GigaDevice MCU Team	Version	23 Pages
	English V 1.1	
	Name： Embedded Builder User Manual	

Embedded Builder User Manual

Table of Contents

1. Introduction

Embedded Builder stands as an intricately crafted C-language integrated development environment (IDE) rooted in the Eclipse platform. It serves as an exceptional embedded software development platform based on the RISC-V and ARM® Cortex processor architectures.

The objective of this document is to provide guidance on the utilization of Embedded Builder. It offers functionalities encompassing project creation, graphical MCU configuration, and project debugging.

1.1. Installation

Embedded Builder is a software based on Eclipse and Java platform. To facilitate its utilization, it is imperative to ensure the successful installation and configuration of the Java Development Kit (JDK) using the version 1.8 on your computer beforehand.

Embedded Builder is a green and portable software that operates without the need for installation.

1.2. Launch

Click the **EmbeddedBuilder.exe** executable file in the Eclipse package provided by GigaDevice to start the software.

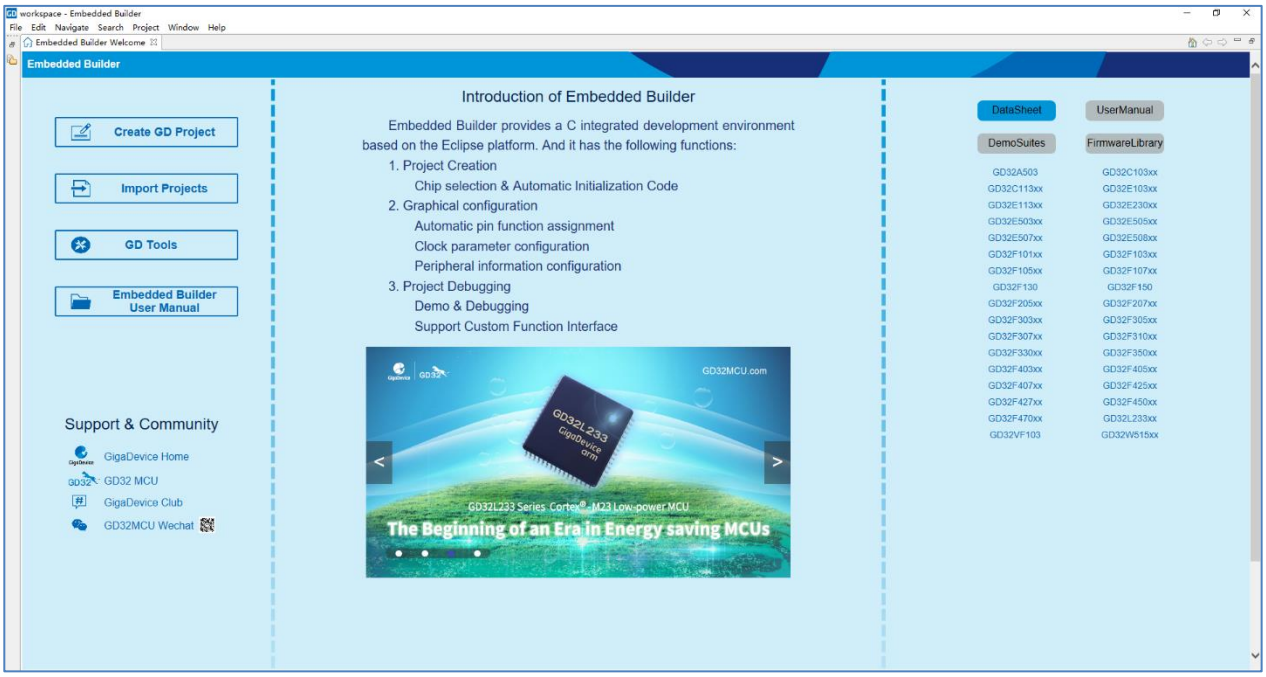

Figure 1-1

2. Starting a GD Project

2.1. Creating a GD Project

1) Navigate to **File > New > Other > GD Project** within the menu commands, or **Create GD Project** on the **Welcome** page to start the new project wizard. Click the **Next** button to continue.

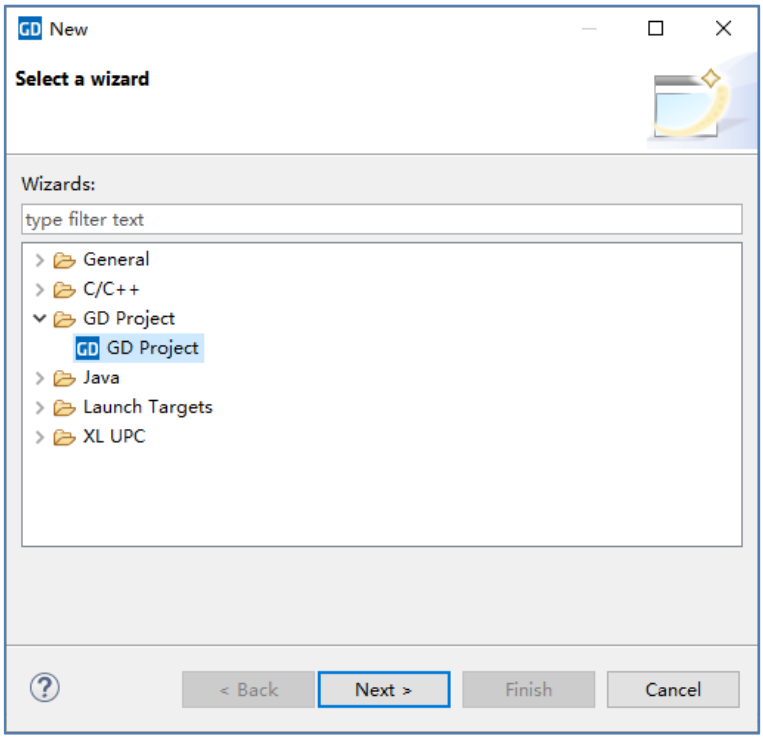

Figure 2-1

2) Input a designated name into the **Project Name** field, then proceed by clicking the **Next** button.

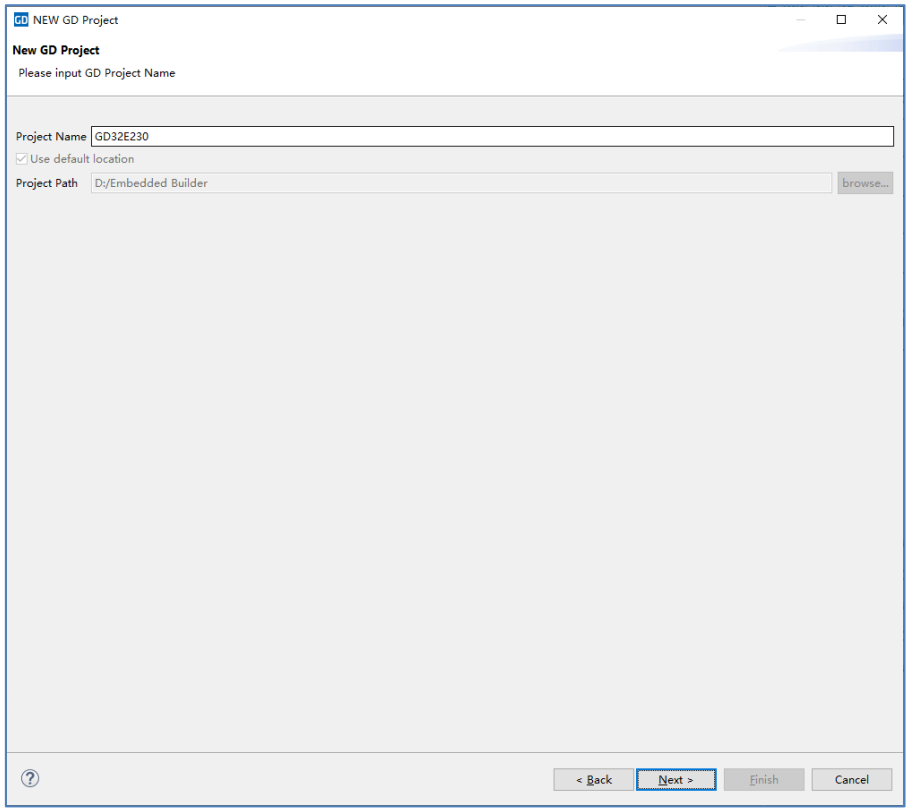

Figure 2-2

3) Select a required device from the **Device** list, then click **Finish** button.

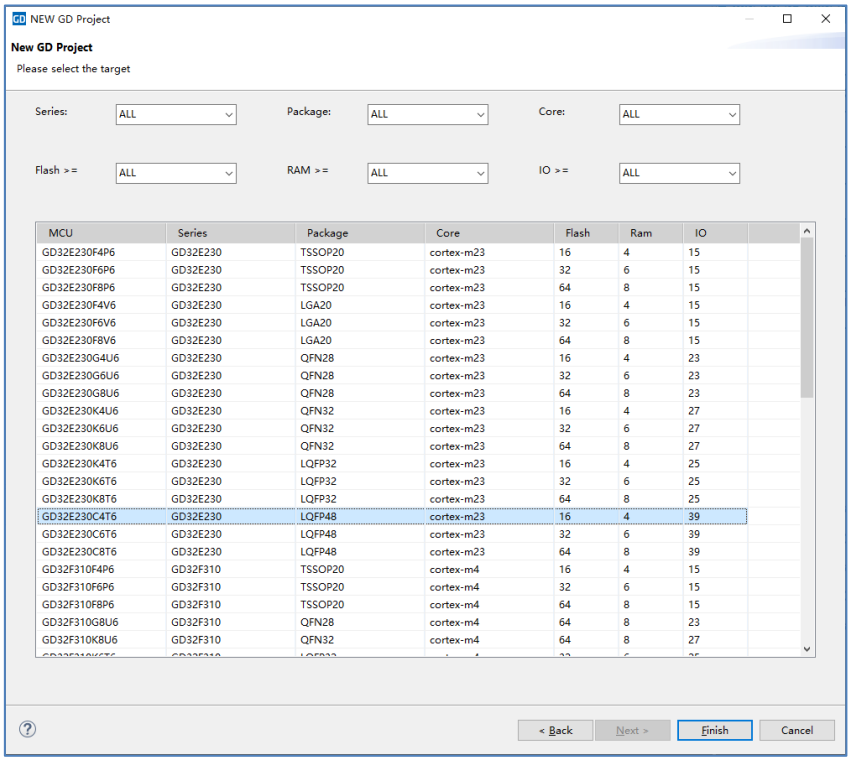

Figure 2-3

2.2. Graphical Configuration

Embedded Builder provides automatic pin function assignment, clock parameter configuration and peripheral information configuration.

