



Gowin RiscV_AE350_SOC RDS Software

User Manual

MUG1025-1.3E, 02/07/2024

Copyright © 2025 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
11/16/2023	1.0E	Initial version published.
01/26/2024	1.1E	● Gowin Cable (dual-channel and single-channel FTDI), Olimex Cable and their driver software Zadig-2.7 supported for software single-step debugging. ● Flash Burner supports Gowin Programmer. ● More application demos supported. ● RT-Thread Standard version software development kit supported. ● More both Chinese and English help documents supported.
05/20/2024	1.2E	● Gowin GoBridgeDriver supported. ● Gowin Programmer software for Flash Burner updated. ● Application demos updated. ● Help documents updated.
02/07/2025	1.3E	Update Flash Burner to support FPGA Fabric and RiscV_AE350_SOC in sharing Gowin FTDI USB programmer and WinUSB driver.

Contents

Contents	i
List of Figures	iv
List of Tables	xi
1 Introduction to RDS Software	1
1.1 Overview	2
1.2 Start Software	4
1.3 Component Revision	4
1.4 Workspace	6
1.5 Language	7
1.6 Quick Start	8
1.7 Help Documents	14
1.8 RTOS Source Code	14
2 Create and Build a Software Project	16
2.1 Create a New Project	16
2.1.1 Create a New Project	16
2.1.2 Import an Existing Software Project	19
2.2 Software Project File	22
2.2.1 Import an Existing File	22
2.2.2 Create Source/Header File	24
2.3 Target Configuration	26
2.4 Create a Build System	31
2.4.1 Specify Build Configuration	31
2.4.2 Set Build Configuration	31
2.5 Build a Software Project	63
2.6 Static Analysis	63

2.6.1 Function Code Size View.....	65
2.6.2 Static Stack Analysis View.....	66
3 Target Management	69
3.1 Create a Target System	70
3.2 Target Manager.....	71
3.3 Reset Target Board.....	75
3.4 Shut Down Target	76
4 Debug a Software Project	78
4.1 Debug Perspective	78
4.2 Debug View	79
4.2.1 Debug View	80
4.2.2 Memory View	81
4.2.3 Memory Browser View.....	87
4.2.4 Memory Map View	91
4.2.5 Cache Dump View	93
4.2.6 Console View.....	95
4.2.7 GDB Command View.....	95
4.2.8 SoC Registers View.....	98
4.2.9 Registers View.....	100
4.2.10 Variables View	100
4.2.11 Expressions View.....	100
4.2.12 Global Variables Live View	103
4.2.13 Breakpoints View	105
4.2.14 Disassembly View.....	105
4.3 Debugging Session.....	105
4.3.1 Create Debug Configurations	106
4.3.2 Debugging Session Settings	109
4.4 Breakpoints and Watchpoints	113
4.4.1 Set Breakpoints	113
4.4.2 Set Watchpoints.....	117
5 Download Software Project	122
6 Generic Exception Handling and Stack Recording/Protection.....	126

6.1 Stack Recording/Protection	126
6.2 Generic Exception Handling	128
7 Profile Software Project	132
7.1 Profile Perspective	132
7.2 Target Board Profiling View	133
7.3 Profile Session	135
8 ICEman Software	143

List of Figures

Figure 1-1 RDS Structure	1
Figure 1-2 C/C++ Perspective	2
Figure 1-3 Select Window > Perspective > Open Perspective	3
Figure 1-4 Perspective Toolbar	3
Figure 1-5 Select Window > Show View	4
Figure 1-6 Select Help > About AndeSight_RDS v511	5
Figure 1-7 About RDS	5
Figure 1-8 Component Revision Information	6
Figure 1-9 RDS Launcher	6
Figure 1-10 Select Window > Preferences > Language	7
Figure 1-11 Change the Language	8
Figure 1-12 RDS Launcher	9
Figure 1-13 Project Creator View.....	9
Figure 1-14 Create a C Project.....	10
Figure 1-15 Build a New Project	11
Figure 1-16 Compilation Results in the Console View.....	11
Figure 1-17 Debug a New Project.....	12
Figure 1-18 Debug View	12
Figure 1-19 Download Software Project	13
Figure 1-20 Cable Setting	13
Figure 1-21 Select Help > Documents Window	14
Figure 1-22 Documents Window.....	14
Figure 2-1 Project Creator View.....	17
Figure 2-2 Create a C Project.....	18
Figure 2-3 Select Platforms and Configurations	19

Figure 2-4 Select File > Import	20
Figure 2-5 Select General > Existing Projects into Workspace	20
Figure 2-6 Import Projects	21
Figure 2-7 Select Copy projects into workspace	22
Figure 2-8 Select Import.....	23
Figure 2-9 Select General > File System.....	23
Figure 2-10 Select the Imported File.....	24
Figure 2-11 Select New > Source File or Header File	25
Figure 2-12 Create New Source/Header File	25
Figure 2-13 Select Target Configuration	26
Figure 2-14 Select Chip Profile	27
Figure 2-15 Select Targets and Toolchain.....	27
Figure 2-16 Select Target Settings.....	28
Figure 2-17 Select Connection Configuration (Andes AICE-MINI+) Debugger	29
Figure 2-18 Select Gowin FTDI USB Programmer.....	29
Figure 2-19 Select Olimex Debugger	30
Figure 2-20 Select Clean and Rebuild Project.....	30
Figure 2-21 Select Build Configurations > Set Active	31
Figure 2-22 Select Build Settings.....	32
Figure 2-23 Tool Settings Tab	32
Figure 2-24 Select Andes C Compiler > Directories	33
Figure 2-25 Select Andes C Compiler > Optimization	34
Figure 2-26 Select Andes C Compiler > Debugging	37
Figure 2-27 Select Andes C Compiler > Miscellaneous	38
Figure 2-28 Select Andes C Linker > General	44
Figure 2-29 Select Andes C Linker > Libraries	45
Figure 2-30 Select Andes C Linker > Miscellaneous	46
Figure 2-31 Select Andes Assembler > General.....	54
Figure 2-32 Select NM (symbol listing) > General	57
Figure 2-33 Select Readelf (ELF info listing) > General	58
Figure 2-34 Select Objdump (disassembly) > General.....	59
Figure 2-35 Select Objcopy (object content copy) > General	60

Figure 2-36 Select Size (section size listing) > General	61
Figure 2-37 Select LdSaG Tool > General	62
Figure 2-38 Build a Software Project	63
Figure 2-39 Static Analysis Setting	64
Figure 2-40 Enable Static Analysis	65
Figure 2-41 Function Code Size View	65
Figure 2-42 Locate Function Code	66
Figure 2-43 Static Stack Analysis View.....	67
Figure 2-44 Expand All.....	67
Figure 2-45 Export to CSV File	68
Figure 2-46 Locate Function Code	68
Figure 3-1 Target Management Flow	70
Figure 3-2 Connect the RDS Software and the Target Board.....	70
Figure 3-3 Target Manager	71
Figure 3-4 Diagnose the Target System	71
Figure 3-5 Test Memory Access (Gowin FTDI USB Programmer)	72
Figure 3-6 Test Memory Access (Olimex Debugger)	72
Figure 3-7 Create a Target System.....	74
Figure 3-8 Start Target System	75
Figure 3-9 Reset Target Board.....	76
Figure 3-10 Reset Target Board.....	76
Figure 3-11 Shut Down Target	77
Figure 4-1 Debug Perspective	78
Figure 4-2 Select Window > Perspective > Open Perspective > Debug	79
Figure 4-3 Select Window > Show View > Other... ..	79
Figure 4-4 Select Debug View	80
Figure 4-5 Debug View	80
Figure 4-6 Memory View	81
Figure 4-7 Select a Thread or Stack	82
Figure 4-8 Add Memory Monitor Interface	82
Figure 4-9 Add Memory Monitor	82
Figure 4-10 Traditional Memory Rendering	83

Figure 4-11 Create a Floating Point Memory Rendering	83
Figure 4-12 Byte Order Setting	84
Figure 4-13 Contents of Memory	84
Figure 4-14 Automatic Rendering Settings	85
Figure 4-15 Setting the Display.....	85
Figure 4-16 Traditional Memory Rendering Setting	86
Figure 4-17 Import Memory Values.....	86
Figure 4-18 Export Memory Values	87
Figure 4-19 Memory Browser View.....	87
Figure 4-20 Traditional Memory Rendering	88
Figure 4-21 Floating Point Memory Rendering.....	88
Figure 4-22 New Tab Option	88
Figure 4-23 Byte Order Setting	89
Figure 4-24 Contents of Memory	89
Figure 4-25 Select Find/Replace.....	90
Figure 4-26 Find/Replace	90
Figure 4-27 Memory Map View.....	91
Figure 4-28 Add/Modify Memory Regions	91
Figure 4-29 Memory Map Region Wizard Dialog Box	92
Figure 4-30 Delete Memory Regions	93
Figure 4-31 Cache Dump View.....	93
Figure 4-32 Cell Size Setting	94
Figure 4-33 Check Data Consistency with Memory.....	94
Figure 4-34 Data and Memory Matches	94
Figure 4-35 Copy Cache Line	95
Figure 4-36 Console View.....	95
Figure 4-37 GDB Command View	96
Figure 4-38 GDB Command Part.....	96
Figure 4-39 GDB Command Console	97
Figure 4-40 GDB Command Input	97
Figure 4-41 Export GDB Command.....	98
Figure 4-42 SoC Registers View.....	98

Figure 4-43 Modify Register Values	99
Figure 4-44 Watch SoC Registers	99
Figure 4-45 Registers View	100
Figure 4-46 Variables View	100
Figure 4-47 Expressions View	101
Figure 4-48 Add Global Variables	102
Figure 4-49 Export a Global Variable	102
Figure 4-50 Global Variables Live View	103
Figure 4-51 Add Global Variables	104
Figure 4-52 Select Display Format	104
Figure 4-53 Breakpoints View	105
Figure 4-54 Disassembly View	105
Figure 4-55 Select Debug As > Debug	106
Figure 4-56 Select Run > Debug Configurations... ..	107
Figure 4-57 Create New Debugging Configuration.....	108
Figure 4-58 Debugging Configurations	108
Figure 4-59 Main Tab	109
Figure 4-60 Debugger Tab	110
Figure 4-61 Startup Tab	110
Figure 4-62 Source Tab.....	111
Figure 4-63 Common Tab	112
Figure 4-64 Exception Handling Tab.....	113
Figure 4-65 Select Breakpoint Types	114
Figure 4-66 Select Toggle Breakpoint.....	114
Figure 4-67 Breakpoints.....	115
Figure 4-68 Select Toggle Breakpoint in Outline View.....	115
Figure 4-69 Select Breakpoint Properties... in Breakpoints View	116
Figure 4-70 Select Breakpoint Properties... in Code Editor	116
Figure 4-71 Properties for C/C++ Line Breakpoint	117
Figure 4-72 Select Run > Toggle Watchpoint	118
Figure 4-73 Select Toggle Watchpoint in Outline View	118
Figure 4-74 Select Add Watchpoint (C/C++)... in Memory View	119

Figure 4-75 Select Add Watchpoint (C/C++)... in Breakpoint View	119
Figure 4-76 Properties for C/C++ Watchpoint.....	120
Figure 4-77 Watchpoints in Breakpoints View	121
Figure 4-78 Remove Watchpoints in Breakpoints View.....	121
Figure 5-1 Launch Flash Burner	123
Figure 5-2 Launch Gowin Programmer	123
Figure 5-3 Cable Setting	124
Figure 5-4 Device Configuration	124
Figure 5-5 Reload	125
Figure 6-1 Select Hardware Stack Recording	127
Figure 6-2 Set the Top of Stack Boundary	127
Figure 6-3 Display Stack Boundary	128
Figure 6-4 Stack Overflow	128
Figure 6-5 Select Break on General Exceptions.....	129
Figure 6-6 Specify Exceptions	129
Figure 6-7 Select Hardware Stack Protection.....	130
Figure 6-8 Display the Stack Protection Status	131
Figure 7-1 Profile Perspective.....	132
Figure 7-2 Select Window > Perspective > Open Perspective > Profile	133
Figure 7-3 Target Board Profiling View	133
Figure 7-4 Select Windows > Show View > Other... ..	134
Figure 7-5 Select Profile > Target Board Profiling.....	134
Figure 7-6 Profile As > Profile Configurations... ..	136
Figure 7-7 Create Profile Configurations	136
Figure 7-8 Select Enable manual profiling.....	137
Figure 7-9 Select User-specified region	137
Figure 7-10 Debug View	138
Figure 7-11 Analysis Data in Target Board Profiling View.....	138
Figure 7-12 Enable Manual Profiling.....	139
Figure 7-13 Locate Source Code.....	139
Figure 7-14 Set Breakpoints	140
Figure 7-15 Select Create chart... ..	140

Figure 7-16 Create chart from selection.....	141
Figure 7-17 Chart View	141
Figure 7-18 Select Instruction Stepping Mode.....	142
Figure 7-19 Disassembly View	142

List of Tables

Table 1-1 RTOS Source Code	15
Table 2-1 Configuration Options	17
Table 2-2 Configuration Options	18
Table 2-3 Optimization Options	34
Table 2-4 Code Model	35
Table 2-5 Optimization Options	35
Table 2-6 Miscellaneous Options	38
Table 2-7 GCC Compiler Options	39
Table 2-8 LLVM Compiler Options	41
Table 2-9 General Options	44
Table 2-10 Libraries Options	45
Table 2-11 Linker Options	46
Table 2-12 Assembler Options	54
Table 2-13 General Option of NM (symbol listing)	57
Table 2-14 General Option of Readelf (ELF info listing)	58
Table 2-15 General Option of Objdump (disassembly)	59
Table 2-16 General Options of Objcopy (object content copy)	60
Table 2-17 General Option of Size (section size listing)	61
Table 2-18 General Options of LdSaG Tool	62
Table 4-1 Memory Mapping Region Options	92
Table 4-2 Debugging Configuration Type	107
Table 4-3 Main Options	109
Table 4-4 Startup Options	111
Table 4-5 Common Options	112
Table 4-6 Watchpoints Options	120

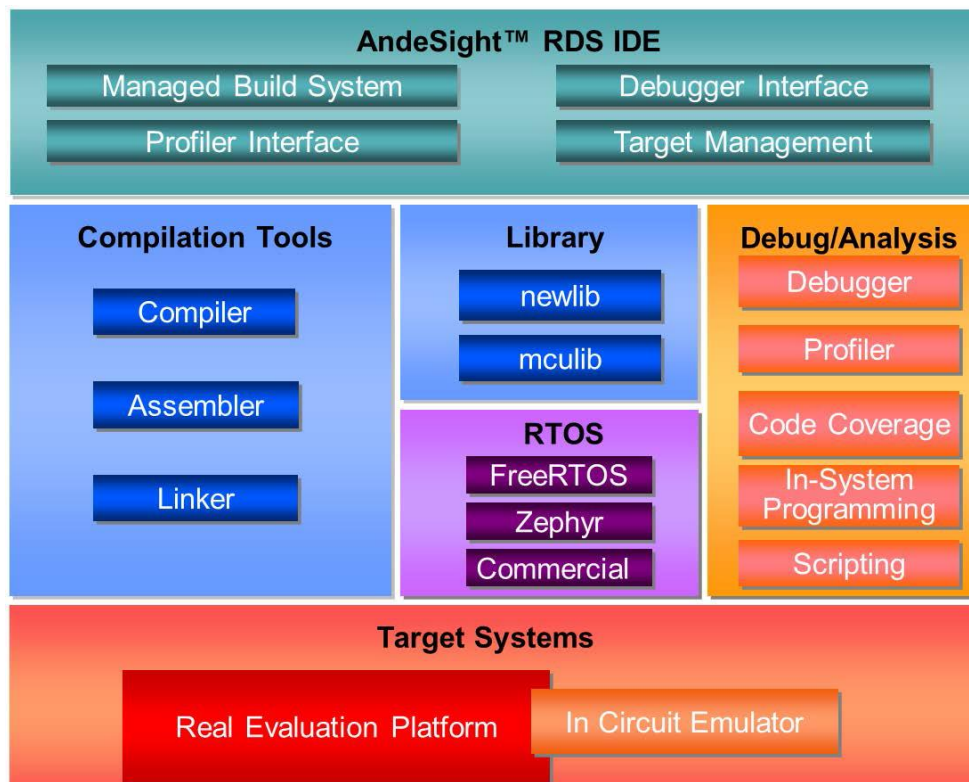
Table 7-1 Columns of the Target Board Profiling View	135
Table 8-1 ICEman Options.....	143

1 Introduction to RDS Software

RDS software is a software developed by Gowin Semiconductor based on [Andes Technology](#)'s Integrated Development Kit, used for creating, building, debugging, downloading and managing Gowin RiscV_AE350_SOC application programs. RDS software includes a variety of customized components such as compilers, toolchains, ICE (In Circuit Simulator) software and Andes AICE debugger drivers, Flash Burner (Gowin Programmer), RTOS source code, help documentation, etc. It can help users to run, edit, debug, download, and analyze application programs on Gowin RiscV_AE350_SOC system.

The main components and functions of the RDS software are as shown in Figure 1-1.

Figure 1-1 RDS Structure

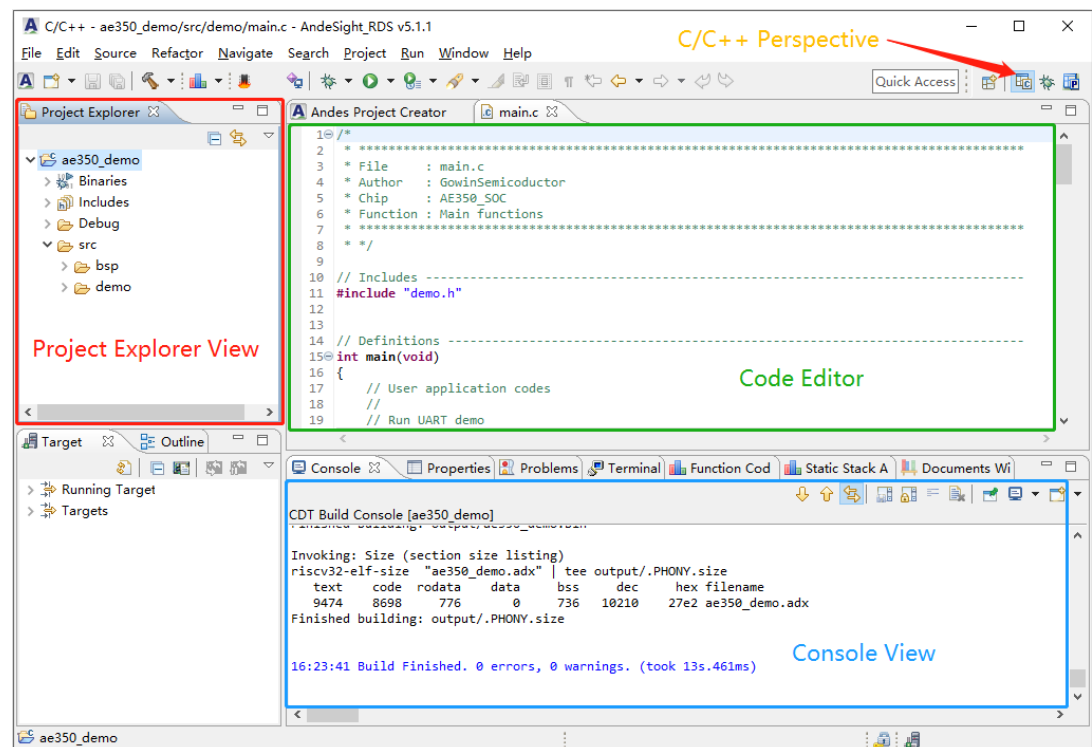


1.1 Overview

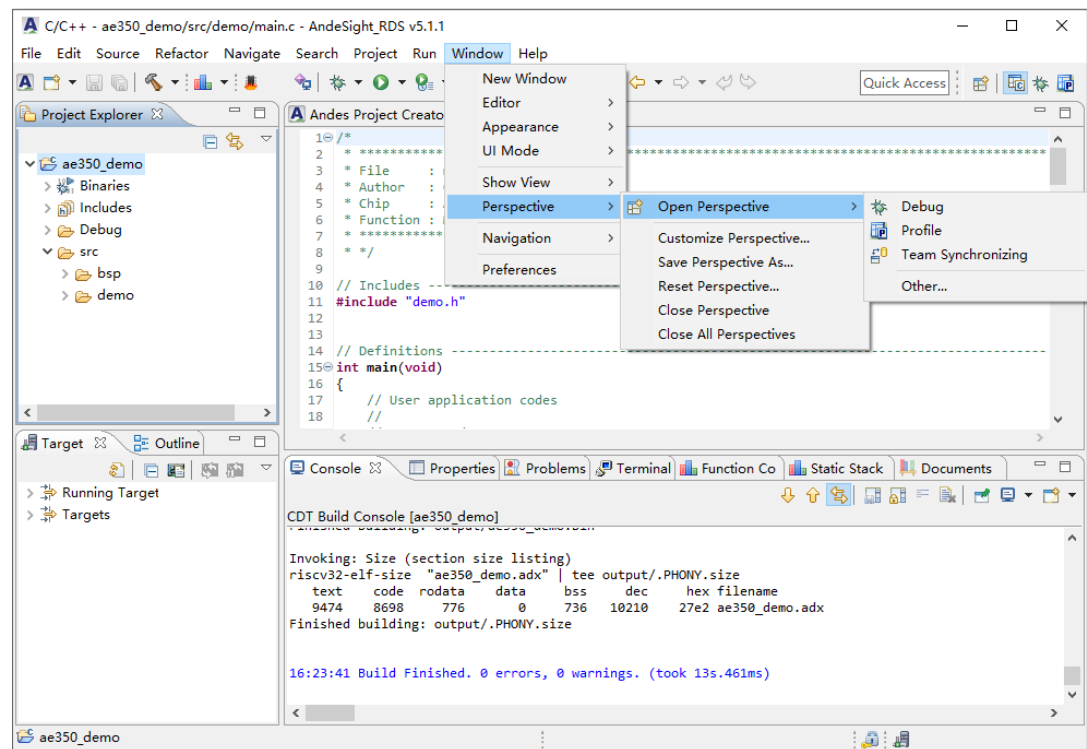
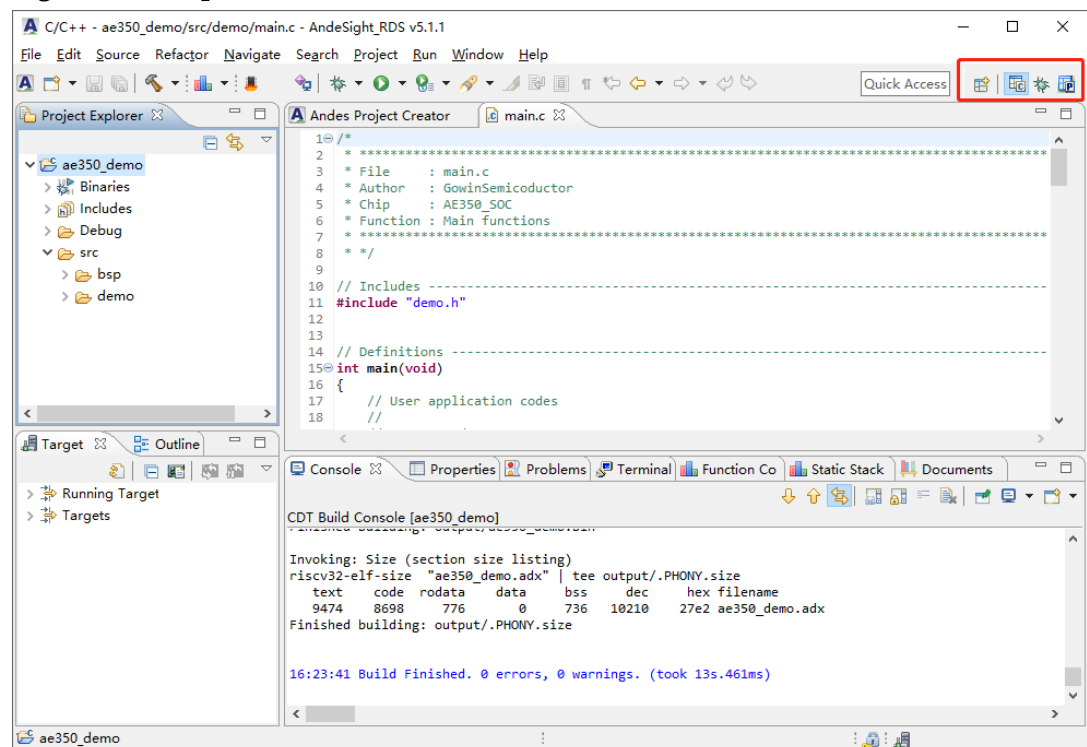
The workspace of the RDS software consists of a combination of panes called "views", which are grouped together to form a functionally oriented "perspective".

The C/C++ Perspective, as shown in Figure 1-2, consists of the Project Explorer View, Code Editor, and Console View, which allow you to navigate and locate project files, edit code, and view standard input/output results.

Figure 1-2 C/C++ Perspective

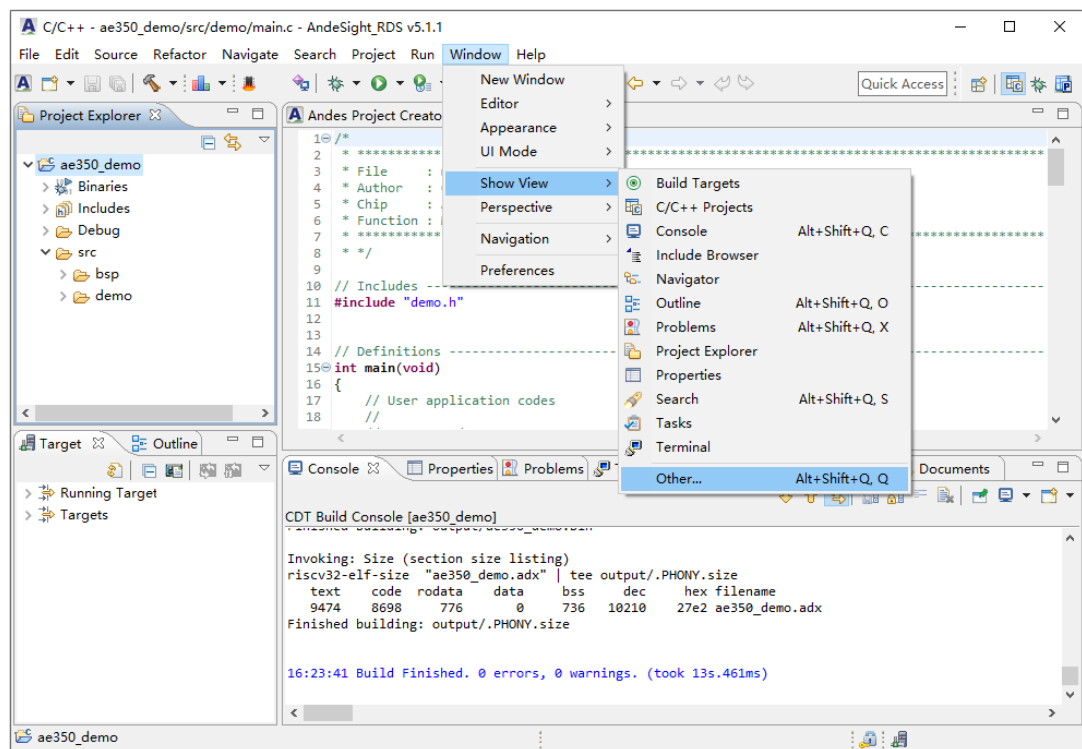


The RDS software also defines Debug Perspective and Profile Perspective, which can be accessed through clicking "Window > Perspective > Open Perspective" on the main menu, or the "Window > Perspective > Open Perspective" icon on the toolbar, as shown in Figure 1-3 and Figure 1-4.

Figure 1-3 Select Window > Perspective > Open Perspective**Figure 1-4 Perspective Toolbar**

If the current perspective does not contain some of the desired views, you can open them through clicking "Window > Show View" on the main menu, as shown in Figure 1-5.

Figure 1-5 Select Window > Show View



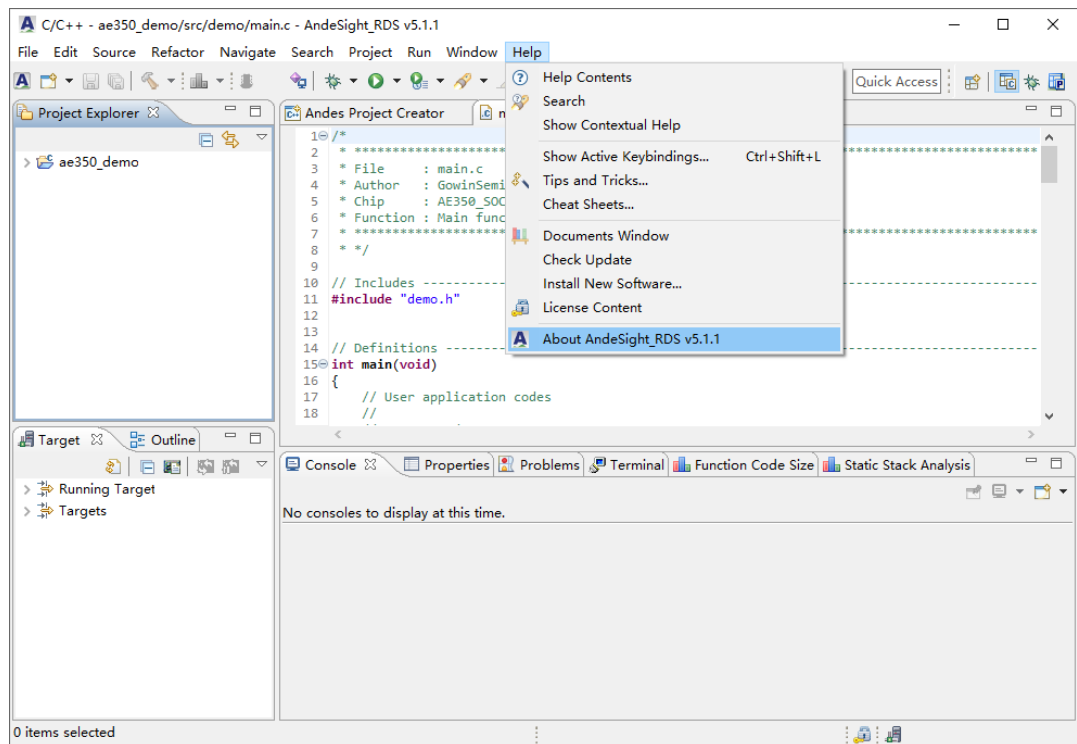
1.2 Start Software

After installing the RDS software on a local Windows PC, you can double-click the desktop shortcut "A" or the installation directory "AndeSight_RDS_v511\ide\AndeSight.exe" to start the RDS software.

