NEW RADIO COMMUNICATION FOR NEW ERA

NRC7394 Evaluation Kit User Guide

(Standalone)

Ultra-low power & Long-range Wi-Fi

Ver 1.3.2
Sep. 25, 2025

NEWRACOM, Inc.

NRC7394 Evaluation Kit User Guide (Standalone)
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 NRC7394 Evaluation kit (EVK). The evaluation kit (EVK) is used to evaluate the performance of the NRC7394 module containing NEWRACOM's IEEE 802.11ah Wi-Fi System on Chip (SoC).
There are 2 types of evaluation boards named after modules - NRM7394 and WHM

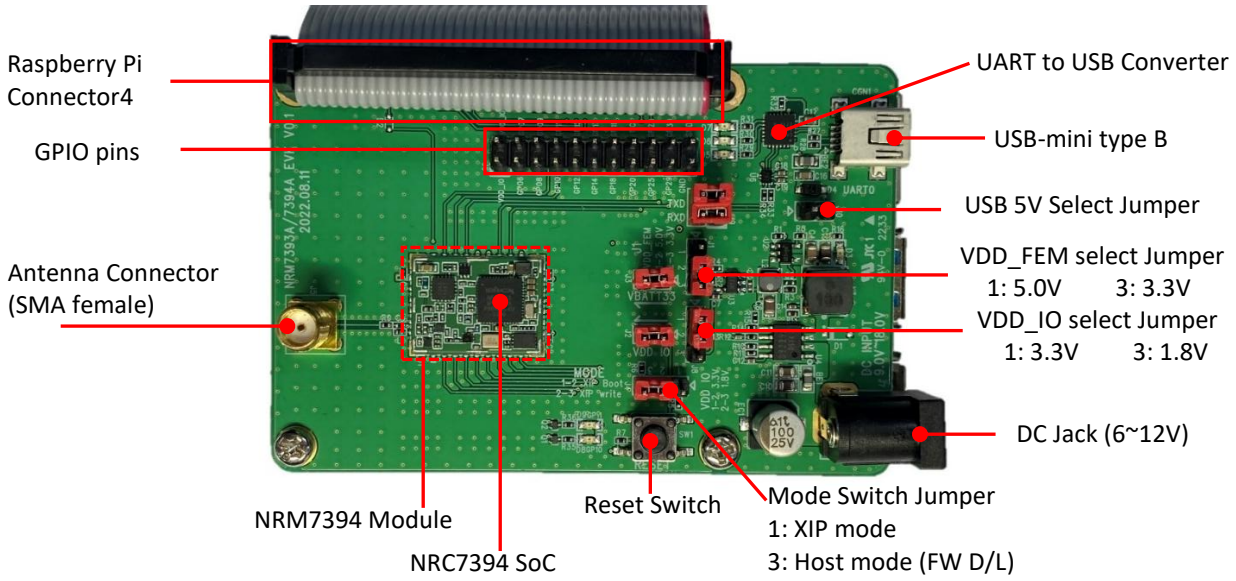


Figure 1.1 NRM7394 evaluation board (Top view)

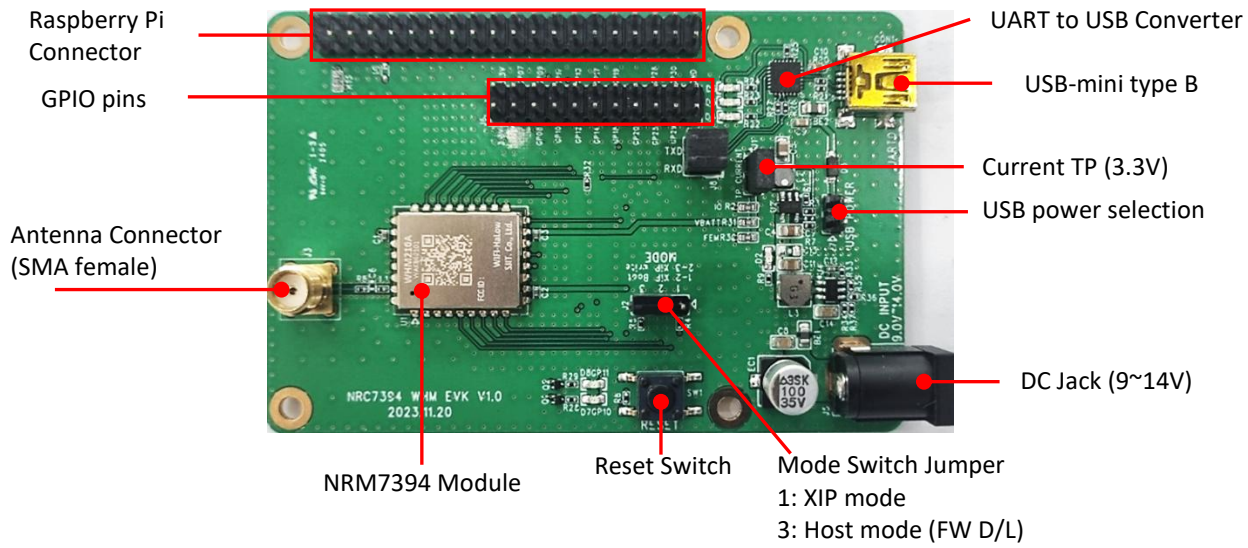


Figure 1.2 WHM evaluation board (Top view)

1.1 HW list

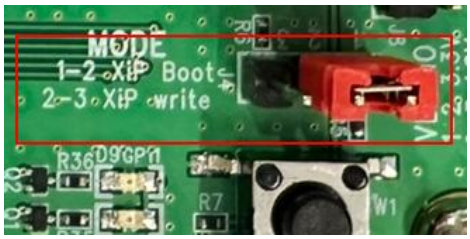
1.1.1 NRC7394 module

NRC7394 module contains IEEE 802.11ah Wi-Fi SoC solution. It also includes a RF front end module (FEM) to increase transmission power up to +27 dBm. Onboard serial flash memory can be used for over-the-air (OTA) software development and with 32KB cache in the NRC7394 supports the execution in place (XIP) feature.