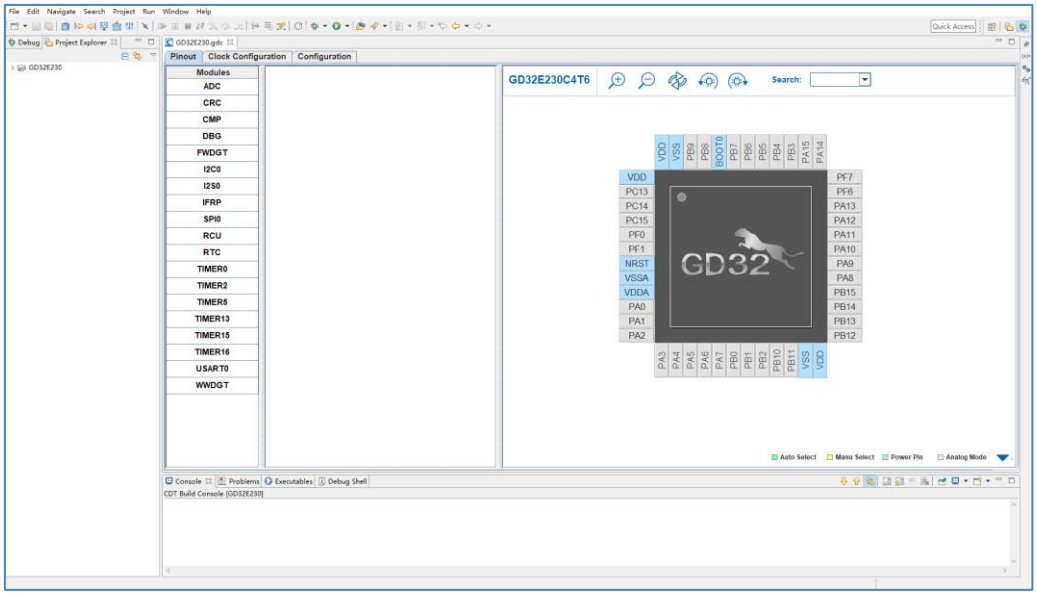

Figure 2-4

Users can customize the peripheral functions according to your specific requirements. The pin assignment function can be operated intelligently, providing both automatic and manual pin assignment support.

The color of pins represents different meanings. **Blue** represents non-configurable pins. **Gray** represents configurable pins. **Green** represents the pins associated with the pinout function, which support automatic and manual assignment. **Yellow** represents the pins that are not associated with the pinout function and only support manual assignment.

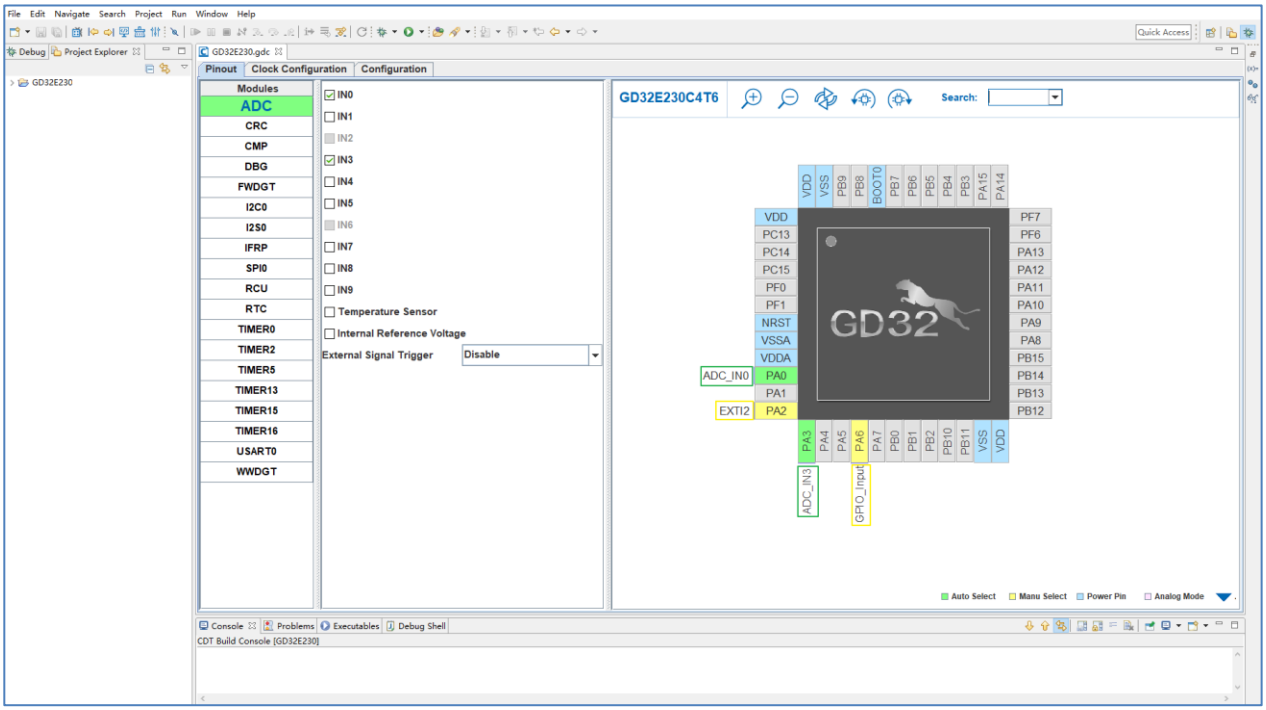

Figure 2-5

Pink indicates that the pin can be configured for multiple functions at the same time.

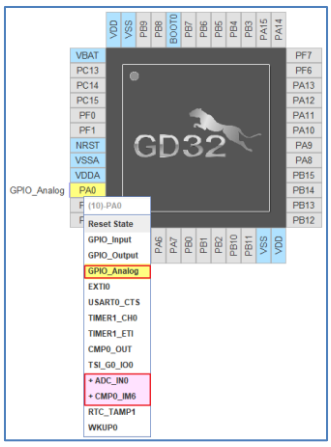

Figure 2-6

Embedded Builder presents the initial clock configuration, enabling users to manually adjust the clock parameters as needed.

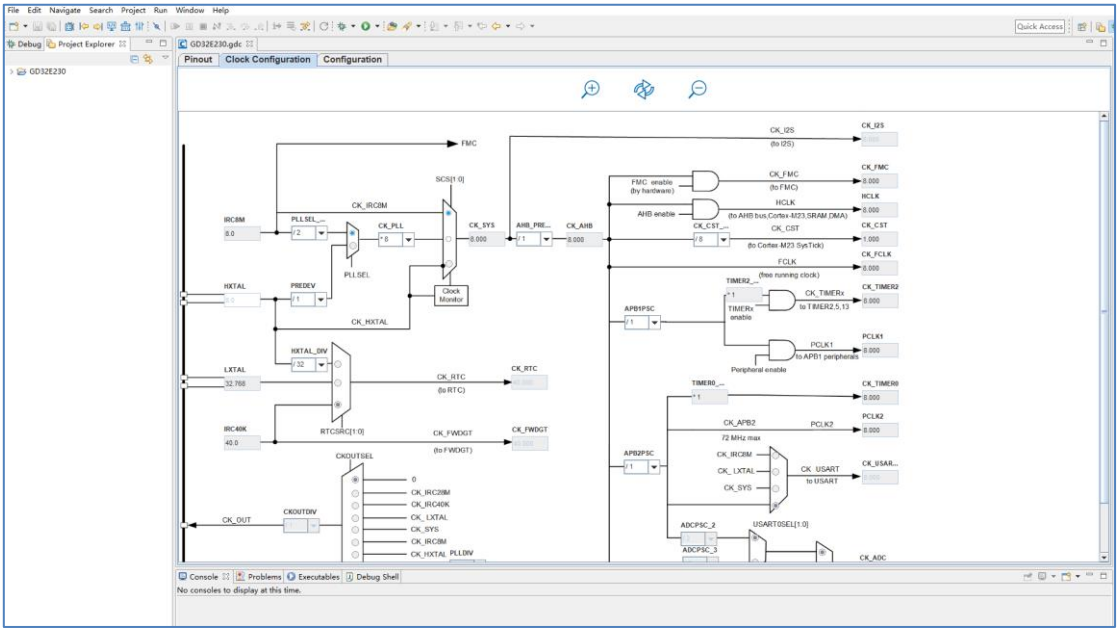

Figure 2-7

Embedded Builder facilitates configuration of peripheral modules, allowing users to customize peripheral parameters. It encompasses the setup of NVIC, DMA and GPIO parameters.

