NuEclipse
User Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

List of Figures

1 Introduction

The **NuEclipse** is designed for cross-platform embedded ARM development. It includes a series of Eclipse plug-ins and tools. The plug-ins allow the user to create, build, and debug ARM-based projects within the Eclipse framework. Its features are listed below:

- **Creating projects by the New Project Wizard**: The New Project Wizard provides several templates for different target chips.
- **Building projects by the GNU ARM Toolchain**: The toolchain contains the ARM Embedded GCC compiler. The user can use it to build projects without restriction.
- **Debugging projects by GDB**: The user can halt, step, run, and monitor target chips. Accessing memory and flash is allowed. Setting hardware breakpoints and watchpoints is supported. In addition, the user can erase target chips and program the user configuration.

Through the **NuEclipse**, the user can develop projects of the NuMicro® Family within the Eclipse framework.

2 System Requirements and Installation Guide

2.1 System Requirements

The following table lists system requirements for the user to run the **NuEclipse**.

	Minimum Requirements	Recommended Specifications
Operating System	Windows®7 x86_64 or GNU/Linux	Windows®10 x86_64 or Ubuntu 18.04 LTS
GNU ARM Embedded Toolchain	6-2017-q1-major	The latest version

Note: To have a fully usable and pleasant experience on Linux, the recommended Linux distribution is Ubuntu 18.04 LTS (64-bit).

2.2 Supported Chips

To see the list of supported chips, please refer to **Supported_chips.htm** in the folder of user manual.

2.3 Installation

To make the **NuEclipse** ready for work, perform the following steps based on your operating system:

1. Performing the NuEclipse installer on Microsoft Windows.
2. Extracting the NuEclipse tar file on GNU/Linux.

2.3.1 Performing the NuEclipse Installer on Microsoft Windows

On Windows, it is very easy to install the NuEclipse only by performing the NuEclipse installer. The installer will ask the user to install the **GNU ARM Eclipse Windows Build Tools** and **GNU ARM Embedded Toolchain** because they are required by NuEclipse.

Figure 2-1 NuEclipse Setup Wizard

2.3.2 Extracting the NuEclipse Tar File on GNU/Linux

On GNU/Linux, it is very easy to install the NuEclipse only by extracting the NuEclipse tar file. After that, **run the install.sh script** to complete the installation process. Please do not use the **sudo** command to run the script.

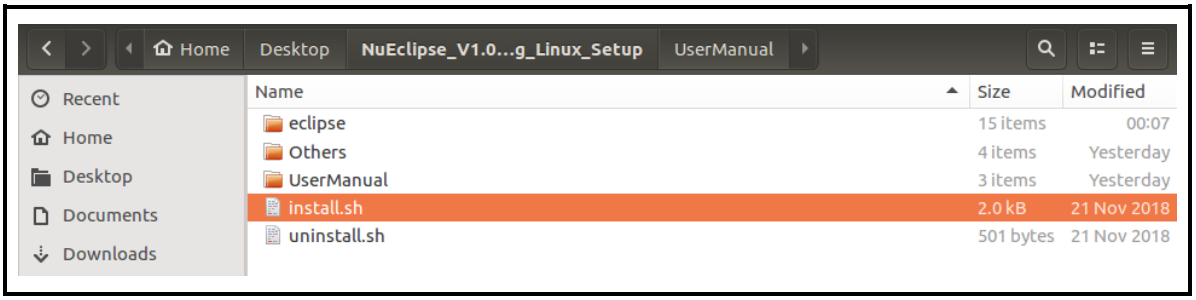

Figure 2-2 Install.sh Script

.

2.3.3 Verifying the Eclipse Preferences

After the installation, the Eclipse preferences are automatically written on Windows. To verify them, click **Window > Preferences**, the Preferences wizard appears. Go to **C/C++ > Build > Global Tools Paths** and make sure the Build tools and Toolchain folder be correctly configured to what the installer has installed in the previous step. Click the **Apply** button to take effect. On GNU/Linux, Build tools folder path is not required. The path should be empty.

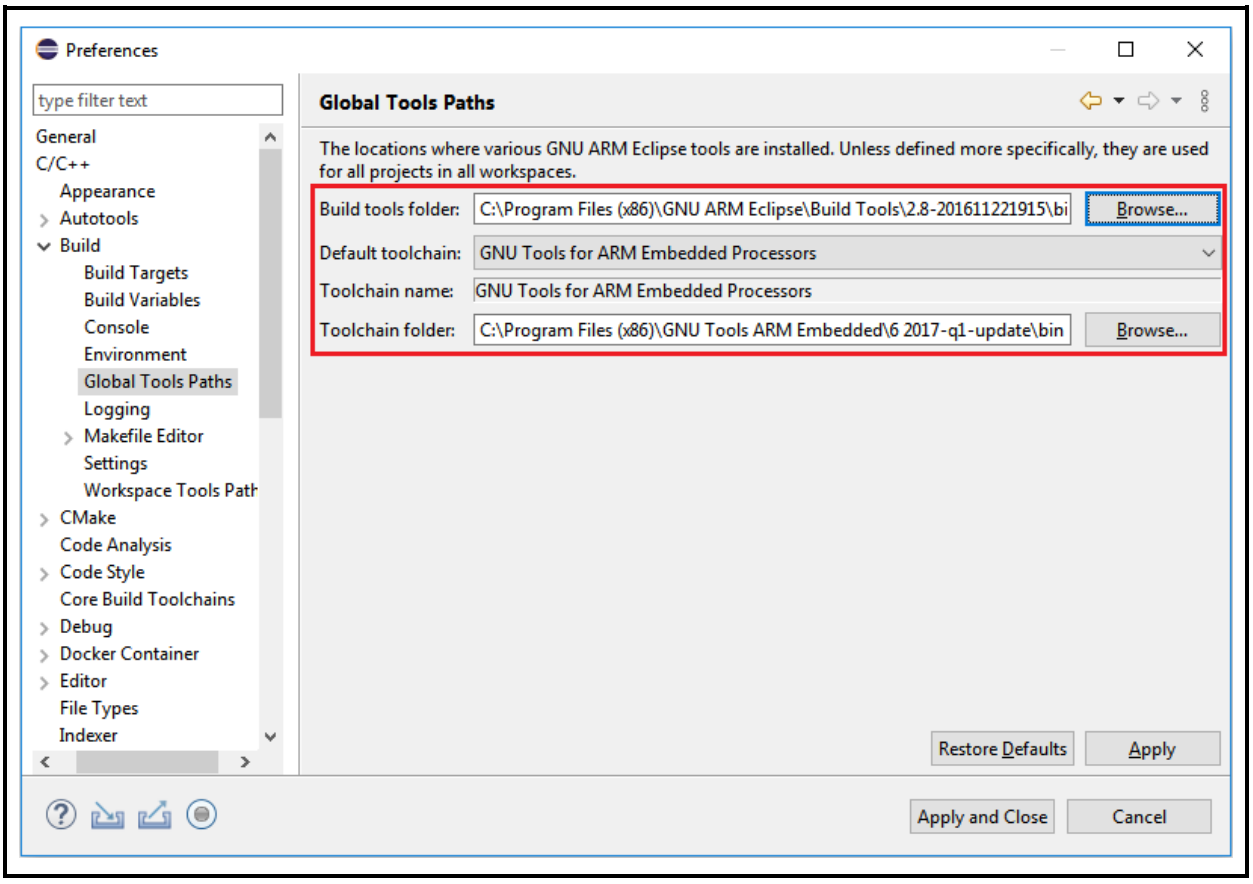

Figure 2-3 Preferences for Global Tools Paths

Subsequently, go to **Run/Debug > OpenOCD Nu-Link** and make sure the OpenOCD folder be configured to where the installer has put the OpenOCD executable. On Microsoft Windows, for example, the path of OpenOCD folder could be C:/Program Files (x86)/Nuvoton Tools/OpenOCD. Similarly, on GNU/Linux it could be /usr/local/OpenOCD. The OpenOCD executable provided by Nuvoton is customized for Nu-Link. If the user tries to use other OpenOCD executable, OpenOCD and Nu-Link may not cooperate well. Click the **Apply** button to take effect.

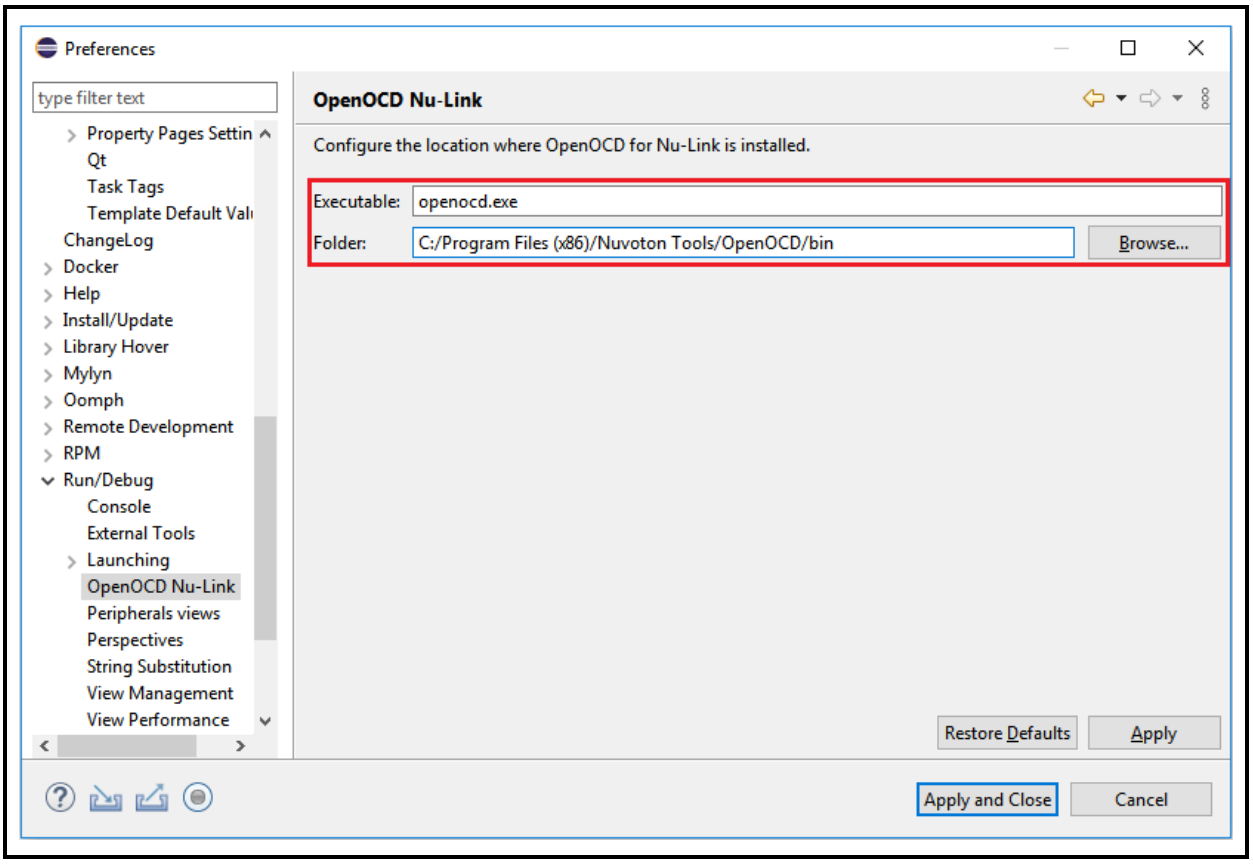

Figure 2-4 Preferences for OpenOCD Nu-Link

2.4 Running Eclipse

To run **NuEclipse**, double-click the **eclipse.exe**. Note that the .exe file and the related folders, such as the OpenOCD folder, should stay in the same directory; otherwise, the application will not work properly.

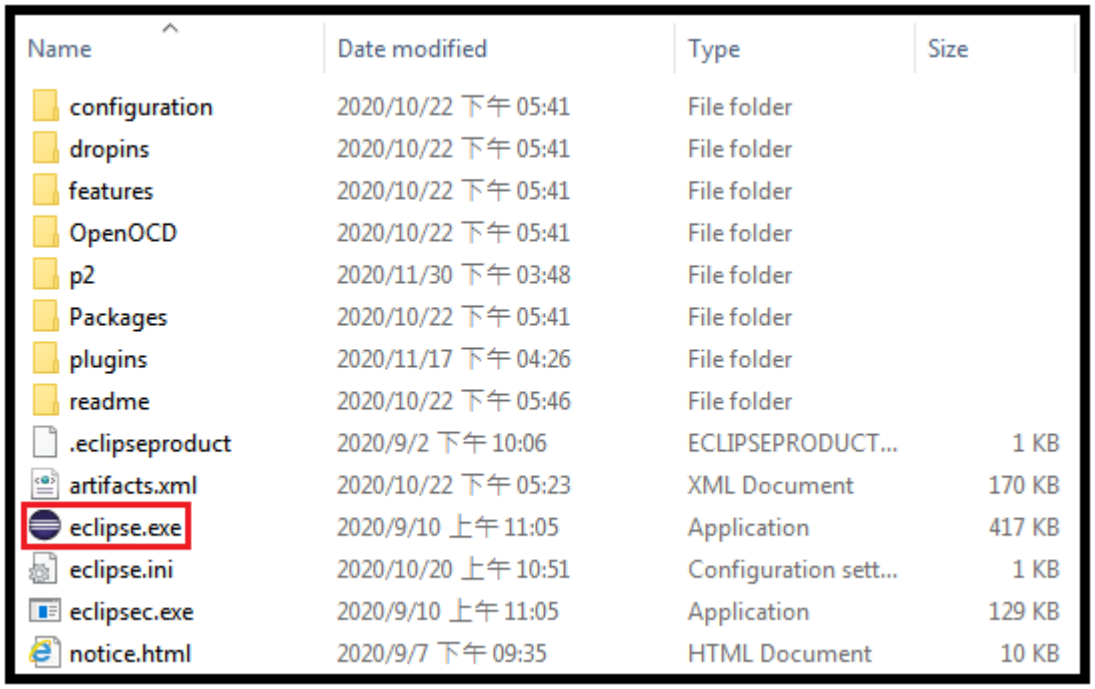

Figure 2-5 Eclipse.exe and Related Folders

3 Development Tutorial

3.1 Select Workspace

When Eclipse launches, we have to select a workspace which groups a set of related projects together that usually make up an application. In addition, some configuration settings for Eclipse and projects are stored here, too. For different computers, the configuration settings may change. We should create our own workspace rather than copying another user's workspace. Only one workspace can be active at one time. The current workspace for Eclipse can be switched by clicking **File->Switch Workspace**.

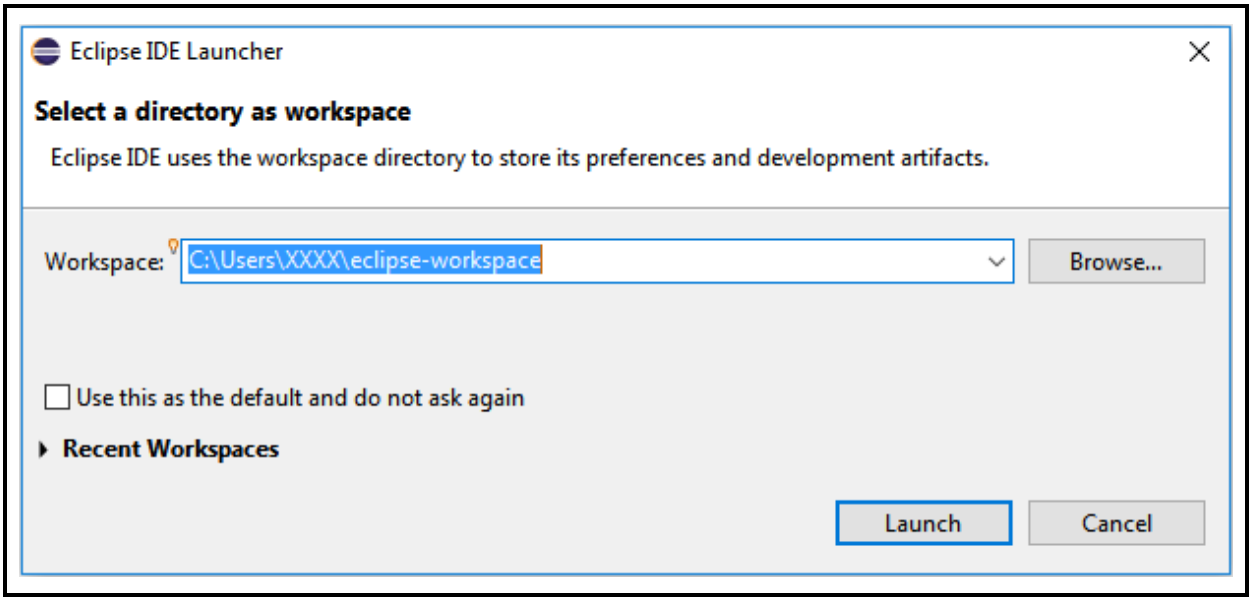

Figure 3-1 Selecting a Workspace

3.2 New Project Wizard

As a beginner, the fastest way to create a C/C++ project is using the New Project Wizard. For instance, to create a C project, click **File > New > C Project**. The New Project Wizard appears. Here we choose **Hello World Nuvoton Cortex-M C Project** for Project type. Input the project name and click the **Next >** button to continue.

