UM12035

Getting Started with RW61x Running Zephyr OS

Rev. 2.0 — 19 March 2025 **User manual**

Document information

Information	Content
Keywords	Zephyr, operating system (OS), Windows, Linux, RW61x, Bluetooth Low Energy (LE), Wi-Fi, scan, connect
Abstract	Provides the steps to bring up RW61x with Zephyr on Windows and Linux.

1 Introduction

This document explains how to bring up RW61x with Zephyr. Zephyr is a lightweight real-time operating system designed for resource-constrained and embedded systems. Zephyr allows for multi-threading, interrupt, memory allocation, inter-thread synchronization, inter-thread data passing, and power management services. To learn more about Zephyr, refer to [2].

Note: *The setup of the environment and the installation of the Zephyr SDK are different on Linux and Windows. But the commands using the command line interface (CLI) are the same for Linux and Windows.*

2 Prerequisites

2.1 Hardware

- RD-RW61X-BGA-IPA board
- Micro-USB to USB-A cable
- Linux or Windows PC

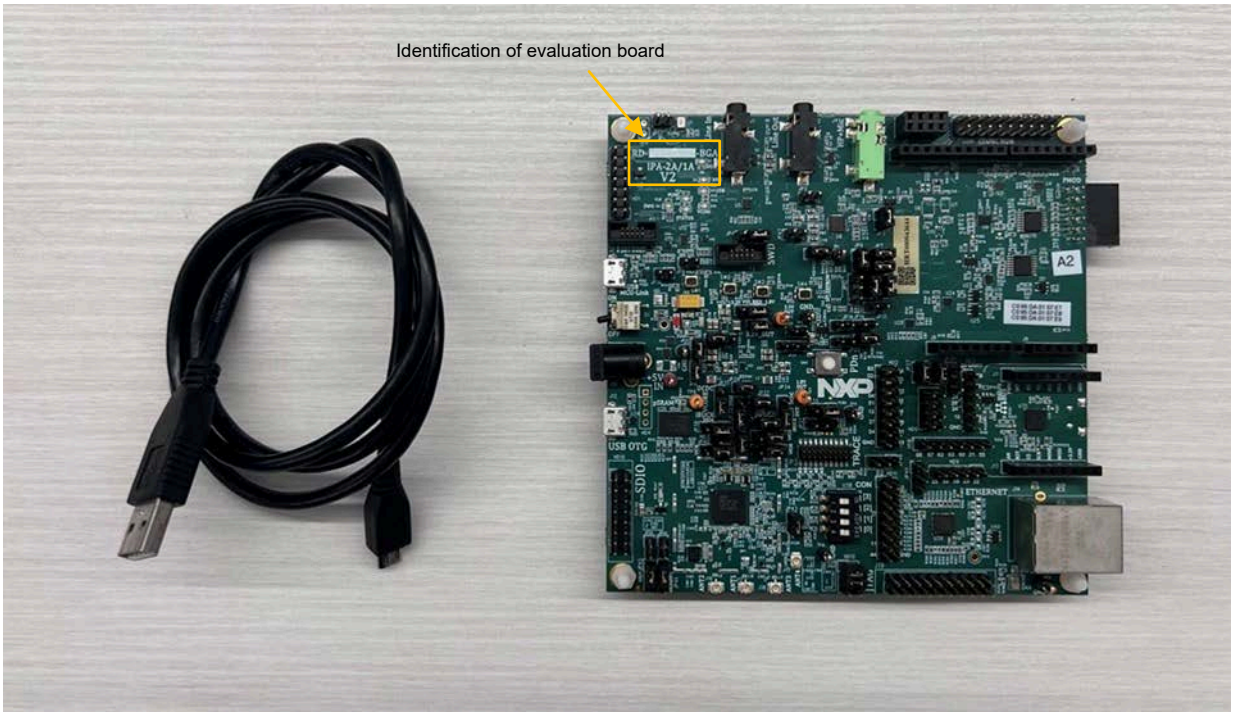


Figure 1. RW61x evaluation board and cable

2.2 Software

2.2.1 Software on Windows

- Visual Studio IDE
  - Refer to Visual Studio Code installation instructions ([3]).
- PIP: Python package manager
  - Refer to pip Installation ([4])
- West tools for Windows

```
pip3 install -U west
```

- MCUXpresso plug-in for Visual Studio Code
  - Open Visual Studio Code and go to Extensions
  - Search for *MCUXpresso for VS Code* and click **Install** (Figure 2)

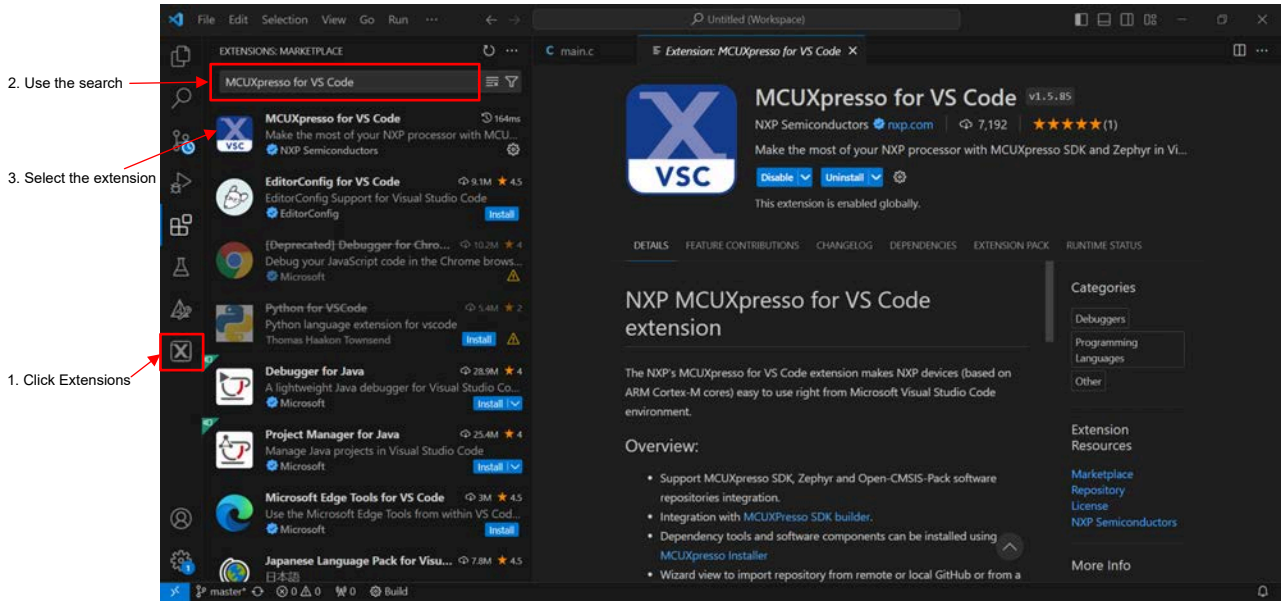


Figure 2. MCUXpresso for VS Code

- Serial console tool for Windows (for example Tera Term)
  - Download Tera Term ([6])

2.2.2 Software on Linux

Table 1 lists the versions required for the main dependencies.

