ARM® Cortex®-M
32-bit Microcontroller

GNU Nuvoton Eclipse
User Manual

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

www.nuvoton.com

Table of Contents

List of Figures

1 Introduction

The **GNU Nuvoton Eclipse** 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 **GNU Nuvoton Eclipse**, 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 **GNU Nuvoton Eclipse**.

	Minimum Requirements	Recommended Specifications
Operating System	Windows®7 with latest service pack or GNU/Linux	Windows®10 with latest service pack or Ubuntu 16.04.2 LTS
GNU ARM Embedded Toolchain	6-2016-q4-major	The latest version
Java	JDK 1.7 or JRE 1.7	The latest JDK
Eclipse IDE	Eclipse 4.4 Luna SR2	Eclipse 4.5 Mars SR2
Eclipse CDT	Eclipse CDT 8.6	The latest Eclipse CDT

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 **GNU Nuvoton Eclipse** ready for work, perform the following steps based on your operating system:

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

2.3.1 Performing the GNU Nuvoton Eclipse installer on Microsoft Windows

On Windows, it is very easy to install the GNU Nuvoton Eclipse only by performing the GNU Nuvoton Eclipse 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 GNU Nuvoton Eclipse.

Figure 2-1 GNU Nuvoton Eclipse Setup Wizard

2.3.2 Extracting the GNU Nuvoton Eclipse Tar File on GNU/Linux

On GNU/Linux, it is very easy to install the GNU Nuvoton Eclipse only by extracting the GNU Nuvoton Eclipse 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.

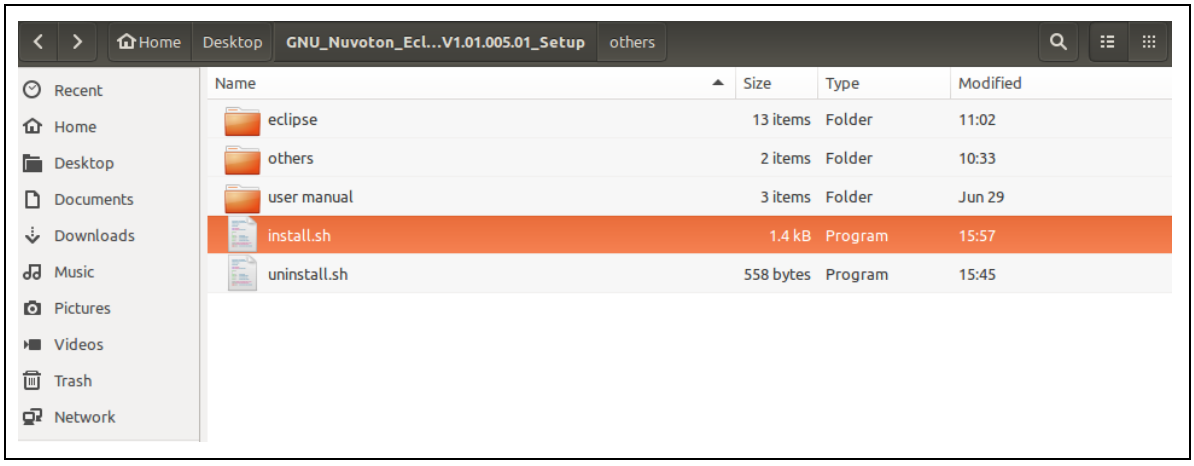


Figure 2-2 Install.sh Script

.

2.3.3 Verifying the Eclipse Preferences

After the installation of **GNU ARM Eclipse** is completed, 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.

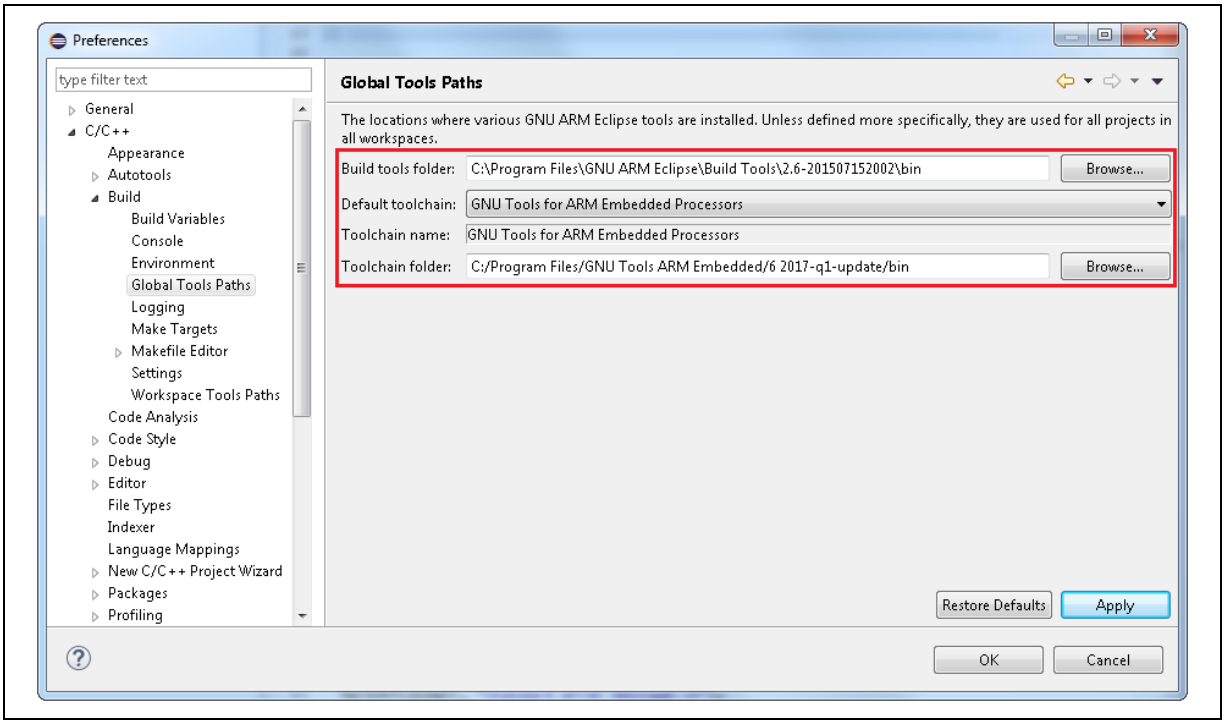


Figure 2-3 Preferences for Global Tools Paths

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/Nuvoton Tools/OpenOCD. Similarily, 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.

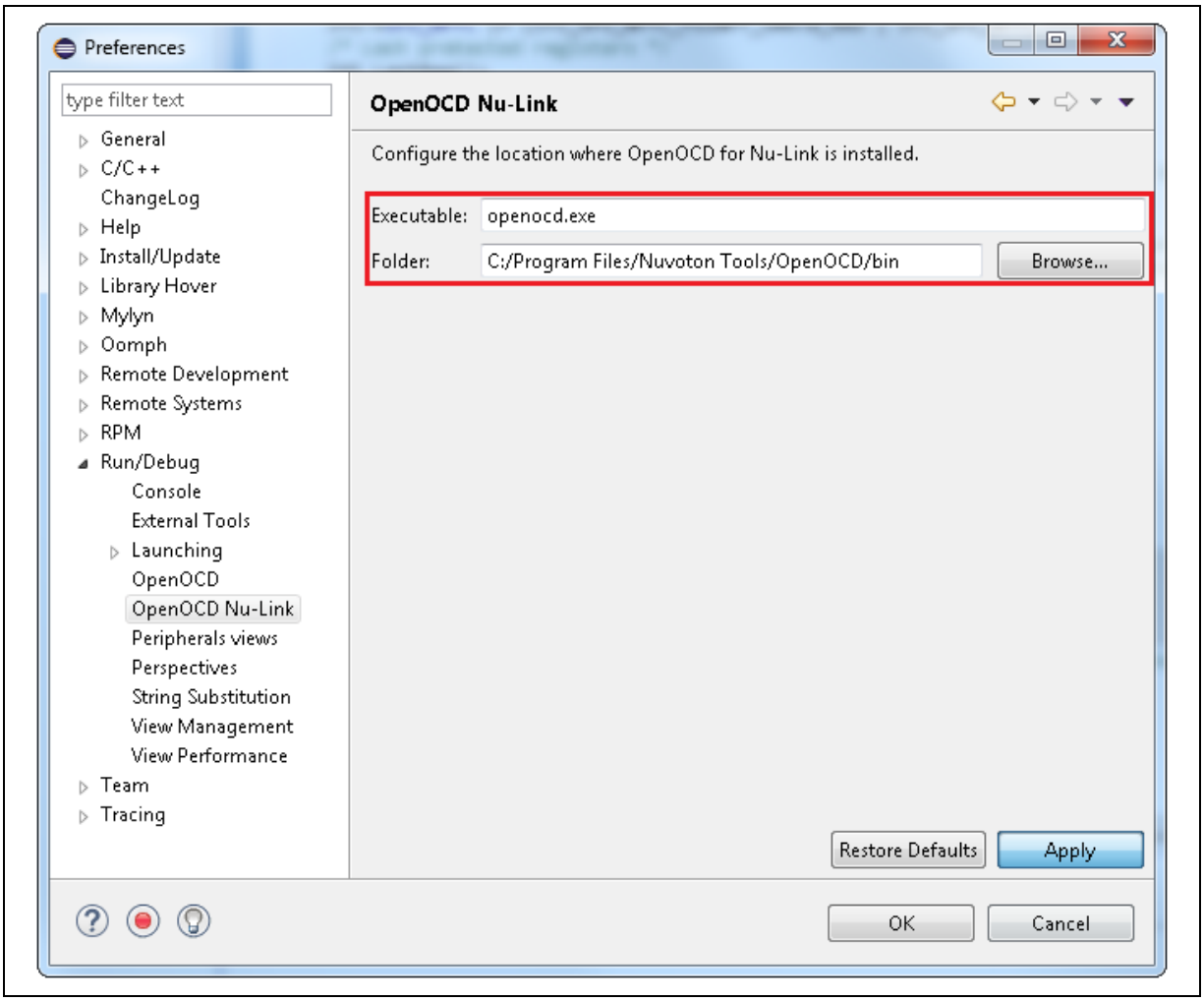


Figure 2-4 Preferences for OpenOCD Nu-Link

2.4 Running Eclipse

To run **GNU Nuvoton Eclipse**, 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.

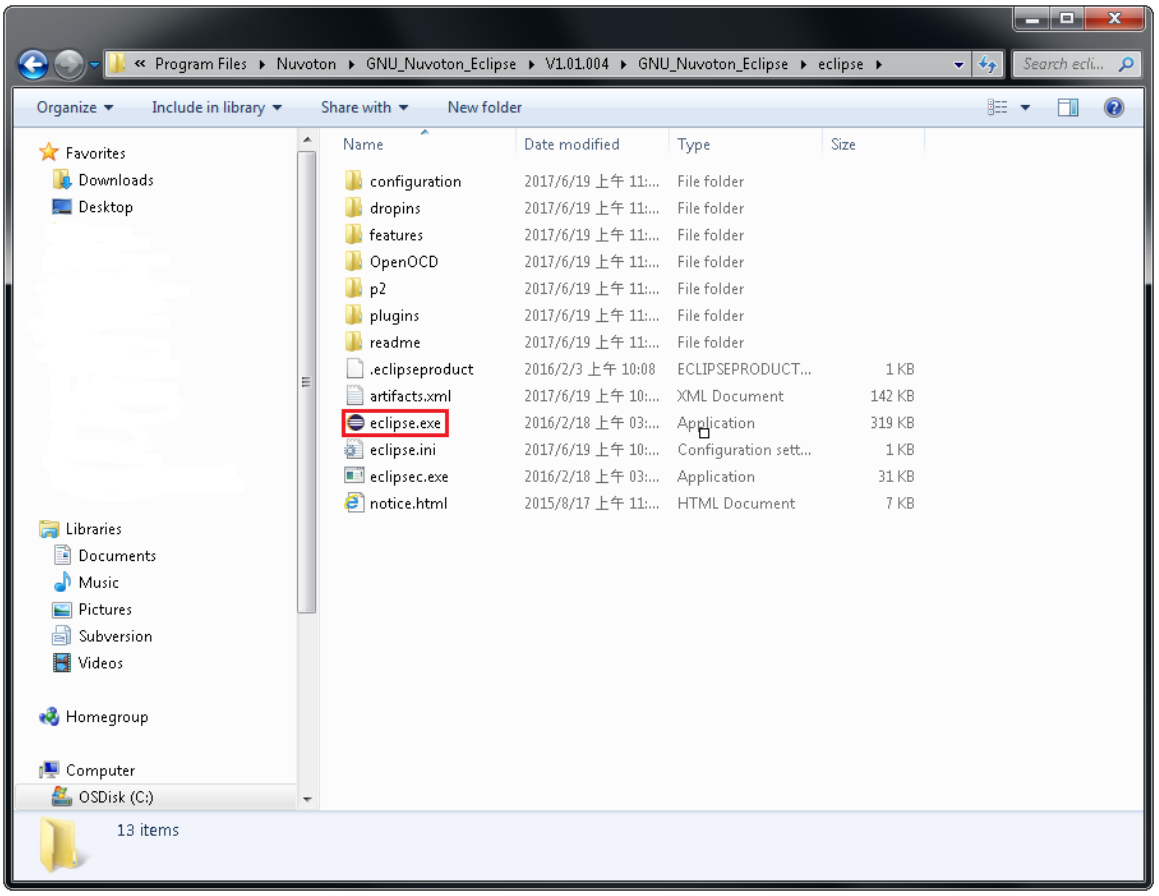


Figure 2-5 Eclipse.exe and Related Folders

3 Development Tutorial

3.1 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.

