Using the BL65x or BL5340 and Nordic nRF Connect SDK v2.x with Visual Studio Code IDE

BL65x, BL5340

Application Note *v2.0*

1 Introduction

This application note is intended to help developers who want to use BL65x or BL5340 modules to develop applications using freely available tools and the Nordic nRF Connect SDK v2.x (NCS). The following are step-by-step instructions for building, programming, and debugging NCS applications using Visual Studio Code (VS Code) IDE. Prior to NCS v2.0.0, both VS Code and SEGGER Embedded Studio IDEs were supported in the nRF Connect for Desktop Toolchain Manager. nRF Connect SDK no longer uses the GNU Arm Embedded Toolchain by default. The current default toolchain is the Zephyr SDK instead, which now supports all three main operating systems (Microsoft Windows, macOS, and GNU/Linux). With NCS v2.0.0 and later only VS Code is made available as an IDE as VS Code provides many features including both Command Line Interface (CLI) and Graphical User Interface (GUI) in one environment. The peripheral_hr example (which provides a peripheral heart rate service) is used to demonstrate how to build, flash, and debug SDK applications on the BL65x or BL5340 development kit. For this application note, the BL654 is used for demonstration.

2 Requirements

The following are required to complete this step-by-step guide:

- BL65x Development kit (This demo uses BL654: product numbers 455-00001 or 455-00002)
- USB A to micro-USB cable (provided with BL65x development kit)

Note: The following sections contain a step-by-step guide on obtaining and installing the additional requirements listed below. However, please note that the examples were completed and tested using what was current at the time and listed below, but it is recommended to always use the most up to date versions available:

- nRF Connect SDK v2.2.0
- Visual Studio Code V1.75.0
- nRF Command Line Tools v10.19.0
- nRF Connect for Desktop v3.12.0

For more information regarding VS Code for nRF Connect SDK, you can visit the Nordic tutorials at:
nRF Connect for VS Code Tutorials

3 Development Environment Setup

3.1 nRF-Command-Line-Tools

First the nRF Command Line Tools needs to be installed. The nrfjprog is a command line tool used for erasing and flashing the BL65x with applications. The tool is bundled as part of the nRF-Command-Line-Tools. You can install and setup the command line tools as follows:

1. Navigate to nRF Command Line Tools
2. Select the **Downloads** tab.
3. Download the installer for your platform.

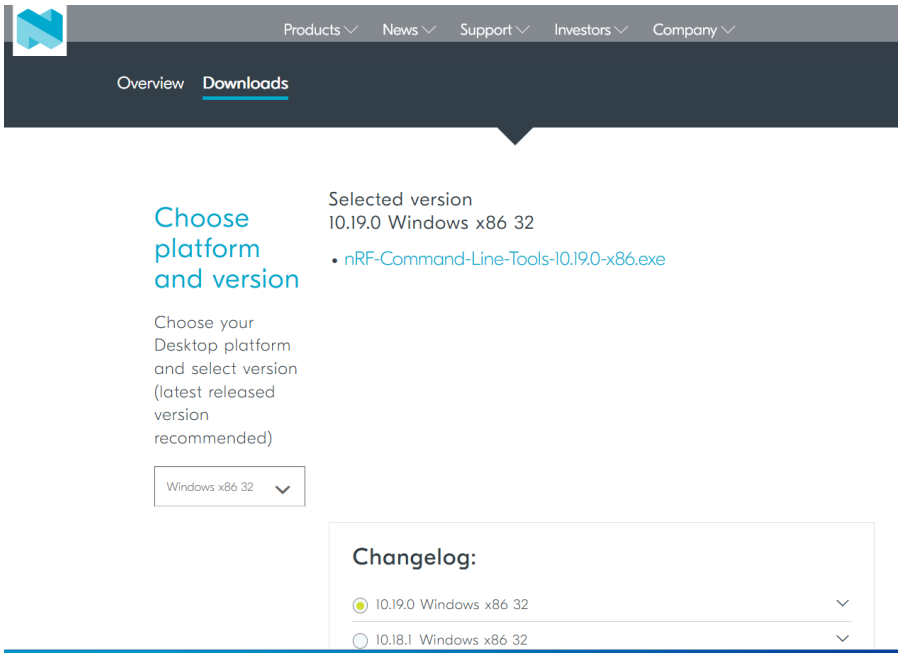


Figure 1: nRF Command Line Tools Download

4. Once downloaded, launch the executable. Proceed through the setup.

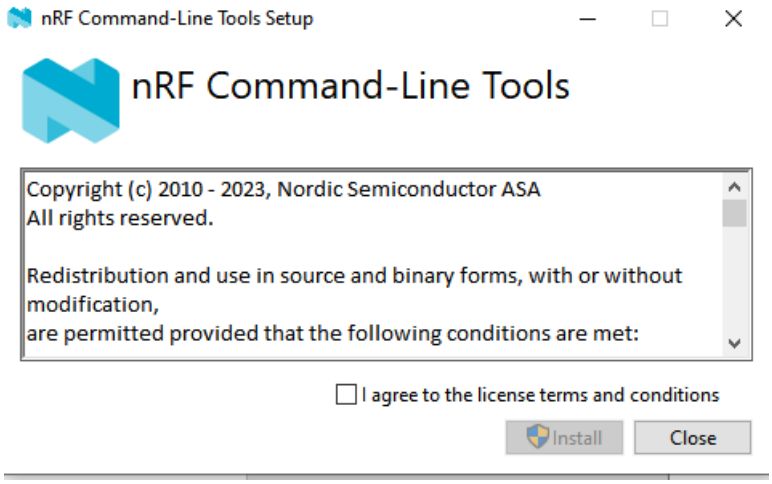


Figure 2: nRF Connect Command Line Tools Install Launch

Note: During the setup process, you will be prompted to install SEGGER J-Link. This is required for the command line tools to work, so proceed by accepting and installing it.

---

5. Once downloaded, the **nrfjprog** executable can be found at **C:\Program Files (x86)\Nordic Semiconductor\nrf-command-line-tools\bin** (Figure 3).

Note: For 32-bit Windows installations, the files will be installed in:
C:\Program Files\Nordic Semiconductor\nrf-command-line-tools\bin

---

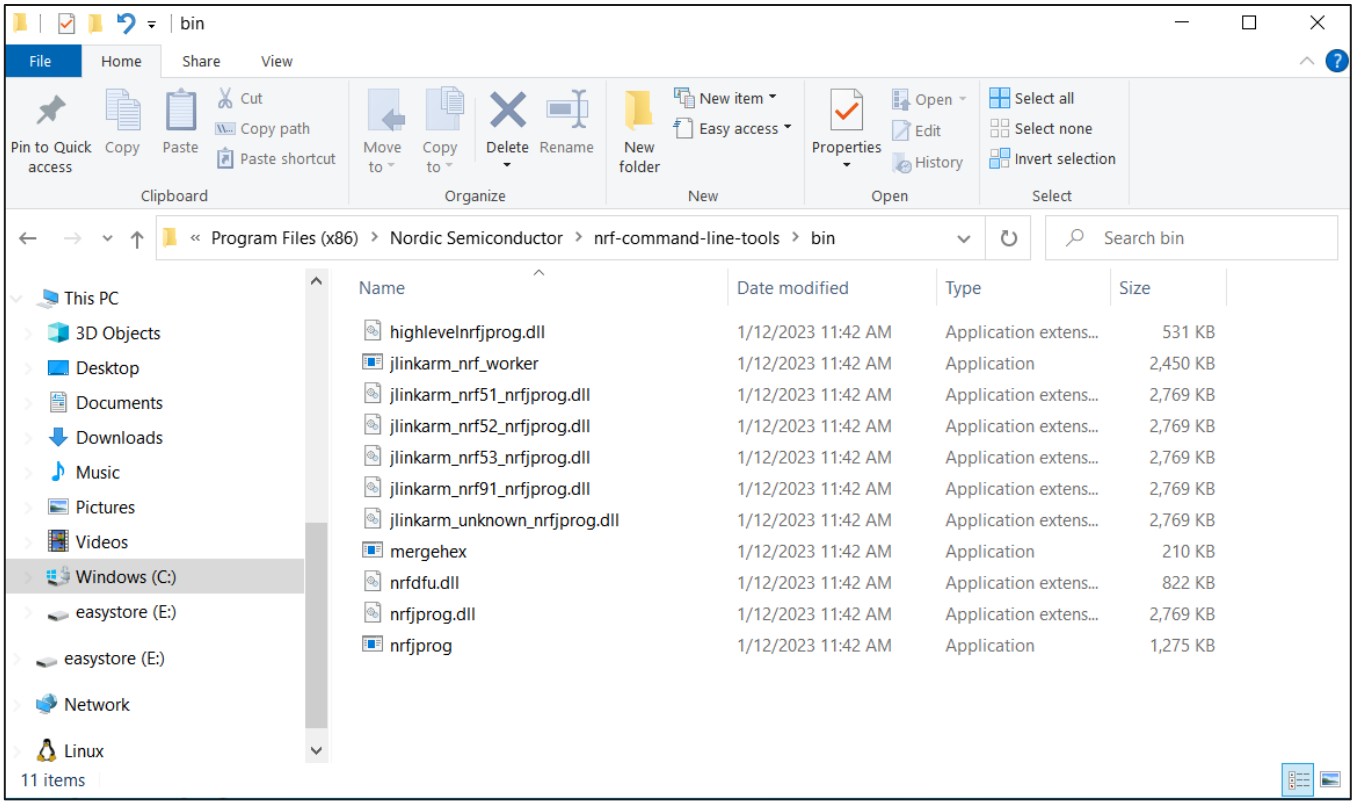


Figure 3: Location of nrfjprog.exe

3.2 Nordic SDK

The nRF Connect SDK can be downloaded from within nRF Connect for Desktop, using *Toolchain Manager*. This offers a rich development environment along with example applications for the BL654 module. When development with C programming language is preferred, this application can easily be used to develop custom applications for the BL654, using the freely available software. The SDK offers a wide selection of drivers, libraries, and examples for the module and its peripherals.

To use nRF Connect SDK for BL65x development, complete the following steps:

1. Download nRF Connect for Desktop from nRF Connect for Desktop
2. Select the Downloads tab.
3. Download the Installer for your platform.