Table 1. Minimum dependency versions required

Tool	Minimum version
CMake	3.20.5
Python	3.8
Device tree compiler	1.4.6

Step 1 - Download and execute the Kitware archive script.

```
#wget https://apt.kitware.com/kitware-archive.sh
#sudo bash kitware-archive.sh
```

Step 2 - Install the required dependencies (Table 1).

```
#sudo apt install --no-install-recommends git cmake ninja-build gperf \
           ccache dfu-util device-tree-compiler wget \
           python3-dev python3-pip python3-setuptools python3-tk python3-wheel xz-utils
 file \
           make gcc gcc-multilib g++-multilib libsdl2-dev libmagic1
```

Step 3 - Verify the versions of the main dependencies.

- Verify CMake version.

```
#cmake –version
```

- Verify Python version.

```
#python3 --version
```

- Verify Device tree compiler version.

```
 #dtc –version
```

- To get a more recent version of dependency, download and install a packaged cmake from the CMake project site.

Note: *This does not uninstall the previous version of cmake.*

```
#cd ~
#wget https://github.com/Kitware/CMake/releases/download/v3.21.1/cmake-3.21.1-Linux-
x86_64.sh
#chmod +x cmake-3.21.1-Linux-x86_64.sh
#sudo ./cmake-3.21.1-Linux-x86_64.sh --skip-license --prefix=/usr/local
#hash -r
```

Note: *The hash -r command is necessary if the installation script put cmake into a new location on your PATH.*

3 Zephyr environment setup

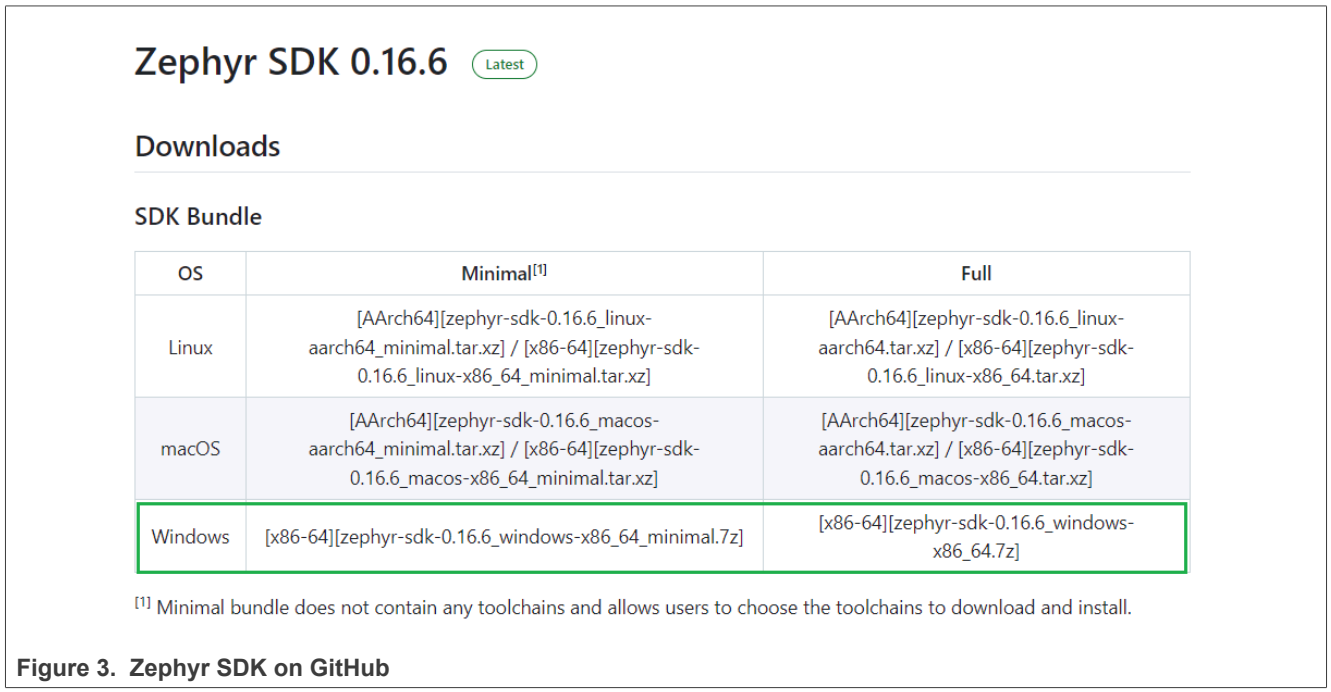
3.1 Setup on Windows

3.1.1 Download Zephyr software development kit (SDK)

This section explains how to download the Zephyr SDK.

Step 1 – Get Zephyr SDK from GitHub ([5]).

Step 2 – Download the SDK bundle for Windows *zephyr-sdk-0.16.X_windows-x86_64.7z*.



Zephyr SDK 0.16.6 Latest

Downloads

SDK Bundle

OS	Minimal[1]	Full
Linux	[AArch64][zephyr-sdk-0.16.6_linux-aarch64_minimal.tar.xz] / [x86-64][zephyr-sdk-0.16.6_linux-x86_64_minimal.tar.xz]	[AArch64][zephyr-sdk-0.16.6_linux-aarch64.tar.xz] / [x86-64][zephyr-sdk-0.16.6_linux-x86_64.tar.xz]
macOS	[AArch64][zephyr-sdk-0.16.6_macos-aarch64_minimal.tar.xz] / [x86-64][zephyr-sdk-0.16.6_macos-x86_64_minimal.tar.xz]	[AArch64][zephyr-sdk-0.16.6_macos-aarch64.tar.xz] / [x86-64][zephyr-sdk-0.16.6_macos-x86_64.tar.xz]
Windows	[x86-64][zephyr-sdk-0.16.6_windows-x86_64_minimal.7z]	[x86-64][zephyr-sdk-0.16.6_windows-x86_64.7z]

[1] Minimal bundle does not contain any toolchains and allows users to choose the toolchains to download and install.

Figure 3. Zephyr SDK on GitHub

Step 3 - Extract the downloaded SDK.

Step 4 – Move to the extracted directory *zephyr-sdk-0.16.X*.

Step 5 - Open a command prompt with administration access for *zephyr-sdk-0.16.X* repository.

Step 6 - Run the setup script.

```
setup.cmd
```

Step 7 – Enter `Y` twice to accept changes.

Example of output:

```
C:\Users\Downloads\zephyr-sdk-0.16.4_windows-x86_64\zephyr-sdk-0.16.4 >setup.cmd
Zephyr SDK 0.16.4 Setup
** NOTE **
You only need to run this script once after extracting the Zephyr SDK
distribution bundle archive.
Install host tools [Y,N]?Y
Register Zephyr SDK CMake package [Y,N]?Y
Installing host tools ...
SKIPPED: Windows host tools are not available yet.
Registering Zephyr SDK CMake package ...
Zephyr-sdk (C:/Users/nxf98556/Downloads/zephyr-sdk-0.16.4_windows-x86_64/zephyr-
sdk-0.16.4/cmake)
has been added to the user package registry in:
HKEY_CURRENT_USER\Software\Kitware\CMake\Packages\Zephyr-sdk
All done.
Press any key to exit ...
```

3.1.2 Build Zephyr workspace

This section explains how to clone zephyr repository and build the image.

Step 1 - Create the workspace and clone the Zephyr repository from a remote URL.

```
west init -m https://github.com/nxp-zephyr-ear/zephyr.git --mr zephyr_rw61x_v3.6_RFP
```

Step 2 - Update the repository.

```
west update
```

Step 3 - Install additional python dependencies from the top-level directory.

```
pip install -r zephyr\scripts\requirements.txt
```

Step 4 - Open Visual Studio Code and select **Import Repository** from Visual Studio Code QUICKSTART PANEL.