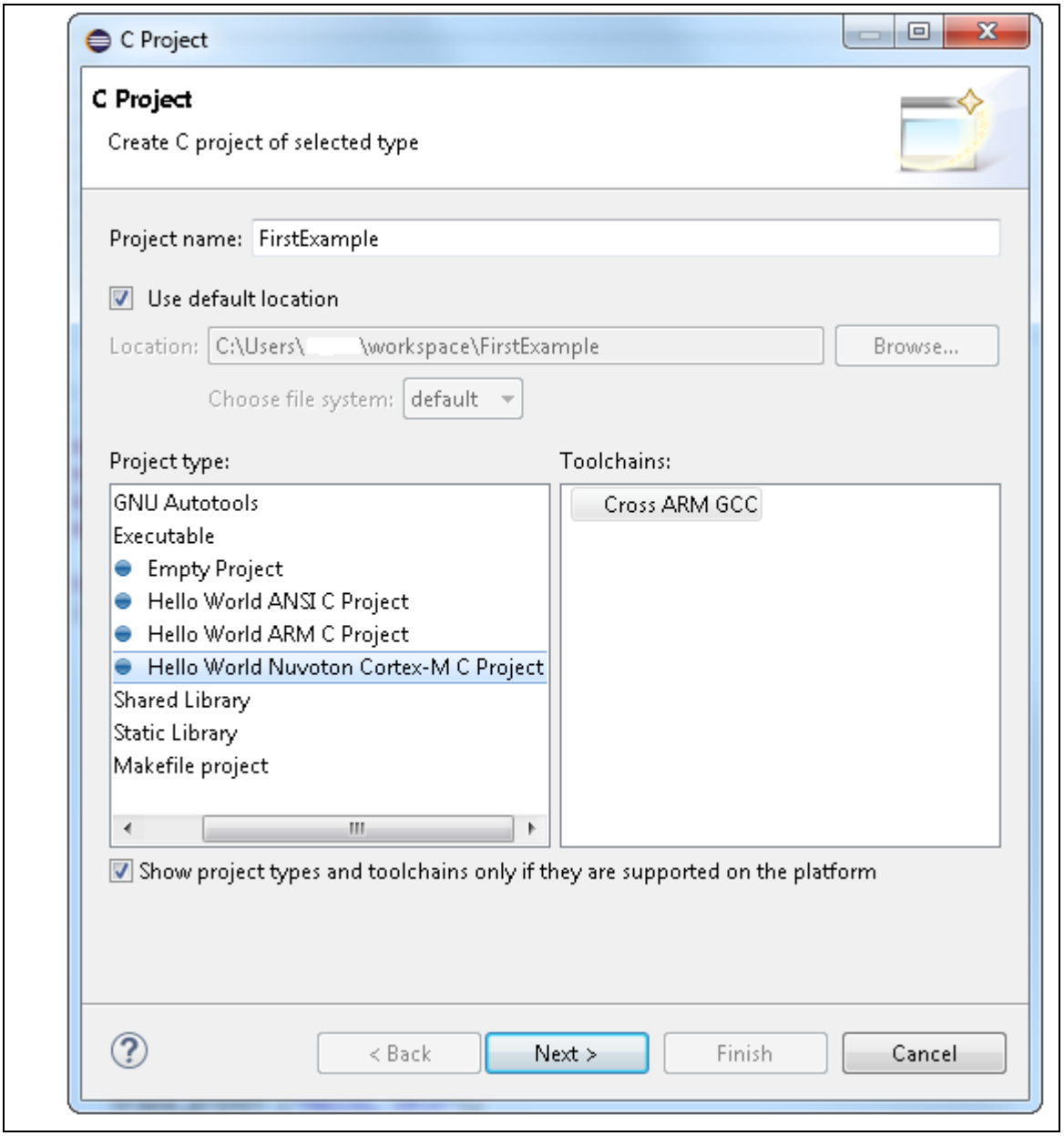


Figure 3-1 New Project Wizard

Based on the actual target chip, we select the corresponding chip series. To enable the Semihosting feature, we select **Semihosting (POSIX system calls via host)** for **Use system calls**. 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.

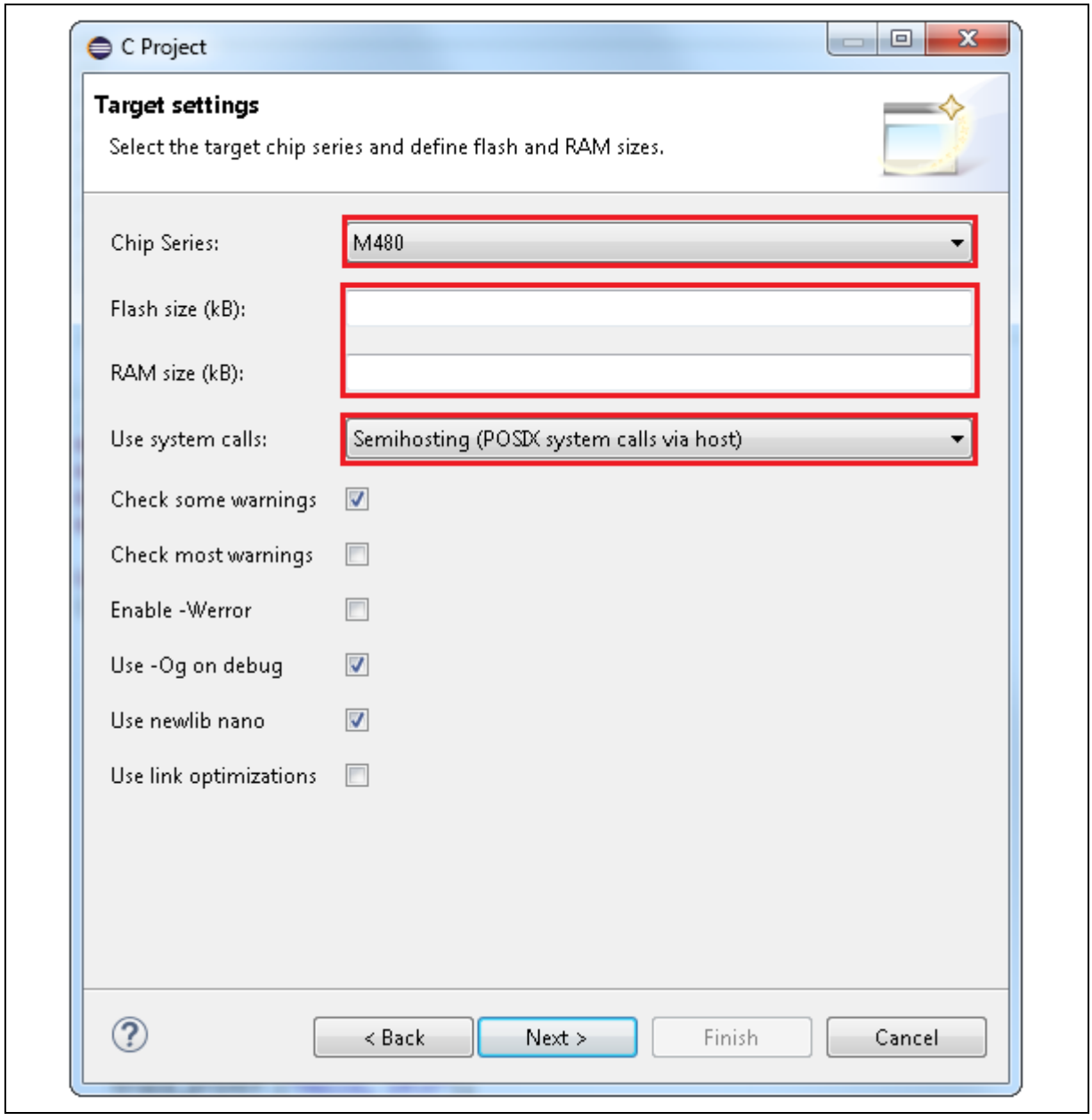


Figure 3-2 Target Processor Settings

3.2 Import Existing Projects

When BSP is 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 folders (refer to the following picture).
4. Under **Projects** select the project or projects which you would like to import and click **Finish**.

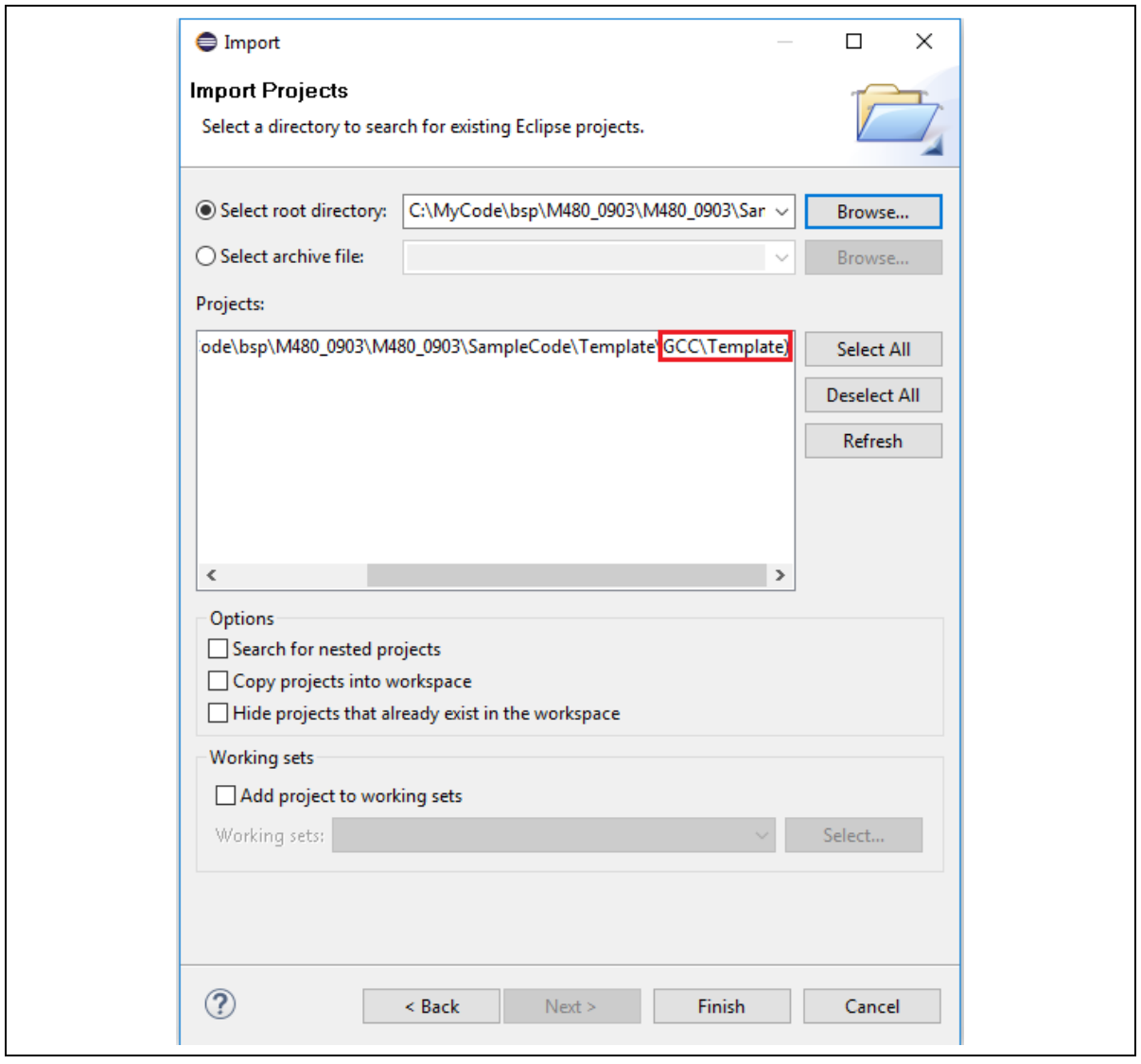


Figure 3-3 Importing Projects

3.3 Build Settings

After projects are 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.