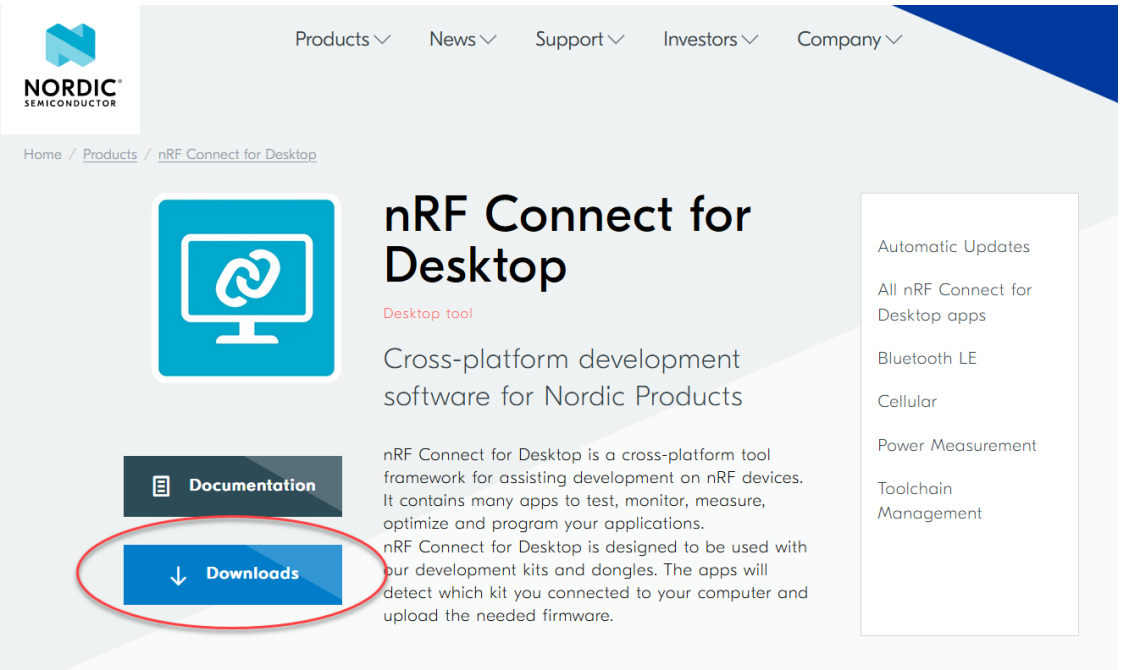


Figure 4: nRF Connect for Desktop Download

4. Once downloaded, open the nRF Connect for Desktop application and open the Toolchain Manager. If there is an update available for Toolchain Manager, first select update. Once the latest version is loaded select **Open**.

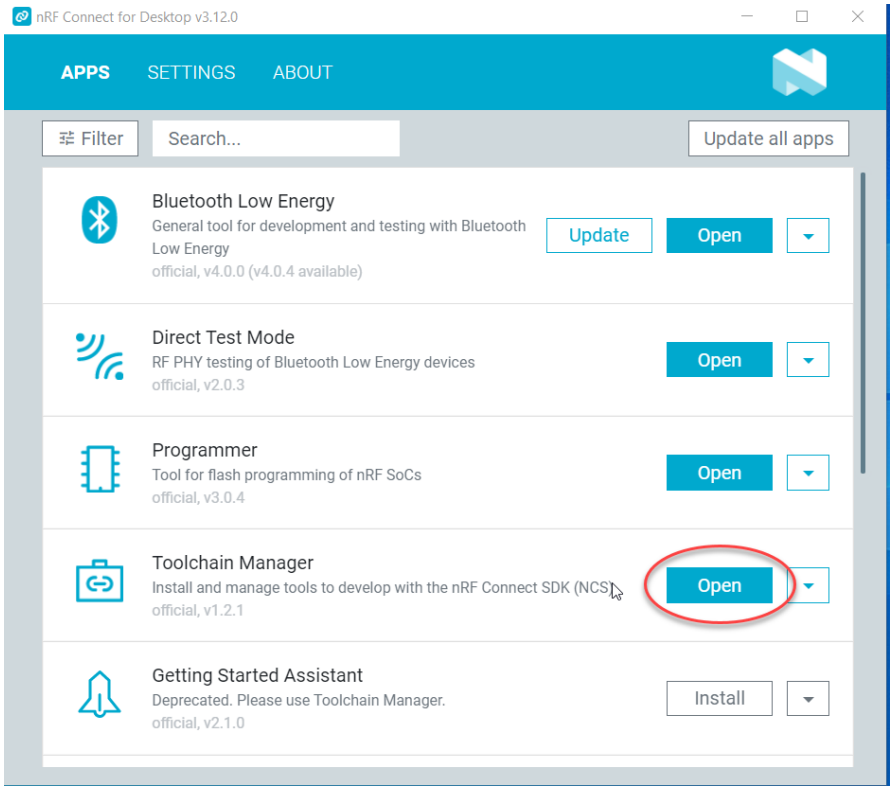


Figure 5: nRF Connect for Desktop

5. Install nRF Connect SDK. At the time this Application Note was written the latest version available is nRF Connect SDK v2.2.0.

Note: During the Installation process, a pop-up option to *Confirm the installation directory* appears. If changing the recommended path avoid long file names and ensure the path is kept as close to the root directory of your system as possible. Also avoid using names with spaces.

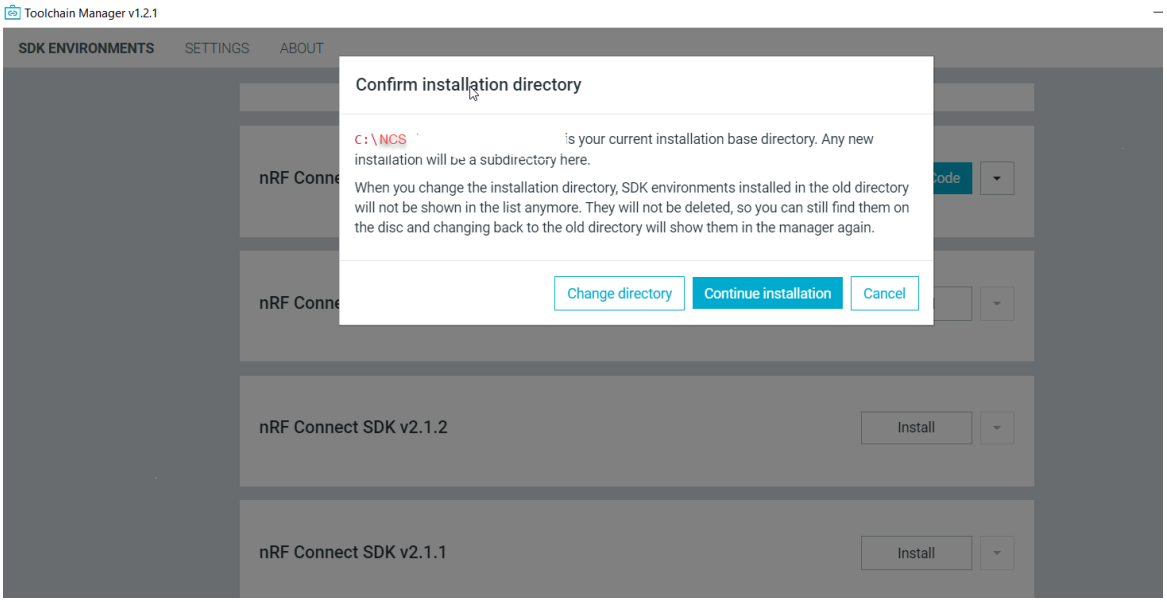


Figure 6: nRF Connect SDK Install Directory

Note: When selecting Open VS Code, the Toolchain Manager will automatically detect if VS Code was previously installed on your system. If was not already installed, select the Install VS Code option. When selecting Install VS Code, you will be automatically be taken to the VS Code web page where can download and install for your system.

6. Once the SDK is installed, select **Open VS Code**.

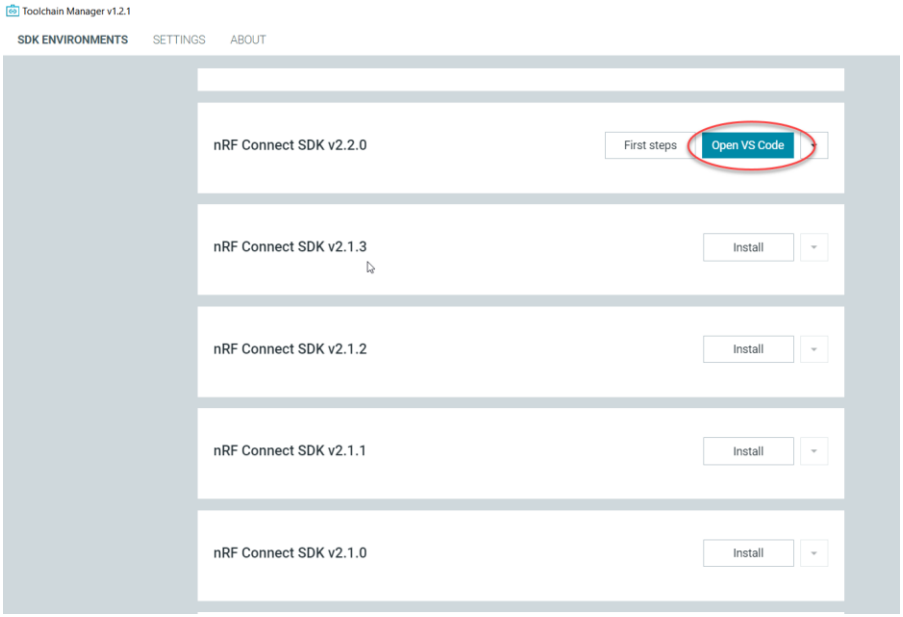


Figure 7: Open VS Code in Toolchain Manager

7. Within VS Code, open the **nRF Connect for VS Code Extension pack**. This can be done by selecting the **Extensions** tab on the left side of VS Code and then selecting the ***nRF Connect for VS Code Extensions Pack***. By typing "nRF" in the search window above the list of extensions, the list will be filtered to nRF related extensions, as shown below.