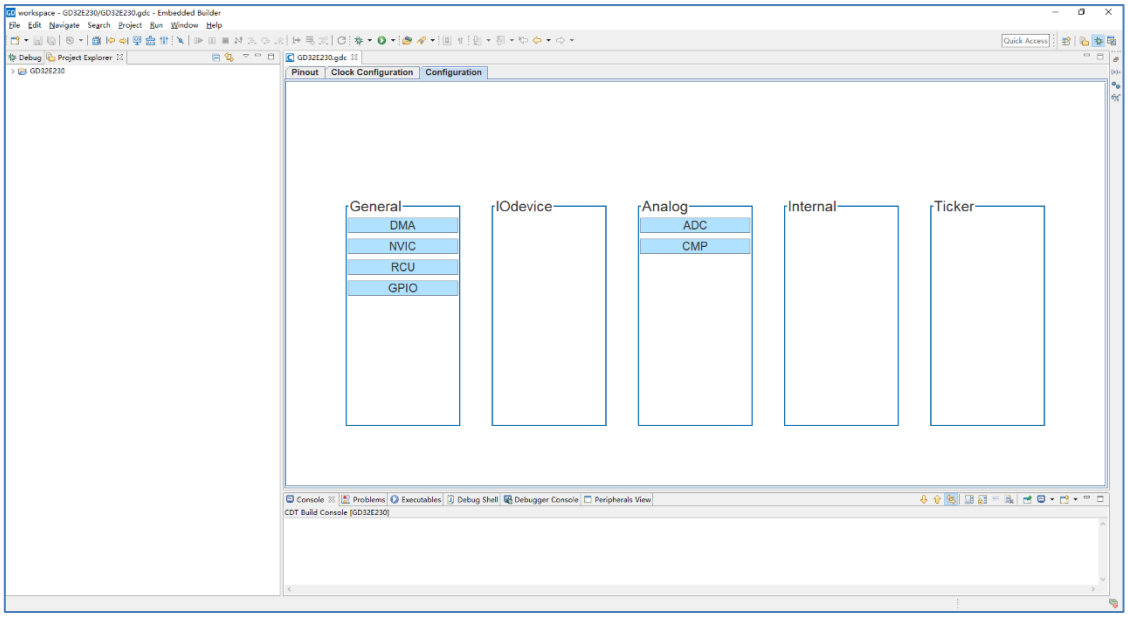

Figure 2-8

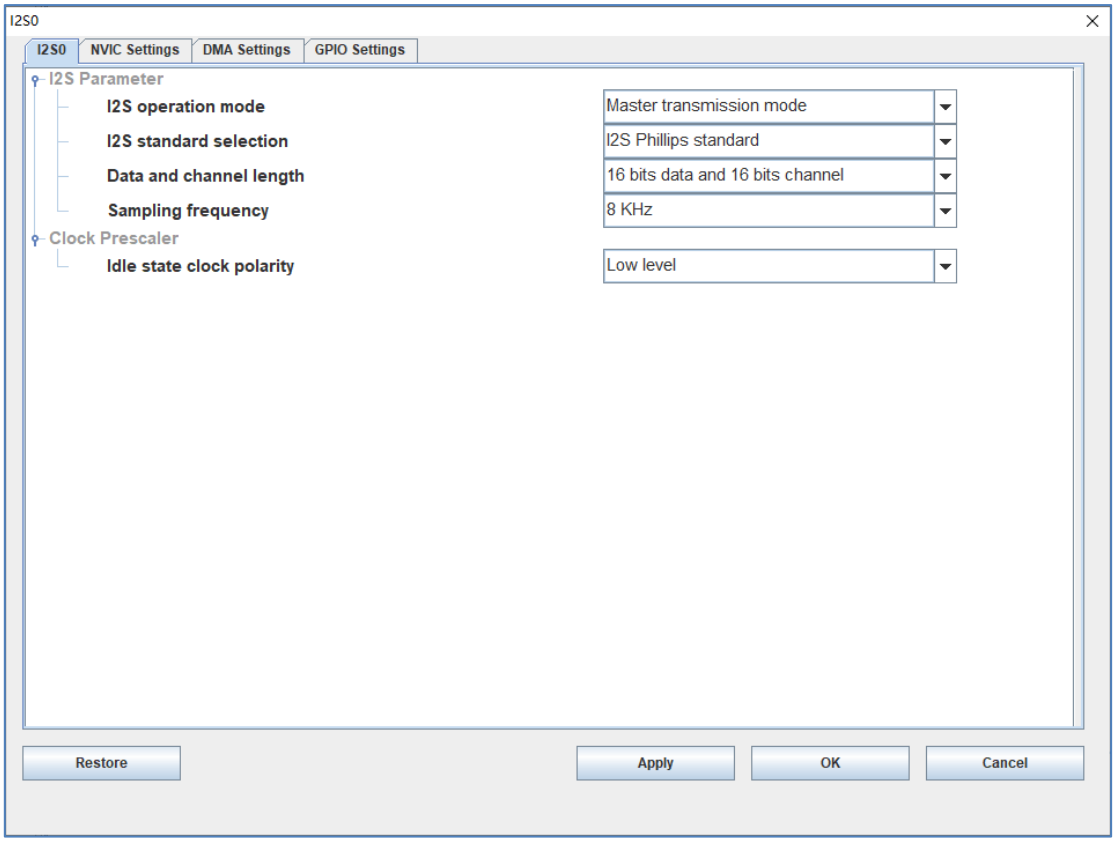

Figure 2-9

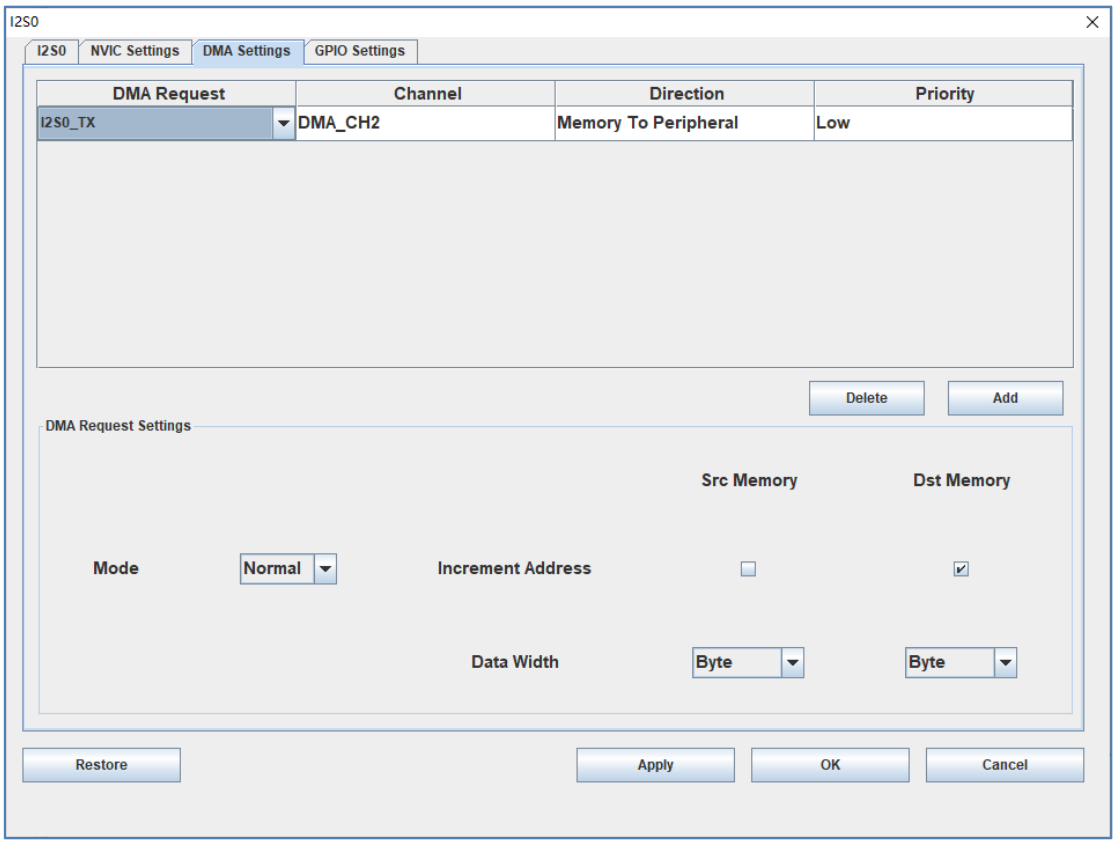

Figure 2-10

2.3. Code Generation

To initiate code generation, double-click the **gdc** file and navigate to **Project > Generate Code** within the menu. Upon successful code generation, Embedded Builder will display a confirmation prompt.