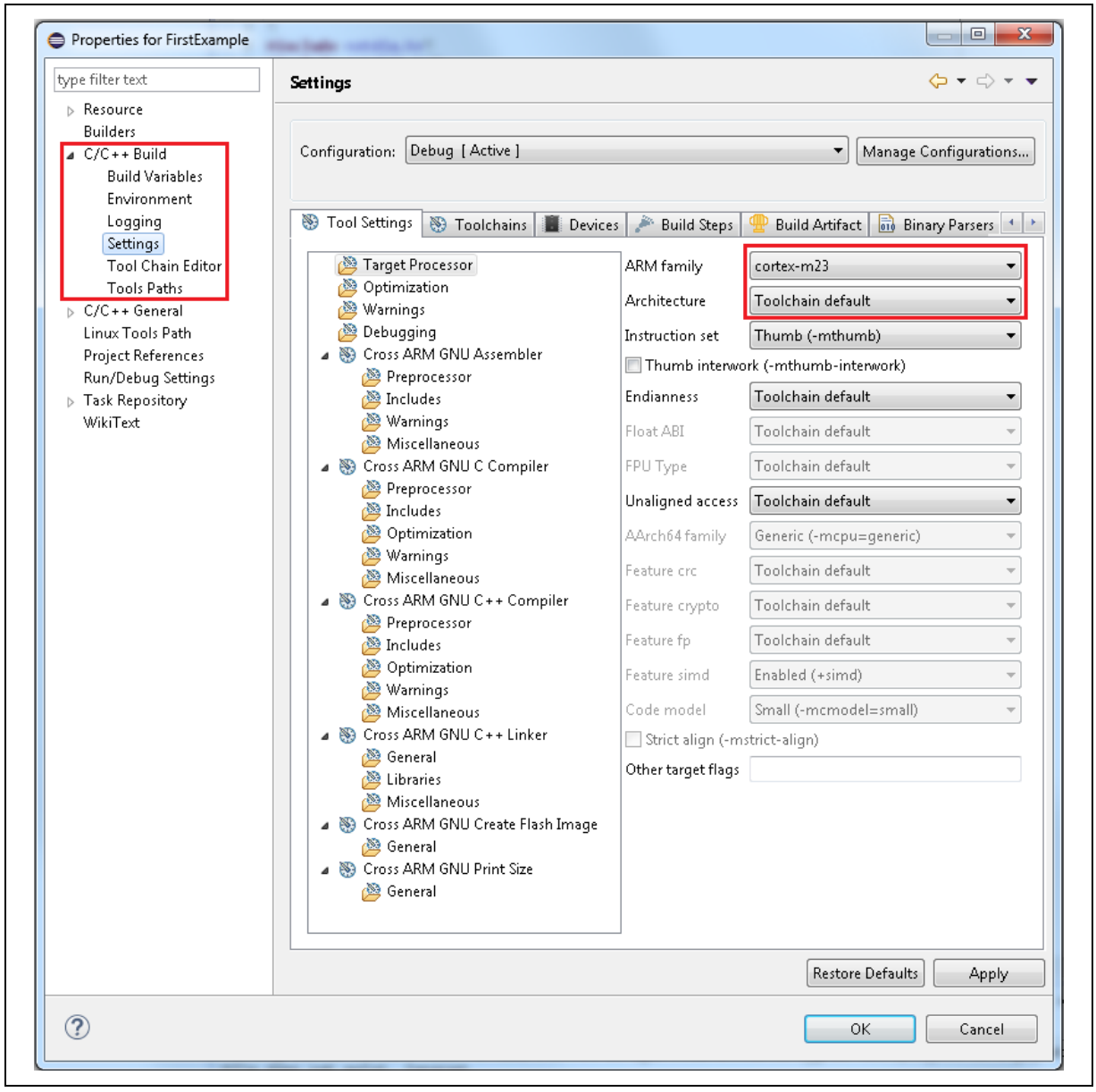


Figure 3-4 Build Settings

3.4 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 and build the project to generate the executable. Then repeat the former operations again.

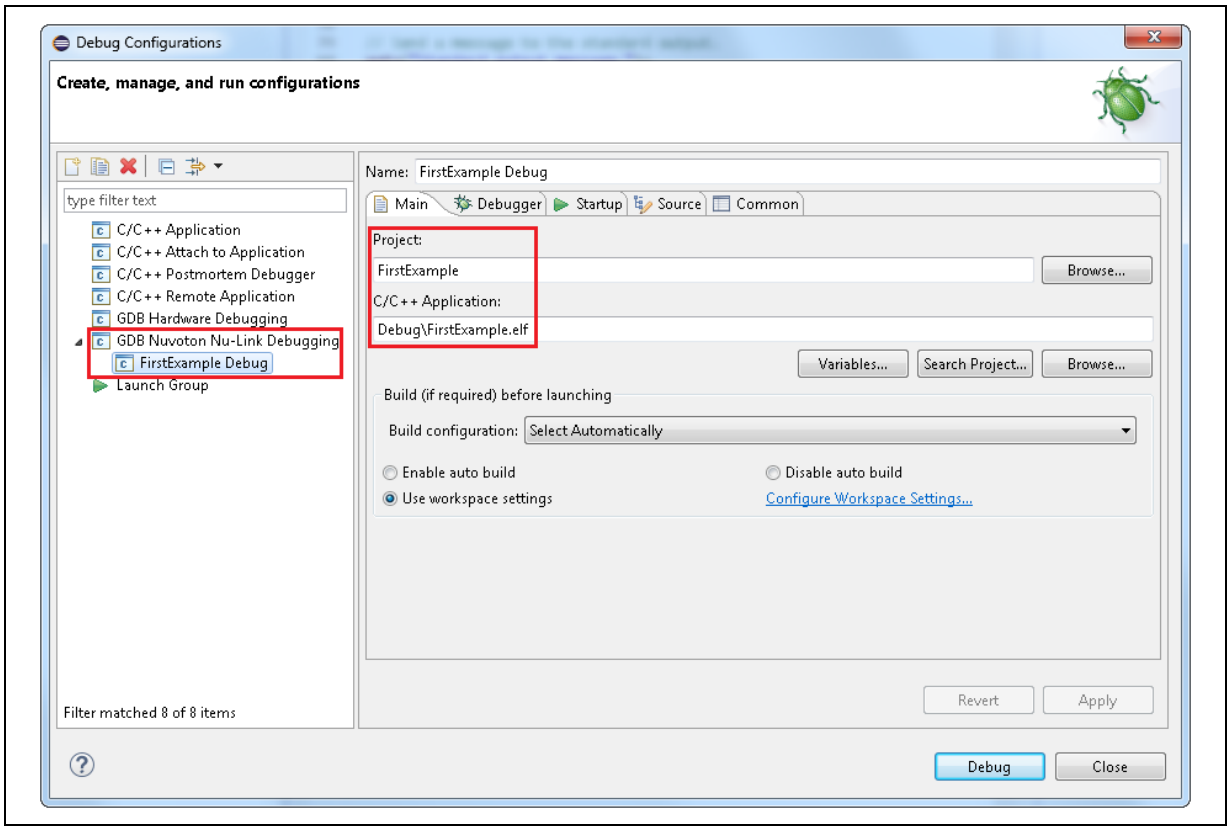


Figure 3-5 Debug Configuration

3.4.1 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's 2nd development, the target configuration file would be **numicroM23_NS.cfg**.

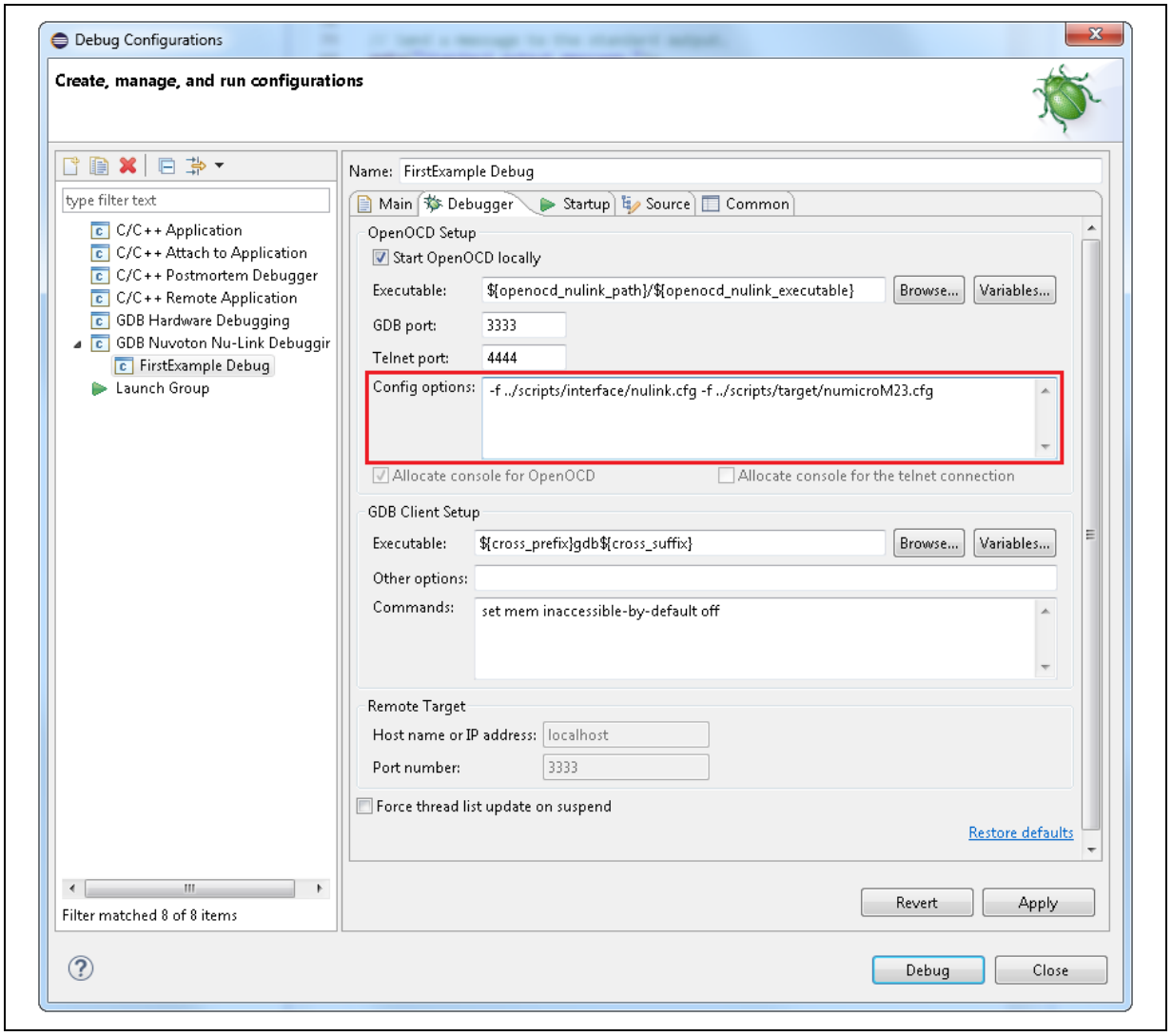


Figure 3-6 Configuring the Debugger Tab

3.4.2 Startup Tab

In the Startup tab, the user can determine whether to enable features, such as Erase chip, by selecting or unselecting the corresponding checkbox. 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.

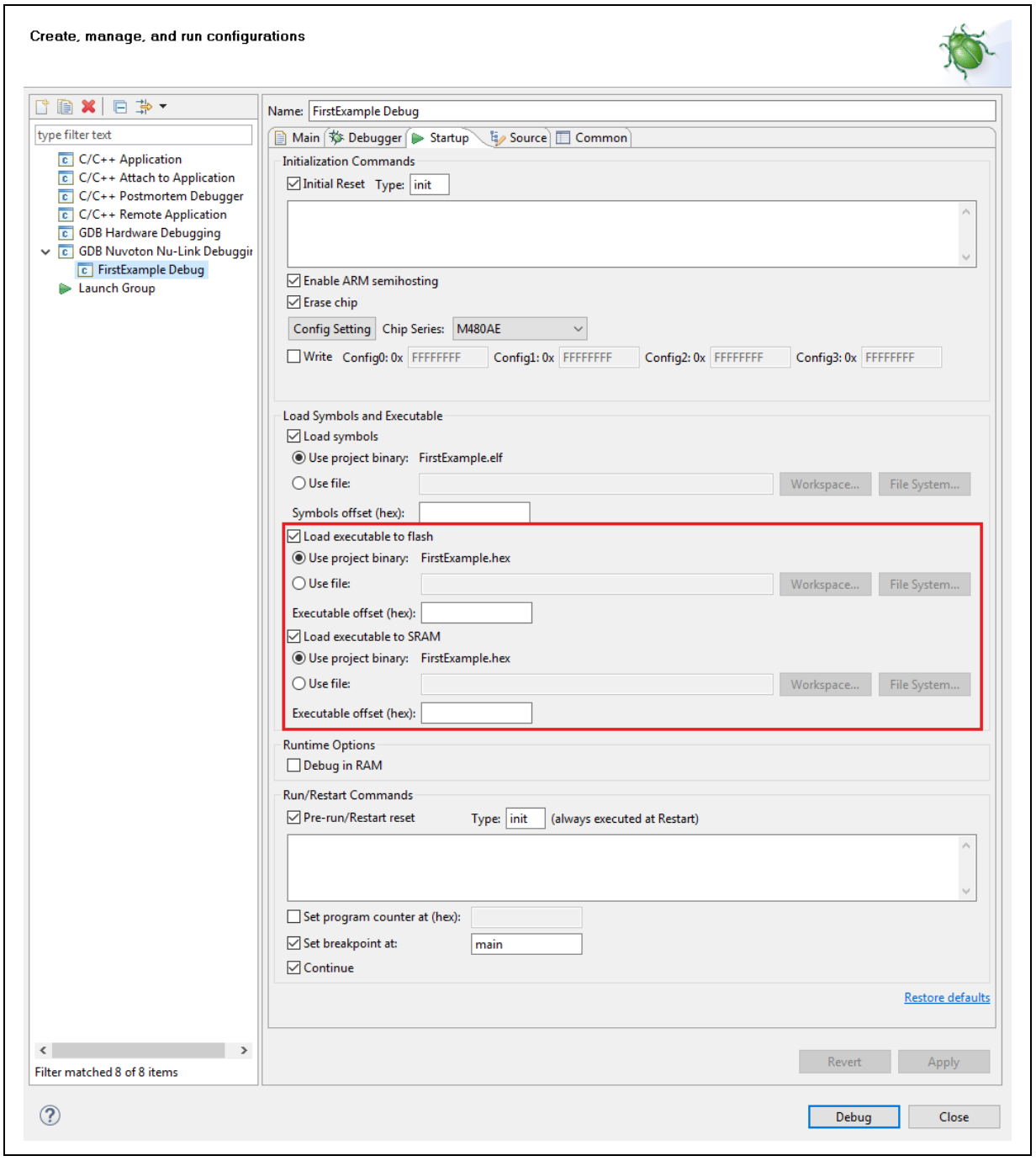


Figure 3-7 Configuring the Startup Tab

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. Besides, we can easily configure the Config values by clicking the **Config Setting** button. A Chip Options dialog shows up. From the dialog, we can configure chip options. When finished, click the **Ok** button. The new Config values will be updated in the Startup tab. Click the **Debug** button to start a debug session.

