N9H20 Eclipse User Manual

Document Information

abstract	N9H20 family processors provide the Keil and Eclipse environment to speed up software development. This document instructs the user how to setup and create Eclipse application program project smoothly.
Apply to	N9H20 series

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

www.nuvoton.com

Table of Contents

1 Eclipse Development Environment

By J-TAG as ICE debug interface, N9H20 provides Keil IDE and Eclipse as Non-OS BSP development environment. This document is only focused on how to use Eclipse and let the user uses this interface to download application program to DRAM for debugging.

1.1 Eclipse Installation

This section introduces the installation steps of Eclipse develop environment. First download Eclipse IDE for C/C++ Developers Tool from Eclipse official website *http://www.eclipse.org/downloads/* and select proper version according to your operating system. Since Eclipse is a Java based application, please also download JRE from Java website and install it.

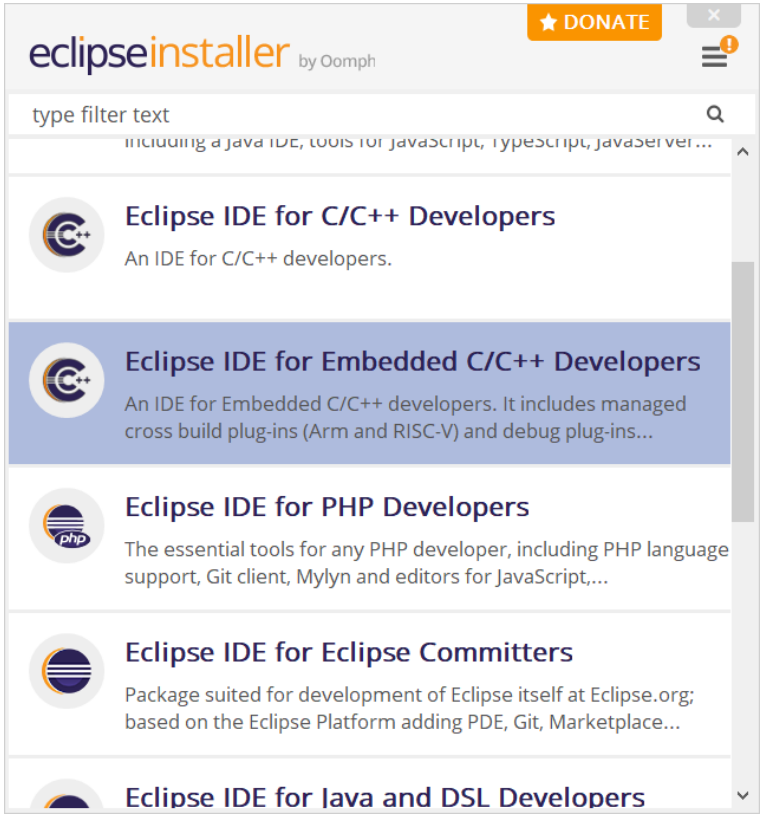

Figure 1-1-1 Select Eclipse IDE

After installing the software packages memtioned above, please execute Eclipse and select **Help ➔ Eclispe Marketplace** as shown in Figure 1-2.

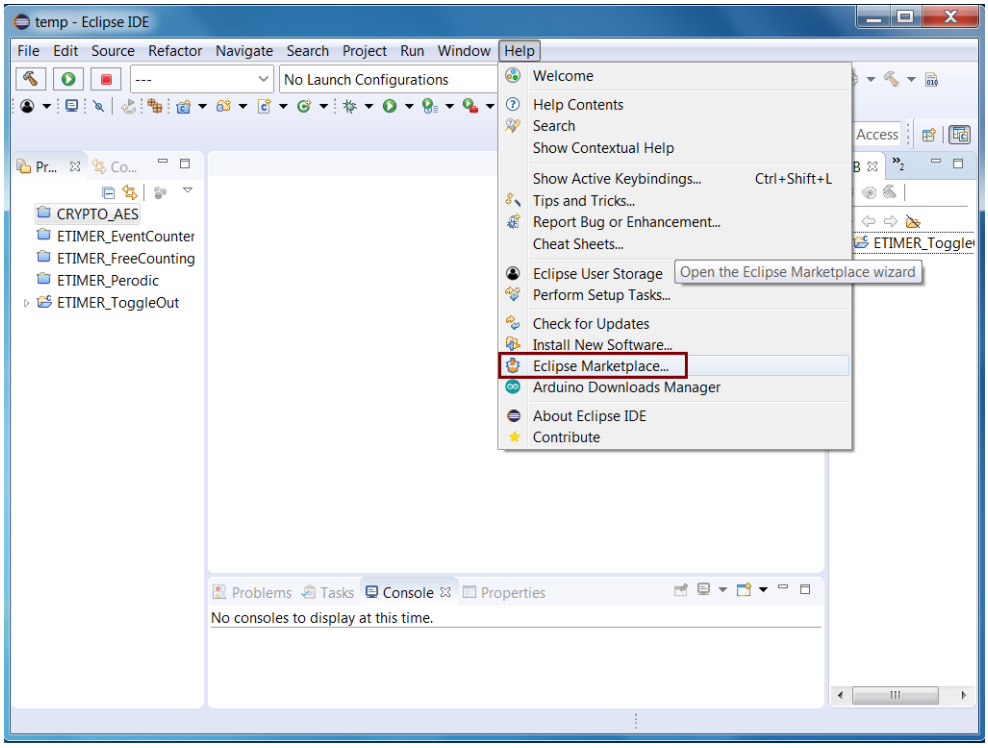

Figure 1-2 Select Eclipse Marketplace

Input gnu mcu in **Find** field, and then the search result will be shown as Figure 1-3. Select latest version and click **Install** button to install the required plug-in. In recent Eclipse version, the related plug-in can be installed at the same time while Eclipse IDE is installed

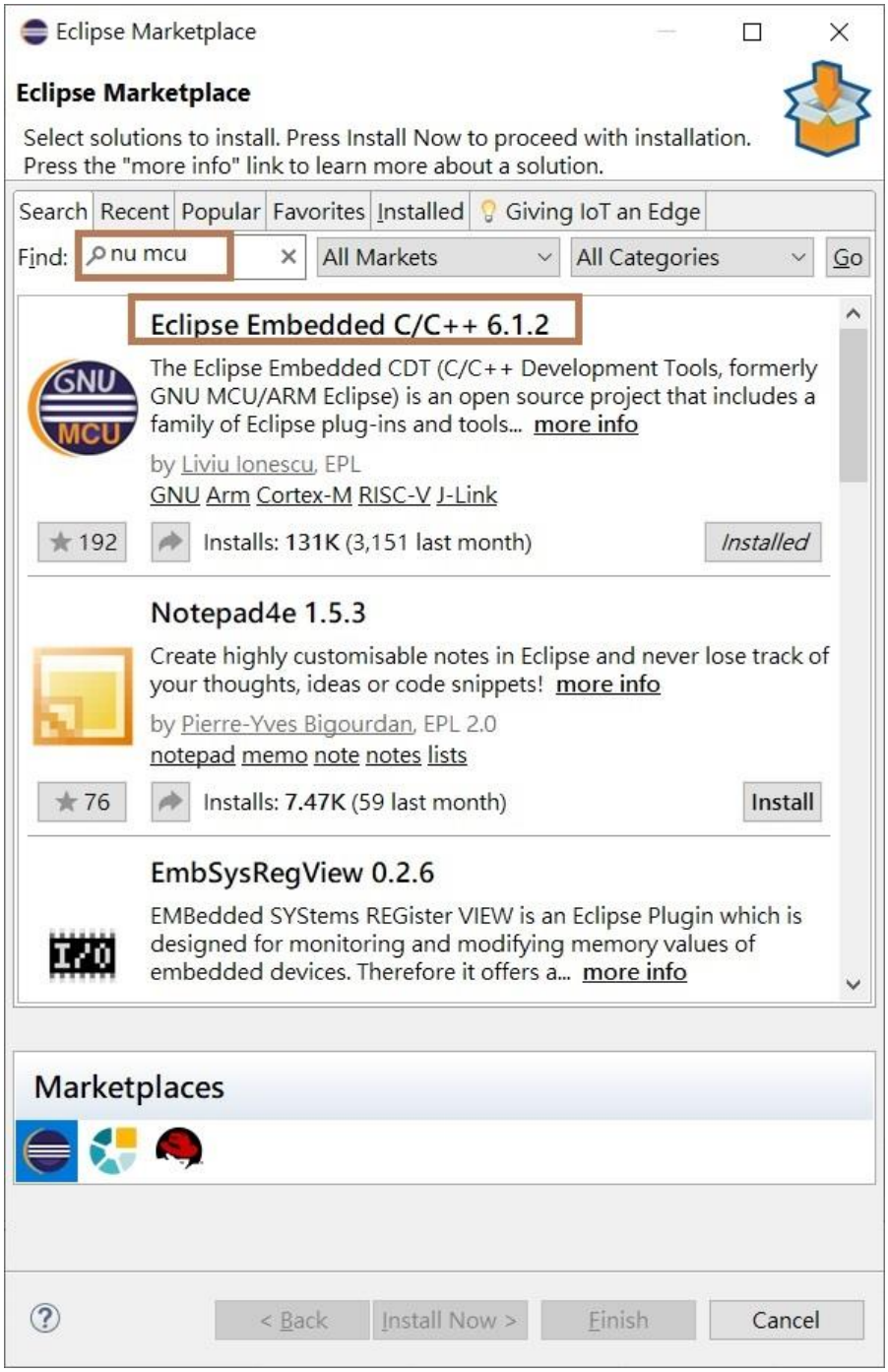

Figure 1-3 Install Plug-in

Click **Help ➔ Install New Software** to install CDT to support C/C++ development (Figure 1-4).

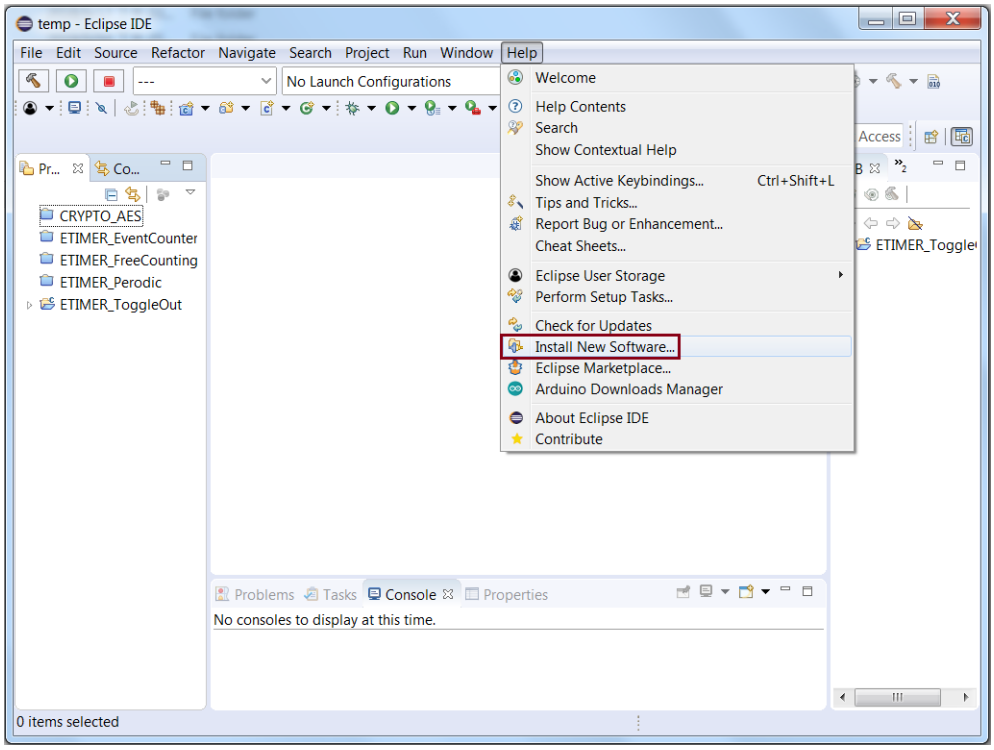

Figure 1-4 Install New Software

Input CDT in **Work with** field as Figure 1-5.