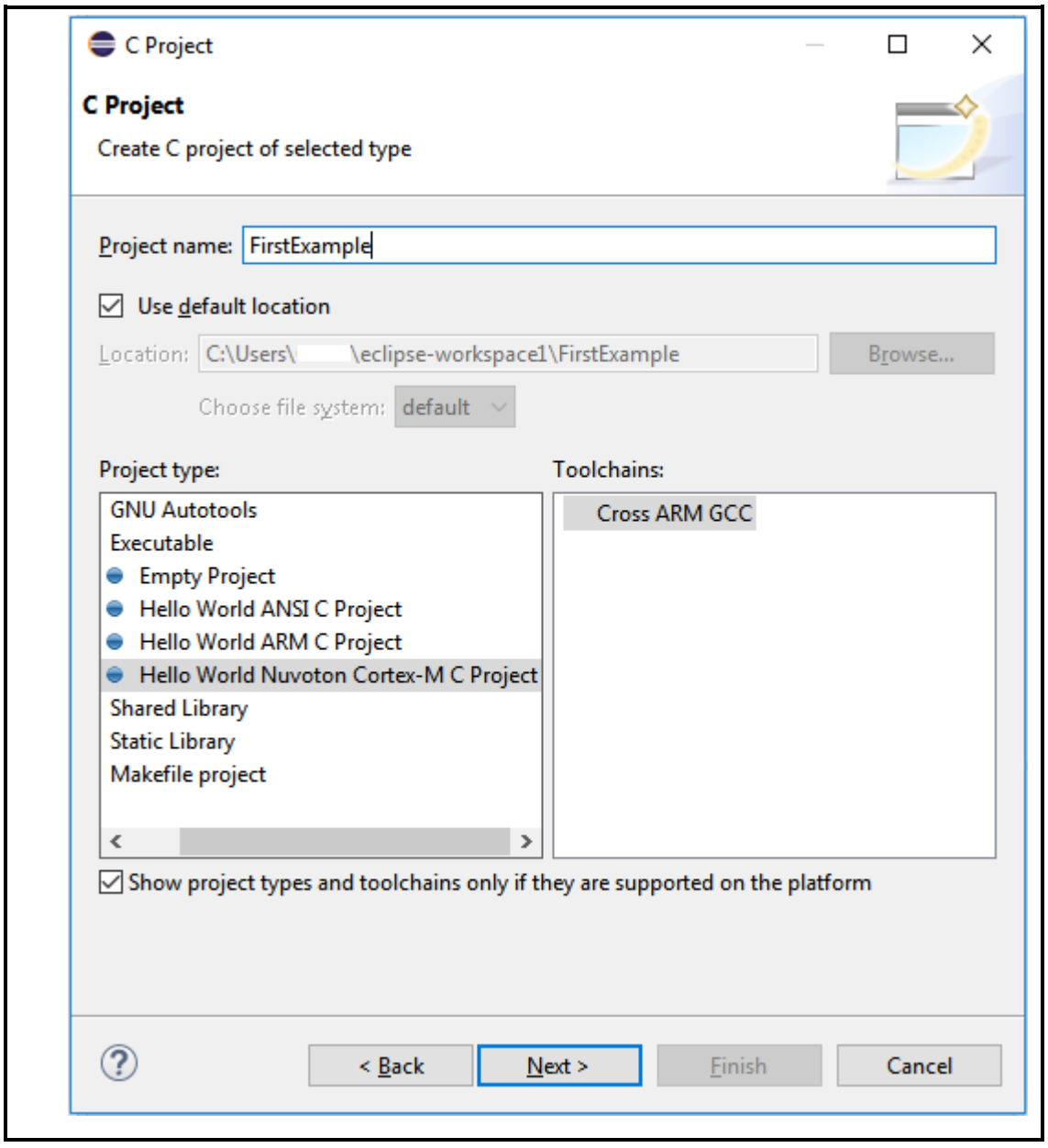

Figure 3-2 New Project Wizard

Based on the actual target chip, we select the corresponding chip series. For some chip series, e.g., M2351_NonSecure, we need to input the additional library path. If not, the build process may fail. In addition, input the real values to Flash and RAM size. If not, the default values will be used. When all the settings are done, click the **Next >** buttons until clicking the **Finish** button.

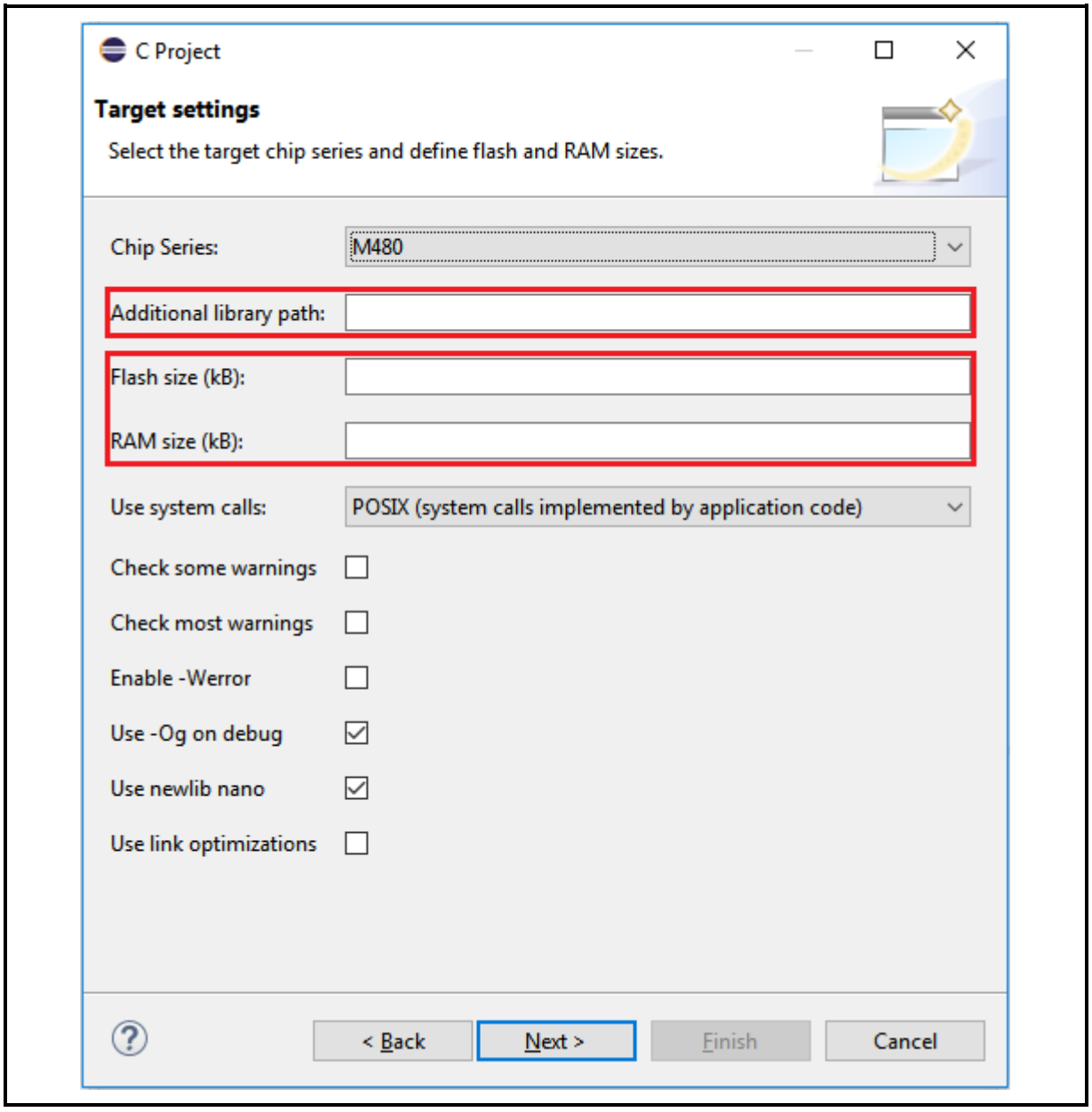

Figure 3-3 Target Processor Settings

3.3 Import Existing Projects

When BSP projects are available, we can import them into the workspace using the following steps:

1. From the main menu bar, select **File > Import**. The Import wizard shows up.
2. Select **General > Existing Project into Workspace** and click Next.
3. Choose either **Select root directory** or **Select archive file** and click the associated **Browse** to locate the directory or file containing the projects. In the Nuvoton BSP, the Eclipse projects are stored in the GCC folder.
4. Under Projects select the project or projects which you would like to import and click **Finish**.

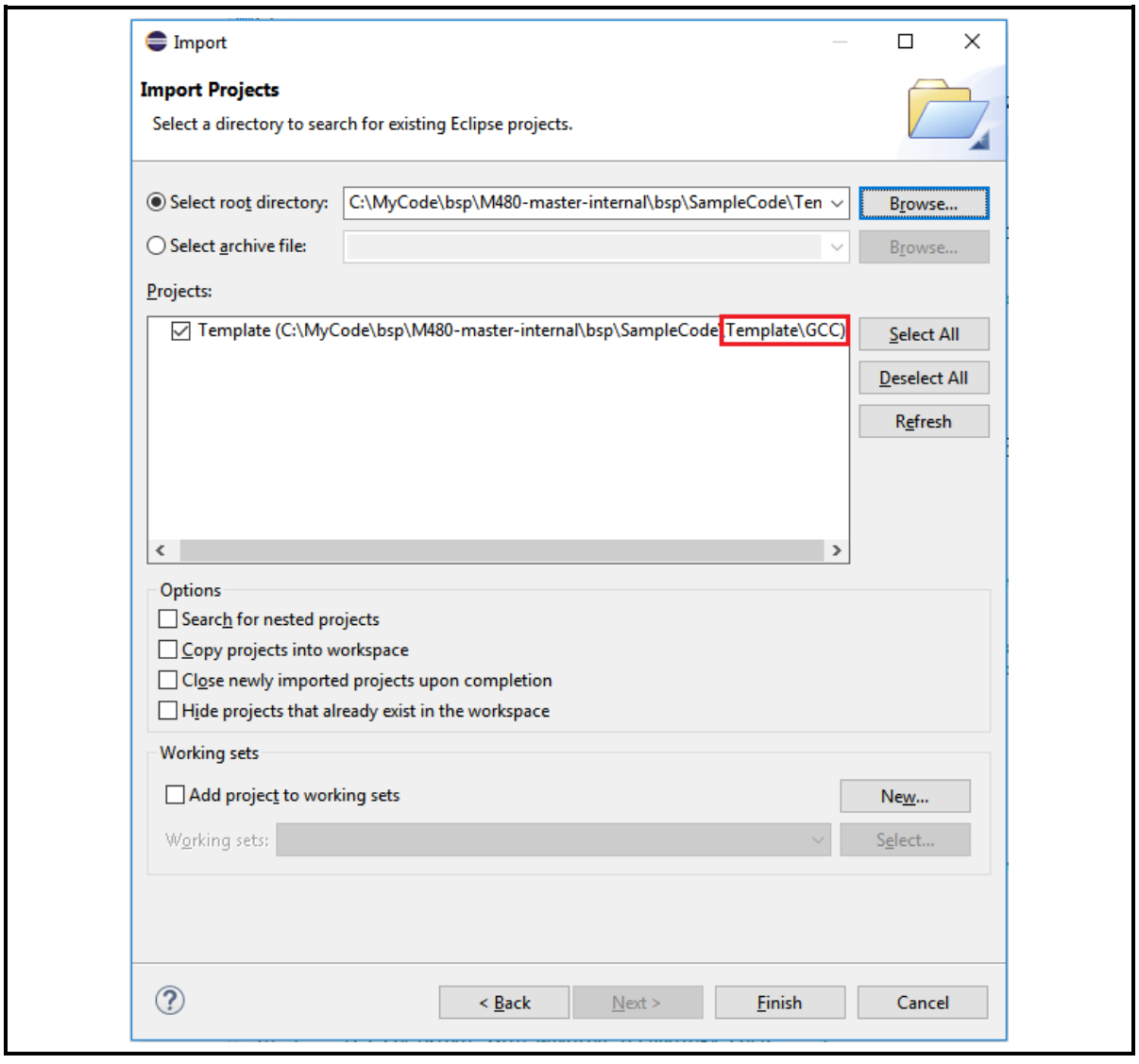

Figure 3-4 Importing Projects

3.4 Build Settings

After projects have been created, we still have a chance to alter the build settings by clicking **Project > Properties**. The Properties wizard shows up. Then go to **C/C++ Build > Settings.** From there, we can alter the build settings according to the actual target chip. Then click the **Apply** button to take effect. After applying build settings, we should be able to build projects successfully.

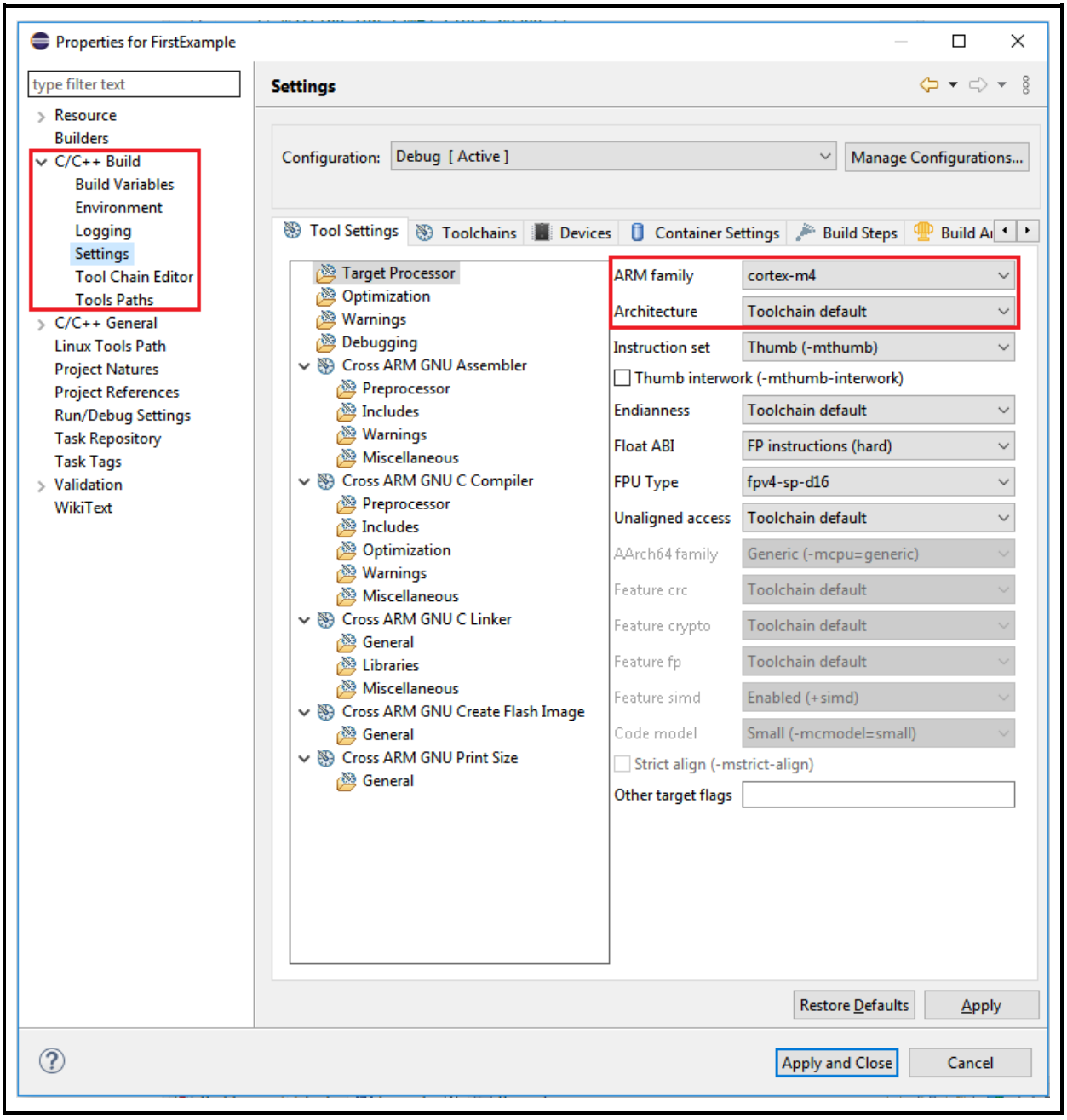

Figure 3-5 Build Settings

3.5 Debug Configuration

Before launching an application into the debug mode, we have to prepare a debug configuration, which contains all the necessary information about the debug mode. Click **Run > Debug Configuration…** to open the debug configuration dialog. Double click on the **GDB Nuvoton Nu-Link Debugging** group. The Nuvoton Nu-Link debug configuration appears on the right-hand side. In the Main tab, the name of Project should coincide with the project name. The C/C++ Application should point to the .elf application generated by the build process. If the project name or C/C++ Application is incorrect, please select the expected project first in the project view, build the project to generate the executable, and expand the tree to make sure the existence of the generated executable. Then repeat the former operations again.