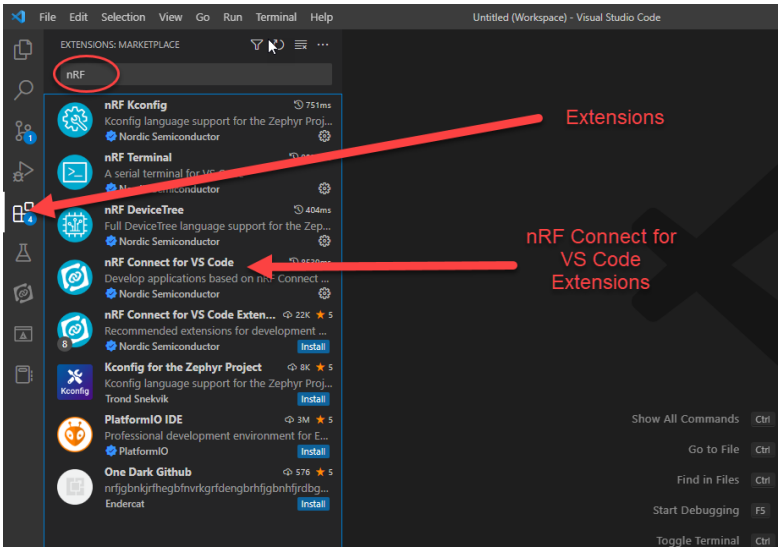


Figure 8: Adding nRF Connect for VS Code Extensions

8. An alternative way to load the required extensions is to close this instance of VS Code, then reopen nRF Connect in the Toolchain Manager and select **Open VS Code**. The Toolchain Manager will detect if any extensions or dependencies are missing. These can then be loaded by selecting Install missing extensions, as shown in Figure 9.

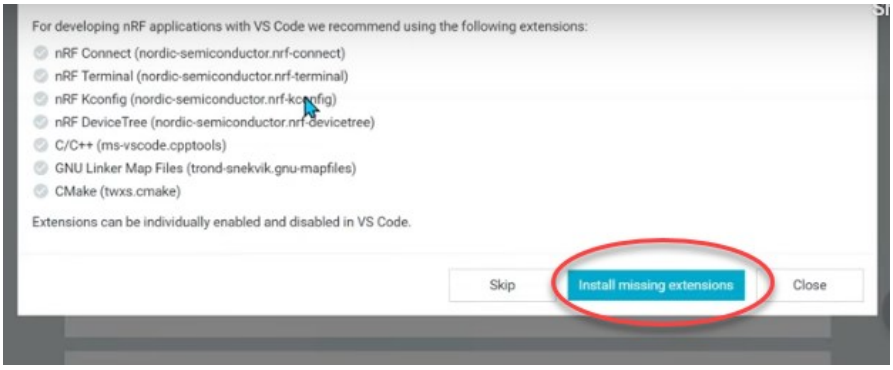


Figure 9: Tools Manager Install missing extensions

9. When the extensions installation has finished, reopen VS Code by selecting the Open VS Code tab.

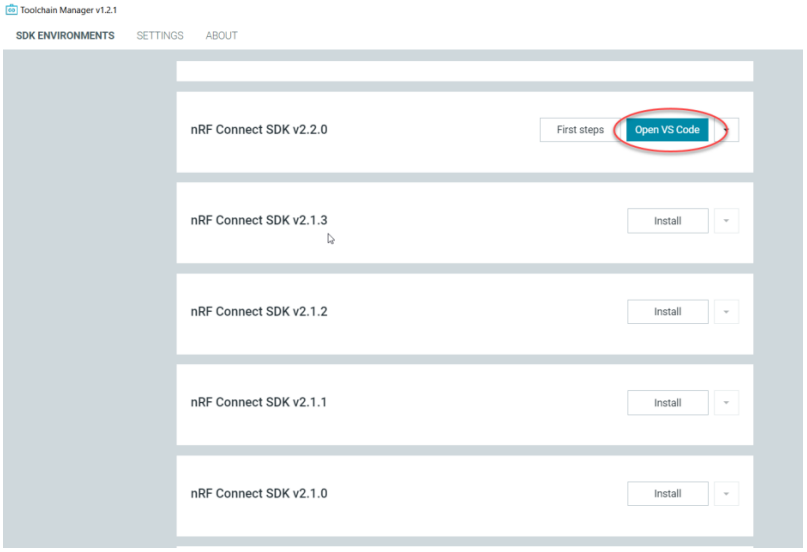


Figure 10: Reopen VS Code in Toolchain Manager after Extensions are added

10. Next select the nRF Connect SDK and toolchain versions from the Welcome to nRF Connect window.

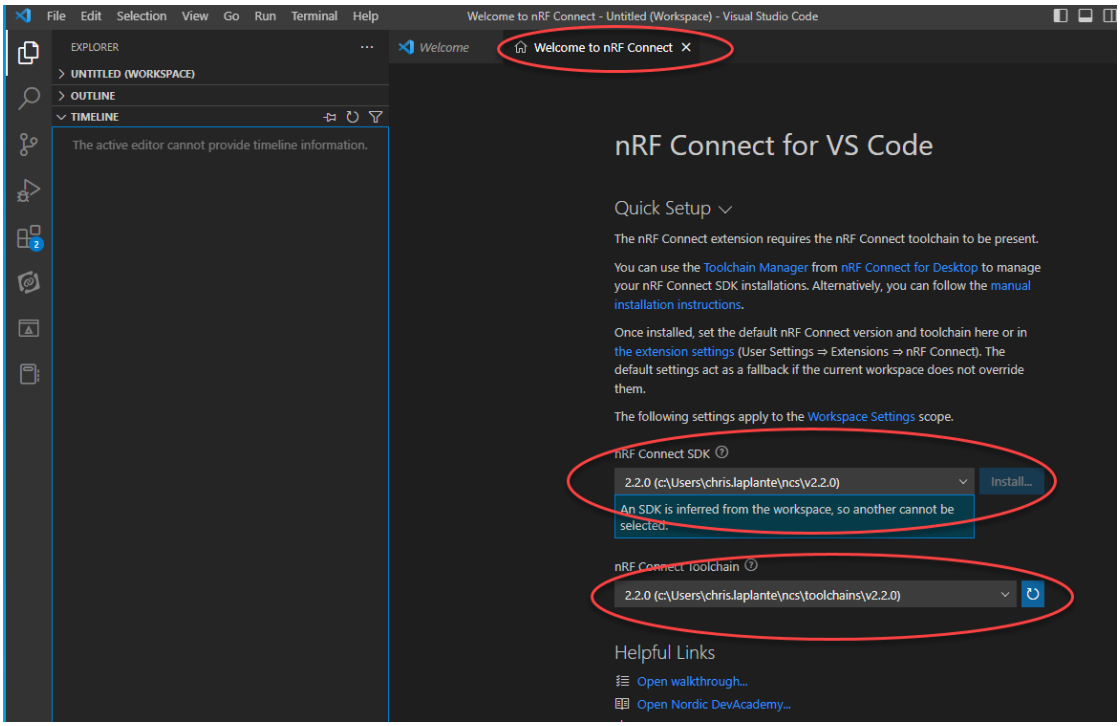


Figure 11: VS Code Toolchain Versions

4 Starting a New Application in VS Code

This section will walk through the steps required to begin a new nRF Connect project in VS Code.

1. In the VS Code environment select the nRF Connect tab on the left hand side. This will open several options such as *Add an existing application*, *Create a new application*, etc. Choose ***Create a new application***.

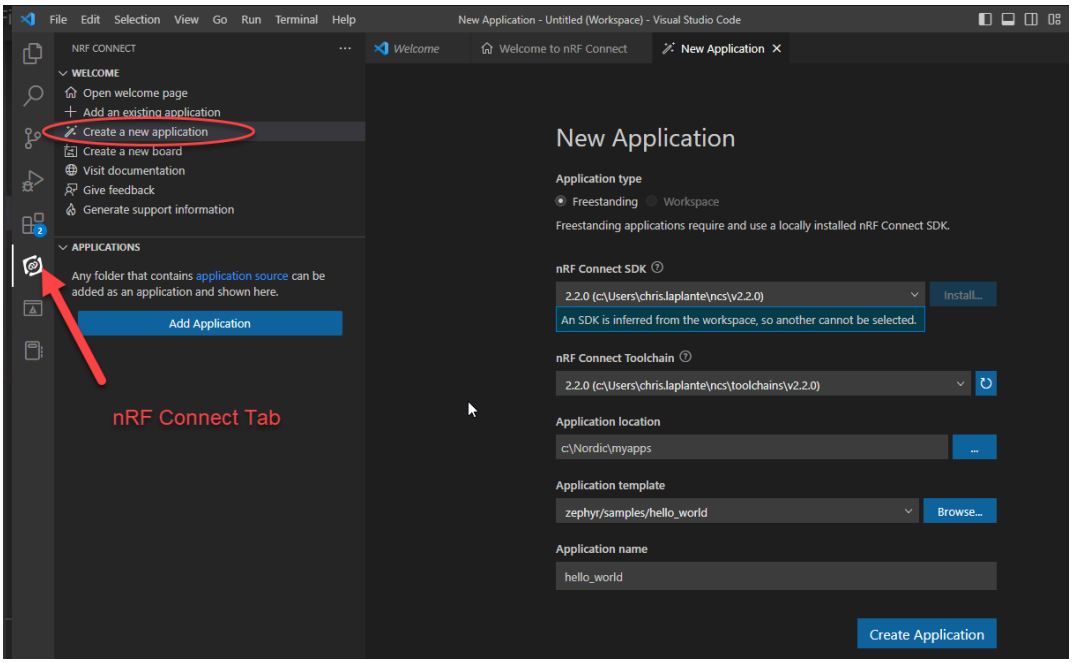


Figure 12: VS Code Create a New Application

2. The next step is to select the location for where the project will be stored. Again, keep this close to your system root directory. Open one of the nRF Connect SDK examples by selecting the Browse tab under the Application Template window.