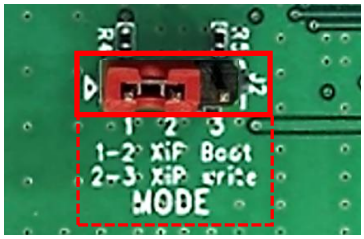
1.1.2 NRC7394 EVB

NRC7394 EVB mainly offers communication interfaces to sensors or an external host.

Configure mode switch to 1-2 (XIP boot) for standalone mode as shown in figures.



NRM7394 EVB



WHM EVB

Figure 1.3 Standalone (XIP) mode configuration

The EVB board has a function of step down the DC input power to each voltage source of the module. 3.3V voltage source of the module has a header type test point for easy current measurement.

Please refer to the Figure 1.4 and Figure 1.5 for the EVB's power structure.

The board has a built-in USB to serial converter and is connected to UART0, so users can check status logs through the onboard USB mini connector.

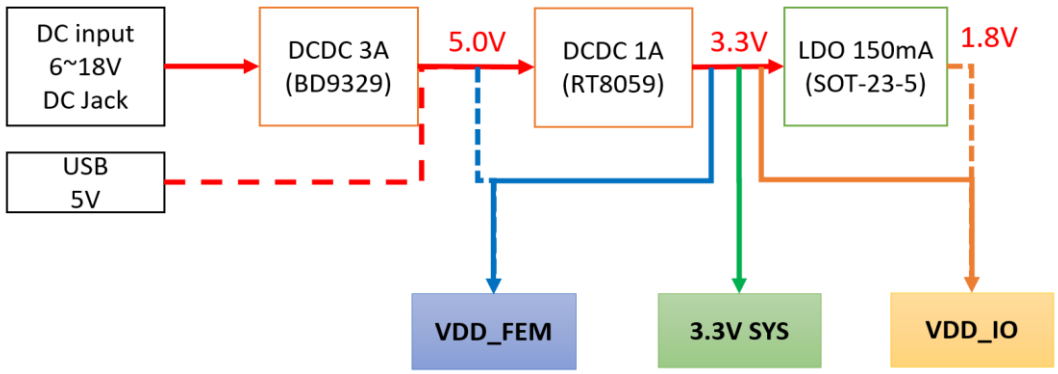


Figure 1.4 NRM7394 EVB Power Structure

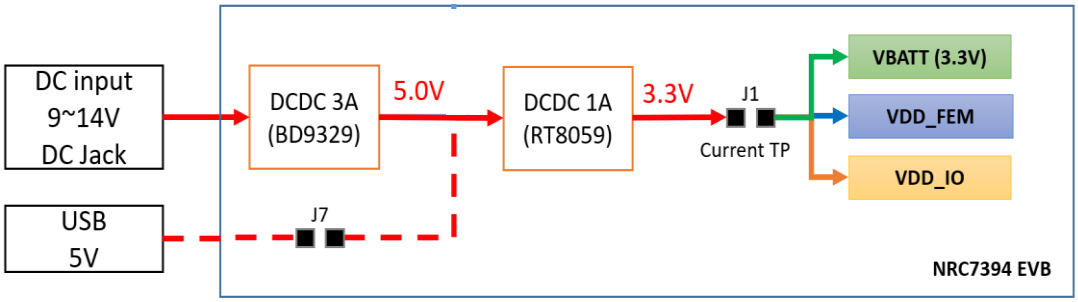


Figure 1.5 WHM EVB Power Structure

1.2 S/W list

1.2.1 NRC7394 SDK

The NRC7394 SDK can be used to develop user's application program running on NRC7394 EVB. The SDK includes various types of Application Program Interfaces (APIs) for controlling Wi-Fi connectivity, Transport Control Protocol/Internet Protocol (TCP/IP) communication, peripherals, timer, etc. In addition, users can attach various sensors on the evaluation board and communicate with them via UART, SPI, or I2C APIs.

As all standalone applications run on FreeRTOS, users can take advantage of FreeRTOS features including multi-tasking, Inter-Task Communication (ITC), memory management, etc.
(Refer to at https://www.freertos.org for more information)

1.3 Kit list

NRC7394 EVK includes:

- NRC7394 EVB (NRM7394 module mounted)
- NRC7394 firmware, Wi-Fi driver, and scripts
- DC 12V (1.5A) power Adaptor
- Antenna (Country frequency)
- USB2.0 mini-B cable
- Raspberry PI4 board (For host mode) with SD card (Linux OS)