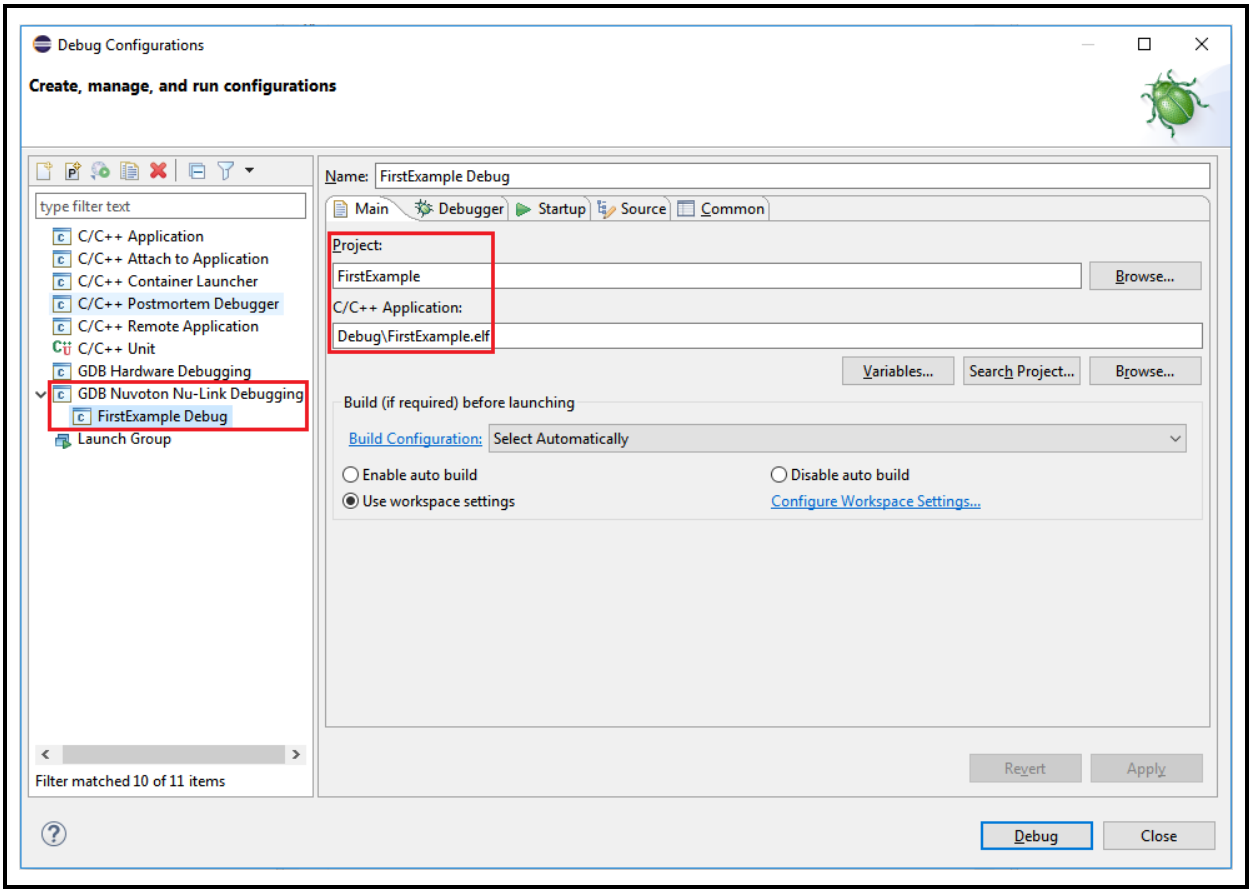

Figure 3-6 Debug Configuration

3.5.1 Debugger Tab

The Debugger tab is used to provide the OpenOCD and GDB Client setup. OpenOCD requires correct configuration files to know how to work with adapters and target chips. The configuration files are specified in the **Config options** field. Nuvoton's adapter is Nu-Link, which uses the interface configuration file named **nulink.cfg**. In addition, Nuvoton has three different ARM families, such as M0, M4, and M23. The corresponding target configuration files are **numicroM0.cfg**, **numicroM4.cfg**, and **numicroM23.cfg**. For M23 2nd development, the target configuration file would be **numicroM23_NS.cfg**.

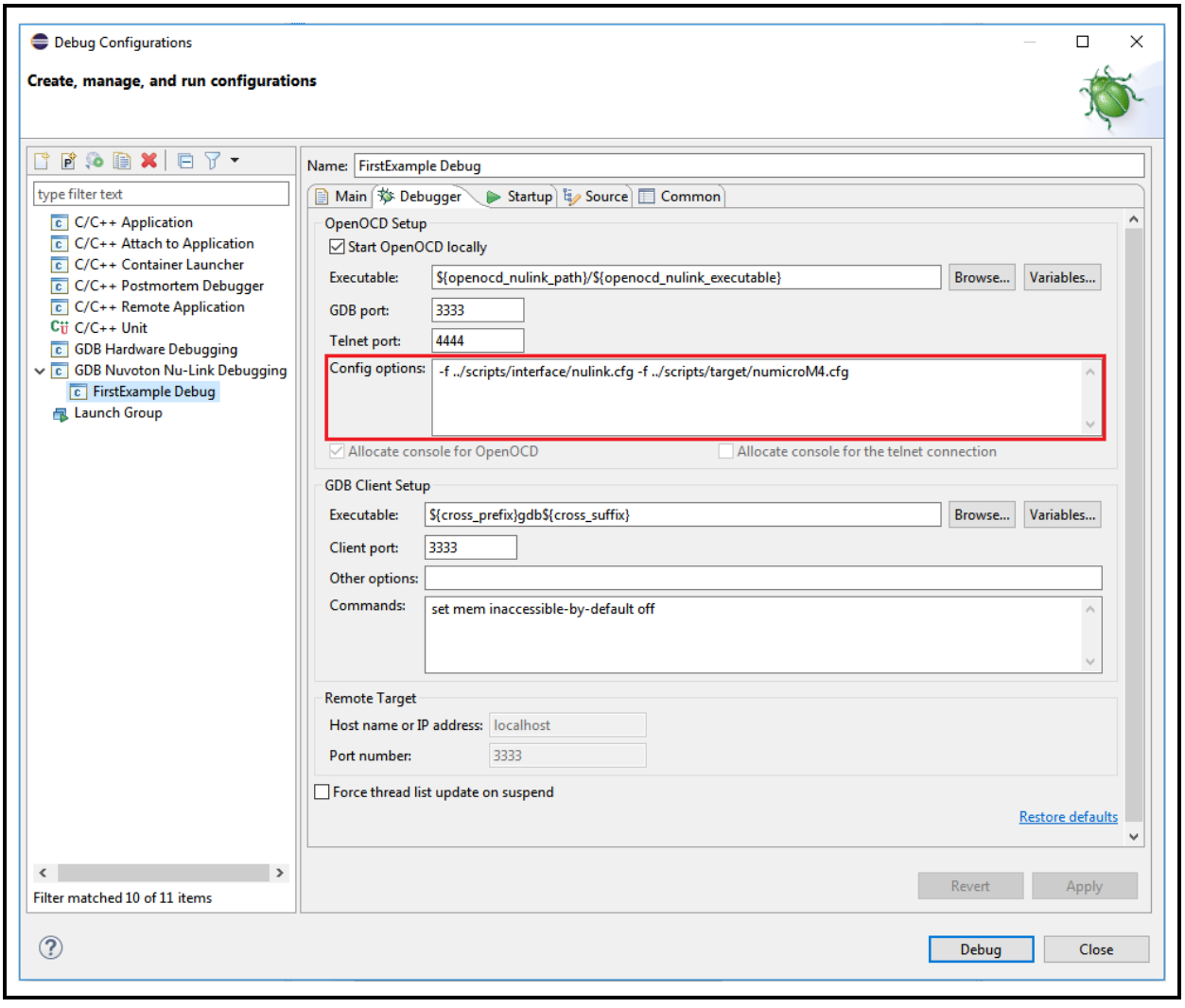

Figure 3-7 Configuring the Debugger Tab

3.5.2 Startup Tab

As the first step, we should **choose the right Chip Series** in the Startup tab. When done, the corresponding target configuration file will be automatically written in the **Config options** field of the Debugger tab. To load executable to flash, we need to select the **Load executable to flash** checkbox. To load executable to RAM, we need to select the **Load executable to SRAM** checkbox. When all the settings are done, click the **Apply** button to take effect. To launch the application into the debug mode, click the **Debug** button.

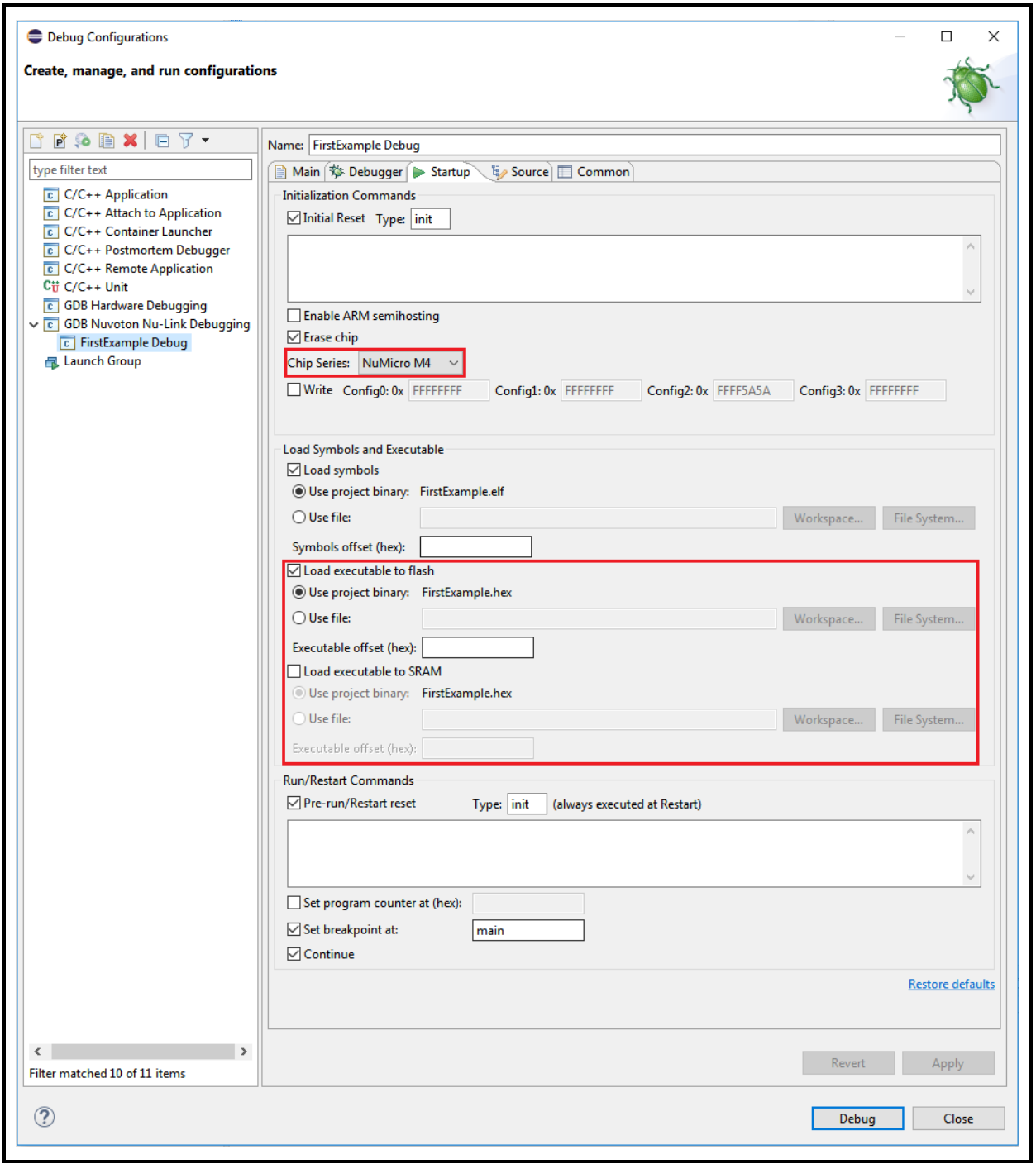

Figure 3-8 Configuring the Startup Tab

3.6 Debug Views

Eclipse provides many debug views. Each of them contains specific information for debugging.

3.6.1 Registers View

When entering the debug mode, we can open the **Registers view** in the bottom of Debug perspective. The Registers view lists information about the registers in a selected stack frame.

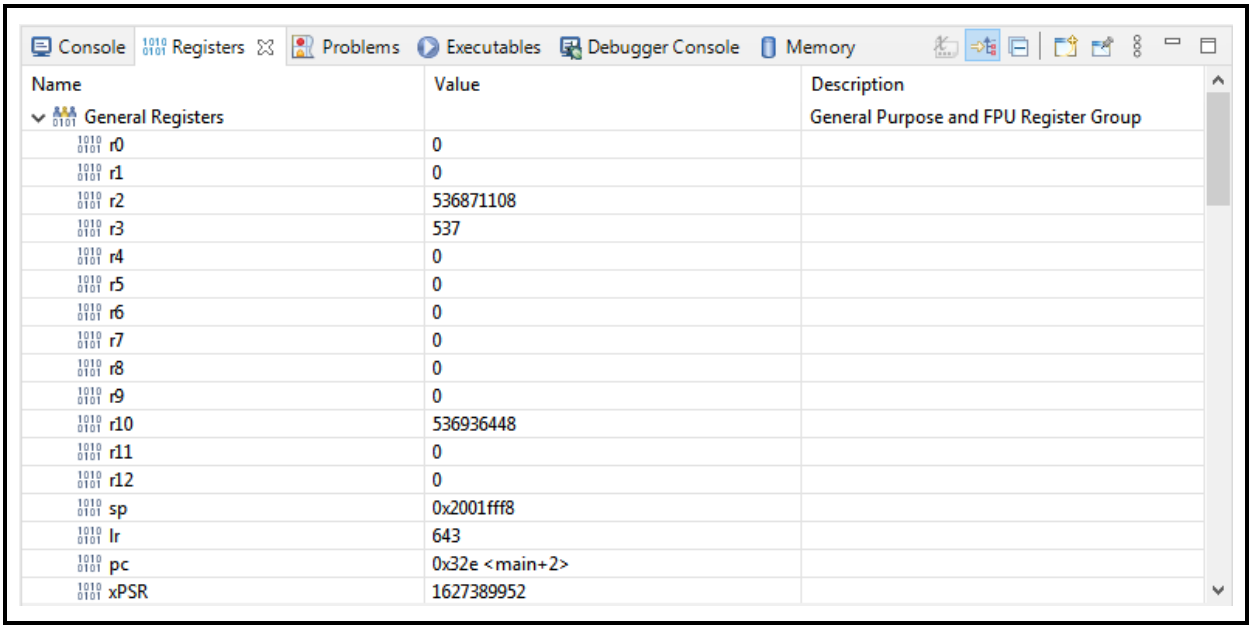

Figure 3-9 Registers View

3.6.2 Memory View

The **Memory view** of the Debug perspective is used to monitor and modify the process memory. The process memory is presented as a list of so called **memory monitors**. Each monitor represents a section of memory specified by its location called **base address**. To open it, click the Memory tab on the lower side of Debug perspective.

