MULTI-MODE WIFI, BLUETOOTH, ZIGBEE

QCA4020: Multi-Mode Dual Band WiFi, Bluetooth 5, and ZigBee (802.15.4)

Copyright © 5GHUB.US

Purpose of the Document

The purpose of this document is to explain the QCA4020 which is multi-mode dual band WiFi, Bluetooth 5, and ZigBee (802.15.4). This document contains the features of the QCA4020 and how to configure it.

Document History

Version	Author	Date	Description
A	5G HUB	8.16.2021	Initial Document

Copyright © 5GHUB.US

Table of Contents

Purpose of the Document 2

Document History 2

1 Introduction 4

2 QCA4020 miniPCIe Layout 5

3 QCA4020 miniPCIe Pin Out 6

4 How to flash Image to QCA4020 miniPCIe 9

4.1 Method 1: Through Emergency Download Mode (EDL) 9

4.2 Method 2: Through JTAG 9

5 QCA4020 Pin Out 10

6 Software Tools to be Installed 12

7 Setting Up the Software Development Environment 12

7.1 Python 12

7.2 Java 12

7.3 OpenOCD 12

7.4 GNU ARM Toolchain 13

7.5 Setup OpenOCD Plugin Usage with Eclipse 13

8 Importing “Hello World” Application 15

9 Building “Hello World” Application 18

10 Flash the Application to the QCA4020 miniPCIe using USB 21

11 Flash the Application to the QCA4020 miniPCIe using JTAG 23

12 Connecting QCA4020 to Serial Terminal Through UART 26

13 Using Serial Terminal 27

14 QCLI Demo Application 30

Copyright © 5GHUB.US

1 Introduction

This is a miniPCIe card that supports multi-mode intelligent connectivity. It supports dual-band WiFi, Bluetooth 5, and ZigBee. It is based on Qualcomm QCA4020 System-On-Chip (SoC). It has low power SoC that integrates a Cortex M4F for application processing, Cortex M0 for network stack processing, and a separate processor for Wi-Fi stack designed to enable a highly concurrent multiple radio solution.

The QCA4020 SDK pre-integrates support for advanced security features and multiple software and cloud ecosystems.

Designed to address IoT fragmentation and support for interoperability, this solution is ideal for multiple IoT industries from home control and automation, networking, home entertainment and smart cities.

Feature Highlights

- Multi-mode SoC supporting dual band Wi-Fi, Bluetooth 5, and IEEE 802.15.4 concurrently
- Dedicated processor for Bluetooth LE LC and 15.4 MAC
- Dedicated processor for 802.11 a/b/g/n
- Zigbee 3.0 and OpenThread support
- Isolated power islands for low power operation
- Advanced hardware-based security featuring secure boot, trusted execution environment, encrypted storage, key provisioning and application-level security
- Comprehensive set of peripherals and interfaces: SPI, I2C, UART, HS-UART, ADC and GPIOs
- Integrated sensor hub for post-processing designed to enable low power sensor use cases
- Small package size allows for optimized form factors
- 300+KB RAM reserved for applications
- Bluetooth radio details: v5.0 with PA =+4dBm/+10dBm (for Long Range)
- 802.15.4 radio details: 2006 compliant, 15.4e, 2.4GHz DSSS +4dBm/+21dBm (for Long Range)

Specifications

Wi-Fi

Standards: 802.11a/b/g, 802.11n
Wi-Fi Spectral Bands: 2.4 GHz, 5 GHz
Peak Speed: 150 Mbps
MIMO Configuration: 1x1 (1-stream)

Bluetooth

Bluetooth Specification Version: Bluetooth 5.0
Bluetooth Technology: Qualcomm Bluetooth mesh, Bluetooth Low Energy

802.15.4

LR-WPAN Protocol: Thread, Zigbee

USB

Copyright © 5GHUB.US

USB Version: USB 2.0

CPU

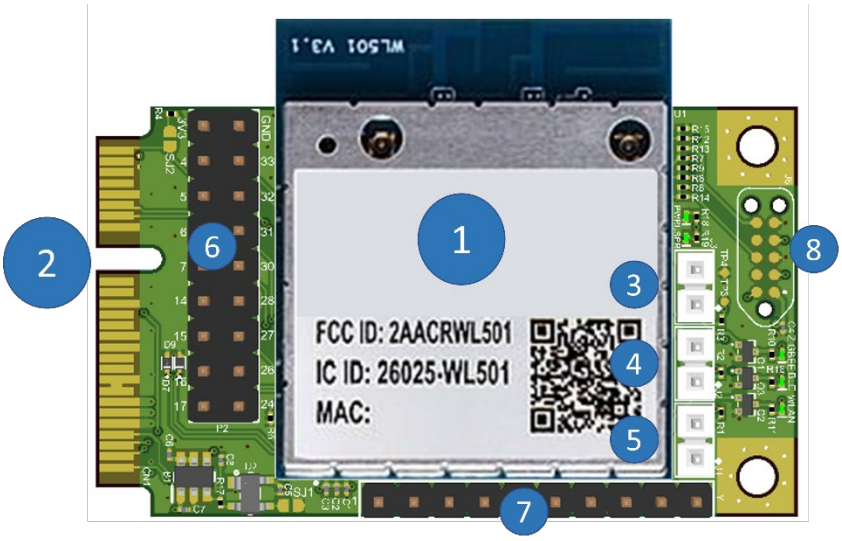
CPU Clock Speed: Up to 128 MHz
CPU Cores: Arm Cortex-M4F CPU, Arm Cortex-M0 CPU, Tensilica Xtensa

Security Support

Security Features: Application-level Security, Hardware-based Crypto Engine, Key Provisioning Security, Qualcomm® Trusted Execution Environment (TEE), Secure Boot, Secure Storage, Software Image Encryption, True Random Number Generator
Wi-Fi Security: WPS Interface

2 QCA4020 miniPCIe Layout

The following figure explains the QCA4020 miniPCIe. It has two headers P1 and P2 which exposes many of the GPIOs and interface of the QCA4020. In addition, it has a JTAG interface for debugging and flashing image. In addition, it has Emergency Download Mode (EDL) jumper header.



1- QCA402 M20 module
2- miniPCIe Interface
3- J3 (EDL)
4- J2 (configure JTAG interface)
5- J1 (Force JTAG) 5- miniPCIe socket
6- P2 Header
7- P1 Header
8- JTAG Interface

Figure 1: QCA4020 miniPCIe Layout.

Copyright © 5GHUB.US

The following figure shows the size and dimension of the QCA4020 miniPCIe.

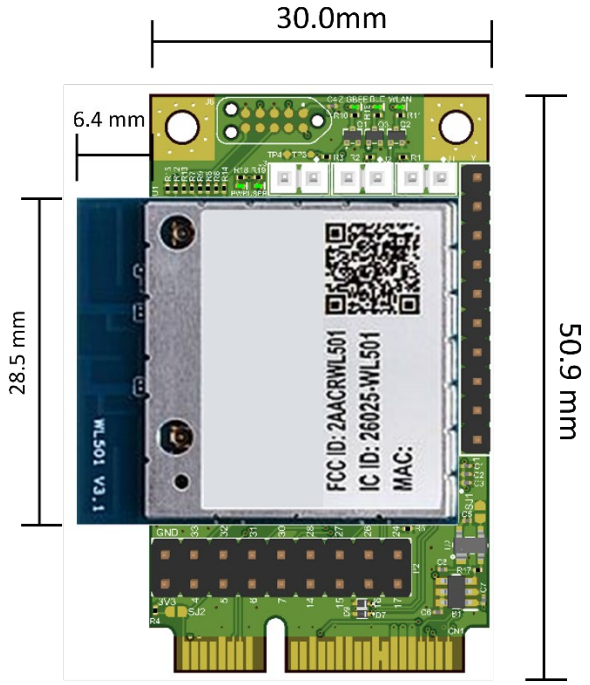


Figure 2: QCA4020 miniPCIe Physical Dimension.

3 QCA4020 miniPCIe Pin Out

The QCA4020 is miniPCIe card and interface. The following figure shows the pin out of the miniPCIe:

Copyright © 5GHUB.US

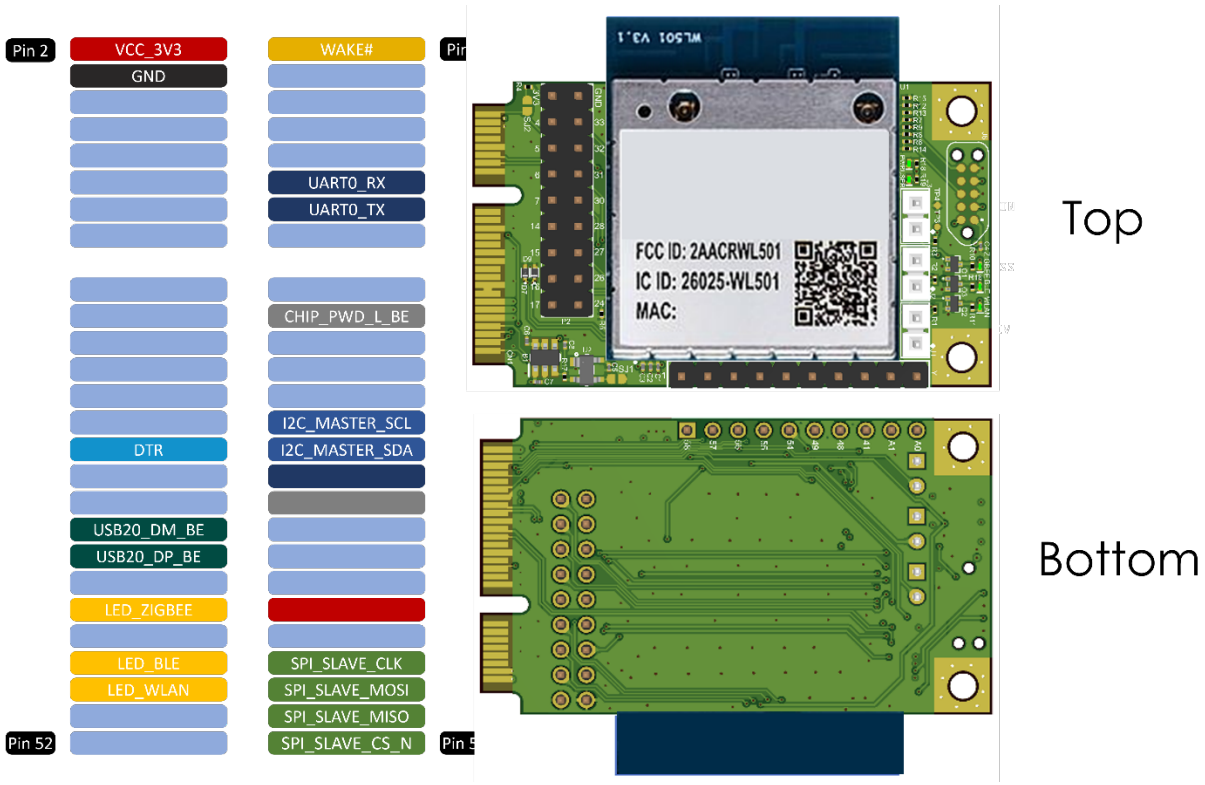


Figure 3: QCA4020 miniPCIe Pin out.

