, Linux Wi-Fi module driver, and NRC7292 firmware.

Table 7.1 Files in AP

File or Folder	Path	Description
ap_halow_open.conf	NRC_PKG/script/conf/COUNTRY	AP configuration file (open mode)
ap_halow_wpa2_conf	NRC_PKG/script/conf/COUNTRY	AP configuration file (WPA2-PSK)
ap_halow_owe_conf	NRC_PKG/script/conf/COUNTRY	AP configuration file (WPA3-OWE)
ap_halow_sae_conf	NRC_PKG/script/conf/COUNTRY	AP configuration file (WPA3-SAE)
start.py	NRC_PKG/script	start script
stop.py	NRC_PKG/script	stop script
CONFIG_IP	NRC_PKG/script/conf/etc	IP and DHCP configuration
ip_config.sh	NRC_PKG/script/conf/etc	IP and DHCP configuration script
clock_config.sh	NRC_PKG/script/conf/etc	RPi Max Clock configuration script
nrc.ko	NRC_PKG/sw/driver	NRC7292 Host Wi-Fi driver
uni_s1g.bin	NRC_PKG/sw/firmware	NRC7292 firmware

※ NRC_PKG = /home/pi/nrc_pkg/

Table 7.2 Files in STA

File or Folder	Path	Description
sta_halow_open.conf	NRC_PKG/script/conf/COUNTRY	STA configuration file (open mode)
sta_halow_wpa2_conf	NRC_PKG/script/conf/COUNTRY	STA configuration file (WPA2-PSK)
sta_halow_owe_conf	NRC_PKG/script/conf/COUNTRY	STA configuration file (WPA3-OWE)
sta_halow_sae_conf	NRC_PKG/script/conf/COUNTRY	STA configuration file (WPA3-SAE)
start.py	NRC_PKG/script	start script
stop.py	NRC_PKG/script	stop script
CONFIG_IP	NRC_PKG/script/conf/etc	IP and DHCP configuration
ip_config.sh	NRC_PKG/script/conf/etc	IP and DHCP configuration script
clock_config.sh	NRC_PKG/script/conf/etc	RPi Max Clock configuration script
nrc.ko	NRC_PKG/sw/driver	NRC7292 Host Wi-Fi driver
uni_s1g.bin	NRC_PKG/sw/firmware	NRC7292 firmware

※ NRC_PKG = /home/pi/nrc_pkg/

8 NRC7292 EVK Sniffer operation

NRC7292 EVK can be used to capture 11ah frames in the air like other Wi-Fi sniffer devices in the market. For the installation of software package and operation for sniffer, please refer to “NRC7292 EVK User Guide (NewraPeek™)”.

9 Revision history

Revision No	Date	Comments
Ver 1.0	11/1/2018	Initial version for customer release created
Ver 1.1	3/25/2019	Description updated according to the new script and sniffer mode operation added
Ver 1.2	4/12/2019	CN Table added for China updated
Ver 1.3	7/2/2019	Manipulation NRC7292 EVK updated WPA3-OWE, WPA3-SAE added Method to enable/disable A-MPDU added
Ver 1.4	7/12/2019	NOTE for “start.py” execution added Static IP address assignment for AP and STA added
Ver 1.5	11/14/2019	Update IP configuration using CONFIG_IP & power save
Ver 1.6	12/4/2019	
Ver1.7	6/29/2020	Version 2.0 HW appearance updated
Ver1.8	8/5/2020	Update Board data and KR MIC Channel
Ver1.9	8/26/2020	Update BSS MAX IDLE element
Ver 2.0	4/19/2021	Update Power Save and NDP Probe Request
Ver 2.1	8/13/2021	Comment for USB interface support added
Ver 2.2	9/20/2021	Added NZ, AU channel table
Ver 2.3	10/15/2021	Added KR-USN channel table
Ver 2.4	10/21/2021	Added Self Configuration on AP
Ver 2.5	6/12/2023	Update for v1.4 SW package

Appendix A.
Upgrade hostapd & wpa_supplicant for supporting WPA3

A.1 Overview

WPA3 is the next generation of Wi-Fi security and provides state-of-the-art security protocols to the market. So, all WPA3 networks:

- Use the latest security methods
- Disallow outdated legacy protocols
- Require use of Protected Management Frame (PMF)

WPA3-Personal brings better protections by providing robust password-based authentication. This capability is enabled by Simultaneous Authentication of Equals (SAE), which replaces Pre-Shared Key (PSK) in WPA2-Personal.

WPA3-Enterprise has two modes. Basic mode is based on WPA2-Enterprise and PMF. An optional mode using 192-bit security protocols is also defined in WPA3-Enterprise, but this is not adequate for the IoT application. So, it is not supported in NRC7292 EVK.

However, NRC7292 EVK supports Wi-Fi Enhanced Open mode, which is based on Opportunistic Wireless Encryption (OWE) and replaces open mode.

In summary, NRC7292 EVK supports following WPA3 security modes.

- Wi-Fi Enhanced Open (OWE mode)
- WFA3-Personal (WPA3-SAE mode)

By the way, it is necessary recommendation to upgrade hostapd and wpa_supplicant to version 2.10 for the full support of WPA3 protocols. Please follow the steps listed below to upgrade version.

A.2 Upgrade hostapd

A.2.1 Download hostapd v2.10 and install required libraries

Please follow the procedure below.

```
$ wget http://w1.fi/releases/hostapd-2.10.tar.gz
$ tar zxf hostapd-2.10.tar.gz
$ sudo apt-get update
$ sudo apt-get install libnl-3-dev libnl-genl-3-dev libssl-dev
```

A.2.2 Build and install hostapd v2.10

Please follow the procedure below.

```
$ cd hostapd-2.10/hostapd
$ cp defconfig .config
$ vi .config
Enable followings:
CONFIG_IEEE80211N=y
CONFIG_OWE=y
CONFIG_WPS=y
Insert following:
CONFIG_SAE=y
CONFIG_TESTING_OPTIONS=y
$ vi ../src/ap/wpa_auth.c
Comment line 72:
69 //static const u32 eapol_key_timeout_subseq = 1000; /* ms */
Add line 73:
70 static const u32 eapol_key_timeout_subseq = 2000; /* ms */
$ make
$ sudo make install
```

Note. eapol_key_timeout_subseq = 2000 should be added to support WPA3-OWE (This is only to support NRC7292 devices.)

A.3 Upgrade wpa_supplicant

A.3.1 Download wpa_supplicant v2.10 and install required libraries

Please follow the procedure below.

```
$ wget http://w1.fi/releases/wpa_supplicant-2.10.tar.gz
$ tar zxf wpa_supplicant-2.10.tar.gz
$ sudo apt-get update
$ sudo apt-get install libnl-3-dev libnl-genl-3-dev libssl-dev
$ sudo apt-get install libdbus-1-dev libdbus-glib-1-dev
```

A.3.2 Build and install wpa_supplicant v2.10

Please follow the procedure below.

```
$ cd wpa_supplicant-2.10/wpa_supplicant
$ cp defconfig .config
$ vi .config
```

Enable followings:

```
CONFIG_IEEE80211N=y
CONFIG_OWE=y
CONFIG_SAE=y
CONFIG_MESH=y
CONFIG_WPS=Y
CONFIG_TESTING_OPTIONS=y
```

Disable followings:

```
#CONFIG_DRIVER_WIRED=y
#CONFIG_DRIVER_MACSEC_LINUX=y
#CONFIG_CTRL_IFACE_DBUS_NEW=y
#CONFIG_WNM=y
```

```
$ make
$ sudo make install
```