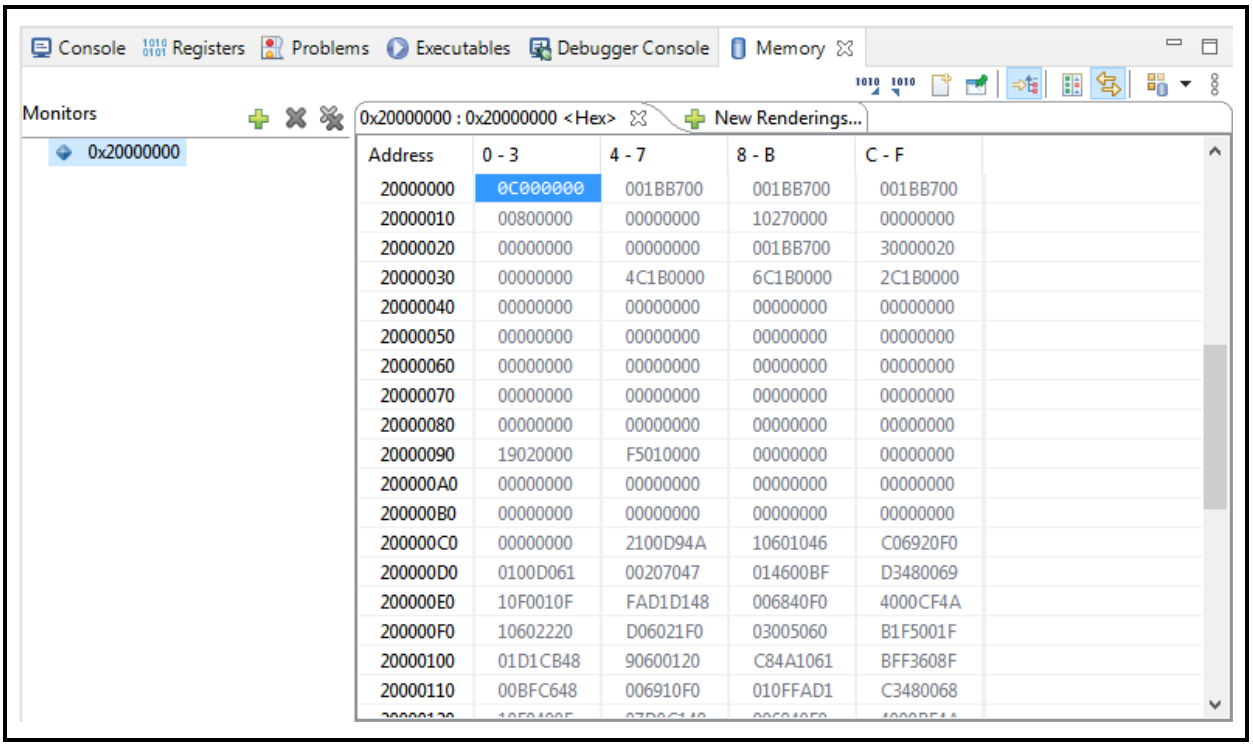

Figure 3-10 Memory View

3.6.3 Disassembly View

The **Disassembly view** shows the loaded program as assembler instructions mixed with source code for comparison. We can do the following tasks in the Disassembly view:

1. Setting breakpoints at the start of any assembler instruction.
2. Enabling and disabling breakpoints.
3. Stepping through the disassembly instructions of the program.
4. Jumping to specific instructions in the program.

To open it, we need to click the **Instruction Stepping Mode** button on the upper toolbar, as follows:

Figure 3-11 Clicking the Instruction Stepping Mode Button

Then the Disassembly view will appear on the right-hand side.

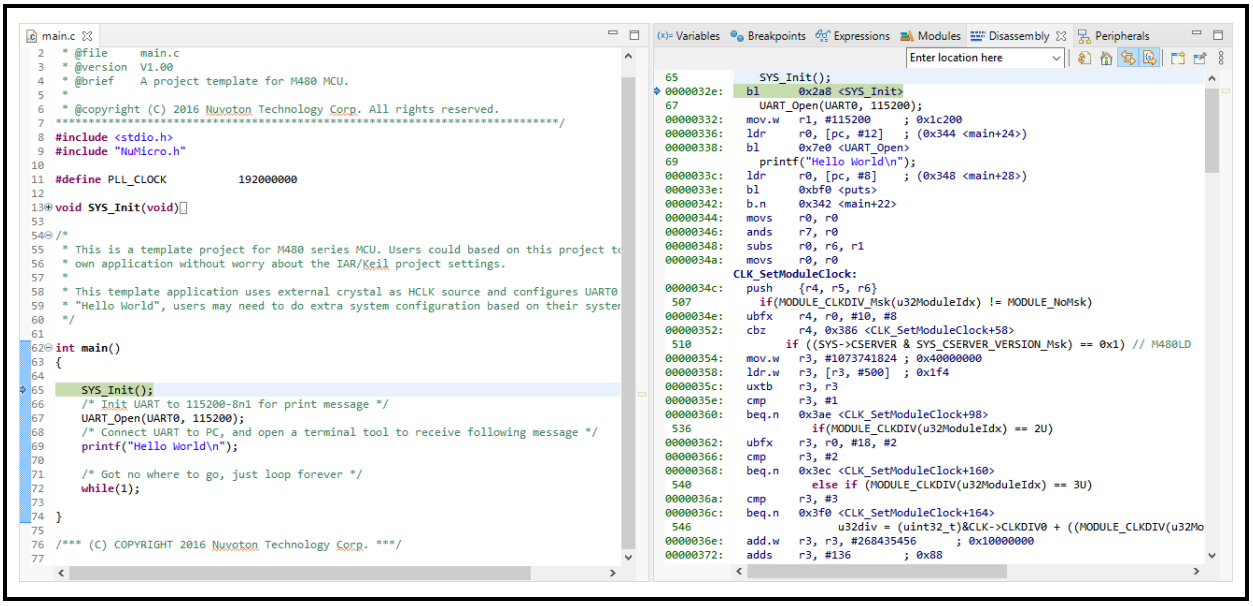

Figure 3-12 Disassembly View

3.6.4 Peripheral Registers View

To display the Peripheral Registers view, we need to utilize **Packs** mechanism**.** Packs can help the user download special function register (SFR) files from the Keil repository. Firstly, we open the Packs perspective by choosing it in the **Open Perspective** dialog.

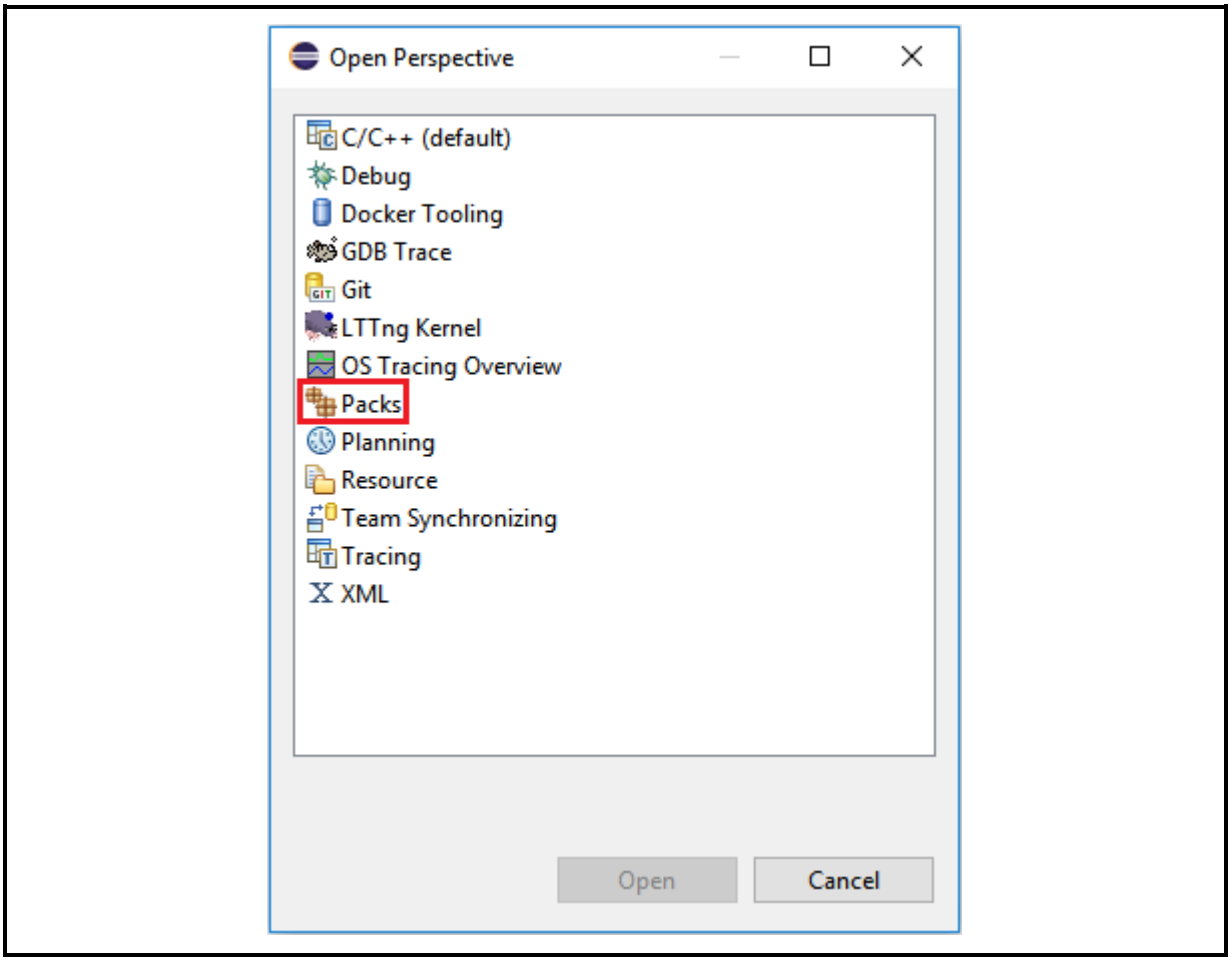

Figure 3-13 Opening the Packs Perspective

For the first time we see the Packs perspective, its content is provided by the NuEclipse installer. If the default content is missing, please switch to a new workspace and try again. To get the latest version, click the **Update the packages definitions from all repositories** button at the upper-right corner. After clicking, Eclipse begins downloading all the SFR files from an online repository.

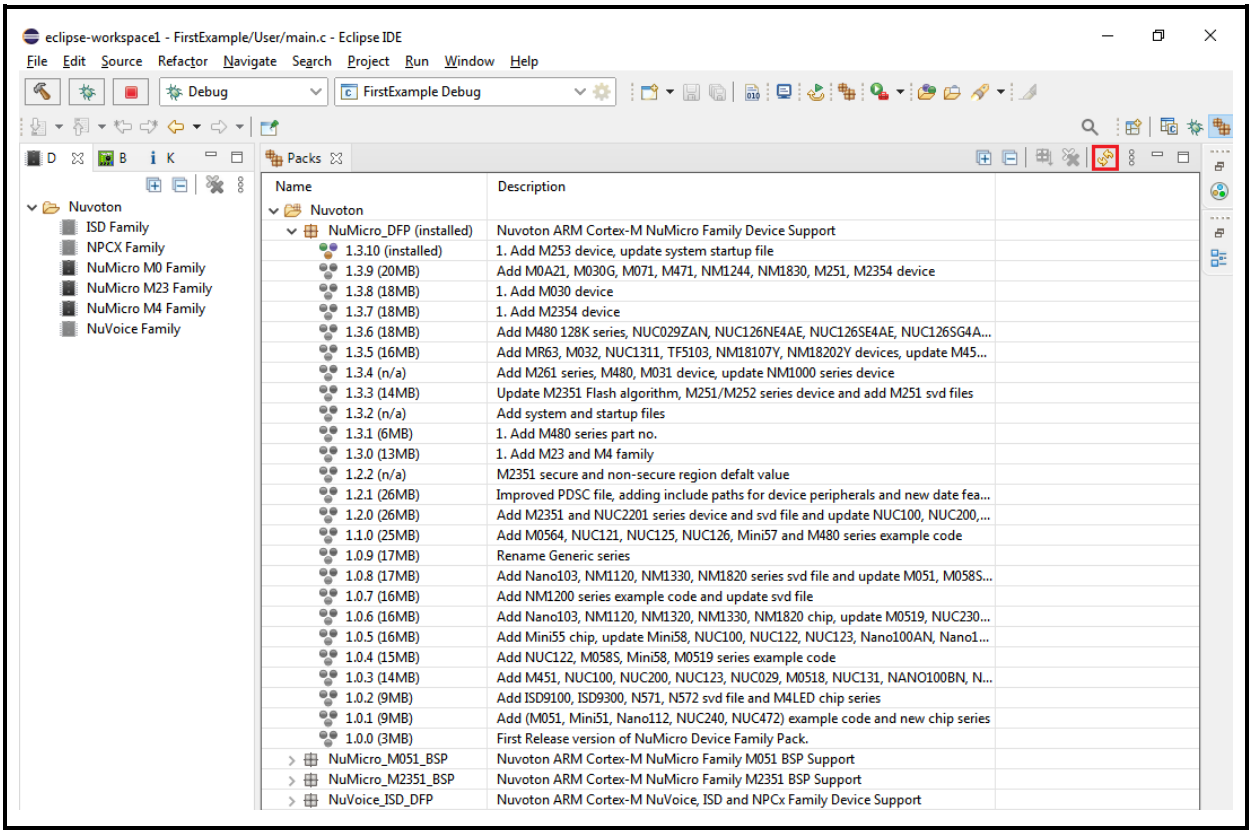

Figure 3-14 How to Download Packages

The locations of repositories are specified in the **Window > Preferences > C/C++ > Packages > Repositories**. The default is from Keil's CMSIS Pack.

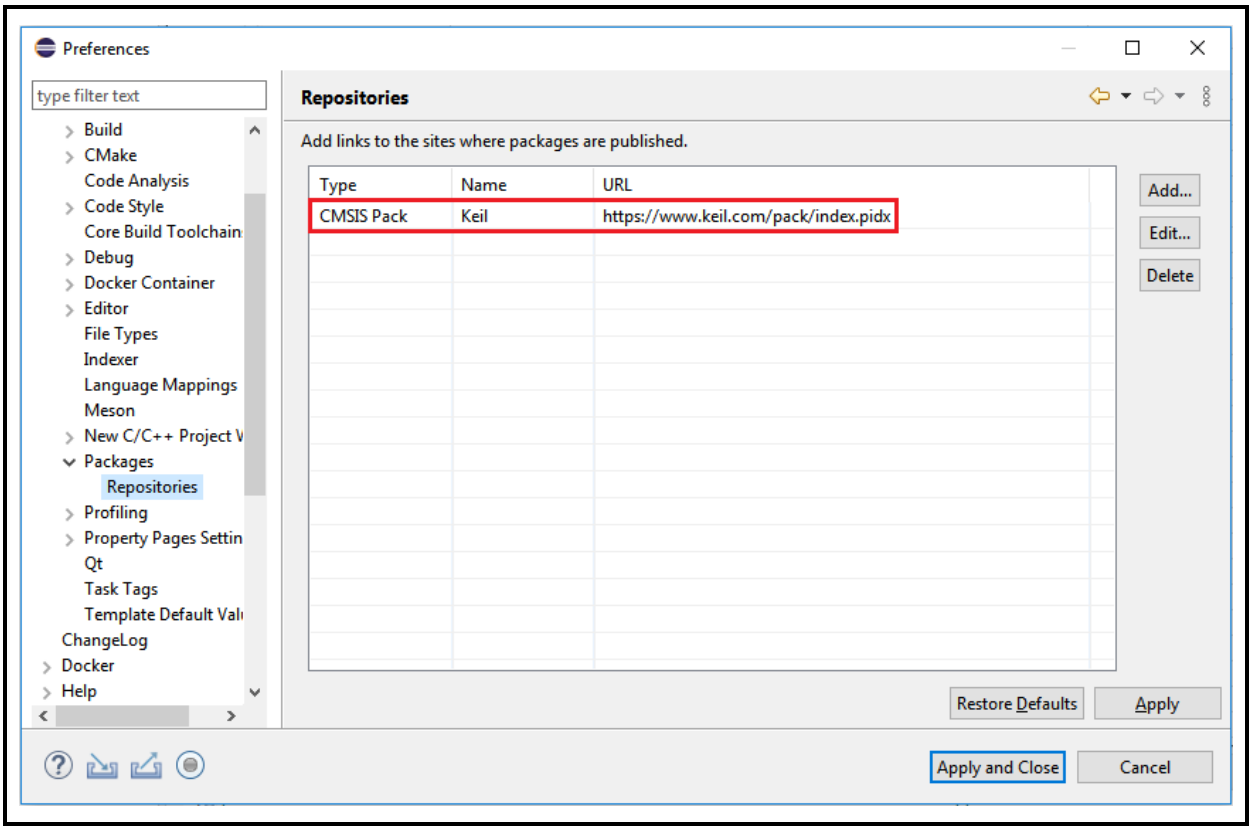

Figure 3-15 Locations of Repositories

When the download is completed, we can find the Nuvoton SFR files and install them on Eclipse if needed.