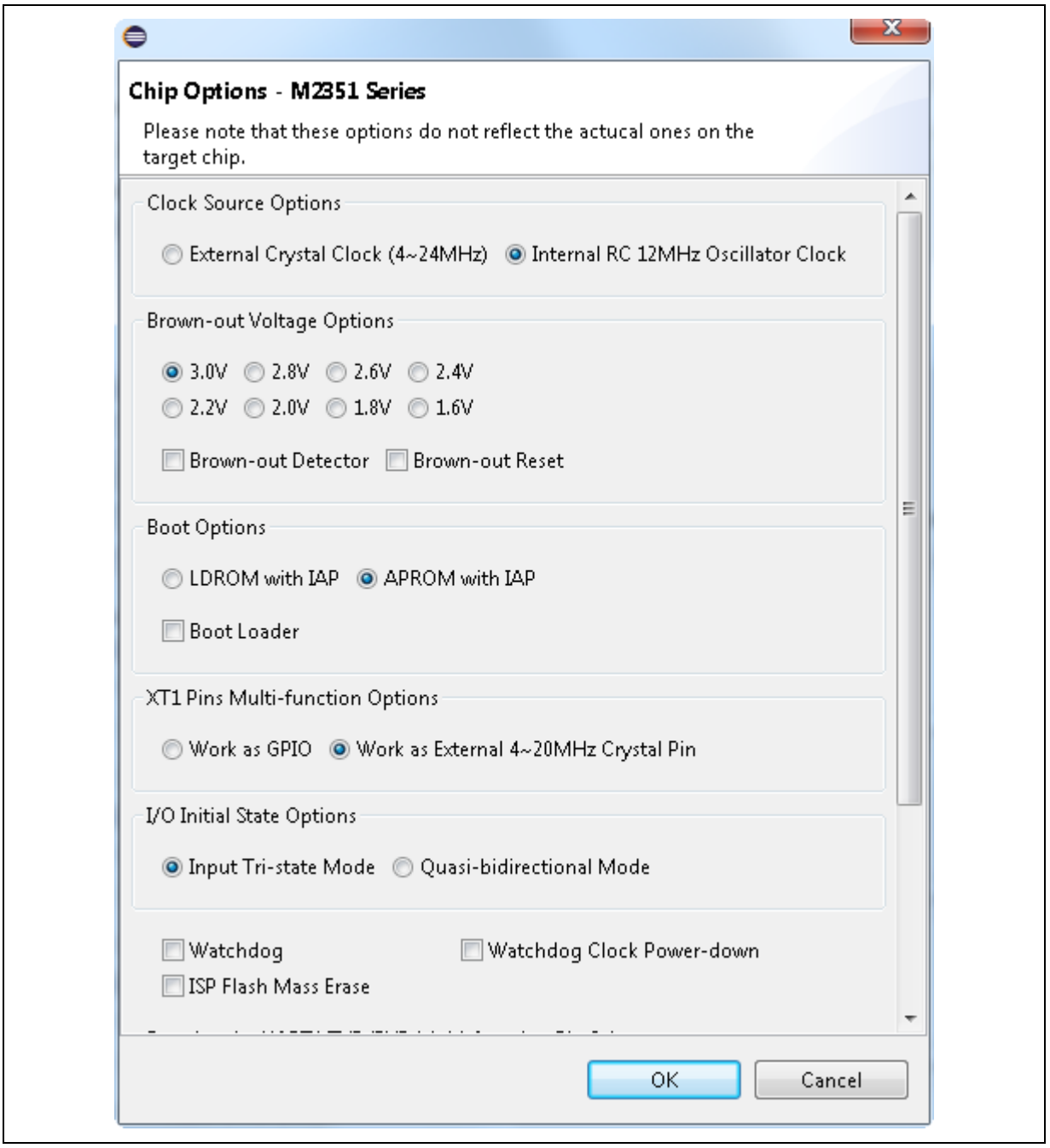


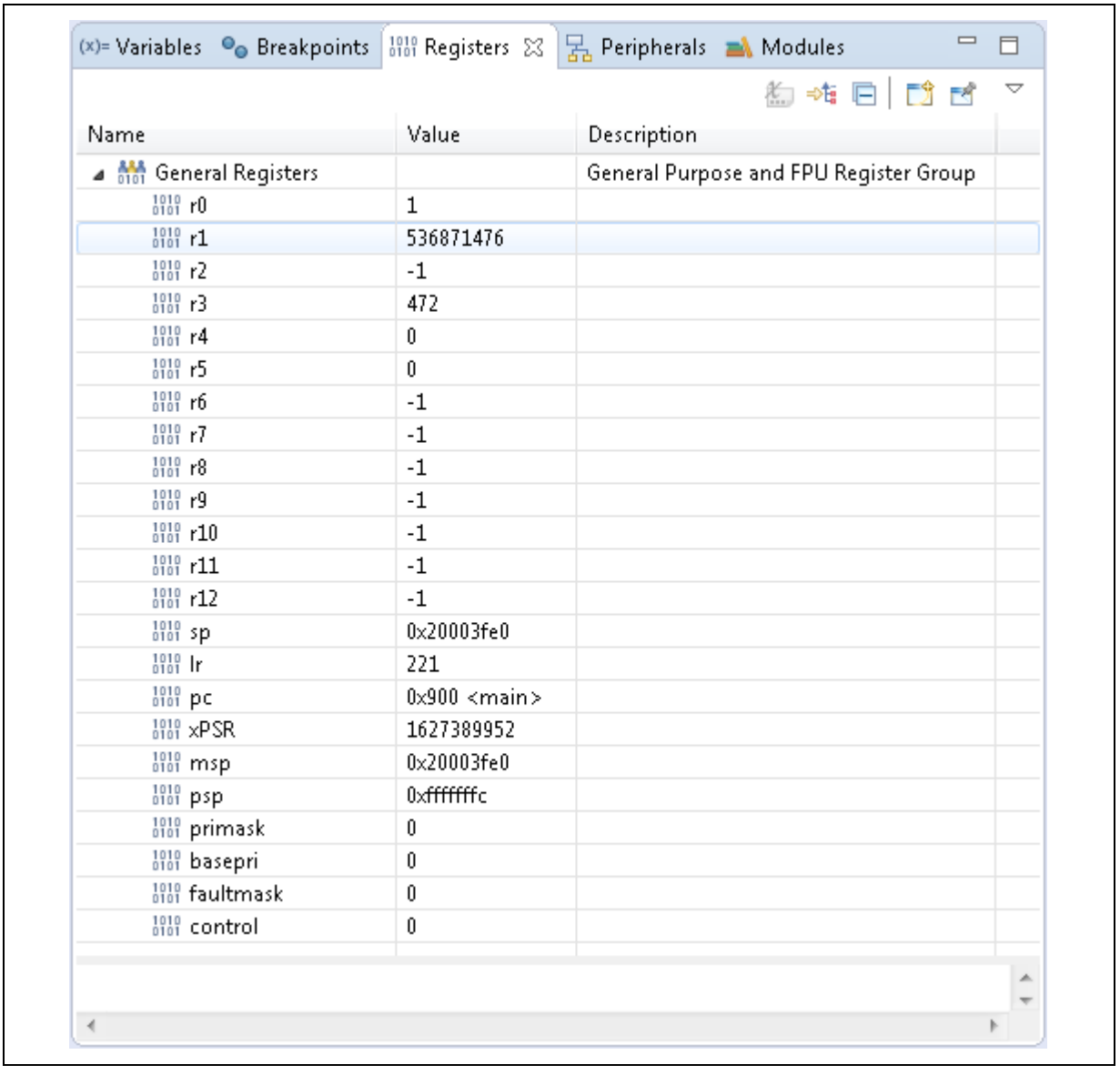
Figure 3-8 Chip Options Dialog

3.5 Debug Views

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

3.5.1 Registers View

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



Name	Value	Description
General Registers		General Purpose and FPU Register Group
r0	1	
r1	536871476	
r2	-1	
r3	472	
r4	0	
r5	0	
r6	-1	
r7	-1	
r8	-1	
r9	-1	
r10	-1	
r11	-1	
r12	-1	
sp	0x20003fe0	
lr	221	
pc	0x900 <main>	
xPSR	1627389952	
msp	0x20003fe0	
psp	0xfffffffc	
primask	0	
basepri	0	
faultmask	0	
control	0	

Figure 3-9 Registers View

3.5.2 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.

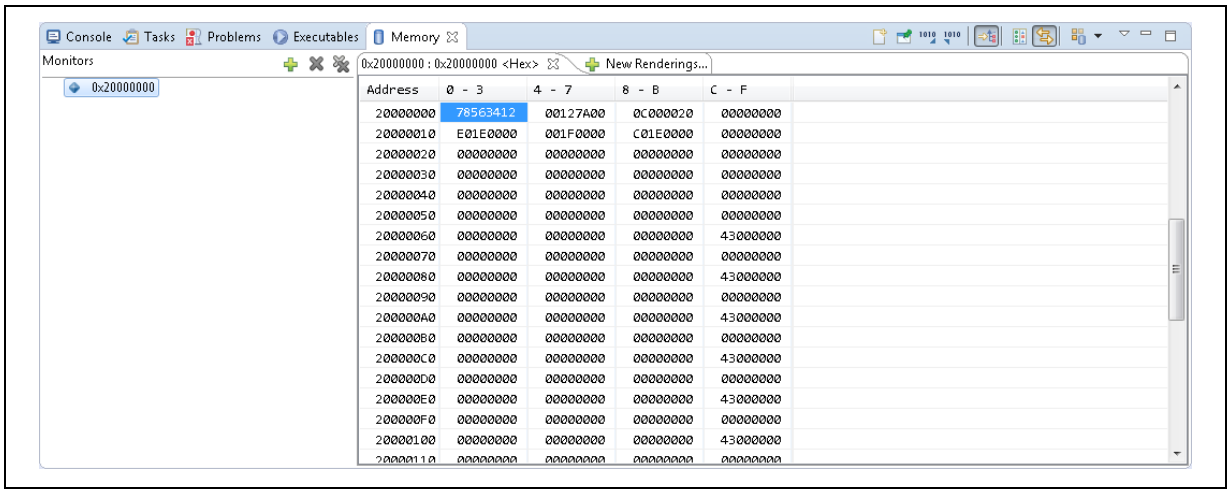


Figure 3-10 Memory View

3.5.3 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.