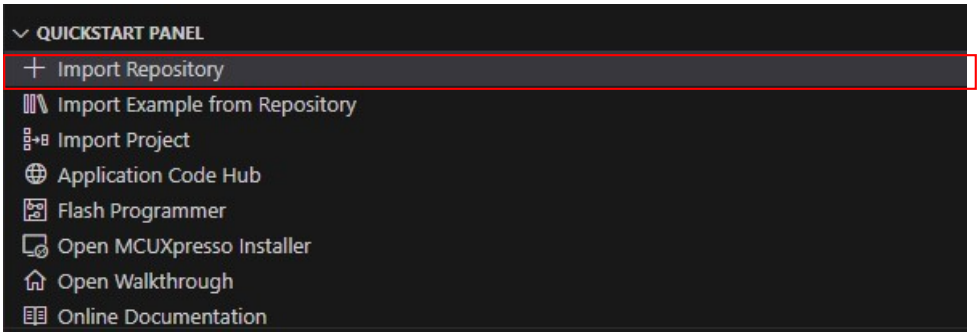


Figure 4. Select Import Repository in QUICKSTART PANEL

Step 5 - Select the **local** location of the cloned Zephyr repository.

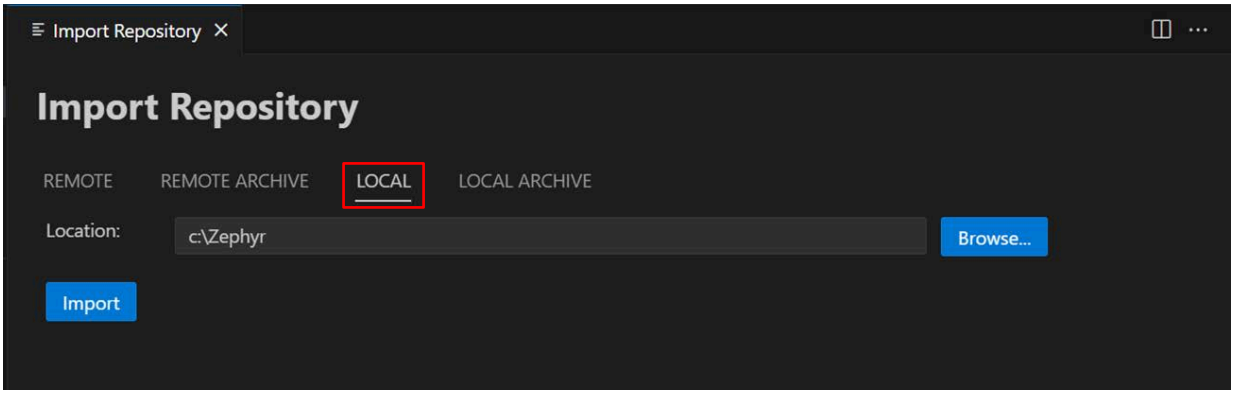


Figure 5. Select the local location

Step 6 – Go to **Import Example from Repository** tab from Visual Studio Code QUICKSTART PANEL.

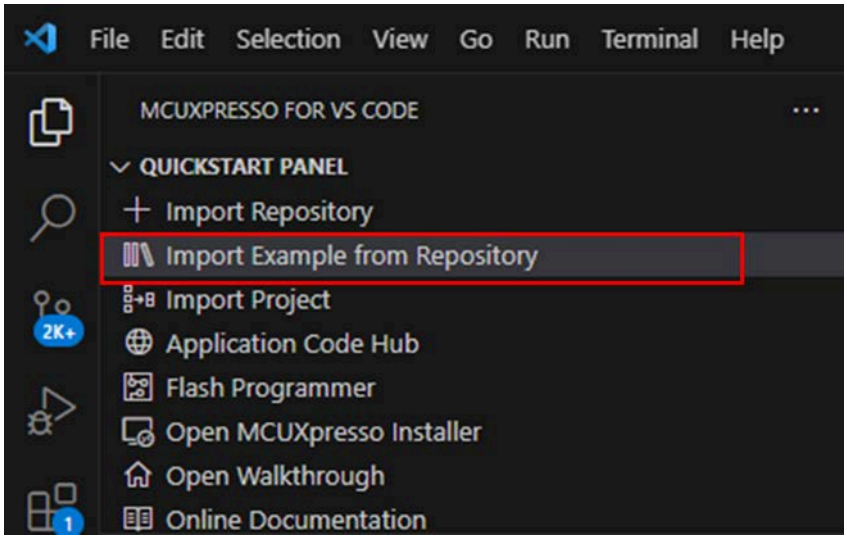


Figure 6. Import Example from Repository

Step 7 - Choose a Zephyr repository, Zephyr SDK, board, and select a sample (Figure 7).