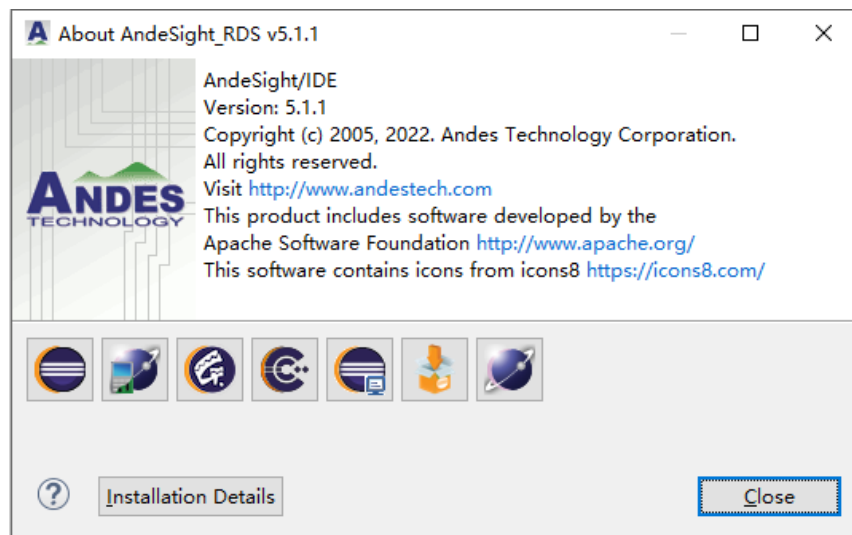
1.3 Component Revision

Refer to the following steps to view the installed components of the RDS software:

1. Select "Help > About AndeSight_RDS v511" on the RDS software main menu, as shown in Figure 1-6.

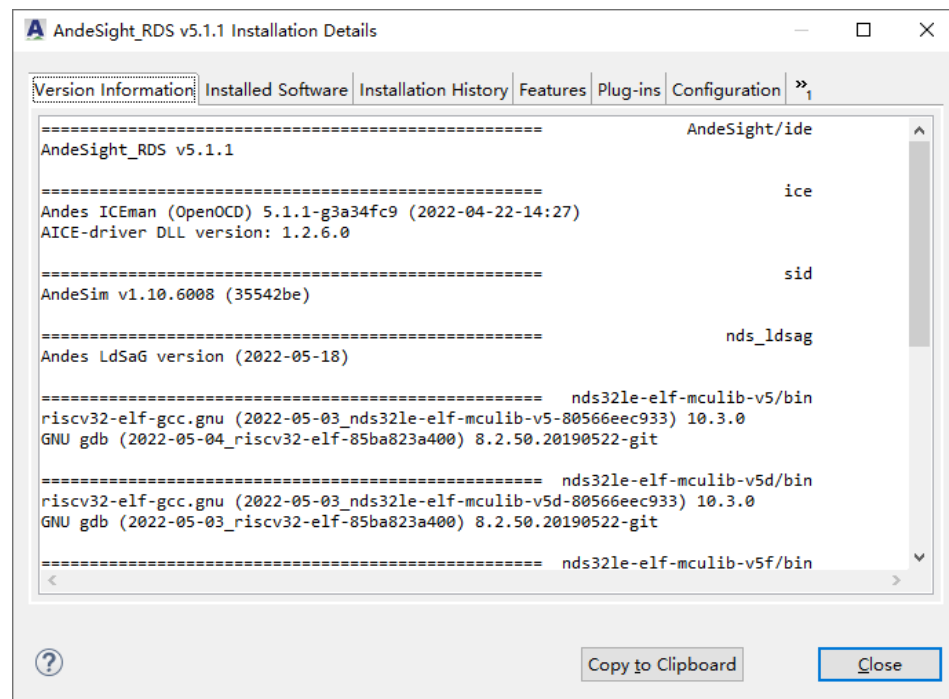
Figure 1-6 Select Help > About AndeSight_RDS v511

2. In the "About AndeSight_RDS v5.1.1" dialog box, view the RDS software version you are currently using, as shown in Figure 1-7.

Figure 1-7 About RDS

3. Click "Installation Details"; and in the "AndeSight_RDS v5.1.1 Installation Details" dialog box, the "Version Information" list displays the version information of the installed components of the RDS software, as shown in Figure 1-8.

Figure 1-8 Component Revision Information

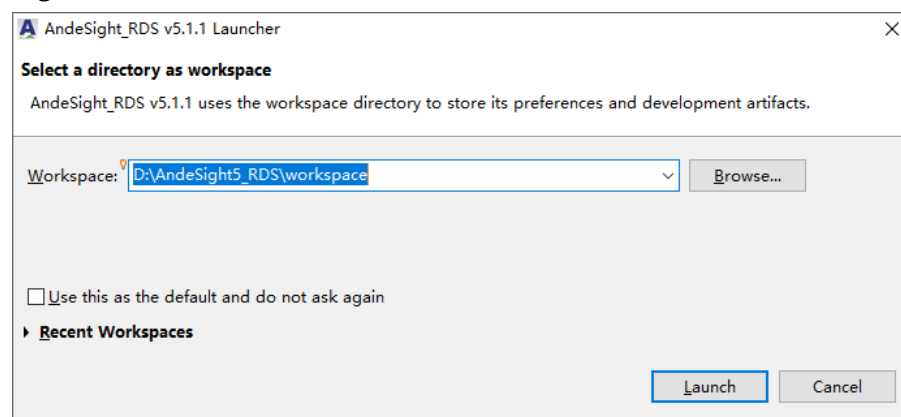


1.4 Workspace

Workspace is used to save and manage the software projects of RDS software, the default path is "USER_HOME\AndeSight5\workspace".

When you start the RDS software for the first time, the "AndeSight_RDS v5.1.1 Launcher" dialog box will show the path to the current workspace of the RDS software, as shown in Figure 1-9. If you want to continue to save and manage the software project in this workspace, click "Launch". If you want to change or create another workspace, click "Browse..." to specify the path to the other workspace. Note that the path to the workspace must consist of characters (A-Z, a-z, 0-9), underscore (_), and hyphen (-).

Figure 1-9 RDS Launcher



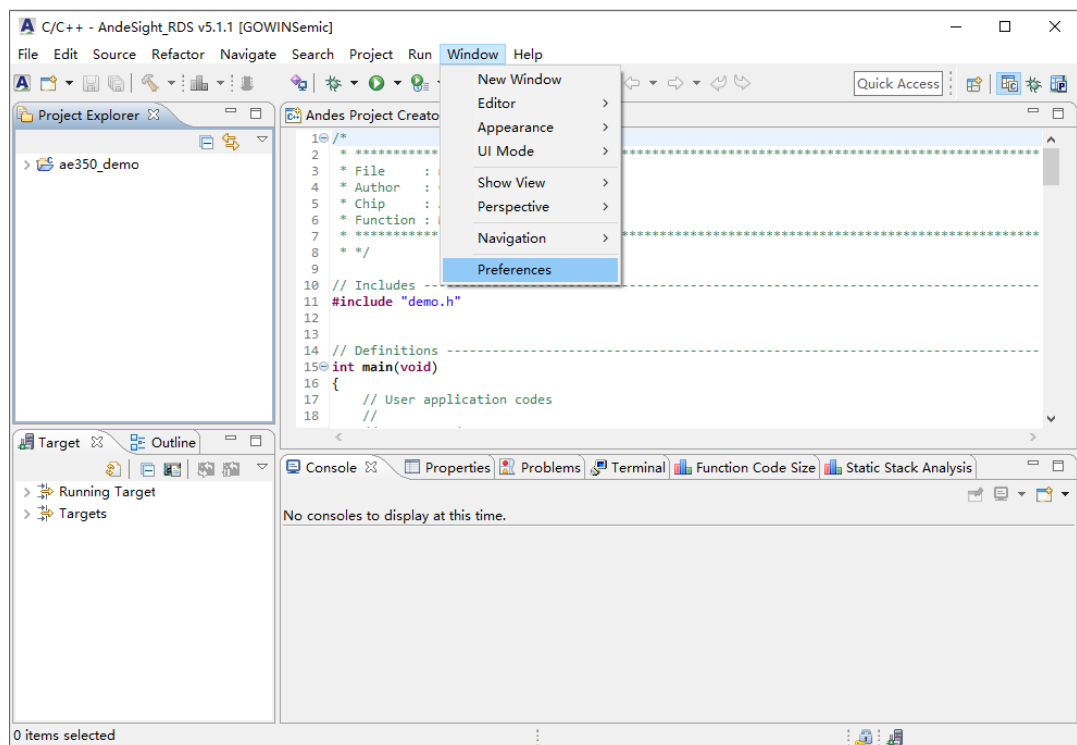
1.5 Language

The RDS software includes 3 languages (English, Japanese and Simplified Chinese) for displaying the software's menus, wizards, dialog boxes, other texts, etc.

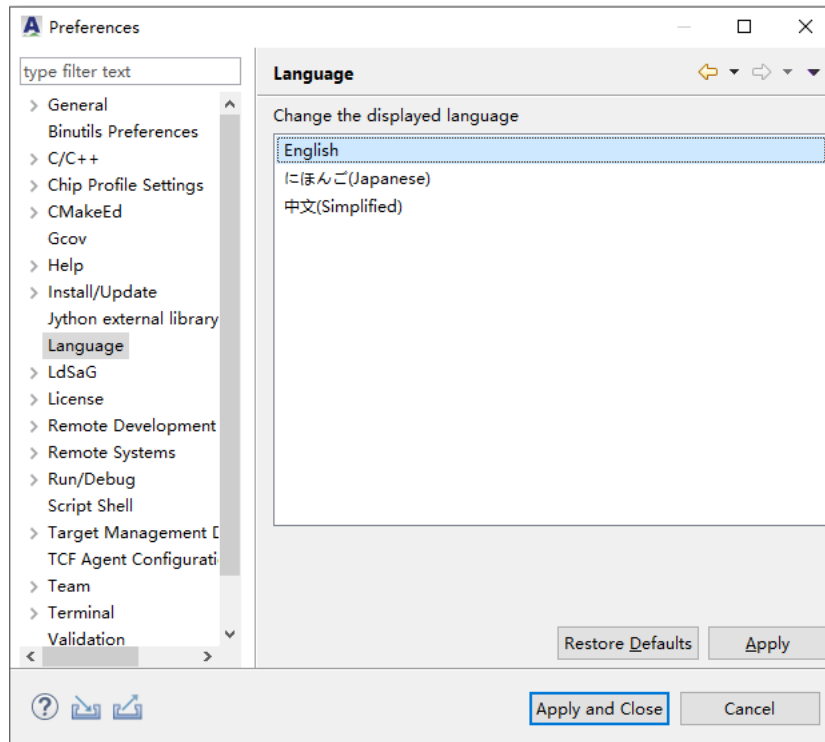
Refer to the following steps to change the language of the software:

1. Select "Window > Preferences > Language" on the RDS main menu, as shown in Figure 1-10.

Figure 1-10 Select Window > Preferences > Language



2. Select a display language and click "Apply". In the "Change the displayed language" dialog box, click "Yes" to restart the RDS software, as shown in Figure 1-11.

Figure 1-11 Change the Language

3. After the RDS software is restarted, the software is displayed in the changed language.

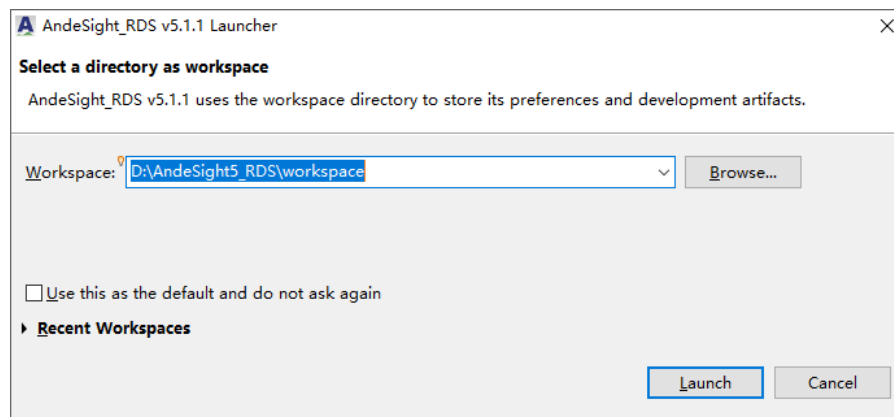
1.6 Quick Start

This section provides quick start method to help users to quickly learn the RDS software to create/build/debug application programs.

Refer to the following steps to learn the application flow of RDS software:

1. Double click the desktop shortcut "A" or the RDS software installation directory "AndeSight_RDS_v511\ide\ AndeSight.exe" to start the RDS software.
2. In the "AndeSight_RDS v5.1.1 Launcher" dialog box, the default path of the workspace is "USER_HOME\AndeSight5\workspace"; click "Launch" to continue, as shown in Figure 1-12.

Figure 1-12 RDS Launcher



3. In the project creator view (Andes Project Creator), select "Connection Configuration", "Project Language" and "Chip Profile"; double-click the selected "Chip Profile" or click "Create Project..." to create a software project, as shown in Figure 1-13.

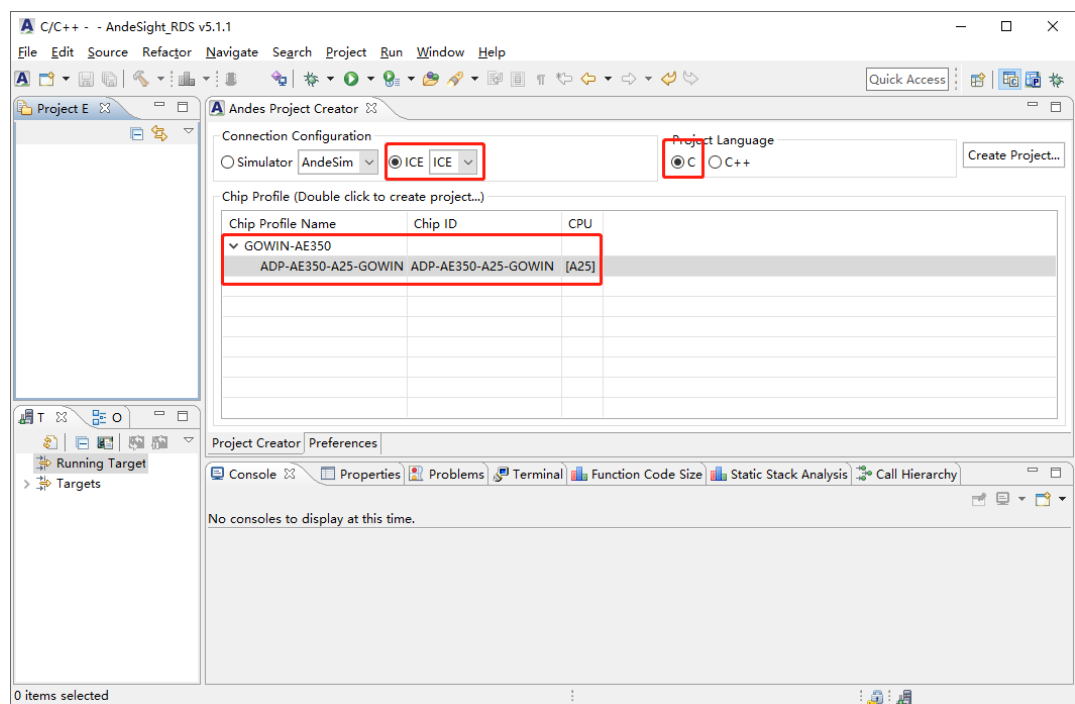
For example:

Connection Configuration: ICE

Project Language: C

Chip Profile: GOWIN-AE350 (ADP-AE350-A25-GOWIN)

Figure 1-13 Project Creator View



4. In the "C Project" dialog box, specify the Project name, Location, Project type and Toolchains, and click "Next", as shown in Figure 1-14.

For example:

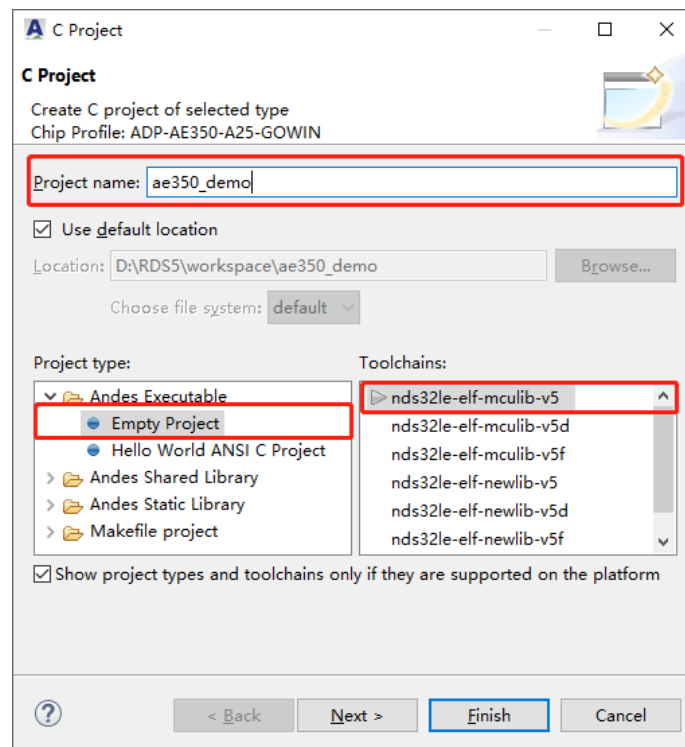
Project name: ae350_demo

Location: D:\RDS\workspace

Project type: Empty Project

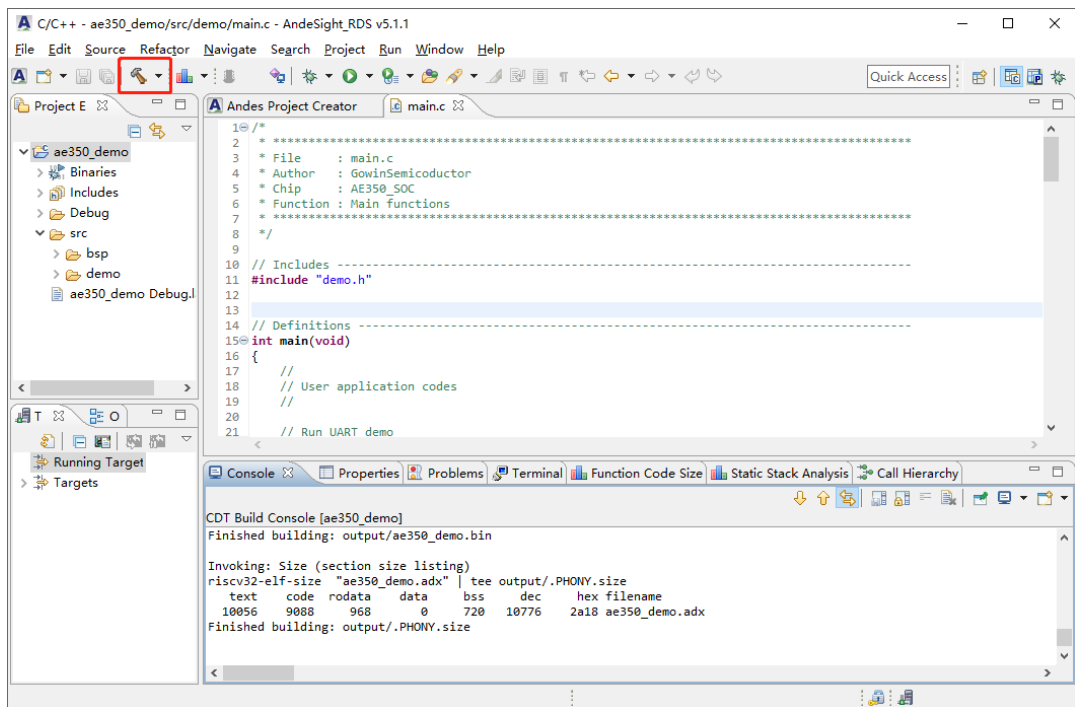
Toolchains: nds32le-elf-mculib-v5

Figure 1-14 Create a C Project



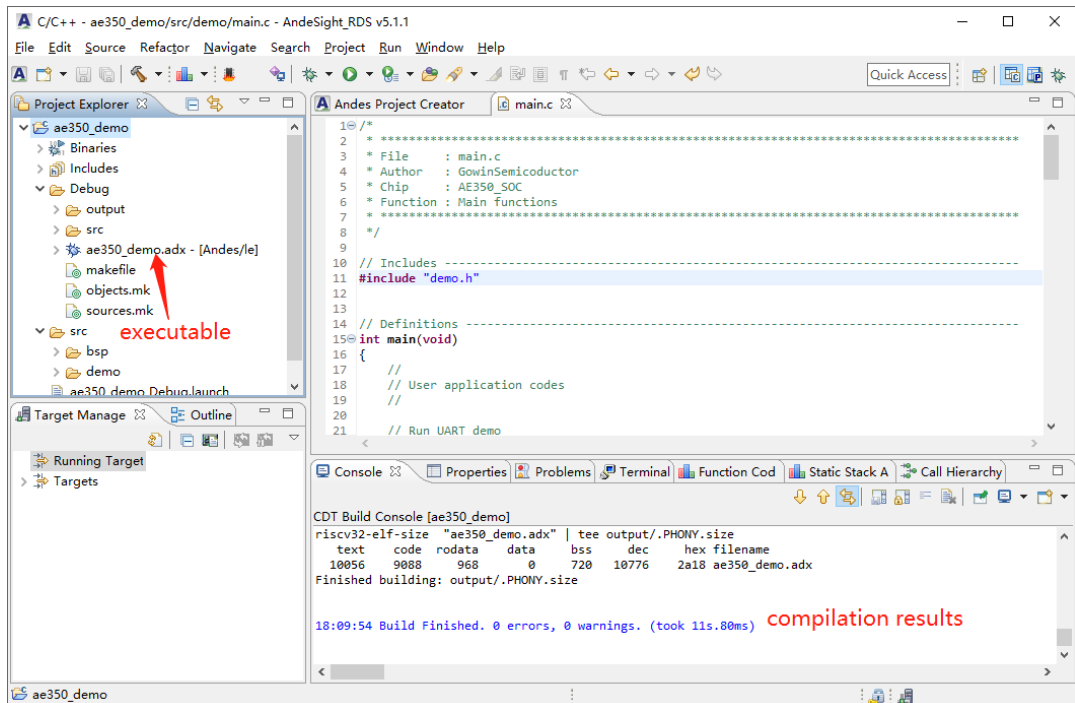
5. In the Project Explorer view, select the created software project and click "🔨" (Build) on the toolbar to build the software project, as shown in Figure 1-15.

Figure 1-15 Build a New Project



6. Check the compilation results in the console view and verify the executable file, as shown in Figure 1-16.

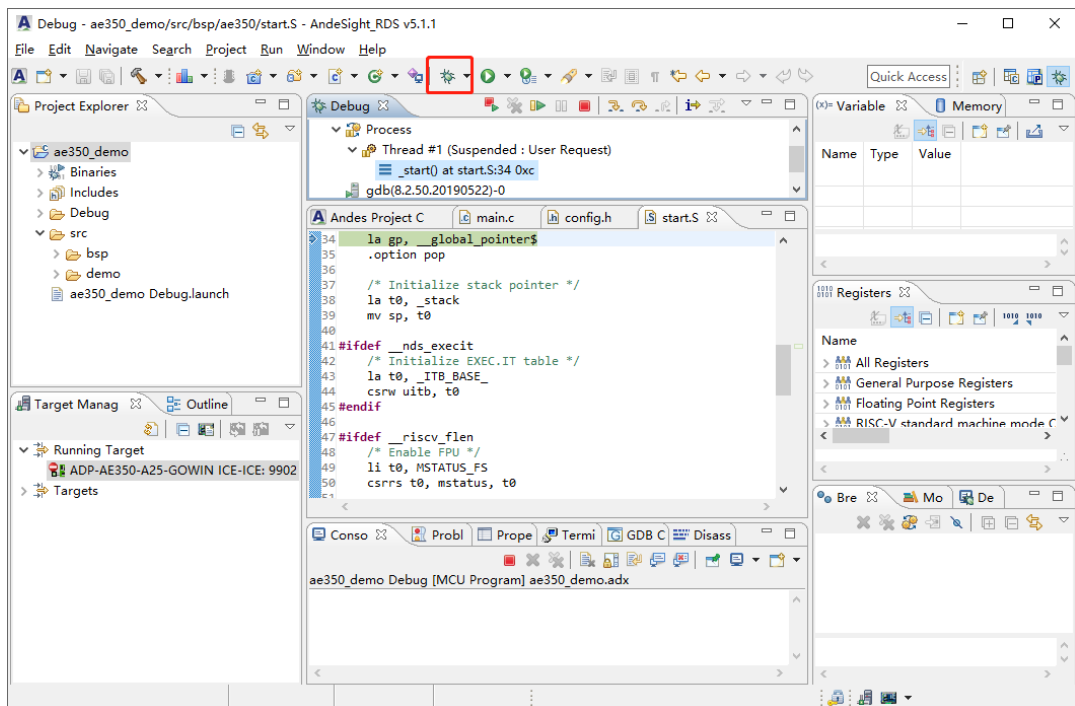
Figure 1-16 Compilation Results in the Console View



7. Insert Gowin FTDI USB programmer into the PC host, and simultaneously connect it to the constrained JTAG interface of the

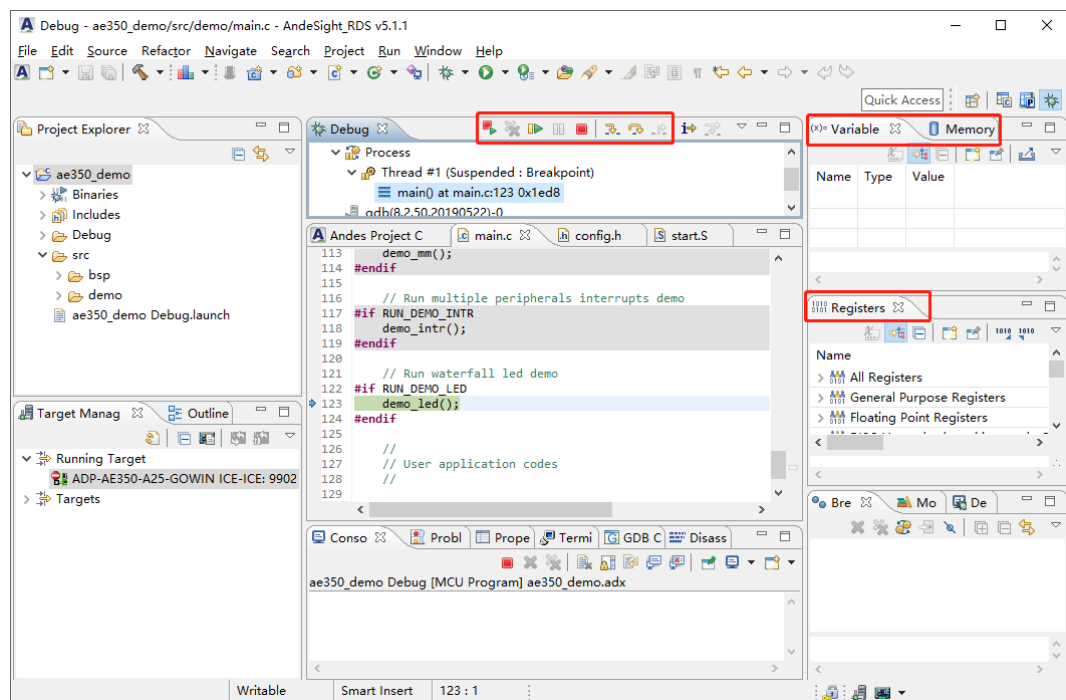
RiscV_AE350_SOC on the development board. Click "🔍" (Debug) on the RDS software toolbar to start debugging, as shown in Figure 1-17.

Figure 1-17 Debug a New Project



8. Enter the debug perspective, and click "▶", "⏏", "🔍" on the debug view toolbar to control debugging thread; you can also view the debugging results through other debugging view (variable view, register view), as shown in Figure 1-18.

Figure 1-18 Debug View



9. Insert the Gowin FTDI USB programmer or GWU2X programmer into the PC host, and simultaneously connect it to the JTAG interface of the FPGA on the development board. Click on the "🔥" (Flash Burner) on the RDS software toolbar to launch Gowin Programmer and download the software programming binary file, as shown in Figure 1-19. If using Gowin FTDI USB programmer and WinUSB driver for the download, you must uncheck "using ft2xx driver" option in the "Cable Setting" dialog box, as shown in Figure 1-20.

Figure 1-19 Download Software Project

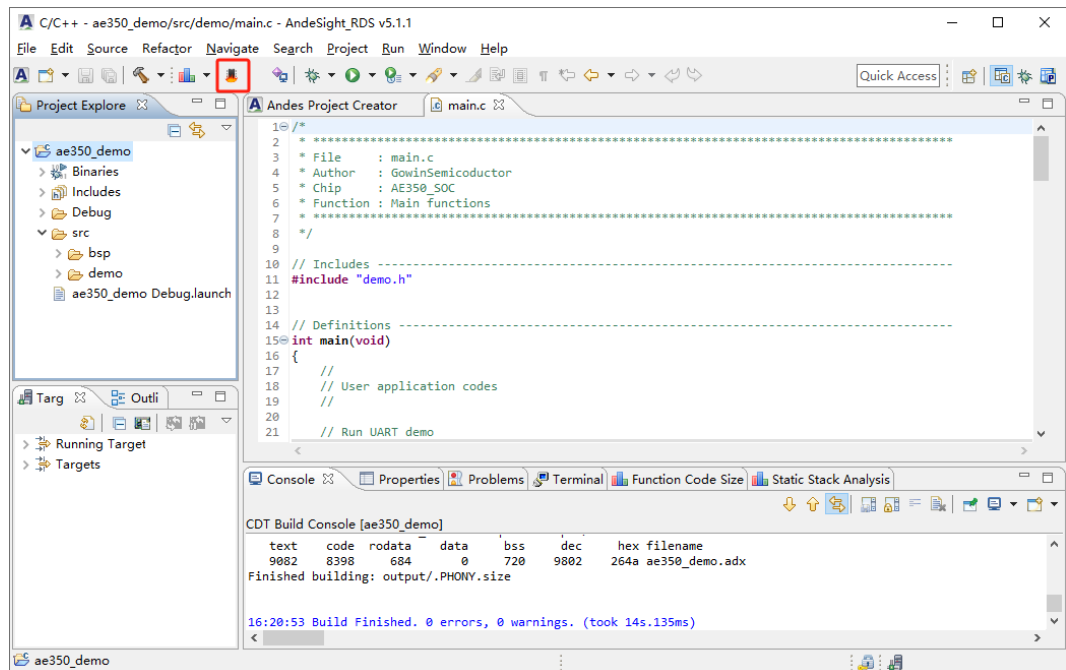
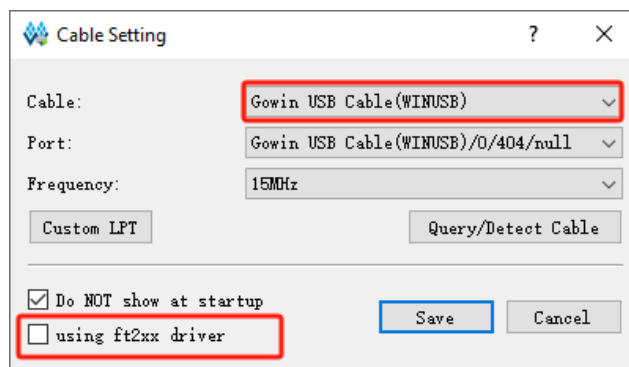


Figure 1-20 Cable Setting

**Note!**

RiscV_AE350_SOC debugger includes Gowin FTDI USB programmer, Andes AICE debugger, Olimex debugger, and others. Gowin FTDI USB programmer is recommended. It supports the FPGA and CPU in sharing Gowin FTDI USB programmer and WinUSB driver.

1.7 Help Documents

The RDS software user manuals, tool guides, tutorials, specification manuals and other supported manuals can be found in the Document Window View by selecting "Help > Documents Window" on the RDS software main menu, as shown in Figure 1-21.