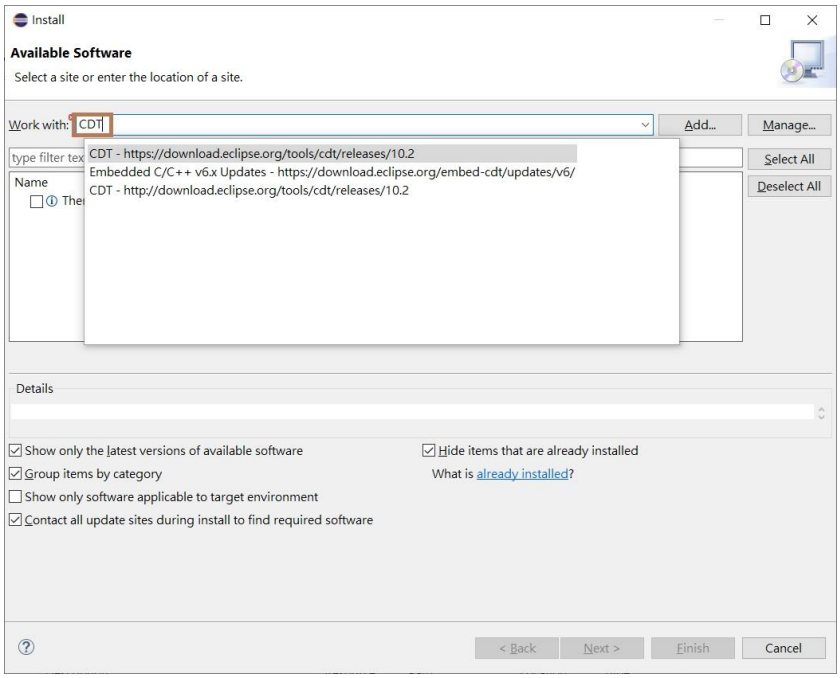

Figure 1-5 Search for CDT

Select **CDT Main Features** and **CDT Optional Features** as shown in Figure 1-6. The user can also select

other packages if necessary.

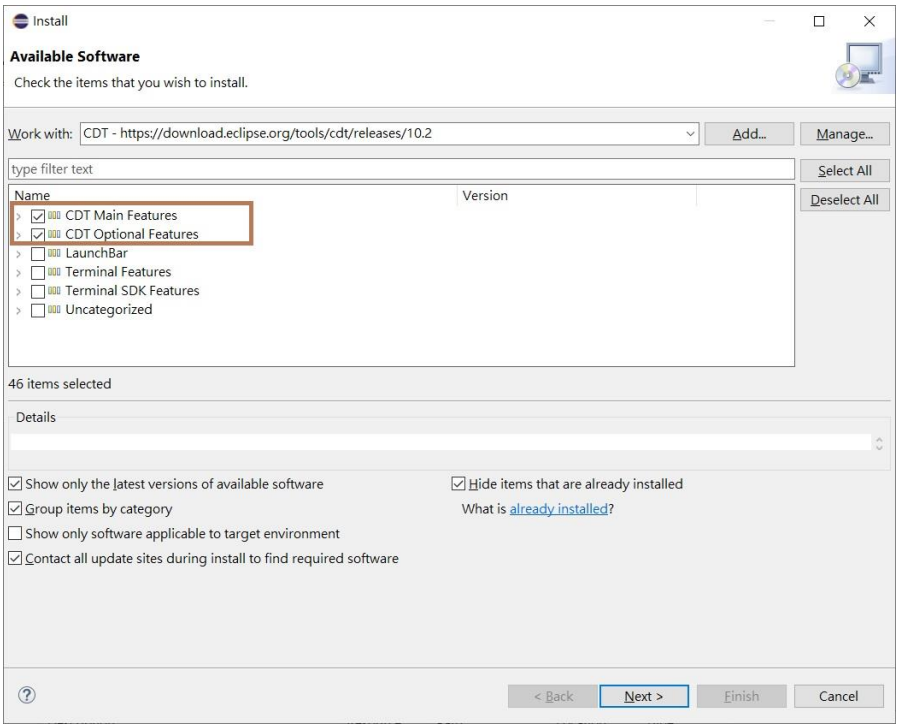

Figure 1-6 Select CDT

1.2 GNU ARM Embedded GCC Toolchain and Build Tools Installation

The cross compiler GNU ARM Embedded Toolchain can be downloaded manually from *https://developer.arm.com/downloads/-/gnu-rm*. As for the Build tools, it can be be downloaded from *http://github.com/gnu-mcu-eclipse/windows-build-tools/releases*. Both the toolchain and Build tools are packed ones from the above websites. To simplify the tool management, these two files can be unpacked to the same folder. For example, the selected toolchain gcc-arm-none-eabi-10.3-2021.10-win32.zip and Build tools xpack-windows-build-tools-4.3.0-1-win32-x64.zip are unpacked individually to the same folder C:\Eclipse. To use the toolchain and Build tools, the related paths can set to the unpacked folder and will be described in later section.

1.3 Start Eclipse

After installing Eclipse, please restart Eclipse and select a directory as the workplace (Figure 1-7).

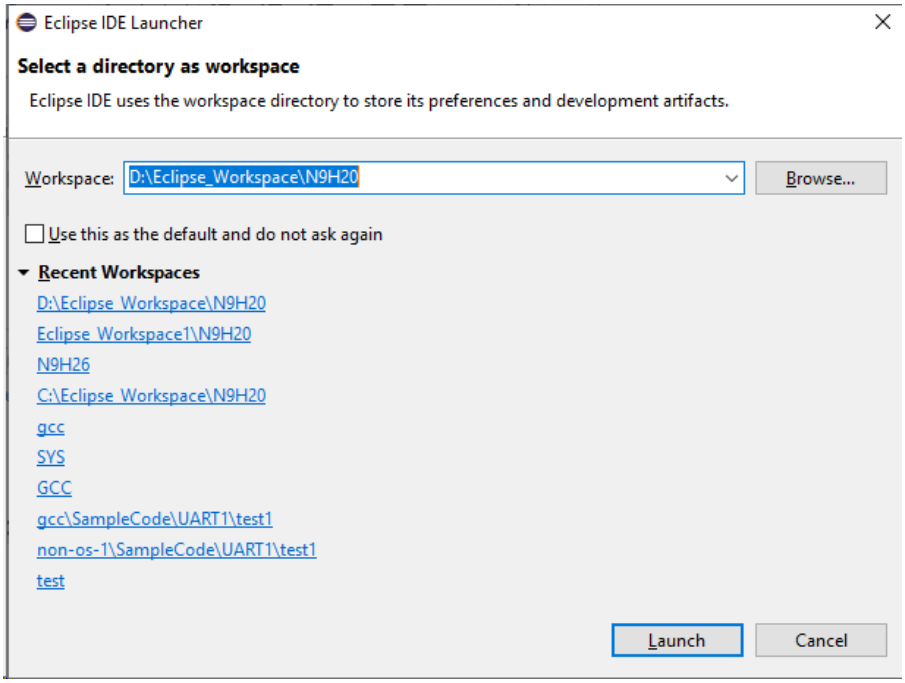

Figure 1-7 Select Directory as Workspace

As shown in Figure 1-8, whenever the new workspac has been opened, we can start Eclipse IDE to create the new project or import the existing project into workspace. The detail procedure will be inroduced in later sections.

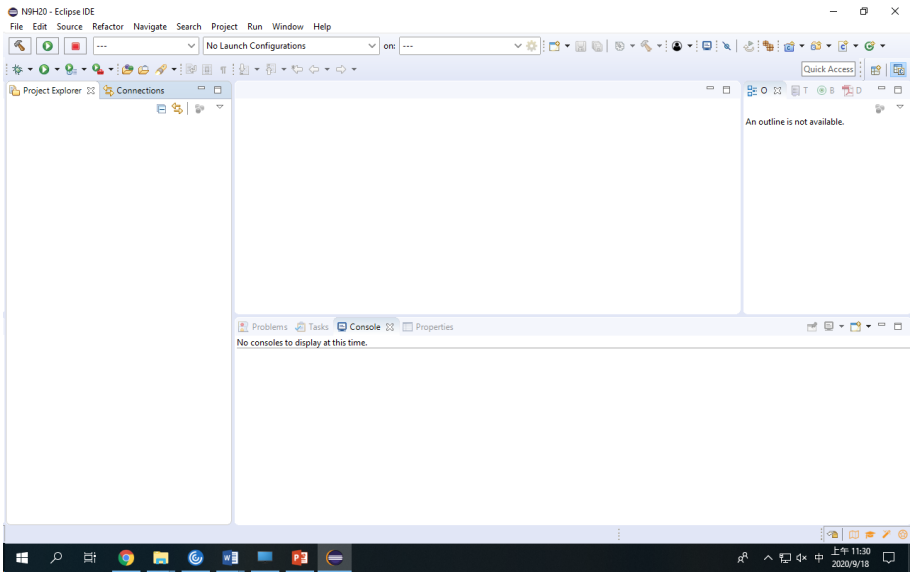

Figure 1-8 Opened Workspace

2 Eclipse Project

This chapter inroduces how to create a new Eclipse software project or import the existing one to the workspace.

2.1 Create New Eclipse Project

To create a new project (Figure 2-1), please select a **New C/C++ Project** and **C Managed Build** as shown in Figure 2-2.

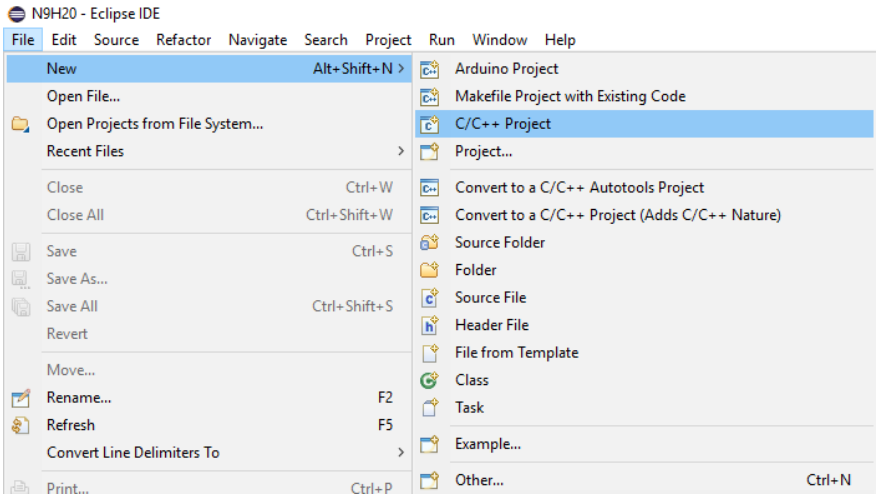

Figure 2-1 Create New Project

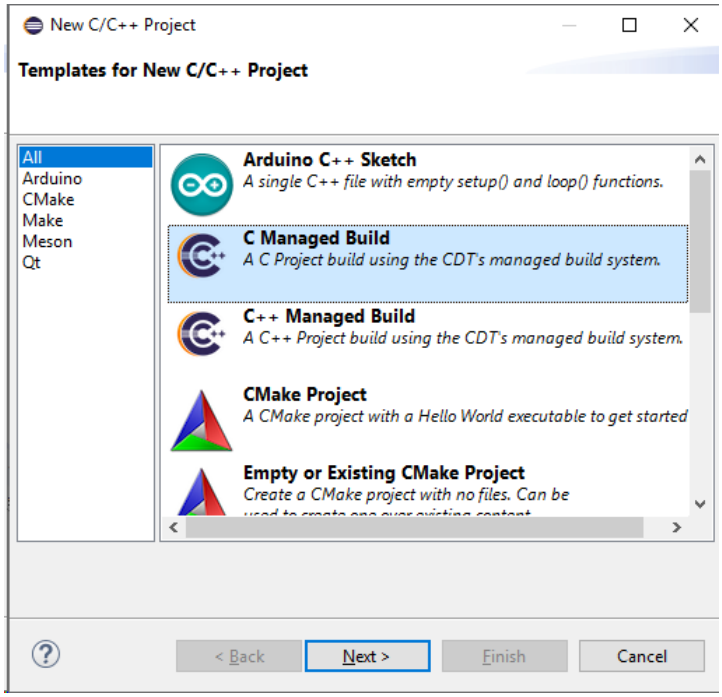

Figure 2-2 Select C Managed Build

Give the project name and **Select Configurations** as shown in Figure 2-3 and Figure 2-4, respectively.