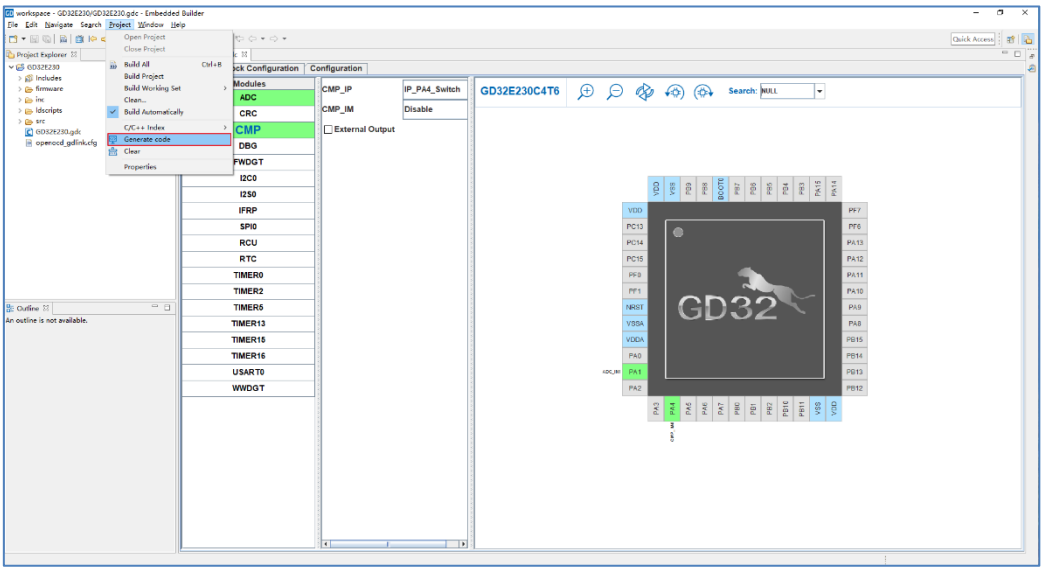

Figure 2-11

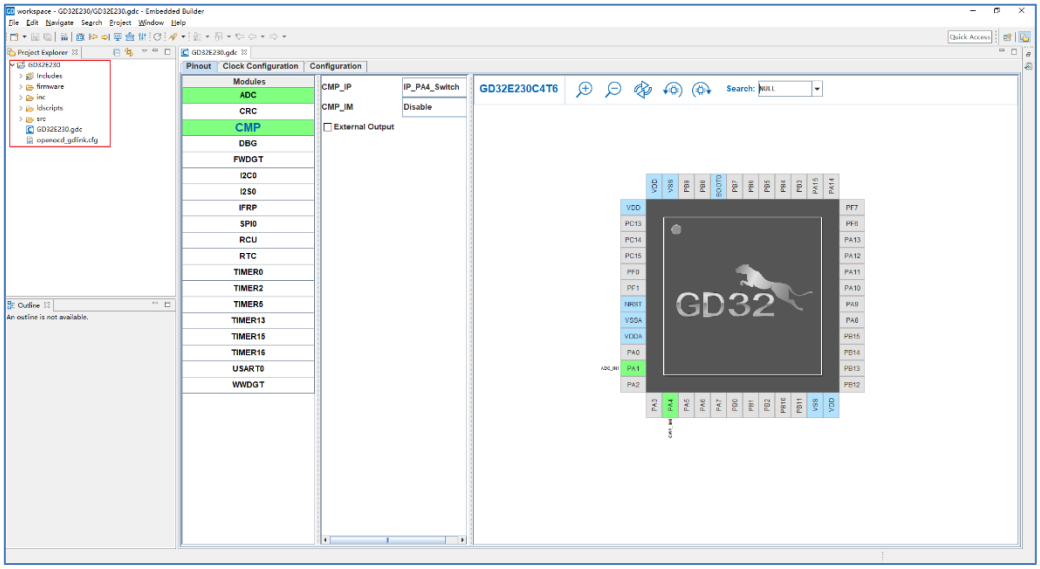

Figure 2-12

3. Starting a GD Template Project

3.1. Creating a C Project

1) Navigate to **File > New > Other > C/C++ > C Project** menu command to start the new project wizard, Click **Next**.

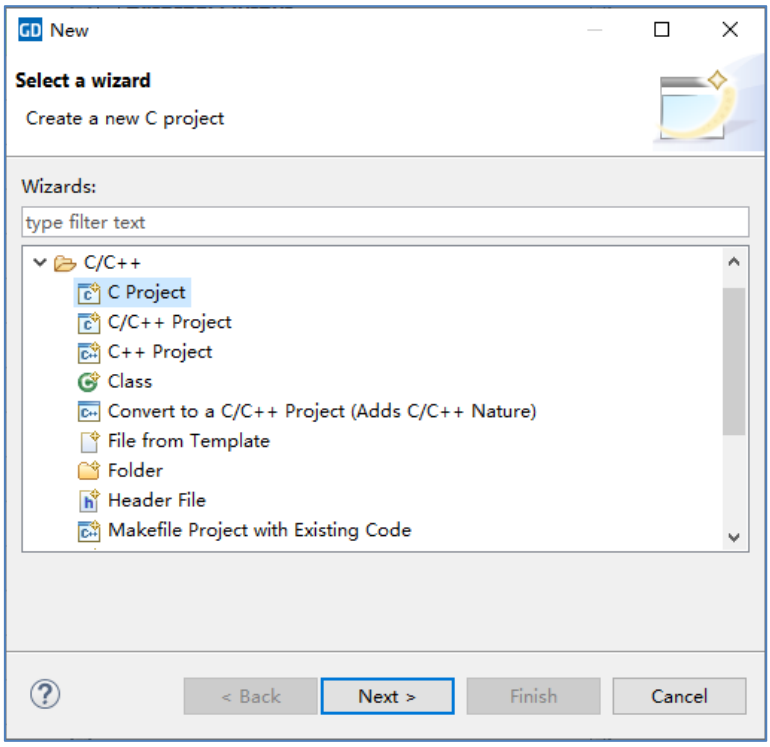

Figure 3-1

2) Select to create **GigaDevice ARM C Project** or **GigaDevice RISC-V C Project**, and the compilation chain on the right will automatically associate with **GD ARM MCU Toolchain** or **GD RISC-V MCU Toolchain**. Then, enter a name in **Project name** field, Click **Next**.

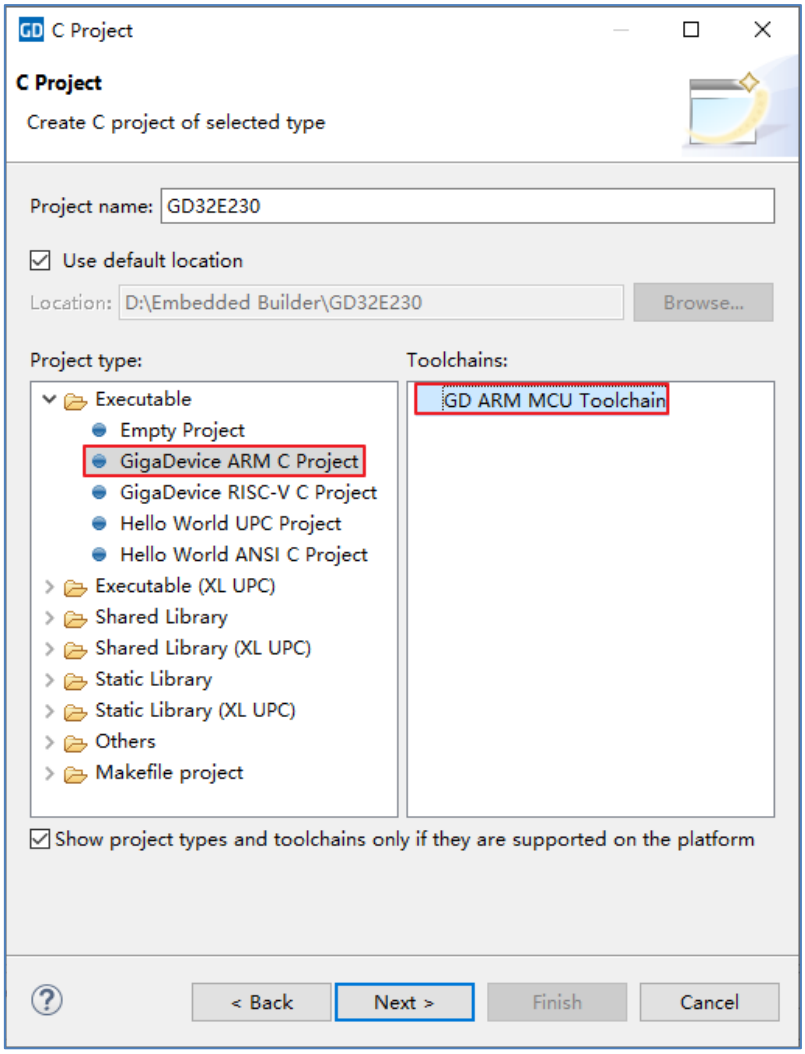

Figure 3-2

3) Select a required device from the **Device** list, click **Finish**.