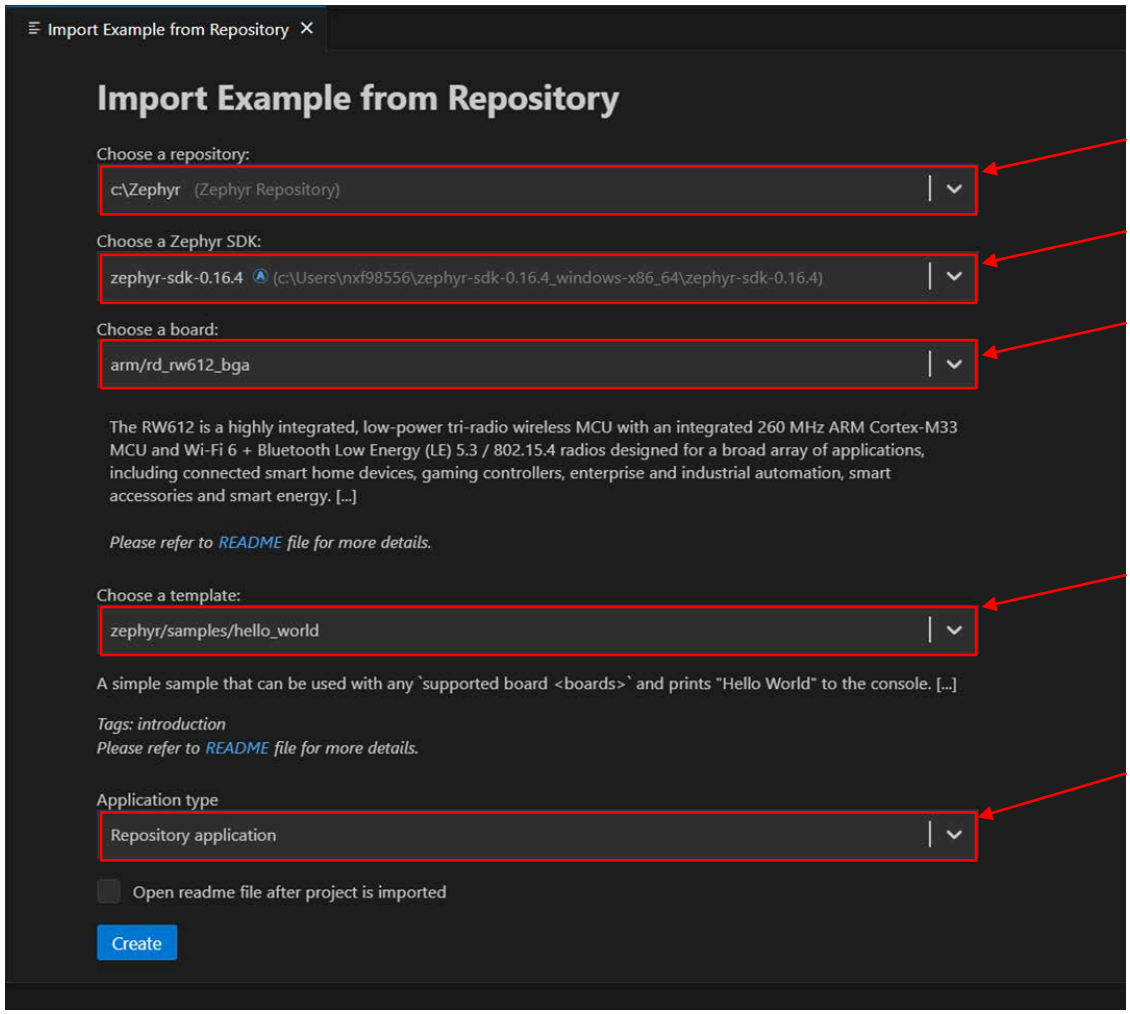


Figure 7. Import *hello_world* example

Step 8 – Click the **build** icon to build the imported example.

Figure 8. Build the imported example

Note: *Section 4.1 explains how to build and flash Sample Hello World. Section 5.1 and Section 6.1 include examples for Wi-Fi and Bluetooth on RW61x.*

3.2 Setup on Linux

3.2.1 Install Zephyr dependencies

Step 1 - Install python venv package.

```
# sudo apt install python3-venv
```

Step 2 - Create a new virtual environment.

```
# python3 -m venv ~/zephyrproject/.venv
```

Step 3 - Activate the virtual environment.

```
# source ~/zephyrproject/.venv/bin/activate
```

Note: *Once activated your shell will be prefixed with (.venv). The virtual environment can be deactivated at any time by running deactivate.*

Remember to activate the virtual environment every time you start working.

Step 4 - Install west.

```
# pip install west
```

Step 5 - Get the Zephyr source code.

```
# west init ~/zephyrproject
# cd ~/zephyrproject
# west update
```

Step 5 - Export a Zephyr CMake package.

```
# west zephyr-export
```

Step 6 - Install the *requirements.txt* file.

```
# pip install -r ~/zephyrproject/zephyr/scripts/requirements.txt
```

3.2.2 Install Zephyr SDK

Step 1 - Download and verify the latest Zephyr SDK bundle.

```
#cd ~
#wget https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.16.1/zephyr-
sdk-0.16.1_linux-x86_64.tar.xz
#wget -O - https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.16.1/
sha256.sum | shasum --check --ignore-missing
```

Step 2 - Extract the Zephyr SDK bundle archive.

```
# tar xvf zephyr-sdk-0.16.1_linux-x86_64.tar.xz
```

Note: *Extract the Zephyr SDK bundle under $HOME, the resulting installation path is $HOME/zephyr-sdk-0.16.1.*

Step 3 - Run the Zephyr SDK bundle setup script

```
#cd zephyr-sdk-0.16.1
#./setup.sh
```

Step 4 - Install udev rules to flash Zephyr boards as regular user.

```
#sudo cp ~/zephyr-sdk-0.16.1/sysroots/x86_64-pokysdk-linux/usr/share/openocd/contrib/60-
openocd.rules /etc/udev/rules.d
#sudo udevadm control --reload
```

4 Hello World examples

4.1 Example on Windows

This section explains how to flash “Hello world” example onto RW61x.

Step 1 – Connect RW61x board and external laptop/PC.

- Plug the USB-A side into the external laptop/PC.
- Plug the micro-USB side into the RW61x’s USB MCU-Link port.

Step 2 – Turn on RW61x.

Step 3 - Select **Flash Programmer** in Visual Studio Code QUICKSTART PANEL.

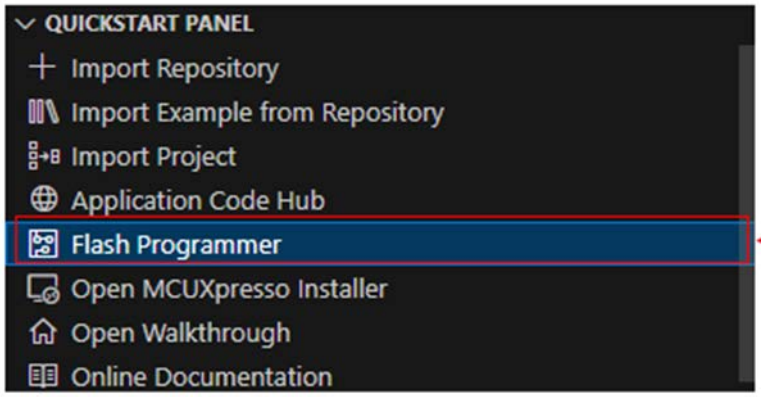


Figure 9. Select Flash Programmer in QUICKSTART PANNEL

Step 4 - Select SEGGER as probe type.

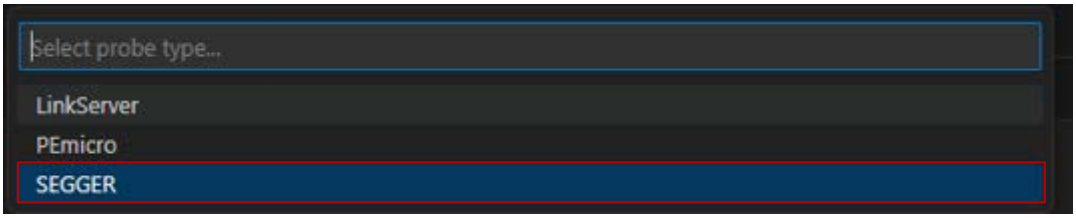
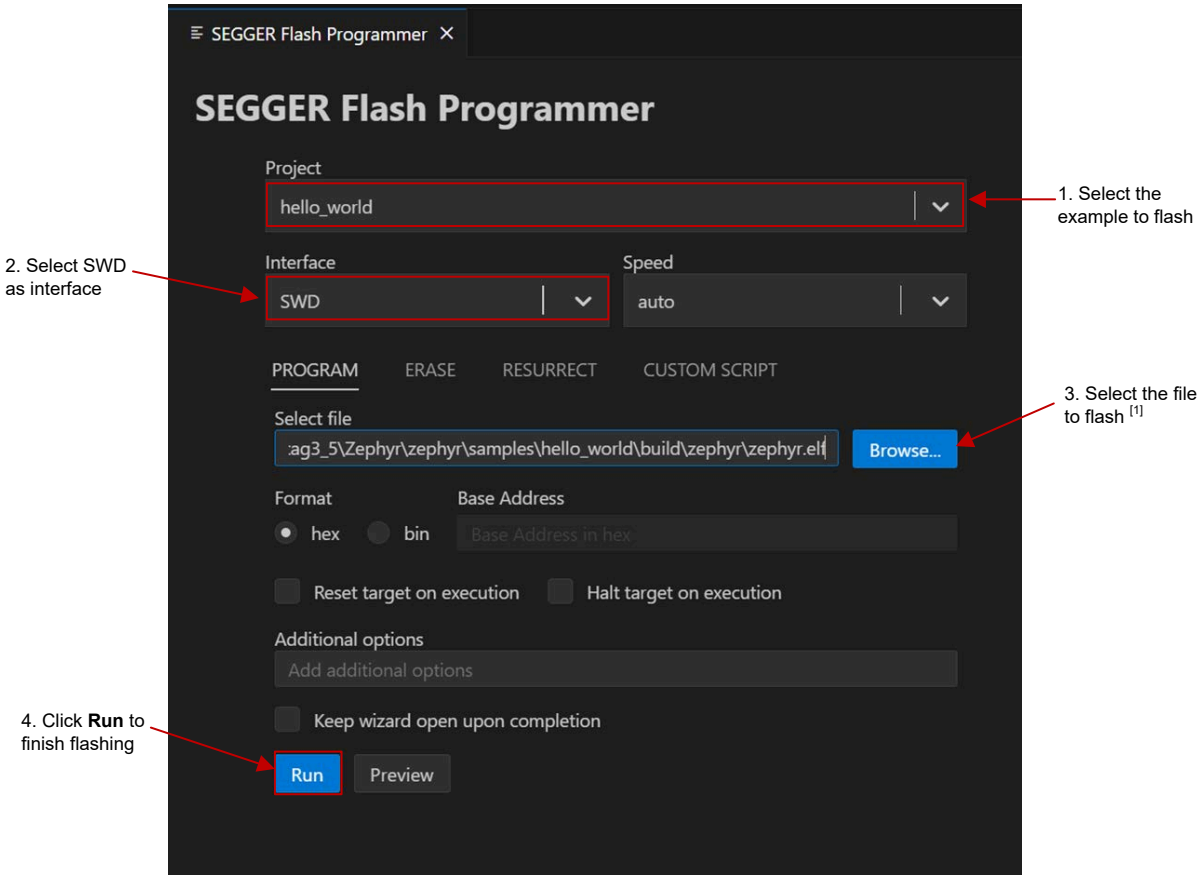