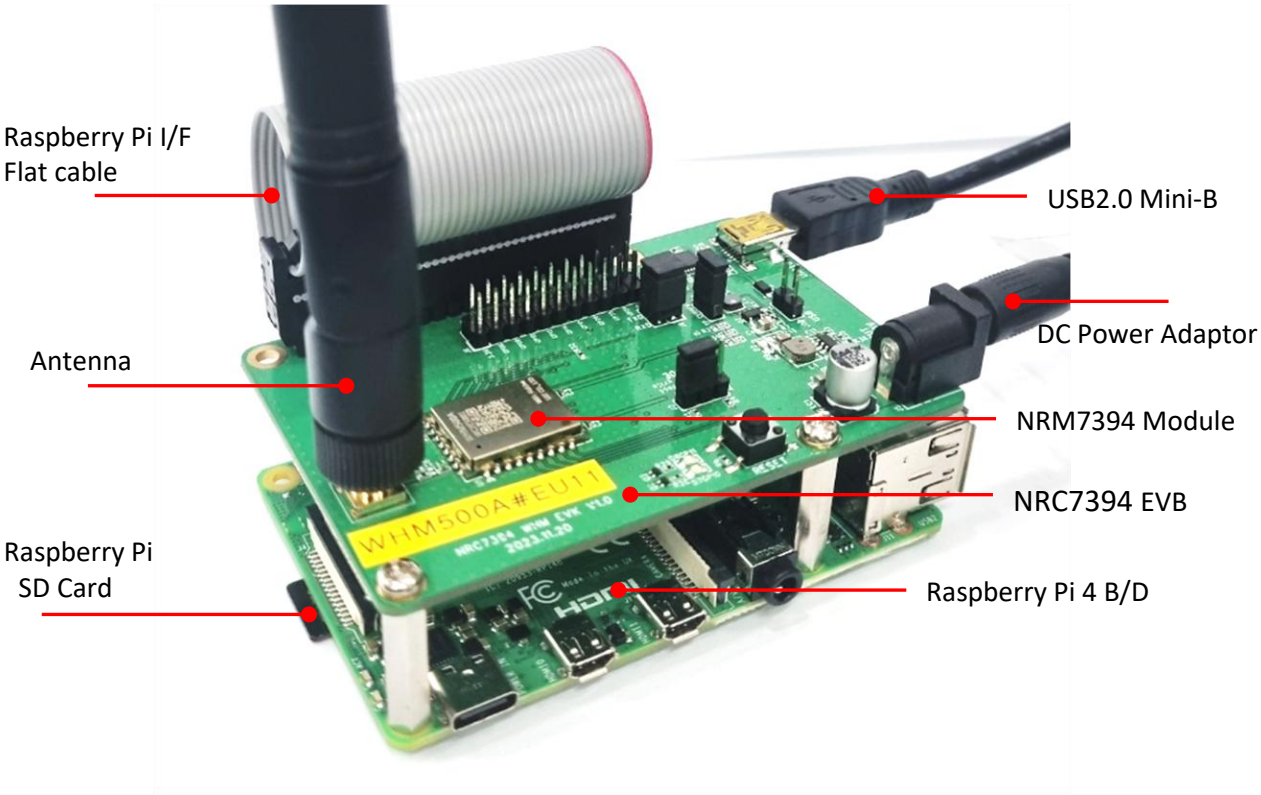


Figure 1.6 NRC7394 hardware set

2 Setup S/W build environment

The NRC7394 SDK supports a Linux environment. This chapter describes how to set up the development environment, build a user's application program, and download the binary on the EVB.

2.1 Toolchain setup

GNU ARM embedded toolchain is required to build the user's application program. Note that users should use 64-bit Linux machine to build successfully.

- Ubuntu 18.04 LTS(64-bit PC(AMD64) desktop image) or later
- GCC toolchain for ARM embedded processors
- Download the GNU Arm embedded toolchain
  - ***gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2*** (version must exactly match) https://developer.arm.com/downloads/-/gnu-rm

Before installing the ARM embedded toolchain, users need some additional packages which can be easily installed through the standard package manager (apt-get) for Ubuntu. The following instructions discuss which packages are required, with instructions on how to install them.

```
sudo apt-get update
sudo apt-get install build-essential python2.7 python-pip git lzop
```

Once the required packages are successfully installed, download the GCC toolchain from the ARM developer website, copy the file to $HOME location, and extract it.

```
tar -xvf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
```

The GCC toolchain will be extracted into ~/ gcc-arm-none-eabi-10.3-2021.10 directory. If the PATH environmental variable is set as shown below, users can run the GCC toolchai n anywhere without giving the complete path for the toolchain.

```
export PATH=$PATH:$HOME/gcc-arm-none-eabi-10.3-2021.10/bin
```

2.2 Download SDK

Users can download the NRC7394 SDK from GitHub (https://github.com/newracom/nrc7394_sdk.git).

```
git clone https://github.com/newracom/nrc7394_sdk.git
```

The NRC7394 SDK in the repository consists of several subdirectories: doc, lib, make, sdk, bdf, and tools. The doc directory contains all documents for the users, including the user guides. The lib directory holds various third-party library codes including FreeRTOS, LwIP, MbedTLS, etc. along with the SDK modem library. The make and apps directories carry makefiles and sample application programs, respectively. The tool holds the NRC7394 Standalone Firmware Downloader, AT command test tool and a firmware Flash Tool. The bdf directory contains a board data files about TX power control. The board data files depend on target hardware and country.

2.3 SDK application program

2.3.1 Sample application programs

The package offers a range of sample application programs located in the 'nrc7394_sdk/package/standalone/sdk/apps' directory. However, it's important to note that the 'wifi_common' directory within 'sdk/apps' does not contain a sample program itself, but rather consists of header and source files for Wi-Fi connectivity. If users intend to utilize Wi-Fi functionalities, they need to include the appropriate header files in their application program. The 'wifi_common' directory serves as an example, demonstrating how developers can implement their own Wi-Fi operations within their applications.

The 'nrc7394_sdk/package/standalone/sdk/inc' directory houses the API header files for various functionalities such as GPIO, I2C, UART, and more. Specifically, this directory includes only the header files for these APIs, which define the functions, constants, and data structures necessary for utilizing and interacting with the respective features. These header files serve as a reference and must be included in the application program to access the corresponding API functionalities.

2.3.2 Application program project structure

Except for the AT-command application, every project directory has the '.config', ‘Makefile’, and ‘nrc_user_config.h’ file in the project. The main source file should contain the ‘void user_init(void)’ function that serves as the entry point of the application.