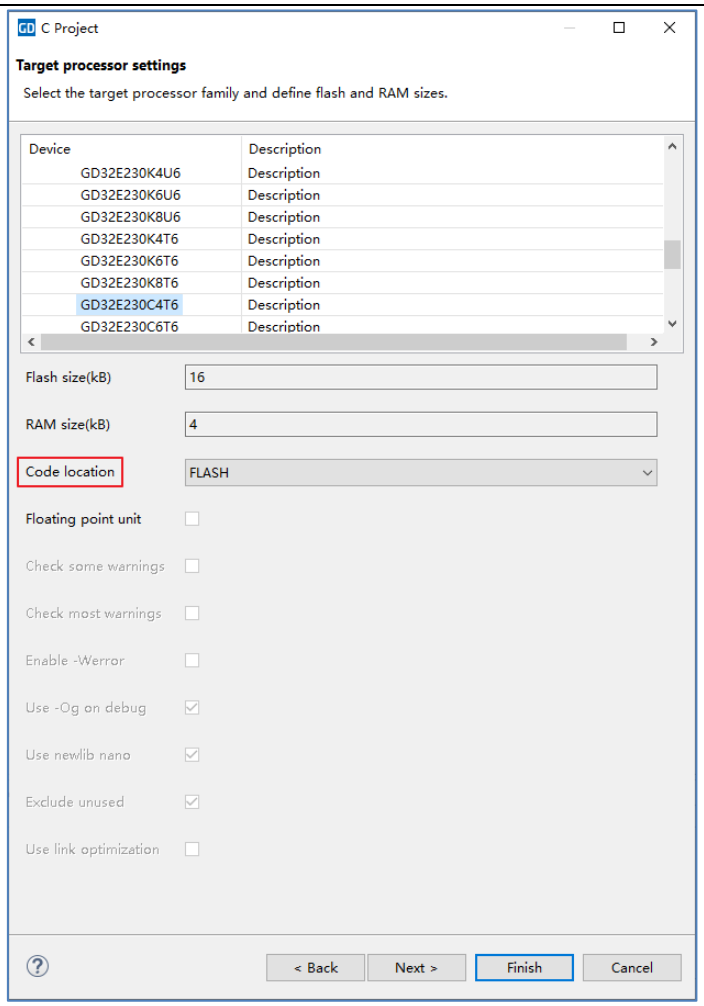

Figure 3-3

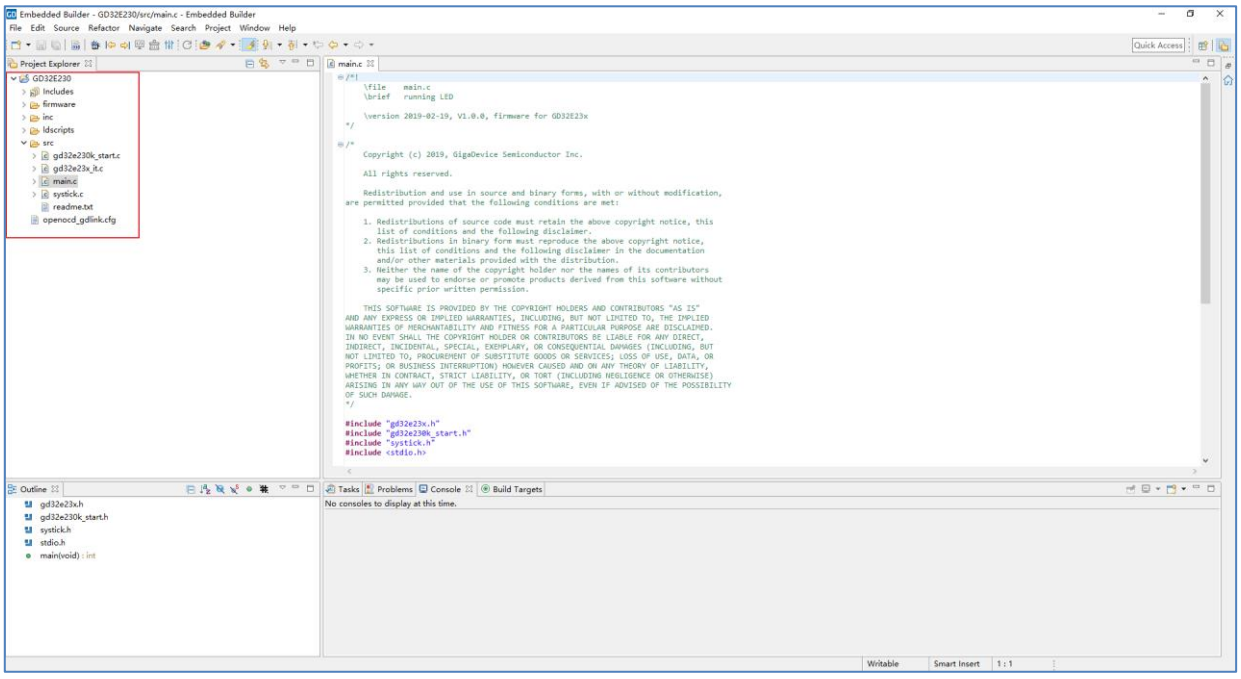

Figure 3-4

4. Project Settings

4.1. Framework of Project

A project is created with the default settings and a full set of configurations based on the project type and the device you selected. **C/C++** perspective is opened by default after the project is created.

Expand the project folder to display the project framework, which includes the GD firmware, linker file and peripheral configuration code.

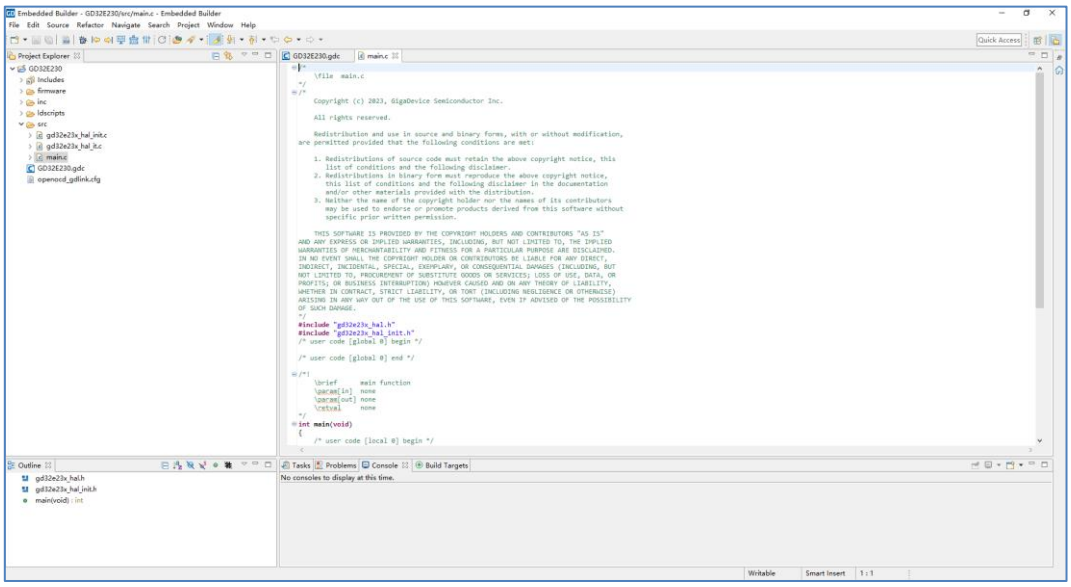

Figure 4-1

In the **Project Explorer** view, right-click the menu of the target project which includes functions of creating a new file, building a project, opening the project properties dialog and so on.

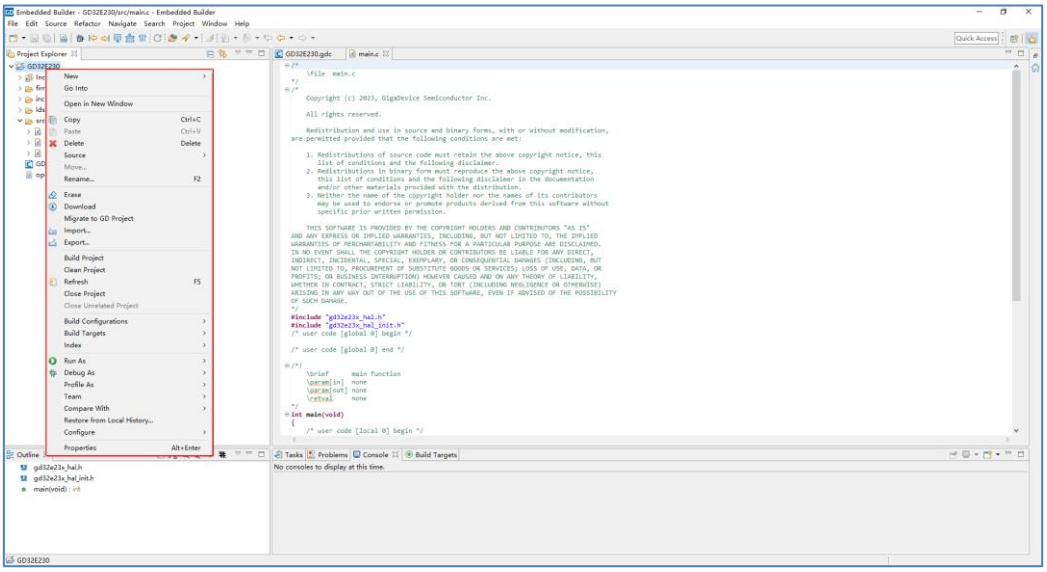

Figure 4-2

4.2. Build Configuration

In the project properties dialog, you can see the default settings and configurations, such as include paths, defined symbols, the linker script file and the toolchain path. Depending on the default settings and configurations, the project can be compiled and linked correctly.

Use the **Tool Settings** tab to configure the compiler and linker options.

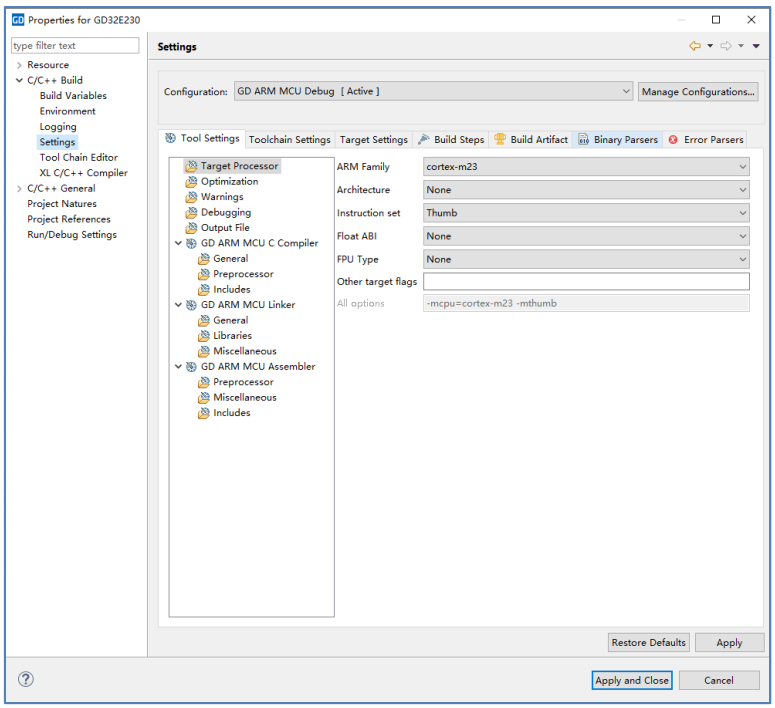

Figure 4-3

The **Target Processor** category is used to configure the compiler options related to the target processor.

The **Optimization** category is used to configure various sorts of optimization options. The **Warnings** category is used to configure the options to request or suppress warnings. The **Debugging** category is used to configure various special options which are used for debugging.