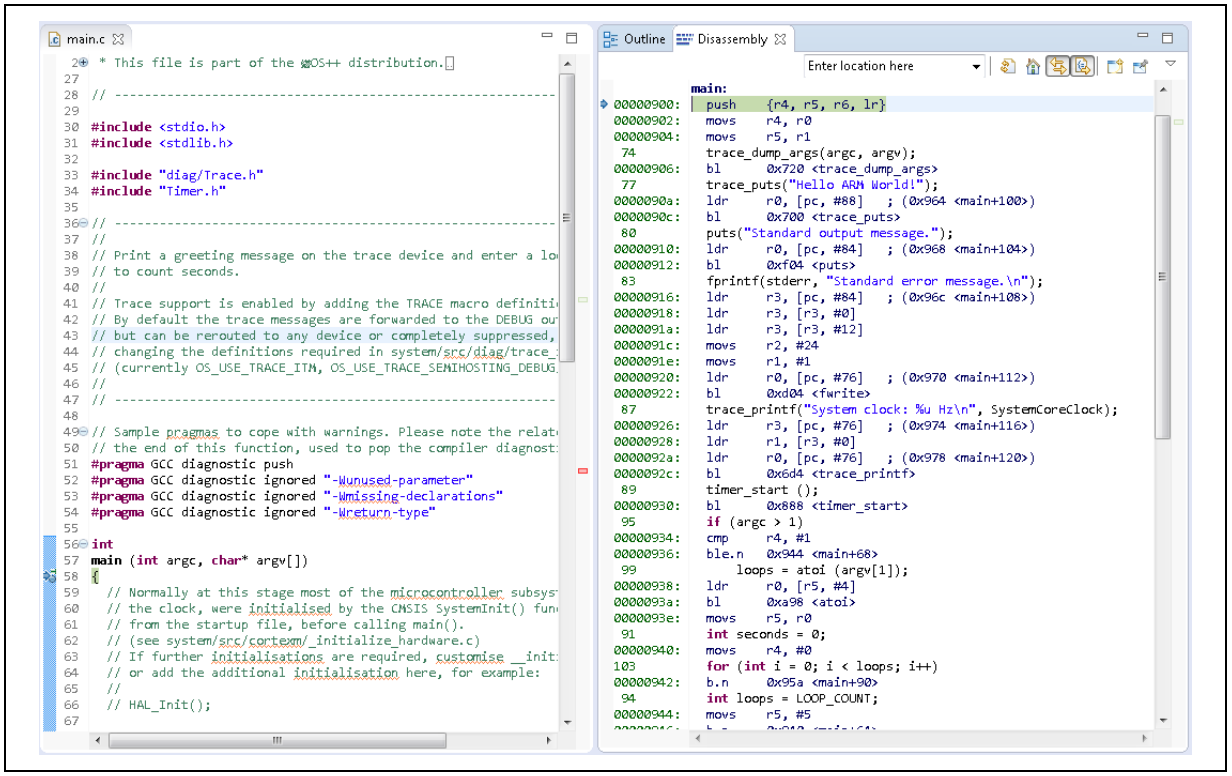


Figure 3-12 Disassembly View

3.5.4 Peripheral Registers View

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

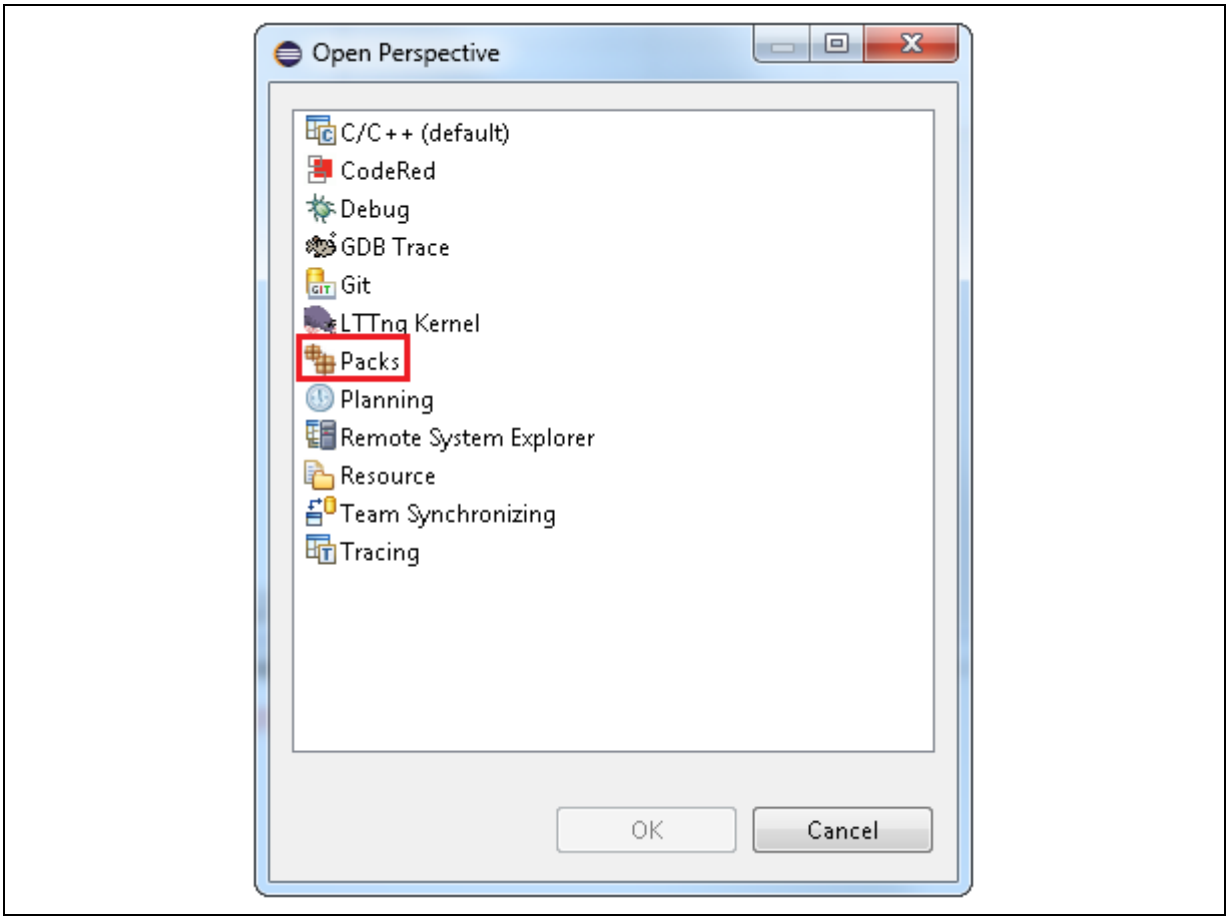


Figure 3-13 Opening the Packs Perspective

For the first time we see the Packs perspective, its content is provided by the GNU Nuvoton Eclipse installer. 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 a repository.

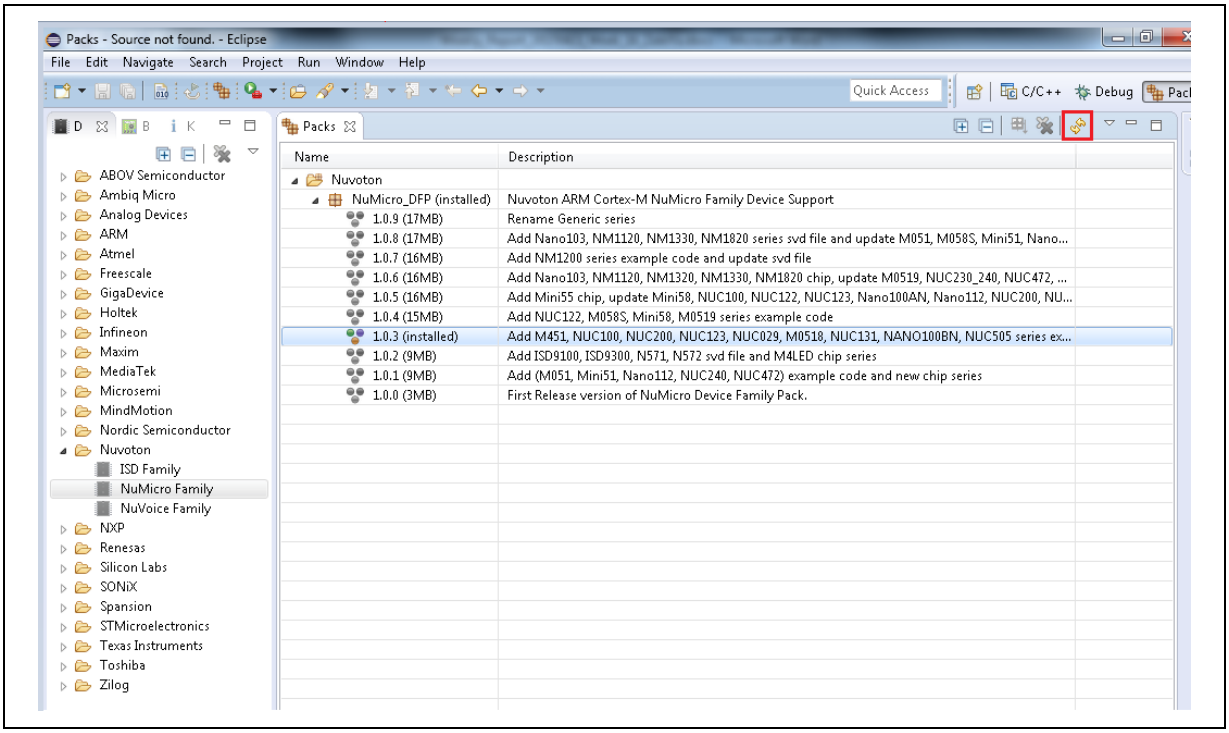


Figure 3-14 How to Download Packages

The locations of repositories are specified in the **Window > Preferences > C/C++ > Packages > Repositories**. One of them is from Keil's CMSIS pack. To speed up the download time, we can delete another repository (GNU ARM Eclipse).

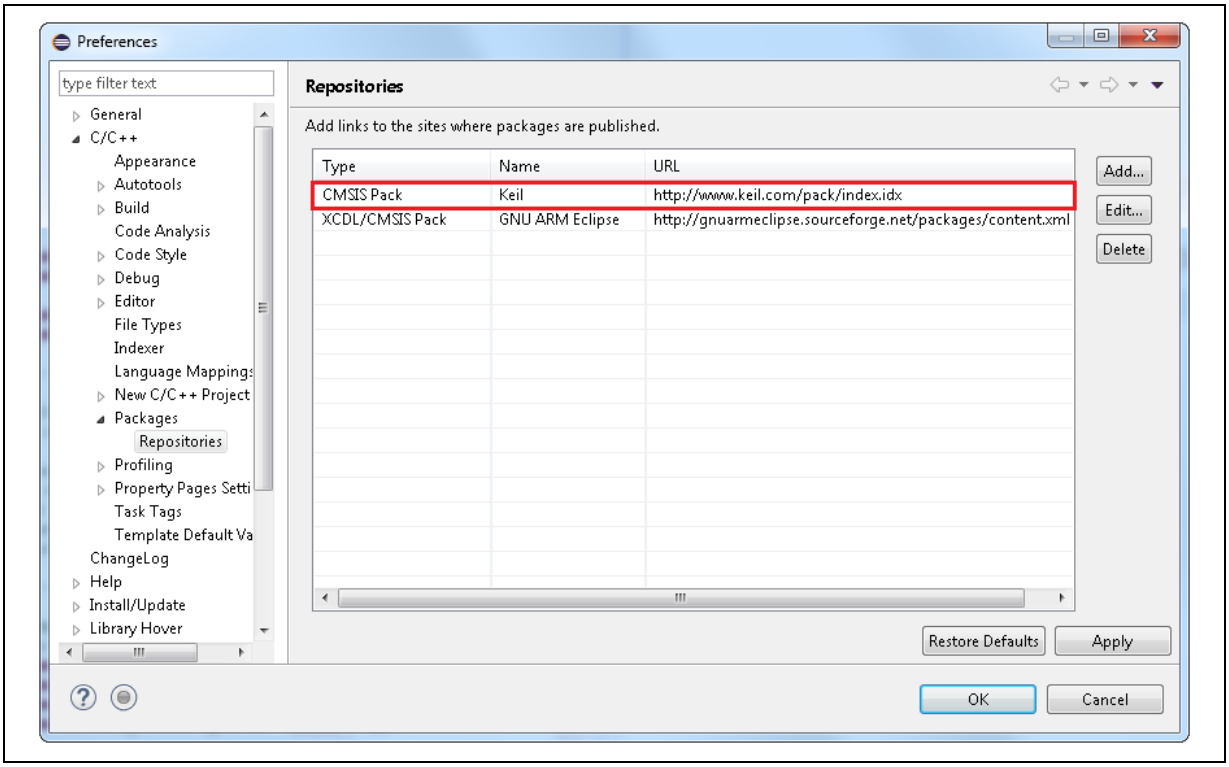


Figure 3-15 Locations of Repositories

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

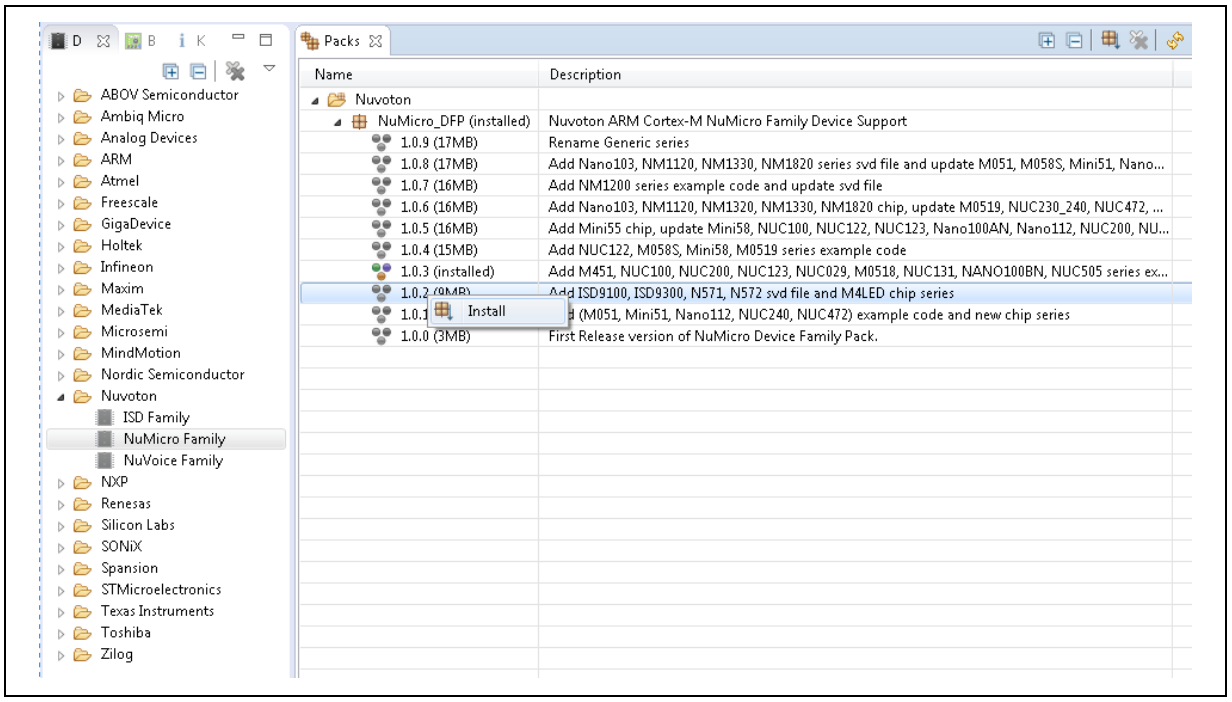


Figure 3-16 Installing SFR Files

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 NUC140VE3CN. Click the **Apply** button to take effect.