The main source file should contain the ‘void user_init(void)’ function that serves as the entry point of the application.

```
├──── sample_tcp_client
│     ├────.config
│     ├──── Makefile
```

```
│   ├── wifi_user_config.h
│   ├── sample_tcp_client_version.h
│   └── sample_tcp_client.c
└── wifi_common
    ├── module.mk
    ├── nvs_config.h
    ├── wifi_config.h
    ├── wifi_config_setup.c
    ├── wifi_config_setup.h
    ├── wifi_connect_common.c
    └── wifi_connect_common.h
```

To enable Wi-Fi connectivity example codes, users need to follow the steps below:

- Add the 'include $(SDK_WIFI_COMMON)/module.mk' at the end of the Makefile.
- Add your application source files to the CSRCS variable in the Makefile.

```
CSRCS += \
        sample_tcp_client.c
include $(SDK_WIFI_COMMON)/module.mk
```

To configure the Wi-Fi settings in Wi-Fi connectivity example codes, users should follow the steps outlined below:

- Open the file "wifi_user_config.h" and insert the desired values for the Wi-Fi configuration. These user-defined values should be used instead of configuration options defined in the "wifi_config.h" file.
- Check if there is an existing key in the NVS (Non-Volatile Storage). If a key is found, the corresponding value in the Wi-Fi configuration is replaced with the value stored in the NVS.

Some third-party libraries (CJSON, MQTT, MXML, AWS, etc.) are provided in the package. Users can easily include these libraries by selecting each of them in the '.config' file. For example, if the application requires the MQTT library, the user can simply change ‘n’ to ‘y’ in the '.config' file for use.

```
CONFIG_MQTT    = y
CONFIG_AWS     = n
.............
```

2.3.3 Build application program

Users can use the 'make' command at the standalone directory (nrc7394_sdk/package/standalone) to build the application program. Before running the 'make' command, however, users must create the build-target file (.build-target) which specifies the makefile used for build and the name of the application project.

(Ex) build-target file:

```
MAKEFILE = nrc7394.sdk.release
PARAM := -- APP_NAME=sample_tcp_client
```

Users can create the build-target file by following the instruction below at the standalone directory.

Usage of make command for build-target file:

```
make select target=nrc7394.sdk.release APP_NAME=($APP NAME)
```

For the general application programs, users need to give the name of the application project as the APP_NAME. For example, to build a sample TCP client application, users can write a command as shown below.

```
make select target=nrc7394.sdk.release APP_NAME=sample_tcp_client
```

Once the build-target file is created at the standalone directory, users can run the 'make' command at the same directory. This command will generate the map, elf, and unified binary file of the application program at the 'out/nrc7394/standalone_xip/{project_name}' directory.

The binary file 'nrc7394_standalone_xip_{project_name}.bin' can then be downloaded onto the module.

Build of ATCMD binary:

For AT-command application programs, the pre-defined name of the application should be provided in the following format.

HSPI mode:

```
make select target=nrc7394.sdk.release APP_NAME=ATCMD_HSPI
```

UART mode (without hardware flow control):

```
make select target=nrc7394.sdk.release APP_NAME=ATCMD_UART
```

UART mode (with hardware flow control):

```
make select target=nrc7394.sdk.release APP_NAME=ATCMD_UART_HFC
```

Usage of custom board data file:

There are two methods available for using custom board data in the NRC7394 SDK:

(1) Overwrite the board data file: By replacing the existing board data file

(ex, the bdf/nrc7394/nrc7394_bd.dat file), you can utilize your own custom board data. This involves replacing the original file with your modified version to ensure the SDK uses the updated board data.

(2) Add ALIAS during target selection

When selecting the target, you have the option to add an ALIAS to the custom board data file by using the ".dat" file extension. By placing the custom file in the same location as the original file, you can specify a different board data file for your specific target. This enables the SDK to identify and utilize your custom board data during the compilation and execution process.

```
make select target=nrc7394.sdk.release+custom_bd.dat APP_NAME=sample_tcp_client
```

Both methods provide options for incorporating your own board data into the NRC73944 SDK, allowing for customization and adaptation to specific hardware configurations.

Using Additional SRAM2 (+extram option):

The NRC7394 provides an additional 192 KB of SRAM2 memory that can be utilized as RAM. To enable this extra memory during build, add the alias +extram to the target when creating the build-target file.

For example:

```
make select target=nrc7394.sdk.release+extram APP_NAME=sample_tcp_client
```

This option allows the application to take advantage of the extended memory region, making more RAM available for complex use cases.

Example of Wi-Fi configuration in sample applications

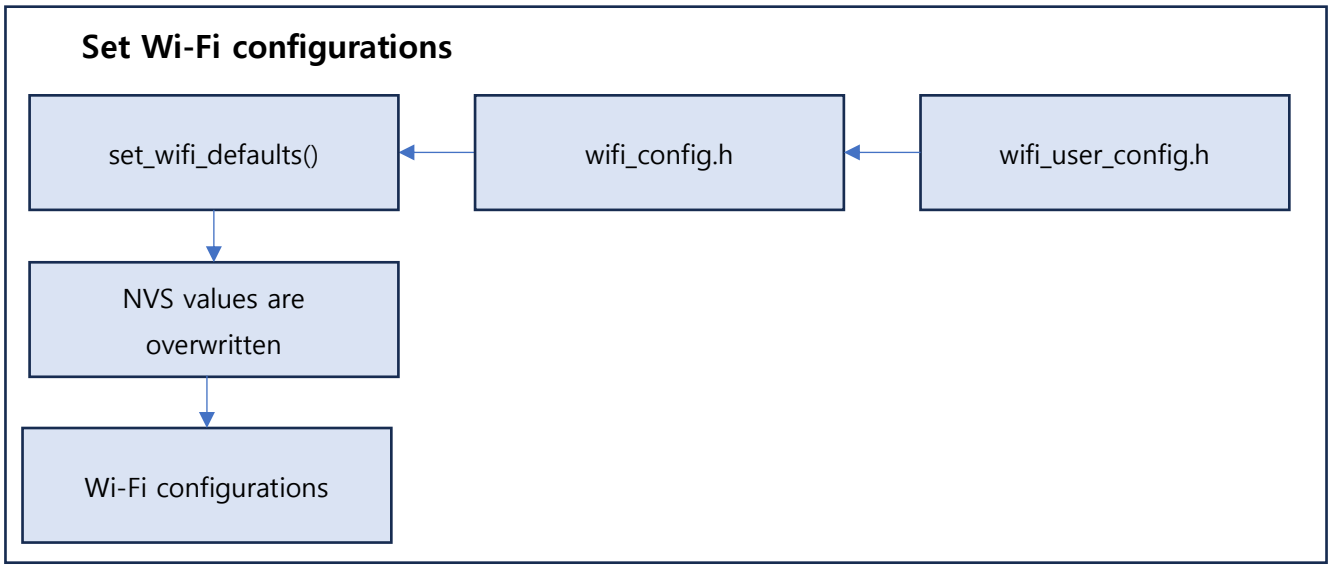
In our code samples, Wi-Fi functionality has been seamlessly integrated through the 'sdk/wifi_common' module. This module demonstrates the utilization of Wi-Fi APIs, serving as an example for developers to employ their preferred methodologies. These APIs enable streamlined Wi-Fi connections and start SoftAP, etc. The configuration procedure for Wi-Fi comprises in below steps:

Configuration within 'wifi_user_config.h':

We prioritize user flexibility by offering comprehensive Wi-Fi configuration options within the 'wifi_user_config.h' file. This file empowers users to tailor various settings to their specific requirements. It's noteworthy that users aren't obligated to define every option; unspecified parameters default to values found in 'wifi_config.h'.

Integration with Non-Volatile Storage (NVS):

When the Non-Volatile Storage (NVS) is utilized, Wi-Fi configurations saved within the NVS environment supersede other configurations. Consequently, values stored in the NVS will take precedence over settings established in both 'wifi_user_config.h' and 'wifi_config.h'. The NVS keys are pre-defined in 'nvs_config.h', and developers also have the option to incorporate their own NVS keys in their application.



2.3.4 SDK APIsQ

Various SDK APIs in several categories are provided for user application programming as shown in Table 2.1. Please refer to UG-7394-005-Standalone SDK API in the packet for more information.

◈*To print out logs via UART console, the debug UART console must first be enabled by calling the API function, "nrc_uart_console_enable()." Once the console is enabled, users can print the logs out to the UART console using the API function, "nrc_usr_print()."*

Table 2.1 NRC7394 SDK APIs

Category	Description
Wi-Fi	Wi-Fi connection
System	System configuration and Log level
UART	UART peripheral I/O
GPIO	GPIO peripheral I/O
I2C	I2C peripheral I/O
ADC	ADC peripheral I/O
PWM	PWM peripheral I/O
SPI	SPI peripheral I/O
HTTP Client	HTTP Client
FOTA	Firmware Over-The-Air
Power Save	Sleep mode (Modem sleep / Deep sleep)
WPS_PBC	WPS pushbutton
Boradcast FOTA	Broadcast Firmware Over-The-Air

2.3.5 Sample applications

Table 2.2 provides the information of the various sample application programs included in the release package.

Table 2.2 Sample applications

Category	Name	Description
Helloworld	hello_world	Repeatedly print hello message
AT CMD	atcmd	AT commands
Wi-Fi	sample_wifi_state	Repeat Wi-Fi connection and disconnection every 3 seconds
	sample_w5500_eth	The Ethernet bridge mode using the W5500 Ethernet controller (spi)
	sample_w5500_nat	The Ethernet Network Address Translation (NAT) mode using the W5500 Ethernet controller (spi)
	sample_softap_udp_server	Run SoftAP and receive UDP data
	sample_softap_tcp_server	Run SoftAP and receive TCP data
	sample_fota	Run FOTA operation
	sample_wifi_relay	Runs a wifi relay
	sample_wifi_roaming	Enable background scan for wifi roaming
	sample_wps_pbc_sta	WPS PBC for STA
	sample_wps_pbc_softap	WPS PBC for softAP
Protocol	sample_udp_client	Send UDP packets
	sample_udp_server	Receive UDP packets
	sample_tcp_client	Connects to a TCP server and sends packets to it.
	sample_tcp_server	Starts a TCP server, waits for an incoming TCP client connection and receives data from the connected client.
Power save	sample_ps_standalone	Deep sleep operation(i2c)
	sample_ps_schedule	Wakes up at set intervals and transmits sensor data.
	sample_nontim_tcp_client	NonTIM mode deep sleep periodically wakes up to transmit TCP data.
	sample_ps_udp	NonTIM mode deep sleep periodically wakes up to transmit UDP data.
Pheripherals	sample_gpio	LED is blinking on board
	sample_uart	Bytes fed into UART CH2
	sample_adc	Communicate with a sensor via ADC
	sample_nvs	Use NVS(Non-volatile Storage) library
	sample_pwm	Enable PWM and configure the PWM duty cycle
	sample_bme680_sensor	Temperature sensor(spi/i2c)
	sample_sgp30_sensor	Air quality sensor(i2c)
	sample_sht30_sensor	Humidity sensor(i2c)
	sample_timer	Hardware Timer
	sample_spi_slave	SPI slave
Middleware	sample_xml	Test XML creation and conversion behavior
	sample_json	Test JSON creation and conversion behavior

	sample_aws_iot_sensor	Connects to AWS (Amazon Web Service) and publishes a message.
	Sample_mqtt	Sends data to an MQTT server using the MQTT protocol.
	sample_http	Sends an HTTP request and receives the corresponding response.
	sample_http_server	Runs a SoftAP with an embedded HTTP server
	sample_sntp	Runs sntp and get UTC
User Scenarios	sample_softap_uart_tcp_server	Runs a TCP server in SoftAP mode and facilitates sending/receiving data from UART.
	sample_uart_tcp_client	Runs a TCP client and facilitates sending/receiving data from UART.
	sample_vendor_ie	Get the vendor ie command and data
	sample_cmd_user	Add console command for user application
	sample_arducam	Camera sample using arducam
	sample_websocket_client	Websocke client

3 How to download compiled binaries

The NRC7394 Standalone Firmware Downloader in the ‘tool’ directory can be used to download the unified binary onto the EVB. The steps outlined below explain how to download the binary.