Figure 1-21 Select Help > Documents Window

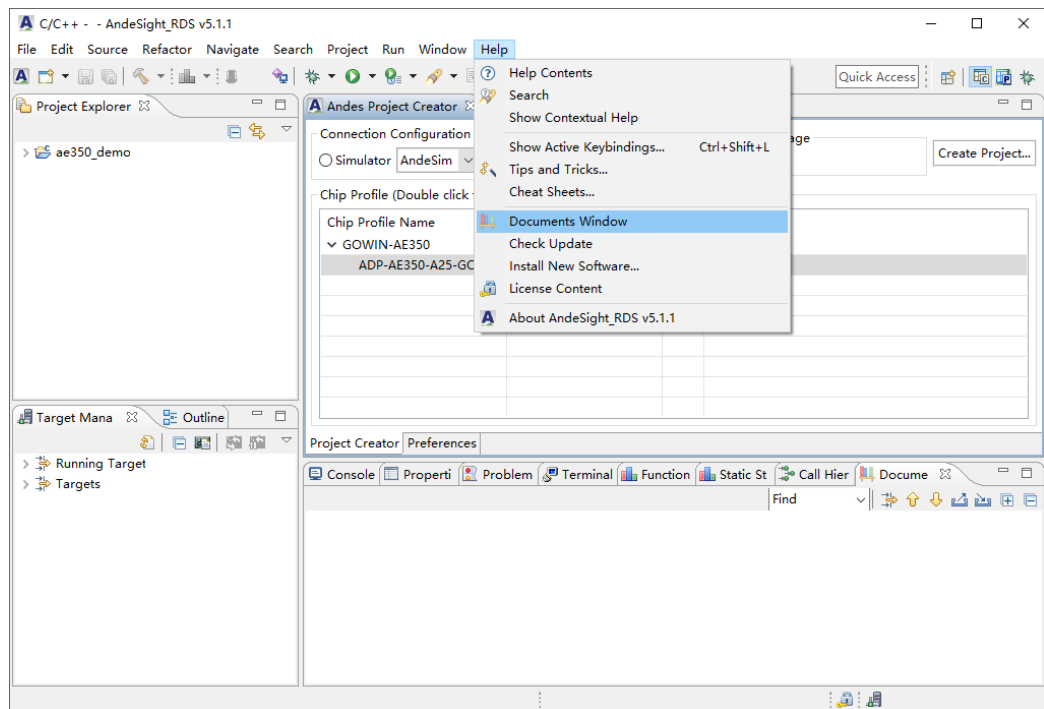
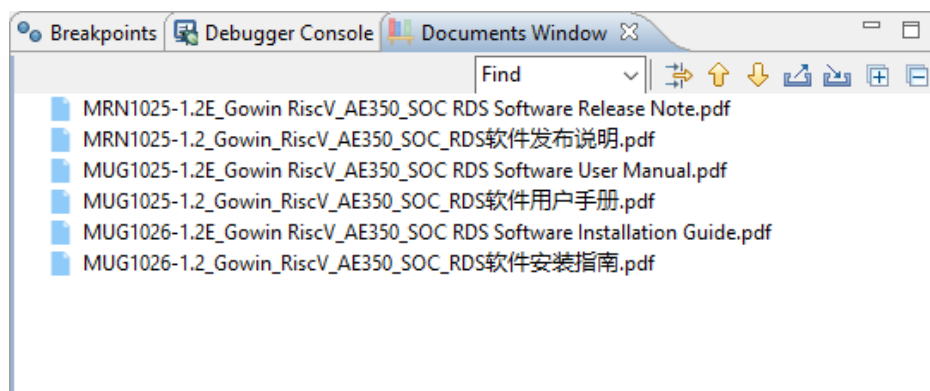


Figure 1-22 Documents Window



1.8 RTOS Source Code

The RDS software installation directory contains the source code of the large RTOS supported by Gowin RiscV_AE350_SOC, located in "AndeSight_RDS_v511\demo", as shown in Table 1-1.

Table 1-1 RTOS Source Code

RTOS Source Code	Description
ae350_rtthread_std	RTOS RT-Thread Standard software development kit
ae350_zephyr	RTOS Zephyr

2 Create and Build a Software Project

You can create a new software project or import an existing software project into the workspace. The process of compiling and linking all the source files of a software project to generate an executable or target file is called building.

2.1 Create a New Project

2.1.1 Create a New Project

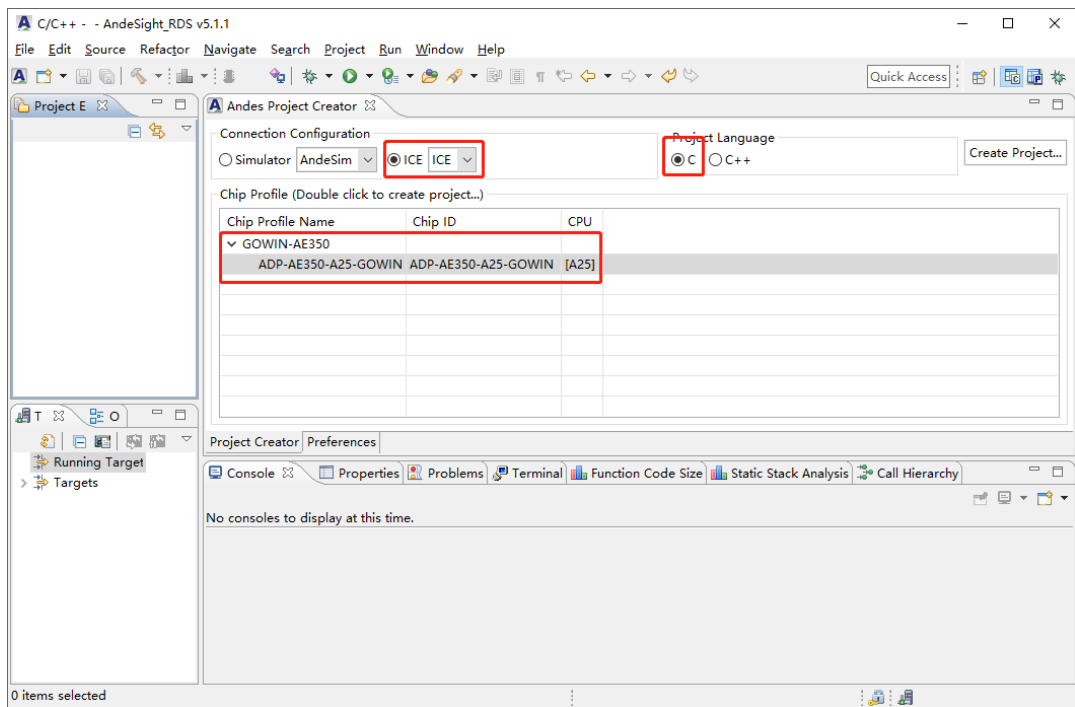
1. After the RDS software starts, locate the Project Creator view or click " (Andes Project Creator) on the RDS software toolbar; select "Connection Configuration", "Project Language" and "Chip Profile", then double-click the selected "Chip Profile" or click "Create Project..." to create a software project, as shown in Figure 2-1.

For example:

Connection Configuration: ICE

Project Language: C

Chip Profile: GOWIN-AE350 (ADP-AE350-A25-GOWIN)

Figure 2-1 Project Creator View

The configuration options in the Project Creation view are as shown in Table 2-1.

Table 2-1 Configuration Options

Option	Configuration	Description
Connection Configuration	Simulator: AndeSim	No connection to target board
	ICE: ICE	Connect the debugger to the target board; The debugger, such as Andes AICE-MINI+, Gowin FTDI USB programmer, or Olimex, etc. Gowin FTDI USB programmer is recommended.
Project Language	C	C programming language
	C++	C++ programming language
Chip Profile	GOWIN-AE350: ADP-AE350-A25-GOWIN	Specify RiscV_AE350_SOC

- In the "C Project" dialog box, specify the Project name, Location, Project type and Toolchains, and click "Next", as shown in Figure 2-2.

For example:

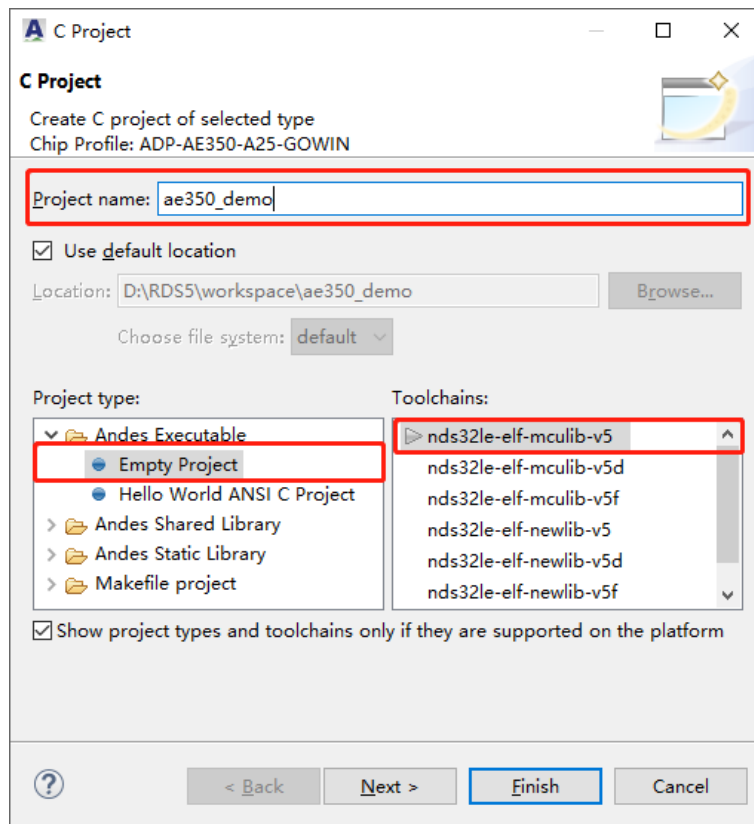
Project name: ae350_demo

Location: D:\RDS\workspace

Project type: Empty Project

Toolchains: nds32le-elf-mculib-v5

Figure 2-2 Create a C Project



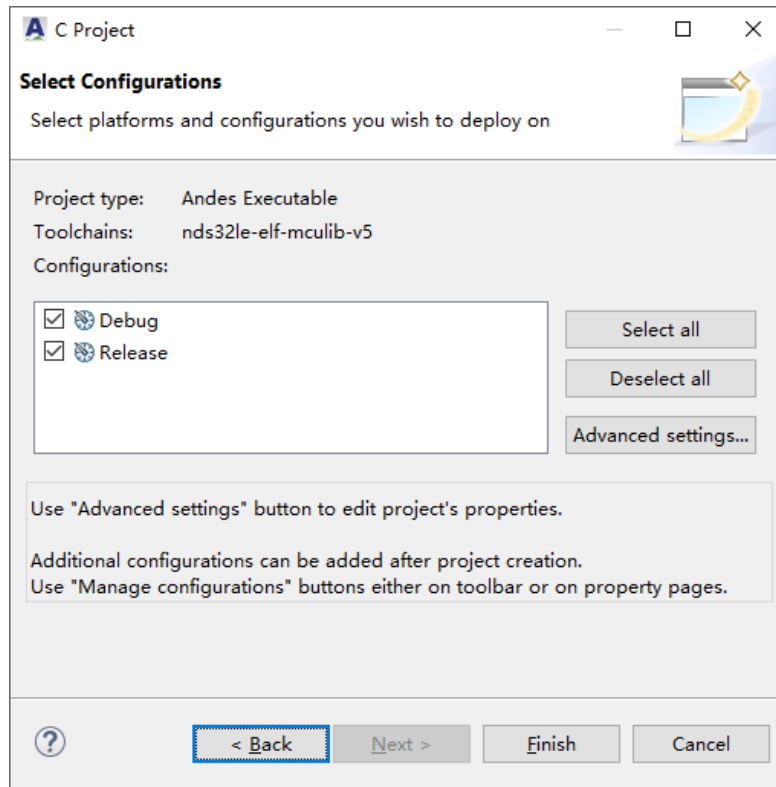
The option configurations in the C Project dialog box are as shown in Table 2-2.

Table 2-2 Configuration Options

Option	Configuration	Description
Project type	Empty Project	Create an empty project
Toolchains	nds32le-elf-mculib-v5 nds32le-elf-mculib-v5d nds32le-elf-mculib-v5f nds32le-elf-newlib-v5 nds32le-elf-newlib-v5d nds32le-elf-newlib-v5f	The RDS software supports the following compilation toolchains: MCULib and Newlib. Newlib is a C software library for embedded systems. MCULib is an optimized and enhanced C software library for MCU applications with a smaller code size than Newlib.
Project name	...	The name of the software project can only contain the following characters: A-Z 0-9 ! @ % ^ [] . - + , ~ _.
Location	Use default location	The software project path

3. Specify the deployment platform and configuration, such as "Debug" or "Release", and click "Finish", as shown in Figure 2-3.

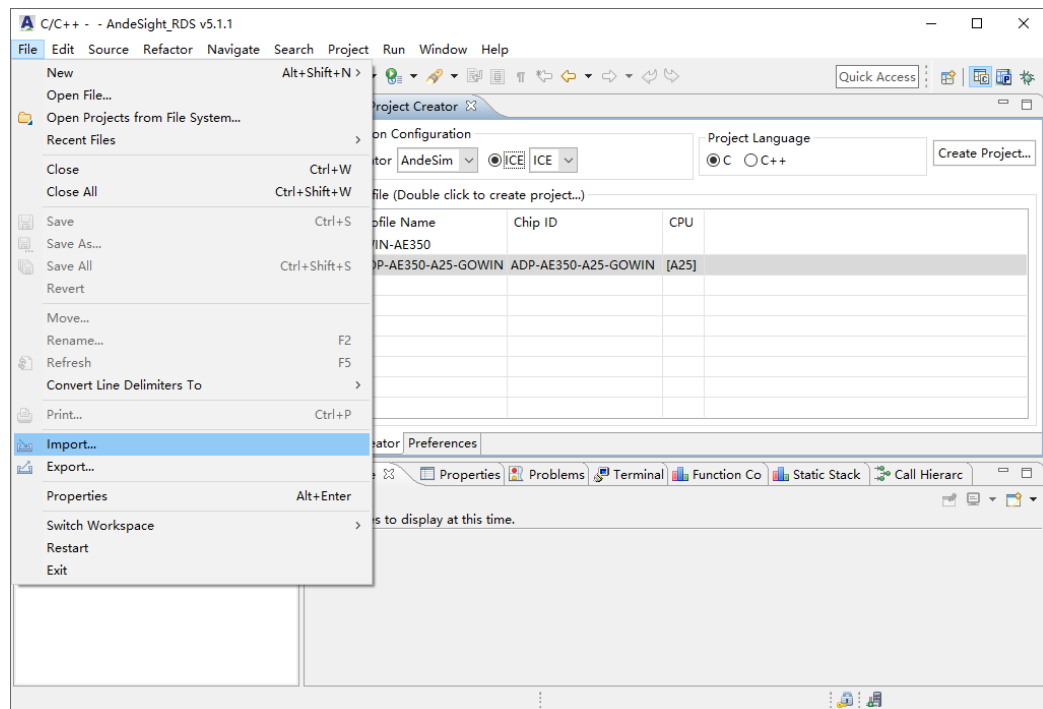
Figure 2-3 Select Platforms and Configurations



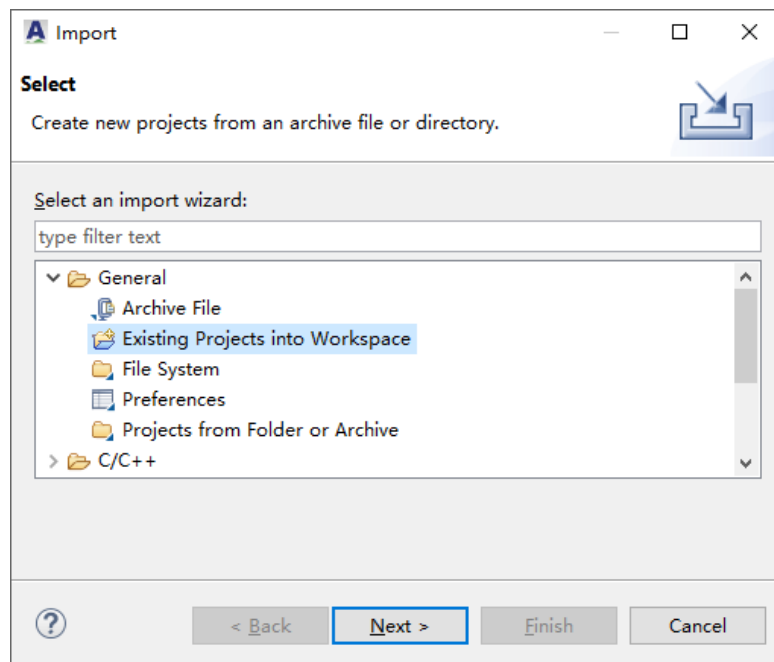
2.1.2 Import an Existing Software Project

Refer to the following steps to import an existing software project into the current workspace:

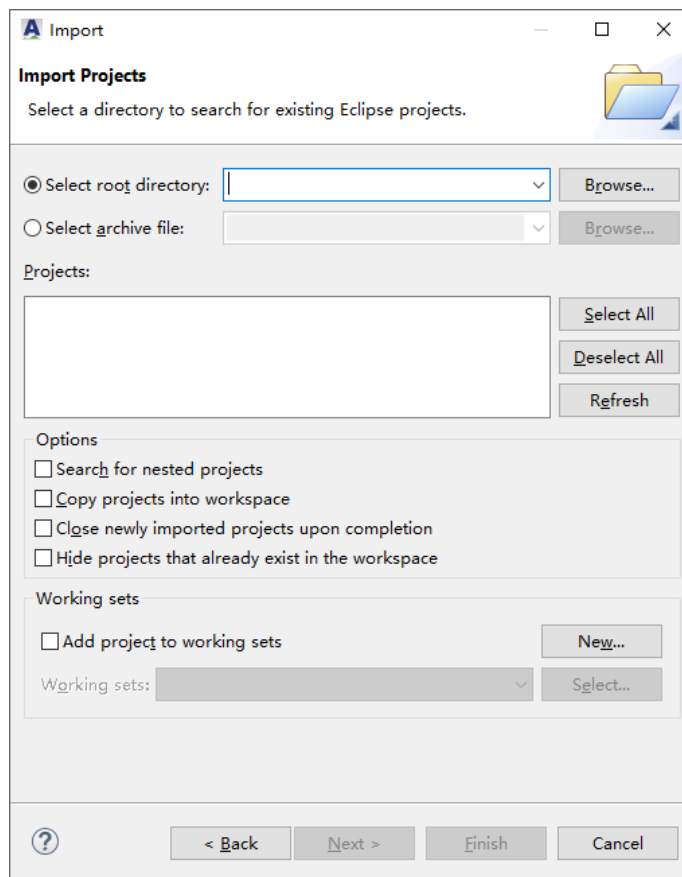
1. Select "File > Import" on the main menu of the RDS software, as shown in Figure 2-4.

Figure 2-4 Select File > Import

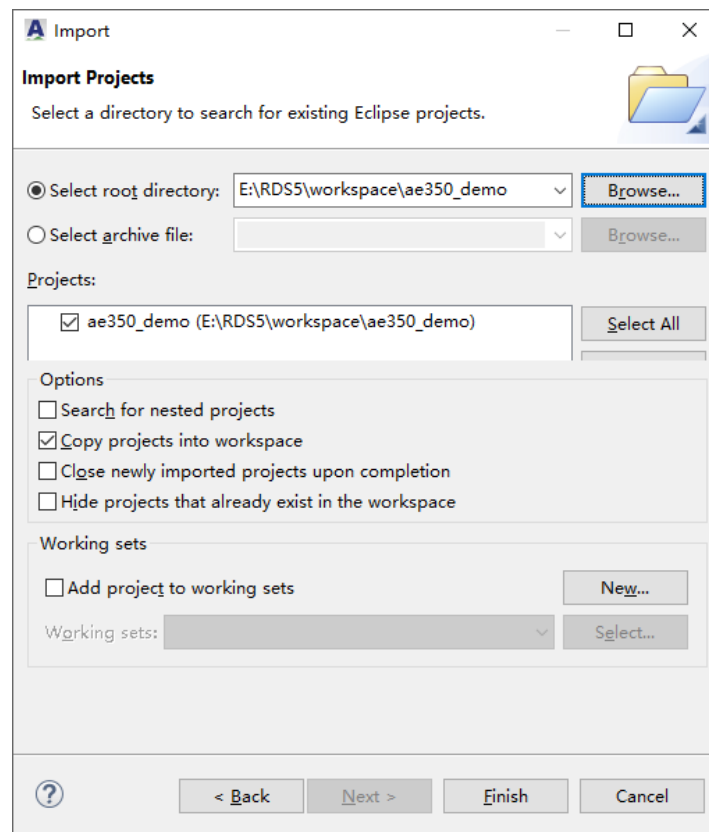
2. In the "Import" dialog box, select "General > Existing Projects into Workspace", and click "Next", as shown in Figure 2-5.

Figure 2-5 Select General > Existing Projects into Workspace

3. Select "Select root directory" to import an existing software project from the file system, or select "Select archive file" to import a compressed file of the software project, and click "Browse...", as shown in Figure 2-6.

Figure 2-6 Import Projects

4. In the "Import Projects" dialog box, select the software project or zip file you want to import, and click "Select Folder" or "Open" to import the software project.
5. After selecting the software project you want to import, select "Copy projects into workspace" and click "Finish", as shown in Figure 2-7. Note that you must select "Copy projects into workspace" for the imported zip file.

Figure 2-7 Select Copy projects into workspace

6. The imported software project is displayed in the Project Explorer view.

To ensure that you can successfully build and run the imported software project, the RDS software automatically checks the chip configuration, toolchain and connection configuration of the imported software project. If the RDS software does not support the chip configuration or toolchain or connection configuration used by the imported software project, refer to the software prompts to select an alternative.

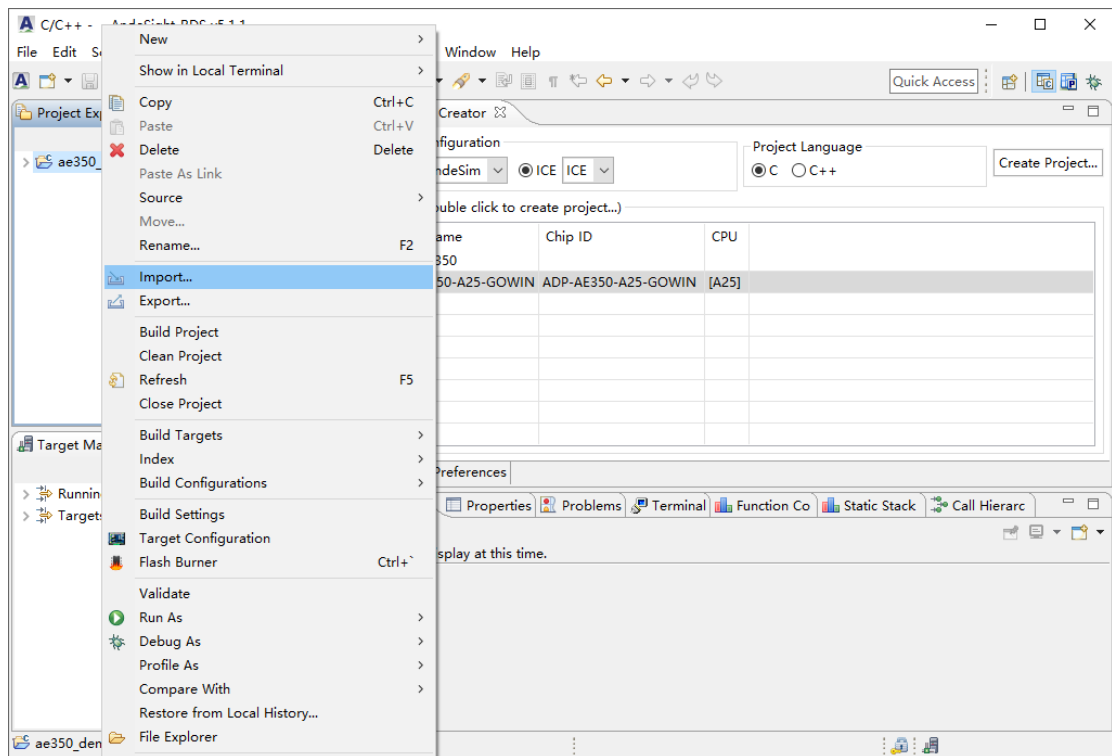
2.2 Software Project File

After creating a software project in the workspace, you can either import an existing source code file or create a new source code file.

2.2.1 Import an Existing File

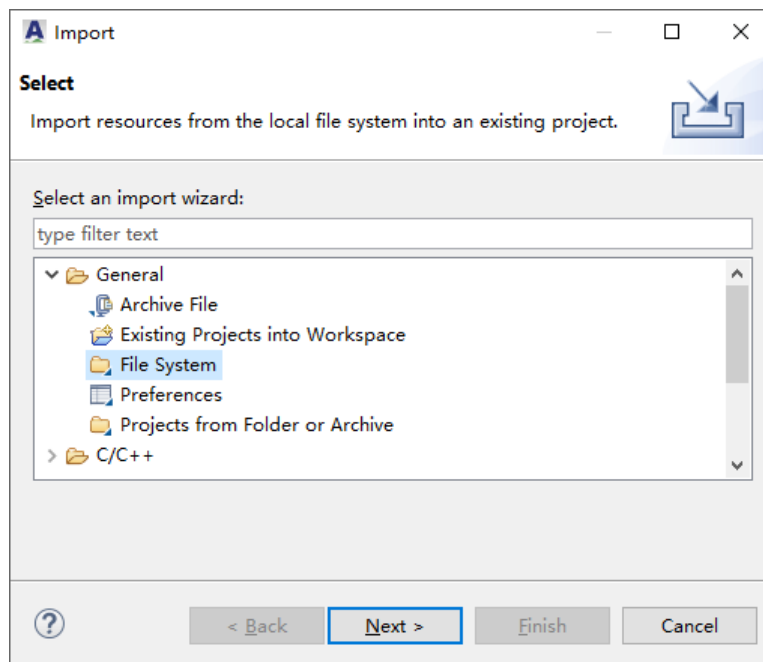
1. In the Project Explorer view, right-click to select a software project, and select "Import..." from the drop-down menu, as shown in Figure 2-8.

Figure 2-8 Select Import...



2. In the "Import" dialog box, select "General > File System", and click "Finish", as shown in Figure 2-9.

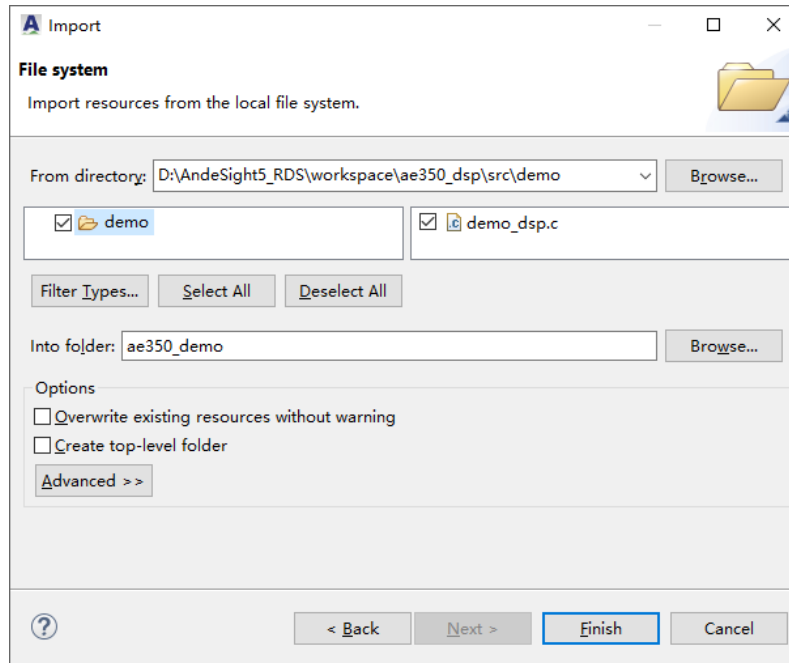
Figure 2-9 Select General > File System



3. In the "File system" dialog box, enter the path to the file you want to import, or click "Browse..." to search for the file path.

4. In the middle of the "File system" dialog box, the left column shows the path to the file you want to import, and the right column shows all files included in this path. Select the file you want to import in the right column, specify the software project you want to import in the "Info folder" text box, and click "Finish", as shown in Figure 2-10.

Figure 2-10 Select the Imported File

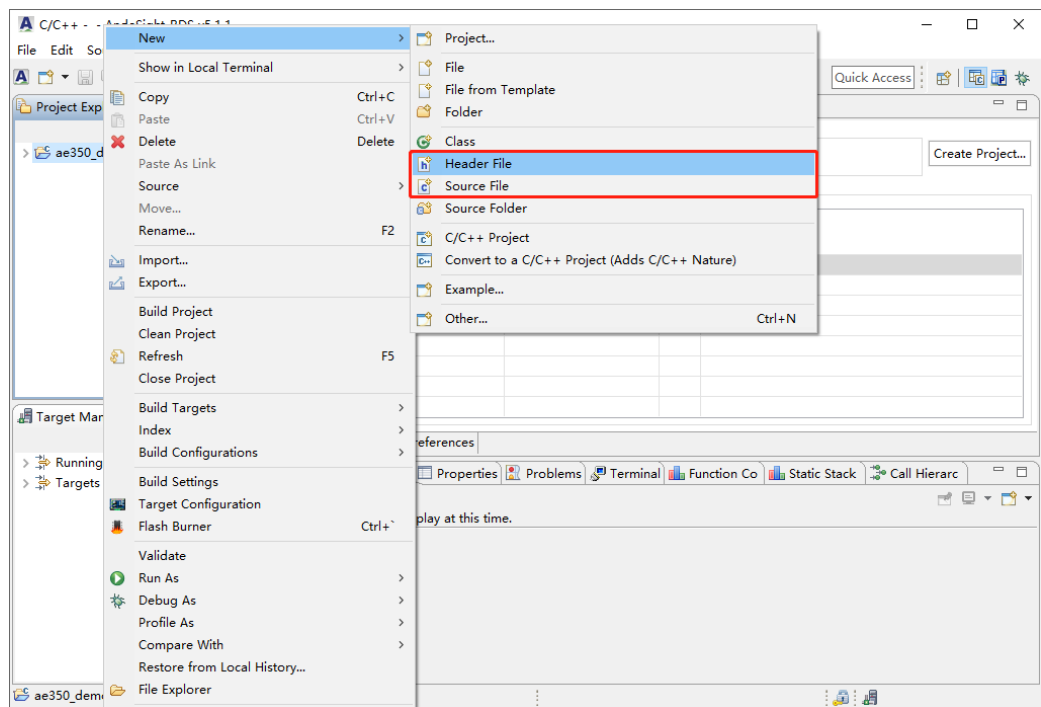


5. Return to the Project Explorer view to see the source code files imported to the software project.

2.2.2 Create Source/Header File

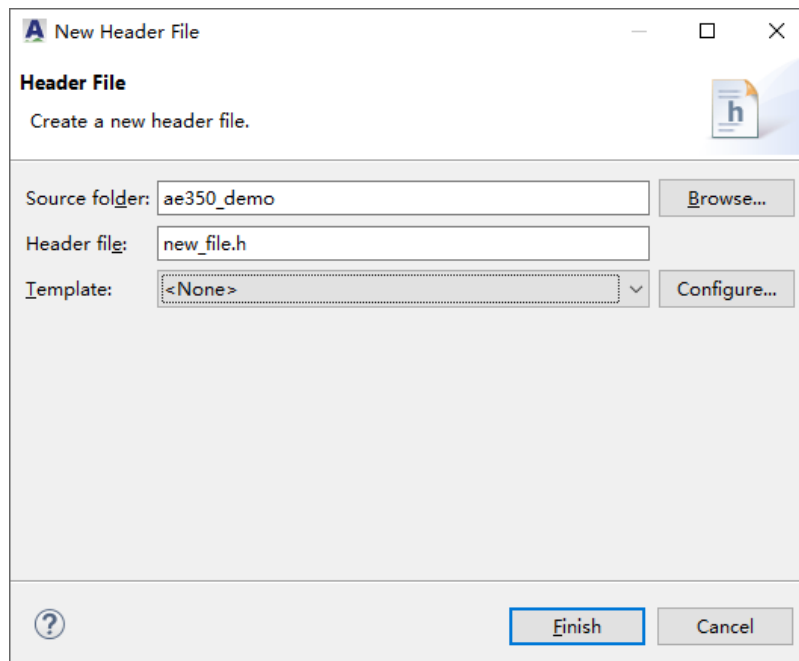
1. Right-click on the selected software project and select "New > Source File" or "Header File" from the drop-down menu, as shown in Figure 2-11.

Figure 2-11 Select New > Source File or Header File



2. In the "New Source File" or "New Header File" dialog box, enter the name of the new source/header file, including the .c or .h extension; select a template from the "Template" drop-down list, e.g., "<None>", and click "Finish", as shown in Figure 2-12.

Figure 2-12 Create New Source/Header File



3. Return to the Project Explorer view and check the newly created source/header file in the selected software project; double-click to

open this file and start writing code.