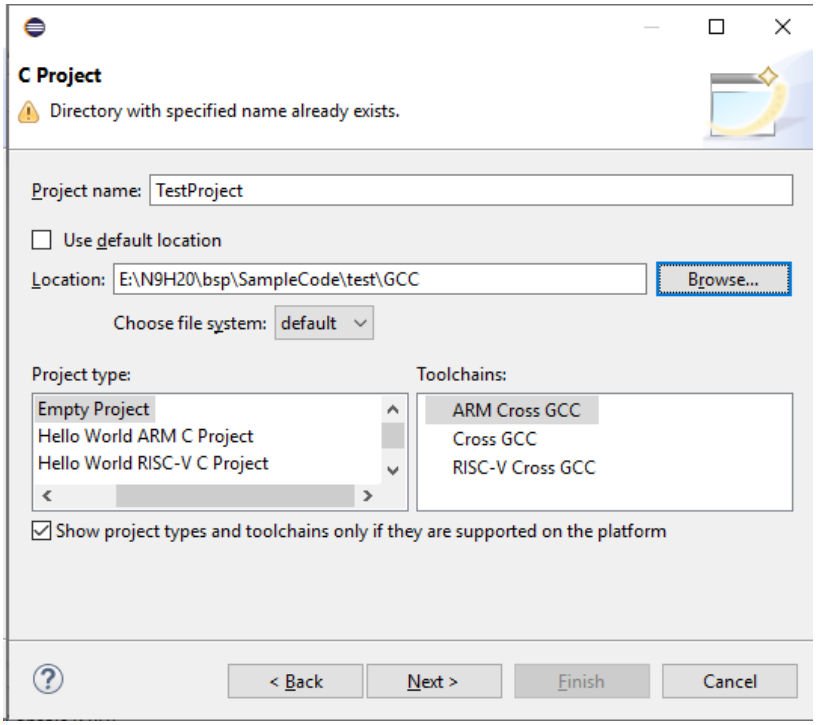

Figure 2-3 Set Project New Name

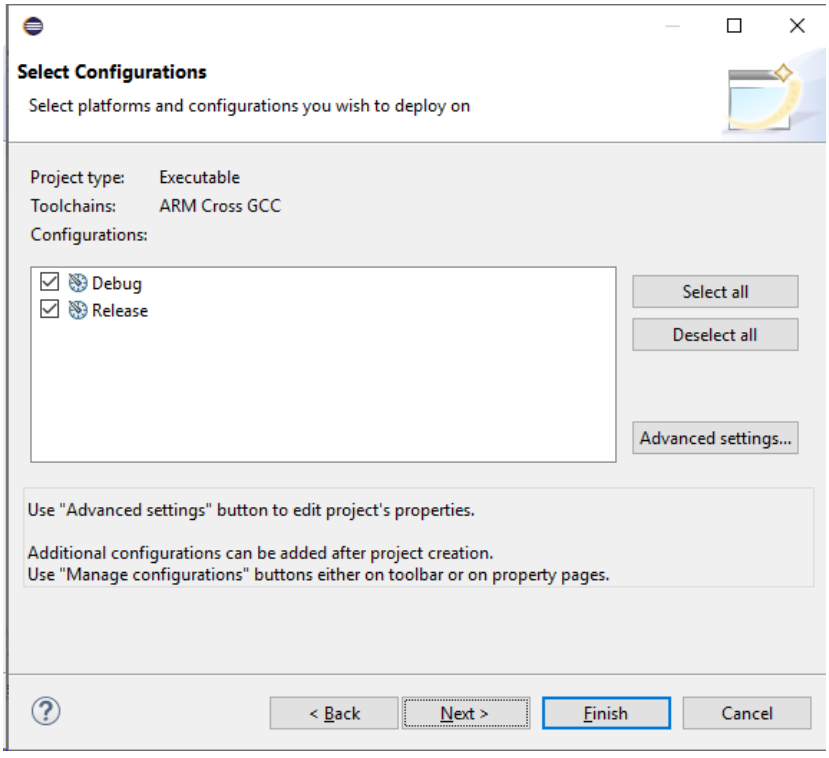

Figure 2-4 Select Configurations

Select the toolchain and configure path as shown in Figure 2-5.

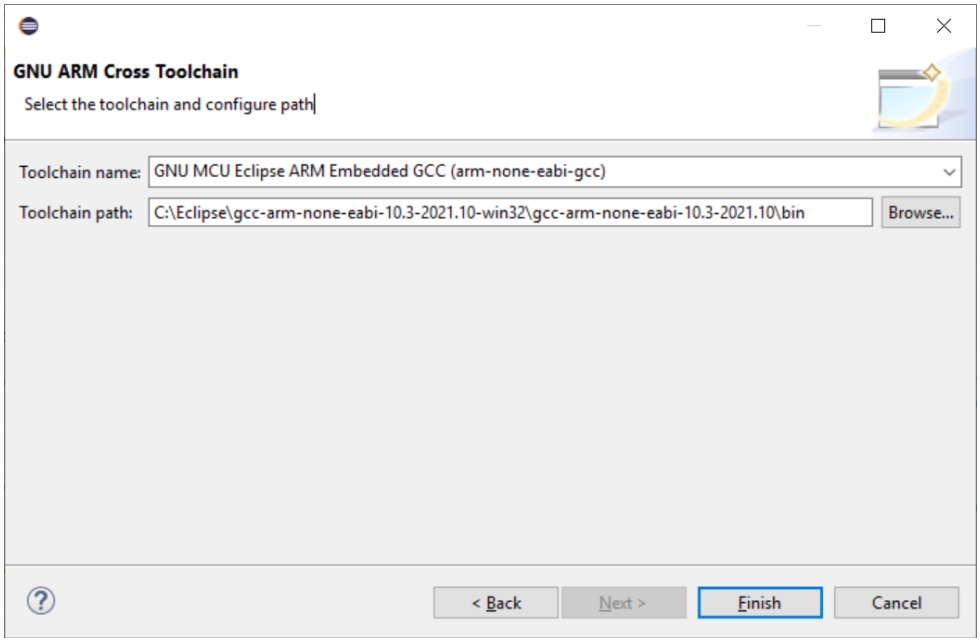

Figure 2-5 Toolchain Setting

If the created project is a library one, as shown in Figure 2-6, please select **Static Library** in Properties ➔ **Settinng** ➔ **Build Artifact** ➔ **Artifact Type** options.

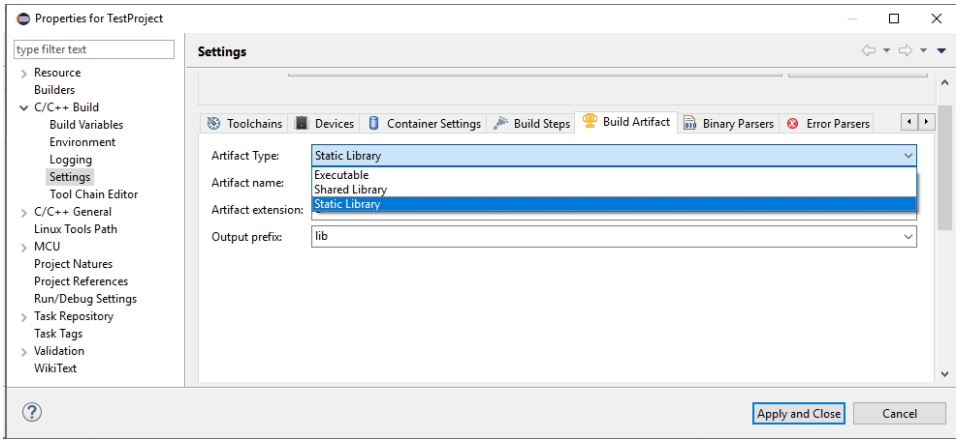

Figure 2-6 Static Library Selection

As shown in Figure 2-7, if the created project is an executable one, please select **Executable** in **Artifact Type** options.

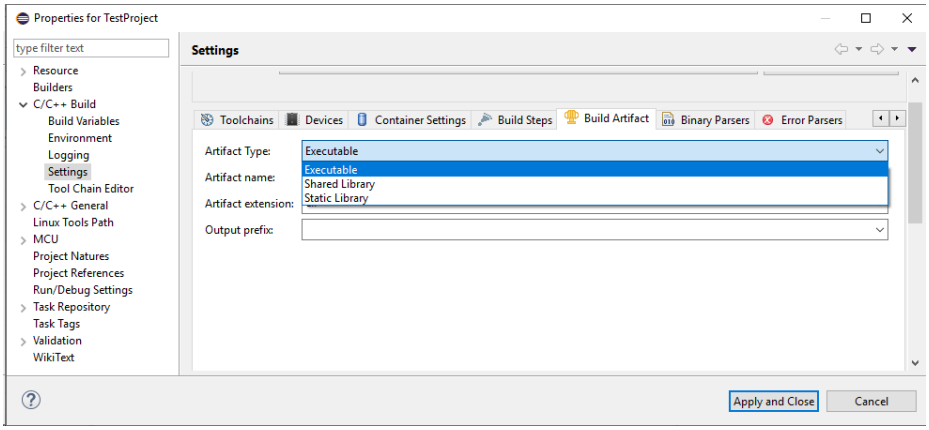

Figure 2-7 Executable Output Selection

Thus, the basic configuration for an empty new project has been setup. Next, other related steps to a new project are lsted below.

1. Select the Build tools and configure path as in Figure 2-8.

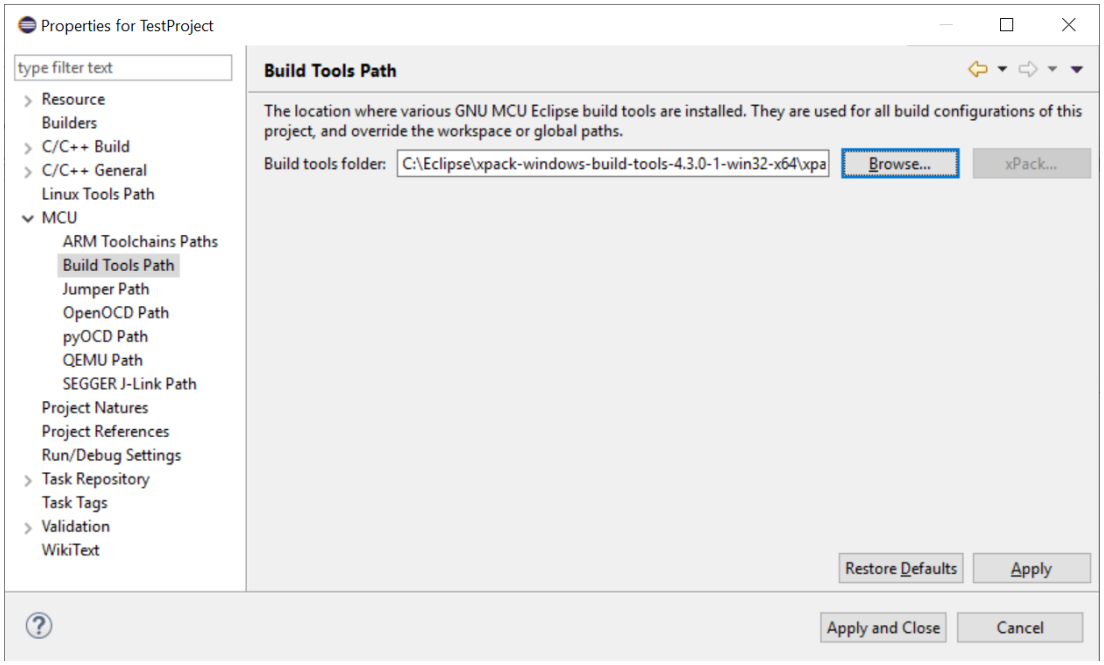

Figure 2-8 Build Tool Setting

If the Build tools path is set successfully, it also can be found in Figure 2-9.