3.1 UART connection between PC and EVB

Connect the PC to the EVB using a UART-USB cable and check the corresponding COM port number using the Device Manager. The COM port number will be required in the next step.

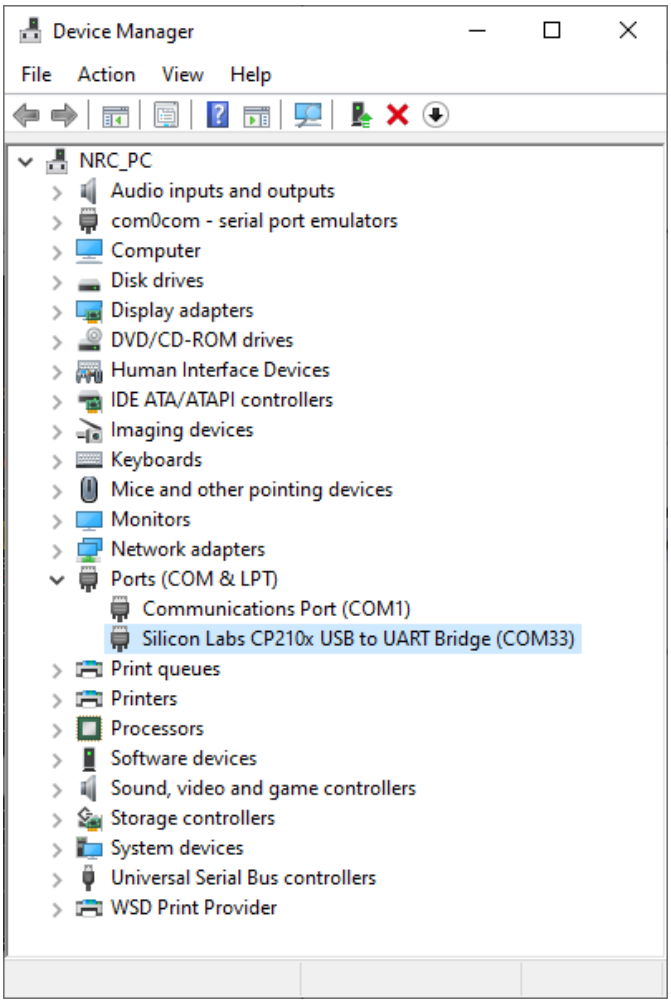


Figure 3.1 COM port in Device Manager

3.2 Upload the unified binary

Launch the NRC7394 Standalone Firmware flash tool and select the correct serial port. Either directly type in the path to the standalone XIP boot and firmware binary or press the '**SET**' button to launch the file selector. The initial bootloader and XIP Boot is located in './firmware/' folder and assigned path automatically. So, the developer does not need to change boot path. MAC addresses (for WLAN0 and WLAN1) can be read from the flash.

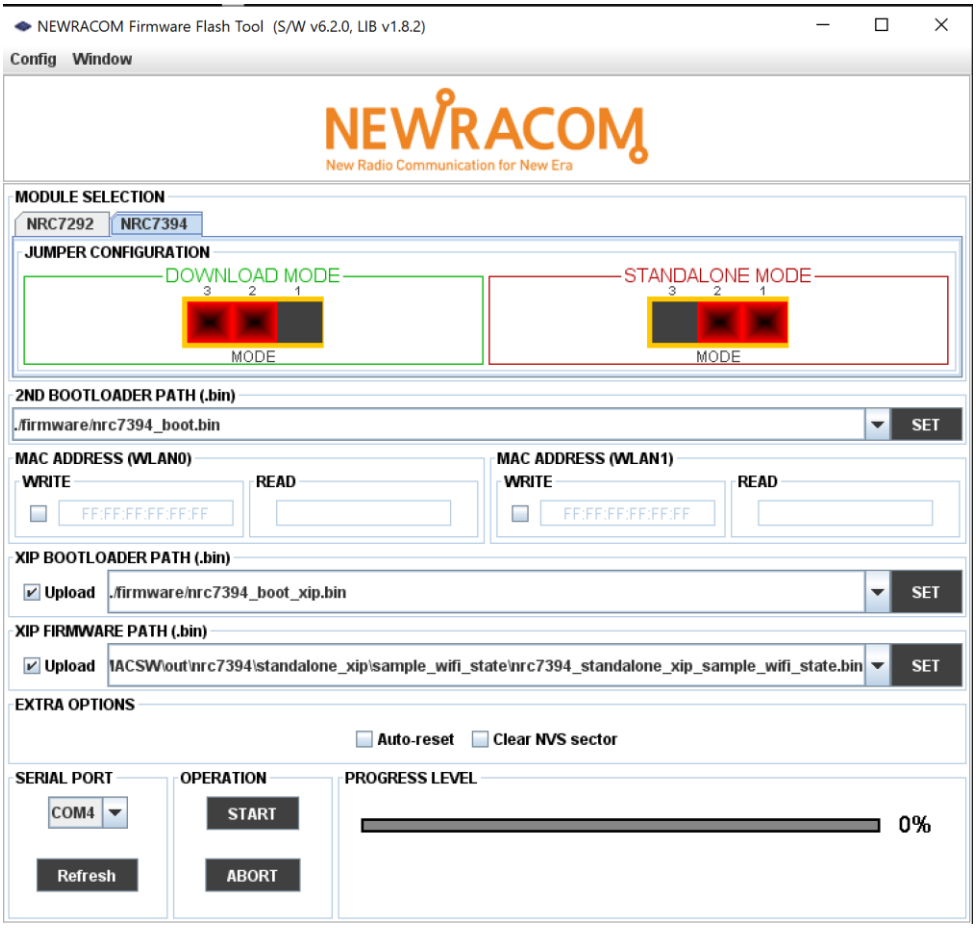


Figure 3.2 Standalone firmware downloader

Before initiating the firmware download process, it is necessary for the developer to change the DIP switch mode to DOWNLOAD MODE by setting it to 'HHHLLH'. To download a unified binary onto the flash memory of the NRC7394 EVB, you can use the NRC7394 Standalone Firmware Downloader, which is included in the release package. For detailed instructions on using the downloader, refer to the guide document located in the 'tools/external/docs/index.html' folder.