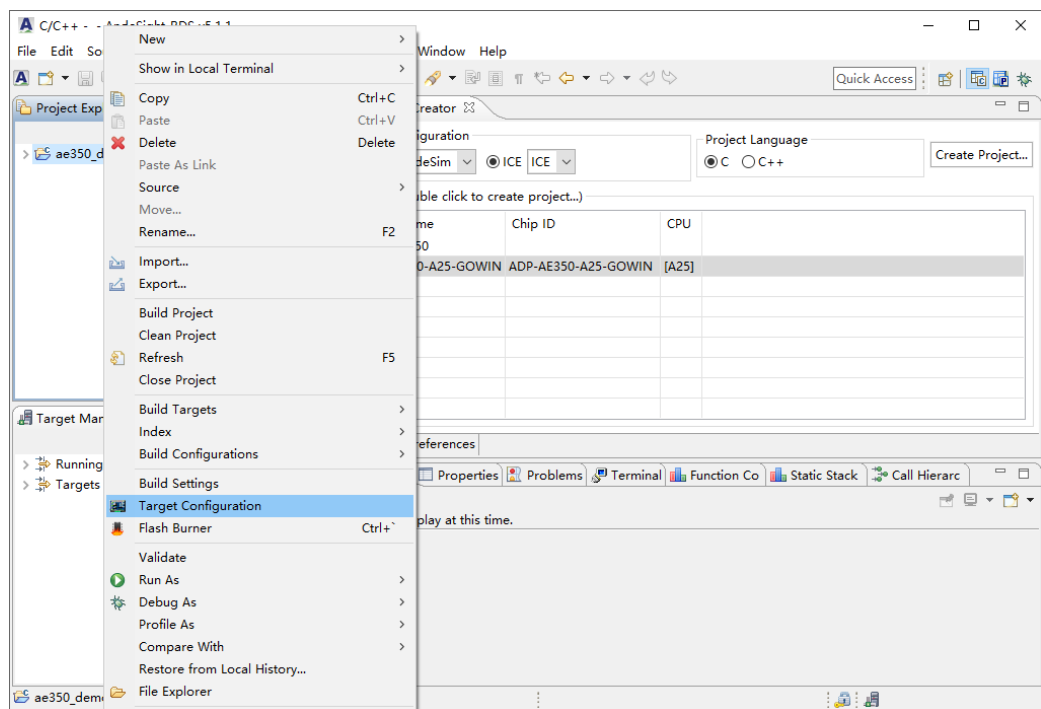
2.3 Target Configuration

The target configuration for the selected software project will override the default settings for target management for the entire workspace.

Refer to the following steps to set the target configuration for the software project:

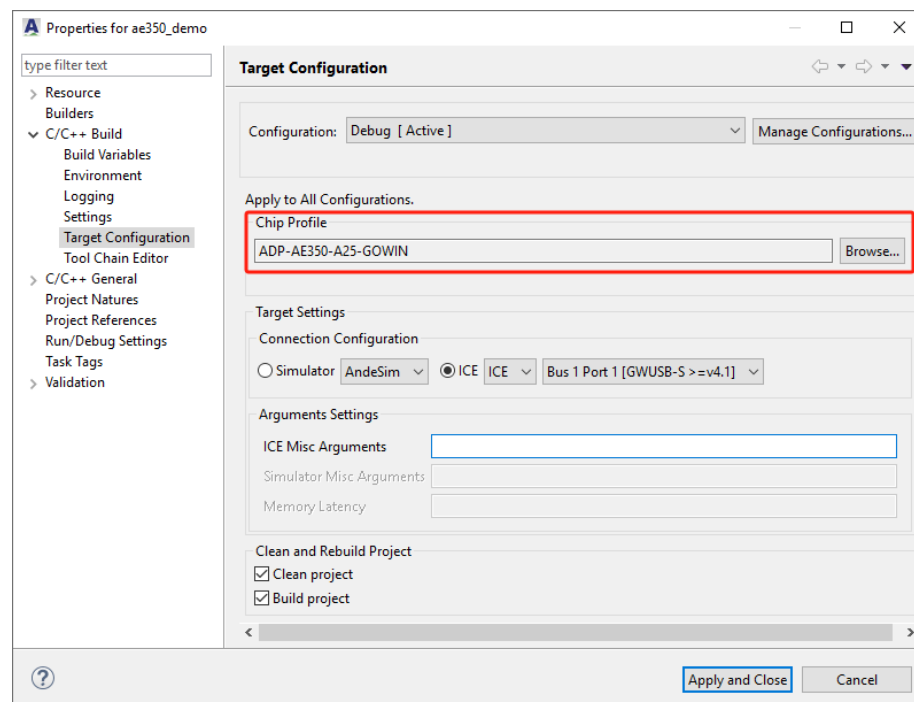
1. Right-click to select a software project, and select "Target Configuration" from the drop-down list, as shown in Figure 2-13.

Figure 2-13 Select Target Configuration



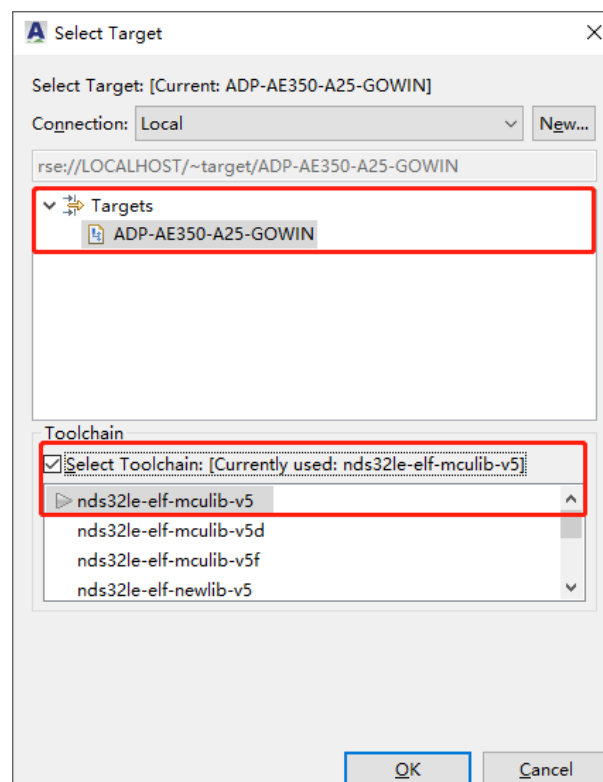
2. In the "Target Configuration" dialog box, select "Chip Profile" and click "Browse...", as shown in Figure 2-14.

Figure 2-14 Select Chip Profile



In the "Select Target" dialog box, select a target in "Targets" option, such as "ADP-AE350-A25-GOWIN". Click "Select Toolchains" and select a toolchain in the drop-down list, such as "nds32le-elf-mculib-v5", then Click "OK", as shown in Figure 2-15.

Figure 2-15 Select Targets and Toolchain



- Specify the connection configuration in "Target Settings" option, as shown in
- Bus 1 Port 1 [GWUSB-S >=v4.1]

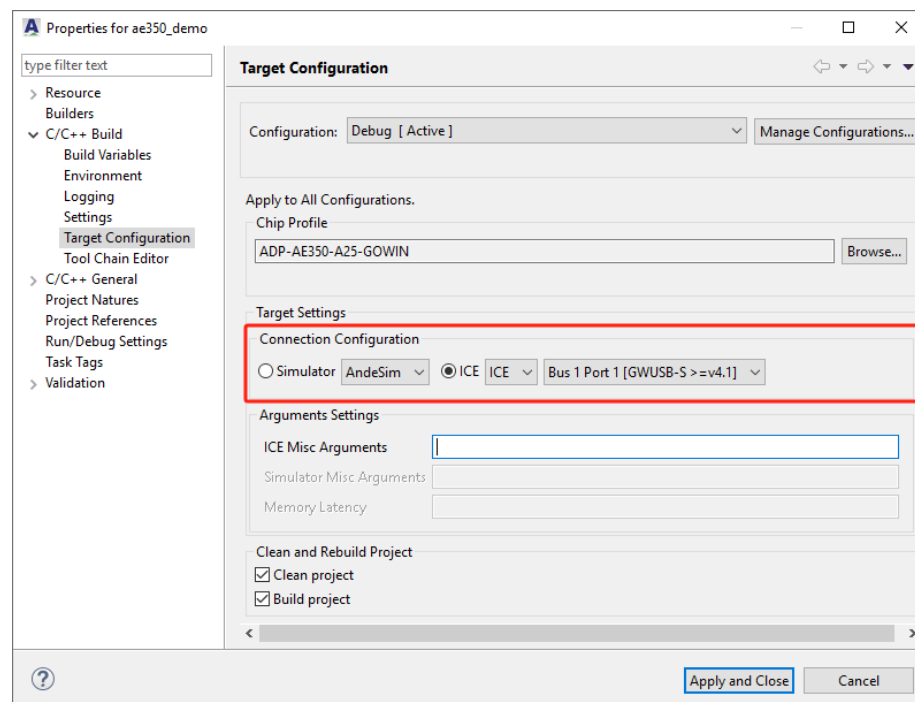
3. Figure 2-16.

For example:

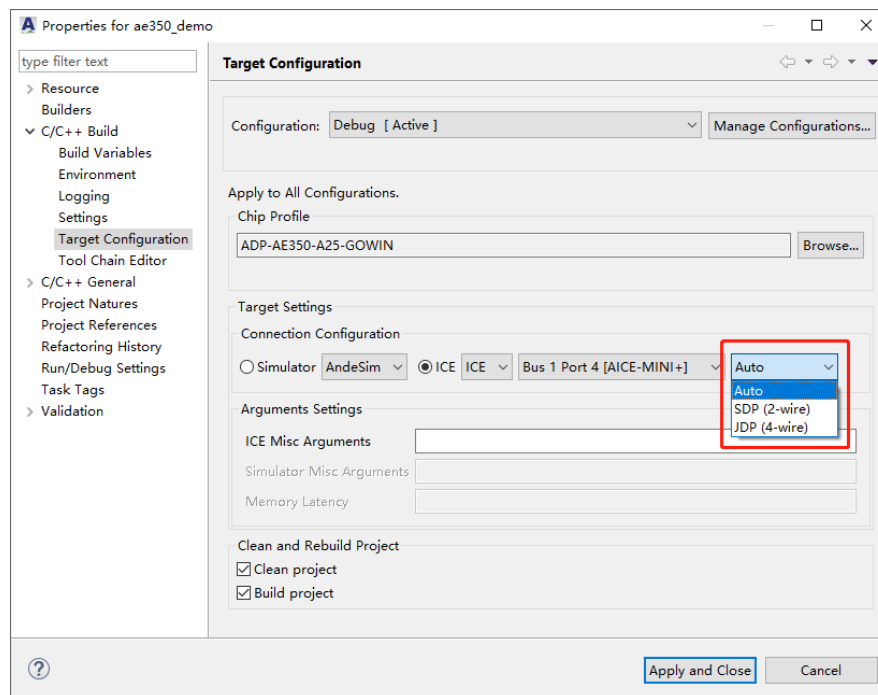
ICE:

- ICE
- Bus 1 Port 1 [GWUSB-S >=v4.1]

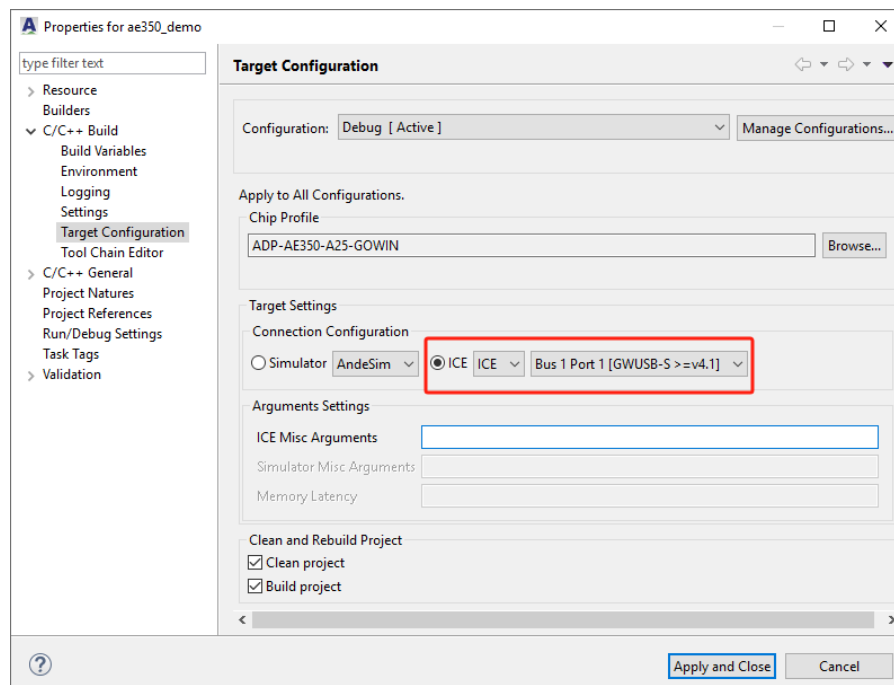
Figure 2-16 Select Target Settings



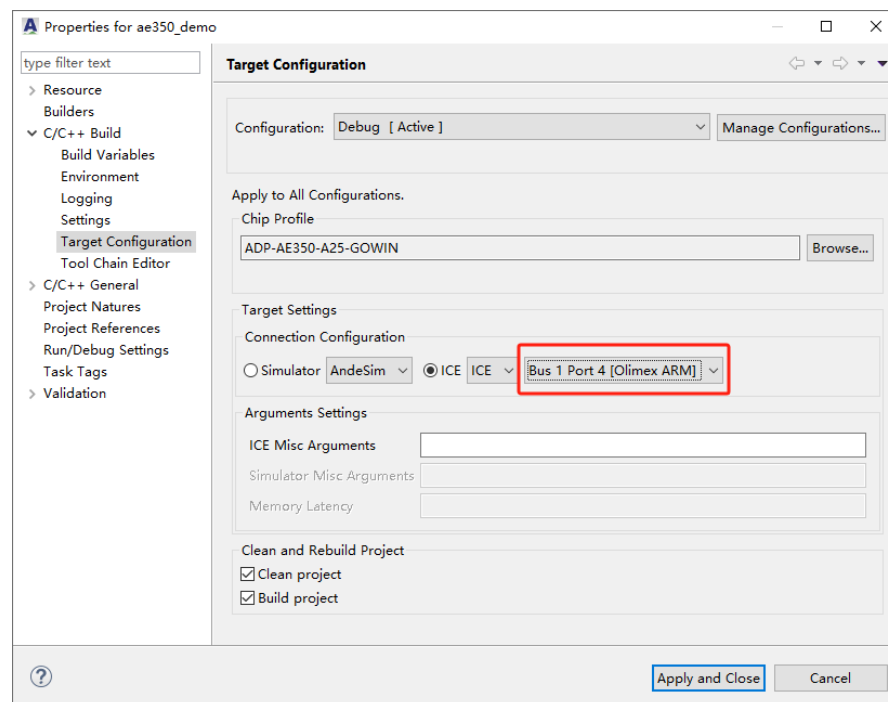
If you choose to connect the local target board, the RDS software will automatically detect and display the inserted ICE debugger. For Andes AICE-MINI+ debugger, the debugging interface of the target board must be specified, such as "SDP (2-wire)" or "Auto", as shown in Figure 2-17.

Figure 2-17 Select Connection Configuration (Andes AICE-MINI+) Debugger

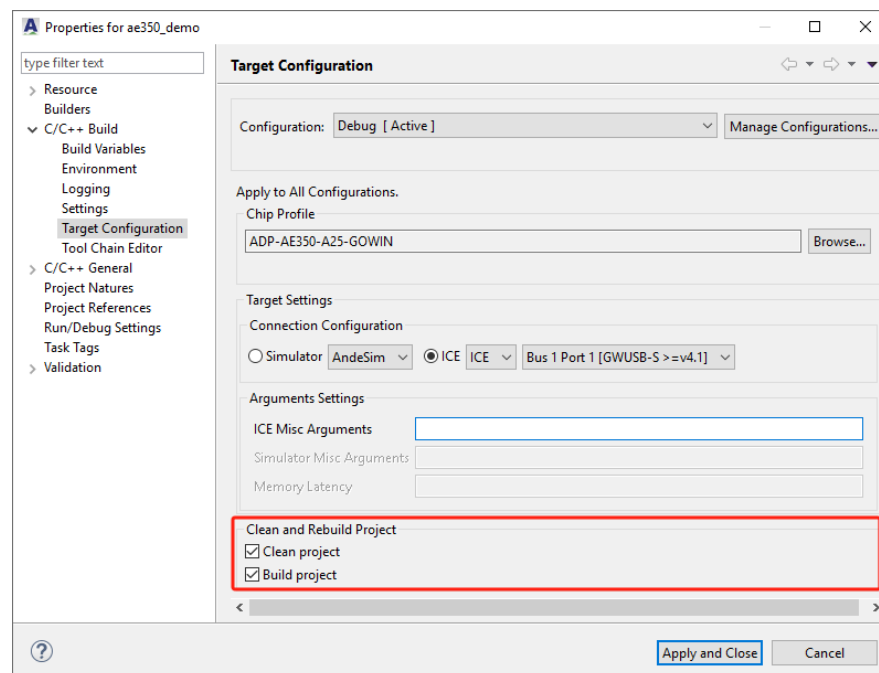
If you choose Gowin FTDI USB programmer, the configuration is as shown in Figure 2-18.

Figure 2-18 Select Gowin FTDI USB Programmer

If you choose Olimex debugger, the configuration is as shown in Figure 2-19.

Figure 2-19 Select Olimex Debugger

4. You can specify whether to clean or rebuild the software project after completing the target configuration in "Clean and Rebuild Project" option, as shown in Figure 2-20. "Clean project" is used to clean the software project. "Build... project" is used to build the software project.

Figure 2-20 Select Clean and Rebuild Project

5. Click "Apply and Close" to complete the target configuration for the

selected software project.

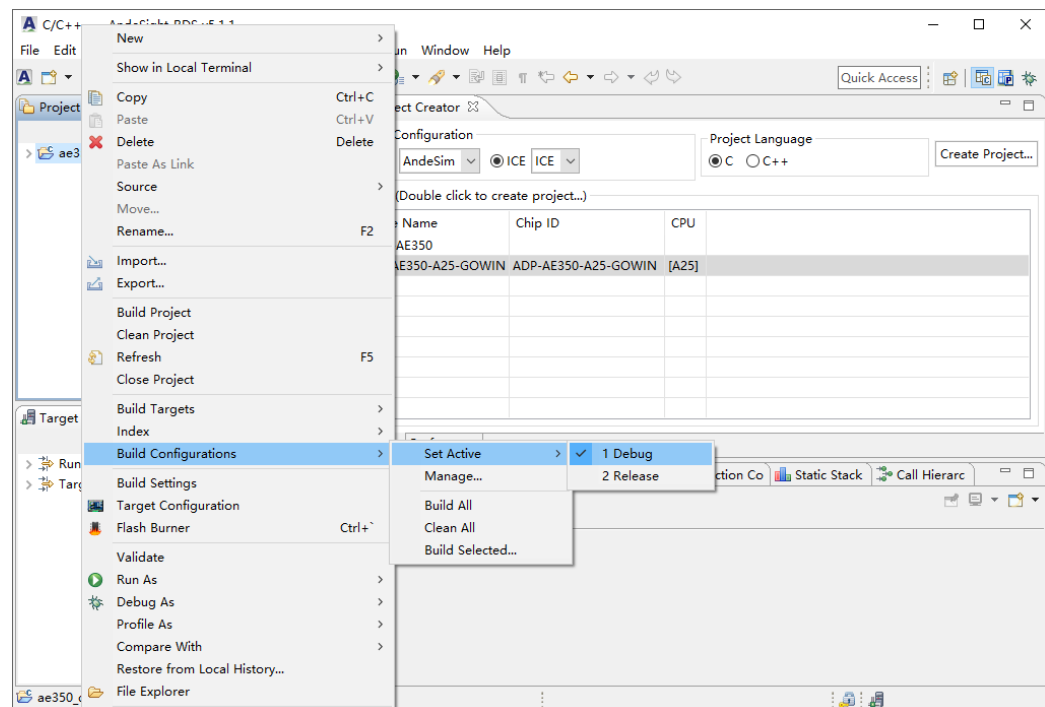
2.4 Create a Build System

Use GUI to select a build configuration for a software project to create a build system for the selected software project.

2.4.1 Specify Build Configuration

In the Project Explorer view, right-click the selected software project and select "Build Configurations > Set Active" in the drop-down list to select a type of build configuration and set it as active, as shown in Figure 2-21.

Figure 2-21 Select Build Configurations > Set Active

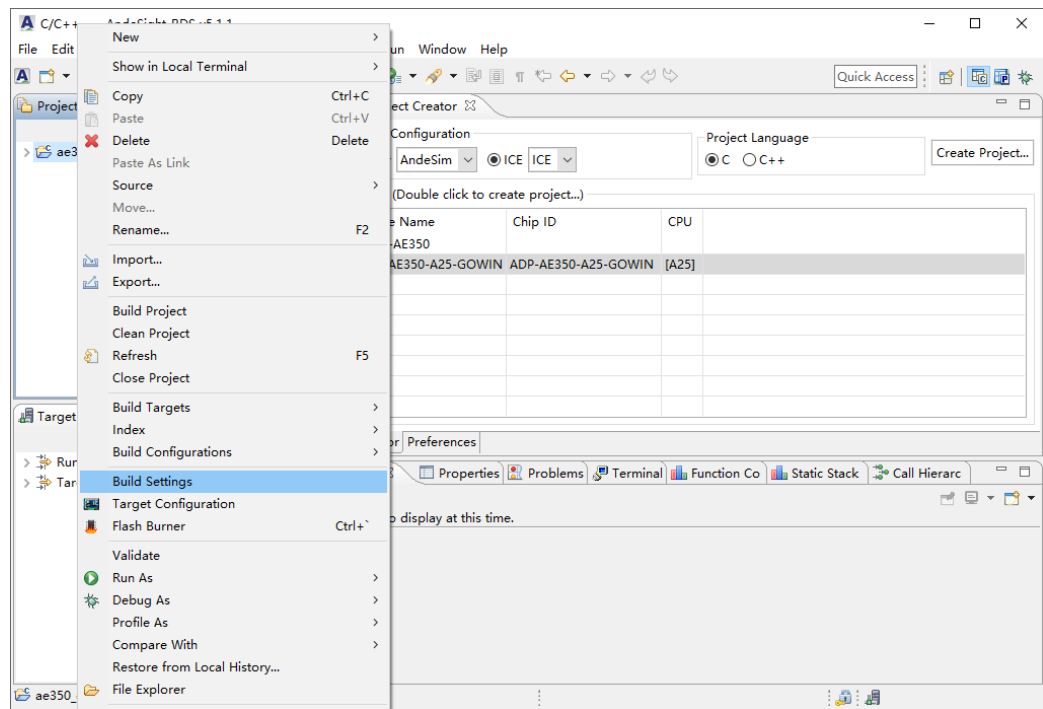


2.4.2 Set Build Configuration

Refer to the following steps to set the build configuration for the software project:

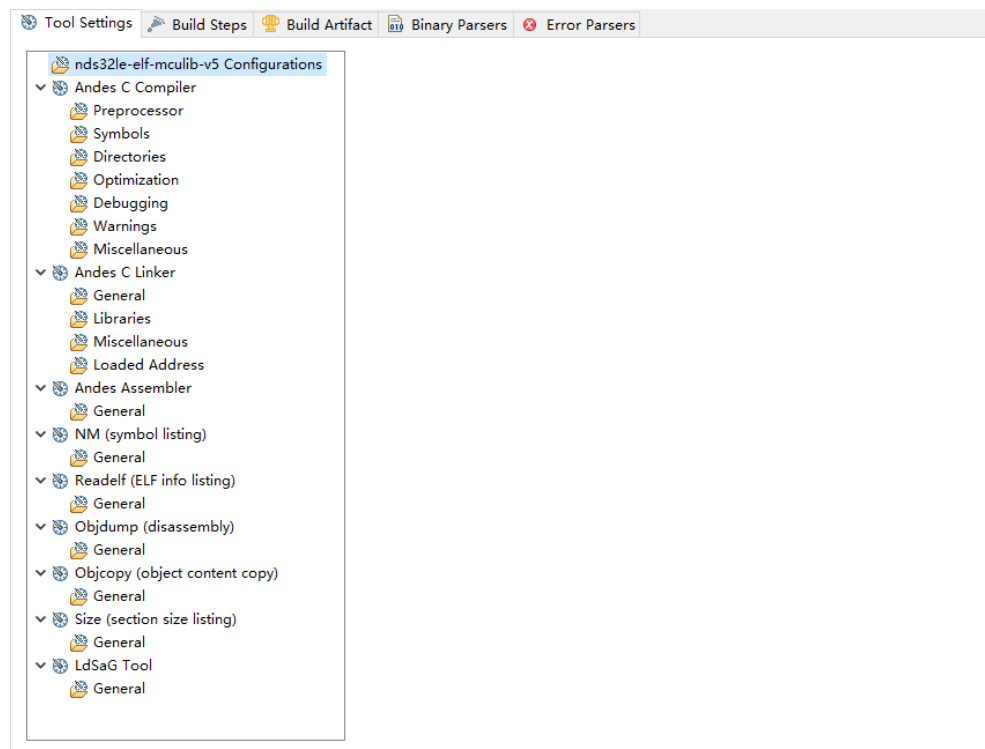
1. In Project Explorer View, right-click to select a software project, and select "Build Settings" from the drop-down list, as shown in Figure 2-22.

Figure 2-22 Select Build Settings



In the "Settings" dialog box, the "Tool Settings" tab lists all the build options that can be configured, so configure each option according to the application requirements, as shown in Figure 2-23.

Figure 2-23 Tool Settings Tab

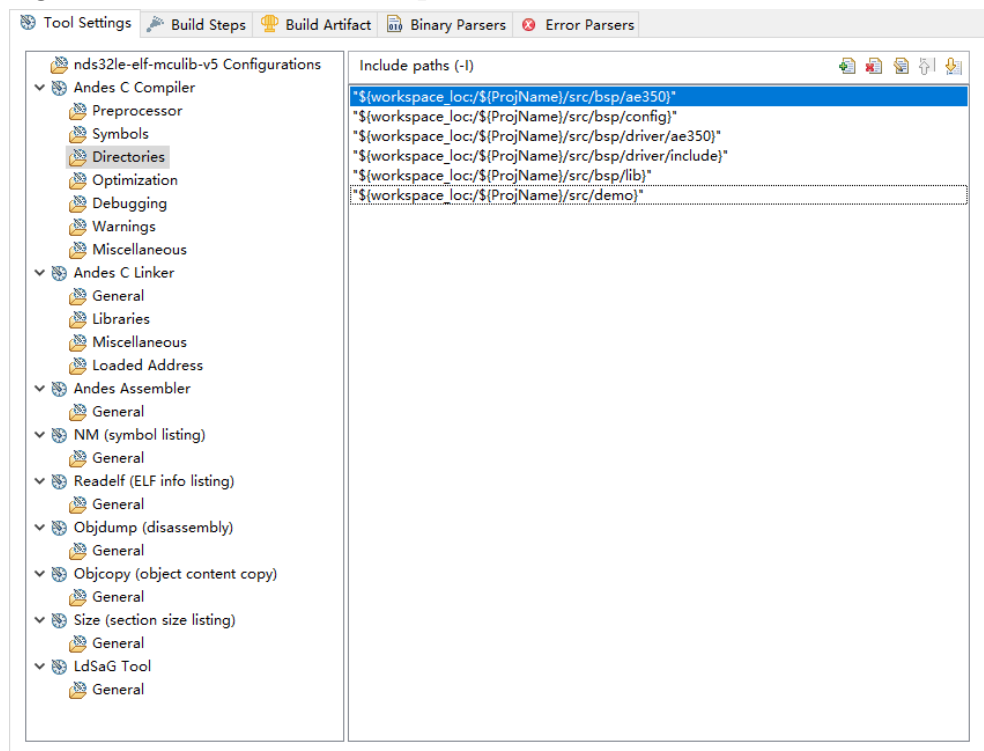


2. Select "Andes C Compiler > Directories > Include paths (-I)" to specify the path of the header files referenced by the C program, as shown in Figure 2-24.

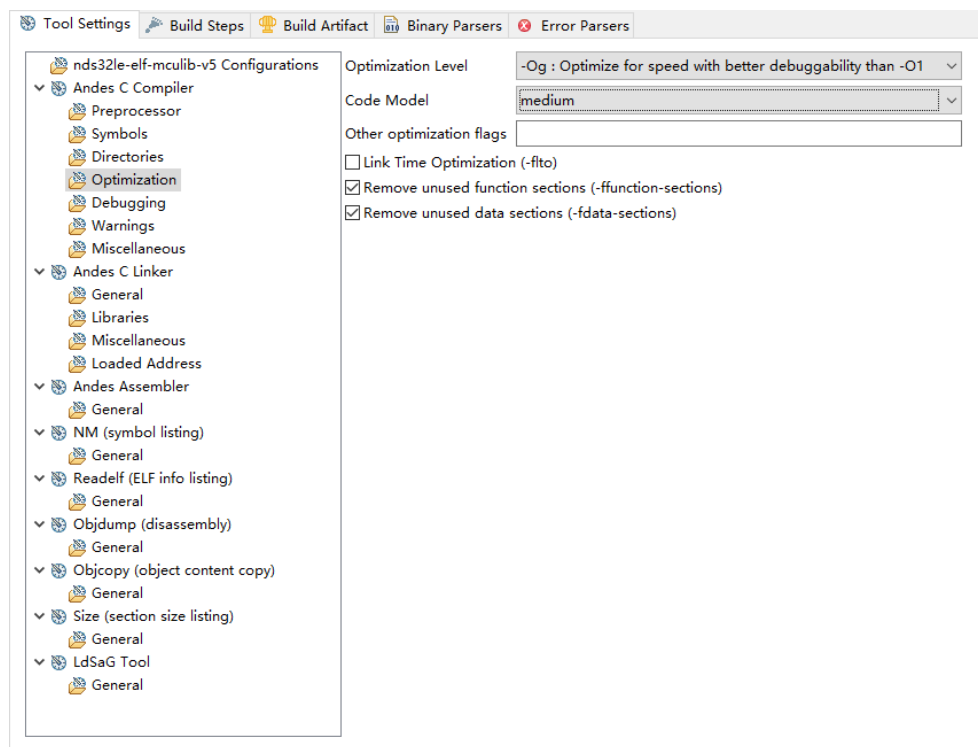
For example:

```
"${workspace_loc}/${ProjName}/src/bsp/ae350";
"${workspace_loc}/${ProjName}/src/bsp/config";
"${workspace_loc}/${ProjName}/src/bsp/driver/ae350"
"${workspace_loc}/${ProjName}/src/bsp/driver/include}"
"${workspace_loc}/${ProjName}/src/bsp/lib}"
"${workspace_loc}/${ProjName}/src/demo}"
```

Figure 2-24 Select Andes C Compiler > Directories



3. Select "Andes C Compiler > Optimization" to specify the optimization level and code model of the software project, as shown in Figure 2-25.

Figure 2-25 Select Andes C Compiler > Optimization

Optimization options are shown in Table 2-3.

Table 2-3 Optimization Options

Option	Configuration	Description
Optimization Level	-O0: Do not optimize -Og: Optimize for speed with better debuggability than -O1 -O1: Optimize for speed -O2: Optimize more for speed -O3: Optimize most for speed -Os1: Optimize for size -Os2: Optimize more for size -Os: Optimize most for size	Control the balance between the performance and code size in software project in accordance with the application requirements.
Code Model	small medium large	As shown in Table 2-4, specify the code model, indicating the instruction and data size.
Other optimization flags	...	Other user-specified optimization options
Link Time Optimization (-flto)	Enable/disable	Used to optimize link time of programs
Remove unused function sections (-	Enable/disable	Used to remove

Option	Configuration	Description
ffunction-sections)		unreferenced program instructions
Remove unused data section (-fdata-sections)	Enable/disable	Used to remove unreferenced program data

Table 2-4 Code Model

Code Model \ Segment	text	data	rodata
small	32-bit addressing space		
medium			
large			

The RDS software compiler has a number of compiler options for optimizations. Table 2-5 lists some of the common compiler optimization options used to control different types of optimizations.