To use the miniPCIe module, you can use USB adaptor (also know as Raspberry PI HAT). The miniPCIe pin mapping as in the table below. If the miniPCIe is used with he Raspberry PI HAT, miniPCIe pins are mapped to Raspberry PI for additional functionality and interface to Raspberry PI.

Table 1: QCA4020 miniPCIe Interface.

PIN #	QCA4020	QCA4020 GPIO PIN#	Raspberry PI PIN#
1	WAKE-UP	GPIO29_BE	
2	VCC_3V3		
3			
4			
5			
6			
7			
8			
9			
10			
11	M0&M4_UART0_RX	GPIO8_BE	GPIO 14 (8)
12			
13	M0&M4_UART0_TX	GPIO9_BE	GPIO 15 (10)
14			

Copyright © 5GHUB.US

15			
16			
17			
18			
19			
20			
21			
22	CHIP_PWD_L_BE	T4	
23			
24			
25			
26			
27			
28			
29			
30	I2C0_MASTER_SCL	GPIO10_BE	GPIO 19 (35)
31	DTR		33
32	I2C0_MASTER_SDA	GPIO11_BE	GPIO 18 (12)
33			
34			
35			
36	USB20_DM_BE	USB20_DM_BE	
37			
38	USB20_DP_BE	USB20_DP_BE	
39			
40			
41			
42	LED_ZIGBEE	GPIO13_BE	
43			
44			
45	SPI_SLAVE_CLK	GPIO18_BE	GPIO 11 (23)
46	LED_BLE	GPIO60_BE	
47	SPI_SLAVE_MOSI	GPIO23_BE	GPIO 9 (21)
48	LED_WLAN	GPIO12_BE	
49	SPI_SLAVE_MISO	GPIO20_BE	GPIO 10 (19)
50			
51	SPI_SLAVE_CS_N	GPIO19_BE	GPIO 8 (24)
52			

Copyright © 5GHUB.US

4 How to flash Image to QCA4020 miniPCIe

Flashing the image o the QCA4020 miniPCIe can be done using either of the following two methods:

4.1 Method 1: Through Emergency Download Mode (EDL)

In this step up, user can do the following:

1- Connect J3 (PIN 1 and PIN 2). This is the GPIO22_BE used for EDL.
2- Connect the USB cable between the PC and the USB port of M20.

4.2 Method 2: Through JTAG

In this step up, user can do the following:

1- Connect J1 (PIN 1 and PIN 2). This is the GPIO20_BE which used to force JTAG mode
2- Connect JTAG cable to the JTAG 10-PIN header which includes the signals **TCK, TDI, TDO, TMS**

Follow normal procedure to flash the *.elf file over the JTAG interface.

QCA4020 JTAG pins used as in the following Table:

Table 2: JTAG Interface.

QCA4020 PIN#	JTAG Signal
GPIO_50	JTAG3_BE_TCK
GPIO_51	JTAG3_BE_TDO
GPIO_52	JTAG3_BE_TMS
GPIO_53	JTAG3_BE_TDI

J2 on the QCA4020 is used to configure JTAG. Connecting J2 (PIN 1 and PIN2), force the following JTAG configuration:

Table 3: J2 Setting for JTAG Interface.

Boot_Configure_BE_0GPIO_9_BE	Boot_Configure_BE_1GPIO_25_BE	Boot_Configure_BE_2GPIO_18_BE	JTAG Interface for M4
0	0	1	JTAG in GPIO[53:50]

Copyright © 5GHUB.US

5 QCA4020 Pin Out

The QCA4020 has abundant number of pins and GPIOs and functionalities which are exposed through P1 and P2 jumper headers. The following table summarizes all pins exposed through P1 and P2 jumper headers and their hardware functionalities.

Table 4: P1 and P2 PIN Mapping.

Header and PIN	GPIO #	SPI or I2C orQSPI	UART	PWMADC/ SenseADC	PTA
P2 - PIN 1	3.3V				
P2 - PIN 2	GPIO4_BE				
P2 - PIN 3	GPIO5_BE				BT_ ACTIVE
P2 - PIN 4	GPIO6_BE				WLAN_ACTIVE
P2 - PIN 5	GPIO7_BE				BT_ PRIORITY
P2 - PIN 6	GPIO14_BE		HS_UART0_DM_CTS		
P2 - PIN 7	GPIO15_BE		HS_UART0_DM_TXD		
P2 - PIN 8	GPIO16_BE	I2C1_Master_SCL	HS_UART0_DM_RFR		BT_ ACTIVE
P2 - PIN 9	GPIO17_BE	I2C1_Master_SDA	HS_UART0_DM_RXD		WLAN_ACTIVE
P2 - PIN 10	GPIO24_BE				
P2 - PIN 11	GPIO26_BE				
P2 - PIN 12	GPIO27_BE				

Copyright © 5GHUB.US

P2 - PIN 13	GPIO28_BE				
P2 - PIN 14	GPIO30_BE				
P2 - PIN 15	GPIO31_BE				
P2 - PIN 16	GPIO32_BE				
P2 - PIN 17	GPIO33_BE				
P2 - PIN 18	GND				
P1 - PIN 10	A0 (SENSEADC_1_BE)				
P1 - PIN 9	A1 (SENSEADC_0_BE)				
P1 - PIN 8	GPIO_41_BE				
P1 – PIN 7	GPIO48_BE				
P1 - PIN 6	GPIO49_BE				
P1 - PIN 5	GPIO54_BE			SENSEADC2	
P1 - PIN 4	GPIO55_BE			SENSEADC3	
P1 - PIN 3	GPIO56_BE			SENSEADC4	
P1 - PIN 2	GPIO57_BE			SENSEADC5	
P1 - PIN 1	GPIO58_BE			SENSEADC6	

Copyright © 5GHUB.US

6 Software Tools to be Installed

Please install the following tools

- **Python 2.7.X** (Download Python | Python.org)
- **Eclipse IDE for C/C++** (Eclipse IDE for C/C++ Developers | Eclipse Packages)
  This is a GUI-based integrated development environment
  Supported Version: Oxygen version - Release 4.7.2
- **Java**:
  Eclipse IDE has dependency on Java, JDK 8 or higher
- **OpenOCD** (Download OpenOCD for Windows (gnutoolchains.com))
  version 0.10.0 [2017-06-09]
- **GNU Arm Embedded Toolchain** (GNU ARM Toolchain)
  version 6.x
- **Qualcomm SDK for QCA4020:** The SDK contains sample demo applications with source code to demonstrate different features and technologies that QCA4020 supports.

7 Setting Up the Software Development Environment

7.1 Python

After installing Python, add the path to python.

Example:

If python.exe is in the folder

```
C:\CRMApps\Apps\Python276-64
```

set path as follows:

```
%PATH%=%PATH%:C:\CRMApps\Apps\Python276-64
```

7.2 Java

After installing Java, add path to Java.

Example:

If Java.exe is in the folder

```
C:\ProgramData\Oracle\Java\javapath
```

set path as follows:

```
%PATH%=%PATH%: C:\Program Files\Java\jdk1.8.0_161\bin
```

7.3 OpenOCD

OpenOCD plugin is required to establish the connection between Eclipse IDE and onboard FTDI JTAG debugger. After installation, add the path to OpenOCD.

Example:

If openocd.exe is in the folder

Copyright © 5GHUB.US

```
C:\Program Files\OpenOCD-20170609\bin
```

set path as follows:

```
set %PATH%=%PATH%:C:\Program Files\OpenOCD-20170609\bin
```

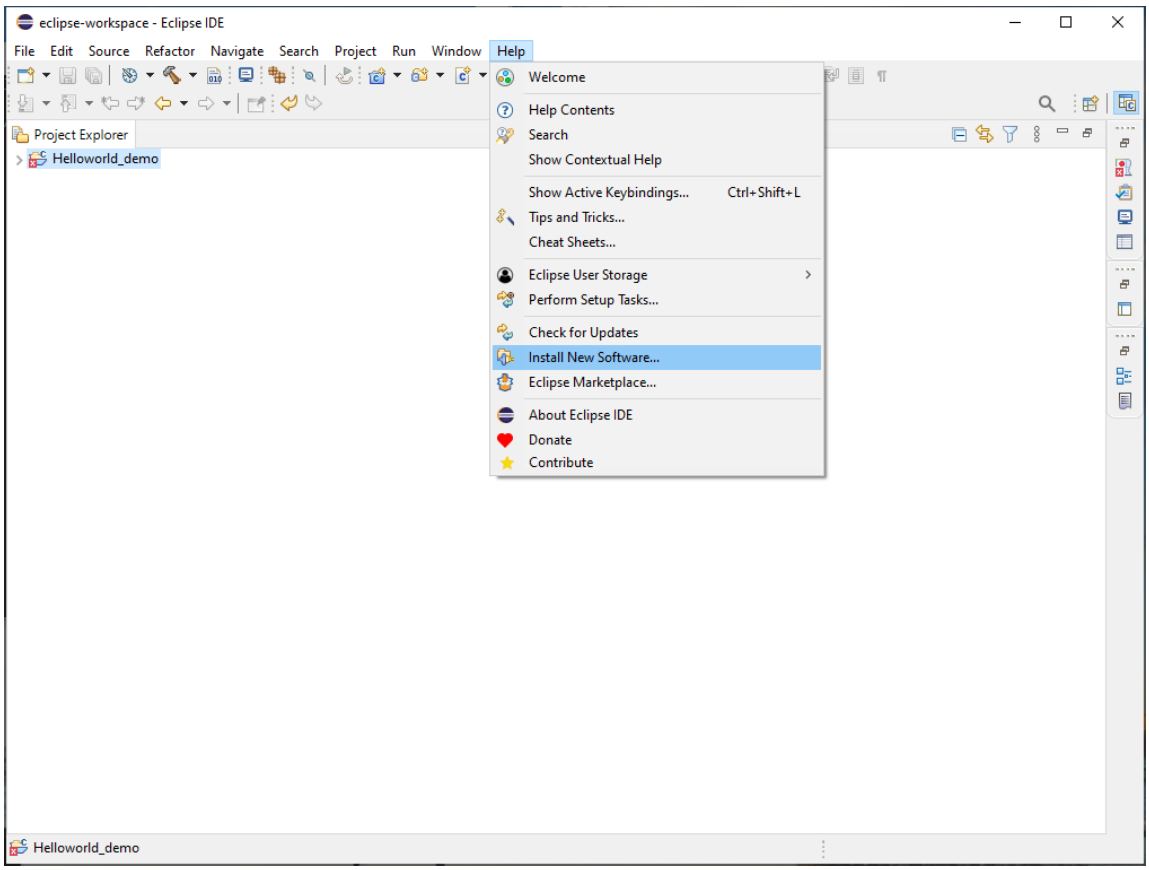
7.4 GNU ARM Toolchain

Install the ARM toolchain by running the ".exe" file and make sure you select the option to "Add path to environment variables" during the final step.