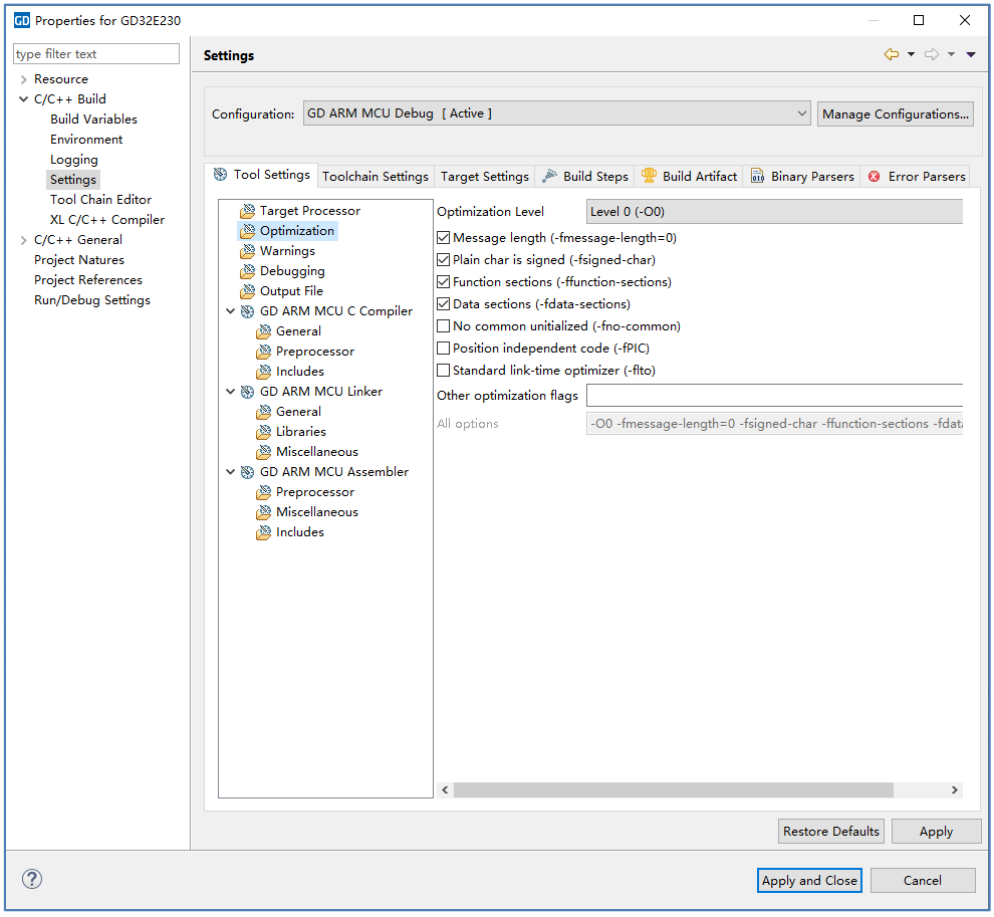

Figure 4-4

The **Output File** category is used to output files in other formats, such as list, dump, map, hex, bin and size file. You can enter some options in **Other objcopy flags** edit box to control the hex or bin file. The **Other objdump flags** edit box is used to configure the list options. The **Other size flags** edit box is used to enter some size options.

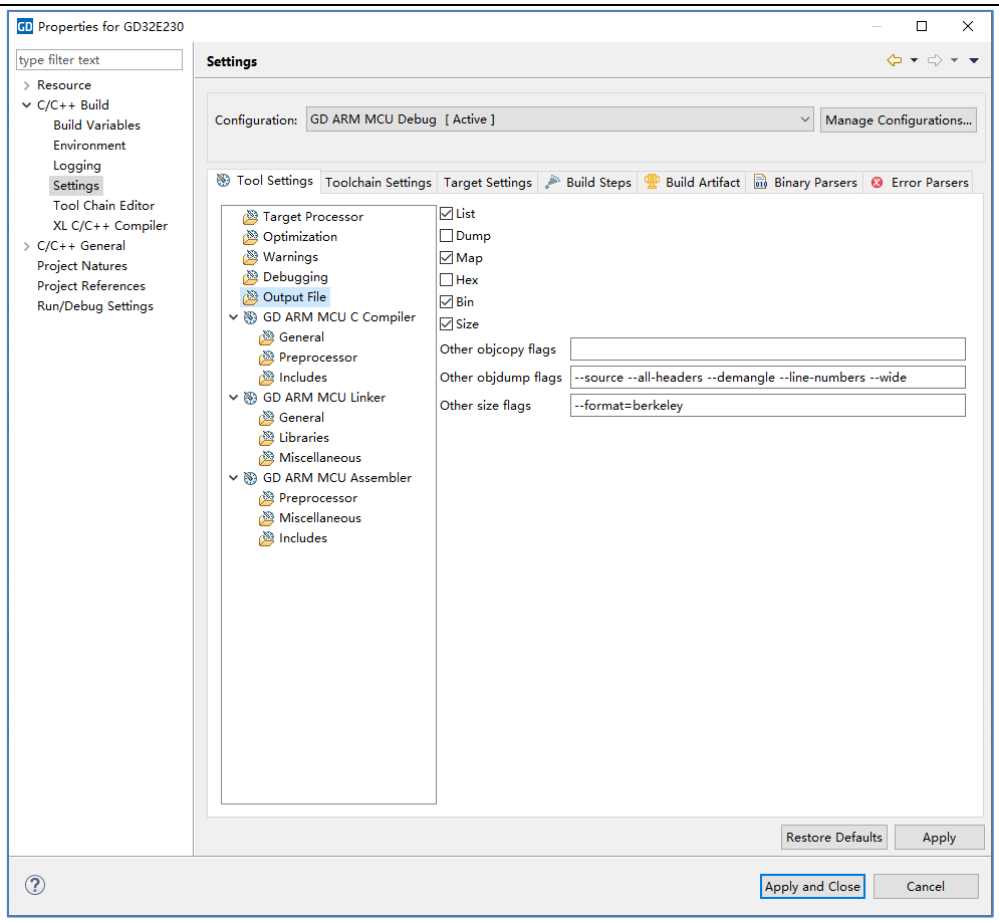

Figure 4-5

The **GD ARM MCU Assembler** tool is used to configure the options for the assembler.

The **GD ARM MCU C Compiler** tool is used to configure the options for C compiler. In the **General** category, you can configure the start address and length attributes of Flash and RAM, which are correspond to the memory configuration of the linker script file in the valid format.

4.3. Toolchain Configuration

Paths of the toolchain and build tool is the default configuration.

Select **Project > Properties > C/C++ Build > Settings**. In the **Toolchain Settings** tab, click **Browse** button to choose required toolchain path and the build tool path. Then, click **Apply** or **Apply and Close** button.

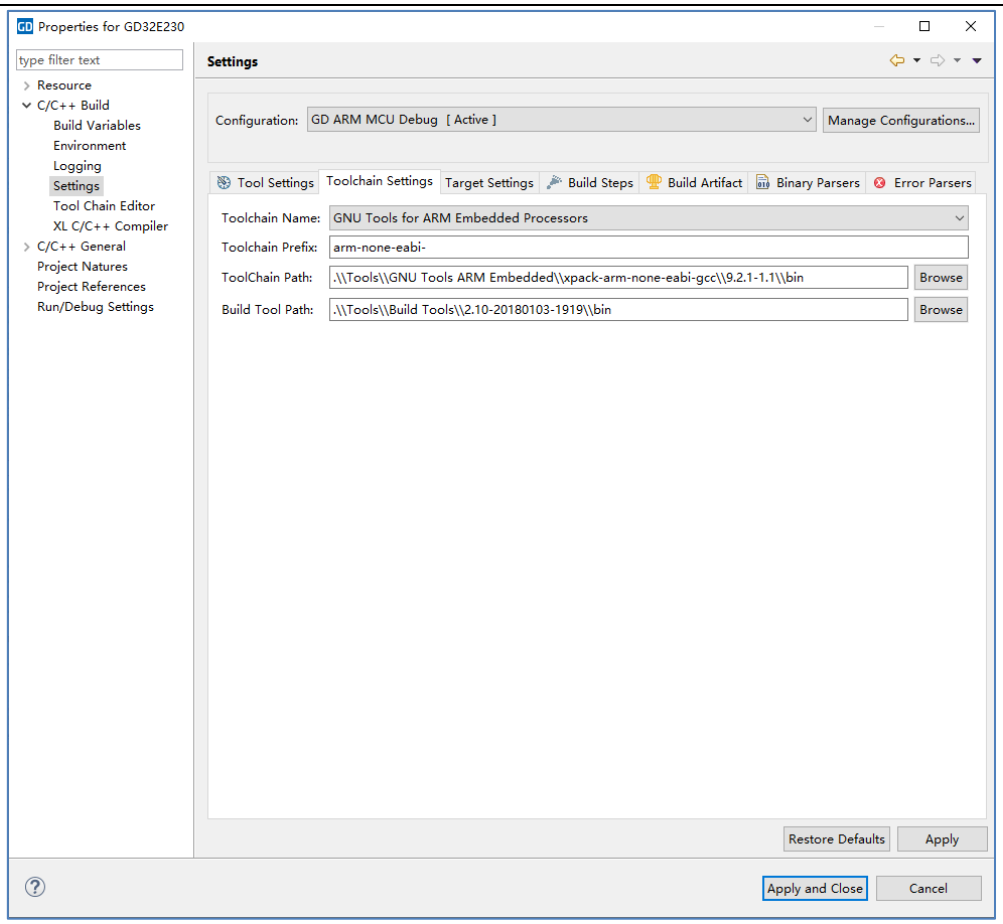

Figure 4-6

4.4. Modify Toolchain

Right click the project name, click **Properties,** then click **Properties > C/C++ Build > Settings >Toolchain Settings,** modify the content in the **Toolchain Prefix** to the new toolchain prefix, then modify the **ToolChain Path** to the **new toolchain path**, then click **Apply**. When debugging, please open **Debug Configuration** of the debugging project, then click **Apply** and **Debug**.

5. Debugging and Running Project

5.1. Creating a Debug/Run Configuration

Select your target project, then click the debug configuration icon on the toolbar as shown in Figure 5-1. Then, the **Debug Configurations** dialog window will be opened.

Figure 5-1

Double-click **GDB General Debugging** to create a new debug launch configuration. If the configuration is created successfully, the debugging project name will be in the **Name** field.

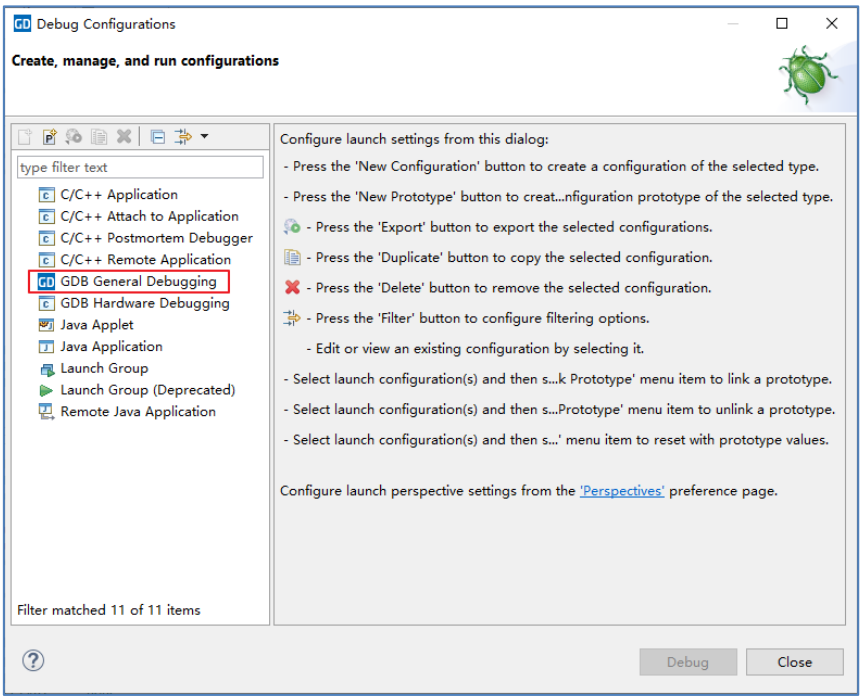

Figure 5-2

Please make sure that the project has been built successfully.