Table 2-5 Optimization Options

Optimization Type	Option	Description
Code size	-Os1	This option enables minimum code size optimizations. Performance is still concerned.
	-Os2	This option enables partial code size optimizations with little performance concern.
	-OS3 (-Os)	This option works the same as the -Os option to enable all code size optimizations. Performance may seriously drop with this option.
	-Oz	Supported by LLVM only, this option works the same as -Os option to enable all code size optimizations. Performance may seriously drop with this option.
Code speed	-O3	Sometimes the code size may increase dramatically after this option is applied. This is because -O3 also implies -finline-functions that can expand the content of callee within the caller (See Table 17 for enabled options at -O3). To avoid such function inlining optimization, just use the option -fno-inline-functions.
	-funroll-loops	Use with GCC: Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. The compiler has a set of heuristics to estimate whether to unroll loop or not.

Optimization Type	Option	Description
		Use with LLVM: The compiler has a set of heuristics to estimate whether to unroll loop or not, even if the number of iterations is uncertain when the loop is entered. This option may make programs run more slowly if it loses locality after unrolling.
	-funroll-all-loops	Use with GCC: Unroll all loops, even if their number of iterations is uncertain when the loop is entered. This option may make programs run more slowly if it loses locality after unrolling. Use with LLVM: This option is accepted but ignored by clang.
	-malways-align	Enforces 4-byte alignment on jump targets, return addresses and function entries. The two options are to prevent extra performance penalty due to misalignment. They are not default applied at -Os (including -Os1, -Os2, -Os3, and -Oz) since they may slightly increase code size.
	-mcpu or -mtune	Optimizes the program for the given processor by specification of the CPU name or series. For example, -mcpu=a25 and -mtune=andes-25-series. Both options will optimize the program according to processor feature such as pipeline latency.
Undefined behavior	-fno-delete-null-pointer-checks	In the C language standard, programs cannot safely dereference null pointers, and no code or data element resides there. However, this assumption is not true in some cases, especially for embedded platform. Thus, if you have to dereference the memory address 0x00000000, please use -fno-delete-null-pointer-checks to tell the compiler not to optimize out null pointer checking.
	-fno-strict-aliasing	In the compiler framework, it may enable strict aliasing optimization on the assumption that the strictest aliasing rules applicable to the language being compiled. If a program contains pointer casting, it may break the strict aliasing rule. Therefore, it would be better not to use pointer casting in your programs. If you must use it, having the option -fno-strict-aliasing is recommended. Otherwise, the execution result may be unexpected.
	-fwrapv	The C language standard considers the overflow of a signed value is undefined behavior. That means a valid program must never generate signed overflow when computing an expression and the compiler is able to perform some optimization under such condition. If you must have invalid

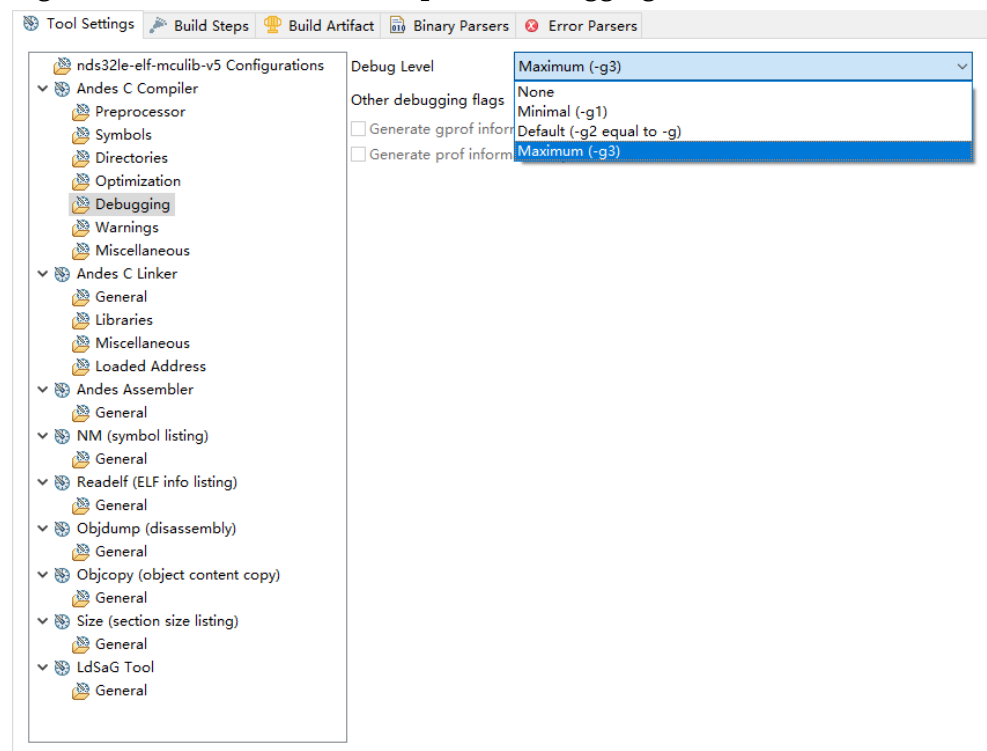
Optimization Type	Option	Description
		code containing signed overflow, please compile it with -fwrapv, which tells the compiler to treat signed overflow as wrapping.
	-mfma	The option -fma is used to generate floating-point fused multiply-add (FMA) instructions. It is enabled by default in RDS Compiler to increase the performance, albeit an unsafe floating-point math optimization which may decrease precision. If you need a program result compatible with IEEE 754, use -mno-fma to disable this optimization.

4. Select "Andes C Compiler > Debugging" to specify the debugging level of the software project, as shown in Figure 2-26.

For example:

Debug Level: Maximum (-g3)

Figure 2-26 Select Andes C Compiler > Debugging



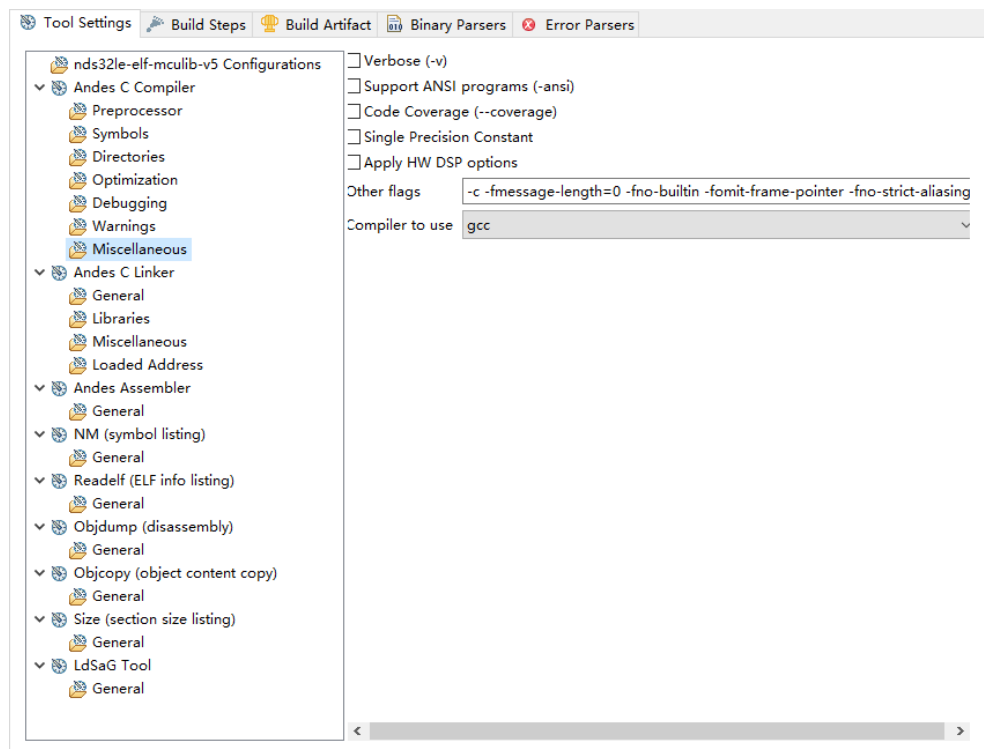
5. Select "Andes C Compiler > Miscellaneous" to specify other compiler options for the software project, as shown in Figure 2-27.

For example:

Other flags: -c -fmessage-length=0 -fno-builtin -fomit-frame-pointer -fno-strict-aliasing;

Compiler to use: gcc

Figure 2-27 Select Andes C Compiler > Miscellaneous



Miscellaneous options are as shown in Table 2-6.

Table 2-6 Miscellaneous Options

Option	Configuration	Description
Other flags	-c -fmessage-length=0 -fno-builtin -fomit-frame-pointer -fno-strict-aliasing	-fmessage-length=N: Formatting error messages, and the default value is 0
Code Coverage (--coverage)	Enable/disable	Code coverage analysis
Single Precision Constant	Enable/disable	Floating-point constants are treated as single-precision constants
Apply HW DSP options	Enable/disable	Supports DSP ISA extended option
Compiler to use	gcc llvm	Applicable to the RDS RISC-V elf toolchain that includes both LLVM and GCC compilers

The RDS software compiler is built from GNU and LLVM, inheriting the gcc, as and ld options from GNU and the clang and ld options from LLVM. In addition to the GNU and LLVM options, specific options are

provided for special features, such as the balance between performance and code size.

The GCC compiler options for RDS software are shown in Table 2-7.

Table 2-7 GCC Compiler Options

Option	Description
-pass-exit-codes	Exit with highest error code from a phase.
--help	Display this information.
--target-help	Display target specific command line options.
--version	Display compiler version information.
-dumpspecs	Display all of the built in spec strings.
-dumpversion	Display the version of the compiler.
-dumpmachine	Display the compiler's target processor.
-print-search-dirs	Display the directories in the compiler's search path.
-print-libgcc-file-name	Display the name of the compiler's companion library.
-print-file-name=<lib>	Display the full path to library <lib>.
-print-prog-name=<prog>	Display the full path to compiler component <prog>.
-print-multiarch	Display the target's normalized GNU triplet, used as a component in the library path.
-print-multi-directory	Display the root directory for versions of libgcc.
-print-multi-lib	Display the mapping between command line options and multiple library search directories.
-print-multi-os-directory	Display the relative path to OS libraries.
-print-sysroot	Display the target libraries directory.
-print-sysroot-headers-suffix	Display the sysroot suffix used to find headers.
-Wa,<options>	Pass comma-separated <options> on to the assembler.
-Wp,<options>	Pass comma-separated <options> on to the preprocessor.
-Wl,<options>	Pass comma-separated <options> on to the linker.
-Xassembler <arg>	Pass <arg> on to the assembler.
-Xpreprocessor <arg>	Pass <arg> on to the preprocessor.
-Xlinker <arg>	Pass <arg> on to the linker.
-save-temps	Do not delete intermediate files.
-save-temps=<arg>	Do not delete intermediate files.
-no-canonical-prefixes	Do not canonicalize paths when building relative prefixes to other gcc components.

Option	Description
-pipe	Use pipes rather than intermediate files.
-time	Time the execution of each subprocess.
-specs=<file>	Override built-in specs with the contents of <file>.
-std=<standard>	Assume that the input sources are for <standard>.
--sysroot=<directory>	Use <directory> as the root directory for headers and libraries.
-B <directory>	Add <directory> to the compiler's search paths.
-v	Display the programs invoked by the compiler.
	Preprocess only; do not compile, assemble or link.
-S	Compile only; do not assemble or link.
-c	Compile and assemble, but do not link.
-o <file>	Place the output into <file>.
-pie	Create a position independent executable.
-shared	Create a shared library.
-mabi=	Specify integer and floating-point calling convention.
-mace	Compile with ACE.
-mace-s2s=	Argument for pass to ACE source-to-source translator.
-malways-align	Always align function entry, jump target and return address.
-march=	Generate code for given RISC-V ISA (e.g. RV64IM). ISA strings must be lower-case.
-matomic	Use atomic extension instructions.
-mbf16	Support BFLOAT16 conversion extension
-mb20282	Avoid conditional jump erratum
-mbranch-cost=N	Set the cost of branches to roughly N instructions.
-mcmmodel=	Specify the code model.
-mconfig-mul=	Specify configuration of instruction mul.
-mcpu=	-mcpu=PROCESSOR Optimize the output for PROCESSOR. Same as -mtune=.
-mctor-dtor	Enable constructor/destructor feature.
-mdiv	Use hardware instructions for integer division.
-mexecit	Use special directives to guide linker to perform exec.it optimization.
-mexplicit-relocs	Use %reloc() operators, rather than assembly macros, to load addresses.
	Use hardware DSP instructions.
-mext-vector	Use vector instructions.

Option	Description
-mext-zbabcs	Use RVB hardware instructions including zba, zbb, zbc, zbs.
-mfddiv	Use hardware floating-point divide and square root instructions.
-mfma	Generating fma instructions.
-mgp-insn-relax	Use special directives to guide linker doing gp insn relax optimization.
-minnermost-loop	Insert the innermost loop directive.
-mmove-bytes-per-loop=	-mmove-bytes-per-loop=N Set <N> bytes of data movement can be handled per loop iteration.
-mno-16-bit	Do not generate C extension instructions.
-mplt	When generating -fpic code, allow the use of PLTs. Ignored for fno-pic.
-mpreferred-stack-boundary=	Attempt to keep stack aligned to this power of 2.
-mrelax	Take advantage of linker relaxations to reduce the number of instructions required to materialize symbol addresses.
-mrestrict-even-reg-for-regpair	Restrict register pair must start with even register number.
-msave-restore	Use smaller but slower prologue and epilogue code.
-msmall-data-limit=N	Put global and static data smaller than <number> bytes into a special section (on some targets).
-mstrict-align	Do not generate unaligned memory accesses.
-mtune=PROCESSOR	Optimize the output for PROCESSOR.
-mvh	Enable Virtual Hosting support.
-mzfh	Support Zfh half-precision floating-point extension.

The LLVM compiler options for RDS software are shown in Table 2-8.

Table 2-8 LLVM Compiler Options

Option	Description
-###	Print (but do not run) the commands to run for this compilation
-B <dir>	Add <dir> to search path for binaries and object files used implicitly
-c	Only run preprocess, compile, and assemble steps
-E	Only run the preprocessor
--help-hidden	Display help for hidden options
-help	Display available options
-o <file>	Write output to <file>
-no-canonical-prefixes	Use relative instead of canonical paths

Option	Description
-pipe	Use pipes between commands, when possible
-print-file-name=<file>	Print the full library path of <file>
-print-libgcc-file-name	Print the library path for the currently used compiler runtime library ("libgcc.a" or "libclang_rt.builtins.*.a")
-print-multi-directory	Print the directory name corresponding to the supported multilib
-print-multi-lib	Print the mapping from command line options to multilib search directories
-print-prog-name=<prog>	Print the full program path of <name>
-print-search-dirs	Print the paths used for finding libraries and programs
-save-temps=<value>	Save intermediate compilation results.
-save-temps	Save intermediate compilation results
-std=<value>	Language standard to compile for
--sysroot=<value>	Set the root directory for headers and libraries
-S	Only run preprocess and compilation steps
-time	Time individual commands
--version	Print version informatio
-v	Show commands to run and use verbose output
-Wa,<arg>	Pass the comma separated arguments in <arg> to the assembler
-Wl,<arg>	Pass the comma separated arguments in <arg> to the linker
-Wp,<arg>	Pass the comma separated arguments in <arg> to the preprocessor
-Xassembler <arg>	Pass <arg> to the assembler
-Xlinker <arg>	Pass <arg> on to the linker.
-Xpreprocessor <arg>	Pass <arg> on to the preprocessor.
-x <language>	Treat subsequent input files as having type <language>
-mabi=<value>	Set the RISC-V ABI
-mace-s2s=<value>	Argument for pass to ACE source-to-source translator.
-mace	Compile with ACE.
-malways-align	Always align function entry, jump target and return address.
-march=<value>	Set the RISC-V architecture
-matomic	Use atomic extension instructions.
-mb20282	Avoid conditional jump erratum
-mb22827.1	Insert a NOP after FSHW conditionally
-mb22827	Insert a fclass.x after FDIV.x/FSQRT.x conditionally
-mbf16	Enable bf16 support

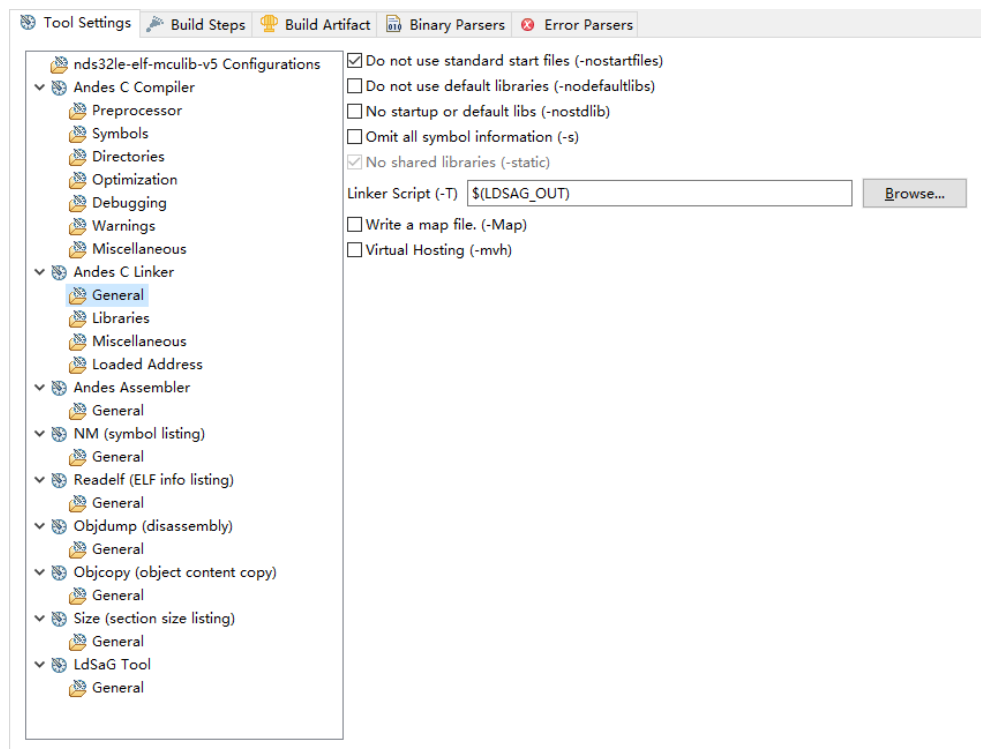
Option	Description
-mcmmodel=<value>	Specify the code model.
-mctor-dtor	Enable constructor/destructor feature
-mdiv	Use hardware instructions for integer division
-mexecit	Use special directives to guide linker doing execit optimization.
-mext-dsp	Enable DSP support
-mext-zbabcs	Use RVB hardware instructions including zba, zbb, zbc, zbs.
-mfdiv	Use hardware floating-point divide and square root instructions.
-mgrp-insn-relax	Use special directives to guide linker doing gp insn relax optimization.
-mict-model=<value>	Specify the address generation strategy for ICT call's code model
-minnermost-loop	Insert the innermost loop directive.
-mno-16-bit	Do not generate C extension instructions.
-mrelax	Enable linker relaxation
-msave-restore	Use smaller but slower prologue and epilogue code
-msmall-data-limit=<value>	Put global and static data smaller than <number> bytes into a special section.
-mstrict-align	Force all memory accesses to be aligned (same as mno-unaligned-access)
-mvh	Enable link virtual hosting
-mzfh	Enable zfh support

6. Select "Andes C Linker > General" to specify the linker for the software project, as shown in Figure 2-28.

For example:

Do not use standard start files (-nostartfiles): Enable

Linker Script (-T): \$(LDSAG_OUT)

Figure 2-28 Select Andes C Linker > General

General options are as shown in Table 2-9.

Table 2-9 General Options

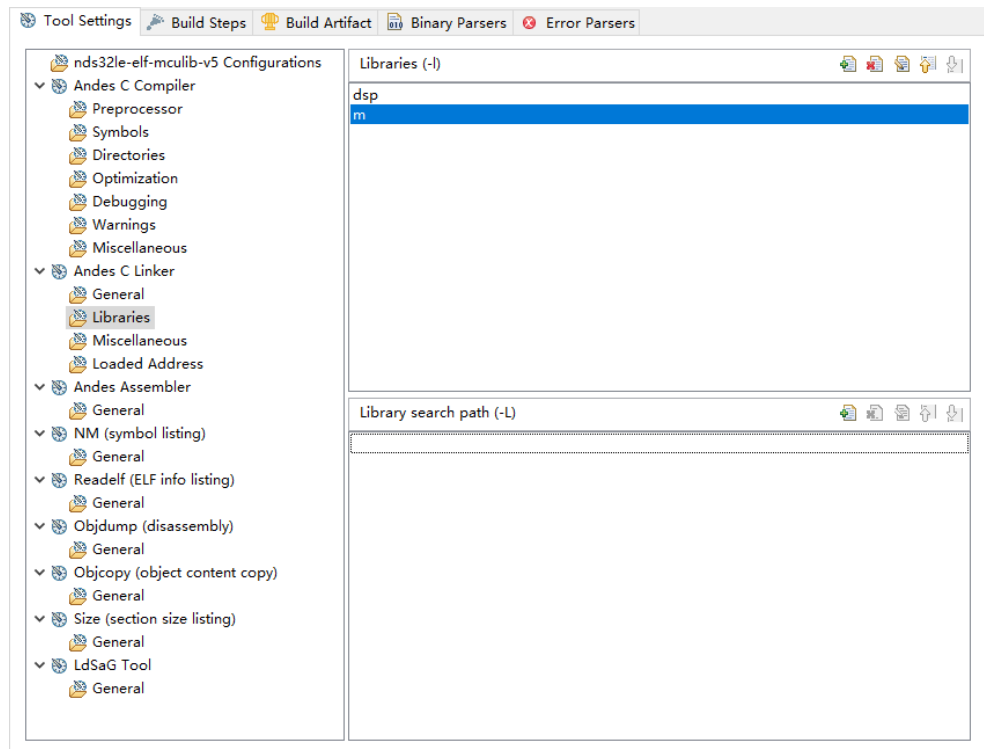
Option	Configuration	Description
Do not use standard start files (-nostartfiles): Enable	Enable/disable	This option must be enabled, and the standard start files are not used.
Linker Script (-T)	\$(LDSAG_OUT)	This option is associated with the "LdSaG Tool > General" option. If a SaG file has been specified, this link script file is generated by the SaG file, otherwise it must be specified manually.
Write a map file. (-Map)	Enable/disable	Generate map.txt file
Virtual Hosting (-mvh)	Enable/Disable	Activating the virtual host allows redirecting I/O requests from target boards without I/O devices to the host-side GDB.

7. Select "Andes C Linker > Libraries" to specify the external software libraries linked to the software project, as shown in Figure 2-29.

For example:

Libraries (-l):

- dsp
- m

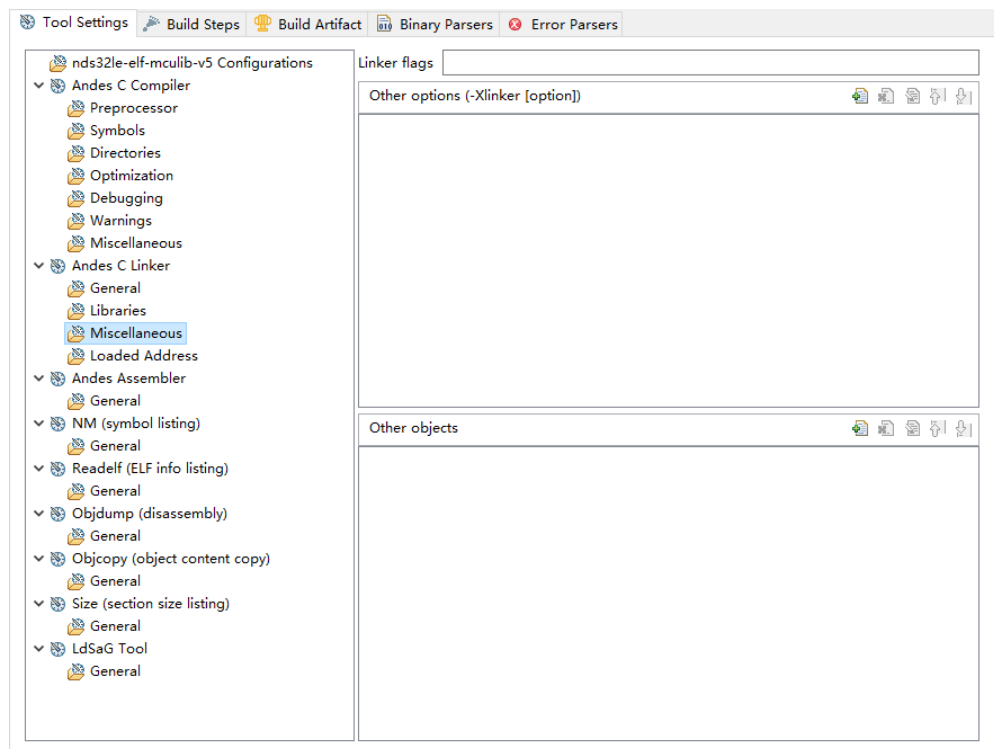
Figure 2-29 Select Andes C Linker > Libraries

Libraries options are as shown in Table 2-10.

Table 2-10 Libraries Options

Option	Description
Libraries (-l)	Specify the external software libraries linked to the software project, e.g., dsp library, math library, and nn library etc.
Library search path (-L)	Path to the external software libraries

8. Select "Andes C Linker > Miscellaneous" to specify other options for the software project, as shown in Figure 2-30.

Figure 2-30 Select Andes C Linker > Miscellaneous

The linker options for RDS software are as shown in Table 2-11.

Table 2-11 Linker Options

Option	Description
--allow-multiple-definition	Allow multiple definitions
--as-needed	Only set DT_NEEDED for shared libraries if used
--auxiliary <value>	Set DT_AUXILIARY field to the specified name
--Bdynamic	Link against shared libraries
--Bshareable	Build a shared object
--Bstatic	Do not link against shared libraries
--Bsymbolic-functions	Bind defined function symbols locally
--Bsymbolic	Bind defined symbols locally
--build-id=<value>	Generate build ID note
--build-id	Generate build ID note
--b <value>	Change the input format of the inputs following this option
--call_shared	Link against shared libraries
--color-diagnostics=<value>	Use colors in diagnostics
--color-diagnostics	Use colors in diagnostics
--compress-debug-	Compress DWARF debug sections

Option	Description
sections=<value>	
--compress-debug-sections <value>	Compress DWARF debug sections
--dc	Assign space to common symbols
--define-common	Assign space to common symbols
--defsym=<value>	Define a symbol alias
--defsym <value>	Define a symbol alias
--demangle	Demangle symbol names
--disable-new-dtags	Disable new dynamic tags
--discard-all	Delete all local symbols
--discard-locals	Delete temporary local symbols
--discard-none	Keep all symbols in the symbol table
--dn	Do not link against shared libraries
--dp	Assign space to common symbols
--dynamic-linker=<value>	Which dynamic linker to use
--dynamic-linker <value>	Which dynamic linker to use
--dynamic-list=<value>	Read a list of dynamic symbols
--dynamic-list <value>	Read a list of dynamic symbols
--dy	Link against shared libraries
-d	Assign space to common symbols
--eh-frame-hdr	Request creation of .eh_frame_hdr section and PT_GNU_EH_FRAME segment header
--emit-relocs	Generate relocations in output
--enable-new-dtags	Enable new dynamic tags
--end-lib	End a grouping of objects that should be treated as if they were together in an archive
--entry=<entry>	Name of entry point symbol
--entry <entry>	Name of entry point symbol
--error-limit=<value>	Maximum number of errors to emit before stopping (0 = no limit)
--error-limit <value>	Maximum number of errors to emit before stopping (0 = no limit)
--error-unresolved-symbols	Report unresolved symbols as errors
--exclude-libs=<value>	Exclude static libraries from automatic export
--exclude-libs <value>	Exclude static libraries from automatic export

Option	Description
--export-dynamic-symbol=<value>	Put a symbol in the dynamic symbol table
--export-dynamic-symbol <value>	Put a symbol in the dynamic symbol table
--export-dynamic	Put symbols in the dynamic symbol table
	Put symbols in the dynamic symbol table
-e <value>	Name of entry point symbol
--fatal-warnings	Treat warnings as errors
--filter=<value>	Set DT_FILTER field to the specified name
--filter <value>	Set DT_FILTER field to the specified name
--fini=<symbol>	Specify a finalizer function
--fini <symbol>	Specify a finalizer function
--force-exe-suffix	Force generation of file with .exe suffix
--format=<input-format>	Change the input format of the inputs following this option
--format <input-format>	Change the input format of the inputs following this option
--full-shutdown	Perform a full shutdown instead of calling _exit
-F <value>	Set DT_FILTER field to the specified name
-f <value>	Set DT_AUXILIARY field to the specified name
--gc-sections	Enable garbage collection of unused sections
--gdb-index	Generate .gdb_index section
--hash-style=<value>	Specify hash style (sysv, gnu or both)
--hash-style <value>	Specify hash style (sysv, gnu or both)
--help	Print option help
-h <value>	Set DT_SONAME
--icf-data	Enable ICF to also fold identical read only data
--icf=all	Enable identical code folding
--icf=none	Disable identical code folding
--image-base=<value>	Set the base address
--image-base <value>	Set the base address
--init=<symbol>	Specify an initializer function
--init <symbol>	Specify an initializer function
--just-symbols=<value>	Just link symbols

Option	Description
--just-symbols <value>	Just link symbols
--library-path=<dir>	Add a directory to the library search path
--library-path <dir>	Add a directory to the library search path
--library=<libName>	Root name of library to use
--library <libName>	Root name of library to use
--lto-aa-pipeline=<value>	AA pipeline to run during LTO. Used in conjunction with -lto-newpm-passes
--lto-newpm-passes=<value>	Passes to run during LTO
--lto-O<opt-level>	Optimization level for LTO
--lto-partitions=<value>	Number of LTO codegen partitions
-L <value>	Add a directory to the library search path
-l <value>	Root name of library to use
--Map=<value>	Print a link map to the specified file
--Map <value>	Print a link map to the specified file
--mexecit-limit=<value>	Maximum number of entries in exec.it table
--mexecit-loop-aware	Avoid generating exec.it instruction inside loop
--mexecit	Perform exec.it optimization
--mexport-execit=<value>	Export exec.it table after linking
--mexport-symbols=<value>	Export defined global symbols in form of linker script
--mimport-execit=<value>	Import exec.it table
--mkeep-import-execit<value>	Keep exec.it table
--mno-execit	Don't perform exec.it optimization.
--mupdate-execit<value>	Update exec.it table
-M	Print a link map to the standard output
-m <value>	Set target emulation
--no-as-needed	Always DT_NEEDED for shared libraries
--no-color-diagnostics	Do not use colors in diagnostics
--no-define-common	Do not assign space to common symbols
--no-demangle	Do not demangle symbol names
--no-dynamic-linker	Inhibit output of .interp section
--no-eh-frame-hdr	Do not create .eh_frame_hdr section
--no-gc-sections	Disable garbage collection of unused sections