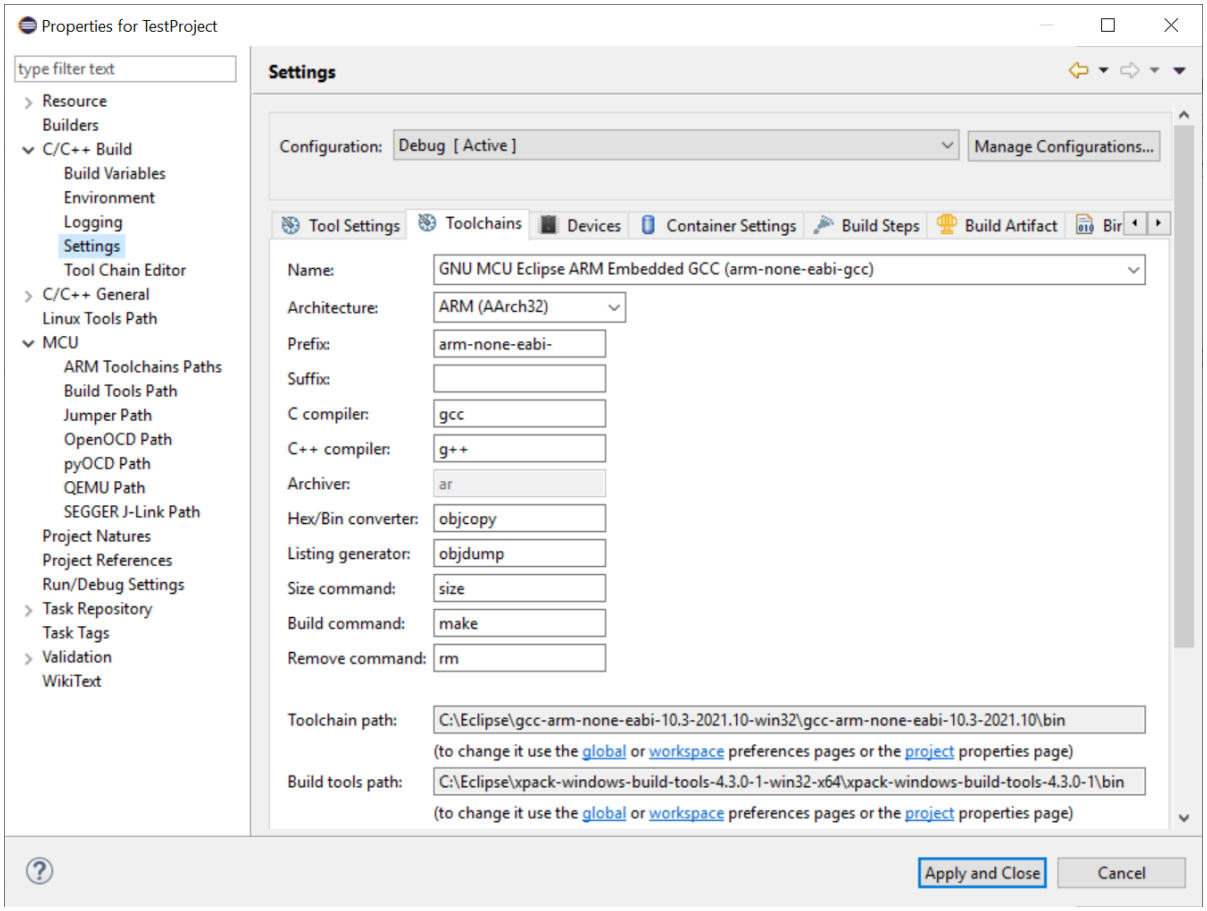

Figure 2-9 Toolchain and Build Tools Path Settings

2. Select **ARM family arm926ej-s** in **Properties ➔ Settinng ➔ Tool Settings ➔ Target Processor** options as shown in Figure 2-10.

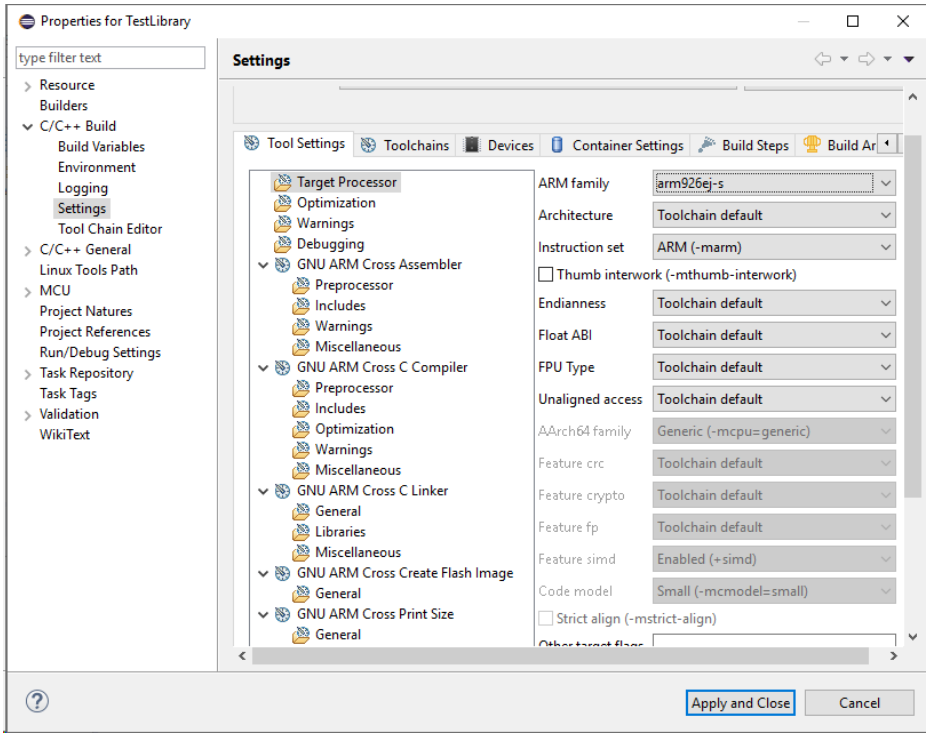

Figure 2-10 Target Processor Selection

3. Figure 2-11 shows how to set the Include path in **GNU ARM Cross C** and **Assembler Compiler** if necessary.

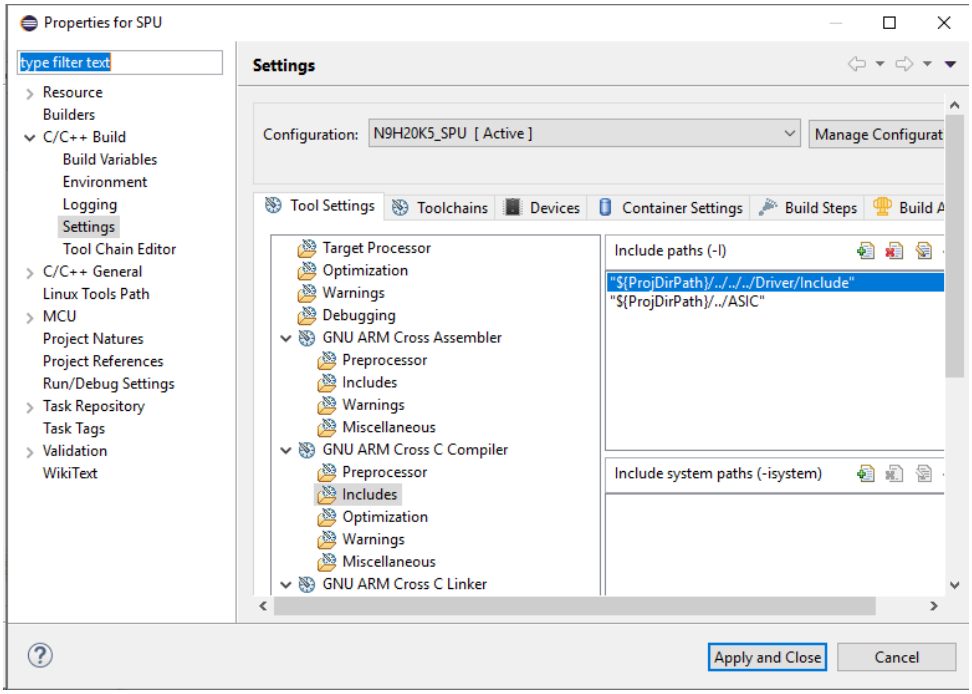

Figure 2-11 Include Path Settings

4. Defines the symbol for diferent devices (N9H20K1/N9H20K3/N9H20K5) in **Assembler/C Compiler Preprocessor** as sohwn in Figure 2-12 and Figure 2-13, respectively

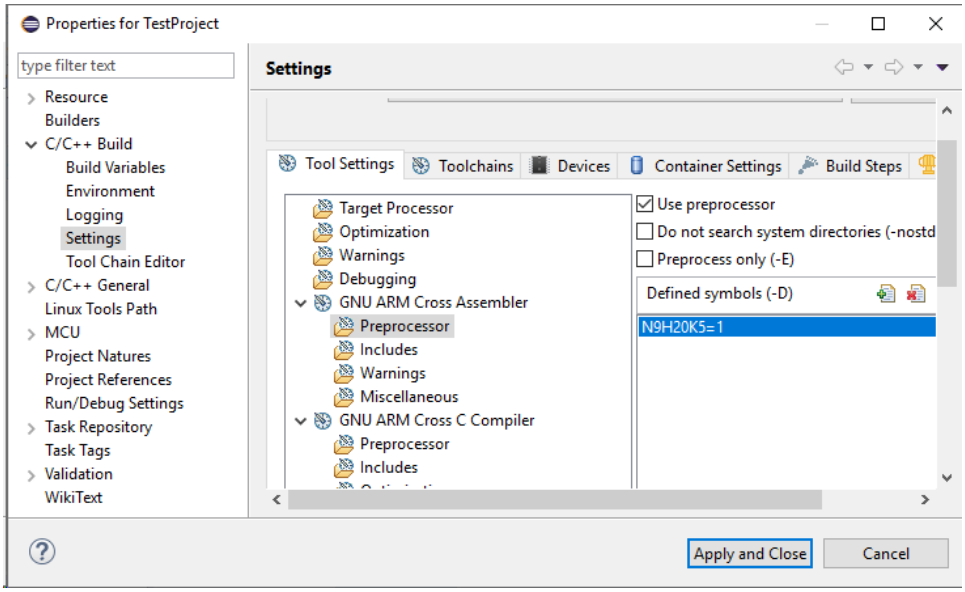

Figure 2-12 Symbol Definition in Assembler

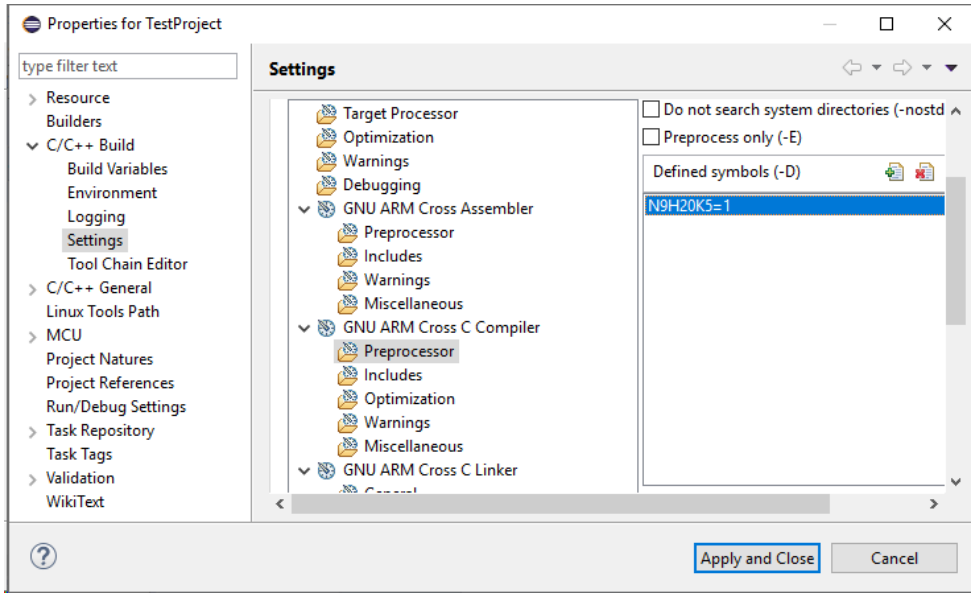

Figure 2-13 Symbol Definition in C Compiler

5. Setup a new folder and given folder name “Driver” in Virtual Floder as shown in Figure 2-14 and Figure 2-15, respectively.