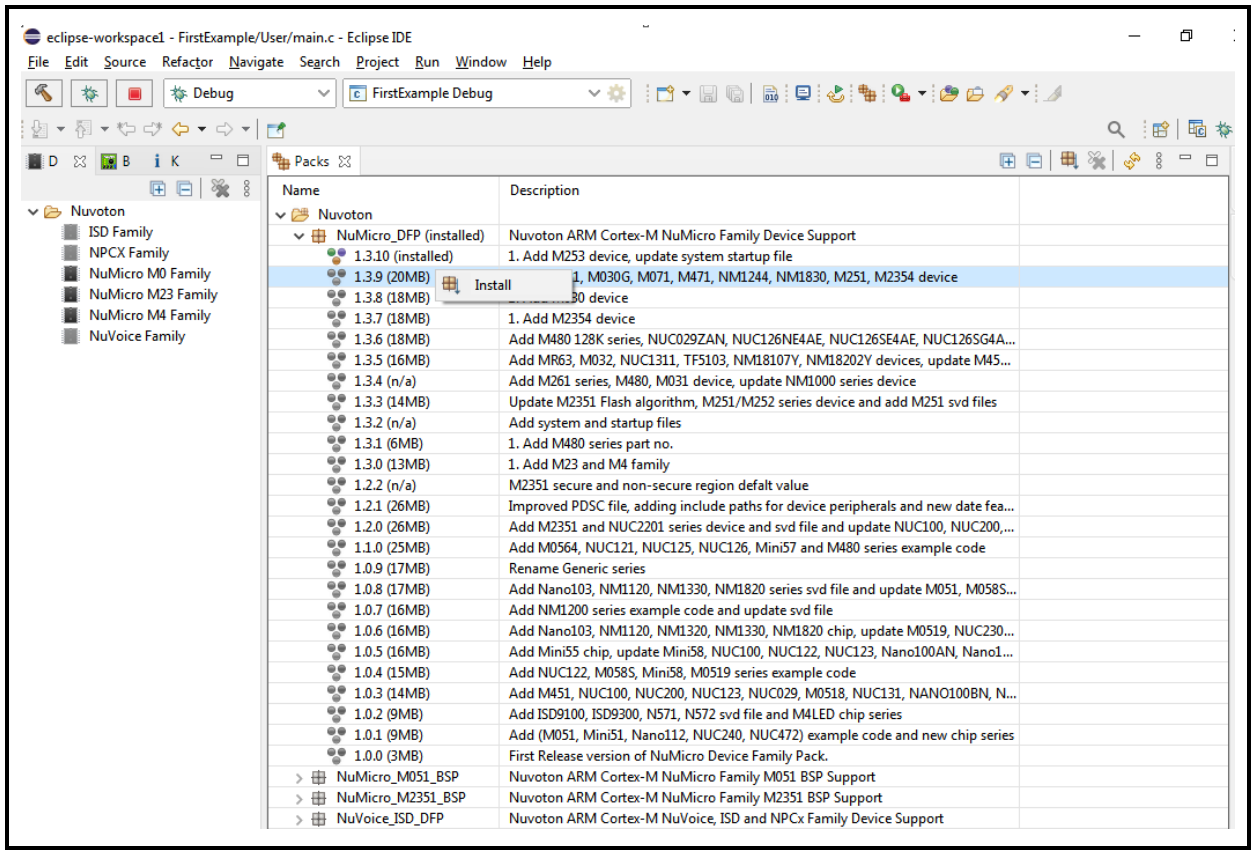

Figure 3-16 Installing SFR Files

To use the specific SFR file, open the project's properties dialog and go to the **C/C++ Build > Settings**. From there, we should choose the specific device matching the real one. In this case, it is M487JIDAE. Click the **Apply** button to take effect.

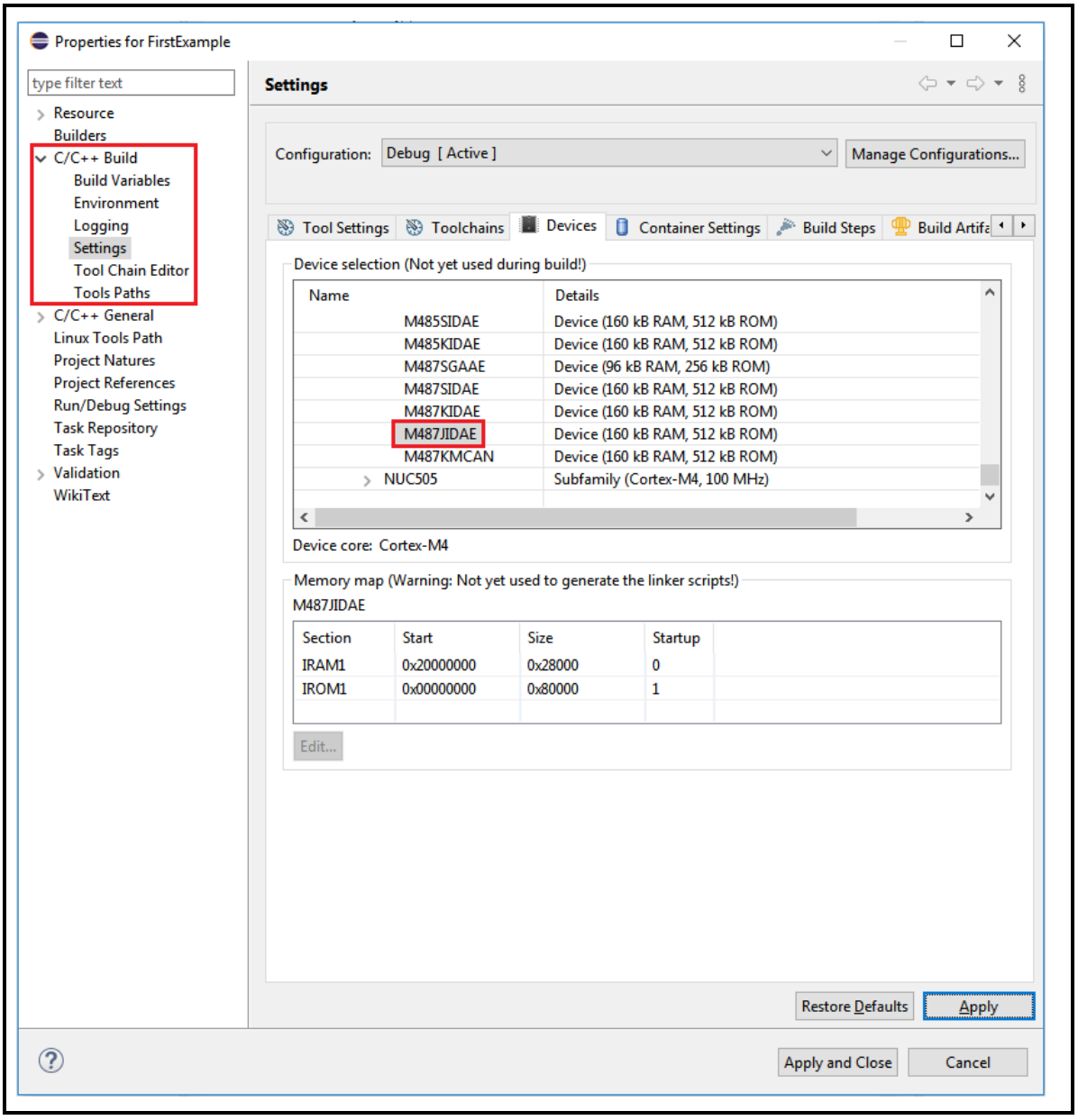

Figure 3-17 Device Selection

As a result, we can monitor the peripheral registers when debugging.

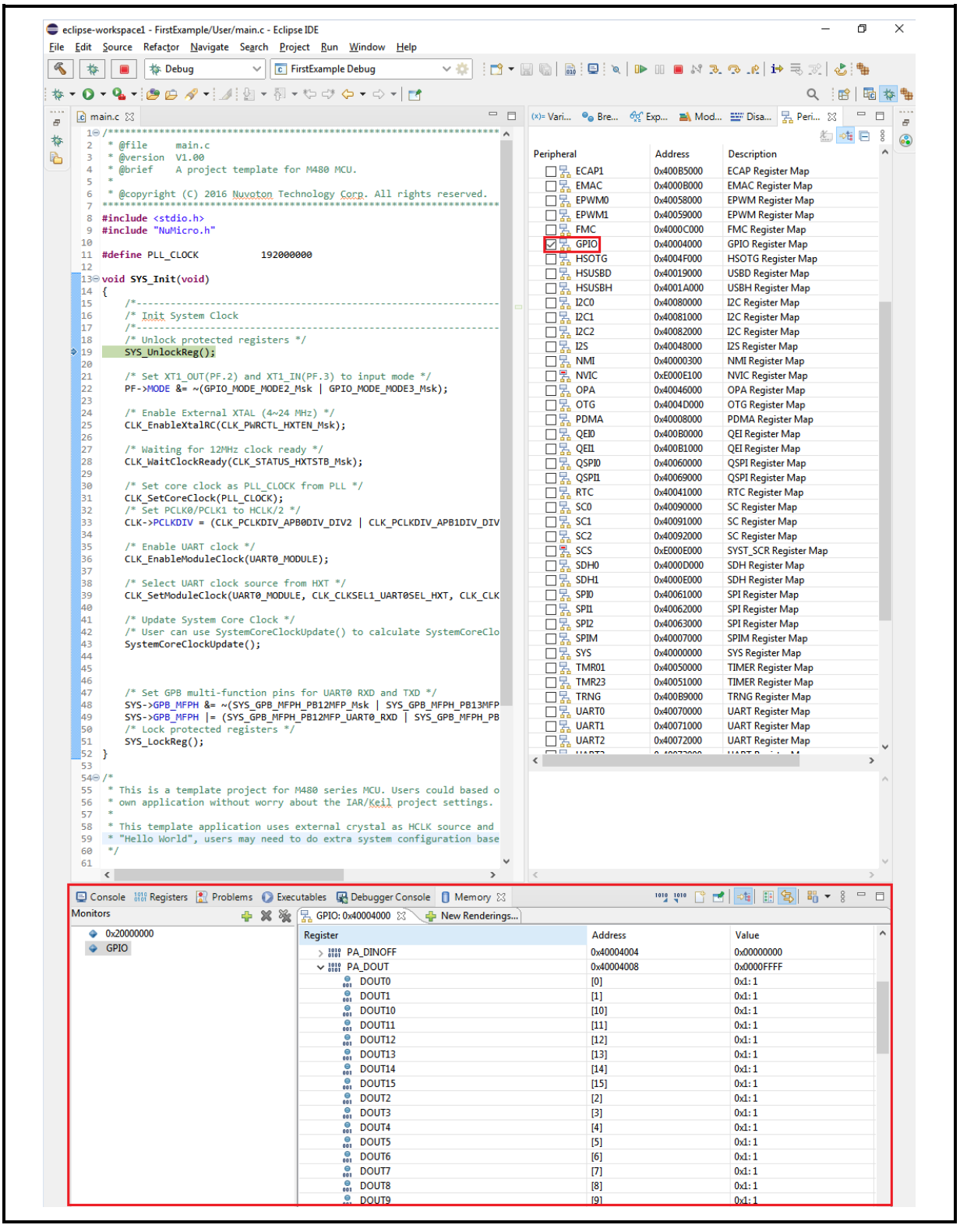

Figure 3-18 Peripheral Registers View

3.7 **Watchpoints**

To add watchpoints on Eclipse, we need to do the following steps:

1. Selecting a **globe variable**, i.e. g_seconds, in the Outline view.
2. Right-clicking on the global variable and choosing **Toggle Watchpoint**.

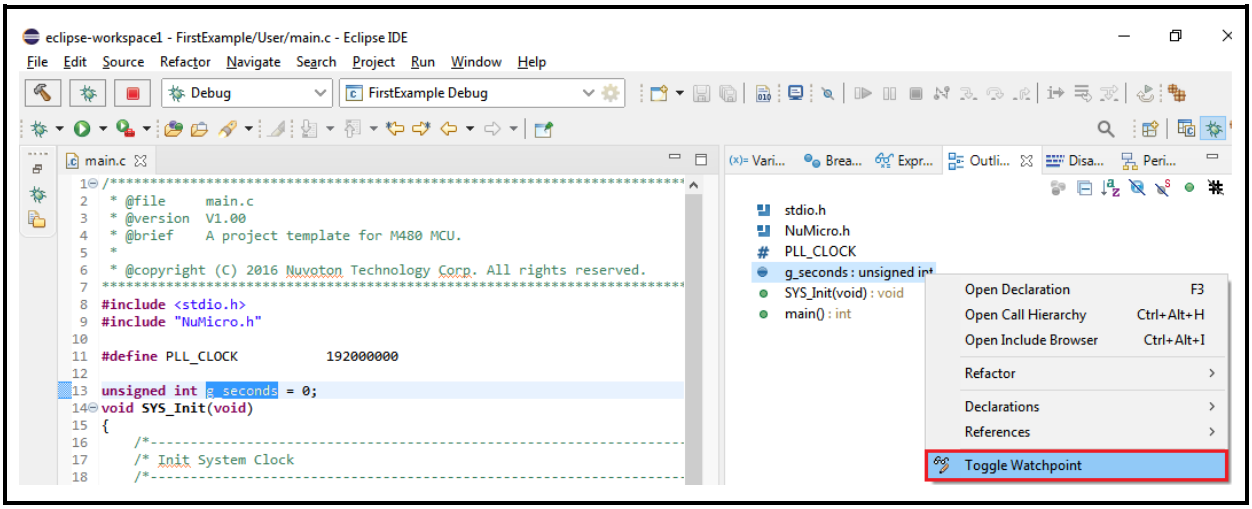

Figure 3-19 Toggle Watchpoint

3. Configuring the settings for watchpoints. To stop execution when the watch expression is read, select the **Read** checkbox. To stop execution when the watch expression is written to, select the **Write** checkbox.

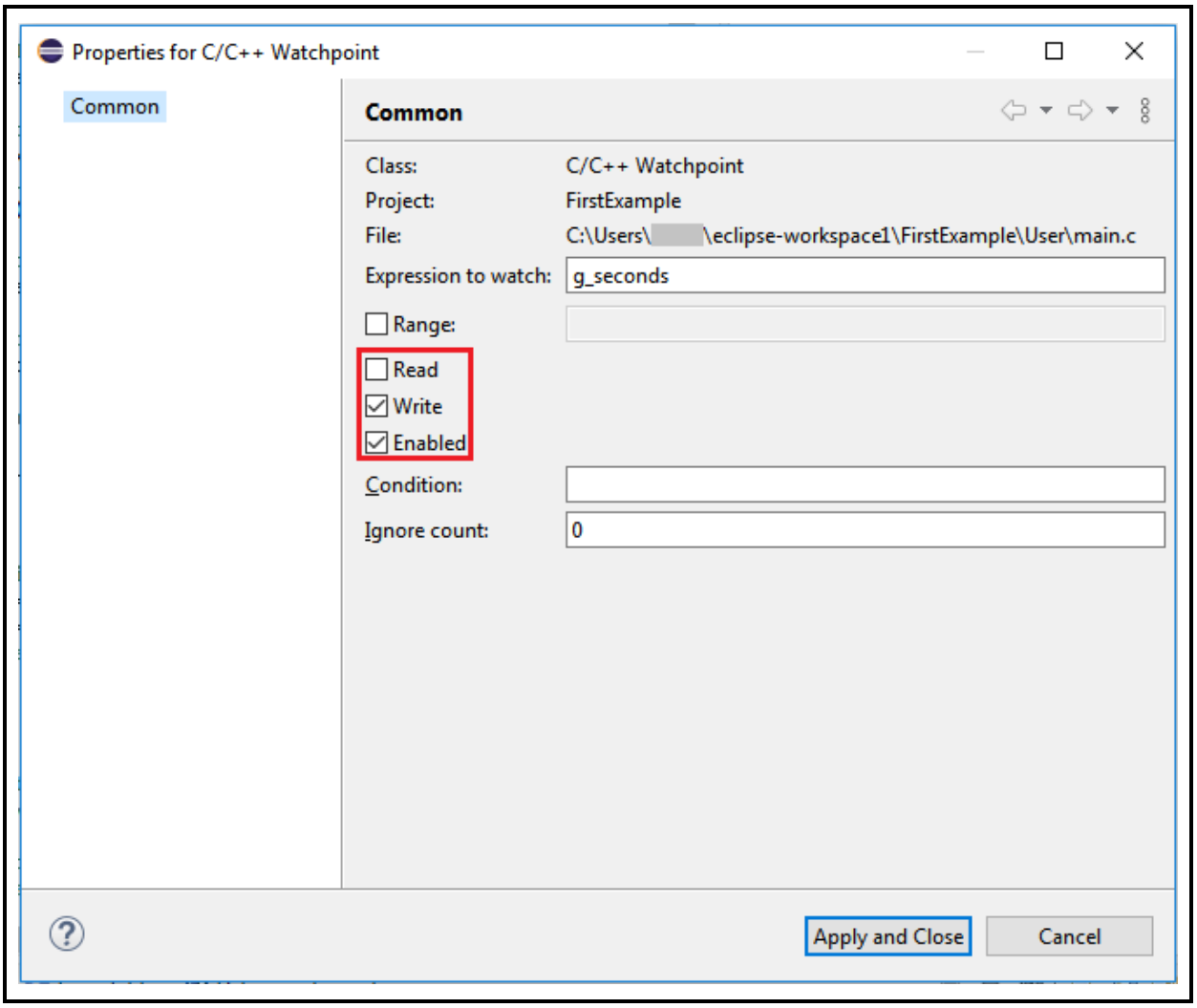

Figure 3-20 Properties for C/C++ Watchpoint

When the watchpoint is added, it appears in the **Breakpoints view**.

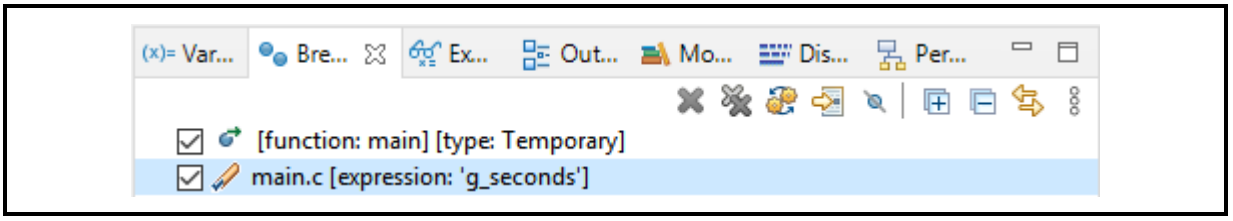

Figure 3-21 Added Watchpoint in the Breakpoints View

3.8 **Debug in RAM**

To debug in RAM, there are several steps to follow:

1. Modifying the ld script.
2. Assigning PC to the specific RAM address.
3. Assigning SP to the specific RAM address.
4. Downloading the binary file to RAM.