Figure 10. Select SEGGER as probe

Step 5 – Fill the information in **SEGGER Flash Programmer** panel.

Note: *Select an example with .elf extension.*



[1] For example C:\Zephyr\zephyr\samples\hello_world\build\zephyr\zephyr.elf

Figure 11. SEGGER Flash Programmer panel

Step 6 – The image is flashed onto RW61x. To run, the example, open Tera Term or any serial terminal.

Step 7 – Look for the COM port setup information and configure the COM port to a baudrate of 115200. Figure 12 shows the Serial port setup and connection window of Tera Term.

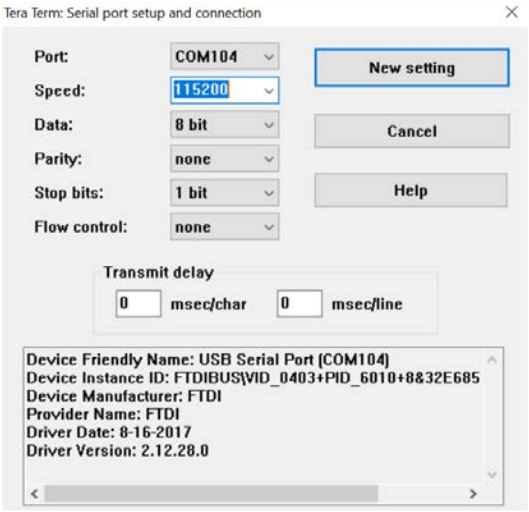


Figure 12. COM port setup information

Step 8 – Power cycle the board and look for the outcome on the terminal.

```
*** Booting Zephyr OS build 264c778fb56c ***
Hello World ! rd_rw612_bga
```

4.2 Example on Linux

Step 1 - Clone the source code.

Syntax:

```
West init <destination_folder_path> -m <github_link>
```

Command for RW61x:

```
# west init code/zephyr_rw610 -m https://github.com/nxp-zephyr-ear/zephyr.git --mr
 zephyr_rw61x_v3.6_RFP
```

Step 2 - Verify the tag name and update code.

- Move to zephyr directory.

```
#cd code/zephyr_rw610/zephyr
```

- Checkout the tag name.

```
git checkout <tag_name>
#git checkout zephyr_rw61x_v3.6_RFP
```

- Update the code.

```
#git pull
#cd ..
#west update
```

Note: *Run the* `west update` *in the directory ~/code/zephyr_rw610/*

Step 5 - Build Hello World example

Syntax:

```
west build -b <BOARD> path/to/source/directory --pristine
```

Command for RW61x:

```
# west build -b rd_rw612_bga samples/hello_world   --pristine
```

5 Wi-Fi examples

5.1 Build Wi-Fi examples on Windows

Follow the steps in Section 3.1.2 and Section 4.1 to build and flash a Wi-Fi application example.

Note: *Read more about Wi-Fi shell on Zephyr website [9].*

Step 1 – Import Wi-Fi application example *wifi*.

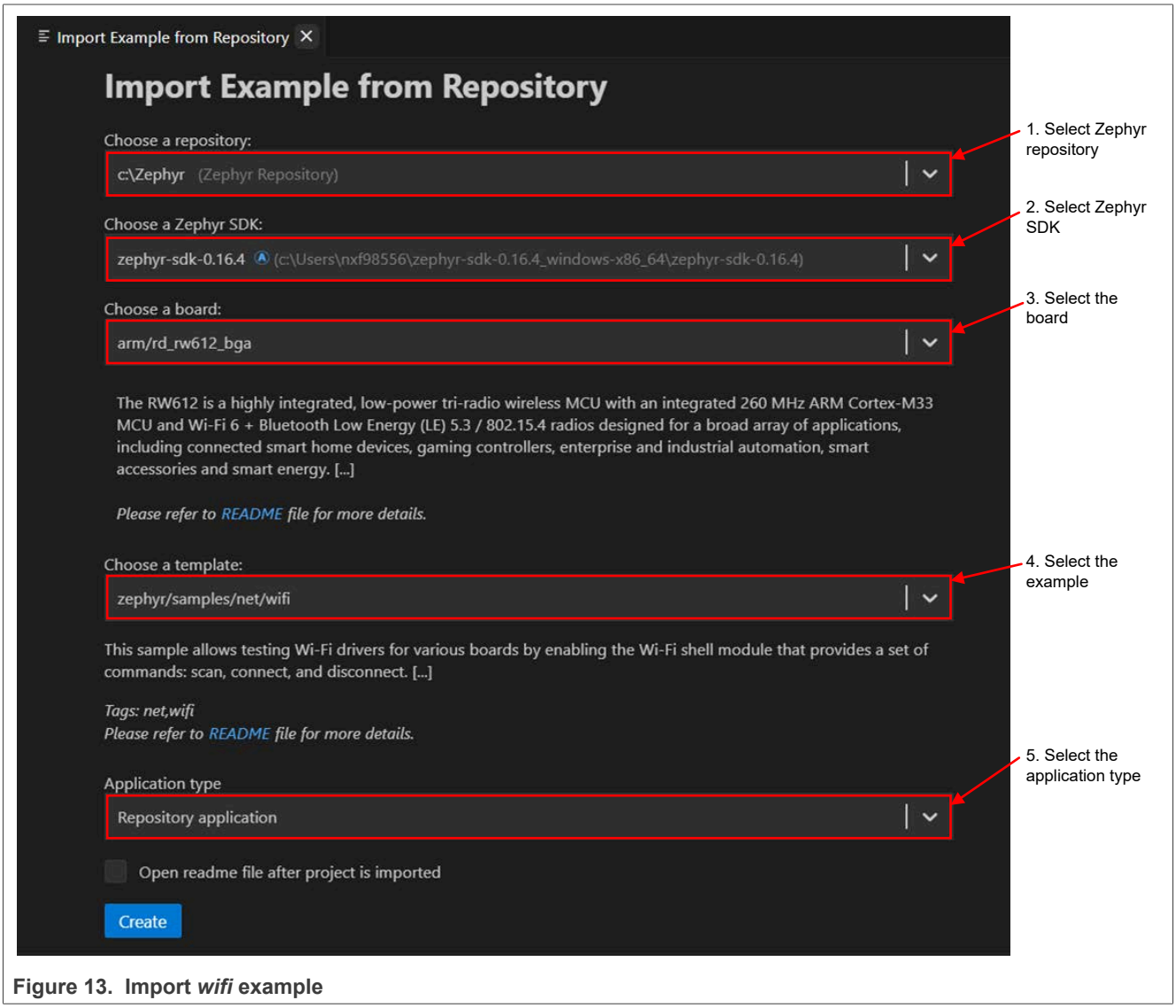


Figure 13. Import *wifi* example

Step 2 – Build the application example.