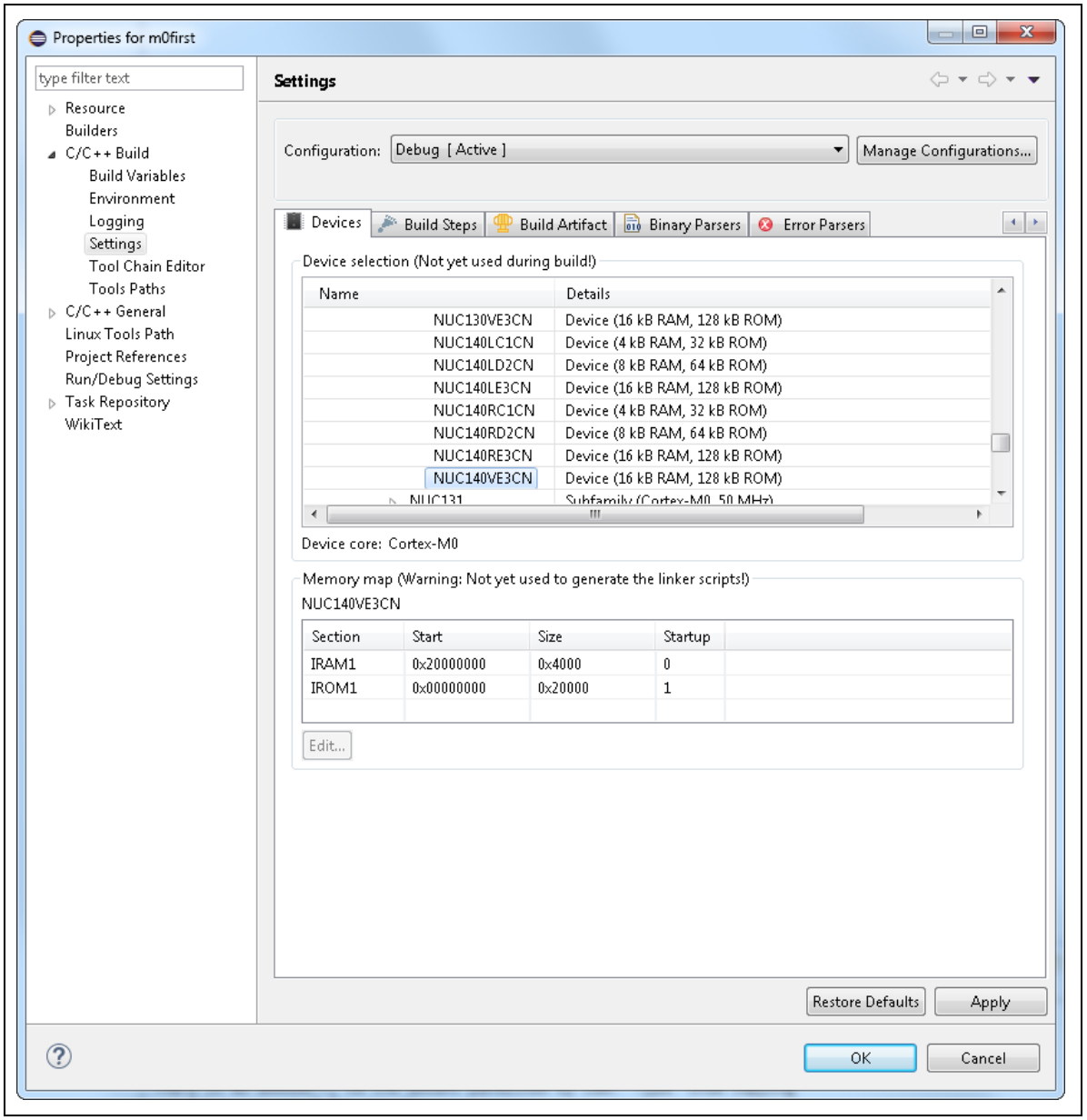


Figure 3-17 Device Selection

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

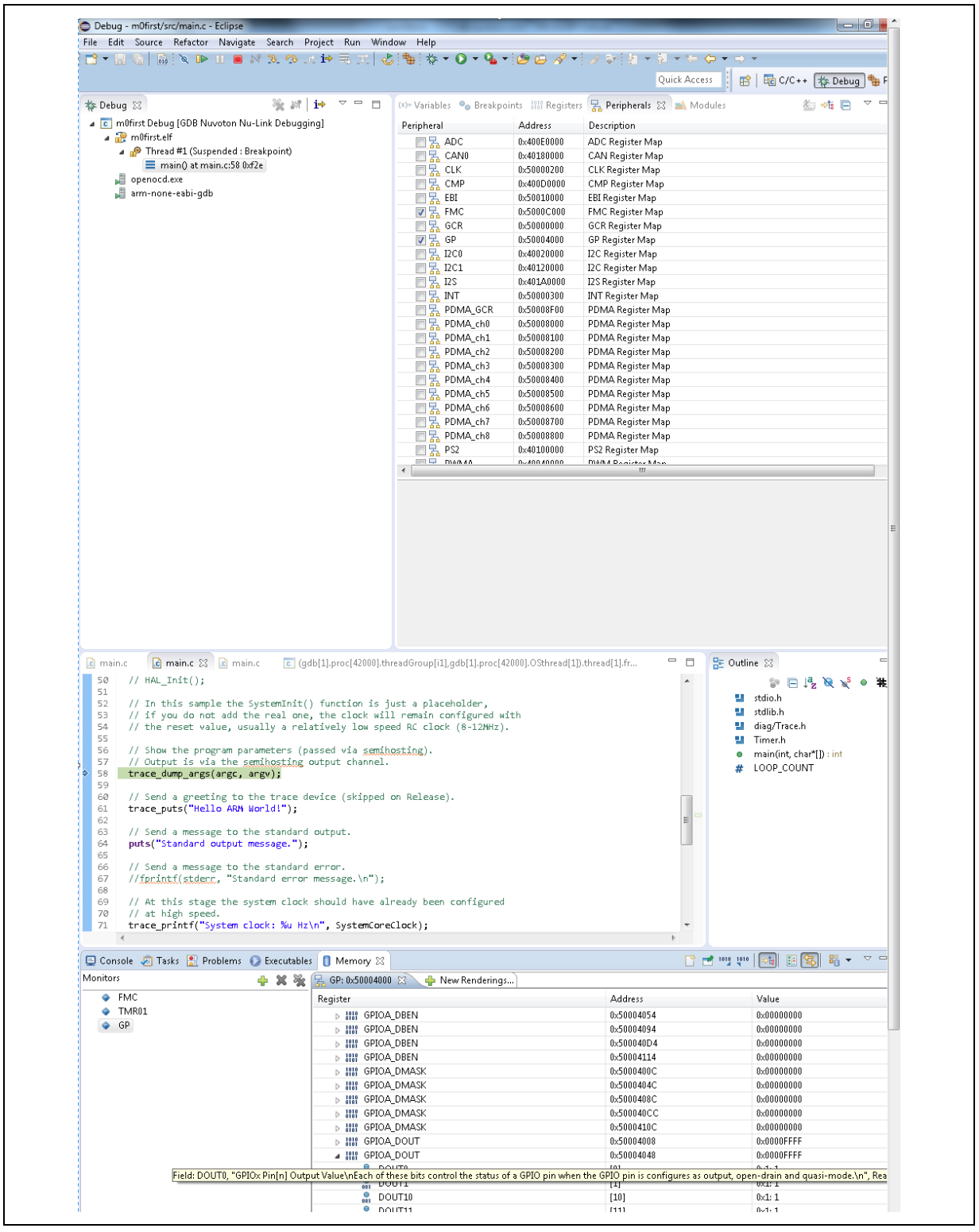


Figure 3-18 Peripheral Registers View

3.6 **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**.

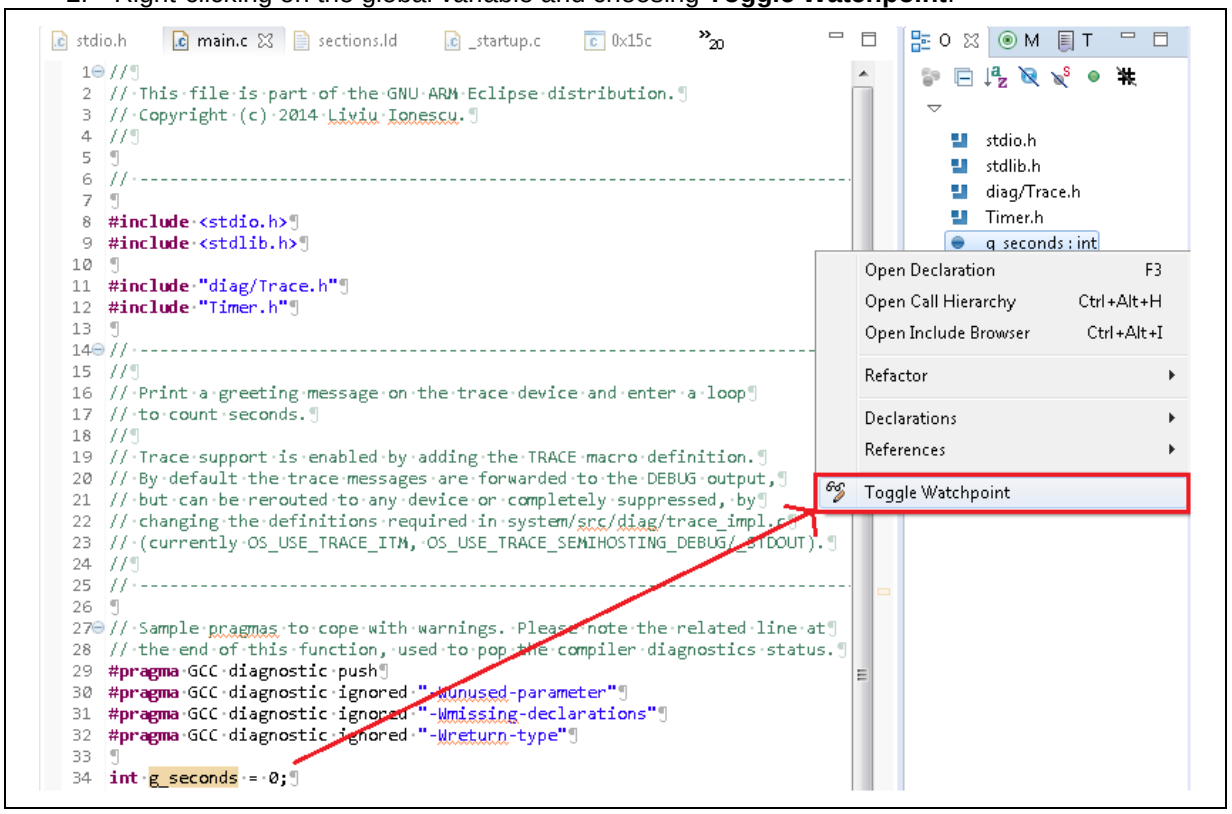


Figure 3-19 Toggle Watchpoint

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.