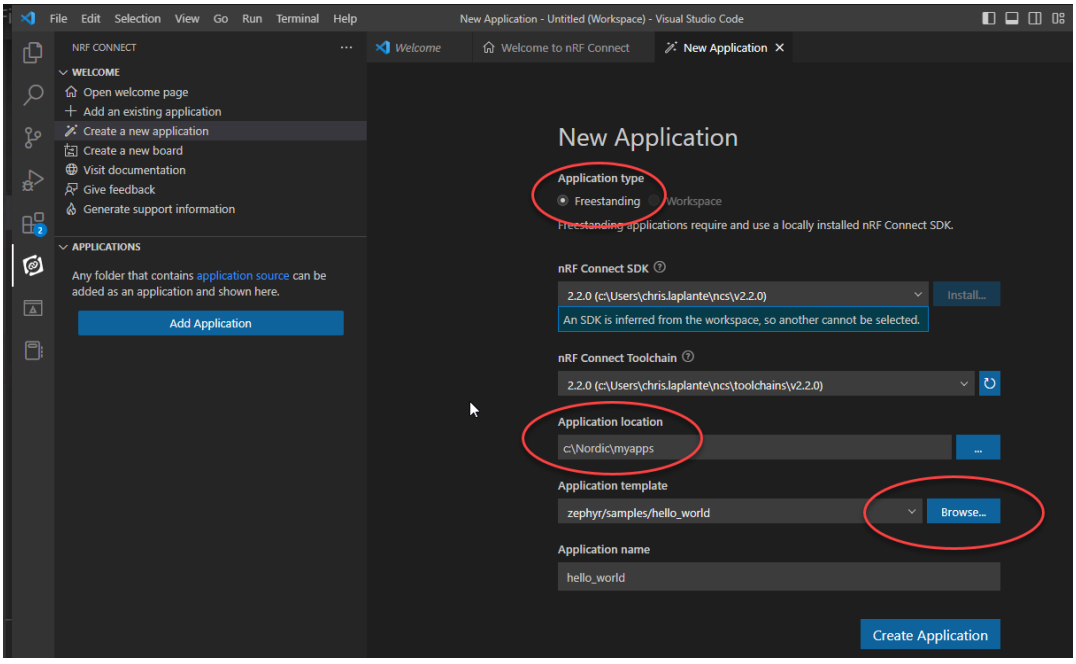


Figure 13: VS Code Select Workspace Location

Note: Ensure Freestanding is selected as this will use the SDK version and the files already in the file system loaded onto your machine. Workspace is used for long term development with many users and useful for version control.

3. When the ***Application*** tab is selected, you can find the desired example project by selecting the filter button in the search window. You can also browse the SDK directories themselves by selecting the ***Open in Browser*** tab.

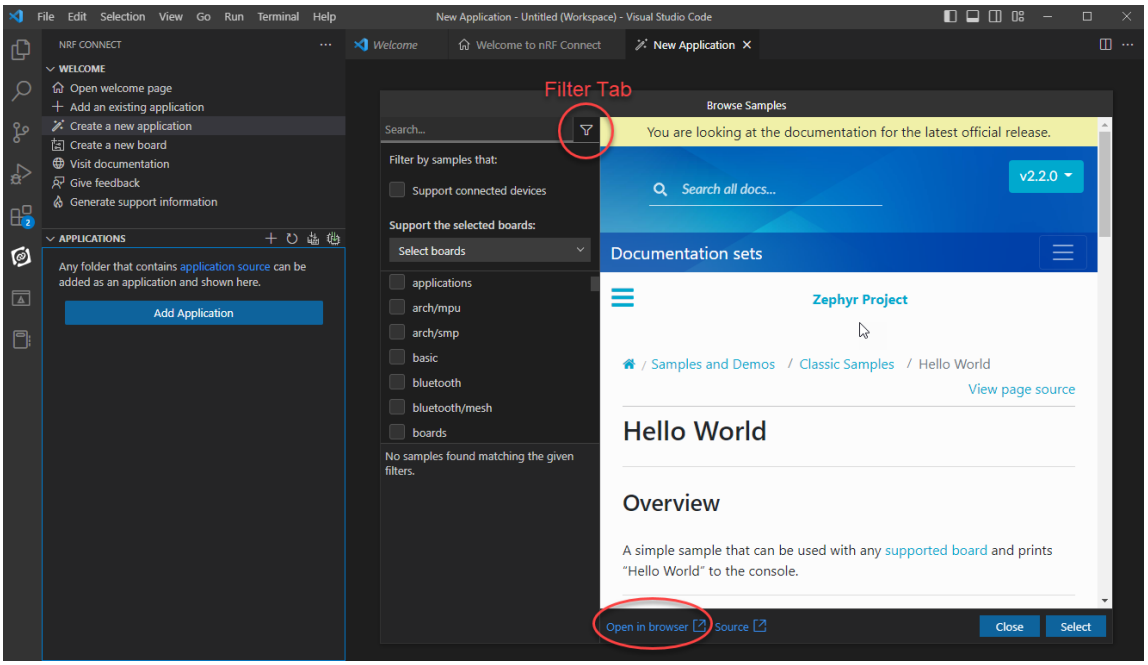


Figure 14: VS Code Finding nRF Connect SDK Applications

4. Searching for an SDK project via the Filter tab, we will next open the Bluetooth examples to search for the peripheral_hr sample. When peripheral_hr is found and highlighted press the ***Select*** tab in the bottom right.

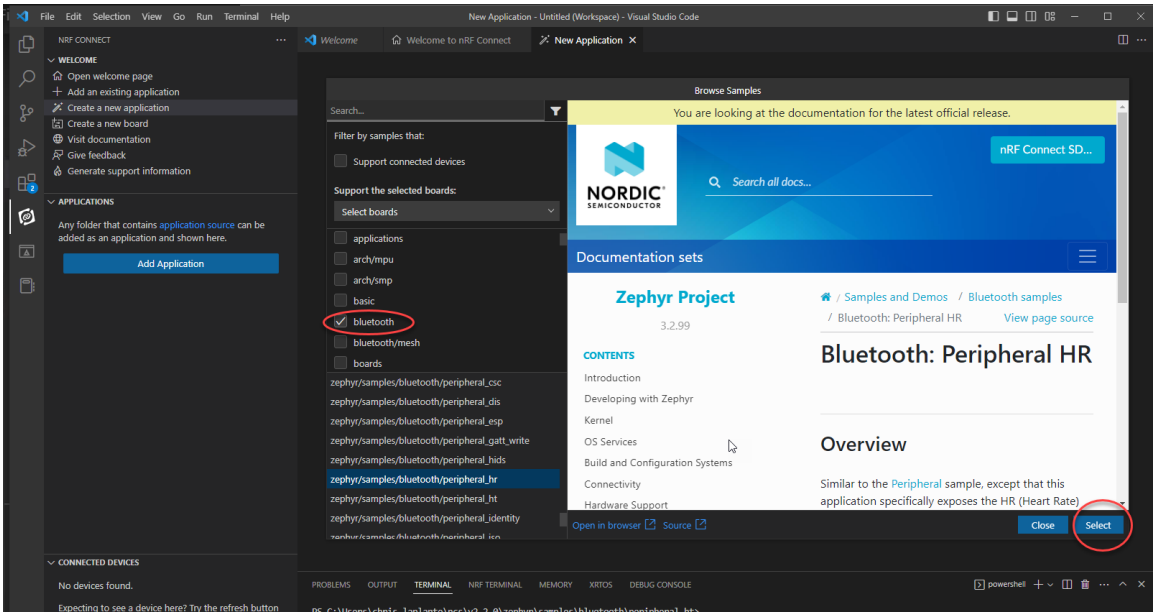


Figure 15: VS Code Select peripheral_hr application

5. Now select ***Create Application*** tab.

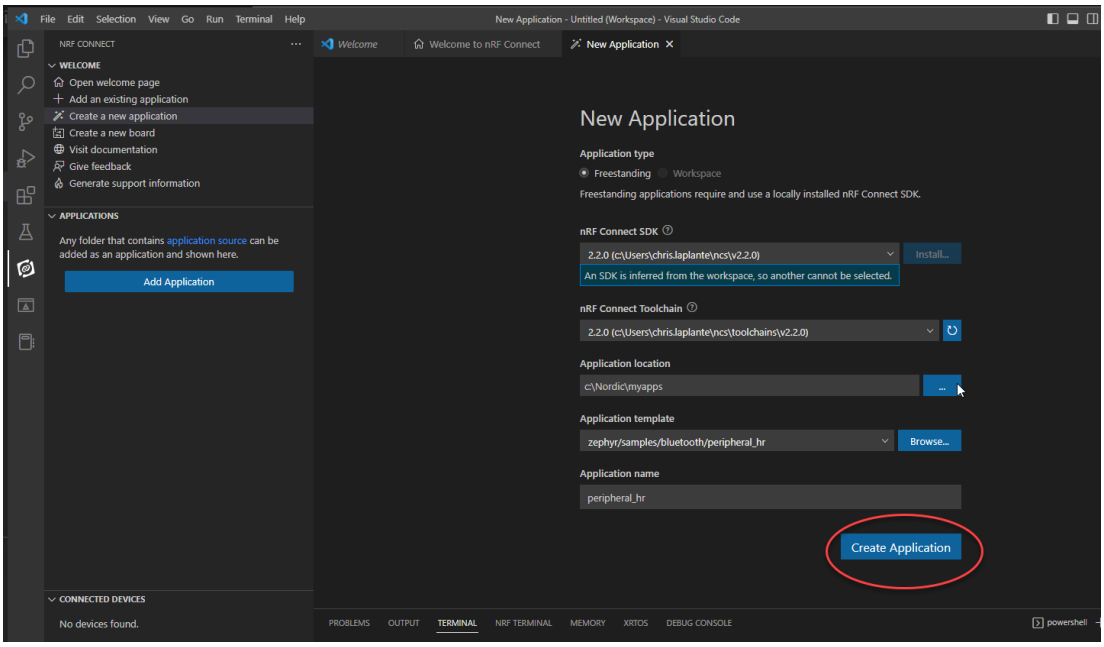


Figure 16: VS Code Create Application Tab

5 Building SDK Examples in VS Code

Now with the VS Code environment and dependencies installed, and the peripheral_hr application created, the next step is to build the application as per the following.

1. The first step is to create a build configuration. Ensure the ***nRF Connect SDK*** tab on the left side is selected. This displays the Welcome option, applications created and connected devices if any.
2. Under the ***peripheral_hr*** application, select the ***No build configurations*** tab. Clicking the This will open the ***Add Build Configuration*** window which enables selection of the development board being used, enabling debug options and the option to build after the build configuration has been processed.