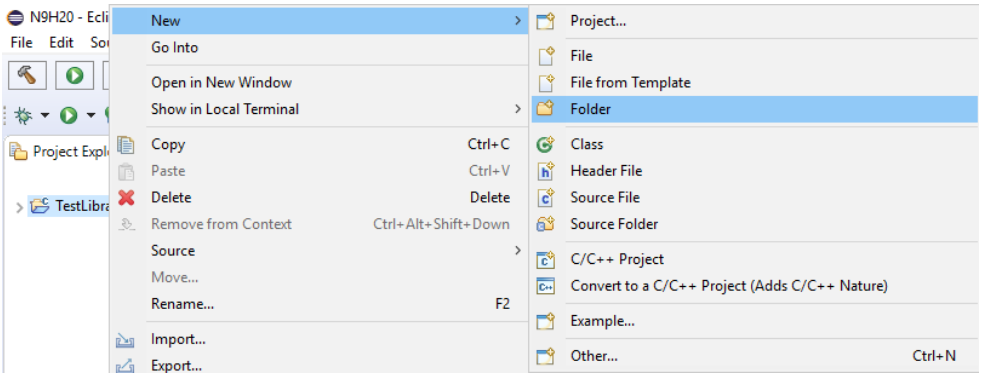

Figure 2-14 Setup New folder

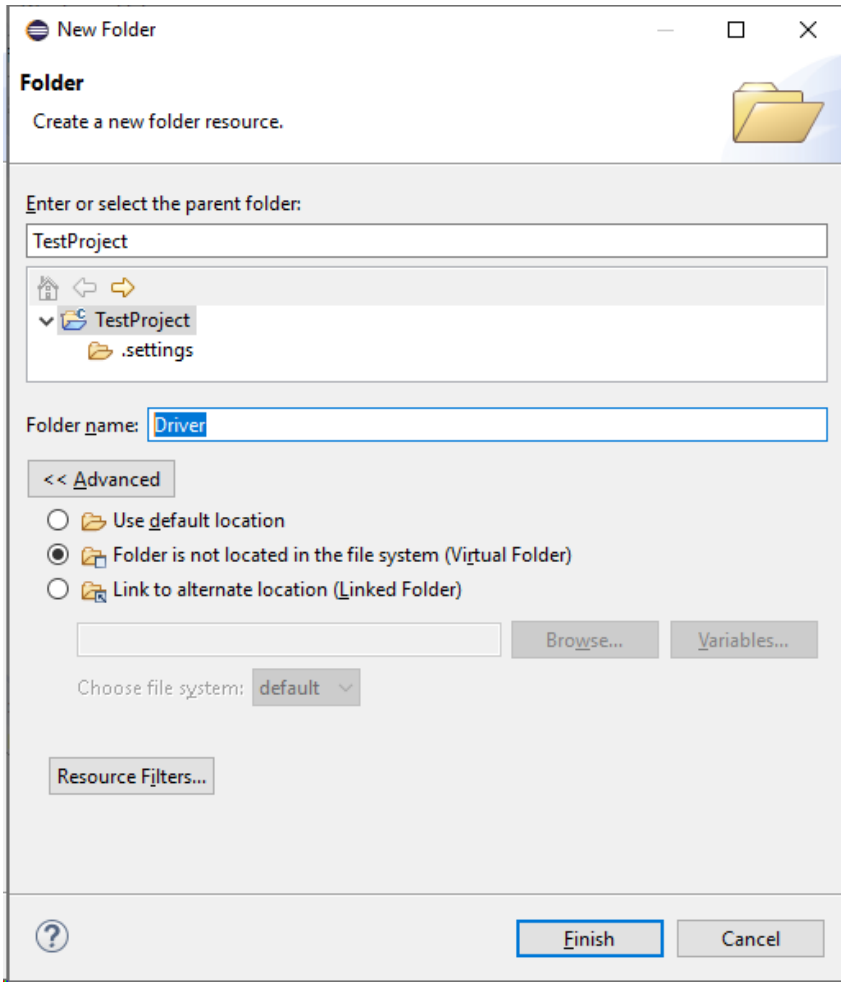

Figure 2-15 Setup New Folder Name

6. If the created project is a executable one, please also add startup code into “Driver” virtual floder as shown in Figure 2-16 and Figure 2-17, respectively.

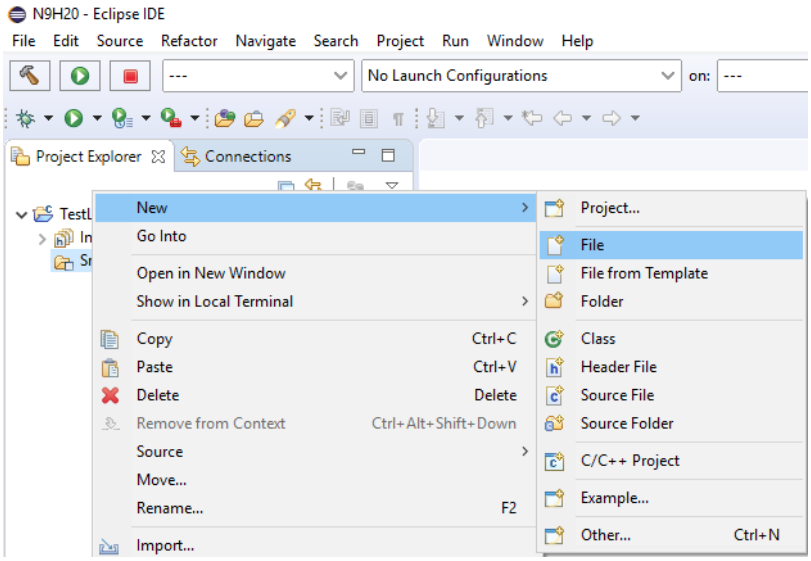

Figure 2-16 Setup New File

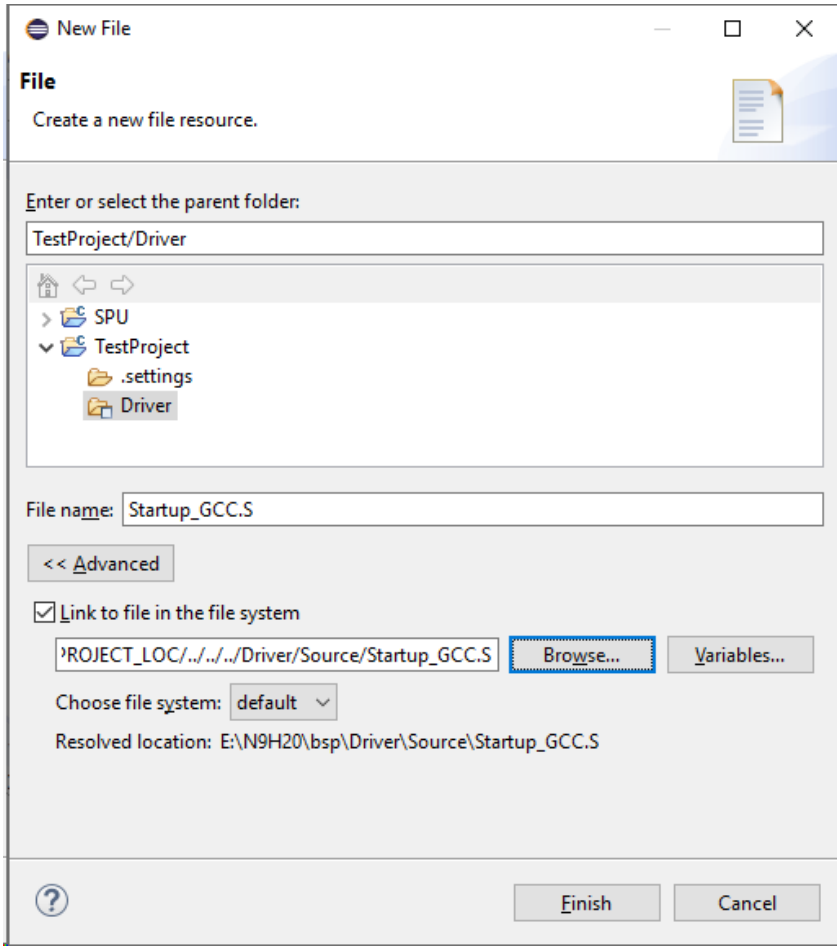

Figure 2-17 Add Startup Initial Code in Folder

7. Setup another virtual folder name “Src” and then add necessary files as shown in Figure 2-18.

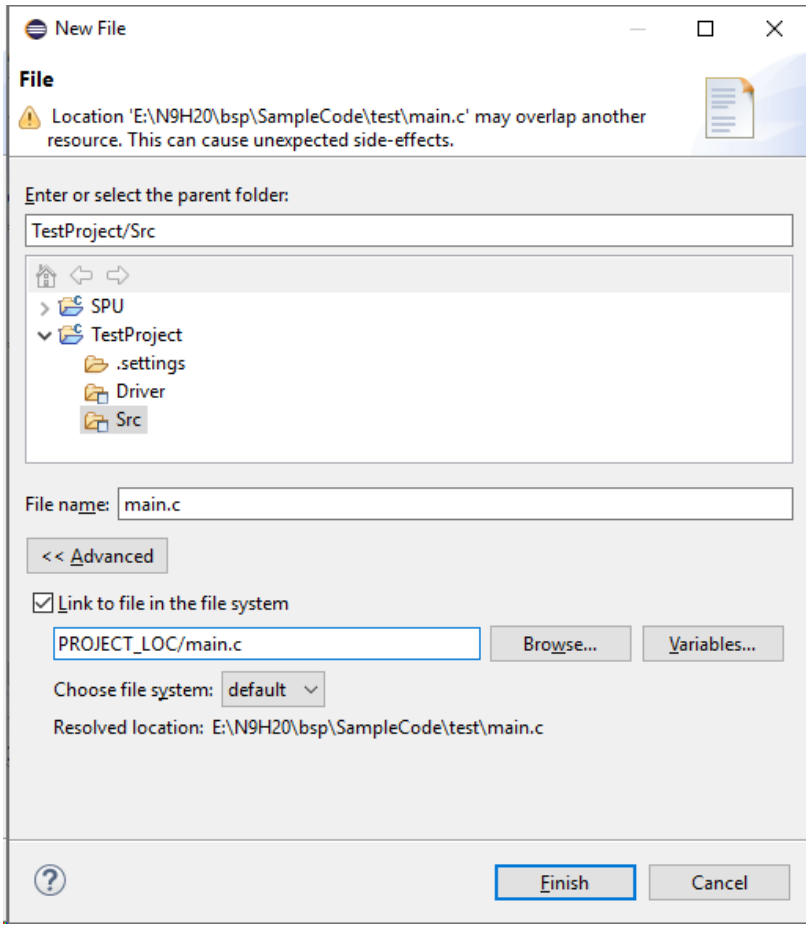

Figure 2-18 Add Necessary File in Src Folder

8. Add configuration name (Figure 2-20) in **Build Configurations ➔ Manage** window (Figure 2-19), and we can create new configuration.

Figure 2-19 Build Configuration

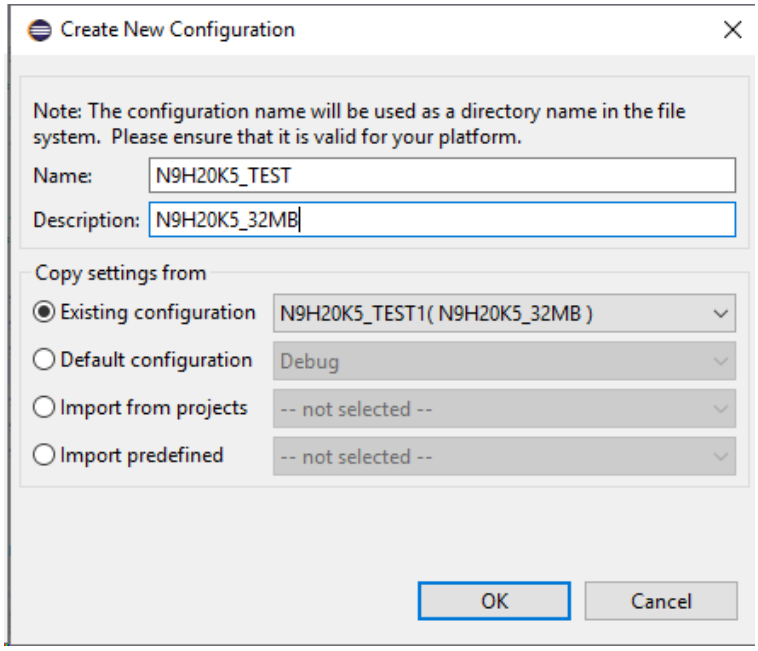

Figure 2-20 Create New Configuration

9. If the created project is a executable one, please select the linker Script file in **GNU ARM Cross C Linker**. Before to select Script file, some statements need to add in **Properties ➔ Settinng ➔ Build Steps ➔ Pre-bui**d Steps Command window as shown in Figure 2-21.