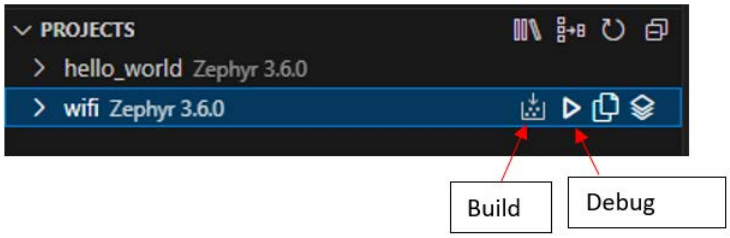


Figure 14. Build *wifi* application

Step 3 – Flash the example application.

- Click the **Debug** icon to launch and flash the application.

Note:

- *When changing the firmware, rename the file with the default firmware name.*
- *Check the imprinted silicon version on the board (Figure 1). To use A1 version of RW61x silicon, define the macro `CONFIG_SOC_SERIES_RW6XX_REVISION_A1=y` in the file samples/net/wifi/boards/rd_rw612_bga.conf.*

Step 4 – Power cycle RW61x.

Step 5 – The image is flashed onto RW61x. To run, the example, open Tera Term or any serial terminal.

Step 6 – Look for the COM port setup information and configure the COM port to a baudrate of 115200. Figure 15 shows the Serial port setup and connection window of Tera Term.

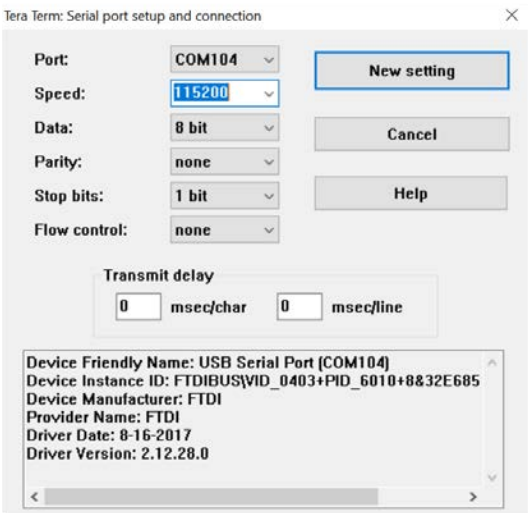


Figure 15. COM port setup information

Step 7 – Power cycle the board and look for the outcome on the terminal.

```
2muart:~$ MAC Address: C0:95:DA:01:20:D1
supplicant_main_task: 298 Starting wpa_supplicant thread with debug level: 3

Successfully initialized wpa_supplicant
Using interface ml

Initializing interface 0: ml

PKG_TYPE: BGA
Set BGA tx power table data
[00:00:01.419,740] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 12
uart:~$ *** Booting Zephyr OS build 264c778fb56c ***
[00:00:01.434,612] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN initialized
[00:00:01.447,332] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 14
[00:00:01.458,074] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: PS_ENTER
[00:00:01.468,091] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 14
[00:00:01.478,913] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: PS_ENTER
```

Step 8 – Issue `help` to get the list of available commands.

```
uart:~$ help
```

Example of command output:

```
Please press the <Tab> button to see all available commands.
You can also use the <Tab> button to prompt or auto-complete all commands or its
 subcommands.
You can try to call commands with <-h> or <--help> parameter for more information.

Shell supports following meta-keys:
Ctrl + (a key from: abcdefklnpuw)
Alt  + (a key from: bf)
Please refer to shell documentation for more details.

Available commands:
clear     : Clear screen.
date      : Date commands
device    : Device commands
devmem    : Read/write physical memory
Usage:
Read memory at address with optional width:
devmem address [width]
Write memory at address with mandatory width and value:
devmem address <width> <value>
help      : Prints the help message.
history   : Command history.
kernel    : Kernel commands
net       : Networking commands
nxp_wifi  : NXP Wi-Fi commands (Use help)
rem       : Ignore lines beginning with 'rem '
resize    : Console gets terminal screen size or assumes default in case the
readout fails. It must be executed after each terminal width change
to ensure correct text display.
retval    : Print return value of most recent command
shell     : Useful, not Unix-like shell commands.
wifi      : Wi-Fi commands
zperf     : Zperf commands
```

Note: *All NXP `wifi_cli` commands can be used adding `nxp_wifi` prefix to `wlan` command. For more information on `wlan` commands, refer to [1].*

5.2 Build Wi-Fi examples on Linux

Command syntax:

```
west build -b <BOARD> path/to/source/directory -d path/to/build/directory
```

Command for RW61x:

```
# west build -b rd_rw612_bga samples/net/wifi -d wifi
# cd ~/code/zephyr_rw610/zephyr/wifi/zephyr/
```

Step 1 - Go to *zephyr* directory to access *zephyr.bin* and *zephyr.elf* binaries.

```
cd ~/code/zephyr_rw610/zephyr/wifi/zephyr/
```

Step 2 - Flash the image using J-Link Commander.

Command syntax:

```
loadbin <path_to_bin_file_in_build_directory> 0x08000000
```

Command for RW61x:

```
loadbin wifi/zephyr/zephyr.bin 0x08000000
```

5.3 CLI commands for Wi-Fi

The commands are the same for Linux and Windows.

5.3.1 Wi-Fi scan

The `wlan-scan` command is used to scan nearby Access Points.

```
nxp_wifi scan
```

Example of output:

```
Scan scheduled...
Command wlan-scan
uart:~$ 10 networks found:
04:42:1A:10:A1:A4  "NBtest_matter_asusax88u_5G" Infra
mode: 802.11AX
channel: 149
rssi: -69 dBm
security: WPA2
WMM: YES
802.11V: YES
802.11W: NA
WPS: YES, Session: Not active
```

5.3.2 WLAN version

`Version` command returns the WLAN driver version and WLAN firmware version.

```
nxp_wifi version
```

Example of output:

```
WLAN Driver Version   : v1.3.r48.p9
WLAN Firmware Version : rw610w-V2, IMU, FP99, 18.99.6.p7.1, PVE_FIX 1
Command wlan-version
```

5.3.3 Wi-Fi client mode (STA)

The `wlan-connect` command is used to connect to an access point (AP).

Follow the steps below to connect to an AP.

Step 1 – Add the AP to the list of known networks.