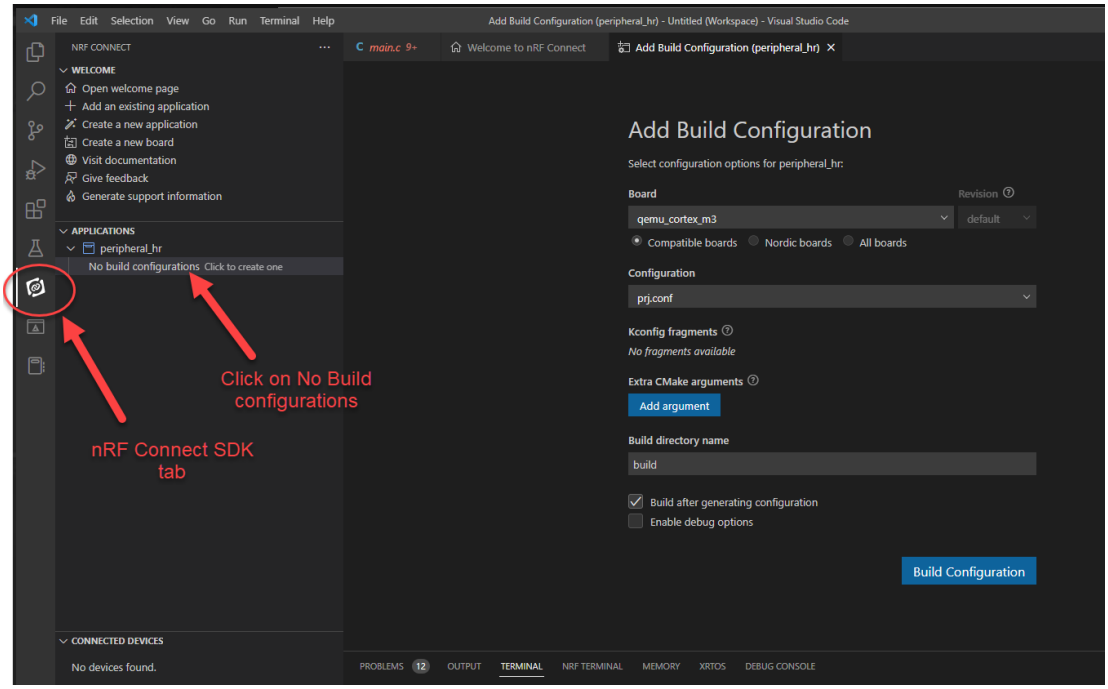
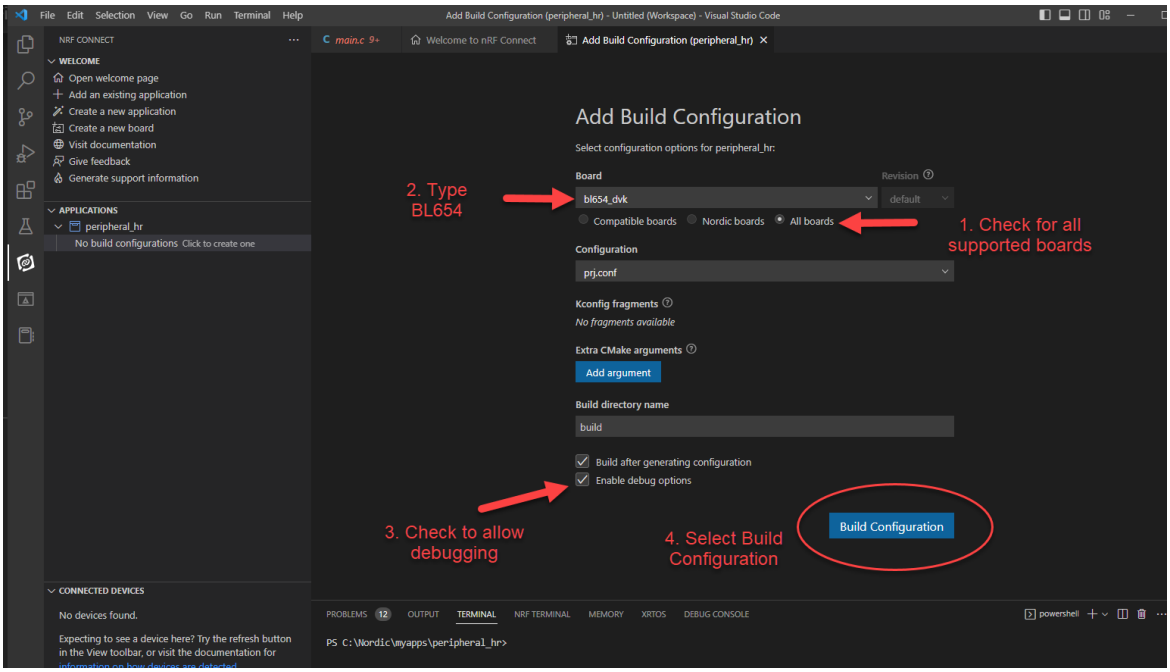


Figure 17: VS Code Build Configuration

3. Check the ***All Boards*** option below the ***Board*** field to allow a search of all supported development boards.
4. In the Build window type BL654 to search for all BL654 development options. Select BL654_dvk. (If working with a different BL65x_DVK, be sure to search for and select the DVK for that board.)
5. Check ***Enable Debug*** button to allow debugging of the application.

6. Finally, click **Build Configuration**.



Note: Build after generating configuration option is checked by default. Leaving this option checked will allow the application to be built automatically after the Build Configuration tab is selected.

7. When the application has successfully built, the terminal window in VS Code displays the memory usage information.

Figure 18: VS Code Terminal Window Output

Note: The Terminal Window may be hidden by default. By locating the ↕ arrow at the bottom of the VS Code window, and pulling up, the Terminal Window can be displayed.

6 Flashing the HRS example in VS Code

Connect the BL654 to the PC through USB2 port on the development kit (Figure 19).

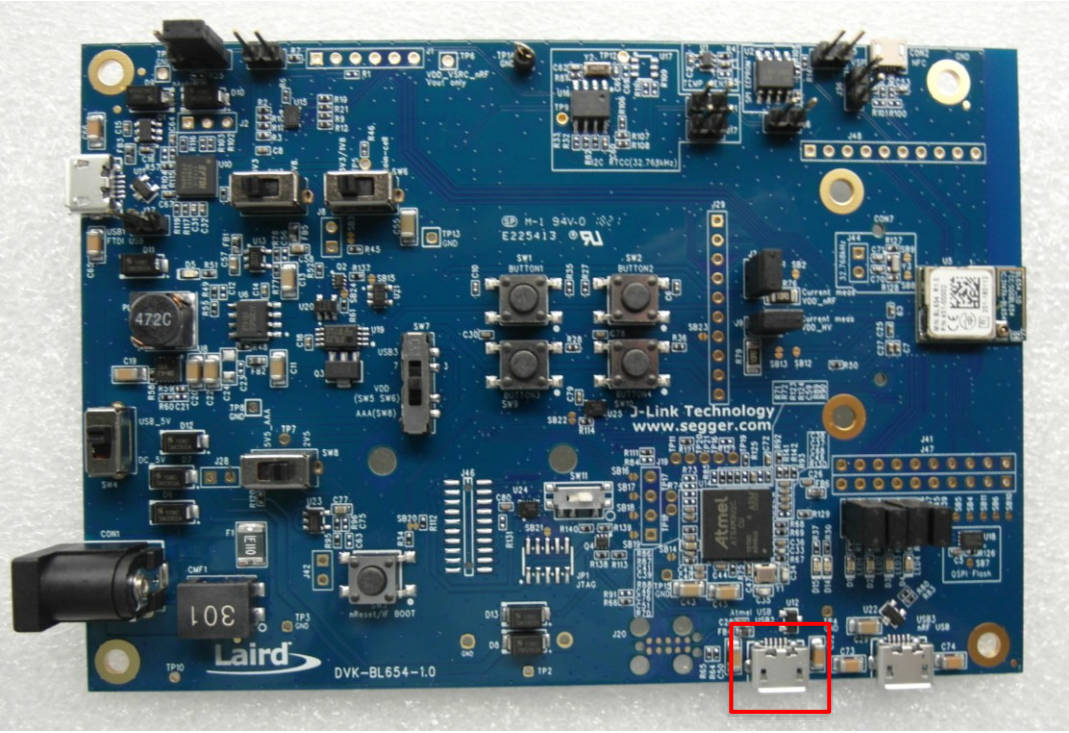


Figure 19: Connecting BL654 to PC through USB2 port

Note: For other DVK-BL65x boards please refer to the DVK-BL65x User Guide for the location of the USB-SWD (JTAG) interface.

With the BL654 DVK plugged into the USB port on your host machine VS Code will automatically detect the DVK in the Connected Device window. Simply click on the ***Flash*** tab in the Actions window to program the ***peripheral_hr*** application to the DVK.

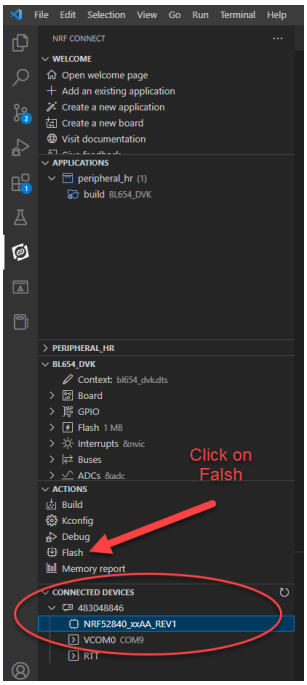


Figure 20: Flash program

7 Debugging the HRS Example

Debugging an application on the BL65x allows you to see what is going on inside the program while it runs. This can help detect problems and issues, as well as show the values of the variables and the flow of the program at different stages. In order to debug the ***peripheral_hr*** application, complete the following steps:

1. Because ***Enable debug options*** was selected during the build configuration, the application is debug friendly. Therefore, to run the application with debug, select ***Debug (Launch build)*** from the ***nRF Connect*** tab, **Actions** window.