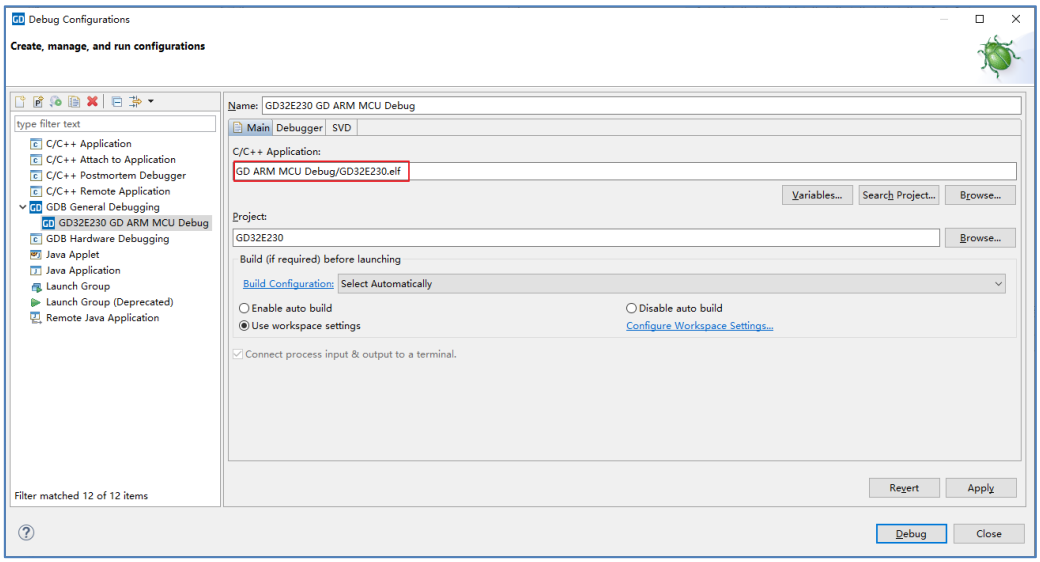

Figure 5-3

Notice it should be **terminated** by clicking the red block when you stop debugging. If you don't click it, it will have influence on the next debugging.

Figure 5-4

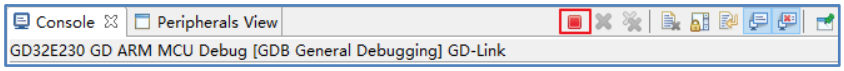

Figure 5-5

5.2. Editing the Debug/Run Configuration

The **GDB General Debugging** wizard contains three tabs as shown in Figure 5-6 and Figure 5-7. You will use these tabs to customize your run and debug configurations. Click the **Debugger** tab to select a debugger type from GD-Link. If GD-link is selected as the debugger, you should select GDB Server between GDLinkGDBServer and Openocd. In addition, click the **Scan** button to read the core ID of the target device by your selected debugger.

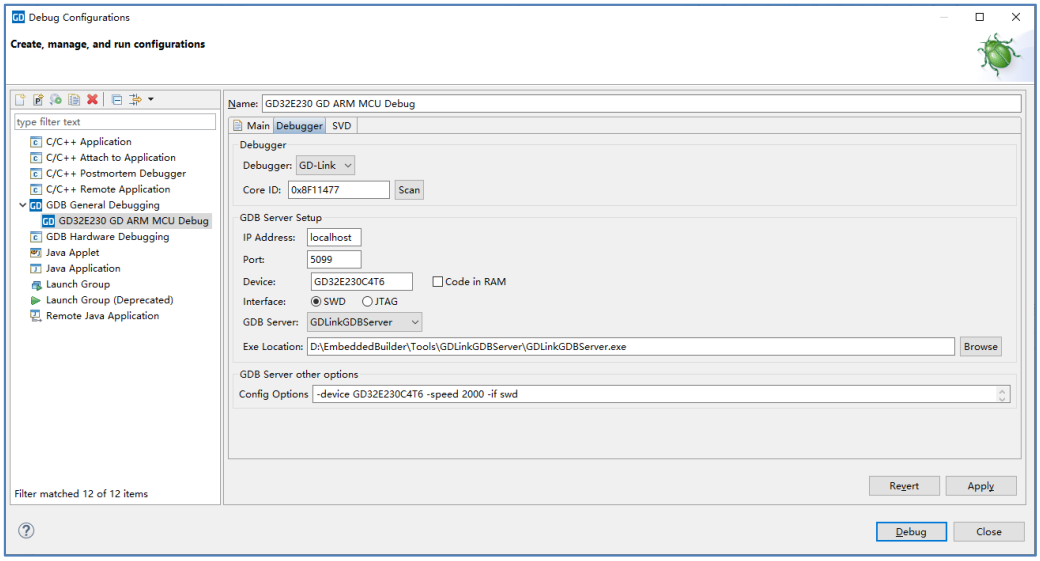

Figure 5-6

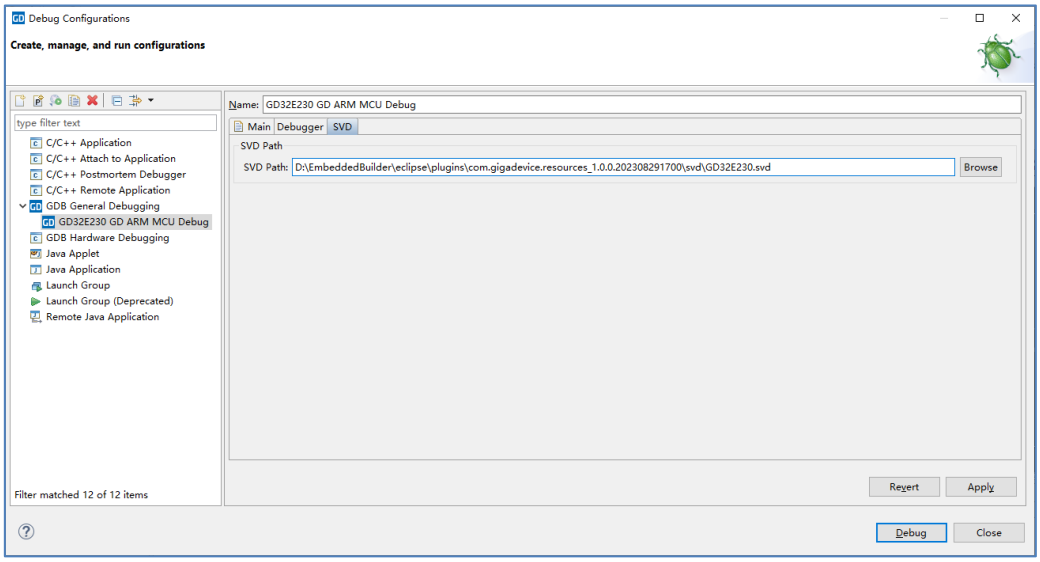

Figure 5-7

5.3. Debugging/Running

On the toolbar, choose **Debug** from the left drop-down menu as shown in Figure 5-8. Choose **Debug** from launch mode menu to start debugging.

Figure 5-8

Click the **Erase** menu command to erase chip, and click the **Download** menu command to download. You can see the output and results of the run command in the **Console** view.

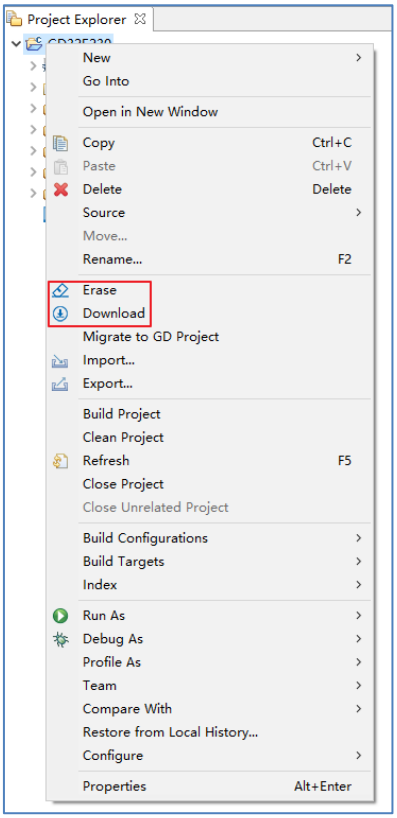

Figure 5-9

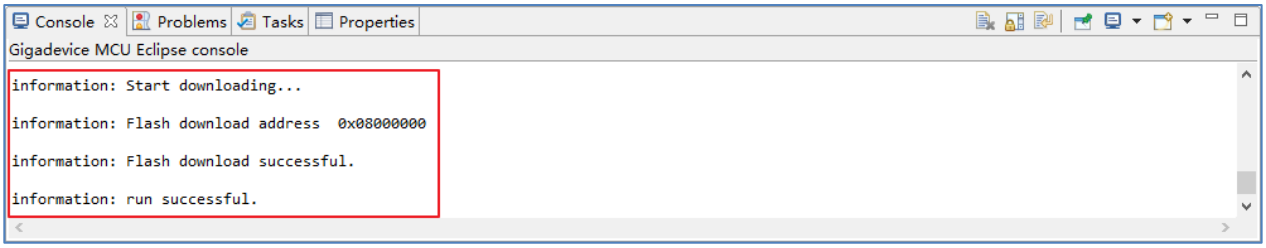

Figure 5-10

5.4. Resetting during Debugging

You can reset device during debugging by clicking the **reset** button on the toolbar.