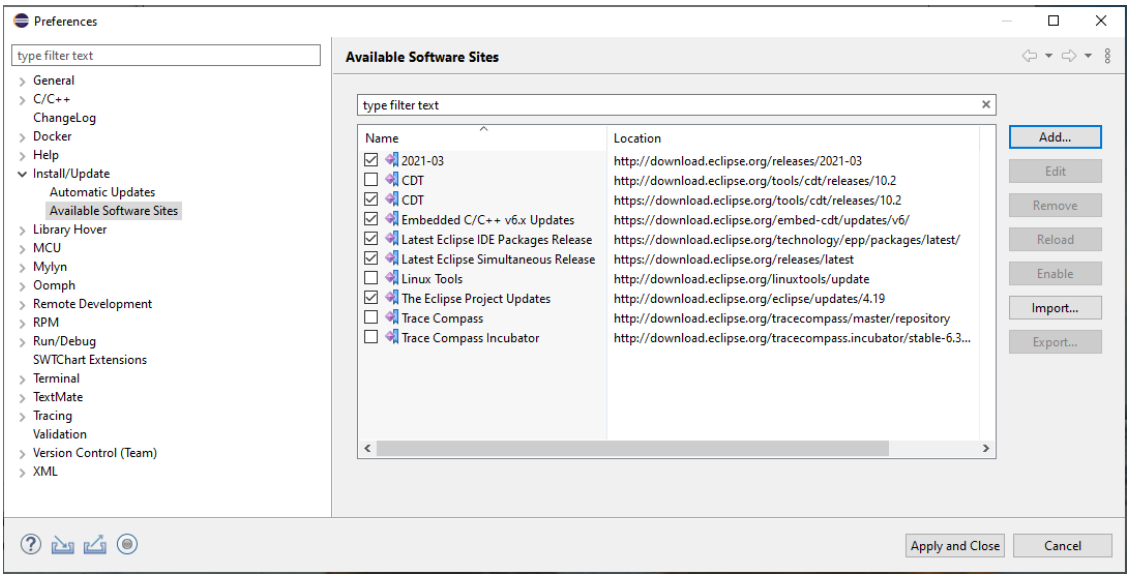
7.5 Setup OpenOCD Plugin Usage with Eclipse

Do the following steps:

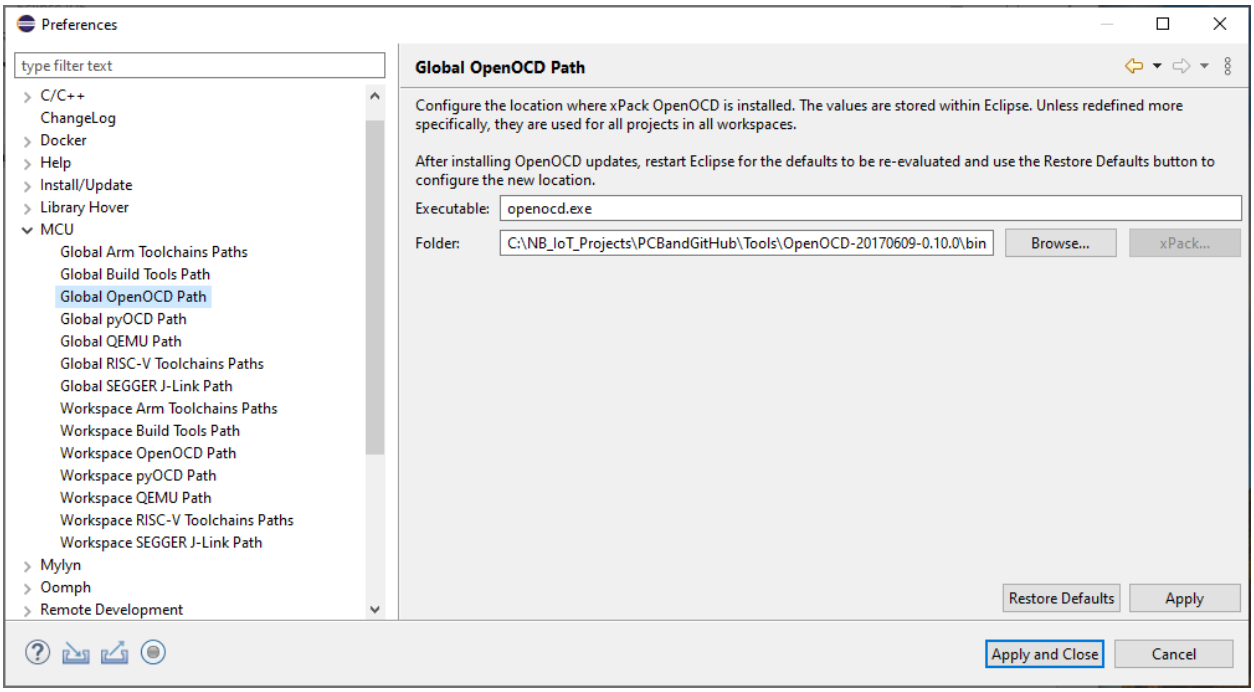
1- Go to **Help > Install new software** in EclipseIDE and make sure the following plugin are installed and enabled by default.



Copyright © 5GHUB.US



2- Set path to openOCD. Restart the Eclipse IDE and under **Window -> preferences**. Set path to openOCD as seen below:



3- After installing the Qualcomm SDK, **QCA4020 OEM SDK+CDB,** Demos samples are in the following folder:

target\quart\demo\

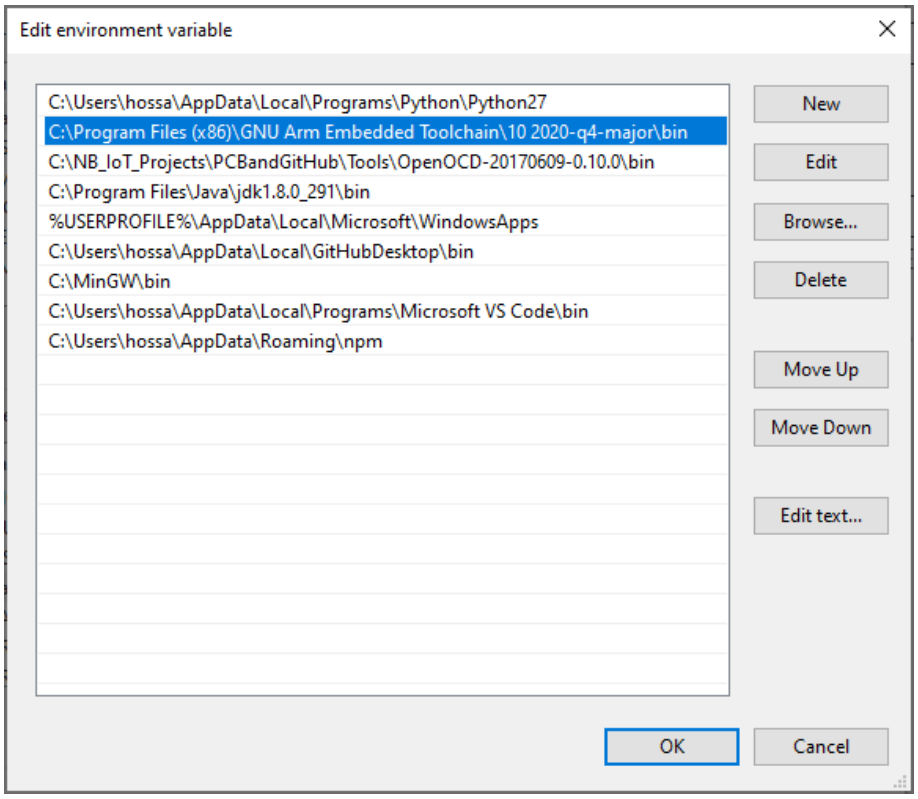
Copyright © 5GHUB.US

4- Install the QCA plugin jar file available at

```
<SDK_source>/target/quartz/demo/EclipseSupportFiles
```

- Copy the jar file (QCA402x_plugin.jar) to the “dropin” folder under the Eclipse IDE installed folder.
- Restart the Eclipse IDE if running. To restart Eclipse, click on the **File->Restart** after the plug-in is installed.

5- Make sure all environment variable are set up correctly



8 Importing “Hello World” Application

1- Install Eclipse project files for sample demo applications.
To install there is eclipseSupport.bat for Windows and eclipseSupport.sh for Linux in the following folder:

```
<SDK_source>/target
```