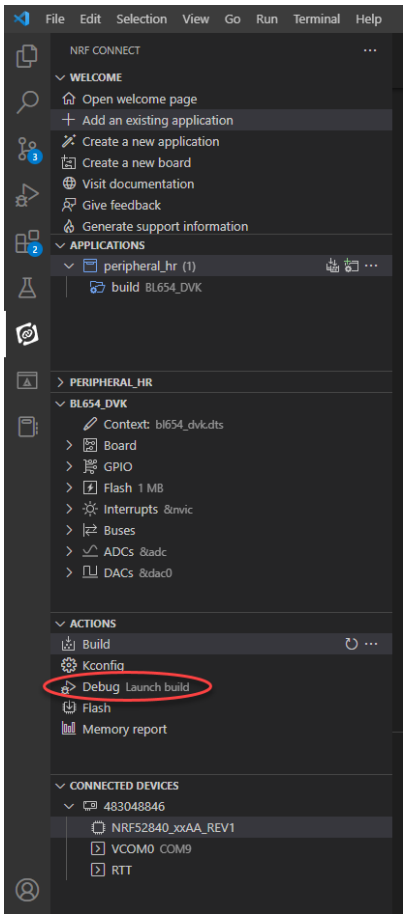


Figure 21: Begin Debug Session

2. You should now be in the debug view. The debug windows allow visibility into CPU registers and variables, a Watch window monitor variables in the code, Breakpoints (as shown in Figure 22 Breakpoints window is expanded but empty given no Breakpoints are set) and peripheral registers.

Note: If the debug windows do not pop up automatically then simply select the *Run and Debug* Tab in the upper left as shown in Figure 22.

---

3. You can step over, step into, and add breakpoints through the Run from the toolbar at the top of the VS Code Debug environment window. Press Go as shown in Figure 22 to run the program.

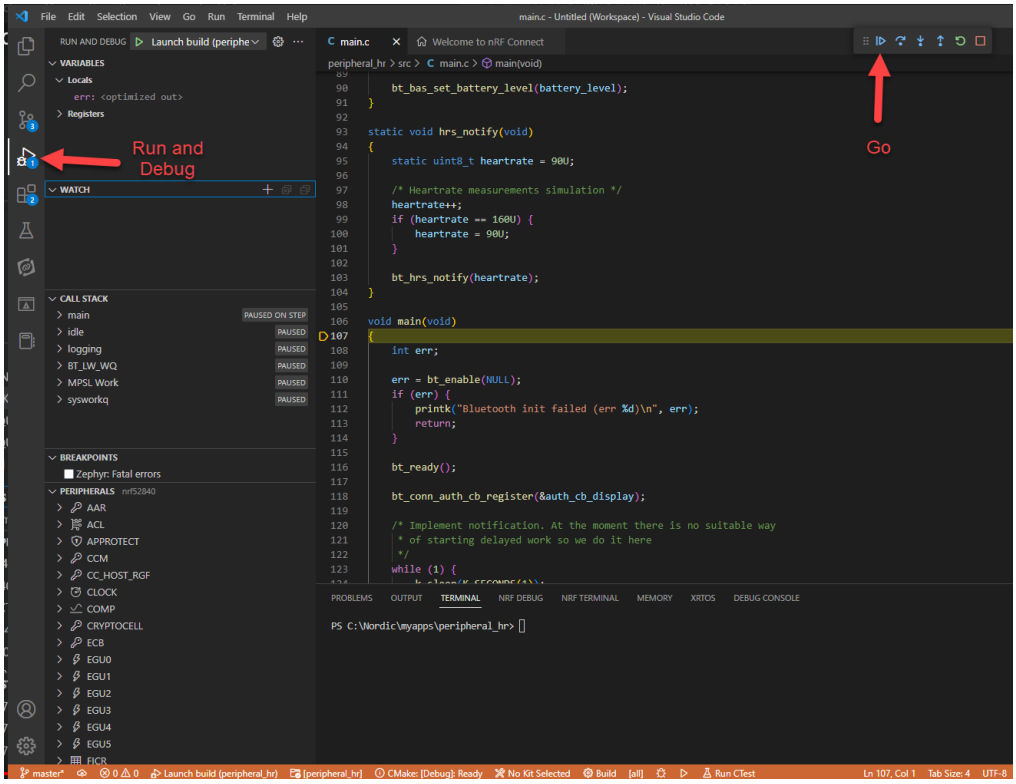


Figure 22: Running the Program

4. Set a Watch point for the variable battery_level by right clicking on it and selecting Add to Watch.

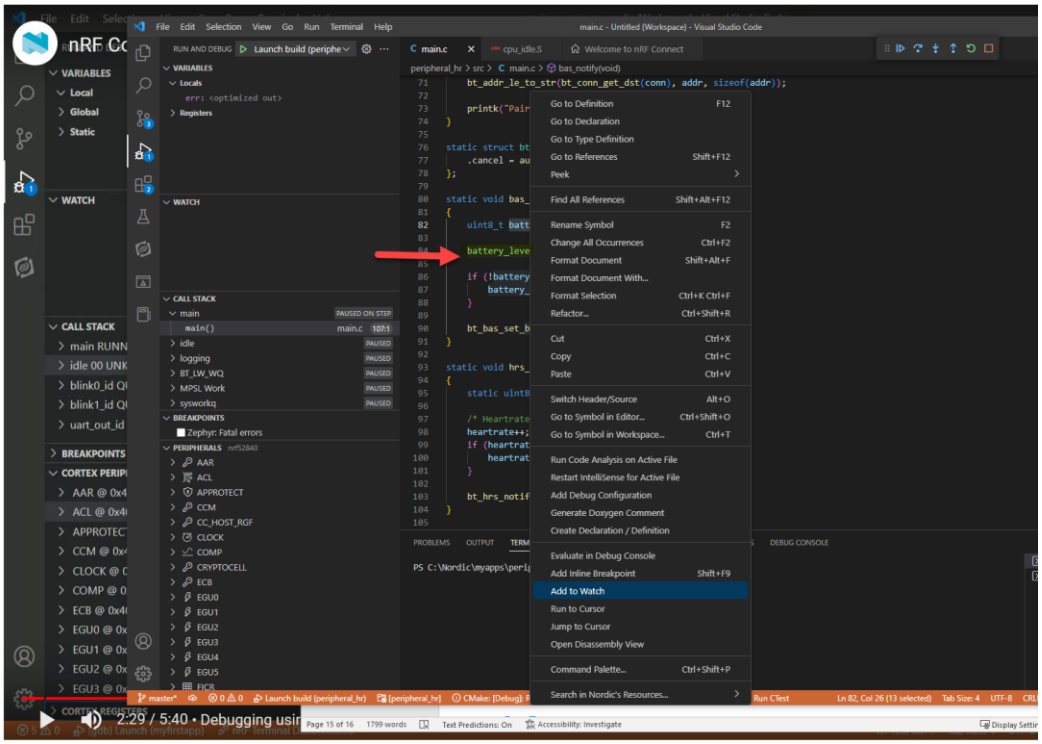


Figure 23: Add a watchpoint

4. Set a breakpoint at function where updated battery level is set. Note this now appears in the Breakpoints window.
5. Start the program again and when the breakpoint is hit the application will stop and the battery level is displayed in the Watch window.

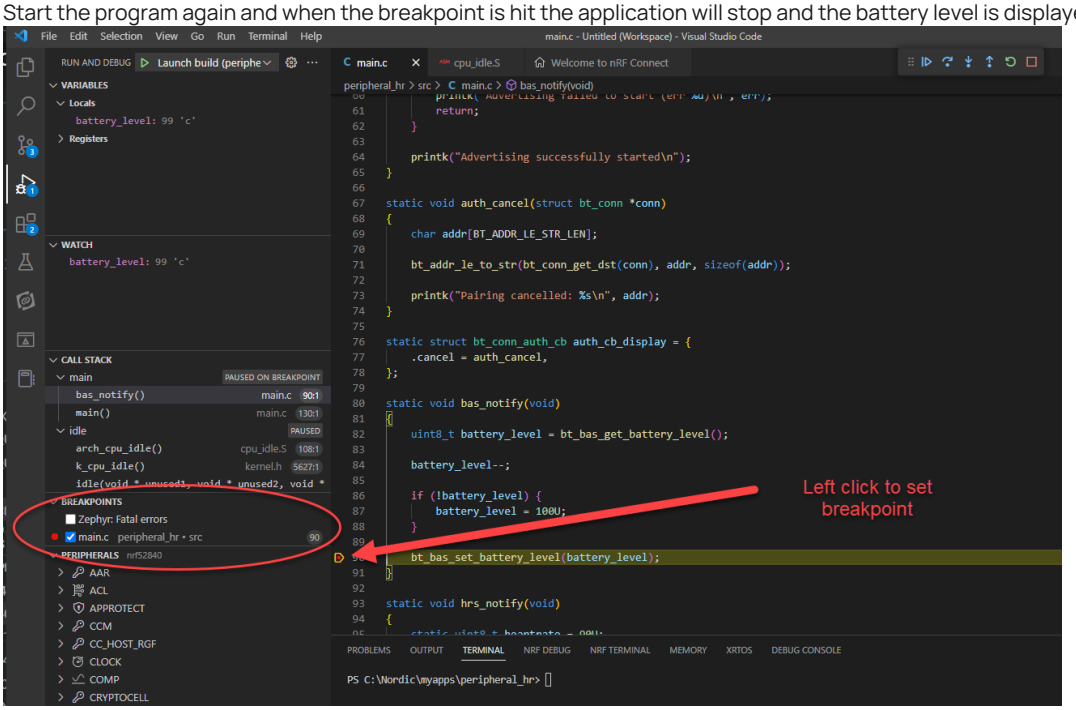


Figure 24: Set Breakpoint and Observe Watchpoint

8 Zephyr Project Settings

Software features supported in an application are defined in the Kconfig file that is included in the application example. Features defined in Kconfig can be overridden in the prj.conf file. The next steps will provide examples of how a developer can enable features from Kconfig in the prj.conf file.

Note: Previously, when using the older nRF5 SDK with Ezurio BL65x DVKs, the default system clock was configured for the optional external xtal. However, by default, the BL65x DVKs do not have the external xtal connected, therefore the SDK configuration needed to be set accordingly in sdk_config.h file to use the onboard RC Oscillator as the clock source. With nRF Connect SDK, the BL65x_DVK board files now correctly configure the application to use the onboard RC Oscillator.

8.1 32K Reference Clock Configuration

In this step, for demonstration purposes, we will show how the clock configuration can be set in Kconfig or prj.conf.