   - ${cross_prefix}cpp -E -D${ConfigDescription}=1 -P
   - ${ProjDirPath}/../../../Driver/Source/gcc_arm_SRAM.ld -o
   - ${ProjDirPath}/${ConfigName}/gcc_arm_SRAM_${ConfigDescription}.ld

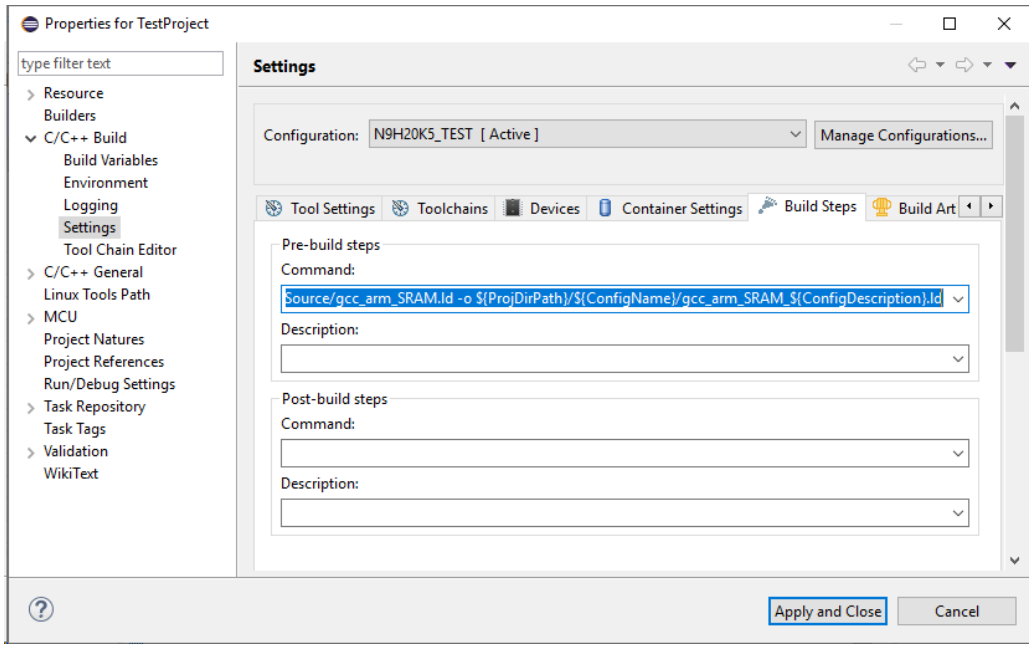

Figure 2-21 Setup Pre-build Command

Then, add the Script file and file path in **Properties ➔ Settinng ➔ GNU ARM Cross C Linker ➔ General** in Figure 2-22 and Figure 2-23, respectively.

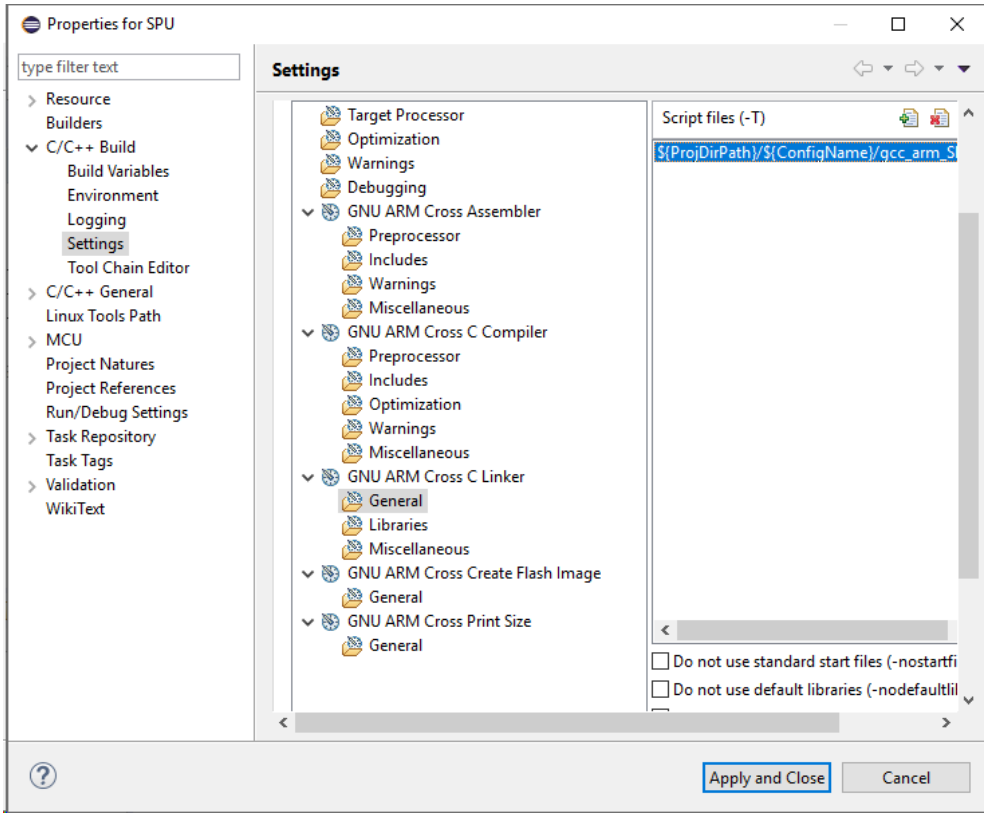

Figure 2-22 Setup Linker Script File

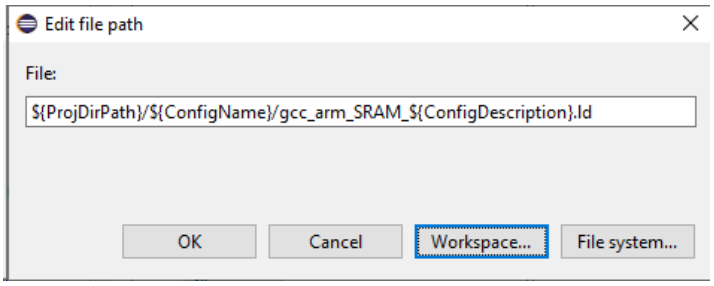

Figure 2-23 Setup Linker File Path

10. Link necessary Libraries and setup library search path as sohwn in Figure 2-24 and Figure 2-25, respectively.

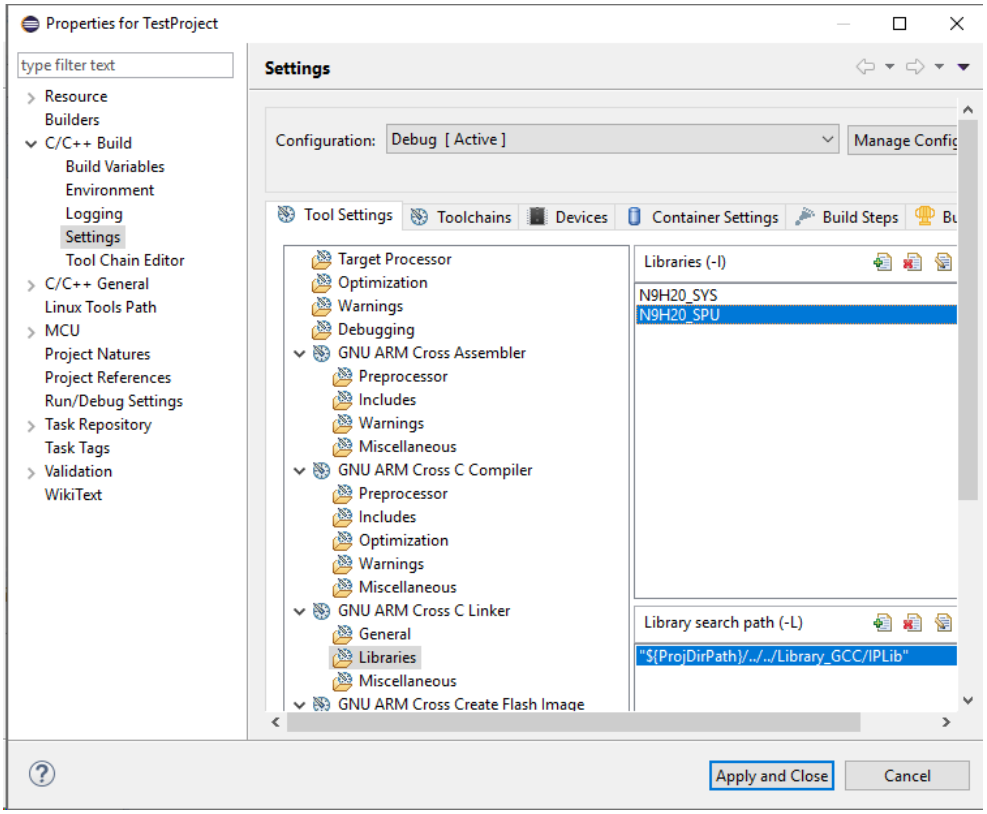

Figure 2-24 Setup Linked Library

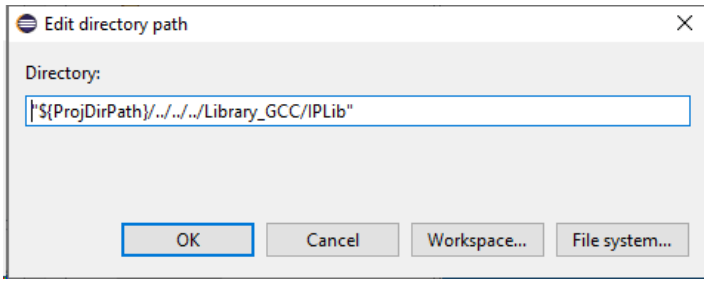

Figure 2-25 Setup Linked Library Path

11. In addition, enable newlib-nano in **Properties ➔ Settinng ➔ GNU ARM Cross C Linker ➔ Miscellaneous** options as shown in Figure 2-26.

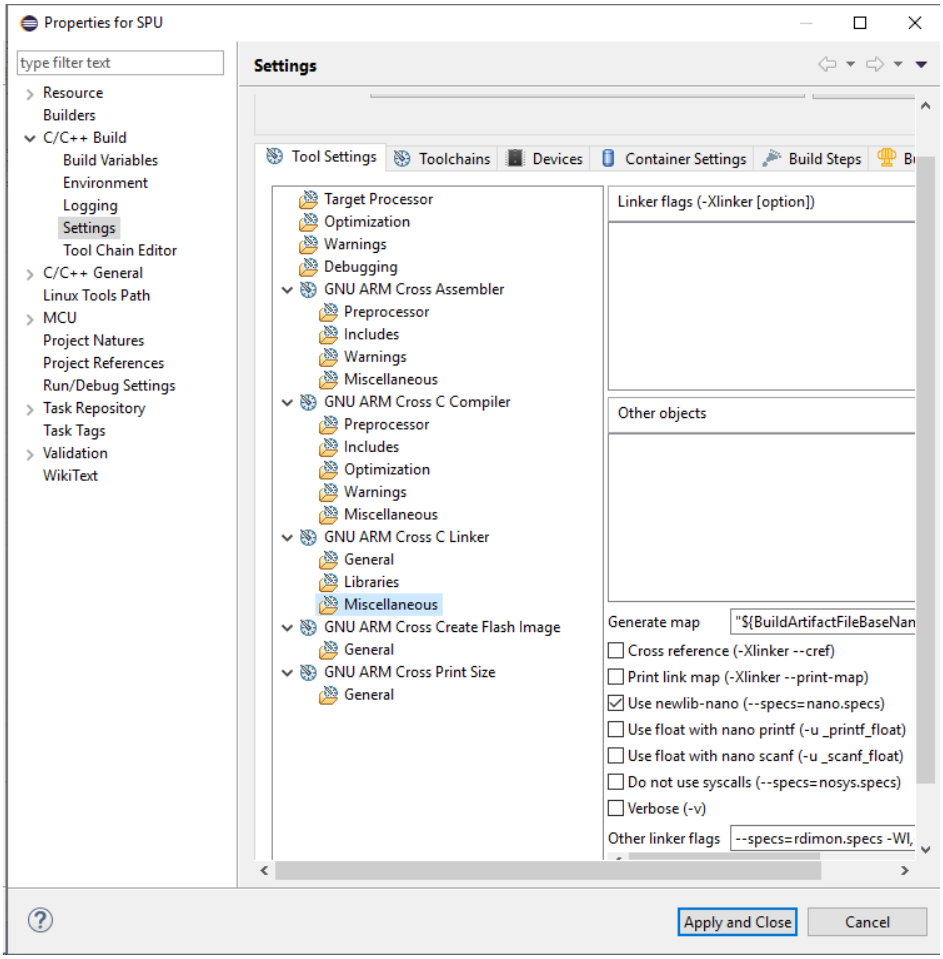

Figure 2-26 Select newlib-nano Option

12. If the creted project is a executable one, Figure 2-27 shows how to select the output file format in **Properties ➔ Settinng ➔ GNU ARM Cross Create Flash Image ➔ General** path.