The ld script is responsible for telling the linker the layout of the compiled executable. For example, the memory layout looks like:

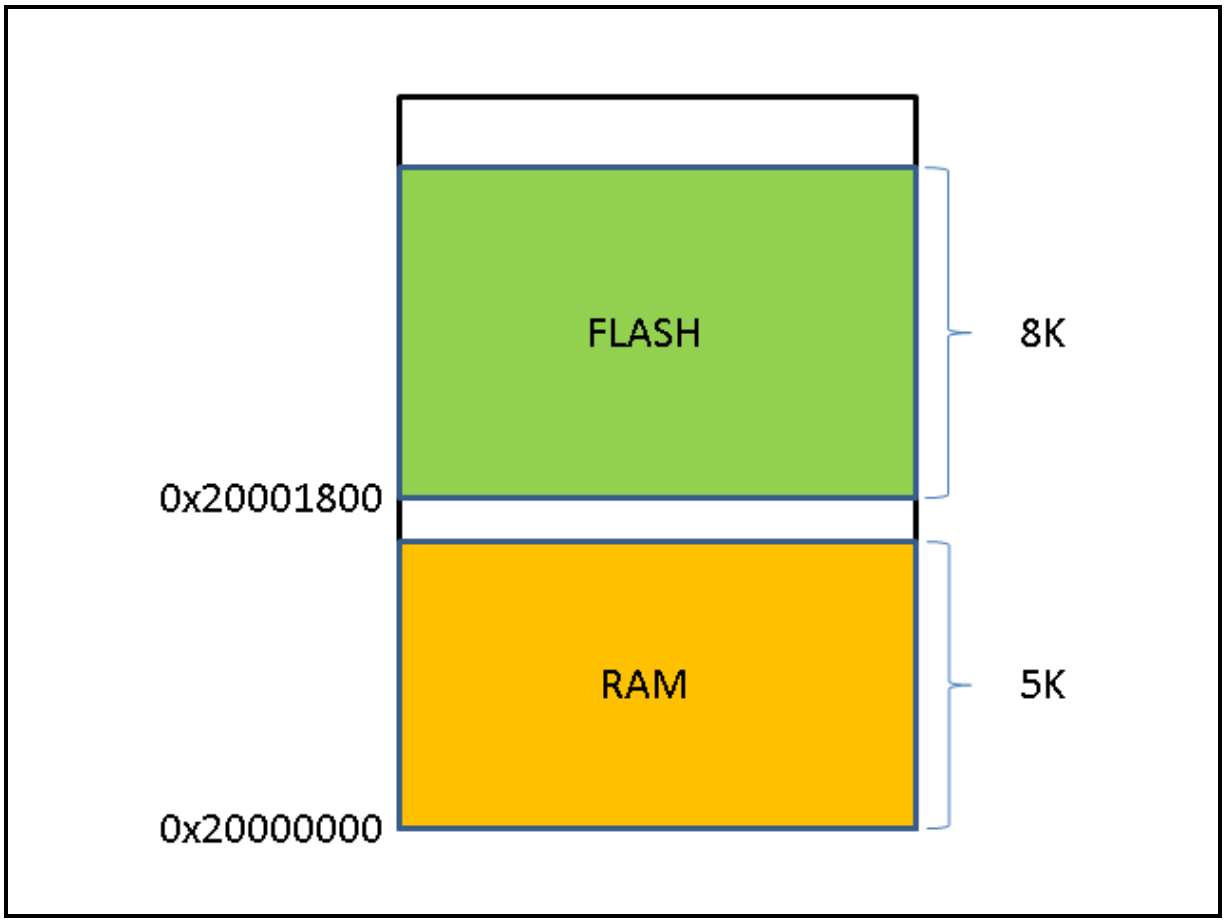

Figure 3-22 Memory Layout

The modified ld script should meet the memory layout design.

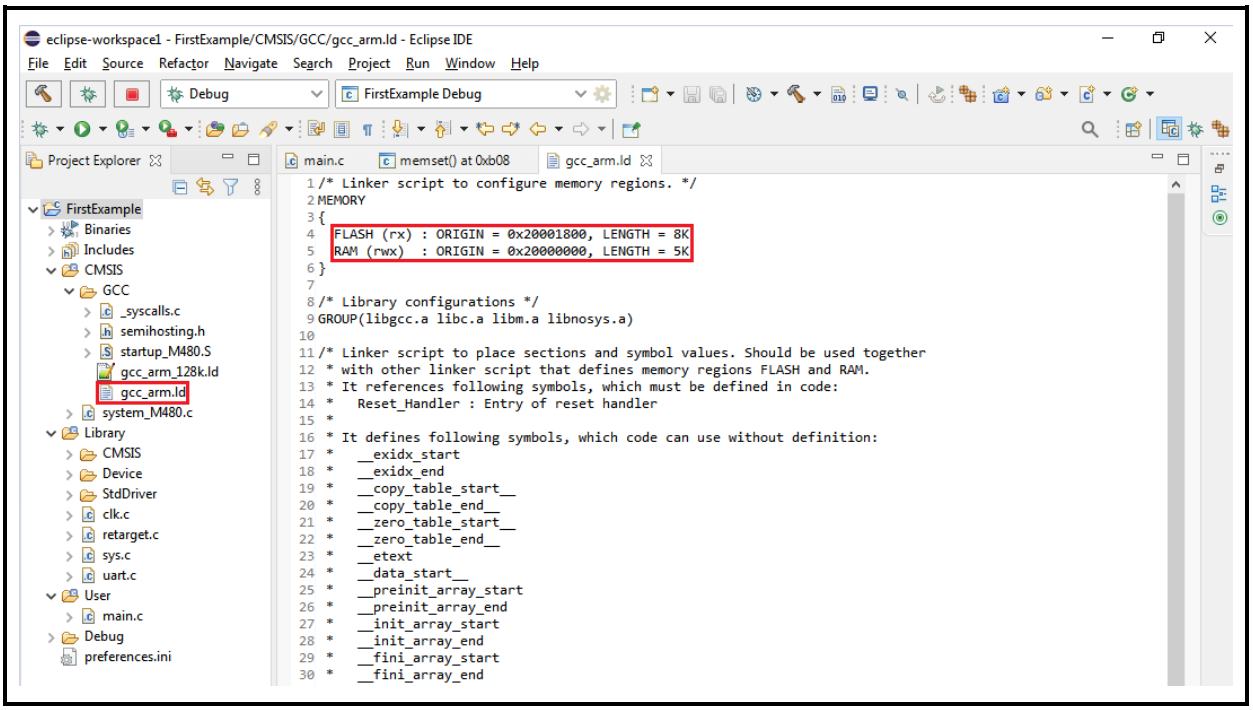

Figure 3-23 Modifying the ld Script

To assign PC and SP to the specific addresses, we need to input them in the debug configuration, as follows. Based on the previous memory layout, the PC and SP addresses should be Reset_Handler and 0x20001400, respectively. In addition, set Vector Table Offset Register (0xE000ED08) should be 0x20000000 and unselect the **Pre-run/Restart reset** Button. To download the binary to RAM, we select the **Load executable to SRAM** button and unselect the **Load executable to flash** button. Click the **Debug** button to start a debug session.

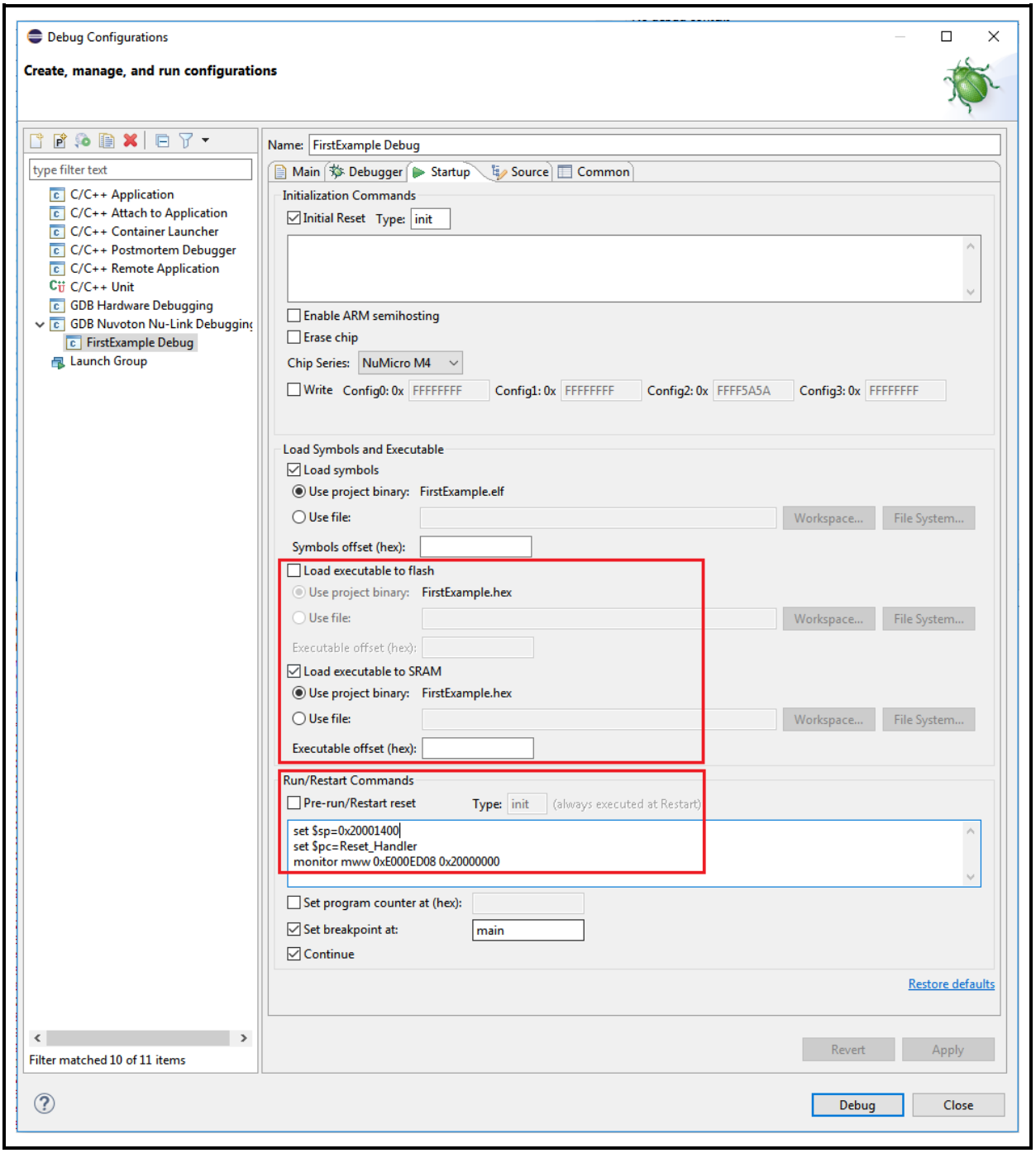

Figure 3-24 Debug Configuration Settings

When the program stops in the main function, we open the Memory view. From there, we can verify that the binary file is successfully downloaded into RAM. The first word denotes the SP address. The following words denote the addresses of handlers.

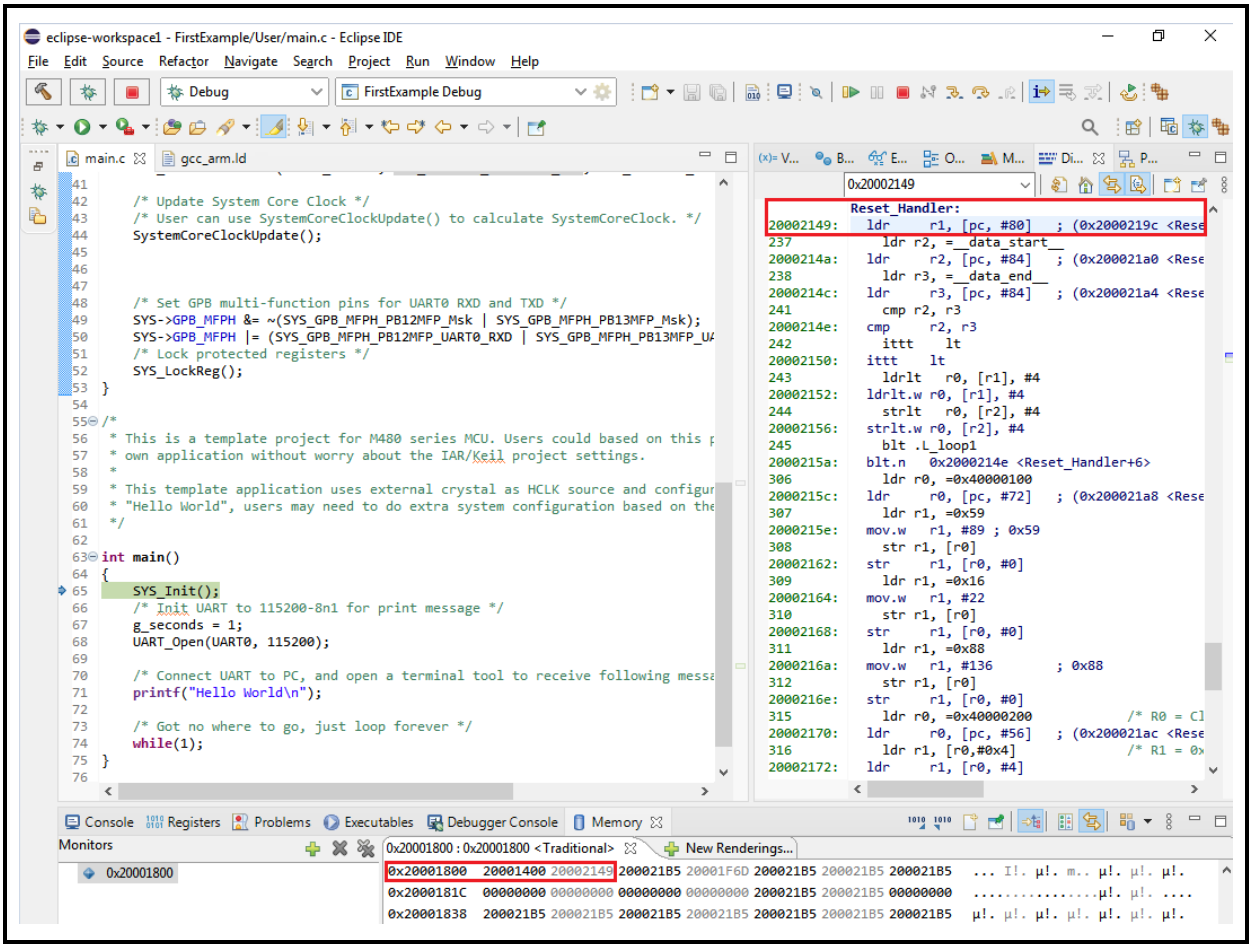

Figure 3-25 Debugging in RAM

3.9 Debug Executable Files Only

If the user has an executable built with **debug symbols** but may not have the relevant environment used to build the executable, he still is able to debug the executable by the following steps:

1. Import an executable for debugging (referring to Figure 3-26).
2. Click Browse following Select executable, then select an executable (referring to Figure 3-27).
3. Choose GDB Nuvoton Nu-Link Debugging as a Launch Configuration (referring to Figure 3-28).
4. Locate the GDB executable in the debug configuration (referring to Figure 3-29).
5. Choose the ELF file to download in the debug configuration (referring to Figure 3-30).
6. Add source lookup path relative to source folders (referring to Figure 3-31).
7. Press on the Debug button.