1. Select the three dots to the right of Kconfig in the Actions window.

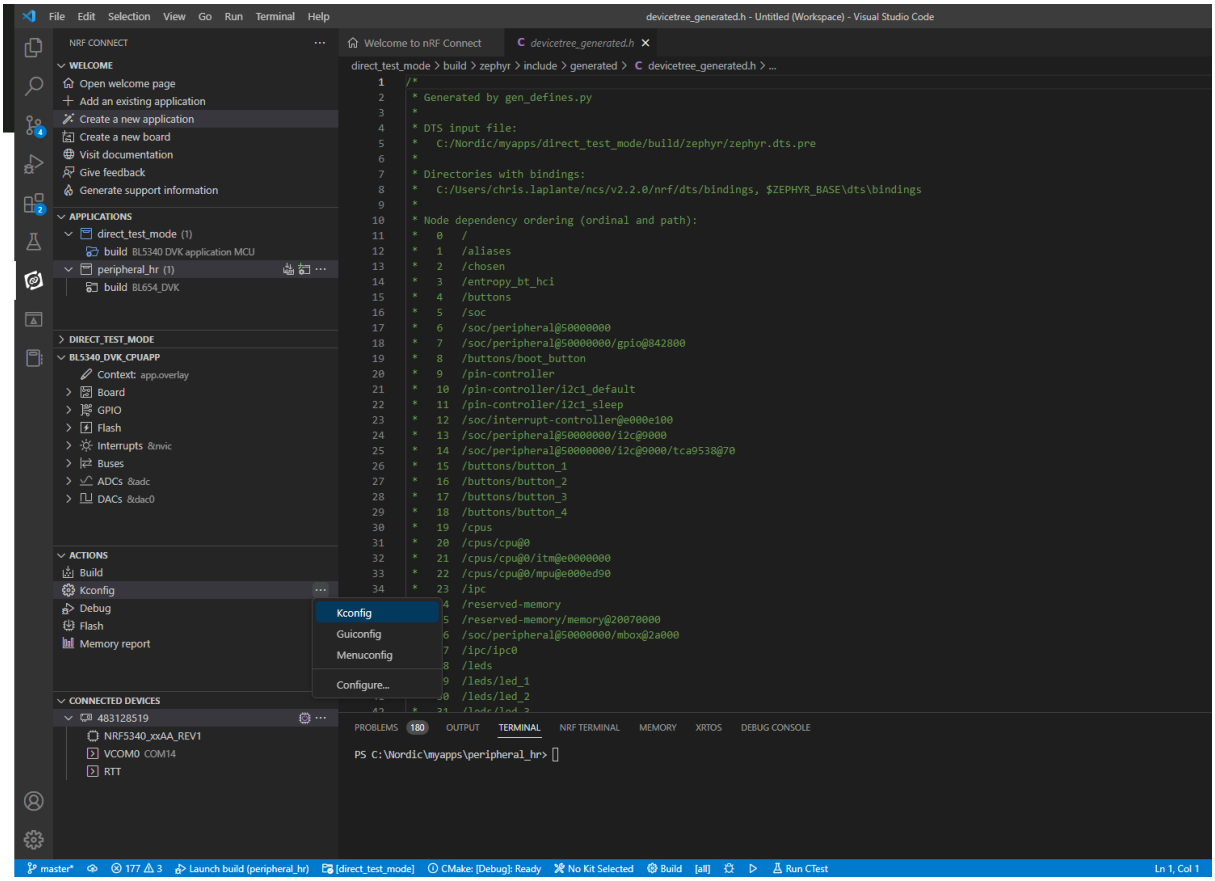


Figure 25: Kconfig

2. In the search window at the top, type clock. Then scroll to the 32MHz clock source
3. Click on the Information button on the far right of 32KHz source.
4. Click on jump to item to go to where the clock can be configured in the Device drivers.

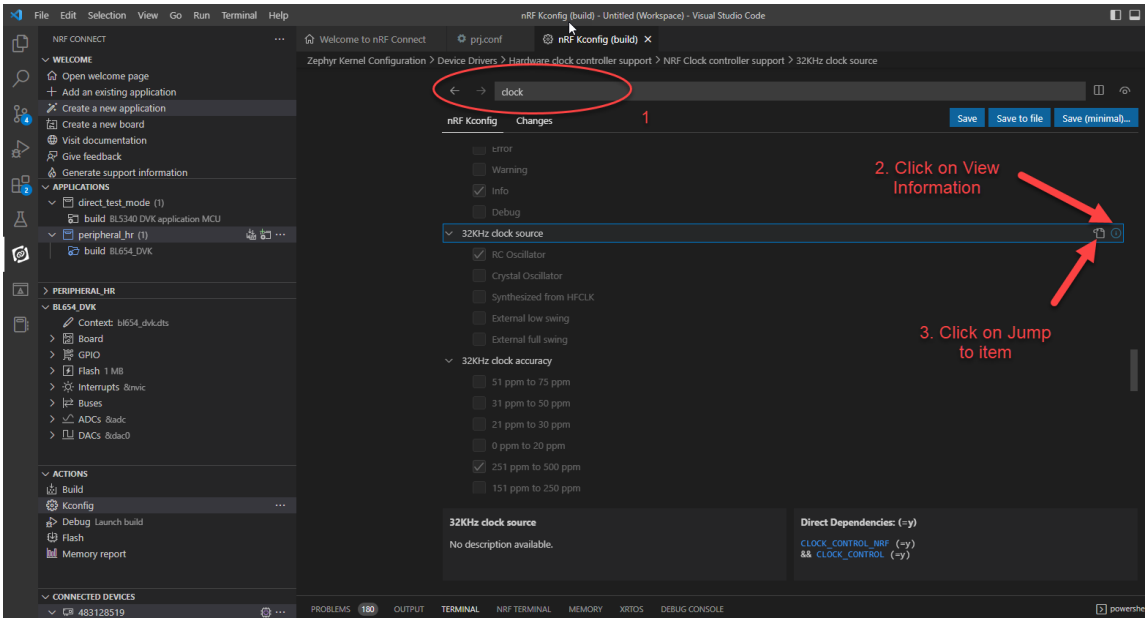


Figure 26: 32KHz clock source

5. Click on View Information to the right of Crystal Oscillator.
6. Notice the name of the Crystal oscillator and its dependencies. If clock config is added to prj.conf these are the configurations that need to be added.
7. For configuration using external crystal oscillator in Kconfig simply check it to select it as shown in Figure 27.

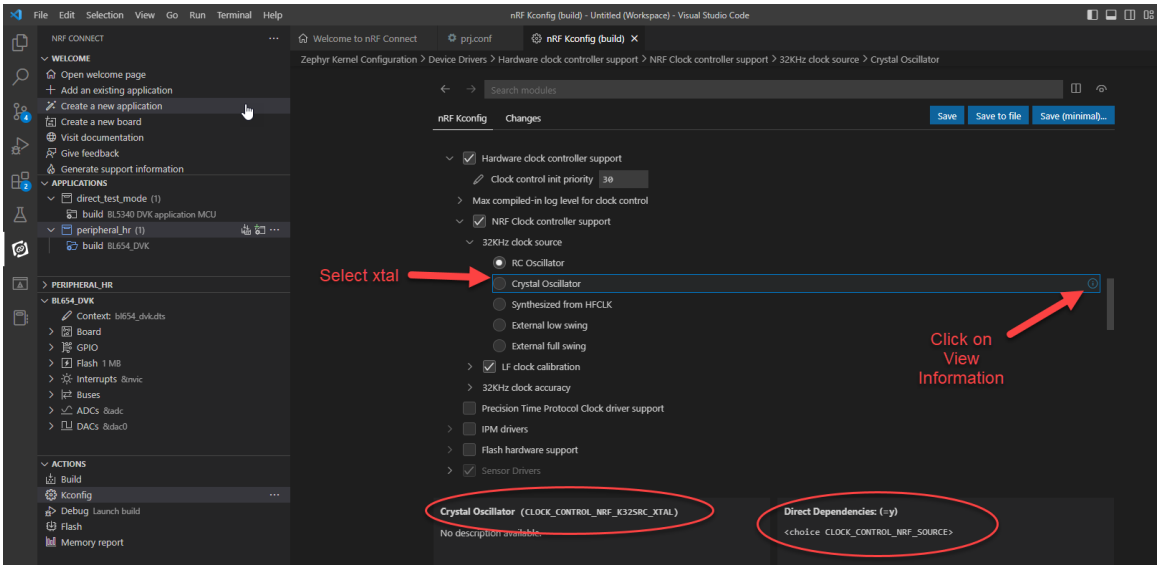


Figure 27: Change 32K Source clock in Kconfig

8. There is also an option to configure the accuracy of the 32KHz clock. Note for Internal RC this is configured for 251ppm to 500ppm. View the information for 31ppm to 50ppm to see the accuracy name. For external xtal this will also be added to prj.conf.

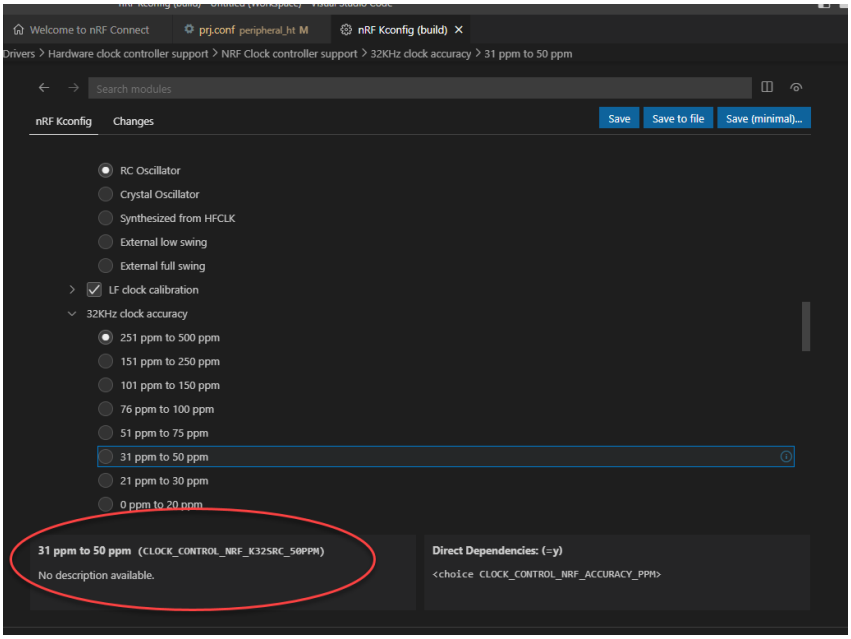


Figure 28: 32K Clock Accuracy in Kconfig

9. Save the configuration and rebuild the application.
10. Alternatively, the clock configuration can be set in prj.conf. Given the clock defaults in the application to use the internal RC Oscillator let's assume that an application will need to use the optional external crystal as the 32K reference and configuring in prj.conf will allow the change for the project here.
11. Open prj.conf and add the name of the external crystal oscillator as found in Kconfig in Figure 27 and the clock accuracy as found in Figure 28.