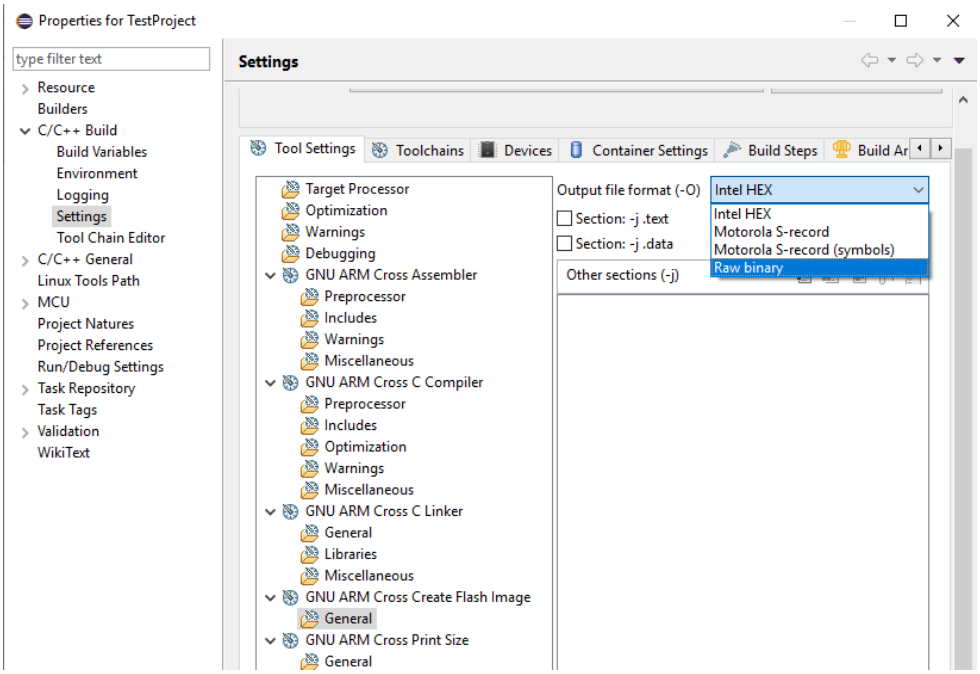

Figure 2-27 Setup Output File Format

2.2 Import Existing Eclipse Project

To import the current project, please click **File** ➔ **Import** to select the existing project into workspace as shown in Figure 2-28 and Figure 2-29, respectively. After importing the project, it can be modified and started debugging if necessary.

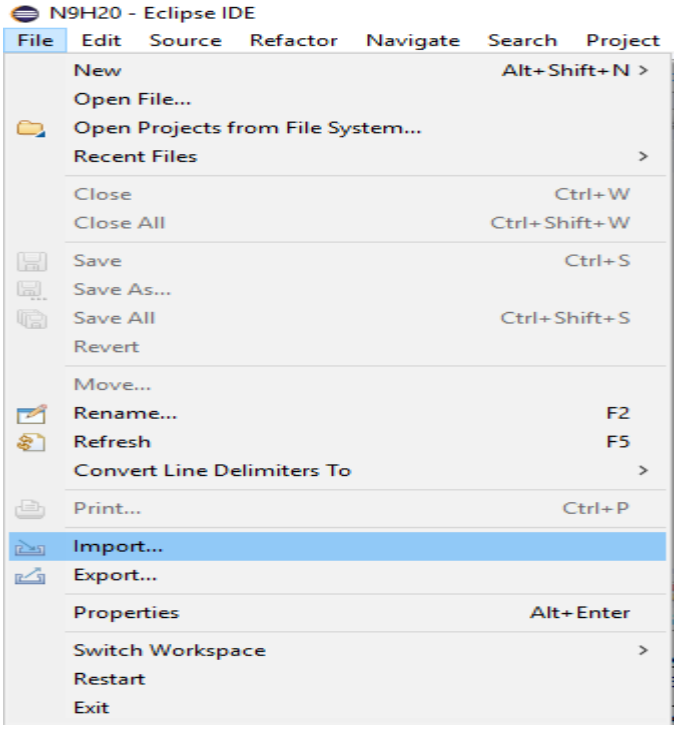

Figure 2-28 Import Existing Project

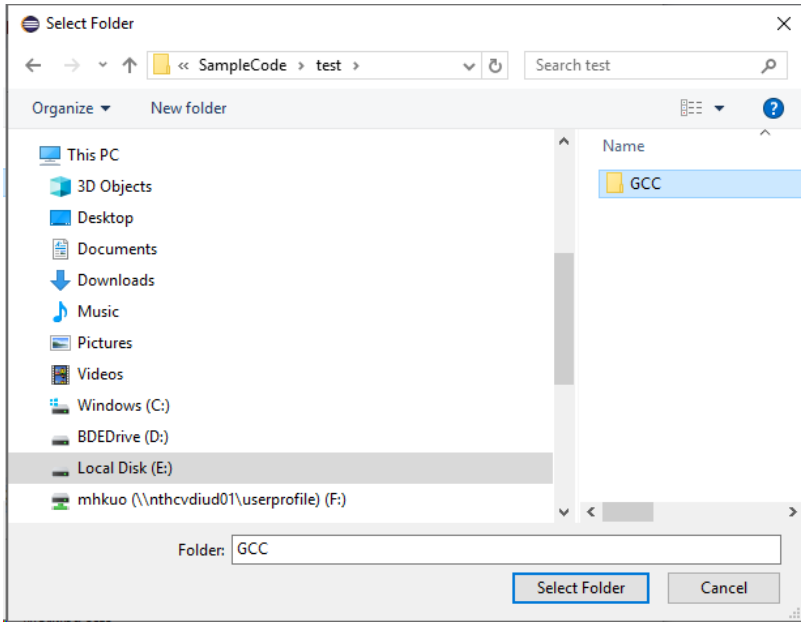

Figure 2-29 Select Imported Project

3 Eclipse Debug

3.1 Setup Debug Environment

Eclipse supports debugging using J-Link ICE. The J-Link plug-in program can be downloaded and installed from the website *http://gnu-mcu-eclipse.github.io/debug/jlink/install/* before to start debugging. After installation, set **J-Link path in Properties ➔ MCU ➔ Global SEGGER J-Link** as shown in Figure 3-1, and then press **Apply** button.

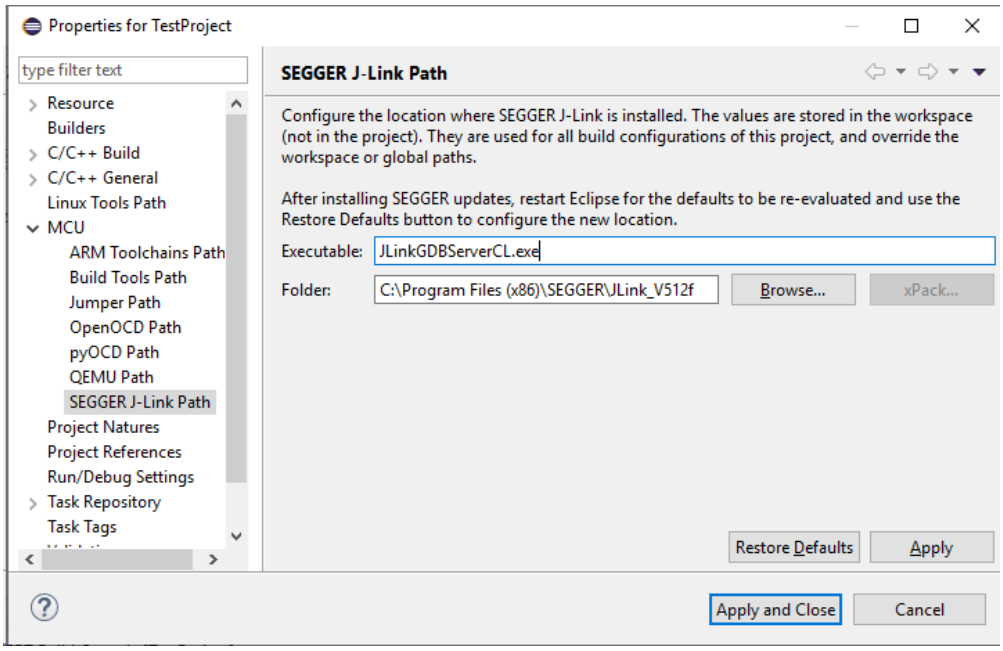

Figure 3-1 Global SEGGER J-Link Path Setting

The next step is to set **GDB SEGGER J-Link Debugging** options. As shown in Figure 3-2, click **Run ➔ Debug Configurations** and then expand **GDB SEGGER J-Link Debugging** confguration.

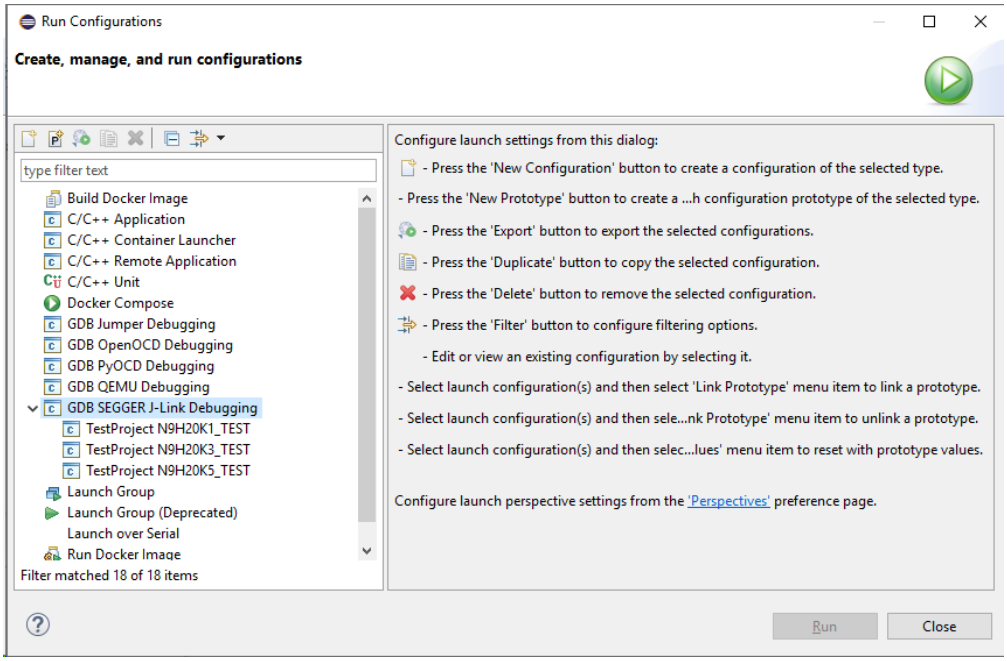

Figure 3-2 GDB SEGGER J-Link Debug

Selet corresponding configuration which match your device as should in Figure 3-3.

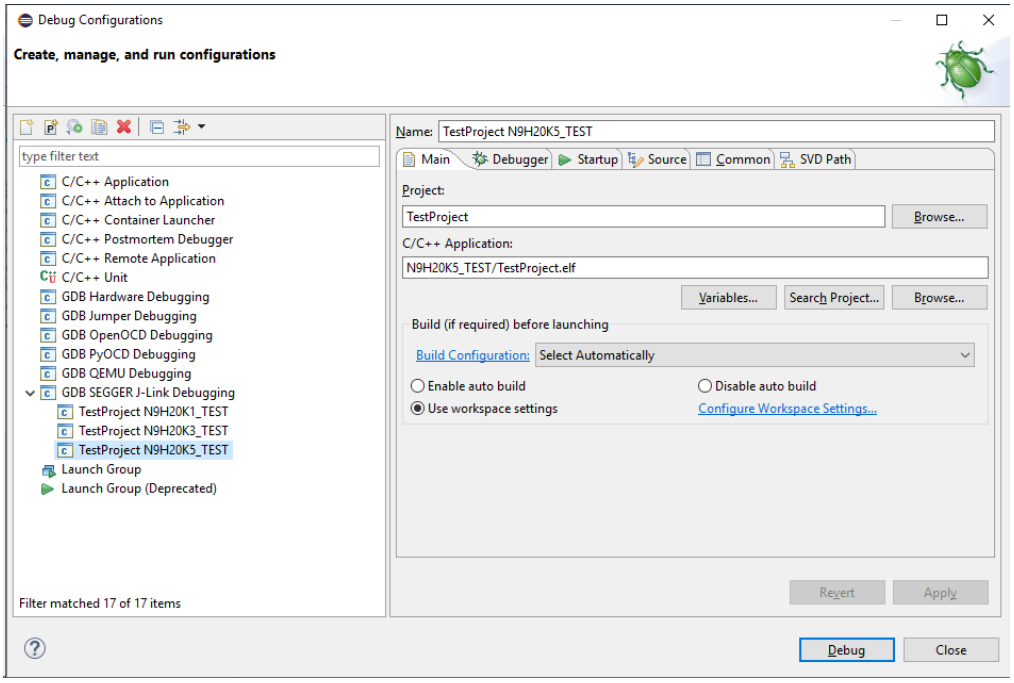

Figure 3-3 J-Link Debugger Main

Goto **Debugger** tab, this configuration is as in Figure 3-4.