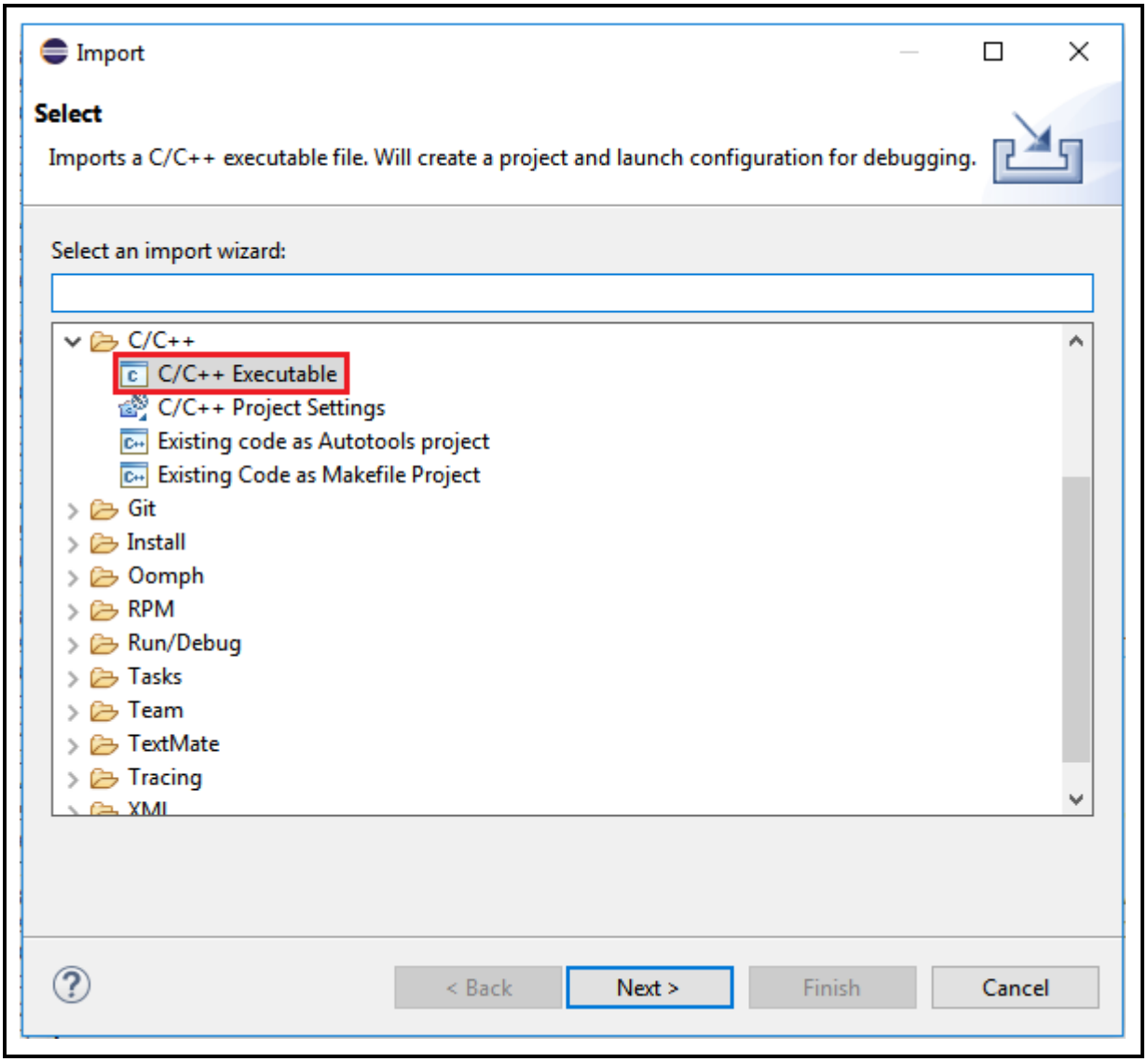

Figure 3-26 Importing Executable for Debugging

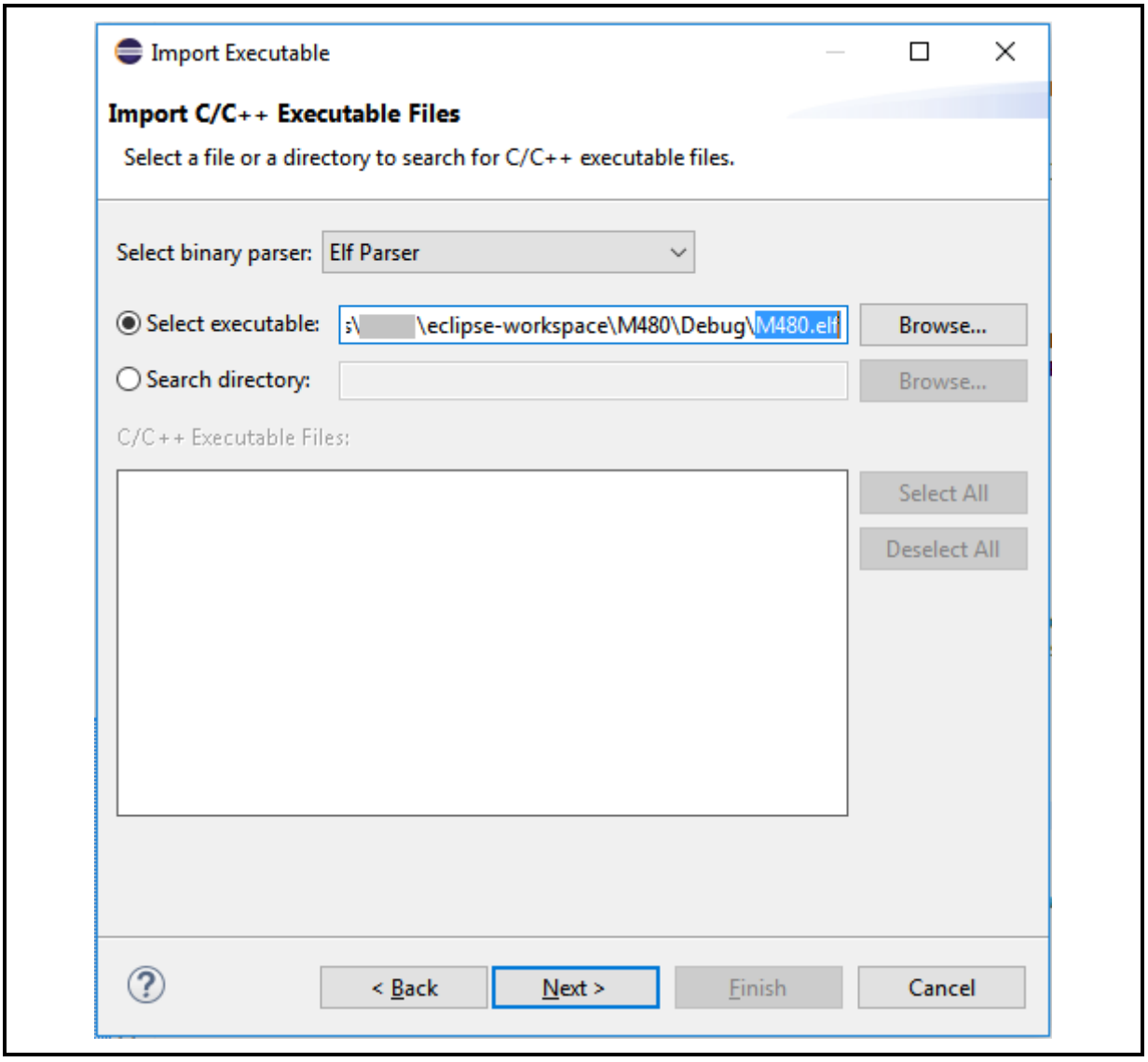

Figure 3-27 Selecting Executable

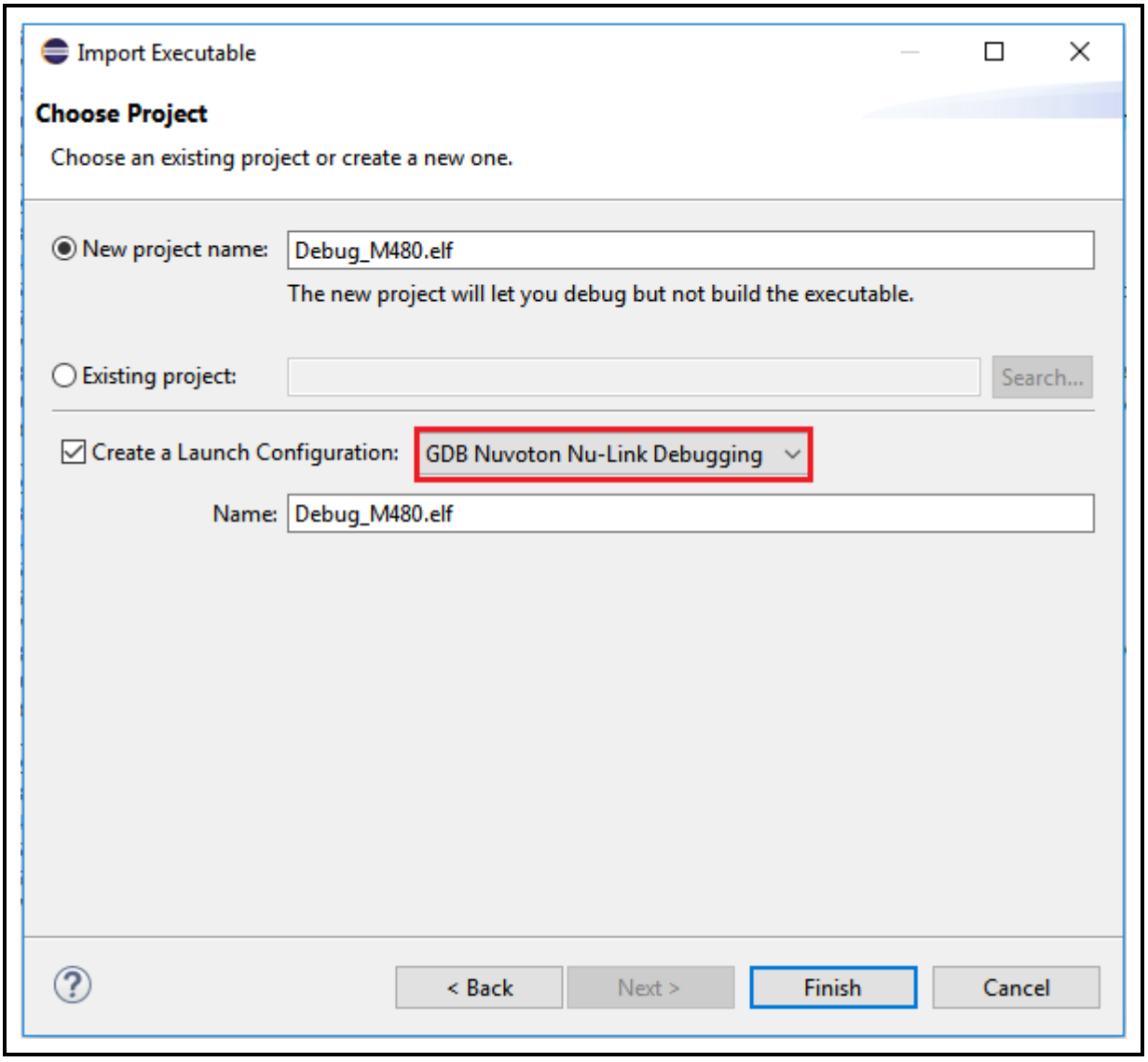

Figure 3-28 Choosing GDB Nuvoton Nu-Link Debugging

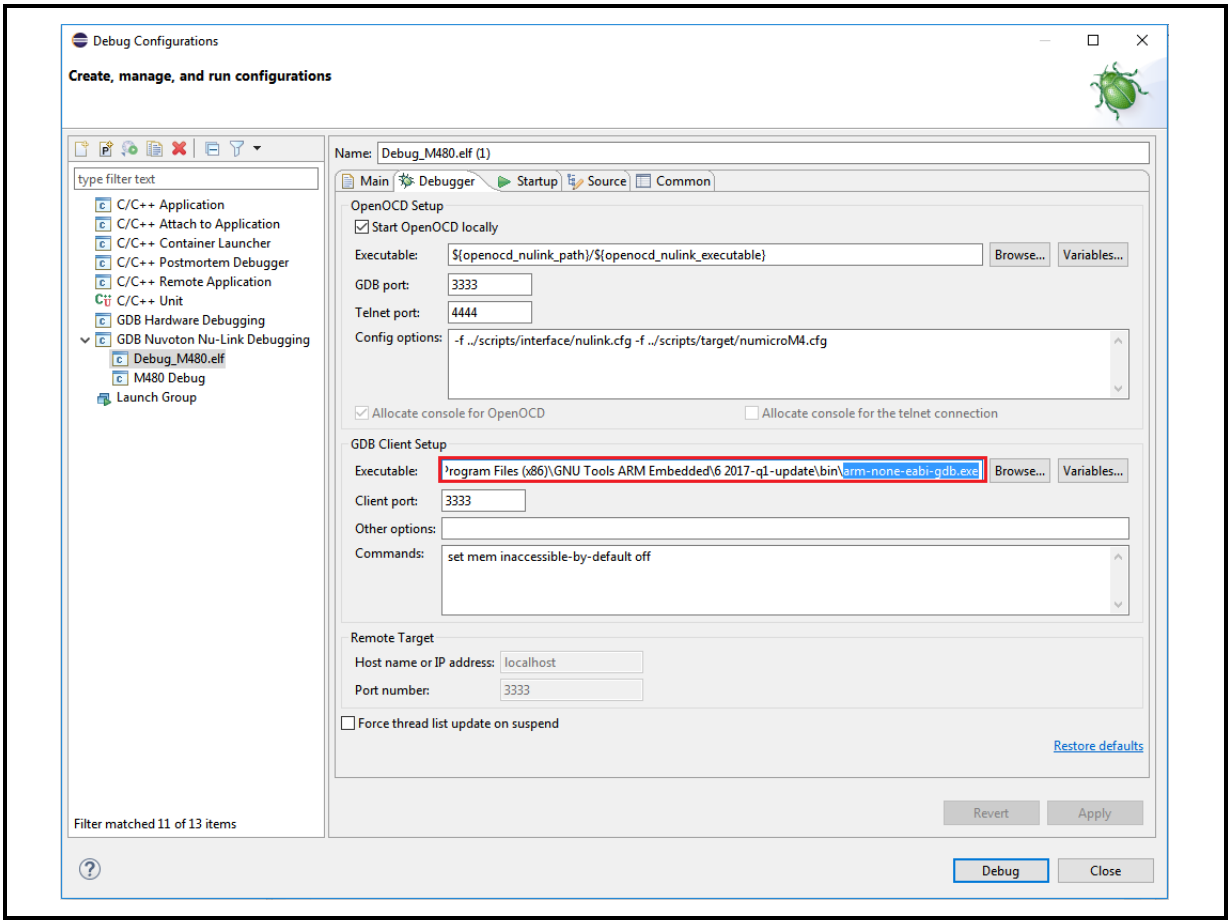

Figure 3-29 Locating the GDB Executable

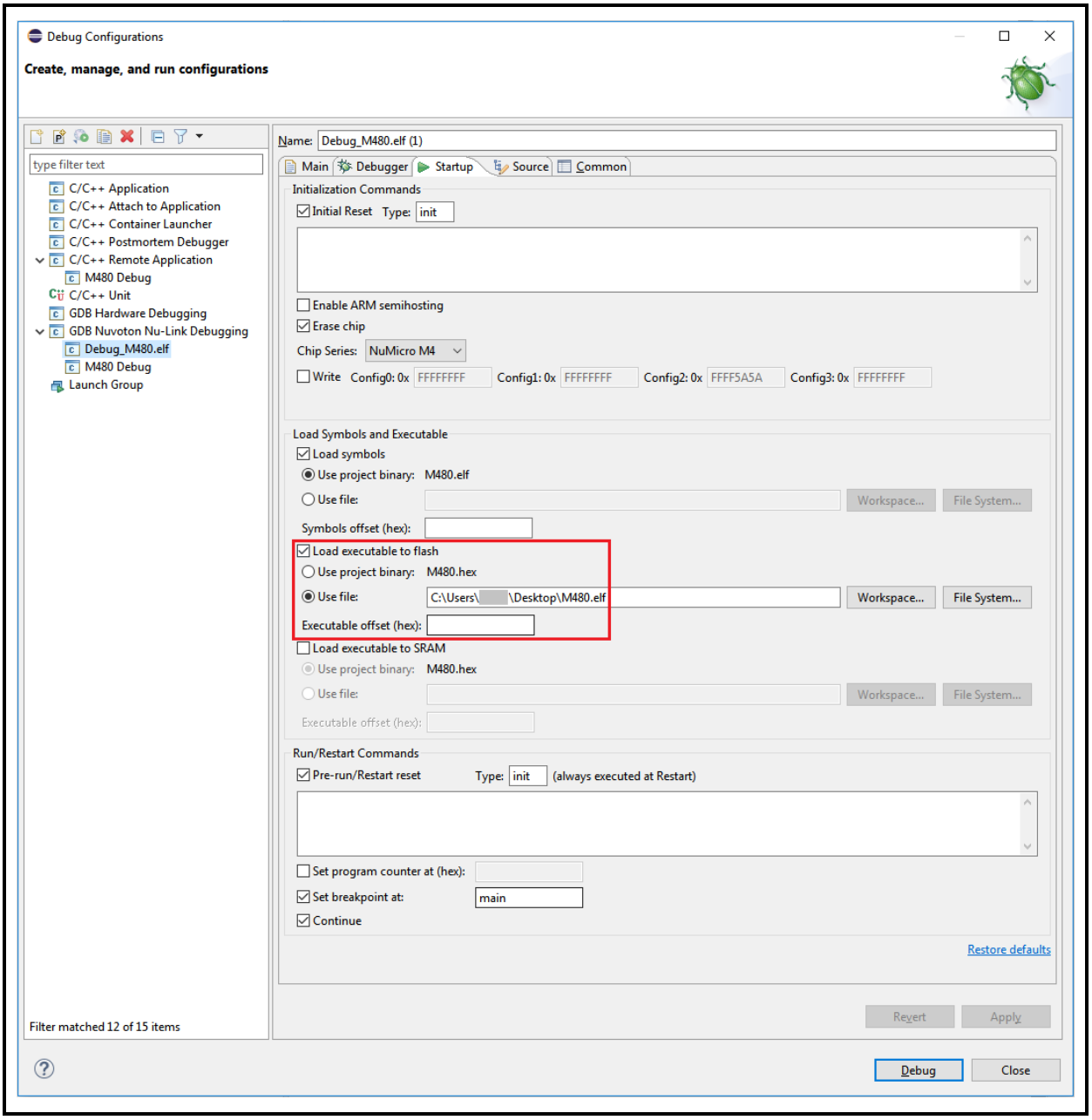

Figure 3-30 Choosing the ELF File to Download

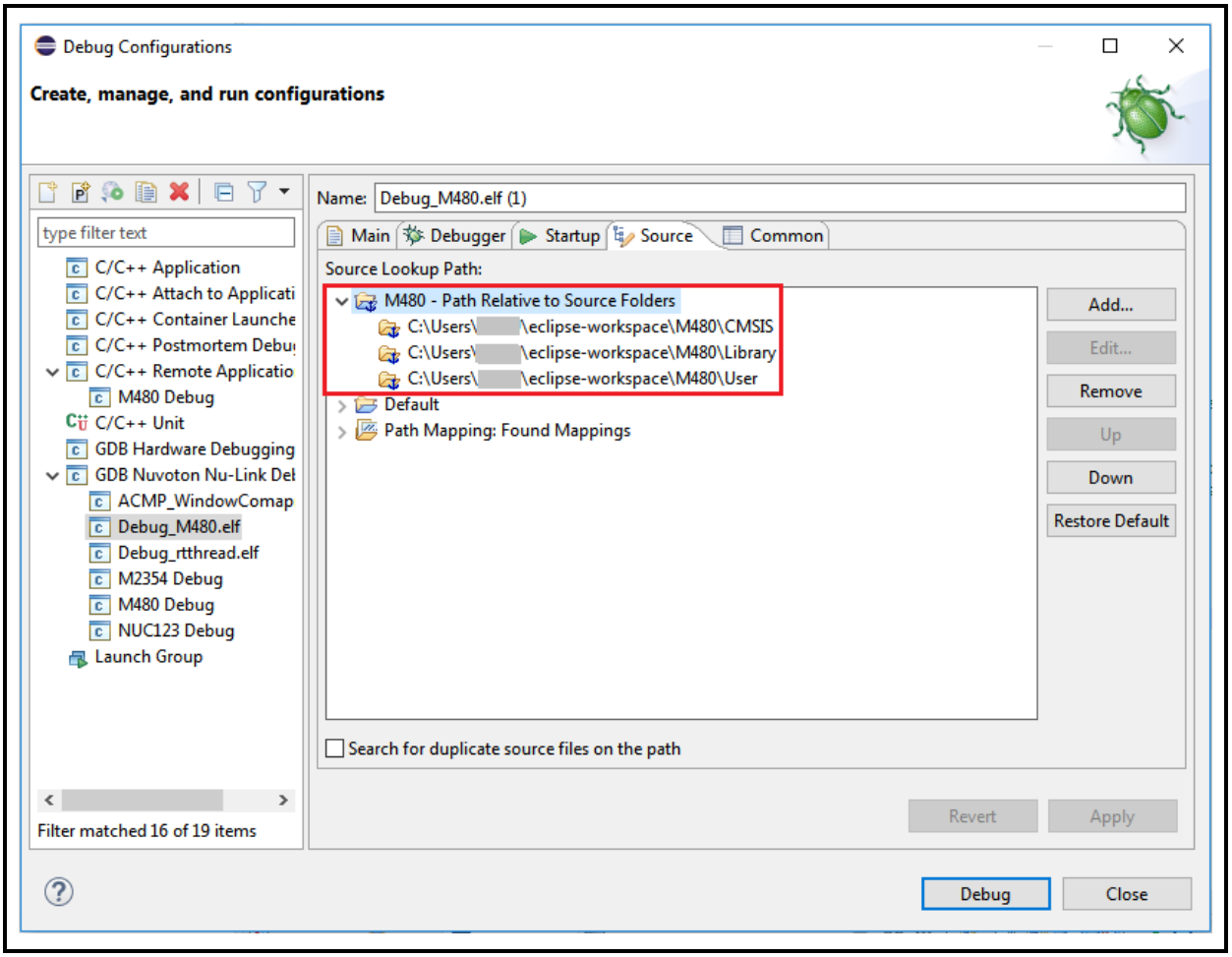

Figure 3-31 Adding Source Lookup Path

4 Q&A

1. Q: Can we simultaneously debug on Eclipse, Keil and Iar?

   A: No, we must stop the debug mode on Eclipse first. Then we can debug on another IDE.

2. Q: Can we simultaneously debug on Eclipse and use the Nuvoton development tools, such as ICP Programming tool?

   A: No, we must stop the debug mode on Eclipse first. Then we can use them and vice versa.

3. Q: How many breakpoints and watchpoints are supported?

   A: It depends on the hardware. For M0 chips, the supported number of breakpoints and watchpoints is 4 and 2, respectively. For M4 chips, the supported number of breakpoints and watchpoints is 6 and 4, respectively. For M23 chips, the supported number of breakpoints and watchpoints is 4 and 4, respectively. For now, we do not support flash breakpoints.

4. Q: How to update firmware for Nu-Link?

   A: Please use ICP Programming tool or Keil to update firmware.

5. Q: How to change Flash and RAM size after projects are created?

   A: Please find and open the ld script in the ld scripts folder. From there, we can change Flash and RAM size.

6. Q: Why can the application not enter the debug mode?

   A: Firstly, we must install all the stuff by following the previously mentioned steps. Then check the following list:

   I. Leave the previous debug mode first if exists.

   II. Flash and RAM size must be correct.

   III. OpenOCD should not be launched before debugging. To check that, please go to Windows Task Manager or System Monitor. If an OpenOCD process has already been running, please end the process.

   IV. The target chip should not be held by other tools or IDE.

   V. The **Config options** field of the Debugger tab must be correct.

   VI. In the Startup tab, the **Initilal Reset** type should be **init**. The **Pre-run/Restart rest** type should be **init**.

   VII. The Eclipse preferences must be correct. Please refer to the previous discussion.

   VIII. **Upgrade Nu-Link firmware and USB driver to the latest one**.

   IX. Check whether the executable has been downloaded to the target chip correctly.

   X. Check SP. If it is wrong, please assign it to the correct location.

   XI. Write a correct Config value into the target chip.

   XII. If the operating system is Windows 7, please use USB 2.0 port, instead of USB 3.0 port.