2- Open a terminal window and navigate to

```
<SDK_source>/target
```

3- Run the following command

Copyright © 5GHUB.US

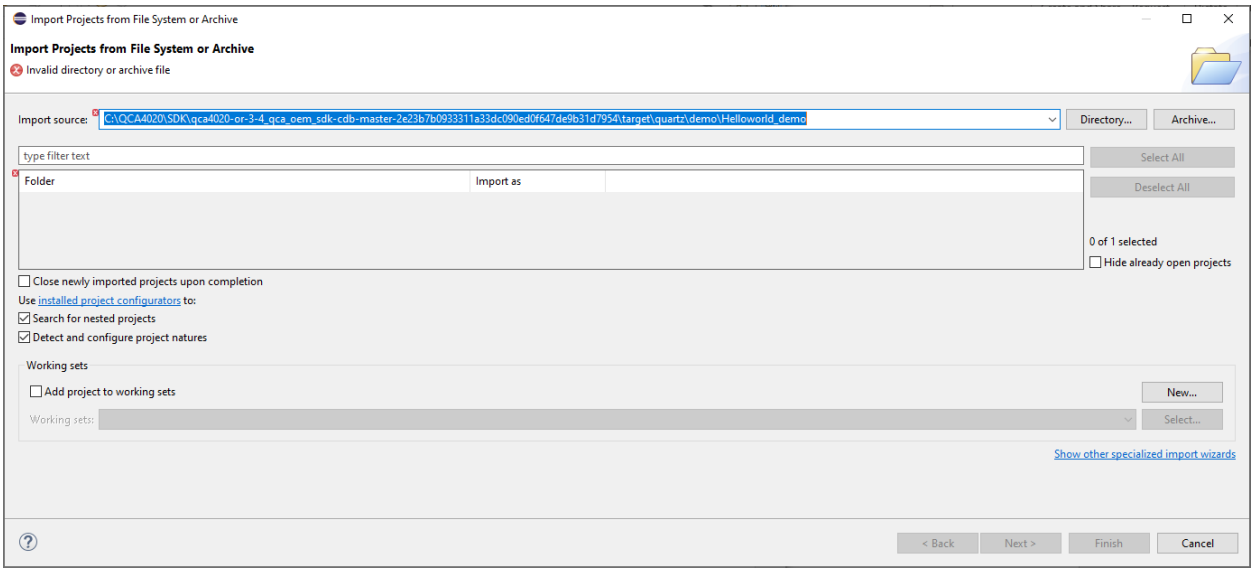
```
sh eclipseSupport.sh
```

Or

```
eclipseSupport.bat
```

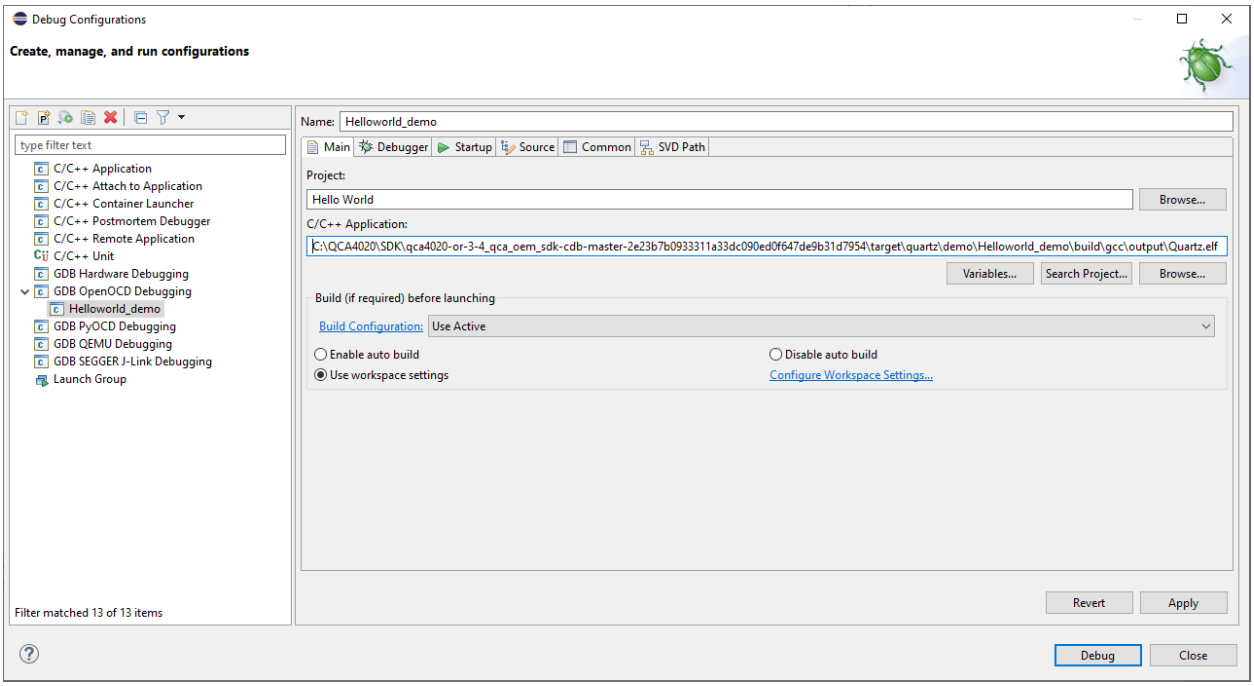
After executing the script, the Eclipse project files ***.cproject, .project*** and ***.settings*** folder are updated in the respective folders of the demo application.

4- Open Eclipse application and Go to **File->Open Project** and choose the folder of the "Hello World" application and check the box beside the application. Click **Finish**



5- To debug the project, set the Debug Config. Go to **Run->Debug Configuration->GDB OpenOCD**. Set the Application and OpenOCD options as show in the images below. Select Quartz.elf as the C/C++ application. If you have already built the image, select Disable auto build. If image is not built, select Use workspace settings.

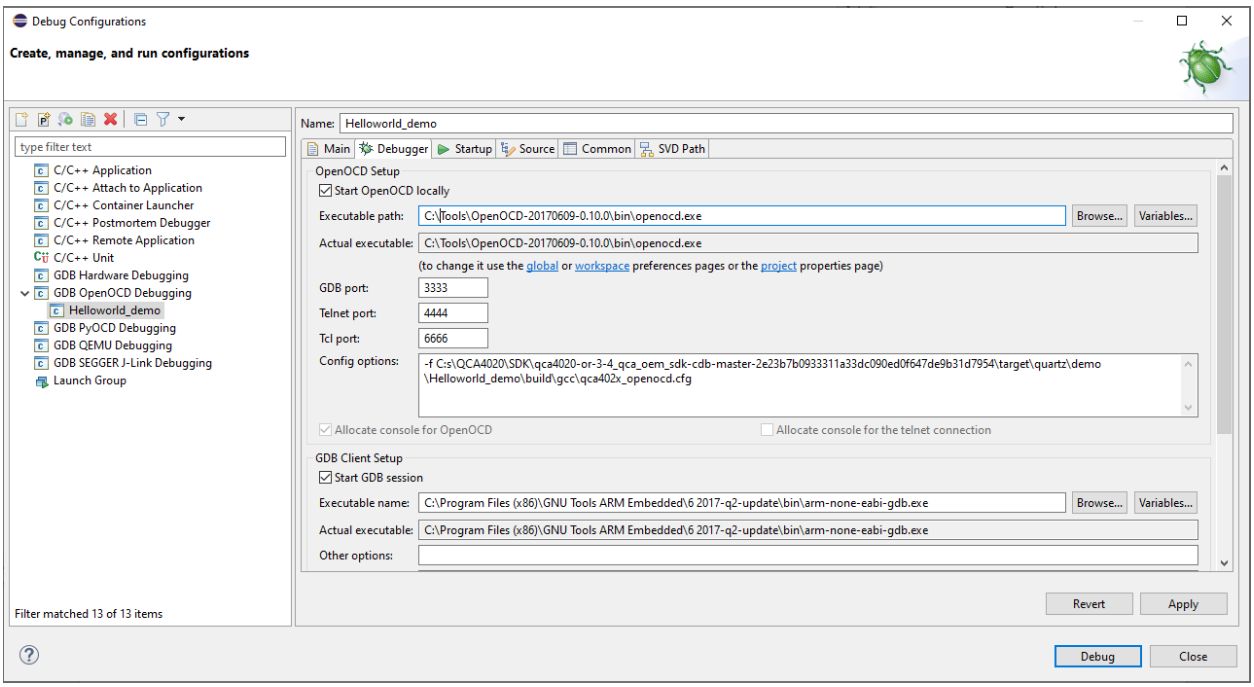
Copyright © 5GHUB.US



6- In the Debugger window, provide the openocd executable path with config option

-f ${project_loc}\build\gcc\qca402x_openocd.cfg

7- Provide the arm-none-eabi-dgb executable path for the GDB client as seen below.

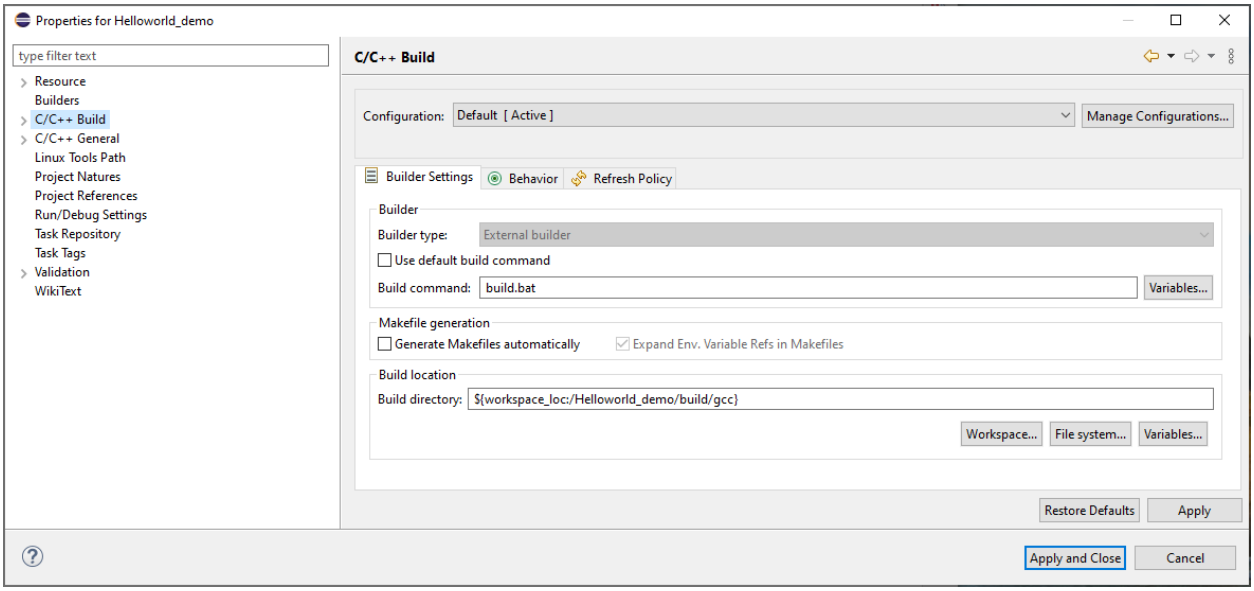


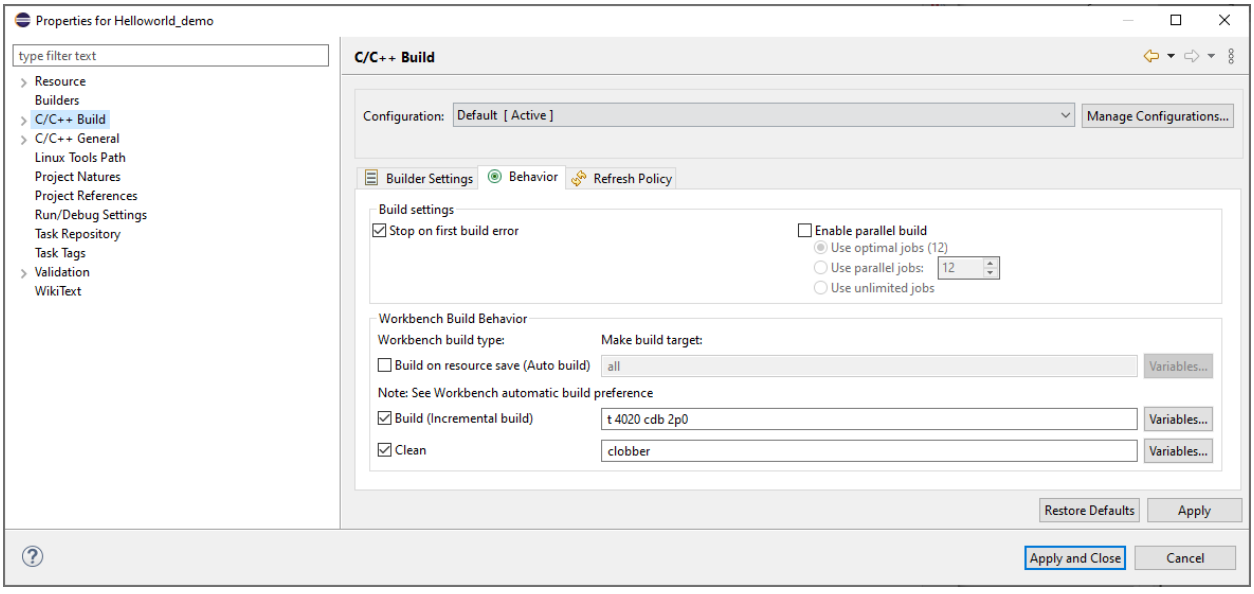
Copyright © 5GHUB.US

9 Building “Hello World” Application

To build the application, do the following:

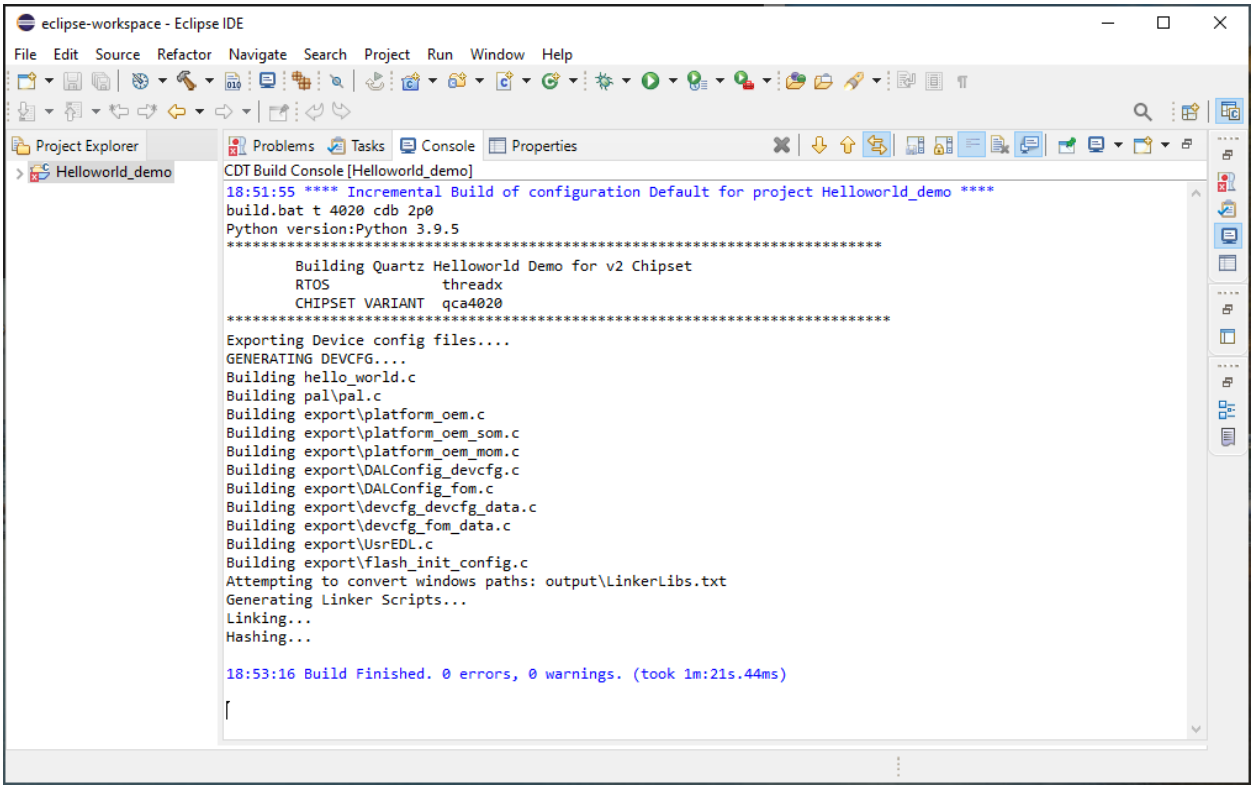
1. Go to Project **properties->Configure build.** Right-click on the project name in project explorer and go to **Properties**
2. Verify the build command and the build directory in the Builder settings tab.
3. Go to C/C++ build and set the build command to “build.bat”, and the build directory to “path to source” as in the following screenshots.
4. Click **Apply and Close**





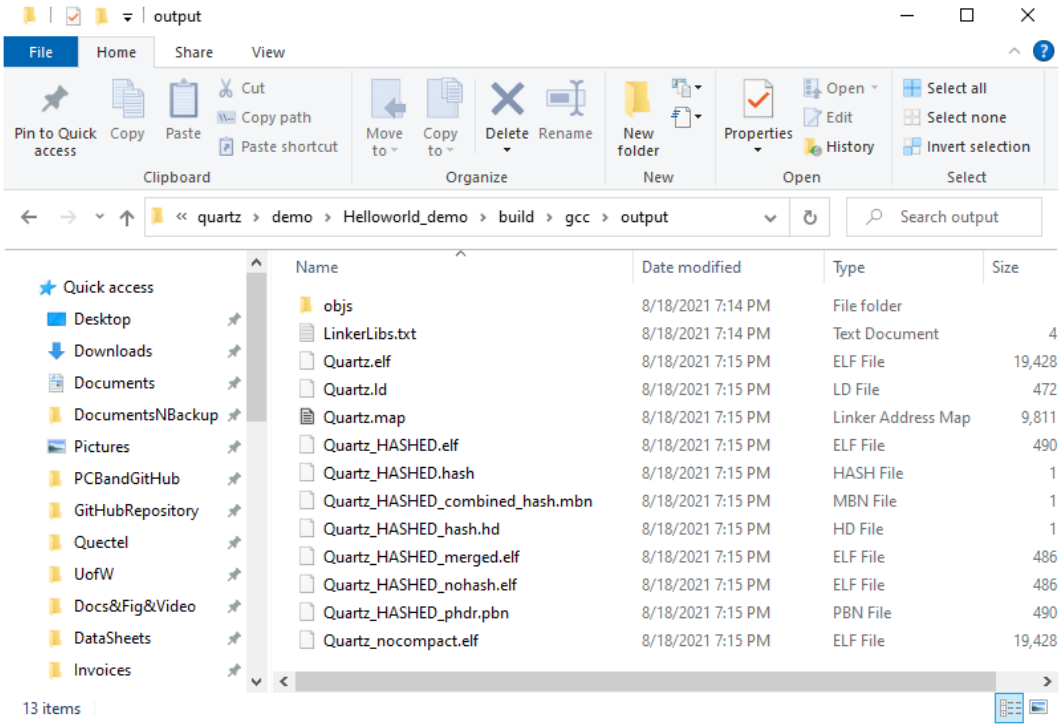
Copyright © 5GHUB.US

5. Build the application and you will see the following message in the Console Window



The output and ***.elf** files are generated and available under the **output** folder:

Copyright © 5GHUB.US



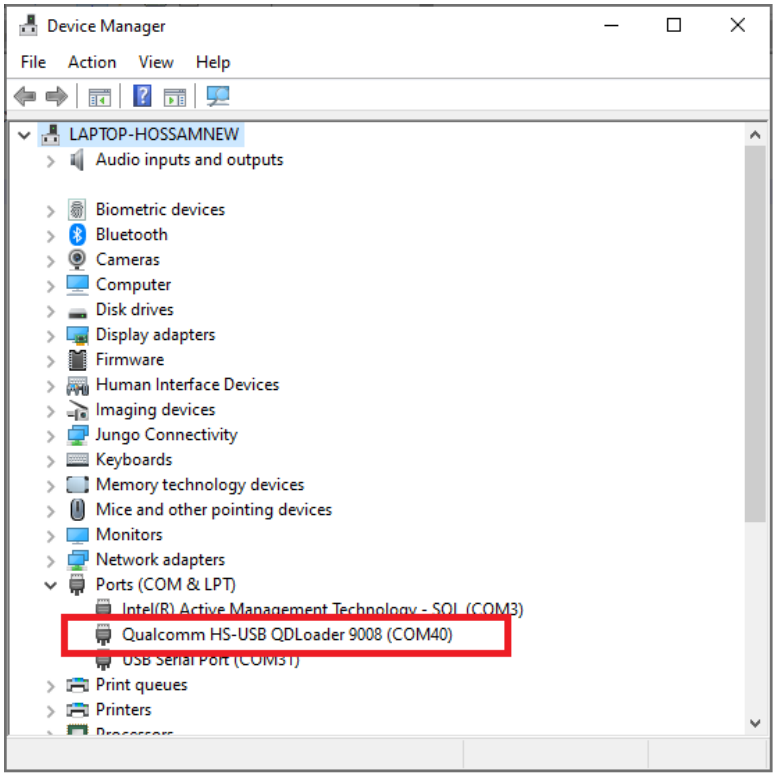
Copyright © 5GHUB.US

10 Flash the Application to the QCA4020 miniPCIe using USB

You can flash an application to the QCA4020 miniPCIe over the USB port. To do so, follow the following steps:

1- Connect a jumper on J3 to short-circuit it. This puts the QCA4020 in EDL (Emergency Download Mode) which allows to download an image to the QCA4020 flash.
2- Connect a Micro USB cable between USB port on the USB adaptor (HAT) and a USB port on a Windows PC.

Open the device manager and you shall see Qualcomm DLoader is enumerated on a serial port (i.e., COM40 in this case).



3- Use the python-based tool called **qflash.py** (which is included in the QCA4020 SDK). The tool allows flashing images over USB. The tool is available at the following folder:

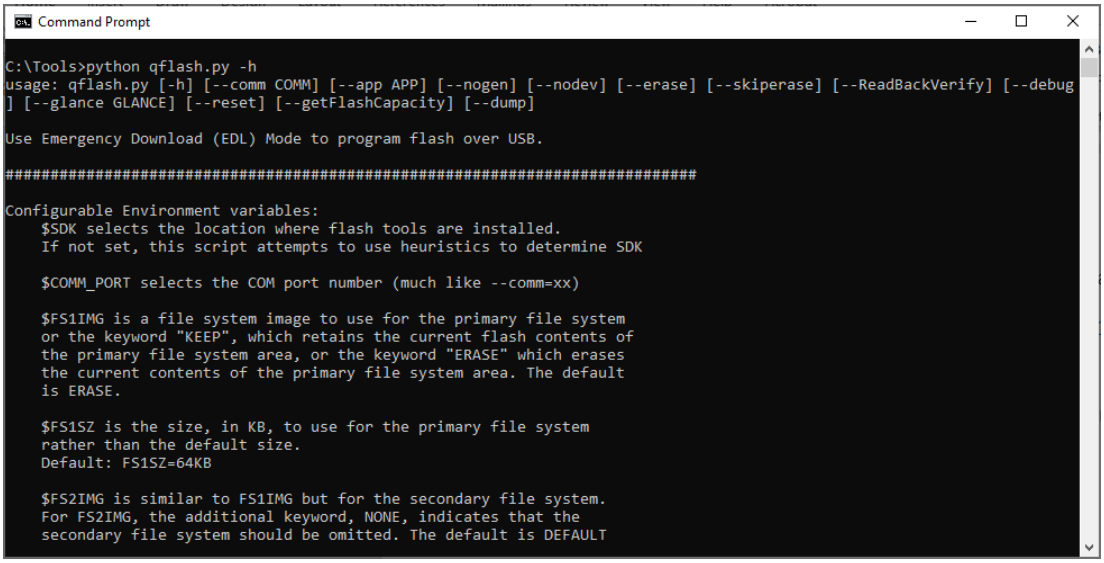
```
target\build\tools\flash
```

If the tool is invoked without any optional parameters, it does he following:

a) Generate a default firmware descriptor table.
b) Generate a default partition table.
c) Flash the default sample application ***.elf** files to the qca4020 flash.