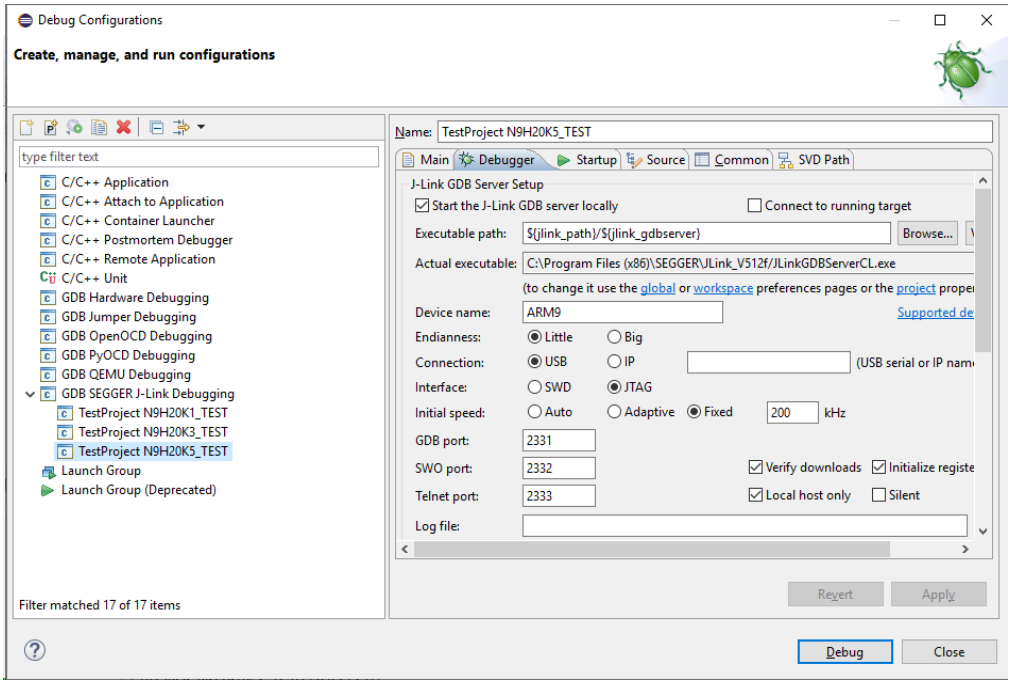

Figure 3-4 J-Link Debugger Setting

Goto **Startup** tab, this configuration is as shown in Figure 3-5. In **Startup**, some initail settings are given, including DRAM initialization, clock settings and so on.

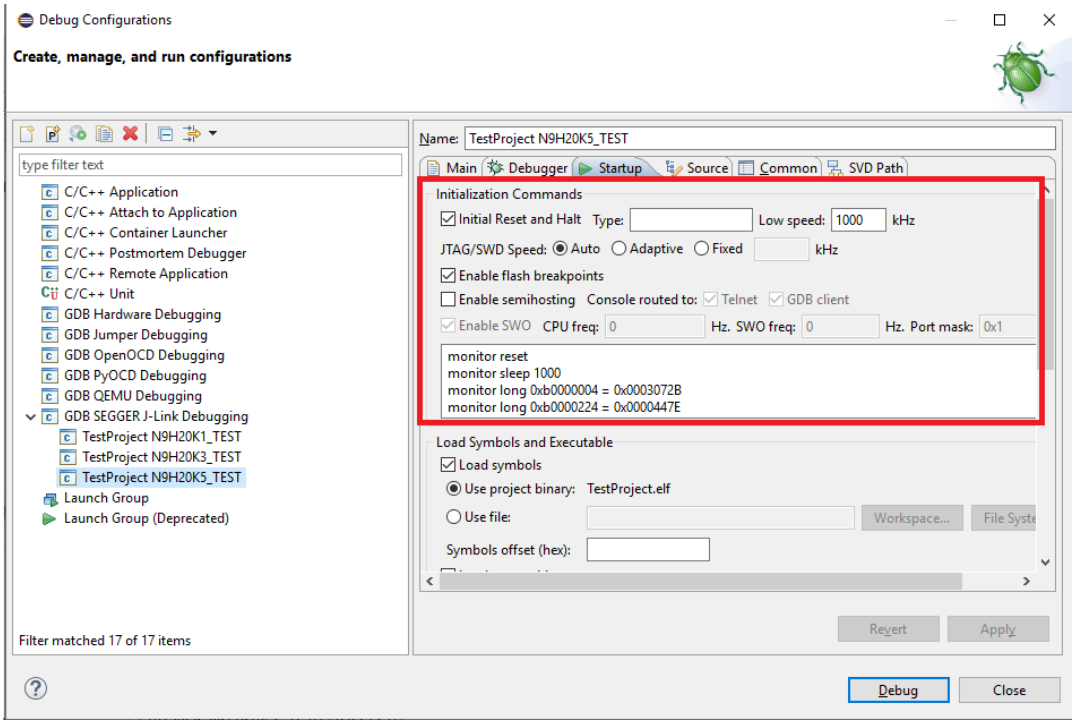

Figure 3-5 J-Link Startup Setting

After completing the settings, click **Debug** button to start debugging with J-Link.

3.2 Start Debug

This section introduces some simple operations for debugging.

1. **Free Run** the test code as shown in Figure 3-6.

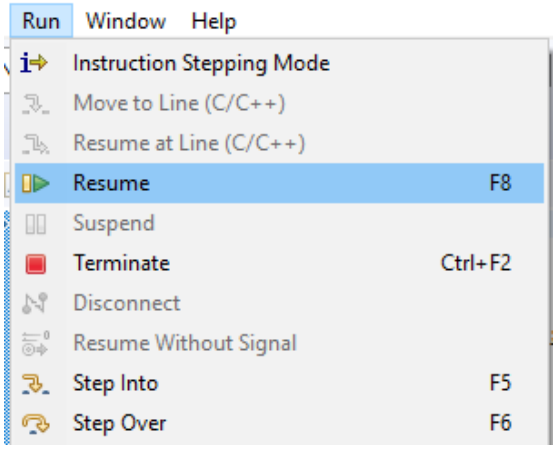

Figure 3-6 Free Run

2. **Step Into** & **Step Over** the function call as shown in Figure 3-7.

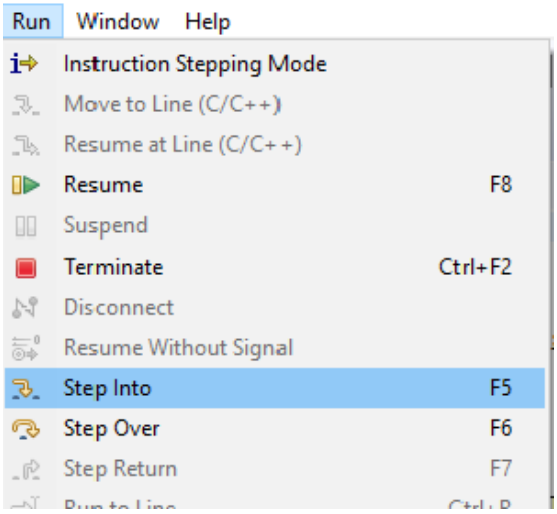

Figure 3-7 Step In & Step Over

3. Dump **Memory** window as in Figure 3-8.

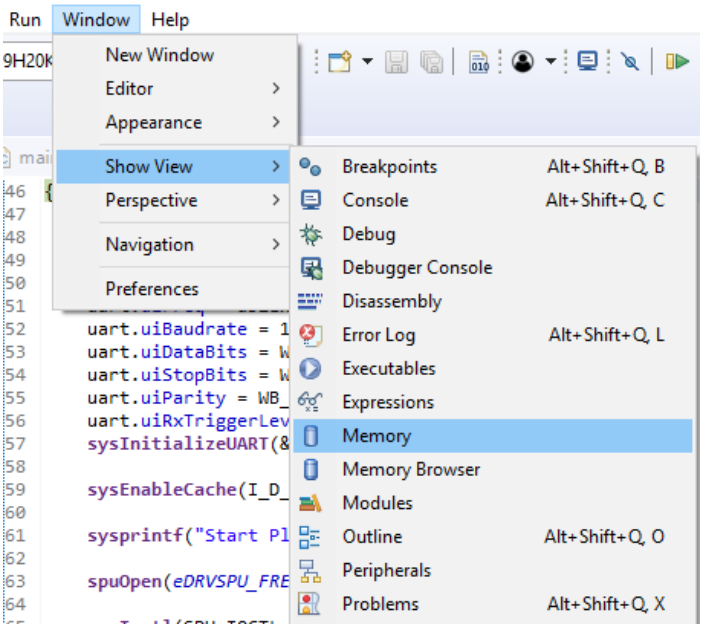

Figure 3-8 Dump Memory

4. Dump general **Registers** window as in Figure 3-9.

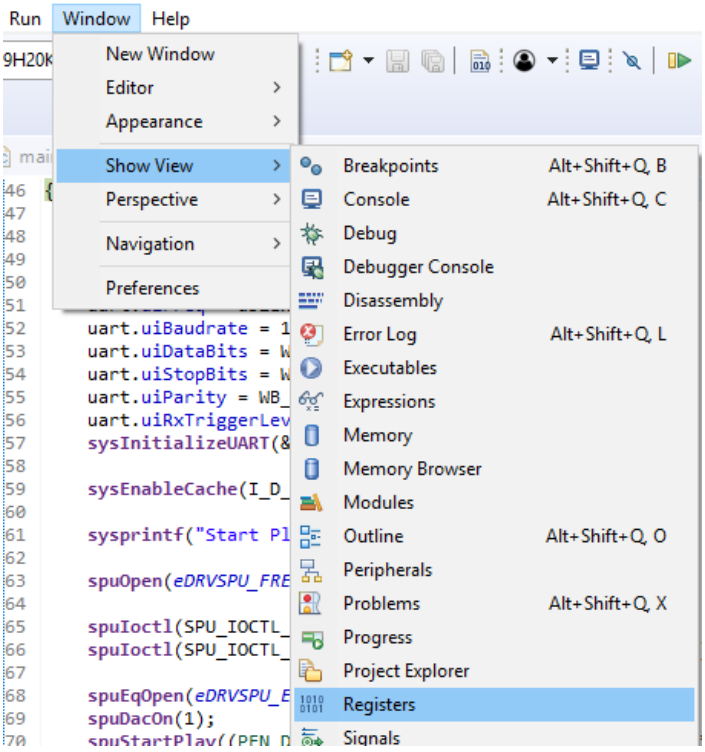

Figure 3-9 Dump General Register

5. A breakpoint can be enabled and disabled by right-clicking on its icon or by right-clicking on its

description in the Breakpoints view

As for other operations in the debugger, the user can refer to **Help** window to get further information.

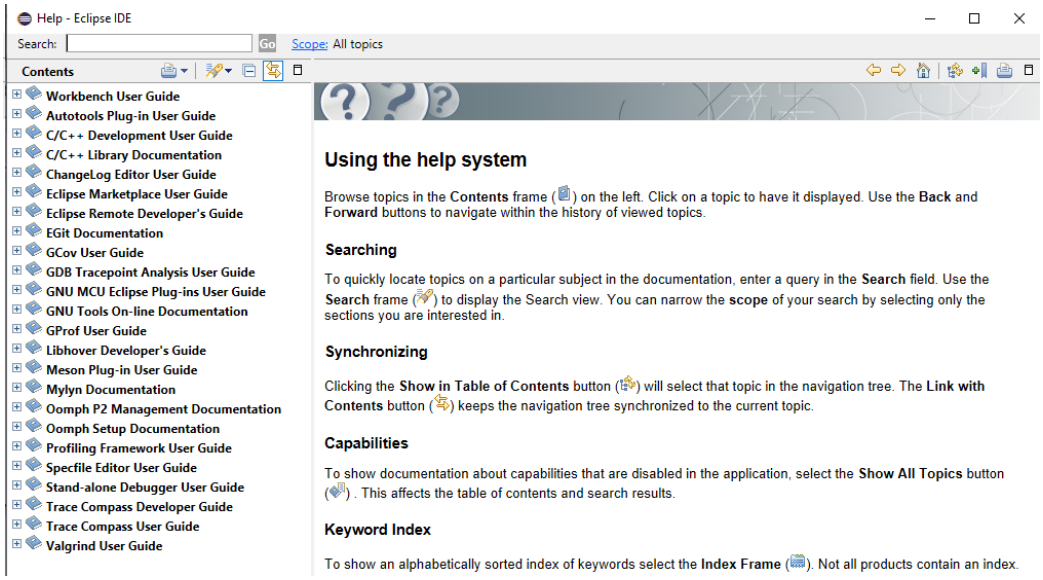

Figure 3-10 Help Window

4 Revision History

Date	Revision	Description
2020.09.25	1.00	Initially issued.
2021.07.14	1.01	Modify Eclipse installation flow.
2022.05.13	1.02	Modify GCC tool chain and Build tools.

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.