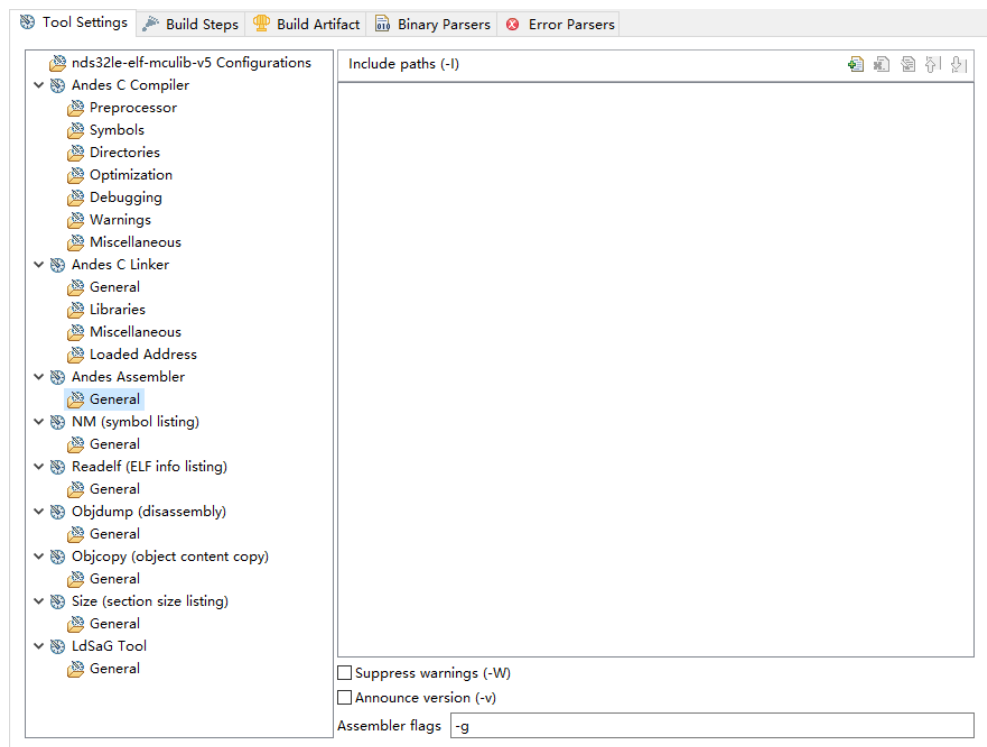
Option	Description
--no-gdb-index	Do not generate .gdb_index section
--no-gnu-unique	Disable STB_GNU_UNIQUE symbol binding
--no-omagic	Do not set the text data sections to be writable
--no-print-gc-sections	Do not list removed unused sections
--no-relax	Do not perform linker relaxation to reduce code size
--no-rosegment	Do not put read-only non-executable sections in their own segment
--no-threads	Do not run the linker multi-threaded
--no-undefined-version	Report version scripts that refer undefined symbols
--no-undefined	Report unresolved symbols even if the linker is creating a shared library
--no-whole-archive	Restores the default behavior of loading archive members
--noinhibit-exec	Retain the executable output file whenever it is still usable
--non_shared	Do not link against shared libraries
--nopic	Do not create a position independent executable
--nostdlib	Only search directories specified on the command line
-N	Set the text and data sections to be readable and writable
--oformat <format>	Specify the binary format for the output object file
--omagic	Set the text and data sections to be readable and writable
--opt-remarks-filename <value>	YAML output file for optimization remarks
--opt-remarks-with-hotness	Include hotness informations in the optimization remarks file
--orphan-handling=<value>	Control how orphan sections are handled when linker script used
--orphan-handling <value>	Control how orphan sections are handled when linker script used
--output=<value>	Path to file to write output
--output <value>	Path to file to write output

Option	Description
-O <value>	Optimize output file size
-o <path>	Path to file to write output
--pack-dyn-relocs=<format>	Pack dynamic relocations in the given format (none or android)
--pic-executable	Create a position independent executable
--pie	Create a position independent executable
--plugin-opt=<value>	specifies LTO options for compatibility with GNU linkers
--plugin-opt <value>	specifies LTO options for compatibility with GNU linkers
--print-gc-sections	List removed unused sections
--print-map	Print a link map to the standard output
-q	Generate relocations in output
--relax	Perform linker relaxation to reduce code size
--relocatable	Create relocatable object file
--reproduce=<value>	Dump linker invocation and input files for debugging
--reproduce <value>	Dump linker invocation and input files for debugging
--retain-symbols-file=<file>	Retain only the symbols listed in the file
--retain-symbols-file <file>	Retain only the symbols listed in the file
--rpath=<value>	Add a DT_RUNPATH to the output
--rpath <value>	Add a DT_RUNPATH to the output
--rsp-quoting=<value>	Quoting style for response files. Values supported: windows posix
--rsp-quoting <value>	Quoting style for response files. Values supported: windows posix
-R <value>	Alias to --rpath or --just-symbols
-r	Create relocatable object file
--script=<value>	Read linker script
--script <value>	Read linker script
--section-start <address>	Set address of section
--shared	Build a shared object
--soname=<value>	Set DT_SONAME
--soname <value>	Set DT_SONAME

Option	Description
--sort-section=<value>	Specifies sections sorting rule when linkerscript is used
--sort-section <value>	Specifies sections sorting rule when linkerscript is used
--start-lib	Start a grouping of objects that should be treated as if they were together in an archive
--static	Do not link against shared libraries
--strip-all	Strip all symbols
--strip-debug	Strip debugging information
--symbol-ordering-file <value>	Layout sections in the order specified by symbol file
--sysroot=<value>	Set the system root
--sysroot <value>	Set the system root
-S	Strip debugging information
-s	Strip all symbols
--Tbss=<value>	Same as --section-start with .bss as the sectionname
--Tbss <value>	Same as --section-start with .bss as the sectionname
--Tdata=<value>	Same as --section-start with .data as the sectionname
--Tdata <value>	Same as --section-start with .data as the sectionname
--thinlto-cache-dir=<value>	Path to ThinLTO cached object file directory
--thinlto-cache-policy <value>	Pruning policy for the ThinLTO cache
--thinlto-jobs=<value>	Number of ThinLTO jobs
--threads	Run the linker multi-threaded
--trace-symbol=<value>	Trace references to symbols
--trace-symbol <value>	Trace references to symbols
--trace	Print the names of the input files
--Ttext-segment=<value>	Same as --section-start with .text as the sectionname
--Ttext-segment <value>	Same as --section-start with .text as the sectionname
--Ttext=<value>	Same as --section-start with .text as the sectionname

Option	Description
--Ttext <value>	Same as --section-start with .text as the sectionname
-T <value>	Read linker script
-t	Print the names of the input files
--undefined=<value>	Force undefined symbol during linking
--undefined <value>	Force undefined symbol during linking
--unresolved-symbols=<value>	Determine how to handle unresolved symbols
--unresolved-symbols <value>	Determine how to handle unresolved symbols
-u <value>	Force undefined symbol during linking
--verbose	Verbose mode
--version-script=<value>	Read a version script
--version-script <value>	Read a version script
--version	Display the version number and exit
-V	Display the version number and exit
-v	Display the version number
--warn-common	Warn about duplicate common symbols
--warn-unresolved-symbols	Report unresolved symbols as warnings
--whole-archive	Force load of all members in a static library
--wrap=<symbol>	Use wrapper functions for symbol
--wrap <symbol>	Use wrapper functions for symbol
-X	Delete temporary local symbols
-x	Delete all local symbols
-y <value>	Trace references to symbols
-z <option>	Linker option extensions

9. Select "Andes Assembler > General" to specify the paths to the header files used by the software project and the assembler, as shown in Figure 2-31.

Figure 2-31 Select Andes Assembler > General

The assembler options for RDS software are as shown in Table 2-12.

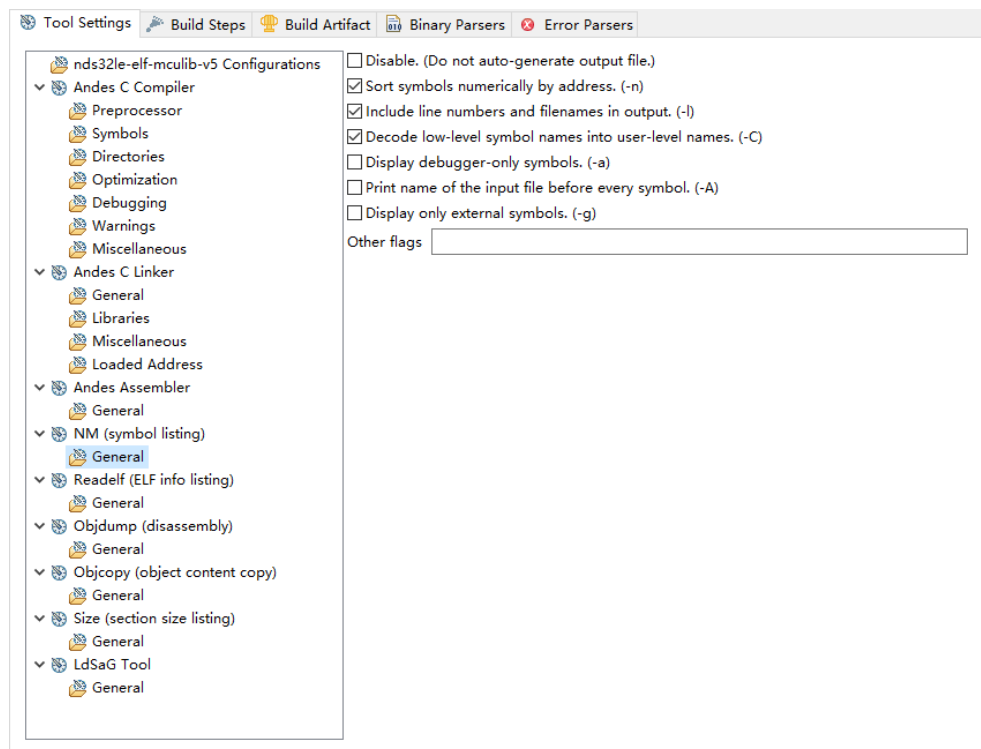
Table 2-12 Assembler Options

Option	Description
-a[sub-option...]	turn on listings Sub-options [default hls]: c omit false conditionals d omit debugging directives g include general info h include high-level source l include assembly m include macro expansions n omit forms processing s include symbols =FILE list to FILE (must be last sub-option)
--alternate	initially turn on alternate macro syntax
--compress-debug-sections[={none zlib zlib-gnu zlib-gabi}]	compress DWARF debug sections using zlib
--nocompress-debug-sections	don't compress DWARF debug sections [default]
-D	produce assembler debugging messages

Option	Description
--debug-prefix-map OLD=NEW	map OLD to NEW in debug information
--defsym SYM=VAL	define symbol SYM to given value
--execstack	require executable stack for this object
--noexecstack	don't require executable stack for this object
--size-check=[error warning]	ELF .size directive check (default --size-check=error)
--elf-stt-common=[no yes]	generate ELF common symbols with STT_COMMON type
--sectname-subst	enable section name substitution sequences
-f	skip whitespace and comment preprocessing
-g --gen-debug	generate debugging information
--gdwarf-2	generate DWARF2 debugging information
--gdwarf-sections	generate per-function section names for DWARF line information
--hash-size=<value>	set the hash table size close to <value>
--help	show this message and exit
--target-help	show target specific options
-I DIR	add DIR to search list for .include directives
-J	don't warn about signed overflow
-K	warn when differences altered for long displacements
-L,--keep-locals	keep local symbols (e.g. starting with 'L')
-M,--mri	assemble in MRI compatibility mode
--MD FILE	write dependency information in FILE (default none)
-nocpp	ignored
-no-pad-sections	do not pad the end of sections to alignment boundaries
-o OBJFILE	name the object-file output OBJFILE (default a.out)
-R	fold data section into text section
--reduce-memory-overheads	prefer smaller memory use at the cost of longer assembly times
--statistics	print various measured statistics from execution
--strip-local-absolute	strip local absolute symbols
--traditional-format	Use same format as native assembler when possible
--version	print assembler version number and exit
-W --no-warn	suppress warnings
--warn	don't suppress warnings
--fatal-warnings	treat warnings as errors

Option	Description
-Z	generate object file even after errors
--listing-lhs-width	set the width in words of the output data column of the listing
--listing-lhs-width2	set the width in words of the continuation lines of the output data column; ignored if smaller than the width of the first line
--listing-rhs-width	set the max width in characters of the lines from the source file
--listing-cont-lines	set the maximum number of continuation lines used for the output data column of the listing
-fpic	generate position-independent code
-fPIC	same as -fpic
-fno-pic	don't generate position-independent code (default)
-march=ISA	set the RISC-V architecture
-mabi=ABI	set the RISC-V ABI
-mno-16-bit	don't generate rvc instructions
-matomic	enable atomic extension
-mace	Support user defined instruction extension
-O1	optimize for performance
-Os	optimize for space
	enable dsp extension

10. Select "NM (symbol listing) > General" to output the symbol listing to the "symbol.txt" file, as shown in Figure 2-32.

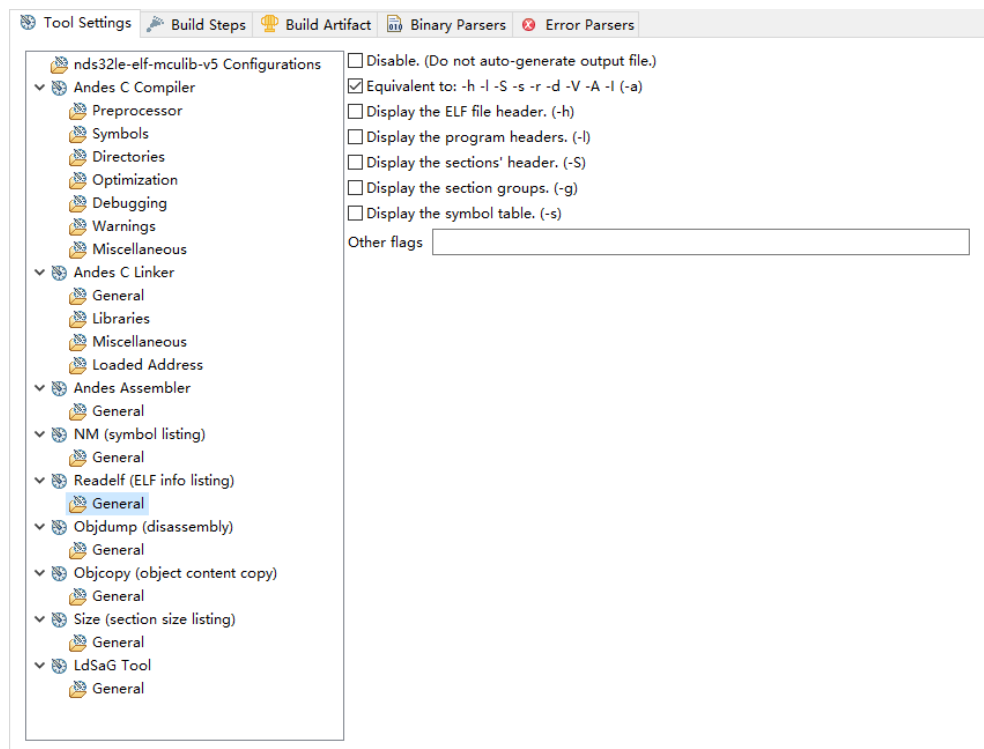
Figure 2-32 Select NM (symbol listing) > General

The General option of NM (symbol listing) is as shown in Table 2-13.

Table 2-13 General Option of NM (symbol listing)

Option	Description
Disable. (Do not auto-generate output file.)	Disable the symbol listing output file
Sort symbols numerically by address. (-n)	Sort symbol numerically by address
Include line numbers and filenames in output. (-l)	Include line numbers and filenames in output
Decode low-level symbol names into user-level names. (-C)	Decode low-level symbol names into user-level names
Display debugger-only symbols. (-a)	Display debugger-only symbols
Print name of the input file before every symbol. (-A)	Print name of the input file before every symbol
Display only external symbols. (-g)	Display only external symbols

11. Select "Readelf (ELF info listing) > General" to output the ELF listing to the "readelf.txt" file, as shown in Figure 2-33.

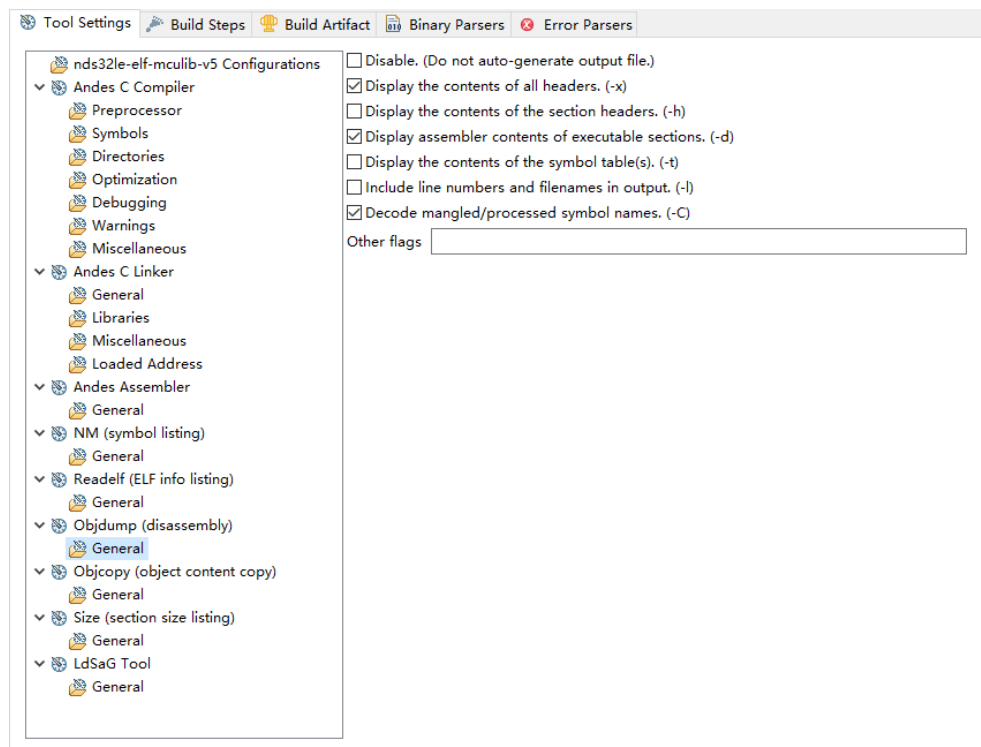
Figure 2-33 Select Readelf (ELF info listing) > General

The General option of Readelf (ELF info listing) is as shown in Table 2-14.

Table 2-14 General Option of Readelf (ELF info listing)

Configuration	Description
Disable. (Do not auto-generate output file.)	Disable the ELF listing output file
Display the ELF file header. (-h)	Display the ELF file header
Display the program headers. (-l)	Display the program headers
Display the sections' header. (-S)	Display the sections' header
Display the section groups. (-g)	Display the section groups
Display the symbol table. (-s)	Display the symbol table

12. Select "Objdump (disassembly) > General" to output the disassembly information to the "objdump.txt" file, as shown in Figure 2-34.

Figure 2-34 Select Objdump (disassembly) > General

The General options of Objdump (disassembly) are as shown in Table 2-15.

Table 2-15 General Option of Objdump (disassembly)

Configuration	Description
Disable. (Do not auto-generate output file.)	Disable the disassembly output file
Display the contents of all headers. (-x)	Display the contents of all headers
Display the contents of the section headers. (-h)	Display the contents of the section headers
Display assembler contents of executable sections. (-d)	Display assembler contents of executable sections
Display the contents of the symbol table(s). (-t)	Display the contents of the symbol table(s)
Include line numbers and filenames in output. (-l)	Include line numbers and filenames in output
Decode mangled/processed symbol names. (-C)	Decode mangled/processed symbol names

13. Select "Objcopy (object content copy) > General" to copy the content

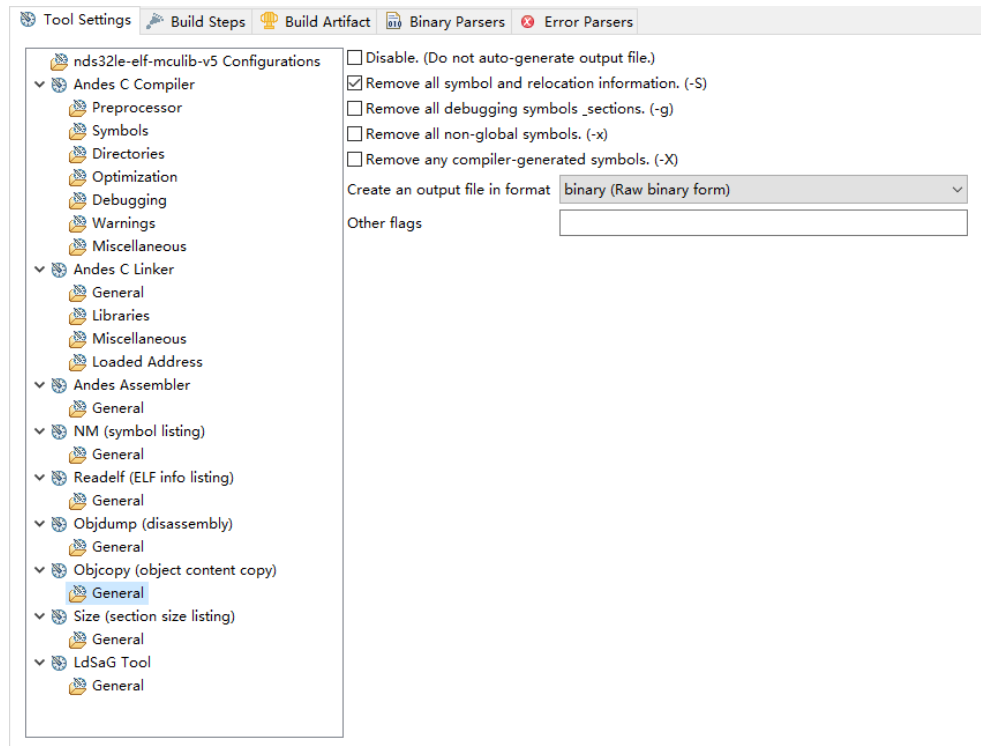
of the target file to another file, which is generally used to generate a Binary file, as shown in Figure 2-35.

For example:

Disable. (Do not auto-generate output file.): Disable

Create an output file in format: binary (Raw binary form)

Figure 2-35 Select Objcopy (object content copy) > General



The General options of Objcopy (object content copy) are as shown in Table 2-16.

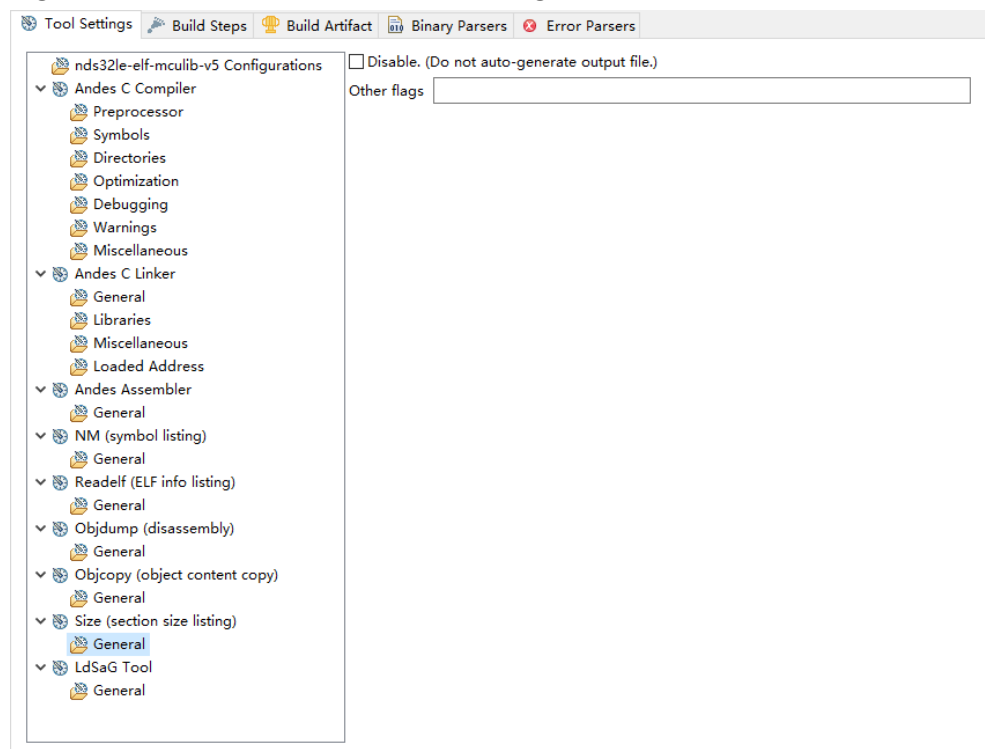
Table 2-16 General Options of Objcopy (object content copy)

Option	Description
Disable. (Do not auto-generate output file.)	Disable the Object output file
Remove all symbol and relocation information. (-S)	Remove all symbol and relocation information
Remove all debugging symbols _sections. (-g)	Remove all debugging symbols sections
Remove all non-global symbols. (-x)	Remove all non-global symbols
Remove any compiler-generated symbols. (-X)	Remove any compiler-generated symbols
Create an output file in format	Specify the format of the target file, and the following three industry-

Option	Description
	standard formats are supported: binary (Raw binary form) ihex (Intel Hex) srec (Motorola S-record)

14. Select "Size (section size listing) > General" to output the section size listing to the ".PHONY.size" file, as shown in Figure 2-36.

Figure 2-36 Select Size (section size listing) > General



The General option of Size (section size listing) is as shown in Table 2-17.

Table 2-17 General Option of Size (section size listing)

Option	Description
Disable. (Do not auto-generate output file.)	Disable the size output file

15. Select "LdSaG Tool > General" to specify the SaG file used for generating the linker script file for the software project, as shown in Figure 2-37.

For example:

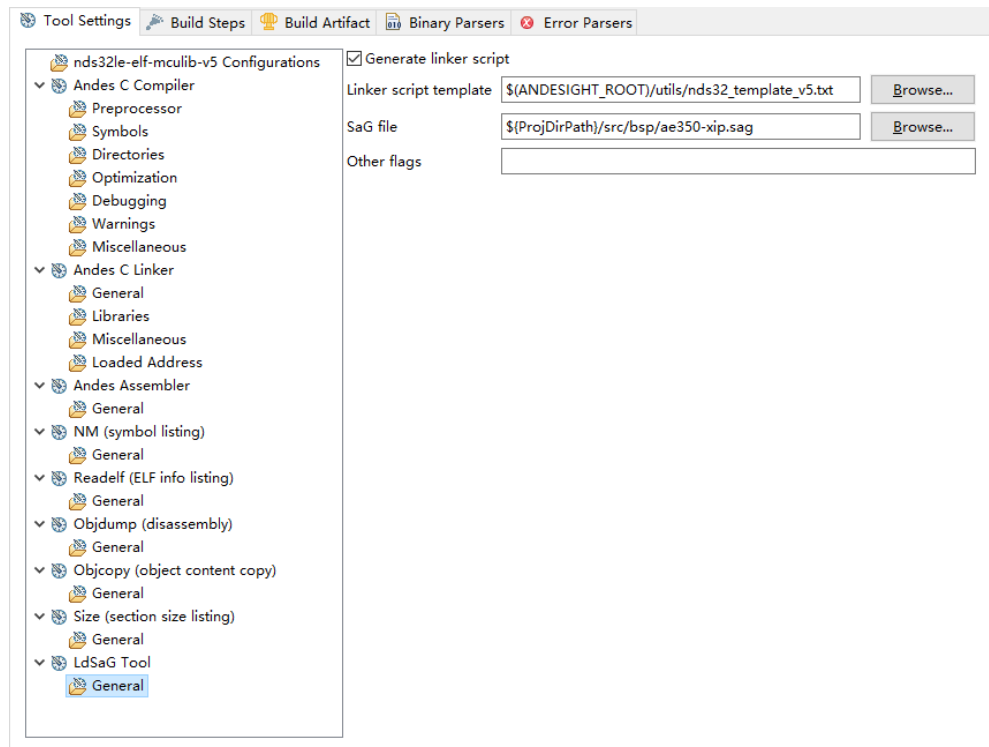
Generate linker script: Enable

Linker script template:

`$(ANDESIGHT_ROOT)/utils/nds32_template_v5.txt;`

SaG file: `${ProjDirPath}/src/bsp/sag/ae350-ddr.sag`

Figure 2-37 Select LdSaG Tool > General



The General options of LdSaG Tool are as shown in Table 2-18.

Table 2-18 General Options of LdSaG Tool

Option	Description
Generate linker script	The "PROJECT\BUILD_CONFIG\output" path generates a linker script file, and the "Andes C Linker > General > Linker Script (-T):" option is automatically set to "\$(LDSAG_OUT)".
Linker script template	Specify the template used to generate the linker script file, and Gowin RiscV_AE350_SOC defaults to "nds32_template_v5.txt"
SaG file	Specify a SaG file that describes the memory map for the executable image

- Click "Apply and Close" to complete the build configuration for the software project.

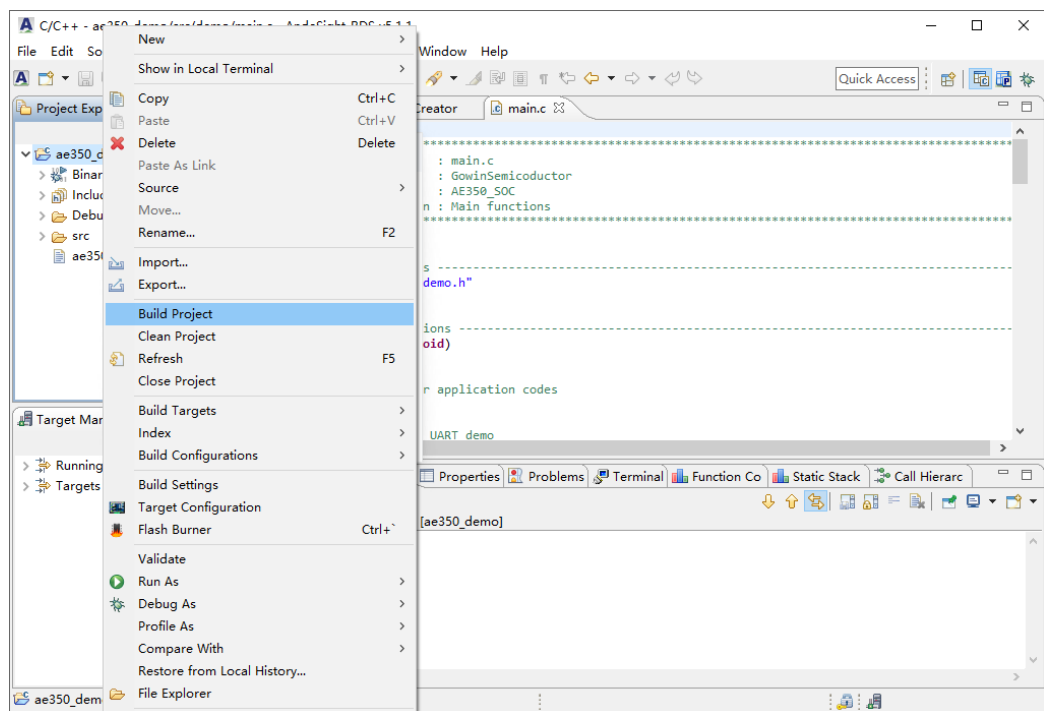
2.5 Build a Software Project

Building a software project refers to the conversion of source code into executables for a specified target system.

Refer to the following steps to build a software project:

1. You can create the build system for the selected software project referring to 2.4 Create a Build System.
2. Click "🔧" (Build) on the RDS software toolbar, or right-click on the selected software project and select "Build Project" from the drop-down list to build the software project, as shown in Figure 2-38.

Figure 2-38 Build a Software Project



3. Verify the build results of the software project in the console view.
4. You can find the generated executable and other files in the path of "PROJECT\BUILD_CONFIG\output".