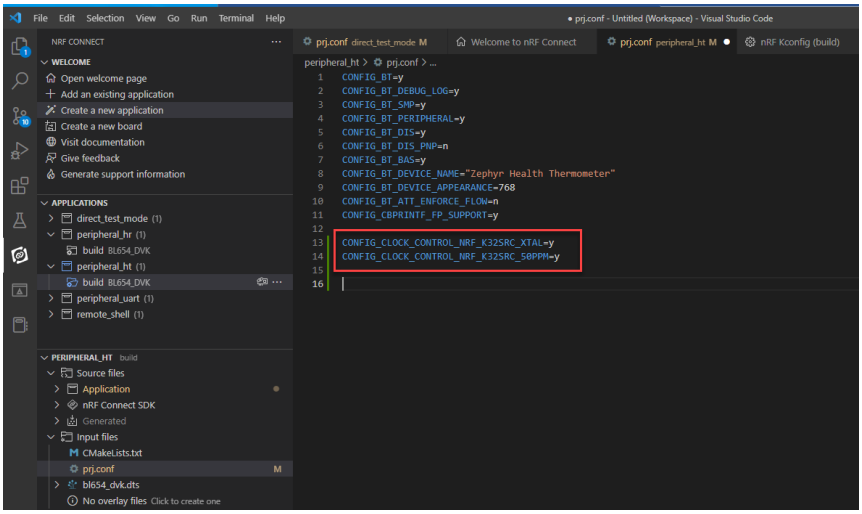


Figure 29: 32K Clock Source in prj.conf

12. Build the application again and flash to board.

Note: This application will not run successfully on the BL65x DVK with external 32KHz xtal configuration. Using the external crystal oscillator was done for demonstration purposes only. To use external crystal oscillator on the BL65x DVK connect as per the DVK User Guide.

8.2 Project Debug Options in prj.conf

Recall previously that the Enable Debug option was selected when setting build configurations. When the build begins you can see from the terminal output that the underlying west build command incudes debug information. With Debug option enabled in build configurations, these debug configuration flags will now allow the application to be debugged in VS Code.

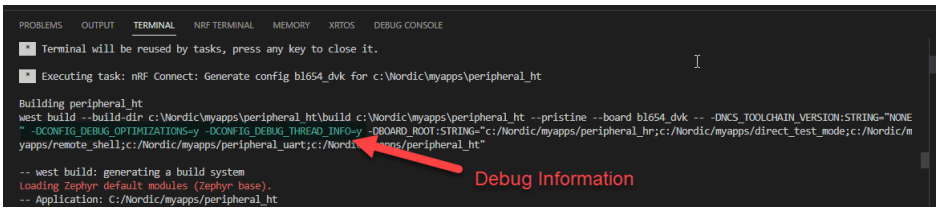


Figure 30: Build Debug Option Flags

1. Open prj.conf to add debug optimizations to the project configuration.

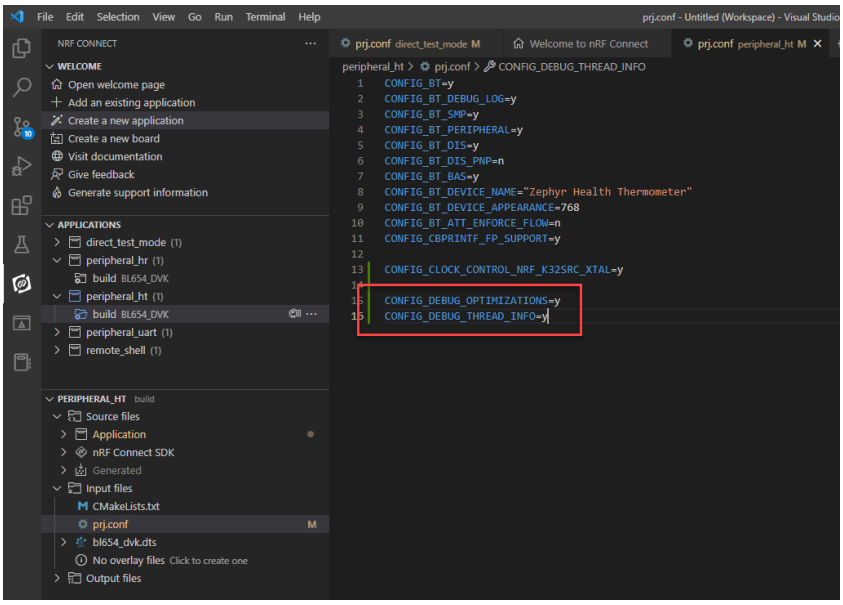


Figure 31: Debug Config Option in prj.conf

2. Now open the Edit Build Configuration and build the application with Enable debug options unchecked.

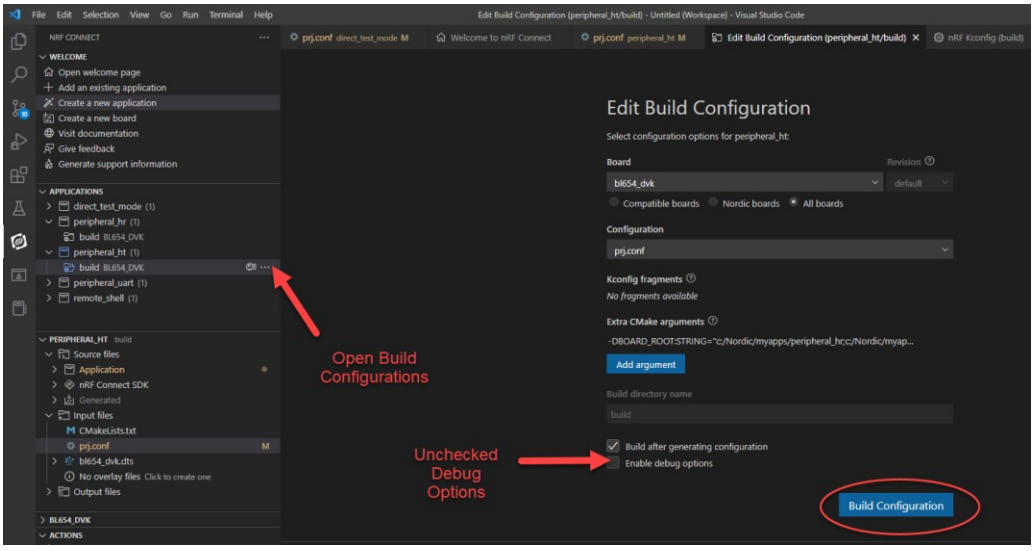


Figure 32: Build Configuration with no Debug Option Checked

3. Now click on Debug in the Actions Window. The program is now flashed onto the DVK, the debug view windows are now exposed and the program breaks at main().

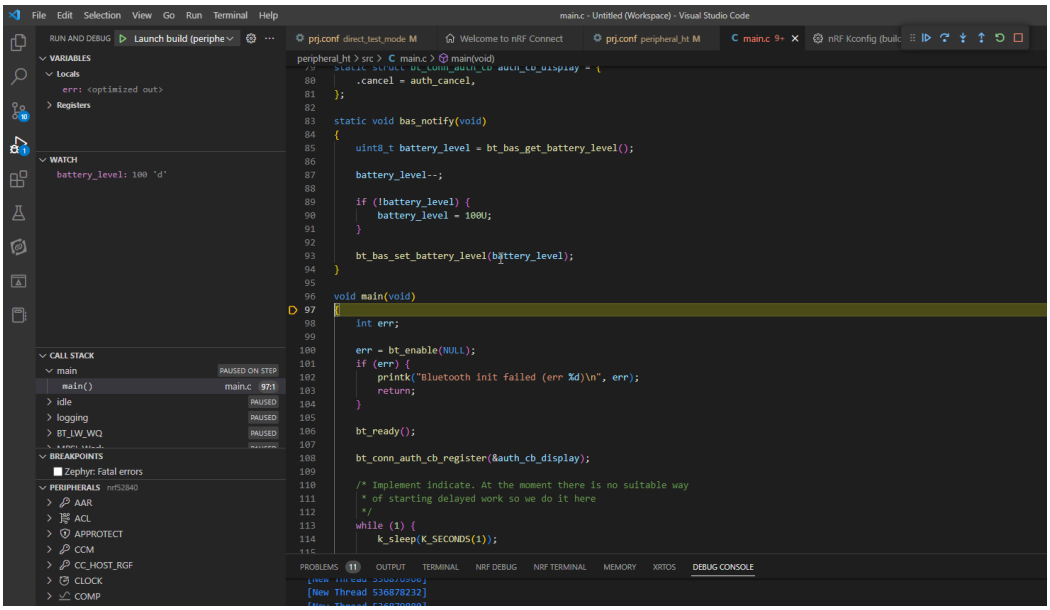


Figure 33: Program Debug Window

9 Reference

Further information relating to different utilities used in this app note can be found here:

- BL654 Product Page - https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl654-series-bluetooth-module-nfc
- Zephyr Kconfig search - https://docs.zephyrproject.org/latest/kconfig.html#CONFIG_CLOCK_CONTROL_NRF_SOURCE
- Nordic Kconfig search - https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/kconfig/index.html#CONFIG_NCS_SAMPLE_REMOTE_SHELL_CHILD_IMAGE
- nRF Connect for VS Code Tutorials - https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-VS-Code
- Visual Studio Code - https://code.visualstudio.com/
- nRF Connect v2.2 SDK - https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/index.html
- nRF Connect for Desktop - https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop
- nRF Connect Command Line Tools - https://www.nordicsemi.com/Products/Development-tools/nRF-Command-Line-Tools

10 Revision History

Version	Date	Notes	Contributor(s)	Approver
1.0	17 Feb 2023	Initial Release	Chris Laplante	Jonathan Kaye
2.0	3 Mar 2025	Converted to Ezurio format	Sue White	Dave Drogowski