Copyright © 5GHUB.US

When you run this tool with **-h** parameter, you get the following as in this screenshot:

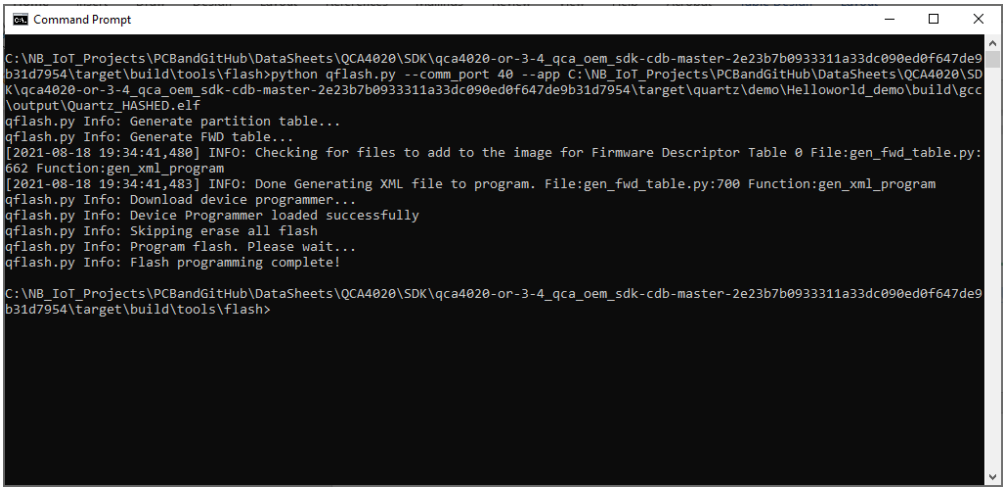


4- Run the following command to flash the application to the QCA4020

python qflash.py --comm_port 40 --app C:\QCA4020\SDK\qca4020-or-3-4_qca_oem_sdk-cdb-master-2e23b7b0933311a33dc090ed0f647de9b31d7954\target\quartz\demo\Helloworld_demo\build\gcc\output\Quartz_HASHED.elf

Noe the image to be flashed is the **Quartz_HASHED.elf**.

You shall see the output as in the below screen shoot which indicates the image is successfully flashed to QCA4020.



Copyright © 5GHUB.US

11 Flash the Application to the QCA4020 miniPCIe using JTAG

You can use JTAG interface on the QCA4020 miniPCIe to flash the image. A SEGGER J-link hardware is used wit connect and interface to the JTAG.

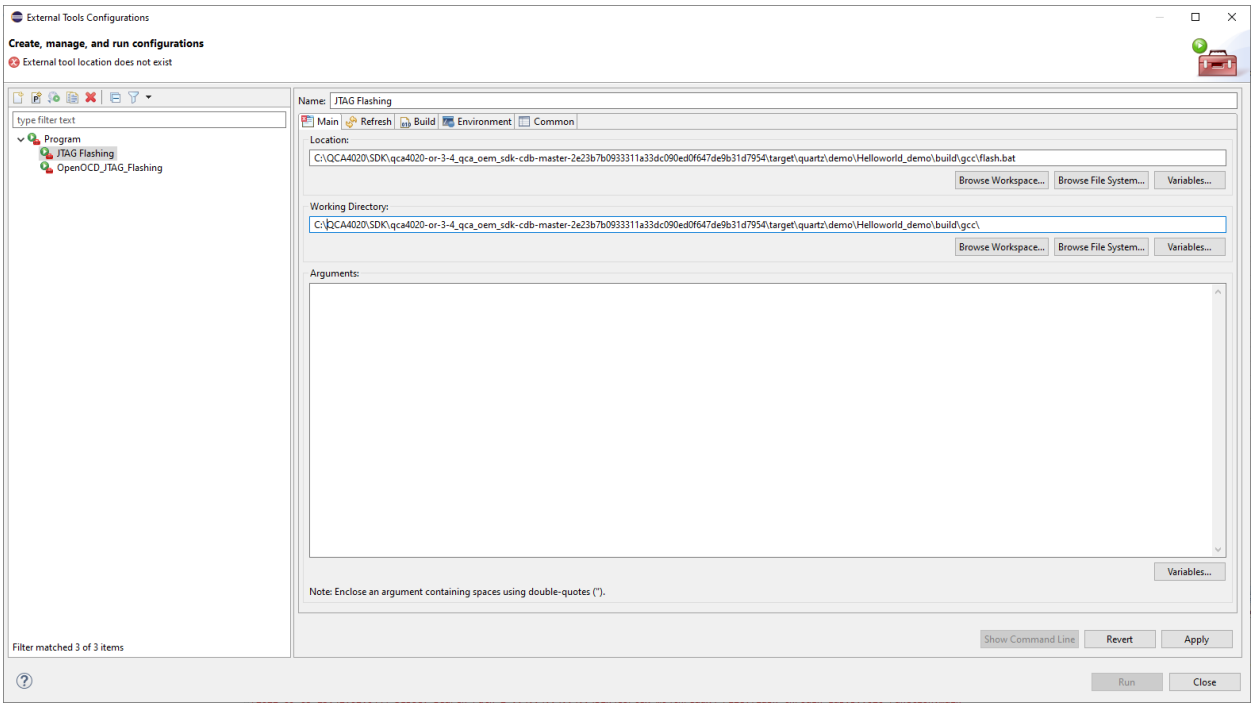
On the QCA4020 miniPCIe, do the following

- Connect J1 (PIN1 and PIN2). This force JTAG mode
- Connect J3 (PIN1 and PIN2). This enable JTAG interface

On Windows environment variables, add the following environment variable:

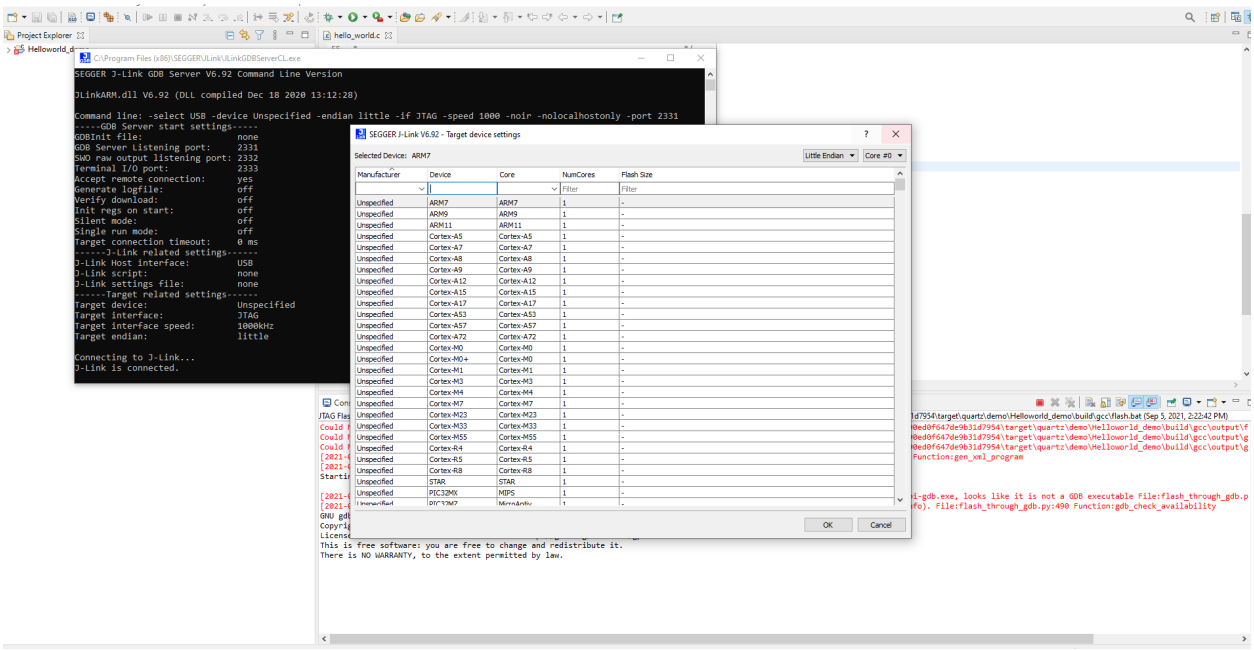
- **JLINK_PATH** set it to J-link GDB server executable file (e.g., **C:\Program Files (x86)\SEGGER\JLink**)
- **CLIENT_PATH** set it to ARM GNU tools (e.g., **C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2020-q4-major\bin**)

Using Eclipse IDE, go to **Run->External Tools->External Tools Configurations** and add a new configuration. Point to the **flash.bat** file in the SDK which is used to configure and launch GDB server. Click **Apply** and then **Run**.

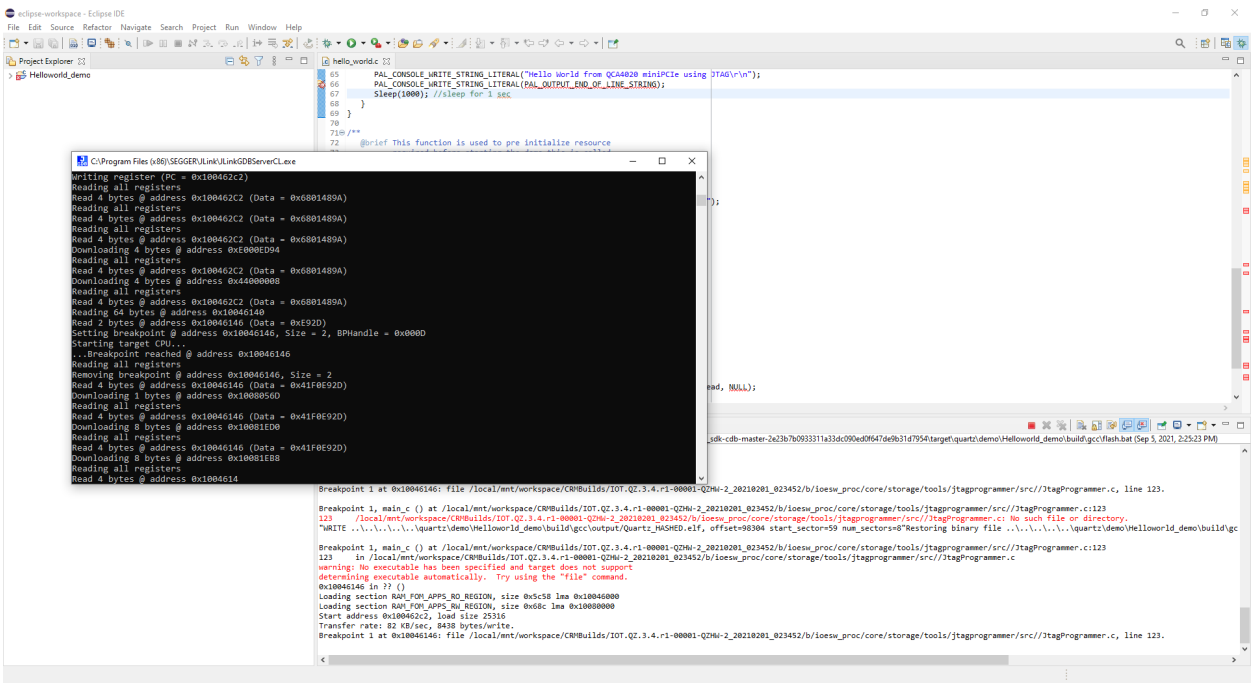


You will see the following screenshot and prompted to select device or core. Choose **Cortex-M4**. Click **OK**.