There are two types of XIP boot binaries available: the normal boot binary and the 5-waits binary. The normal boot binary operates as expected but requires manual switching of the DIP switch between download mode and standalone mode. On the other hand, the 5-waits binary is specifically designed for use with the standalone mode DIP switch setting. When using the 5-waits binary, it remains in download mode for 5 seconds automatically, eliminating the need for manual switching between download mode and standalone mode. This simplifies the process and allows for a smoother firmware update experience.

The firmware flash tool offers extra features, including the 'Clear NVS sector' and 'Auto-reset' options. Enabling the 'Clear NVS sector' option will erase the NVS (Non-Volatile Storage) data. This option is only checked when the Wi-Fi configuration is modified. On the other hand, selecting the 'Auto-reset' option will cause the firmware to automatically restart once the firmware upload process is finished.

To initiate the download of the chosen binary, click the 'START' button.

4 Performance Evaluation

Iperf is a tool used for network performance measurement and optimization. It offers TCP/UDP client and server functionalities, allowing data streams to be generated to assess the throughput between endpoints. The NRC7394 standalone release package includes programs for Iperf TCP/UDP client and server, leveraging the SDK APIs and LwIP sockets for performance evaluation. The package also enables console commands, enabling developers to conveniently test network performance using the iperf command.

4.1 Preparation of test binary

Test binary for iperf testing could be built in below and download the 'nrc7394_standalone_xip_.bin' in the 'out/nrc7394/standalone_xip/standalone/' folder.

```
make select target=nrc7394.sdk.release
make clean
make
```

4.2 Console command

The console command could be used for Wi-Fi connection and applications such as DHCP client, ifconfig, iperf and ping.

4.2.1 WPA

The wpa cli command is supported. Instead of 'wpa_cli', we use the 'wpa'. The common comands are supported for wifi connection. The command could run such as 'wpa [command] [args]'.

Command	Args	Description
wpa scan		request new BSS scan
wpa scan_results		get latest scan results
wpa add_network		add a network
wpa set_network	<network id> <variable> <value>	set network variables
wpa enable_network	<network id>	enable a network
wpa set country	<country>	set country

(Open Mode)

```
wpa set country US
wpa scan
wpa scan_results
wpa add_network
wpa set_network 0 ssid "AP_SSID"
wpa set_network 0 key_mgmt NONE
wpa enable_network 0
```

(WPA2 Mode)

```
wpa set country US
wpa scan
wpa scan_results
wpa add_network
wpa set_network 0 ssid "AP_SSID"
wpa set_network 0 key_mgmt WPA-PSK
wpa set_network 0 psk "PASSWORD"
wpa enable_network 0
```

(WPA2 Mode)

```
wpa set country US
wpa scan
wpa scan_results
wpa add_network
wpa set_network 0 ssid "AP_SSID"
wpa set_network 0 proto RSN
wpa set_network 0 ieee80211w 2
wpa set_network 0 key_mgmt SAE
wpa set_network 0 sae_password "12345678"
wpa enable_network 0
```

(WPA3-OWE Mode)

```
wpa set country US
wpa scan
wpa scan_results
wpa add_network
wpa set_network 0 ssid "AP_SSID"
wpa set_network 0 proto RSN
wpa set_network 0 ieee80211w 2
wpa set_network 0 key_mgmt OWE
wpa set_network 0 owe_only 0
wpa enable_network 0
```

4.2.2 DHCP

The dhcp command is used for getting IP via DHCP client from DHCP server.

Command	Args	Description
dhcp		request ip address

4.2.3 ifconfig

The Ifconfig is used to configure network interfaces. If no arguments are given, ifconfig displays the status of the currently active interfaces.

Command	args	description
ifconfig	ifconfig <interface> <address> [Options] * [options] -n : netmask -g : gateway -m : MTU size -d : dns1 dns2	Display and configure network interface

4.2.4 IPERF

The iperf command for testing throughput. This application based on only iperf, not iperf3. It supports some madatory options.

Command	args	description
iperf	[-s|-c host] [options] * [options] -b : bandwidth -p : port -t : time -i : interval ※for stopping iperf based on [-s|-c host] [options] (ex) For stopping the operation, please us 'stop' in below [Start UDP server] iperf -s <host> -u [Stop UDP server] iperf -s <host> -u stop	iperf tcp/udp server&client.

4.2.5 PING

The ping command is used for testing connection.

Command	args	description
ping	-s : symbol size -c: ping number -t: ping time	send ICMP packet for testing connection

5 Abbreviations and acronyms

Abbreviations Acronyms	Definition
ADC	Analog Digital Converter
AP	Access Point
API	Application Program Interface
AWS	Amazon Wed Service
CJSON	C JavaScript Object Notation
EVB	Evaluation Board
EVK	Evaluation Kit
FEM	Front End Module
FOTA	Firmware Over the Air
GPIO	General Purpose Input Output
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITC	Inter-Task Communication
I2C	Inter-Integrated Circuit
LAN	Local Area Network
LwIP	Lightweight Internet Protocol
LED	Light Emitting Diode
MQTT	Message Queuing Telemetry Transport
MXML	Music Extensible Markup Language
OTA	Over-the-Air
PWM	Pulse Width Modulation
RPi3	Raspberry Pi 3
RTOS	Real Time Operating System
SDK	Software Development Kit
SoC	System on Chip
SPI	Serial Peripheral Interface
STA	Station
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receive Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
XIP	eXecution In Place

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 standalone 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
```

A.3.3 Notice for SAE H2E (Hash-to-Element) support

The following sae_pwe=1 of hostap configuration file is used for SAE H2E, but NRC7292 standalone mode does not support this. So, it's required to undefine sae_pwe parameter.