Figure 5-11

5.5. Watching Registers during Debugging

To monitor registers during debugging, you need to open the **Peripherals View** first. When the MCU is halted, you can watch the peripheral register value.

Select **Window > Show View > Other** menu command to start a new wizard, then expand

the **Debug** folder and select **Peripherals View**. Click **Open**.

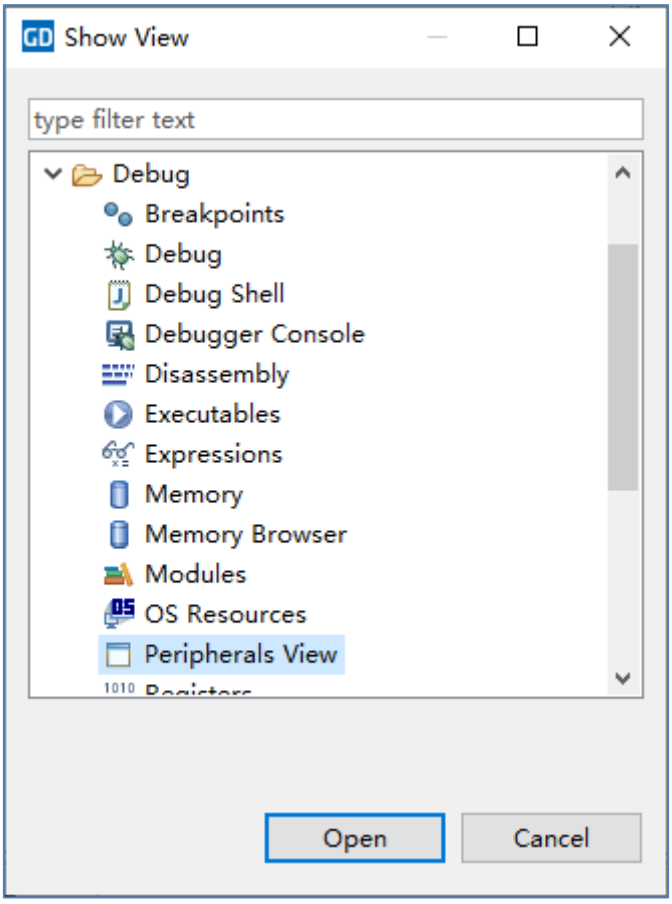

Figure 5-12

You can modify registers in the **Peripherals View**. Type a new value in the **HEX Value** column of target register bits row, then press the **Enter** key to start the modification. If the modification is successful, both the register bits and the register value will be refreshed. Otherwise, the modification is invalid, which may be due to the peripheral clock is not enabled.

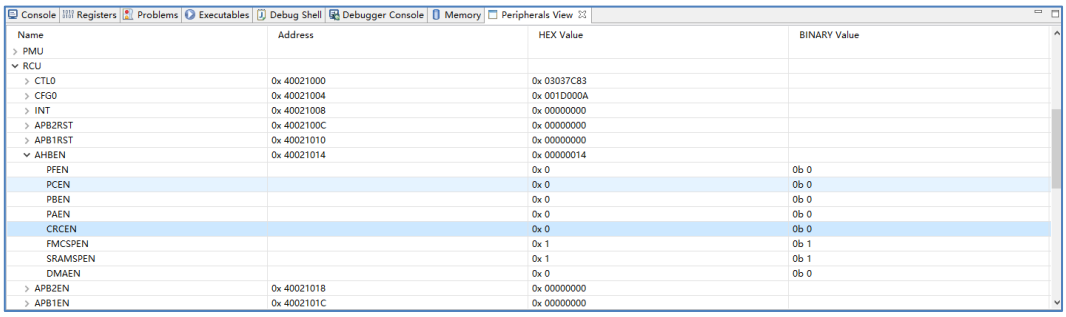

Figure 5-13

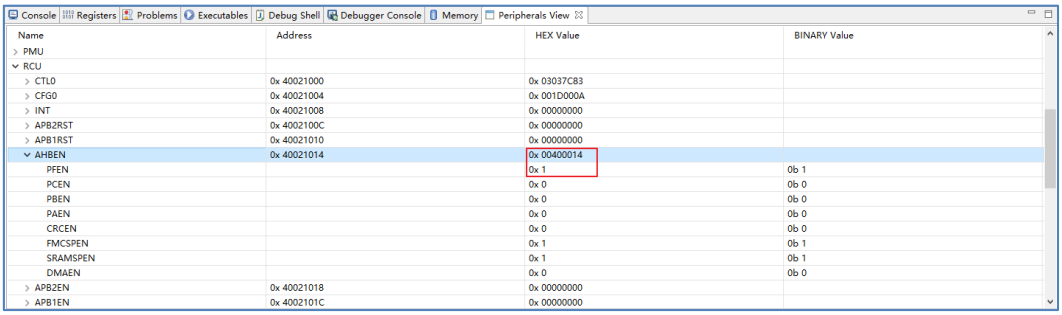

Figure 5-14

6. Importing an Existing GNU Project

1) Right-click in the **Project Explorer** view of **C/C++** perspective to select **Migrate to GD Project** wizard.

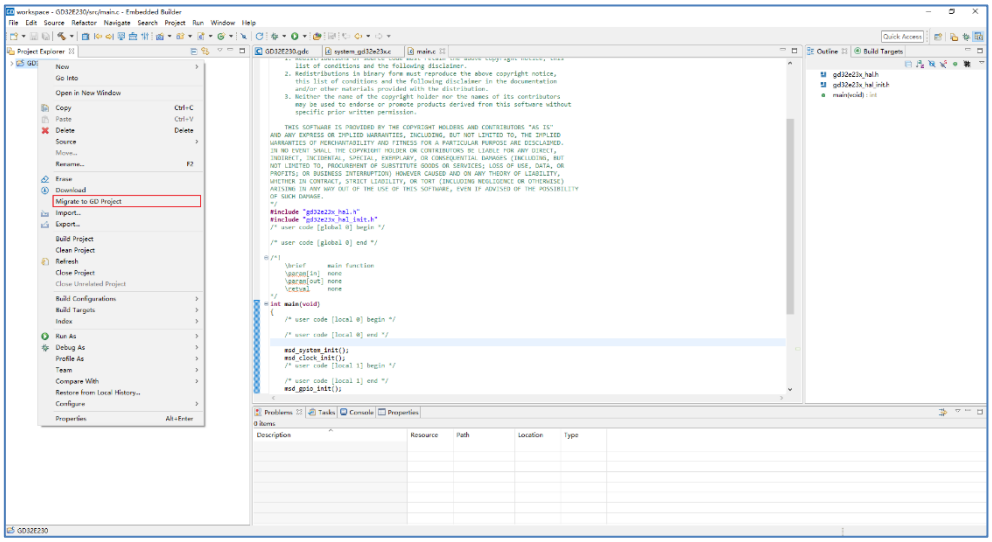

Figure 6-1

2) Select the file with the suffix “cproject” of the GNU project which you want to convert. Click **Finish** button to start the conversion process. If the conversion is successful, you can import and build the GNU project as usual.

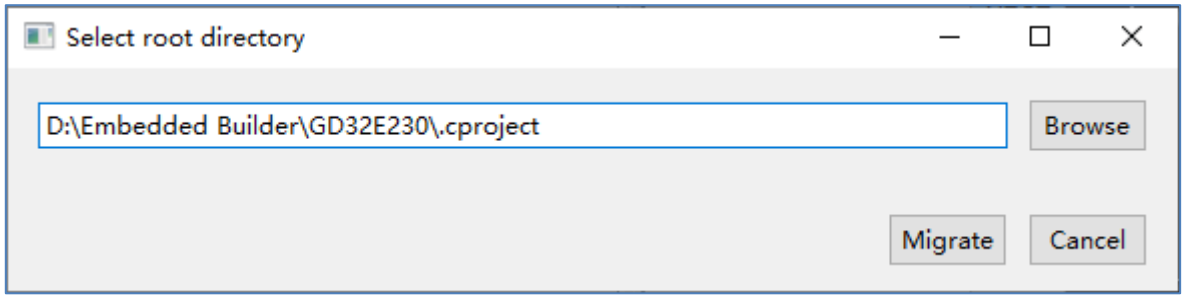

Figure 6-2

7. Q&A

Q1. If the project has compiled with error, like the following, you need to change the .cproject content.

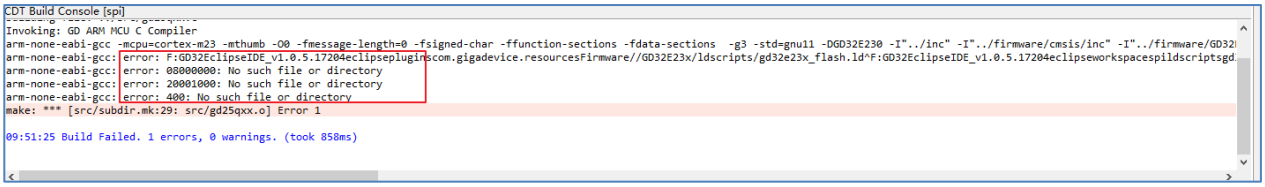

Figure 7-1

Operation flow:

1. Open the .cproject in the project folder.
2. Find and delete the following four options, there is two place, key word is ‘codePosition’.
3. Back to the project in the ide, right click the project name and click ‘Clean Project’, then click ‘Build Project’.

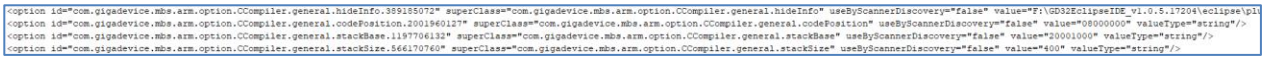

Figure 7-2

8. Tips

1. It is recommended to create a source folder type when creating a folder in a project.