Copyright © 5GHUB.US



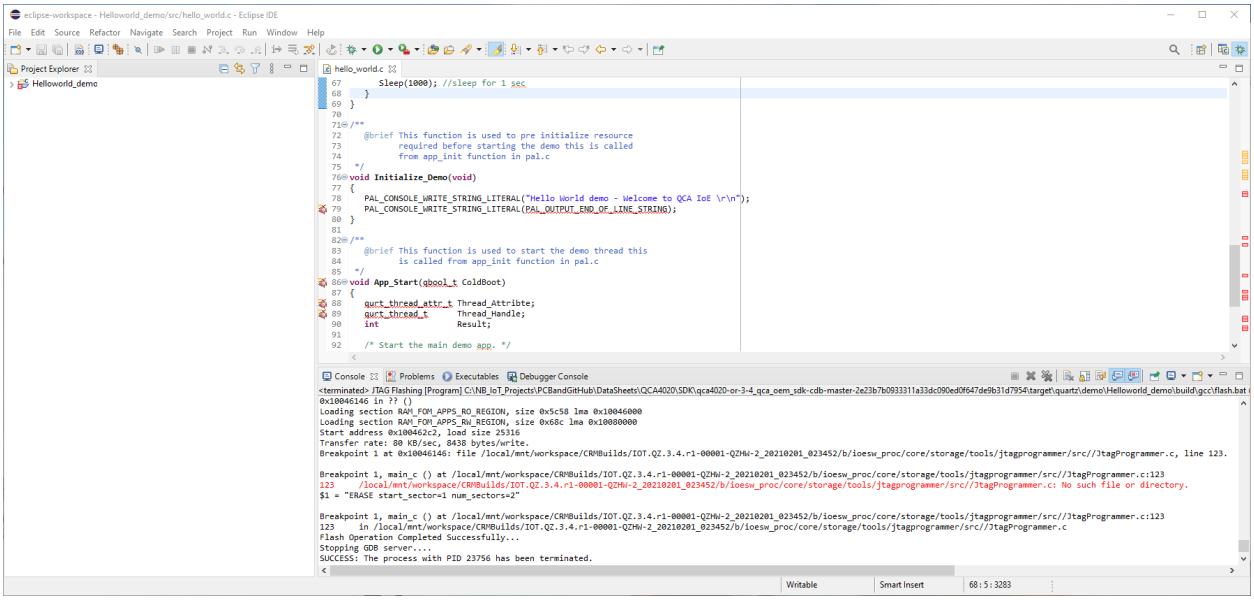
Eclipse will start flashing the image to the QCA4020 miniPCIe over JTAG. You will see the following screenshot.



Copyright © 5GHUB.US

When flashing is completed. You will see the following screen shoot with the output:

```
Flash Operation Completed Successfully...
Stopping GDB server....
SUCCESS: The process with PID 23756 has been terminated.
```



After finishing flashing the image, make sure to remove J1 & J3 on QCA4020 miniPCIe to exit JTAG mode.

Copyright © 5GHUB.US

12 Connecting QCA4020 to Serial Terminal Through UART

You can use the QCA4020 miniPCIe when mounted on USB adaptor (HAT) with any device that has UART (Tx/Rx) interface such as a Raspberry PI board, Arduino board or any other hardware board, with UART interface.

You can also use and connect the QCA4020 miniPCIe/USB adaptor (HAT) to a USB port in the computer through a USB-to-UART cable. You can use cable such as this one:
USB TO UART TTL (Wires) Serial Cable (PL2303HX) MCP00102W Programmer Arduino Compatible in Elecrow bazaar!

Such a cable has four wires colored as follow:

Red: +5V

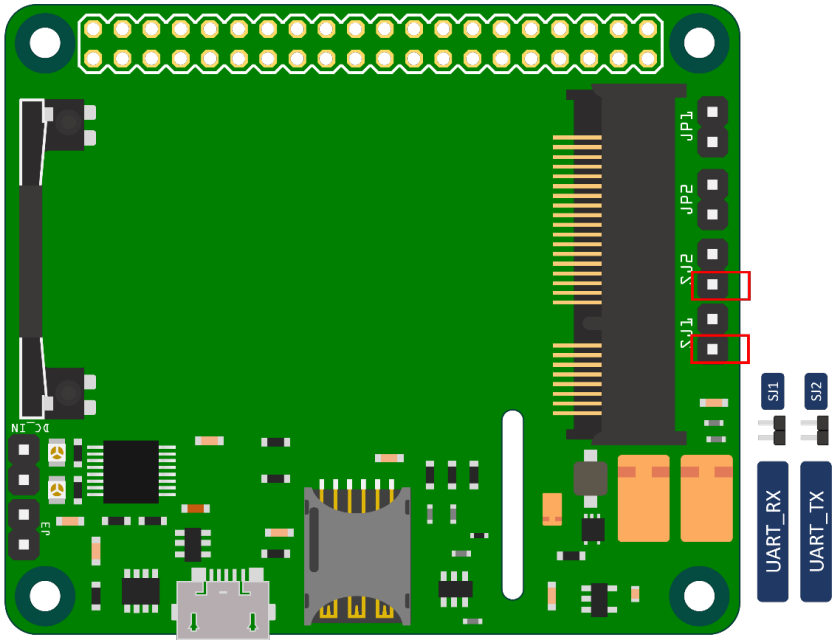
White: Tx

Green: Rx

Black: GND

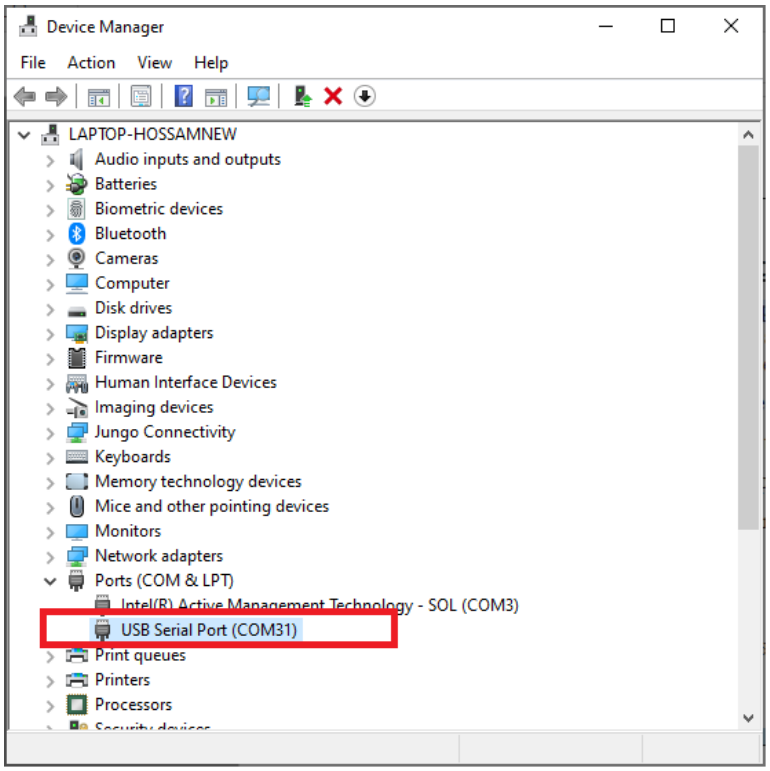
Connect the **White** cable to **UART_TX** and **Green** cable to **UART_RX**. Also connect the **Black** cable to the **GND**).

As in the picture below, connect **UART_TX to SJ2 (PIN2) AND UART_RX to SJ1 (PIN2)**.



Copyright © 5GHUB.US

Make sure when you connect the USB-to-UART cable, it shows correctly in Windows device manager and all its driver is installed as in this screenshot.

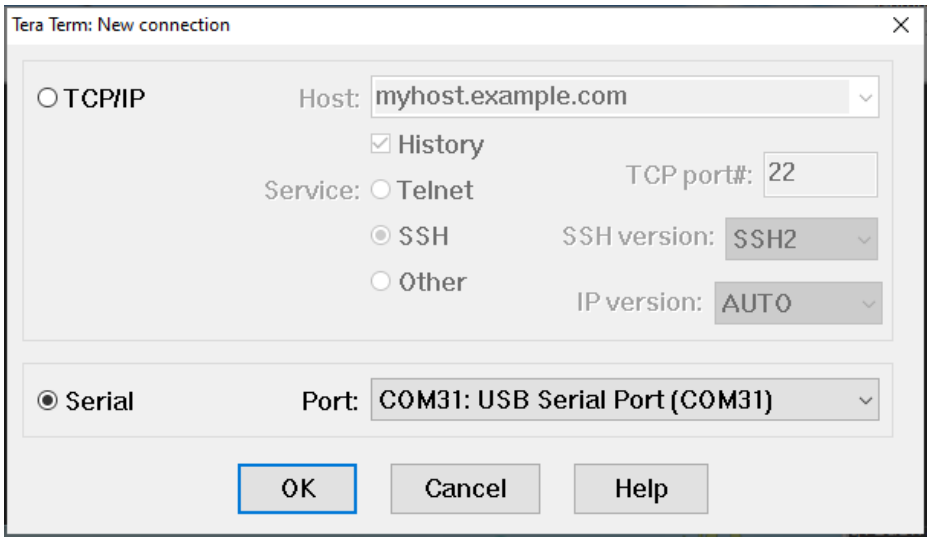


13 Using Serial Terminal

It is recommended to use Tera Term tool as the serial terminal. You can download it from here:

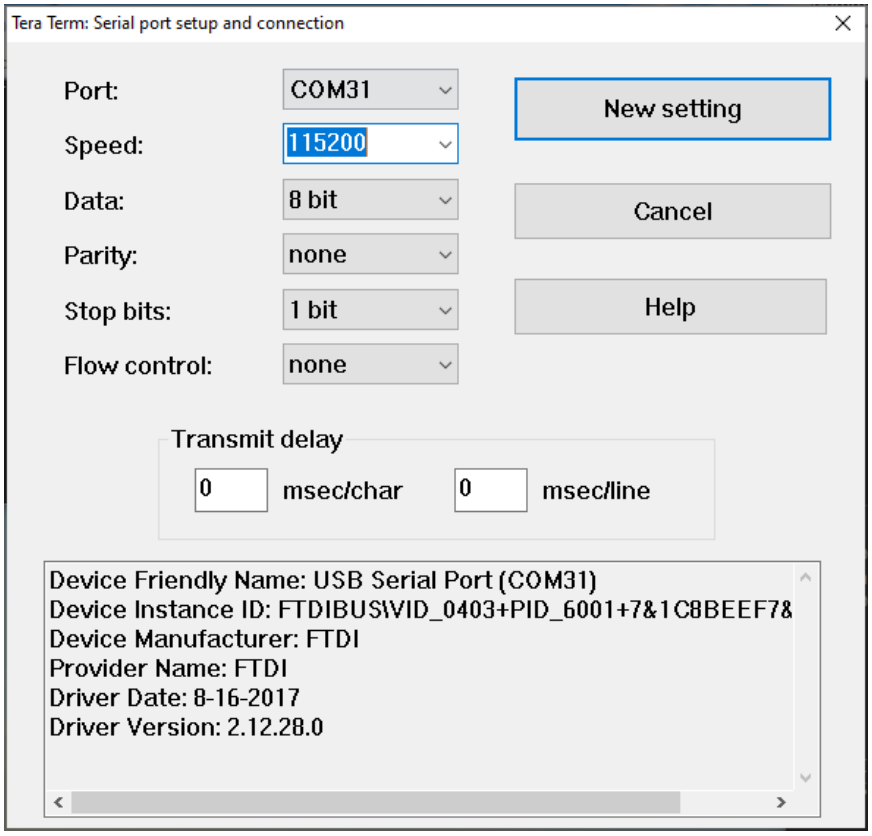
https://osdn.net/projects/ttssh2/downloads/54081/teraterm-4.72.exe/

Launch Tera Terminal and select the **Serial** option and select USB-to-Serial port.