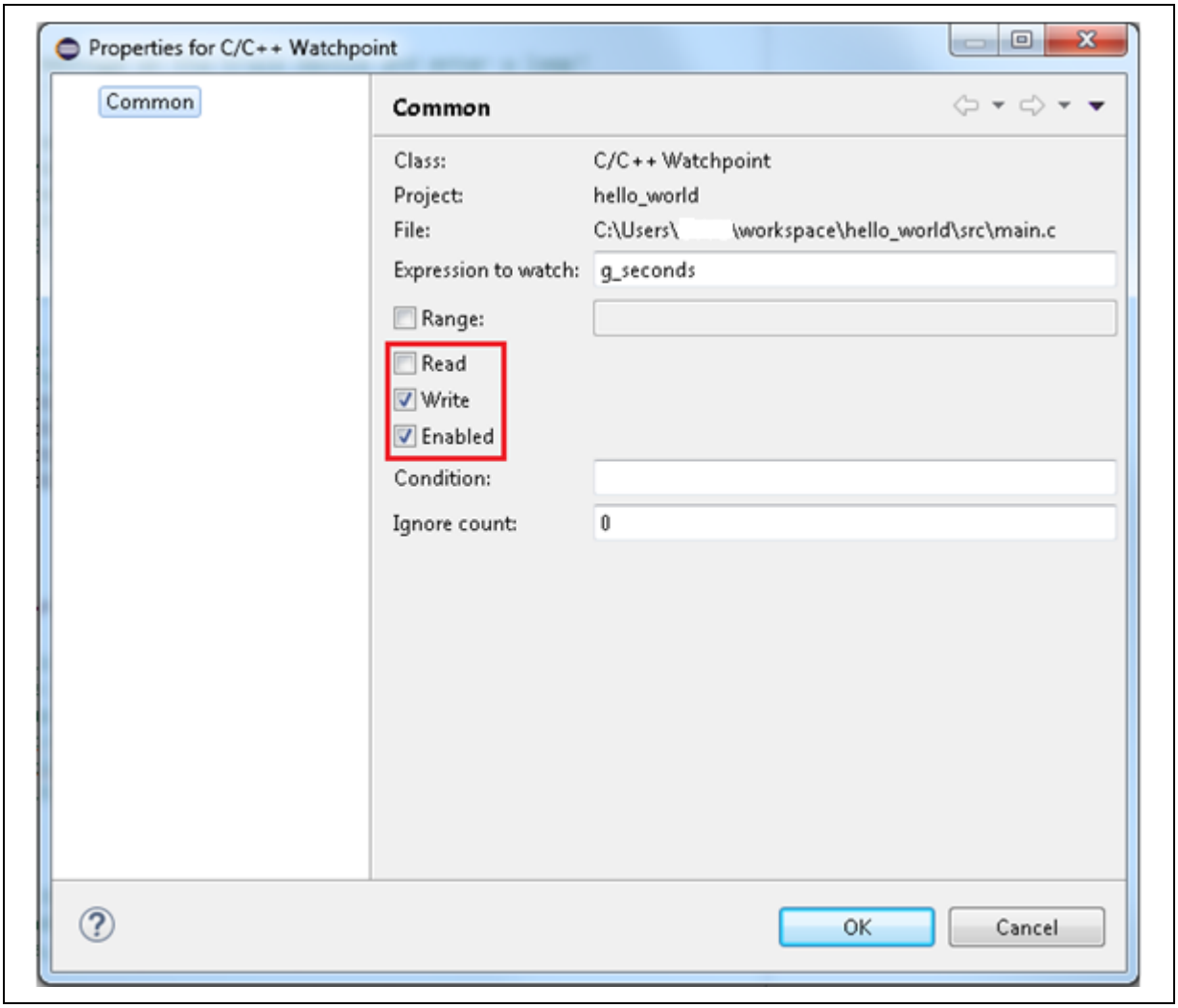


Figure 3-20 Properties for C/C++ Watchpoint

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

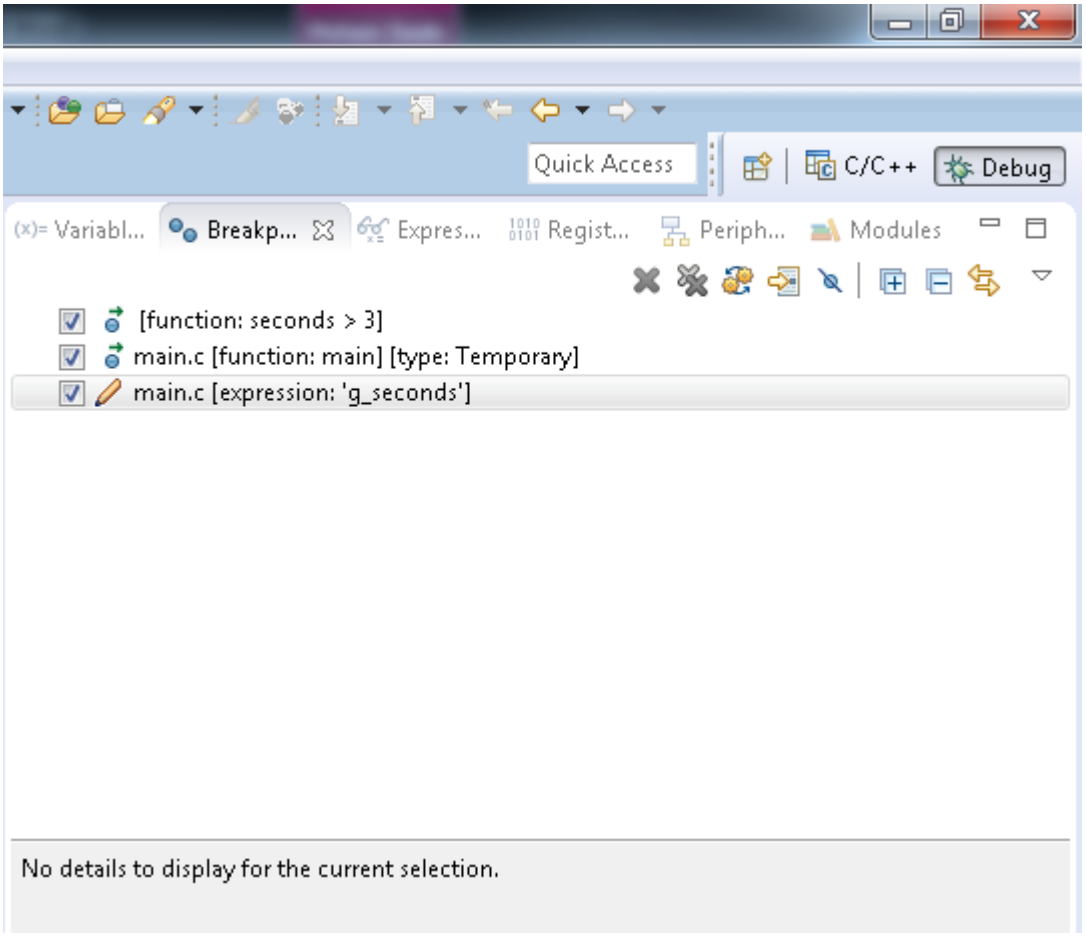


Figure 3-21 Added Watchpoint in the Breakpoints View

3.7 **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:

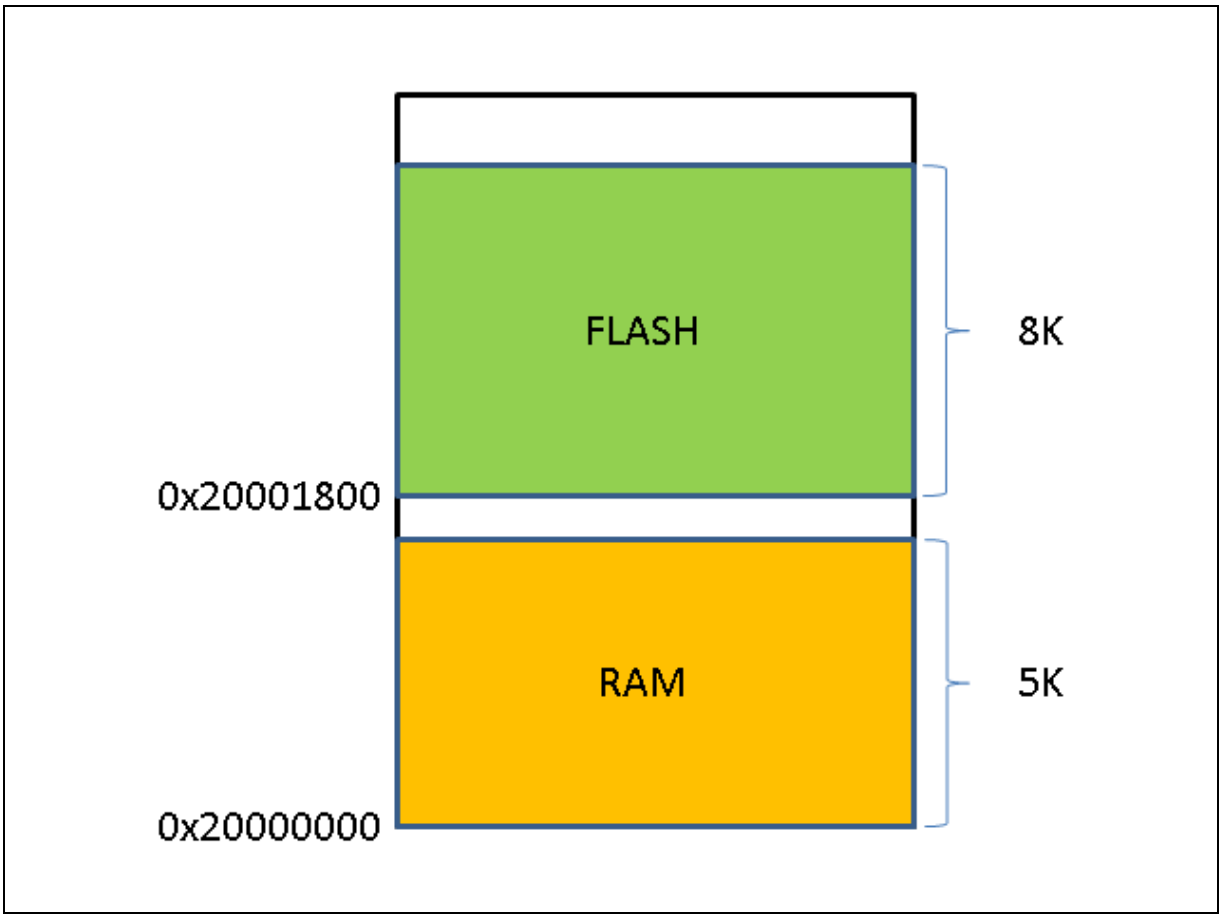


Figure 3-22 Memory Layout

The modified mem.ld script should meet the memory layout design.

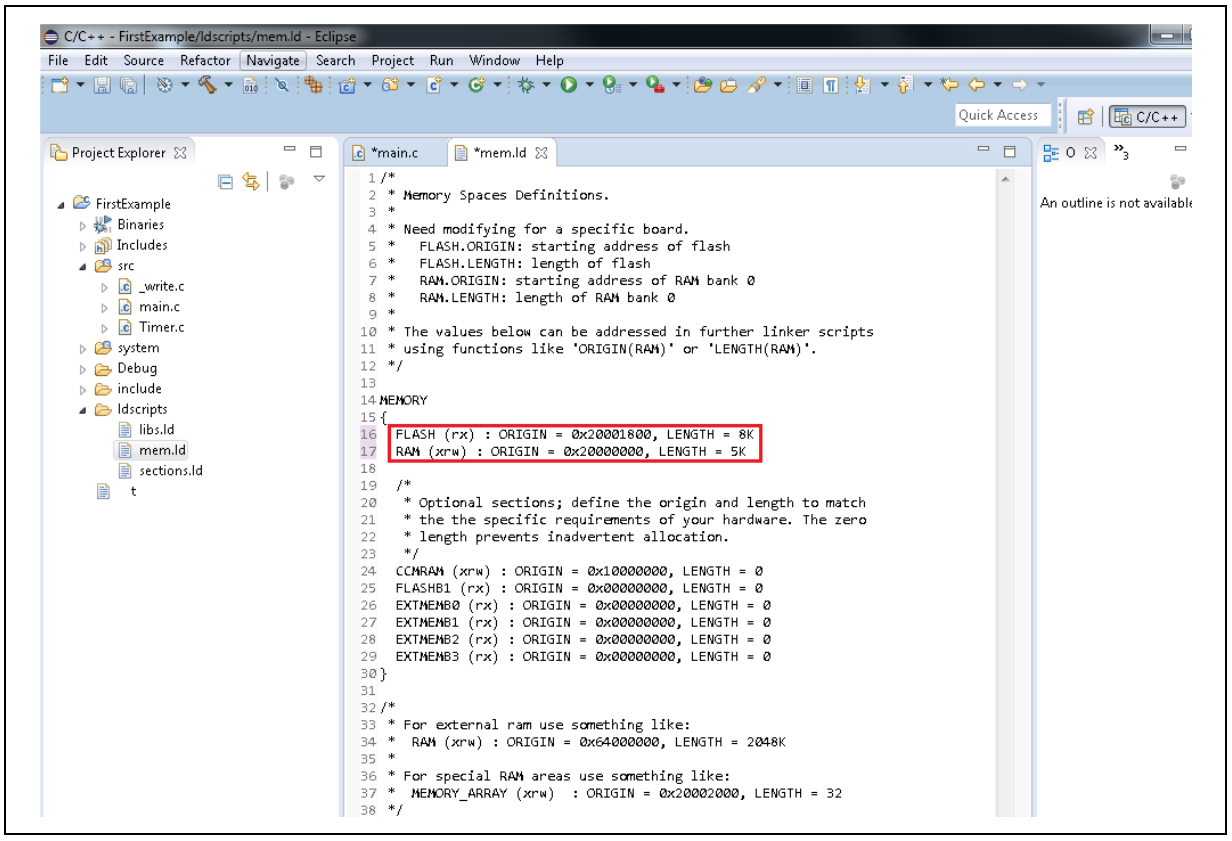


Figure 3-23 Modifying the Mem.ld Script

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 0x20001800 and 0x20001400, respectively. 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.