Command syntax:

```
nxp-wifi add <profile_name> ssid <ssid> [wpa2 <psk/psk-sha256/ft-psk> <secret>]
```

Table 2. `wlan-add` command parameters

Parameter	Description
`profile_name`	Profile name to be added in the list of known networks
`ssid`	AP SSID
`psk/psk-sha256/ft-psk`	Preshared Key, Preshared Key with SHA256 key derivation type, or Preshared Key with Fast Transition
`secret`	AP password

Example of command:

```
nxp_wifi add test1 ssid NXP wpa2 psk UInternet
```

Example of output:

```
Added "test1"
Command nxp-wifi add
```

Step 2 – Connect to the AP

Command syntax:

```
nxp_wifi wlan-connect <profile_name>
```

Example of command:

```
nxp_wifi wlan-connect test1
```

Example of output:

```
Connecting to network...
Use 'wlan-stat' for current connection status.
Command wlan-connect
uart:~$ ml: SME: Trying to authenticate with 3c:51:0e:6f:f3:6e (SSID='NXP' freq=5600 MHz)
ml: Trying to associate with 3c:51:0e:6f:f3:6e (SSID='NXP' freq=5600 MHz)
PKG_TYPE: BGA
Set BGA tx power table data
ml: Associated with 3c:51:0e:6f:f3:6e
ml: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
ml: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
ml: WPA: Key negotiation completed with 3c:51:0e:6f:f3:6e [PTK=CCMP GTK=CCMP]
ml: CTRL-EVENT-CONNECTED - Connection to 3c:51:0e:6f:f3:6e completed [id=0 id_str=]
[00:27:16.043,996] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 1
[00:27:16.054,717] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: authenticated to
 nxp_wlan_network
[00:27:20.440,302] <inf> net_dhcpv4: Received: 192.168.0.217
[00:27:20.448,707] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 0
[00:27:20.459,344] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: connected to
 nxp_wlan_network
[00:27:20.471,232] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: Connected to following
 BSS:
[00:27:20.482,379] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: SSID = [NXP]
[00:27:20.492,588] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: IPv4 Address:
 [192.168.0.217]
[00:27:20.503,954] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: IPv6 Address: Link-Local
   :   fe80::c295:daff:fe01:1fd1 (Preferred)
[00:27:20.518,544] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback:
Connected
```

5.3.4 Wi-Fi micro AP mode (uAP)

This section shows how to bring up a micro AP on RW61x.

Step 1 – Add the AP network configuration details to list of known networks.

Command syntax:

```
nxp-wifi add <profile_name> ssid <ssid>
ip:<ip_addr>,<gateway_ip>,<netmask>
role uap
.
.
.
```

Example of command:

```
nxp_wifi add uapProfile ssid rw61x ip:192.168.0.1,192.168.0.1,255.255.255.0 role uap
```

Example of output:

```
Added "uapProfile"
Command nxp-wofo add
```

Step 2 – Start Micro AP

Command syntax:

```
nxp_wifi start-network <profile_name>
```

Example of command:

```
nxp_wifi start-network uapProfile
```

Example of output:

```
ua: interface state UNINITIALIZED->COUNTRY_UPDATE
ml: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
Command wlan-start-network
uart:~$ ua: interface state COUNTRY_UPDATE->DFS
: DFS-CAC-START freq=5600 chan=120 chan_offset=0 width=1 seg0=5600 seg1=0 cac_time=60s
: DFS-CAC-COMPLETED success=1 freq=5600 ht_enabled=1 chan_offset=0 chan_width=1 cf1=5600
 cf2=0
ua: interface state DFS->ENABLED
: AP-ENABLED
PKG_TYPE: BGA
Set BGA tx power table data
[00:31:08.261,472] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: received event 16
[00:31:08.272,272] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: WLAN: UAP Started
[00:31:08.282,563] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: Soft AP "rw61x" started
 successfully
[00:31:08.294,634] <dbg> nxp_wifi: nxp_wifi_wlan_event_callback: DHCP Server started
 successfully
```

6 Bluetooth LE examples

6.1 Bluetooth LE examples on Windows

Note: *Read me about Bluetooth on Zephyr website [7], [8].*

Follow the same steps as in Section 3.1.2 and Section 4.1 to build and flash a Bluetooth LE example.

Step 1 – Import the Bluetooth LE application example *shell*.

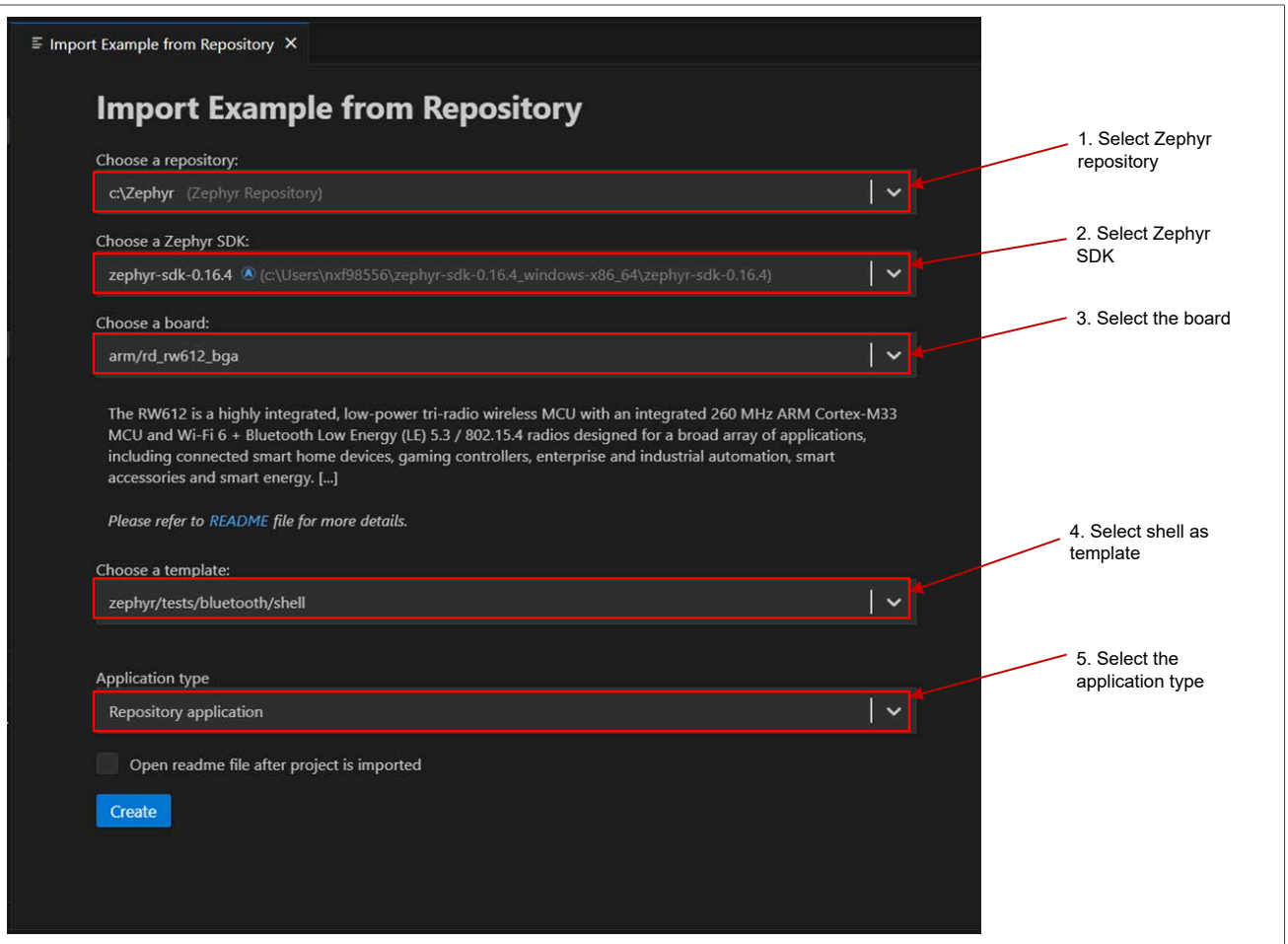


Figure 16. Import *shell* example from the repository

Step 2 – Build the application example.