```
# SAE mechanism for PWE derivation
# 0 = hunting-and-pecking loop only (default without password identifier)
# 1 = hash-to-element only (default with password identifier)
# 2 = both hunting-and-pecking loop and hash-to-element enabled
# Note: The default value is likely to change from 0 to 2 once the new
# hash-to-element mechanism has received more interoperability testing.
# When using SAE password identifier, the hash-to-element mechanism is used
# regardless of the sae_pwe parameter value.
#sae_pwe=1
```

6 Revision history

Revision No	Date	Comments
Ver 1.0	7/17/2023	Initial version
Ver 1.1	11/28/2023	Add example of wifi configuration settings Add upgrade wpa_supplicant Update sample application table Removed samples: sample_epd_2in66b, sample_hink_e116a07, sample_ssd1306, sample_xa1110_gps, sample_user_factory, sample_aws_switch, sample_cmd_user Added samples: sample_sntp, sample_vendor_ie, sample_wifi_relay
Ver 1.2	7/10/2024	Added WHM module Evaluation board
Ver 1.3	11/22/2024	Update sample application table Added samples: sample_wifi_roaming, sample_wps_pbc_sta, sample_ps_udp sample_wps_pbc_softap, sample_cmd_user, sample_arducam, sample_timer, sample_spi_slave, sample_websocket_client
Ver 1.3.1	8/4/2025	Added Appendix B (GPIO Pin Classification)
Ver 1.3.2	9/25/2025	Added Using Additional SRAM2 (+extram option)

Appendix B. GPIO Pin Classification

B.1 Overview

This appendix provides a detailed classification of GPIO pins based on their usage, configurability, and assignment across various operational modes. Pins are categorized using color-coded designations to represent user accessibility, hardware assignment, and system-level constraints.

Name	Standalone Mode	ATCMD UART Mode	ATCMD UART Mode (+Hardware Flow Control)	Host Mode
GP00	XIP_CLK	XIP_CLK	XIP_CLK	Unused
GP01	XIP_MOSI	XIP_MOSI	XIP_MOSI	Unused
GP02	XIP_MISO	XIP_MISO	XIP_MISO	Unused
GP03	XIP_WP_B	XIP_WP_B	XIP_WP_B	Unused
GP04	XIP_HOLD	XIP_HOLD	XIP_HOLD	Unused
GP05	XIP_nCS	XIP_nCS	XIP_nCS	Unused
GP06	Available	Available	Available	HSPI_MOSI
GP07	Available	Available	Available	HSPI_CLK
GP08	UART0_TXD	UART0_TXD	UART0_TXD	UART0_TXD
GP09	UART0_RXD	UART0_RXD	UART0_RXD	UART0_RXD
GP10	TMS/SWD_IO	TMS/SWD_IO	TMS/SWD_IO	TMS/SWD_IO
GP11	TCK/SWD_CLK	TCK/SWD_CLK	TCK/SWD_CLK	TCK/SWD_CLK
GP12	TDO/UART1_TXD	TDO/UART1_TXD	TDO/UART1_TXD	TDO
GP13	TDI/UART1_RXD	TDI/UART1_RXD	TDI/UART1_RXD	TDI
GP14	nTRST/UART1_CTS	nTRST/UART1_CTS	nTRST/UART1_CTS	nTRST
GP15	ANT_SEL (Default)	ANT_SEL (Default)	ANT_SEL (Default)	ANT_SEL (Default)
GP16	POWER_DOWN (Default)	POWER_DOWN (Default)	POWER_DOWN (Default)	POWER_DOWN (Default)
GP17	Available or ADC0 / TX_ON_MONITORING*	Available or ADC0 / TX_ON_MONITORING*	Available or ADC0 / TX_ON_MONITORING*	TX_ON_MONITORING*
GP18	ADC1	ADC1	ADC1	Unused
GP19	Available	Available	Available	Unused
GP20	UART1_RTS	UART1_RTS	UART1_RTS	Unused
GP21	Does not exist	Does not exist	Does not exist	Does not exist
GP22	Does not exist	Does not exist	Does not exist	Does not exist
GP23	Does not exist	Does not exist	Does not exist	Does not exist

GP24	Available or PA_EN (Default)*	Available or PA_EN (Default)*	Available or PA_EN (Default)*	PA_EN (Default)*
GP25	Available	Available	Available	Unused
GP26	Does not exist	Does not exist	Does not exist	Unused
GP27	Does not exist	Does not exist	Does not exist	Unused
GP28	Available	Available	Available	HSPI_CS
GP29	Available	Available	Available	HSPI_MISO
GP30	Available	Available	Available	HSPI_EIRQ
GP31	Does not exist	Does not exist	Does not exist	Does not exist

B.2 Usage Considerations

- Green (User Configurable):

  - GPIOs freely available for application-level use.

  - No restrictions apply unless otherwise noted for specific pins.

- Yellow (RF FEM / System Assigned):

  - Reserved for RF front-end functions (e.g., ANT_SEL, POWER_DOWN).

  - Configurable **only during SYSCONFIG** (i.e., at factory initialization).

  - ***Not available to the user at runtime.****

    : GP15 should not be assigned for any other purpose because it is used for the ANT_SEL.

    : GP16 should not be assigned for any other purpose because it is used for the power supply of the RF front-end switch.

    : GP17 is only configurable if TX_ON_MONITORING is not enabled in SYSCONFIG.

- Red (Fixed or Non-existent):

  - Either physically non-existent or fixed to essential internal hardware functions.

  - ***Not configurable at any level**, including SYSCONFIG.*

- Purple (XIP/UART Reserved):

  - Dedicated to system boot operations, debug UART, or XIP flash interface.

  - Must remain untouched in user applications.