   XIII. The project path must not contain any special character or whitespace, such as #$&`.{}.

   XIV. Restart the computer.

7. Q: How to add udev rules for Nu-Link on GNU/Linux?

   A: When accessing target chips via Nu-Link, GNU/Linux requires the USB permission. We can get the permission by adding udev rules for Nu-Link. Here are the steps to do that:

   I. Add the User to the plugdev Group. Type the command in the Terminal:

      sudo useradd -G plugdev $USER

   II. Add Nu-Link to udev. Type the commands in the Terminal:

      cd /etc/udev/rules.d and **sudo gedit 10-openocd-nulink.rules**

   III. Add the following text to the file

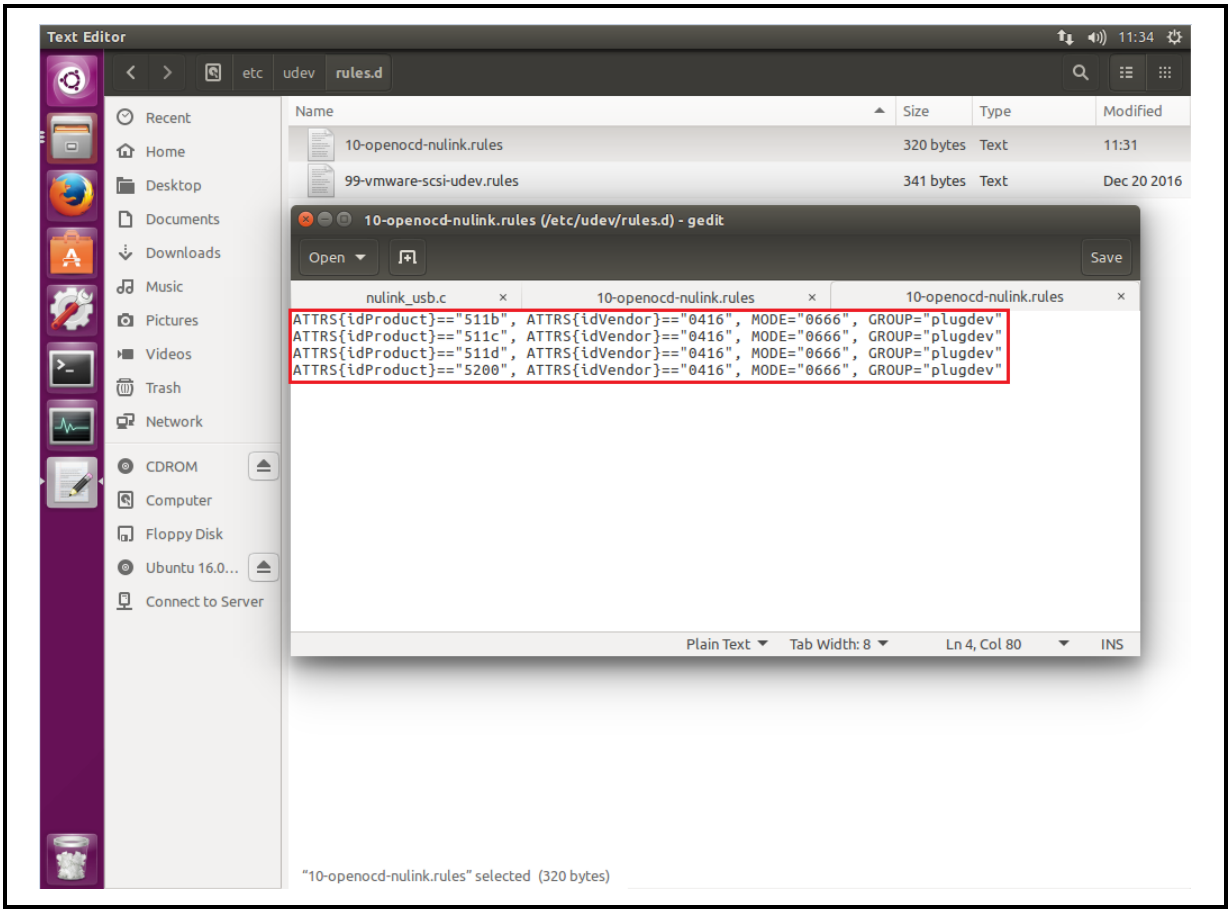

Figure 4-1 Adding Udev Rules

   IV. Reloaded the new udev rules by entering the command in the Terminal:

      sudo udevadm trigger

8. Q: How to edit string substitution for openocd_nulink_path?

   A: The **openocd_nulink_path** string stores where the OpenOCD executable resides. After upgrading NuEclipse, the string may keep the previous OpenOCD path. To fix it, click **Window > Preferences**, the Preferences wizard appears. Go to **Run/Debug > String Substituion**. Find and edit the openocd_nulink_path to the new OpenOCD path.

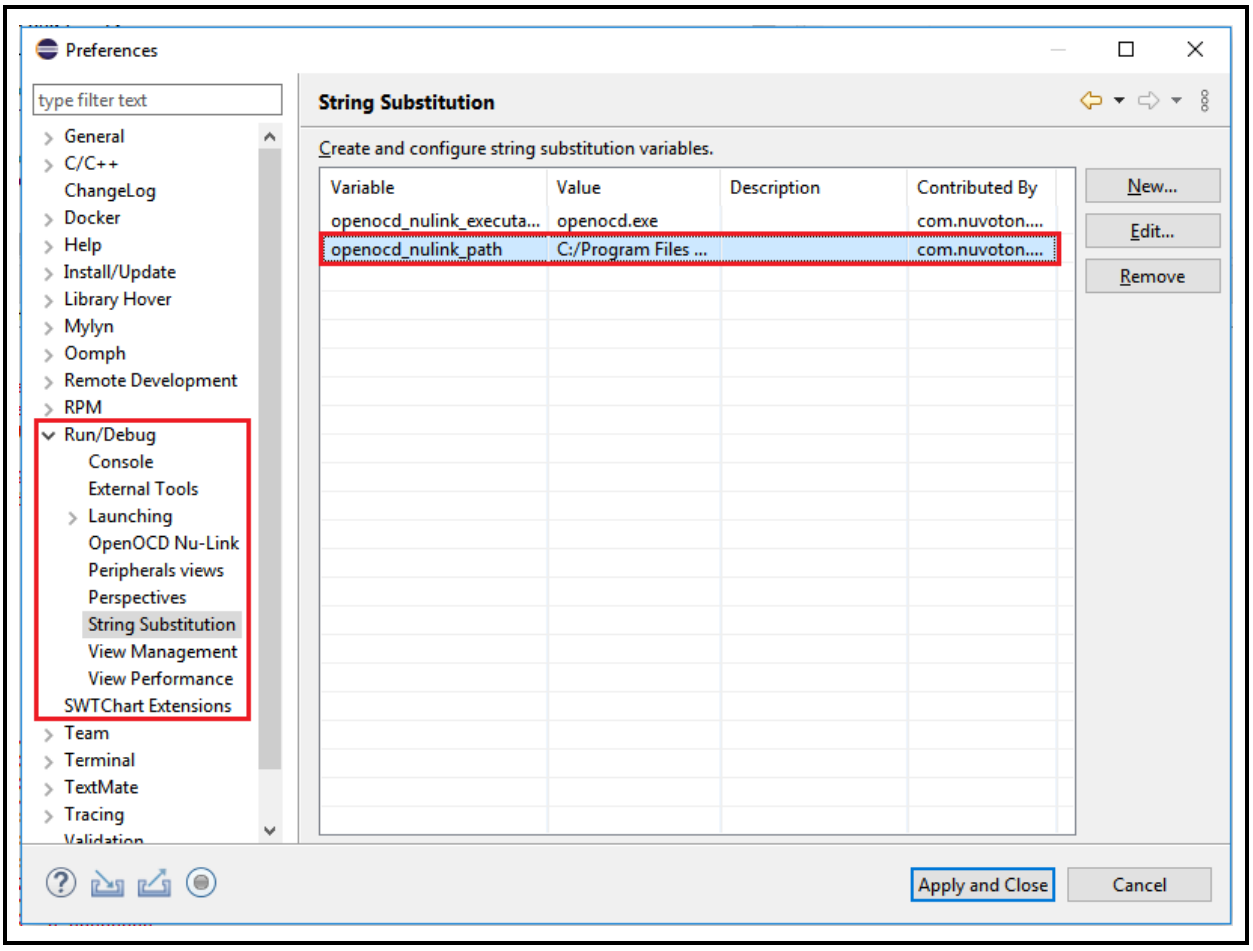

Figure 4-2 Preferences for String Substitution

9. Q: Why does Eclipse fail to update or install new software packs on Windows?

   A: One of possible reasons is that the write permission for Windows folders is denied. We need to find the correct folder and allow the write permission. On Windows, the location where we place software packs is C:\Program Files (x86)\Nuvoton Tools\Packages.

10. Q: How to adjust output voltage of Nu-Link2?

    A: Upgrade the NuLink2 firmware greater than version v3.08.7249, open NU_CFG.TXT file in pop-up "NuMicro MCU" disk, you will see some options in NU_CFG.TXT.

    Set CMSIS-DAP=1 then re-plug in USB cable, it presents one more interface HID_CMSIS-DAP, this is handy if you want to use CMSIS-DAP protocol.

    Set LEVEL=3 then re-plug in USB cable, it presents one of output voltage, this is handy if you want to use 5V voltage.

```
[Power Control]
LEVEL=3
; Default I/O voltage (Level0: 1.8V, Level1: 2.5V, Level2: 3.3V, Level3: 5V)

[Interface configuration]
CMSIS-DAP=1
; 0 = disable
; 1 = enable
```

Figure 4-3 More Options for NuLink2

5 Revision History

Date	Revision	Description
2017.03.31	0.01.000	Alpha version released.
2017.06.30	1.01.000	Beta version released.
2018.09.15	1.01.013	Official version released.
2018.11.30	1.01.014	1. Supported NUC505. 2. Updated the new project wizard.
2019.08.09	1.01.015	Supported M031 ,M261 and M480LD.
2020.03.06	1.01.016	Supported M031BT, NUC1311, M2354 and M479.
2020.09.30	1.01.017	Supported M030G, M071, M0A21, M251 and M471.
2021.03.12	1.01.018	Supported M030GM031G and NUC1262.
2021.12.01	1.01.019	Supported M460, I94100 and KM1M7.