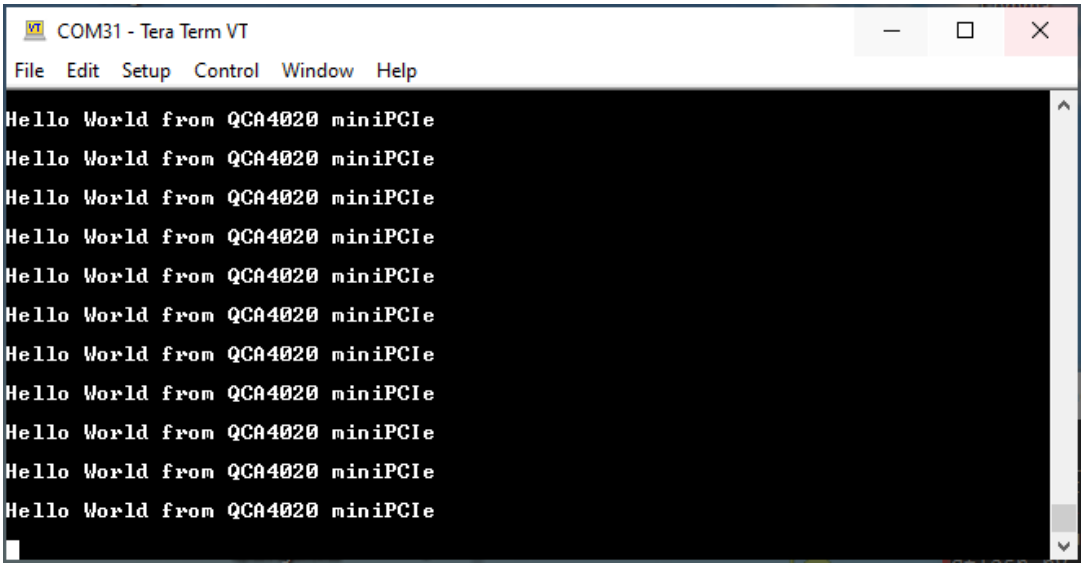


Copyright © 5GHUB.US

In Tera Term, choose **Setup->Serial port** and configure serial ports according to the following:

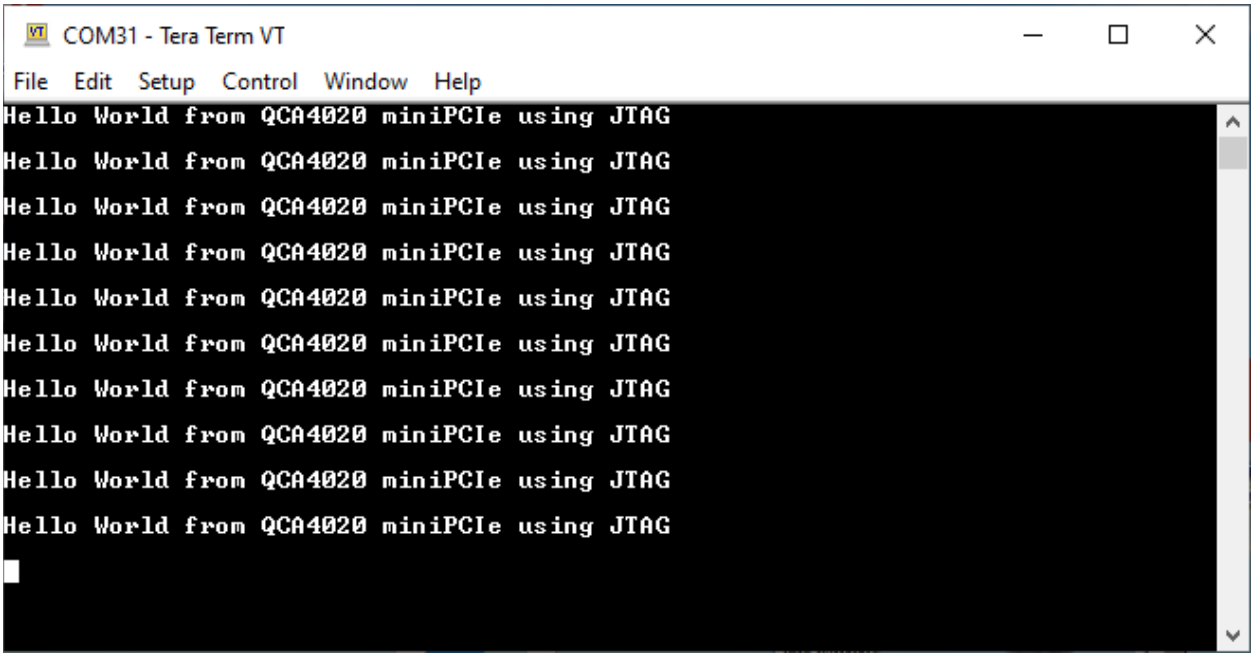


And now you are ready to see the output from “hello World” application on the serial monitor.



Copyright © 5GHUB.US

If you are using JTAG mode to flash the image to the QCA4020 miniPCIe, you shall see the following output on Tera terminal

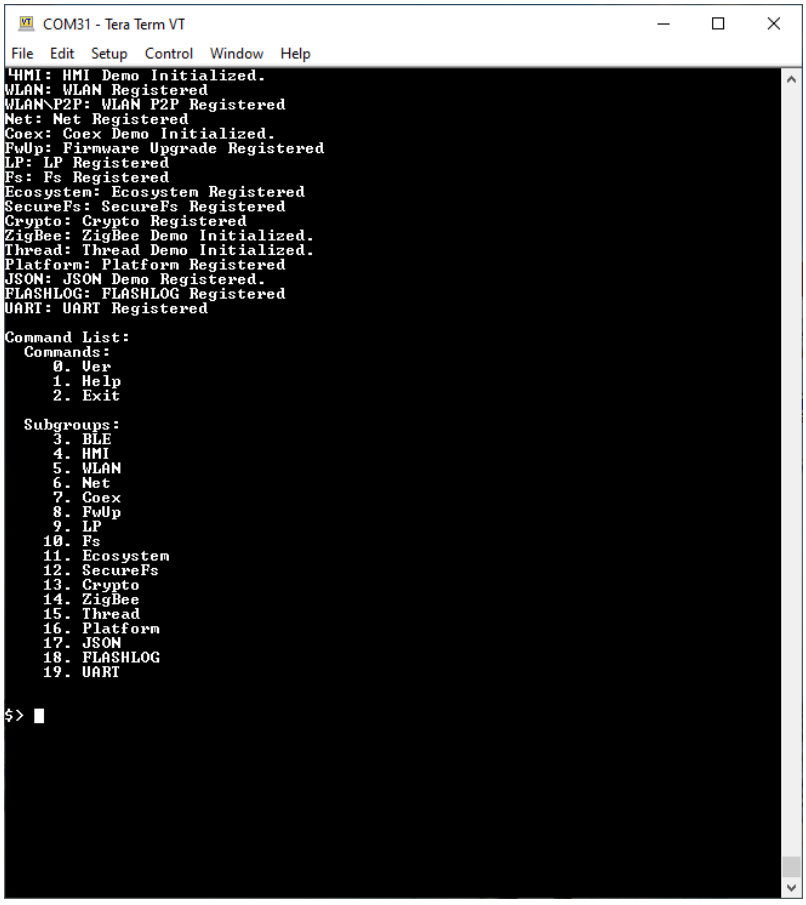


Copyright © 5GHUB.US

14 QCLI Demo Application

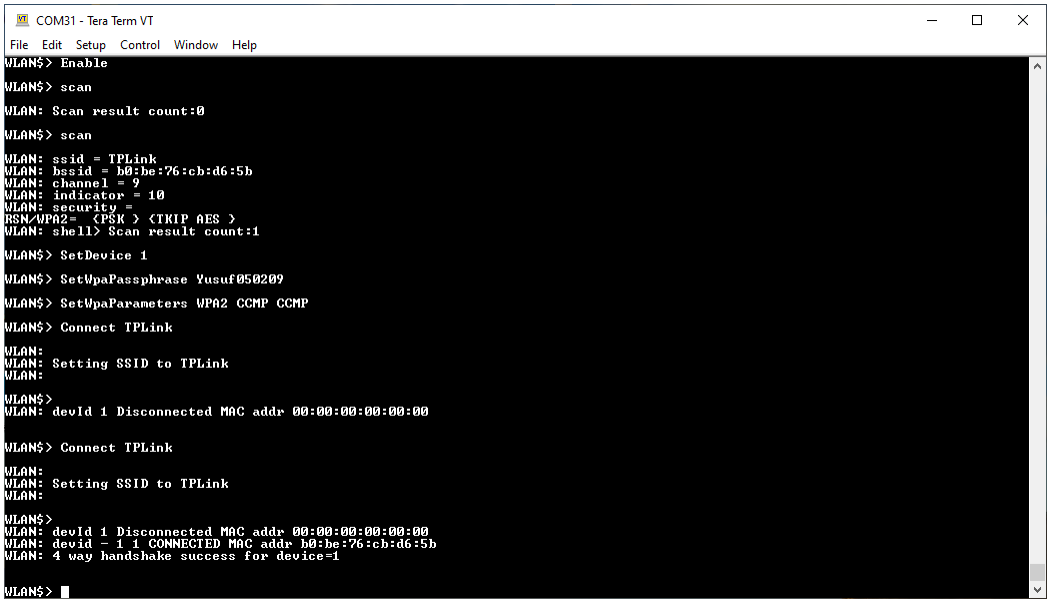
CLI demo application is a comprehensive demo that provides a mechanism to demonstrate different and all features and technologies that QCA4020 miniPCIe supports. It also provides reference implementation and usage of customer facing QAPIs.

The QCLI-demo application is available with the QCA4020 SDK and you can follow the same steps described before in Eclipse to build and flash it to the QCA4020 miniPCIe. Once flashed, you can see the output on the serial terminal as in the screenshot below. Also, you can exercise connecting to your home/office WLAN



The following screenshot shows how to connect to an WLAN access point.

Copyright © 5GHUB.US



Copyright © 5GHUB.US