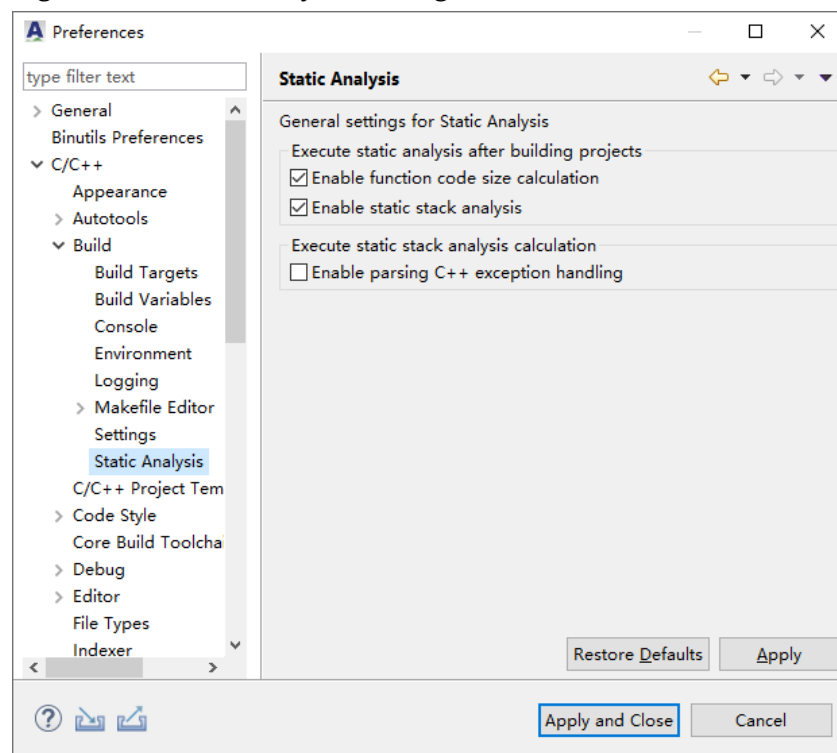
2.6 Static Analysis

The RDS software supports two types of static program analysis based on executables from selected software projects: function code size analysis and static stack analysis. Function code size analysis provides the analysis for code size of both individual function and all user-defined functions. Static stack analysis provides the analysis for stack size used by each function and the entire program.

The static analysis is performed automatically by the RDS software after building the selected software project. Therefore, before building the project, set the relevant properties to enable static analysis by following steps below:

1. Select "Window > Preferences" on the main menu of RDS software to open the "Preferences" dialog box.
2. In the "Preferences" dialog box, select "C/C++ > Build > Static Analysis".
3. In the "Static Analysis" dialog box, select "Enable function code size calculation" and/or "Enable static stack analysis" and click "Apply and Close", as shown in Figure 2-39.

Figure 2-39 Static Analysis Setting

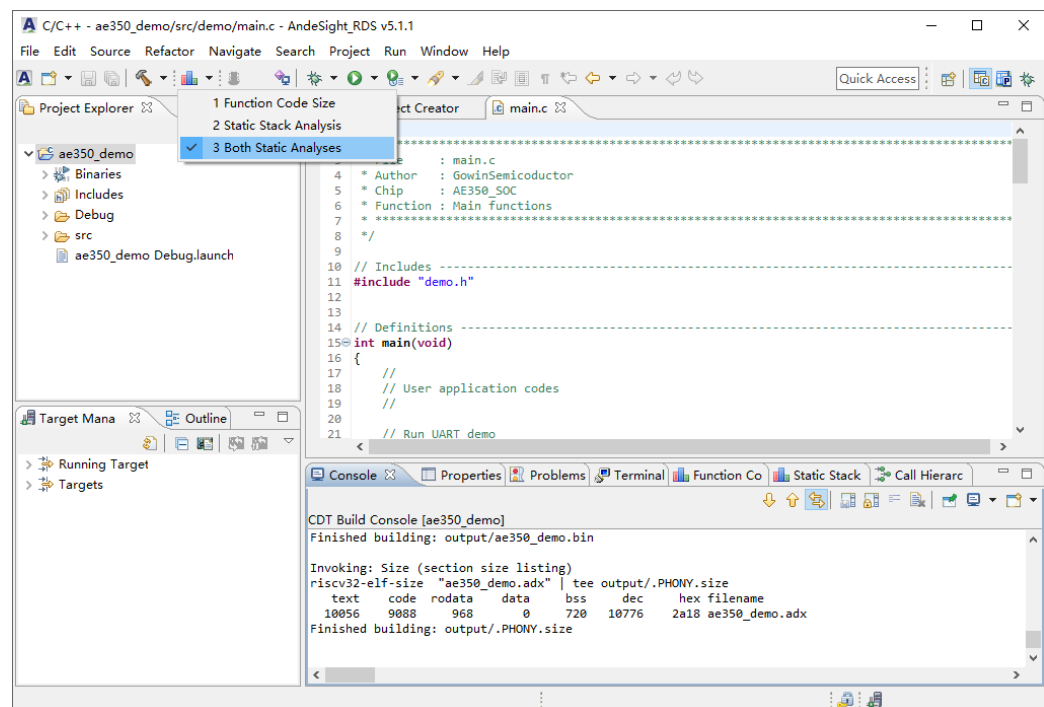


Note!

By default, the static analysis of C++ program does not include the code related to exception handling, if you want to analyze all the code, please select "Enable parsing C++ exception handling".

If static analysis is disabled by default, you can click "🔍" from the drop-down list on the RDS software toolbar after building the software project, and select "1 Function Code Size", "2 Static Stack Analysis" or "3 Both Static Analyses", as shown in Figure 2-40.

Figure 2-40 Enable Static Analysis



When you enable static analysis, a dialog box pops up to indicate that the analysis is in progress. When the static analysis is complete, the Function Code Size View and/or Static Stack Analysis View displays the results of the static analysis.

2.6.1 Function Code Size View

The Function Code Size View is used to view the information about the function size of the built software project. This view displays the size of both all user-defined functions and individual function, as shown in Figure 2-41.

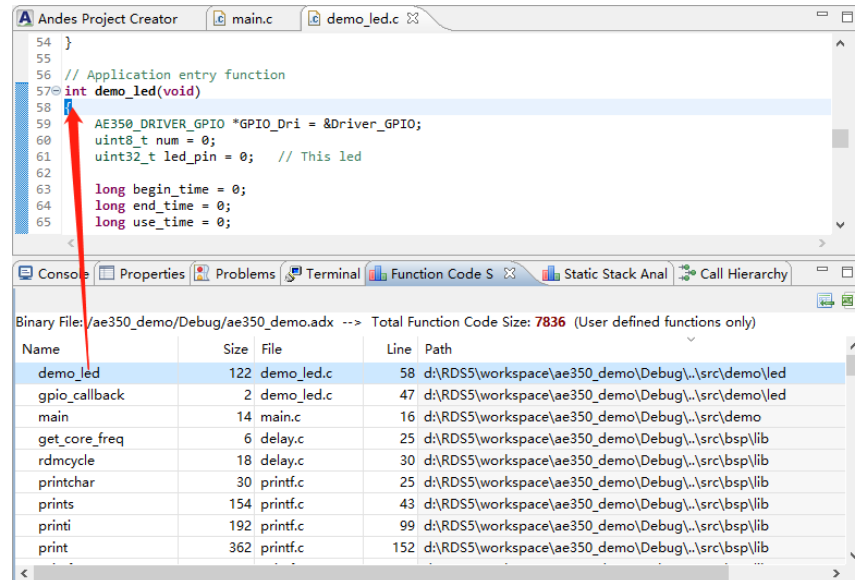
Figure 2-41 Function Code Size View

The screenshot shows the 'Function Code Size' view. The title bar indicates the total function code size is 7836 (User defined functions only). The table below lists the functions and their details.

Name	Size	File	Line	Path
demo_led	122	demo_led.c	58	d:\RDS5\workspace\ae350_demo\Debug\..\src\demo\led
gpio_callback	2	demo_led.c	47	d:\RDS5\workspace\ae350_demo\Debug\..\src\demo\led
main	14	main.c	16	d:\RDS5\workspace\ae350_demo\Debug\..\src\demo
get_core_freq	6	delay.c	25	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
rdmcycle	18	delay.c	30	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
printchar	30	printf.c	25	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
prints	154	printf.c	43	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
printi	192	printf.c	99	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
print	362	printf.c	152	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
printf	34	printf.c	251	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
simple_delay_ms	20	delay.c	55	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
uart_init	52	uart.c	29	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
outbyte	34	uart.c	87	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
uart_putc	18	uart.c	56	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib
time	46	delay.c	49	d:\RDS5\workspace\ae350_demo\Debug\..\src\bsp\lib

This view also displays the code details for each function, and you can sort the functions by name or size by clicking on the column headings, or you can double-click on a function to directly locate the file of its code, as shown in Figure 2-42.

Figure 2-42 Locate Function Code



Note!

This view does not provide the code size of functions that are optimized during the build process. In addition, the "Total Function Code Size" in the view only contains the ".text" segment, if you want the complete function size of the whole program, please sum the ".text segment" and ".nds32_init" segment.

2.6.2 Static Stack Analysis View

The Static Stack Analysis view is used to view the static stack size of the built software project, as shown in Figure 2-43. After generating the analysis results, this view lists the maximum stack size required for the entire program at the top, and the function call hierarchy lists the details of source code. Note that the stack size of interrupt programs and indirectly called functions are not counted in the maximum of required stack.

Figure 2-43 Static Stack Analysis View

Name	Used Stack	Self Stack Size	Graphical Stack Bar	File	Line	Path
> trap_entry	-	80	80 240	trap.c	49	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> gpio_control	-	32	32 160	gpio_ae350.c	185	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> i2c_irq_handler	-	16	16 160	i2c_ae350.c	1468	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
gpio_irq_handler	-	16	16 160	gpio_ae350.c	305	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> gp0_irq_handler	-	16	16 224	interrupt.c	24	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> spi_irq_handler	-	16	16 128	spi_ae350.c	1241	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> rtc_alarm_irq_handler	-	16	16 16	rtc_ae350.c	479	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> default_irq_handler	-	16	16 224	interrupt.c	24	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> dma_irq_handler	-	16	16 96	dma_ae350.c	506	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> reset_handler	-	16	16 272	reset.c	28	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> rtc_period_irq_handler	-	16	16 16	rtc_ae350.c	439	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> uart2_irq_handler	-	16	16 32	uart_ae350.c	1446	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> nmi_handler	-	16	16 32	wdt_ae350.c	259	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> uart1_irq_handler	-	16	16 32	uart_ae350.c	1444	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
< gpio_pinread	-	-	-	gpio_ae350.c	143	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'

For each function, this view provides the following stack information:

- **Self Stack Size:** The stack size used by the current function.
- **Used Stack:** The stack size already used before the current function is called.
- **Graphical Stack Bar:** Graphically display the cumulative stack usage of the current function. This section lists "Self Stack Size", "Used Stack", "sub-function max required stack", and "Max Stack Size". Three kinds of stack sizes are indicated by different colors: "■" means "Self Stack Size", "■" means "Used Stack", and "■" means "sub-function max required stack".

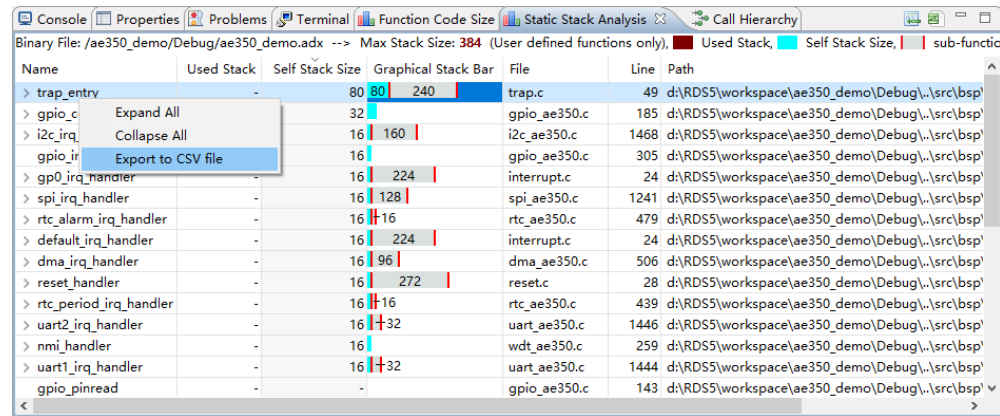
When the root function is first called, this view shows the stack information of the root function, including the entry function, interrupt program and indirectly called functions. Right-click on a root function and select "Expand All" in the drop-down list to expand to show all sub-functions and their stack information, as shown in Figure 2-44. The call relationship diagram of this stack is used to identify the execution path of the function that uses the most of the stack.

Figure 2-44 Expand All

Name	Used Stack	Self Stack Size	Graphical Stack Bar	File	Line	Path
> trap_entry	-	80	80 240	trap.c	49	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> gpio_c	-	32	32 160	gpio_ae350.c	185	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> i2c_irq	-	16	16 160	i2c_ae350.c	1468	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
gpio_ir	-	16	16 160	gpio_ae350.c	305	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> gp0_irq_handler	-	16	16 224	interrupt.c	24	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> spi_irq_handler	-	16	16 128	spi_ae350.c	1241	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> rtc_alarm_irq_handler	-	16	16 16	rtc_ae350.c	479	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> default_irq_handler	-	16	16 224	interrupt.c	24	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> dma_irq_handler	-	16	16 96	dma_ae350.c	506	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> reset_handler	-	16	16 272	reset.c	28	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> rtc_period_irq_handler	-	16	16 16	rtc_ae350.c	439	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> uart2_irq_handler	-	16	16 32	uart_ae350.c	1446	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> nmi_handler	-	16	16 32	wdt_ae350.c	259	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
> uart1_irq_handler	-	16	16 32	uart_ae350.c	1444	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'
< gpio_pinread	-	-	-	gpio_ae350.c	143	d:\RDS5\workspace\ae350_demo\Debug\.\src\bsp'

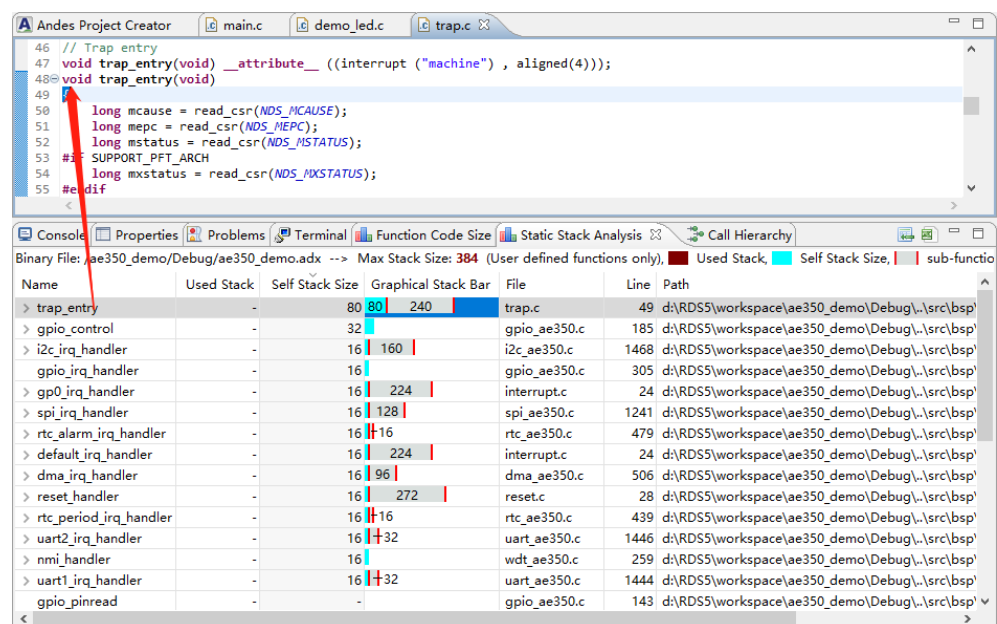
Right-click on a root function and select "Export to CSV file" in the drop-down list to export the stack information and its sub-function stack analysis to a .csv file, as shown in Figure 2-45.

Figure 2-45 Export to CSV File



Functions can be sorted by name, size, "Self Stack Size" or "Used Stack" by clicking on the column headings in the Static Stack Analysis view, or you can double-click on a function to locate it directly in the code file, as shown in Figure 2-46.

Figure 2-46 Locate Function Code



3 Target Management

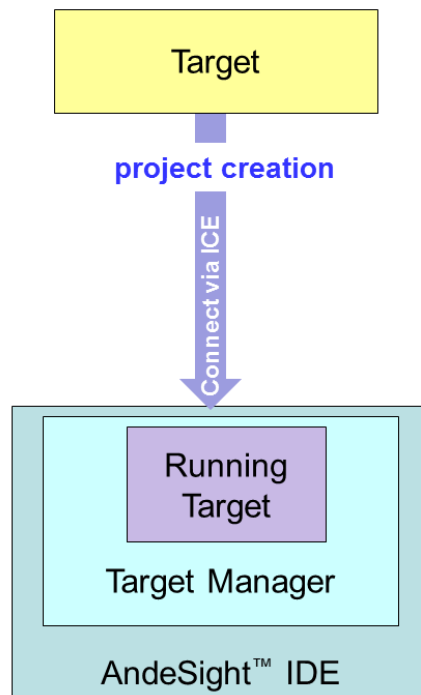
The RDS software manages Targets, Target Boards, and Target Systems through the Target Manager View; for the details, you can see [3.2 Target Manager](#).

The following steps summarize the typical approach to the target management for RDS software.

1. Create a target board ready to run the application program.
2. Create a software project (see [2.1 Create a New Project](#) for details) for the target board.
3. Build a software project (see [2.5 Build a Software Project](#) for details) for generating the executable file.
4. Start the target board, and run/debug/download/analyze the application program, managed by the Target Manager view

Typical target management process for RDS software is as shown in Figure 3-1.

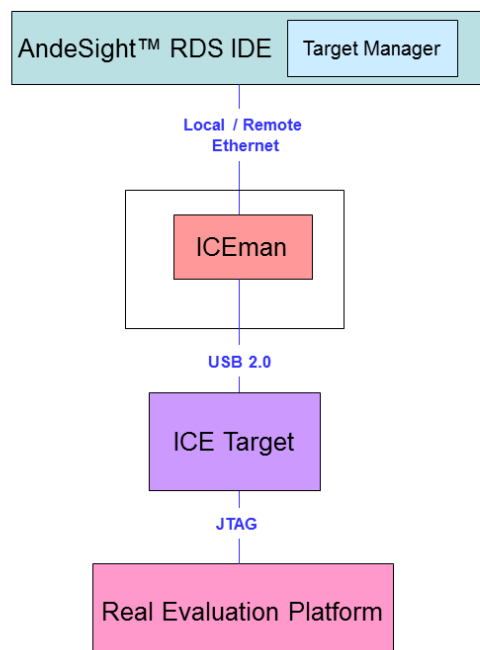
Figure 3-1 Target Management Flow



3.1 Create a Target System

The target system is a real evaluation platform that connects the local target board through an ICE target, such as Andes AICE-MINI+, Gowin FTDI USB programmer, or Olimex debugger. This includes the ICE target, target board, ICEman, and the host RDS software. Before creating the target system, connect the RDS software and the target board; for the connection, you can see Figure 3-2.

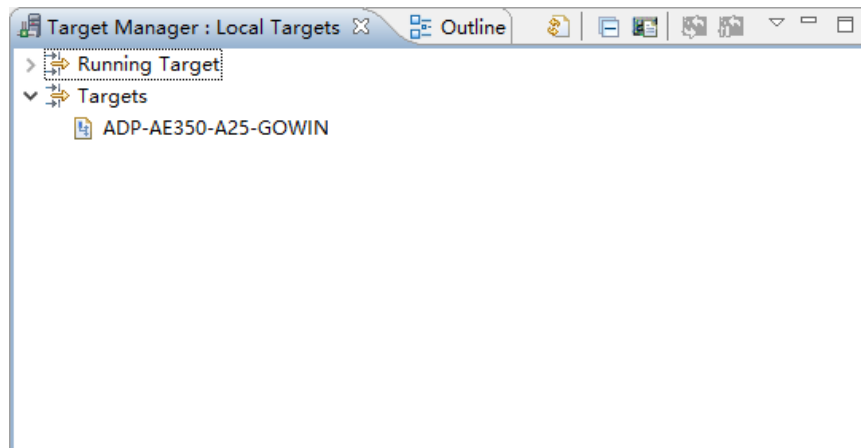
Figure 3-2 Connect the RDS Software and the Target Board



3.2 Target Manager

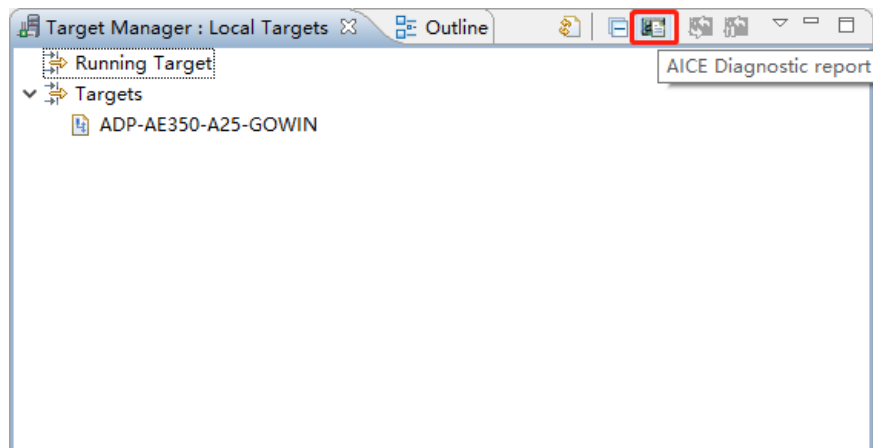
The RDS software uses the Target Manager to manage the connection between the RDS software and the target boards, as shown in Figure 3-3.

Figure 3-3 Target Manager



In the Target Manager, the "🔧" (AICE Diagnostic report) diagnoses the connectivity of the target system, as shown in Figure 3-4.

Figure 3-4 Diagnose the Target System



Click on "🔧" (AICE Diagnostic report), and configure ICE, Misc Argument, and Address in the Test Memory Access for diagnosing the target system.

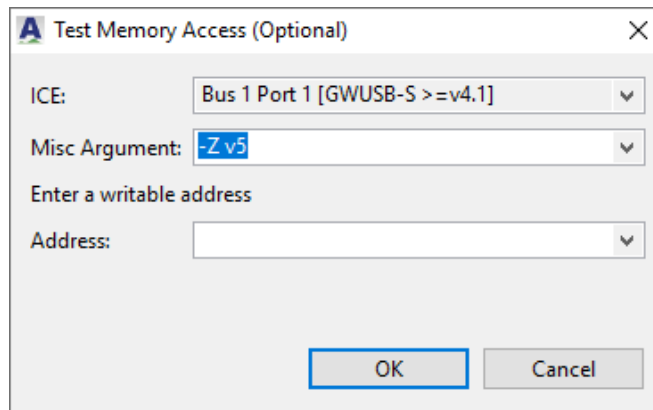
For Gowin FTDI USB programmer, the configuration for Test Memory Access is shown in Figure 3-5.

For example:

ICE: Bus 1 Port 1 [GWUSB-S >=v4.1]

Misc Argument: -Z v5

Figure 3-5 Test Memory Access (Gowin FTDI USB Programmer)



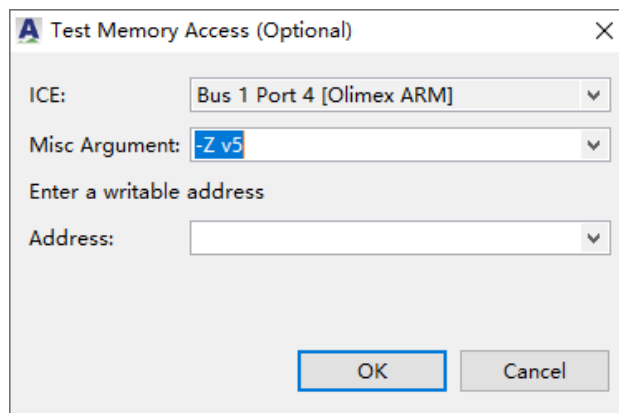
For Olimex debugger, the configuration for Test Memory Access is as shown in Figure 3-6.

For example:

ICE: Bus 1 Port 4 [Olimex ARM]

Misc Argument: -Z v5

Figure 3-6 Test Memory Access (Olimex Debugger)



Click "OK", and the diagnostic report for the target system is as follows:

```
Andes ICEman (OpenOCD) 5.1.1-g3a34fc9 (2022-04-22-14:27)
Burner listens on 2354
Telnet port: 4444
TCL port: 6666
JTAG frequency 10.000 MHz
There is 1 core in tap
target-0 = tap0_target_0 0x0 0x1
The core #0 listens on 1111.
```

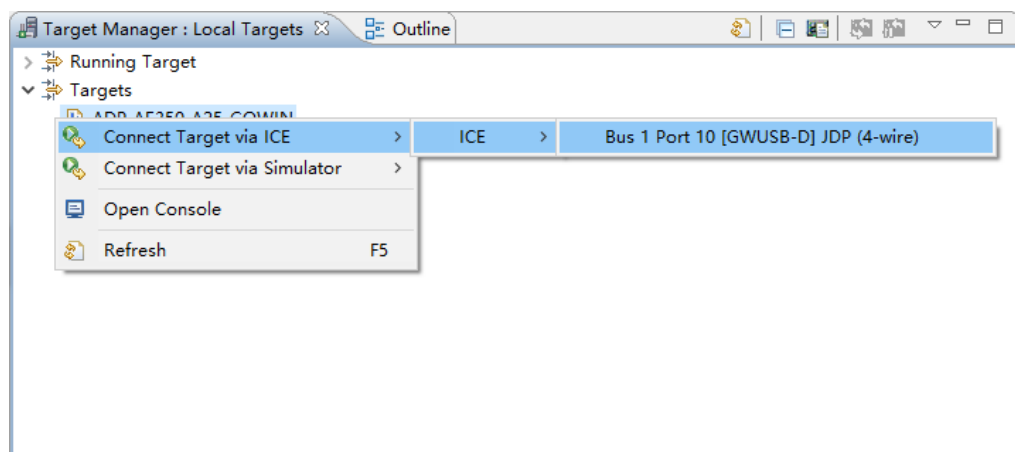
```
ICEman is ready to use.
== TEST target-0 ==
* Initializing Debug Module Interface
* Initializing Debug Module
Scanning number of harts
* test frequency
Setting adapter speed: 5000 kHz
JTAG frequency 5.000 MHz
Current adapter speed: 5000 kHz
JTAG frequency 10.000 MHz
Halting Hart 0
core0: target_xlen = 0x20
core0: mvendorid=0x31e
core0: marchid=0x80000a25 A25
core0: mimpid=0x160
core0: misa=0x4094312d
core0: dcsr=0x400006c3
core0: mnvec=0x80000000
core0: pc = 0x80001ba8
core0: debug_buffer_size=0x8
* Access GPRs testing
RV32 base ISA (32 GPRs)
* Abstractauto testing
* Single Step testing
* Check haltsum0 value
* Check haltsum1 value
Testing reset_and_halt_one_hart
execute ndmreset should not affect these registers:abstractcs,
abstractauto, program buffer and abstract data
clear abstractcs.cmderr(W1C) and abstractauto(0)
Hart 0 pc = 0x80000000
execute DM reset should reset these registers:abstractcs, abstractauto,
program buffer and abstract data
*****
Diagnostic Report
*****
(PASS) check that Debug Module (DM) is operational ...
(PASS) check JTAG/DTM connectivity ...
(PASS) check changing the JTAG frequency ...
(PASS) check GPRs access ...
(PASS) check ABSTRACTAUTO should cause COMMAND to run the
expected number of times ...
(PASS) check all harts single step ...
(PASS) check haltsum0 ...
```

```
(PASS) check haltsum1 ...
(PASS) check reset-and-halt ...
*****
```

In the Target Manager view, "Targets" displays the targets supported by the RDS software.

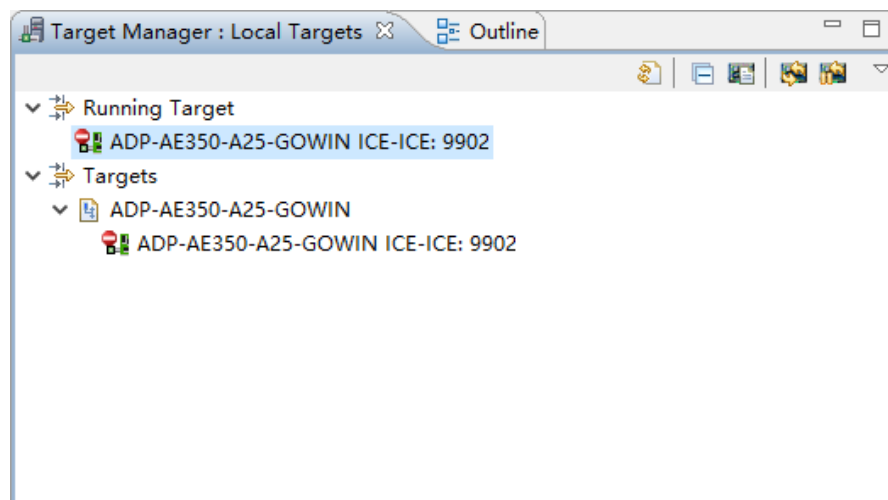
The Target Manager automatically establishes and manages the communication between the host RDS software and the target board in the background, or you can right-click on the selected target in "Targets" and select "Connect Target via ICE > ICE" in the drop-down list, and specify the connection configuration in the drop-down list, e.g. "Bus 1 Port 1 [GWUSB-S >=v4.1] JDP (4-wire)", and start the target system manually. In order to start the target system, you have to specify information such as the ICE debugger to be inserted and the debugging interface, as shown in Figure 3-7.

Figure 3-7 Create a Target System



After starting the target system, "Targets" displays the connection type, connection configuration, and port used for the GDB connection, as shown in Figure 3-8; and "Running Target" displays the same information.

Figure 3-8 Start Target System



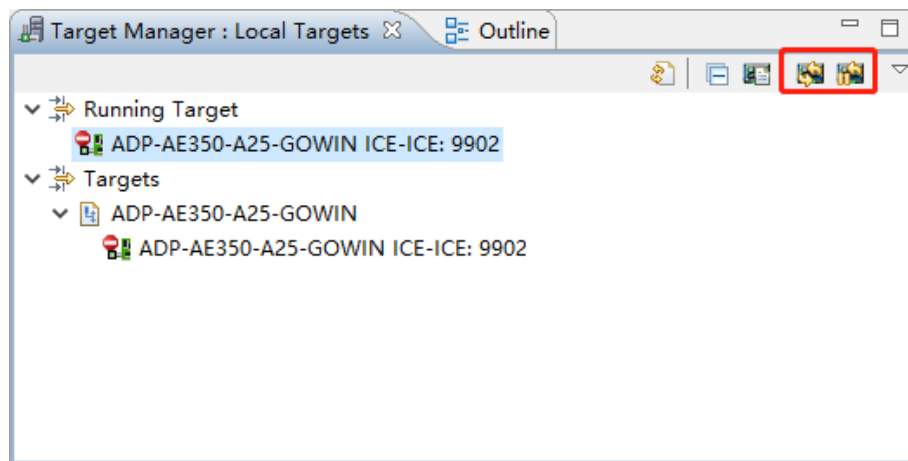
The Target Manager view will also show the connection status of each target system, including:

- Available Status "📁": The target system is in an available status and can be connected and developed.
- Started status "🟢": The target board is connected to the host RDS software, but has not yet loaded the program.
- Busy Status "🔴": The target system has been booted, the program has been loaded, and program development has begun.

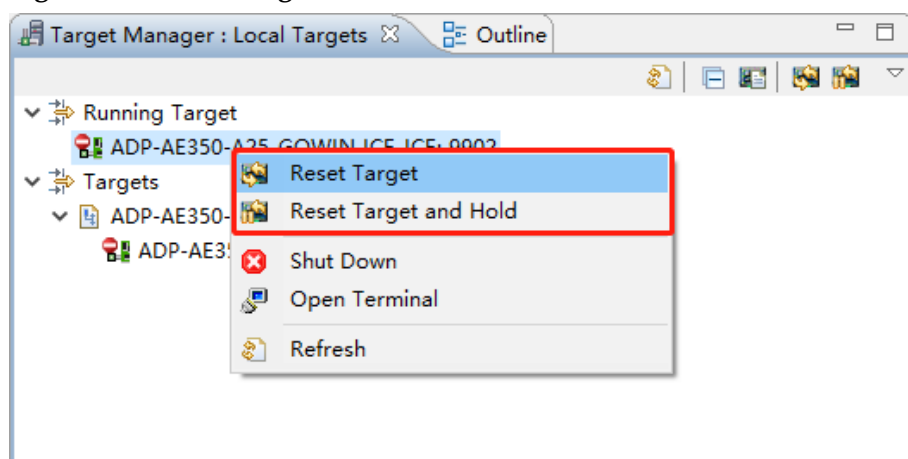
3.3 Reset Target Board

The RDS software provides a user interface for resetting the target board in the event of an exception.