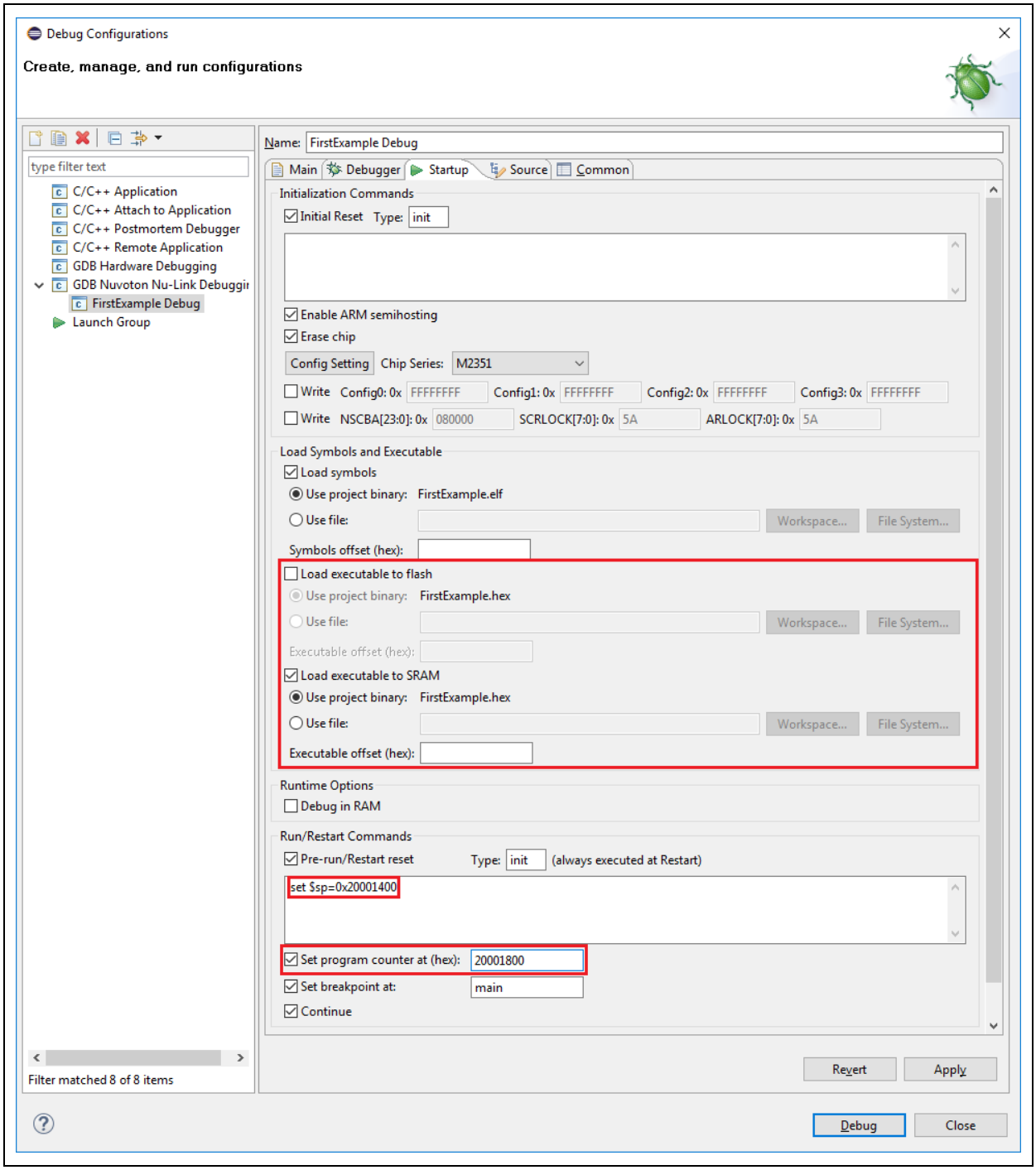


Figure 3-24 Debug Configuration Settings

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.

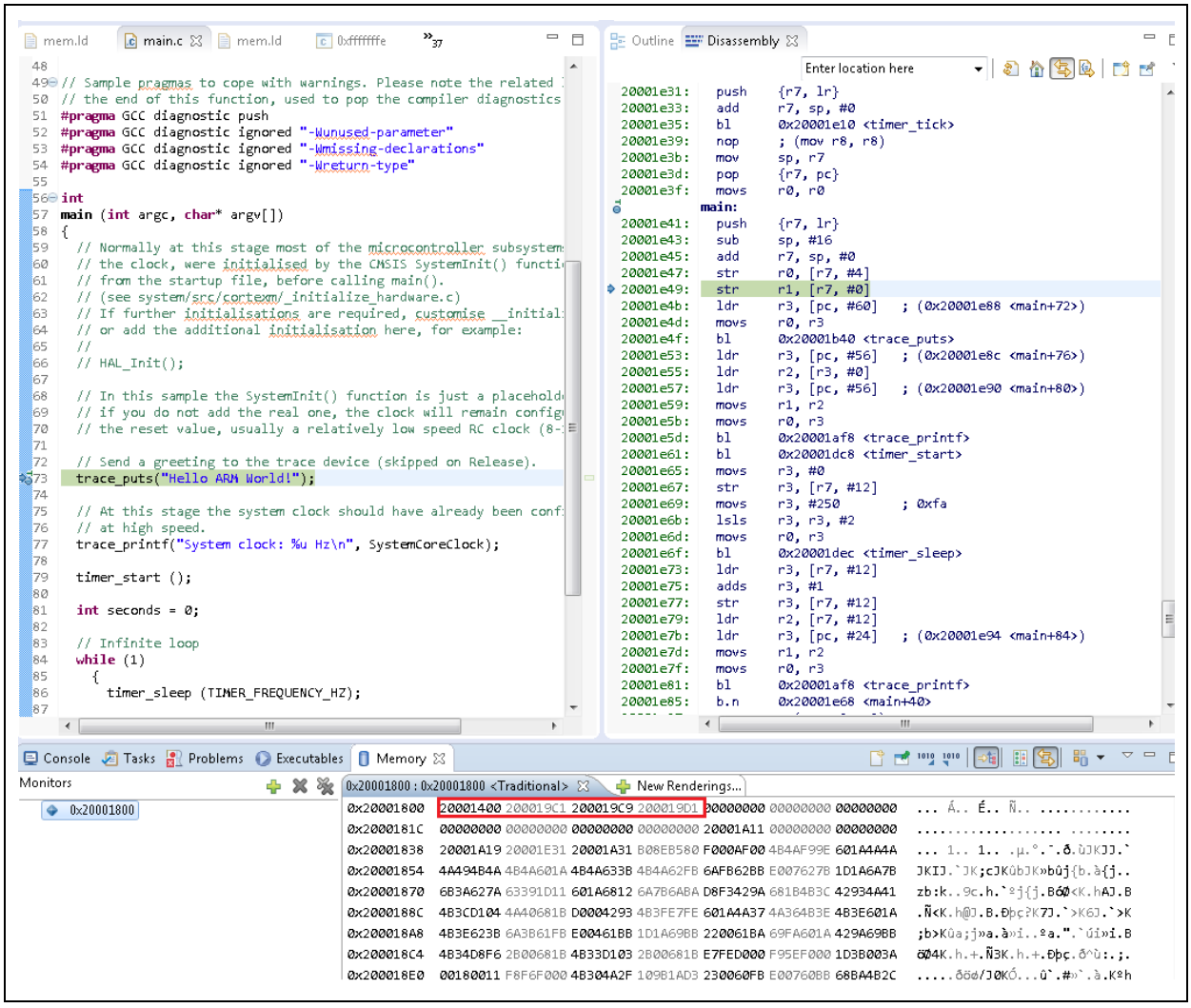


Figure 3-25 Debugging in RAM

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 mem.ld file in the ldscripts 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 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. The path of projects should not contain any white space or special characters.

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

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

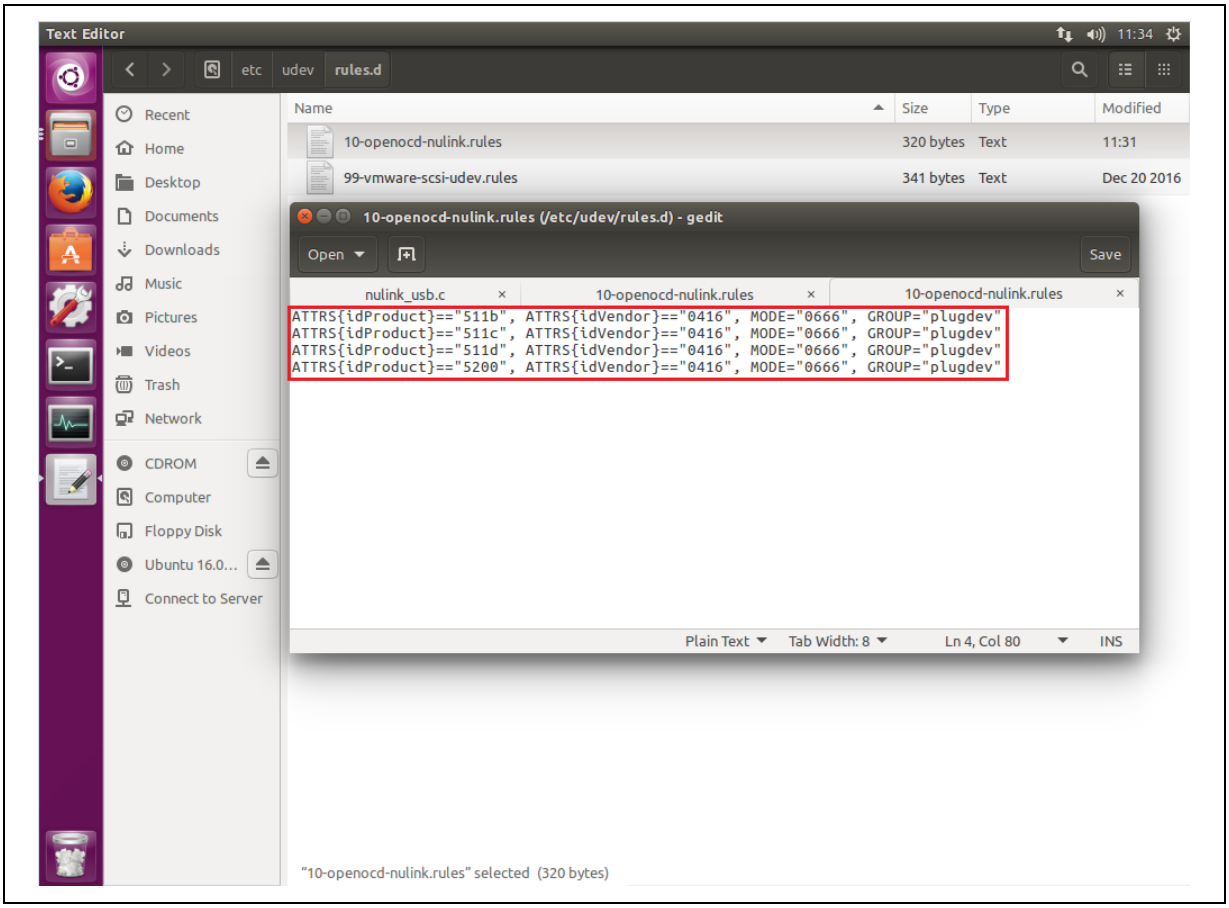


Figure 4-1 Adding Udev Rules

   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 GNU ARM Eclipse, 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.

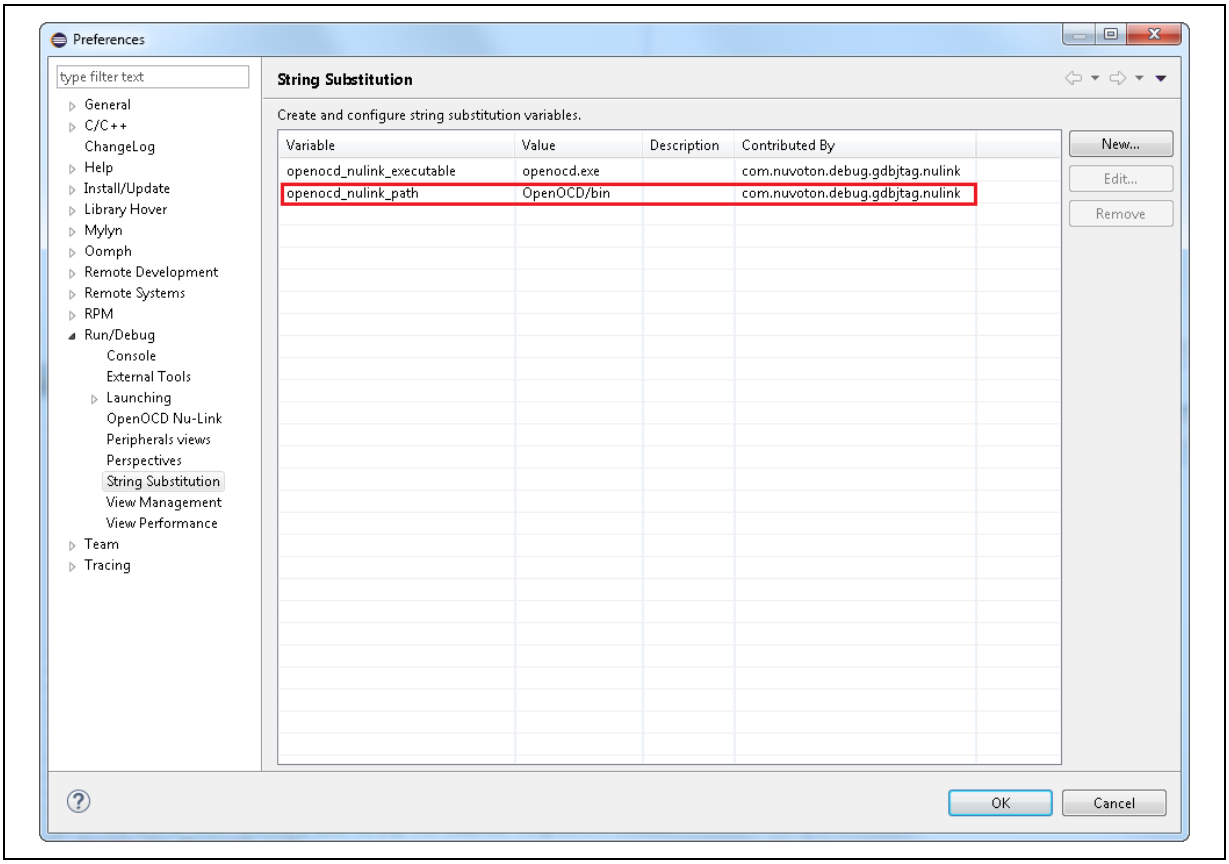


Figure 4-2 Preferences for String Substitution

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 the 64-bit Windows, the location where we place software packs is C:\Program Files (x86)\Nuvoton Tools\Packages. Similarly, On the 32-bit Windows, the location where we place software packs is C:\Program Files\Nuvoton Tools\Packages.

5 Revision History

Date	Revision	Description
2017.03.31	0.01	Alpha version released.
2017.06.30	1.01	Beta version released.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.