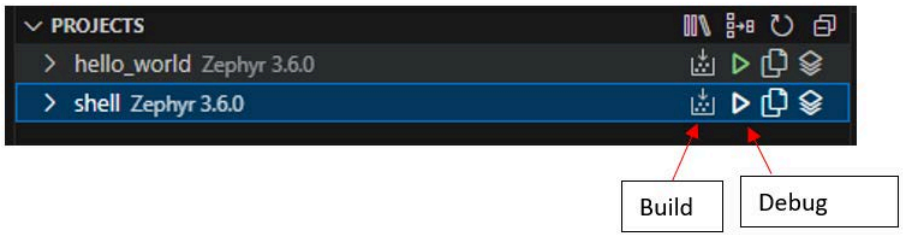


Figure 17. Build *shell* application

Step 3 – Flash the example application.

- Click **Debug** icon to launch and flash the application.

Note:

- *When changing the firmware, rename the file with the default firmware name.*
- *Check the imprinted silicon version on the board (Figure 1). To use A1 version of RW61x silicon, define the macro `CONFIG_SOC_SERIES_RW6XX_REVISION_A1=y` in the file samples/net/wifi/boards/rd_rw612_bga.conf.*

Step 4 – Power cycle RW61x.

Step 5 – The image is flashed onto RW61x. To run the example, open Tera Term or any serial terminal.

Step 6 – Look for the COM port setup information and configure the COM port to a baudrate of 115200. Figure 18 shows the Serial port setup and connection window of Tera Term.

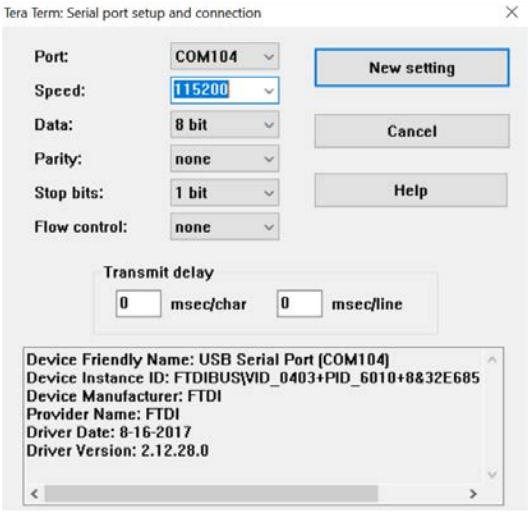


Figure 18. COM port setup information

Step 7 – Power cycle the board and look for the outcome on the terminal.

```
*** Booting Zephyr OS build 264c778fb56c ***
Type "help" for supported commands.Before any Bluetooth commands you must `bt init` to
 initialize the stack.
uart:~$
```

6.2 Build Bluetooth example on Linux

Command syntax:

```
west build -b <BOARD> path/to/source/directory -d path/to/build/directory
```

Command example:

```
# west build -b rd_rw612_bga  samples/Bluetooth/central_ht -d ble
```

Step 1 - Go to *zephyr* directory to access *zephyr.bin* and *zephyr.elf* binaries.

```
cd ~/code/zephyr_rw610/zephyr/ble/zephyr/
```

Step 2 - Flash the image using J-Link Commander.

Command syntax:

```
loadbin <path_to_bin_file_in_build_directory> 0x08000000
```

Command for RW61x:

```
loadbin ble/zephyr/zephyr.bin 0x08000000
```

6.3 CLI commands for Bluetooth

The commands are the same for Linux and Windows.

6.3.1 Bluetooth firmware Init

Command to initialize Bluetooth firmware:

```
bt init
```

Example of output:

```
Bluetooth initialized
Settings Loaded
[00:02:13.887,380] <inf> fs_nvs: 8 Sectors of 4096 bytes
[00:02:13.887,406] <inf> fs_nvs: alloc wra: 0, fe8
[00:02:13.887,409] <inf> fs_nvs: data wra: 0, 0
[00:02:14.146,596] <inf> bt_hci_core: No ID address. App must call settings_load()
[00:02:14.152,308] <inf> bt_hci_core: Identity: C0:95:DA:01:1F:D2 (public)
[00:02:14.152,332] <inf> bt_hci_core: HCI: version 5.4 (0x0d) revision 0x8300,
 manufacturer 0x0025
[00:02:14.152,341] <inf> bt_hci_core: LMP: version 5.4 (0x0d) subver 0x1407
```

6.3.2 Bluetooth scan

Command syntax:

```
bt scan  :<value: on, passive, off> [filter: dups, nodups] [fal] [coded] [no-1m]
```

Example of command:

```
bt scan on
```

Example of output:

```
Bluetooth active scan enabled
[DEVICE]: 44:72:0D:4A:9F:94 (random), AD evt type 0, RSSI -79  C:1 S:1 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 us), SID: 0xff
[DEVICE]: C0:C4:D6:4B:90:3D (random), AD evt type 0, RSSI -77  C:1 S:1 D:0 SR:0 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 us), SID: 0xff
[DEVICE]: C0:C4:D6:4B:90:3D (random), AD evt type 4, RSSI -77  C:1 S:1 D:0 SR:1 E:0 Prim:
 LE 1M, Secn: No packets, Interval: 0x0000 (0 us), SID: 0xff
```

6.3.3 Bluetooth connect

Command syntax to connect to a Bluetooth device:

```
bt connect <address: XX:XX:XX:XX:XX:XX> <type: (public|random)>   [coded] [no-1m]
```

Example of command:

```
bt connect  C0:C4:D6:4B:90:3D random
```

Example of output:

```
Connection pending
Connected: C0:C4:D6:4B:90:3D (random)
LE data len updated: TX (len: 251 time: 2120) RX (len: 251 time: 2120)
LE data len updated: TX (len: 251 time: 2120) RX (len: 251 time: 2120)
LE PHY updated: TX PHY LE 2M, RX PHY LE 2M
```

7 References

[1] User manual – UM11799: NXP Wi-Fi and Bluetooth Demo Applications for RW61x (link)
[2] Webpage – The Zephyr Project (link)
[3] Webpage – Setting up Visual Studio Code (link)
[4] Webpage – pip Installation (link)
[5] Webpage – Zephyr FreeRTOS SDK on GitHub (link)
[6] Webpage – TeraTerm project on GitHub (link)
[7] Webpage – Bluetooth Shell (link)
[8] Webpage – Bluetooth samples (link)
[9] Webpage – Wi-Fi shell (link)

8 Abbreviations

Table 3. Abbreviations

Abbreviation	Definition
AP	access point
CLI	command line interface
DUT	device under test
LE	low energy
repo	repository
STA	station

9 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024-2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10 Revision history

Table 4. Revision history

Document ID	Release date	Description
UM12035 v.2.0	19 March 2025	• Changed the document access to public. • No changes in the content.
UM12035 v.1.0	9 May 2024	• Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Tables

Figures

Contents