To reset a running target board, in the Target Manager view, click "🔄" on the toolbar or "🔄" to reset target board and keep the target location at PC=0, as shown in Figure 3-9. If the reset is done during the debugging, the debug thread will be closed.

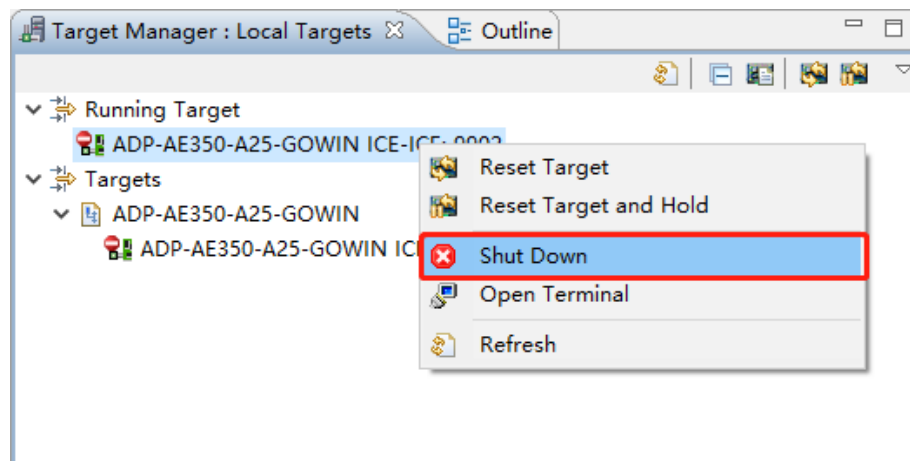
Figure 3-9 Reset Target Board

You can also reset the target board by right-clicking on the selected target and selecting "Reset Target" or "Reset Target and Hold" in the drop-down list, as shown in Figure 3-10.

Figure 3-10 Reset Target Board

3.4 Shut Down Target

In the Target Manager view, right-click on the selected target and select "Shut Down" in the drop-down list to shut down the target, as shown in Figure 3-11.

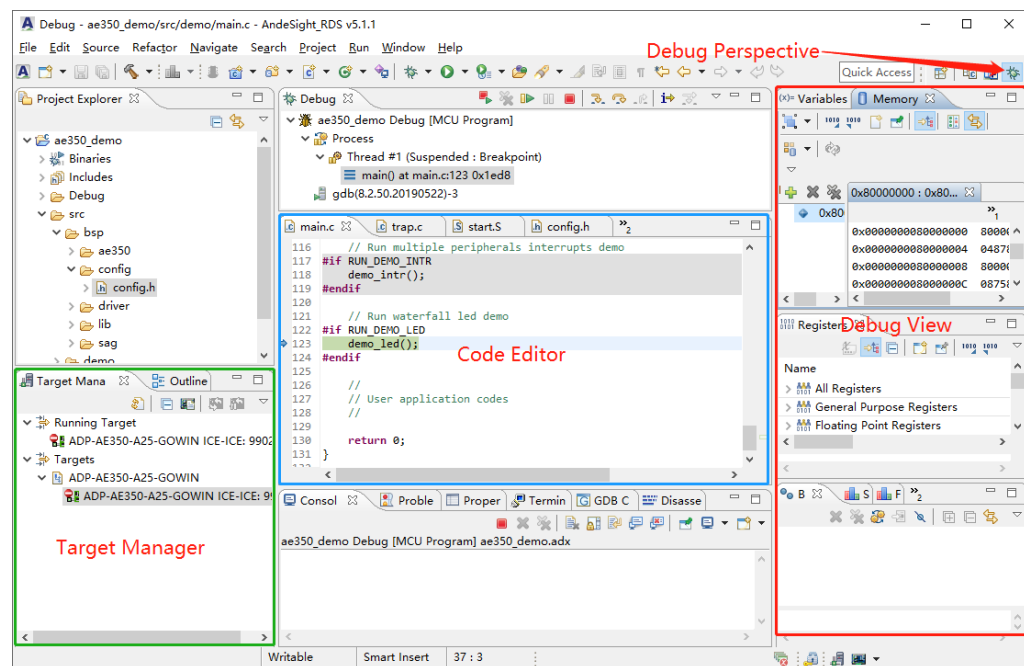
Figure 3-11 Shut Down Target

4 Debug a Software Project

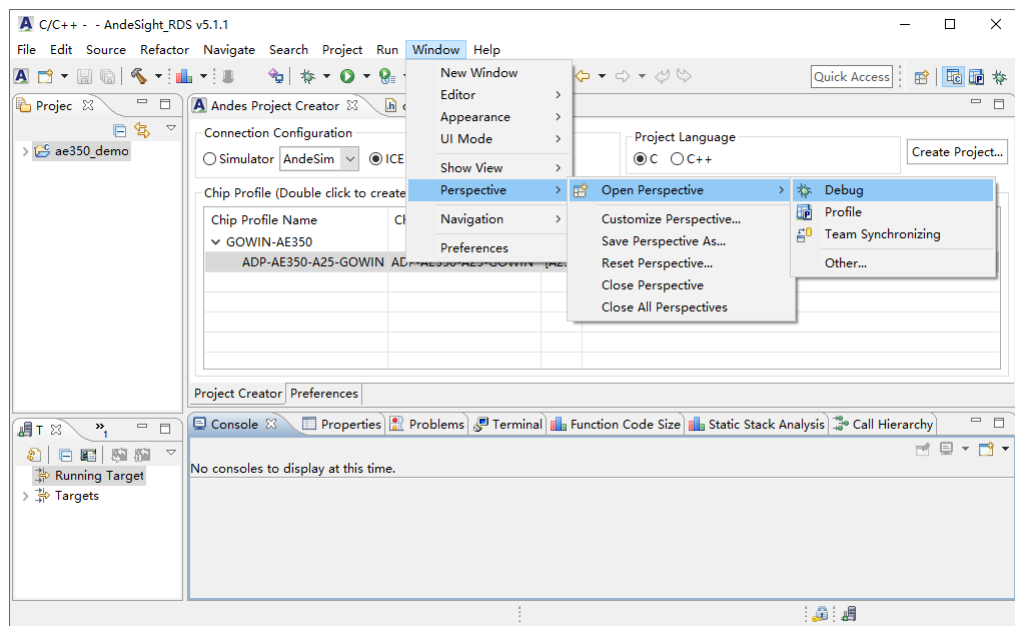
4.1 Debug Perspective

The Debug Perspective includes code editor view, target manager view, and multiple debugging views to control the execution process of the application and diagnose the correctness of each step, as shown in Figure 4-1.

Figure 4-1 Debug Perspective



After the Debug Session is started, the RDS software automatically enters the Debug Perspective, or you can manually enter the Debug Perspective by clicking " " on the Perspective toolbar of the RDS software, or you can select "Window > Perspective > Open Perspective > Debug" on the main menu of the RDS software, as shown in Figure 4-2.

Figure 4-2 Select Window > Perspective > Open Perspective > Debug

4.2 Debug View

When a debug session is started, the RDS software will automatically call the debug view. In order to call the specified debug view, select "Window > Show View> Other ..." on the main menu of RDS software, as shown in Figure 4-3. In the "Show View" dialog box, click the "Debug" drop-down arrow and select the debug view you want to display from the expanded list, as shown in Figure 4-4.

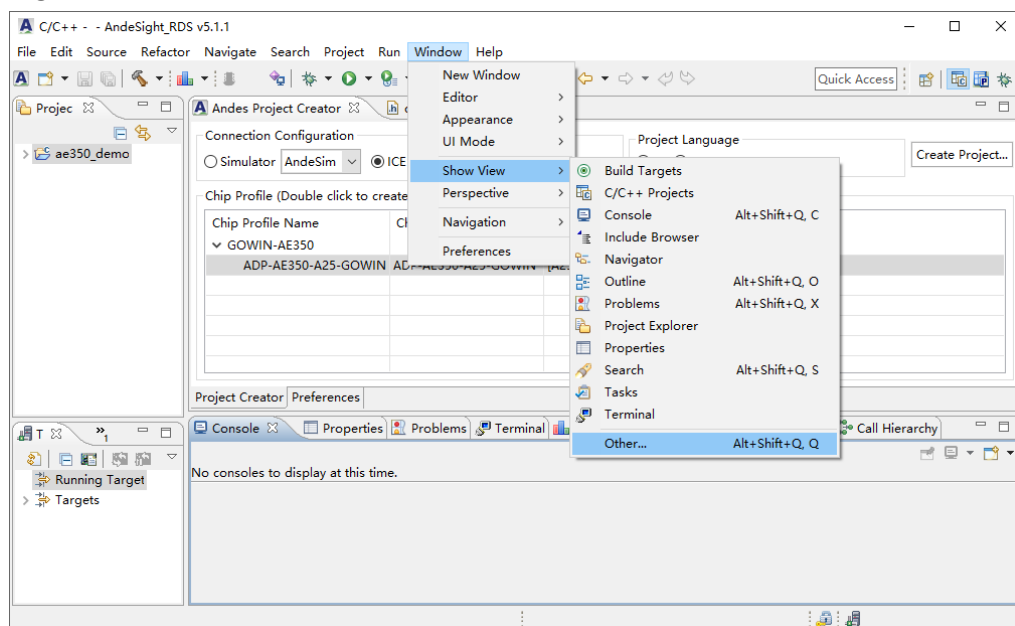
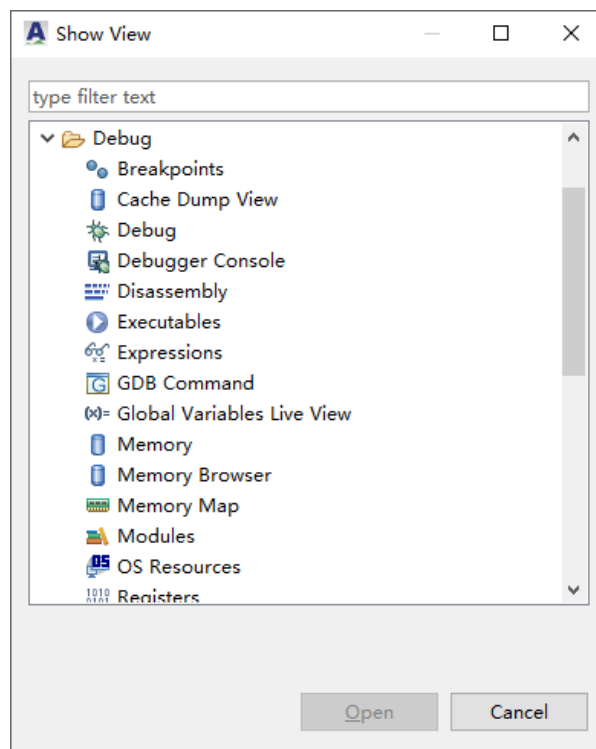
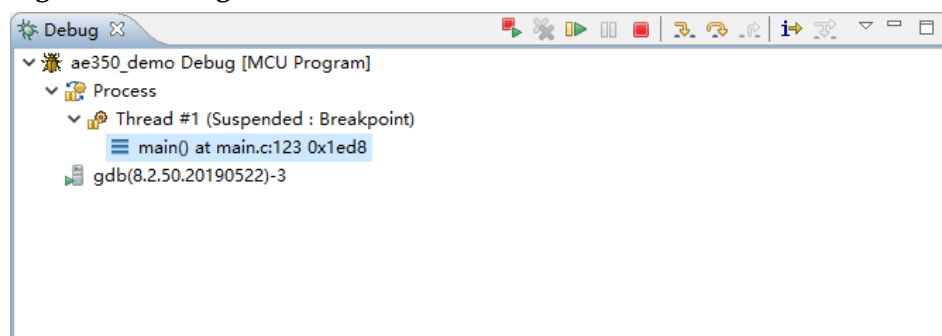
Figure 4-3 Select Window > Show View > Other...

Figure 4-4 Select Debug View

4.2.1 Debug View

The debug view is used to control the execution of an application, show the target system's stack when the program is paused or executed, as shown in Figure 4-5.

Figure 4-5 Debug View

The functions of the debug view toolbar include:

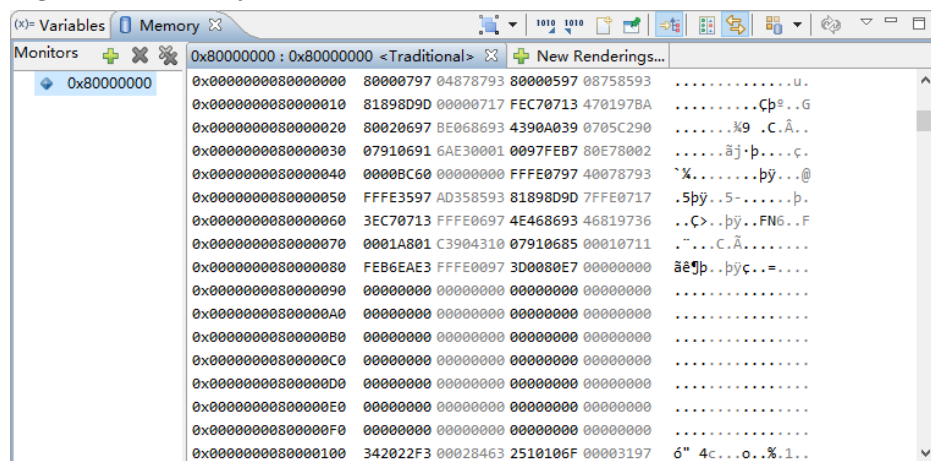
- " [Terminate and Relaunch]: Terminates this debugging thread and relaunch a new debugging thread.
- " [Resume]: Resumes execution of the suspended debug thread.
- " [Suspend]: Temporarily suspend the debugging thread.
- " [Terminate]: Terminates the debugging thread.

- " " (Step Into): Steps through the current statement of the debugging thread, and steps into the sub-function call.
- " " (Step Over): Steps through the current statement of the debugging thread, and steps over the sub-function call.
- " " (Step Return): Steps through the current statement of the debugging thread until it returns to its previous caller.
- " " (Instruction Stepping Mode): Enters the Disassembly View, and steps through the assembly instructions.

4.2.2 Memory View

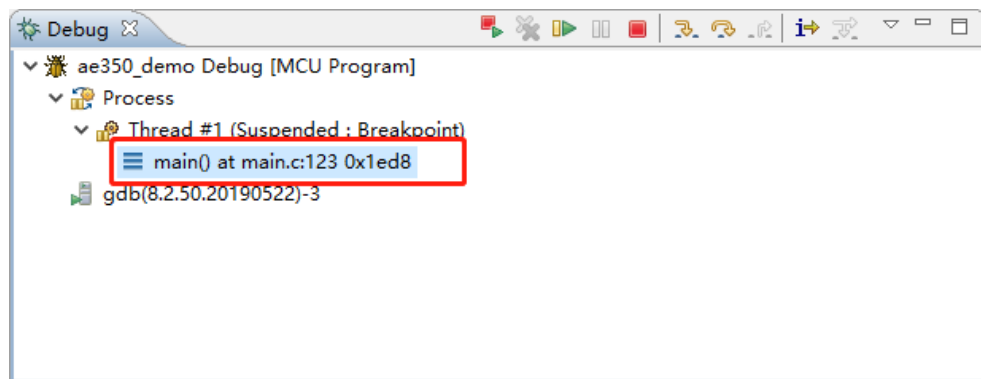
Memory View examines and modifies a program's memory in the form of a list of memory monitors, as shown in Figure 4-6. Each memory monitor is associated with a segment of memory with a specified base address or expression, and the data for each memory monitor can be displayed in a variety of predefined formats.

Figure 4-6 Memory View

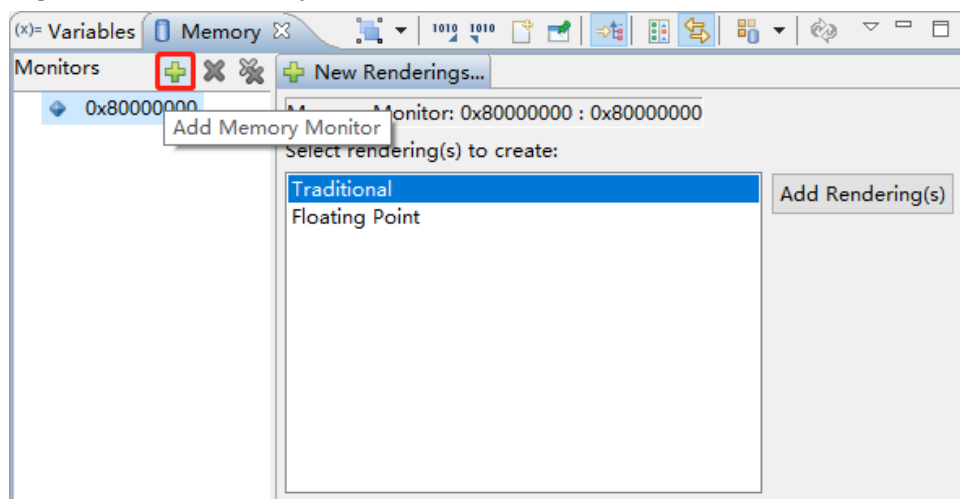
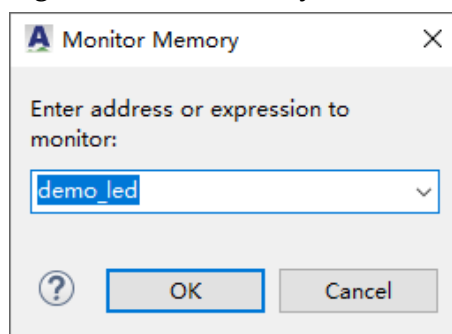


Create a Memory Monitor

1. After the program is paused, select a thread or stack in the debug view that you want to monitor, as shown in Figure 4-7.

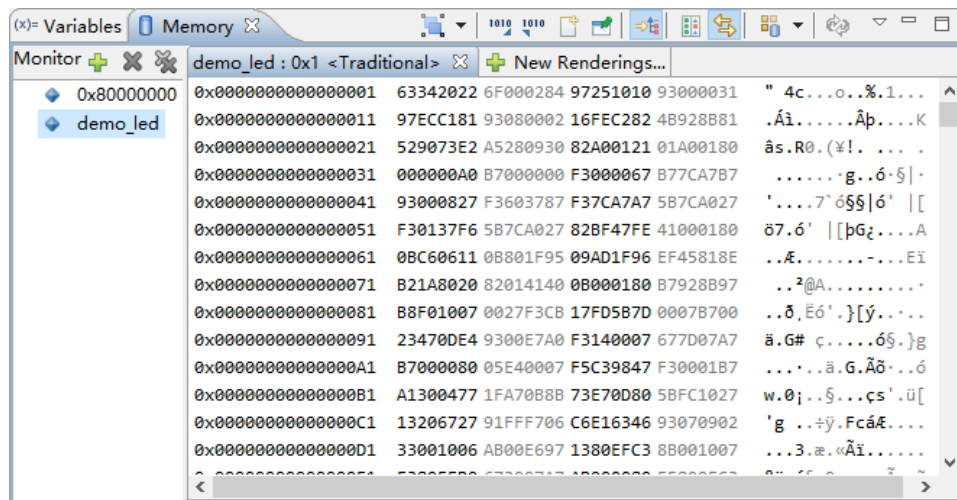
Figure 4-7 Select a Thread or Stack

2. In the Memory view, click "+" (Add Memory Monitor) in the Monitor panel, as shown in +. In the "Monitor Memory" dialog box, enter a memory address or expression that you want to view and click "OK", as shown in Figure 4-9.

Figure 4-8 Add Memory Monitor Interface**Figure 4-9 Add Memory Monitor**

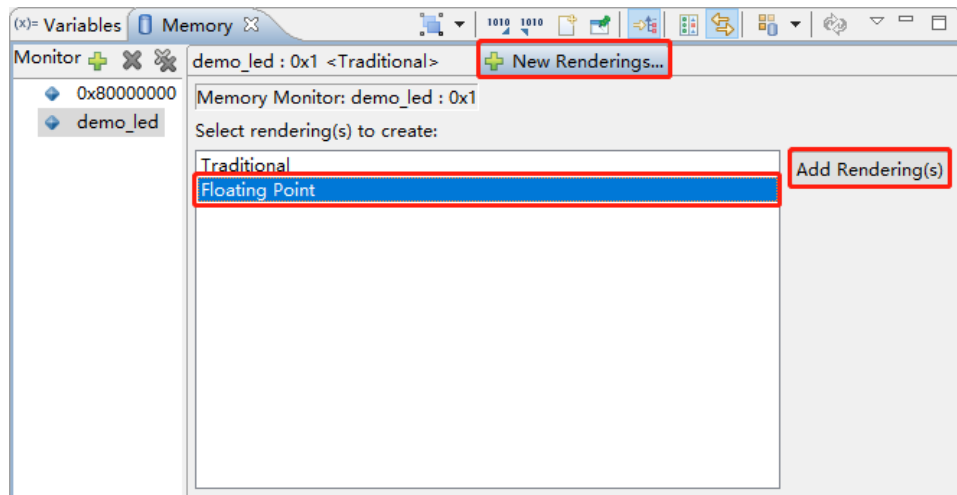
3. The memory view shows a traditional memory rendering by default, as shown in Figure 4-10.

Figure 4-10 Traditional Memory Rendering



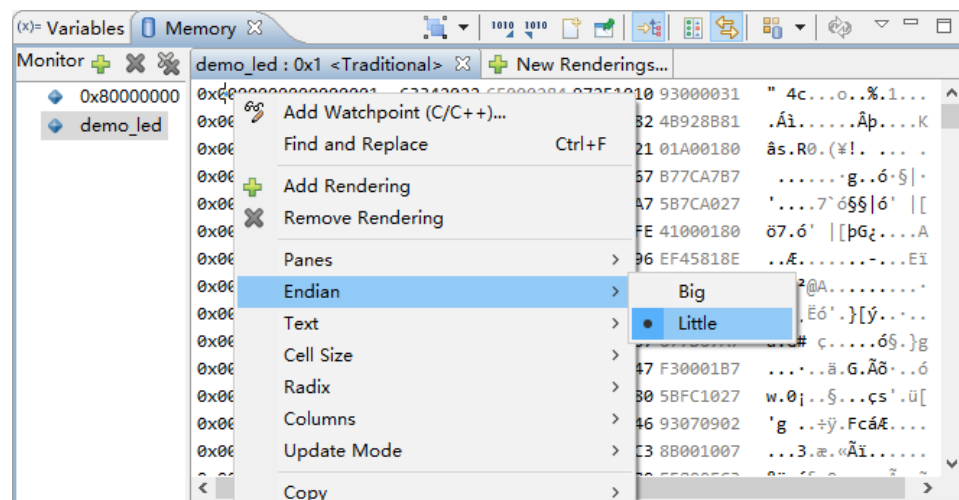
Click "New Renderings..." in the right panel of the Memory View, select "Floating Point", and click "Add Rendering(s)" to create a floating point memory rendering, as shown in Figure 4-11.

Figure 4-11 Create a Floating Point Memory Rendering



4. Right-click on Memory Rendering and select "Endian" from the drop-down list to change the byte order or addressable size of the data display, as shown in Figure 4-12.

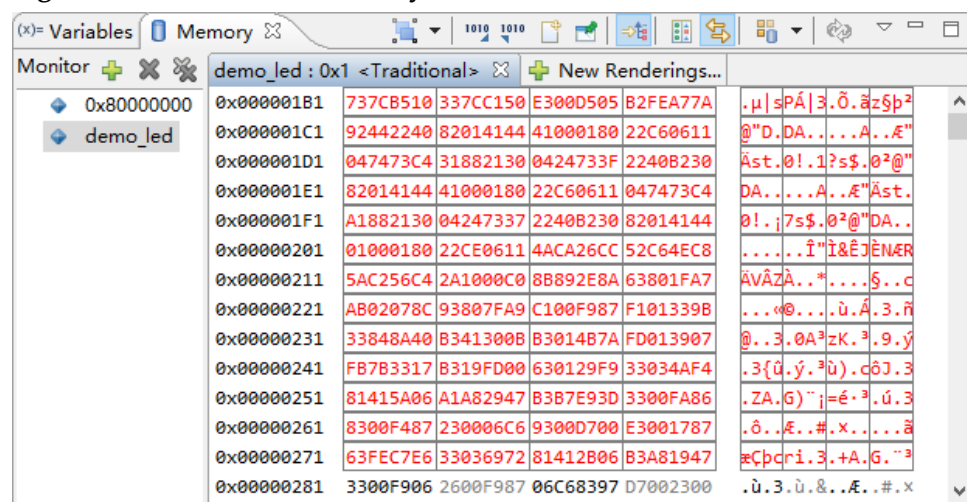
Figure 4-12 Byte Order Setting



It is possible to modify the memory value in each cell, but if an inappropriate value is entered, the program will die.

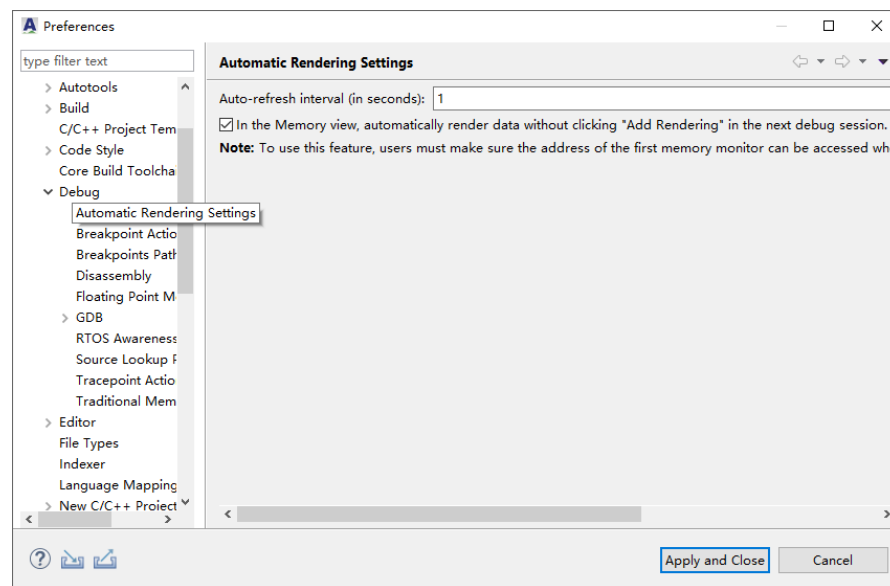
5. Deviations from the contents of the memory are marked in red during program execution, as shown in Figure 4-13.

Figure 4-13 Contents of Memory



6. If you want to automatically render data from the same memory monitor after restarting the debugging session, you have to select "Window > Preferences > C/C++ > Debug > Automatic Rendering Settings" on the main menu of the RDS software, and select "In the Memory view, automatically render data without clicking Add Rendering in the next debug session", as shown in Figure 4-14. Note that this feature only works when the program can access the address of the first memory monitor after stopping, otherwise a memory access error will occur.

Figure 4-14 Automatic Rendering Settings

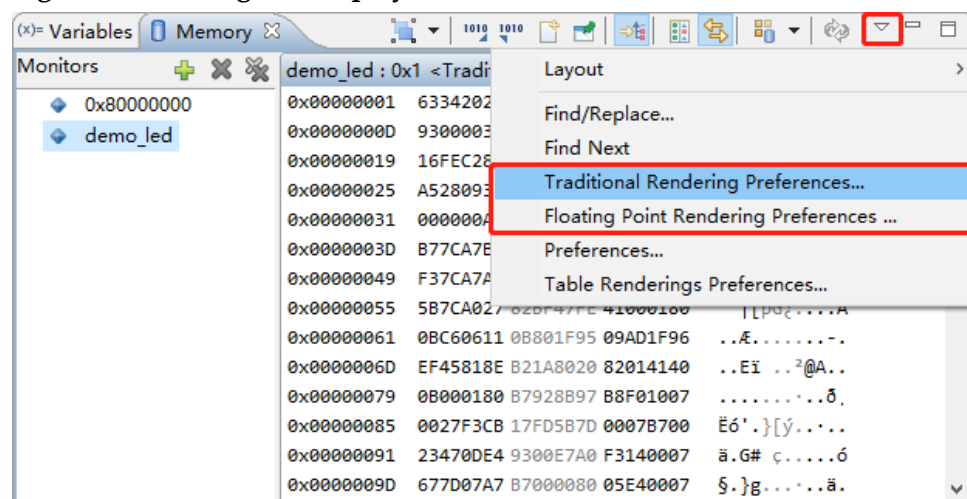


Setting the Display

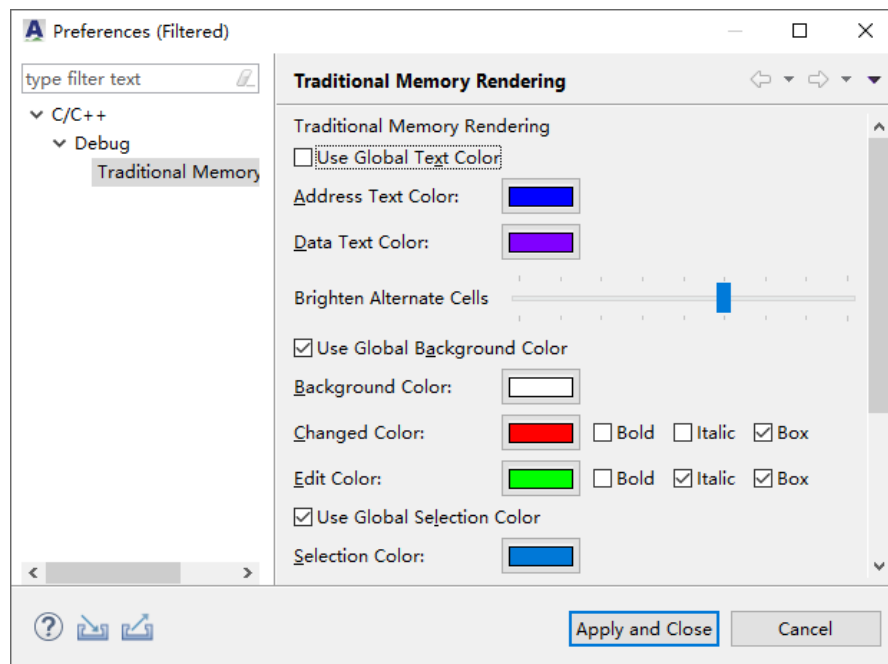
Memory rendering is displayed in black text by default. Refer to the following steps to distinguish the rendered data and addresses with different colors:

1. Click the "▼" (View Menu) on the toolbar of Memory View, and select "Traditional Rendering Preferences..." or "Floating Point Rendering Preferences..." from the drop-down list, as shown in Figure 4-15.

Figure 4-15 Setting the Display



2. In the "Preferences > C/C++ > Debug" dialog box, first uncheck "Use Global Text Color", and then define the text color for the address and data, as shown in Figure 4-16.

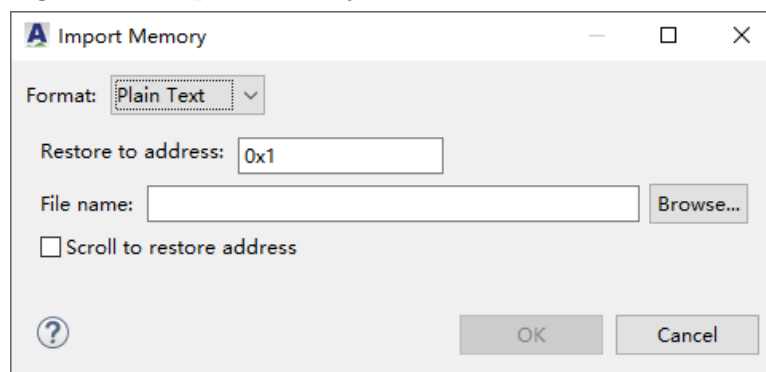
Figure 4-16 Traditional Memory Rendering Setting

3. Click "Apply and Close" to save the setting.

Memory View Toolbar

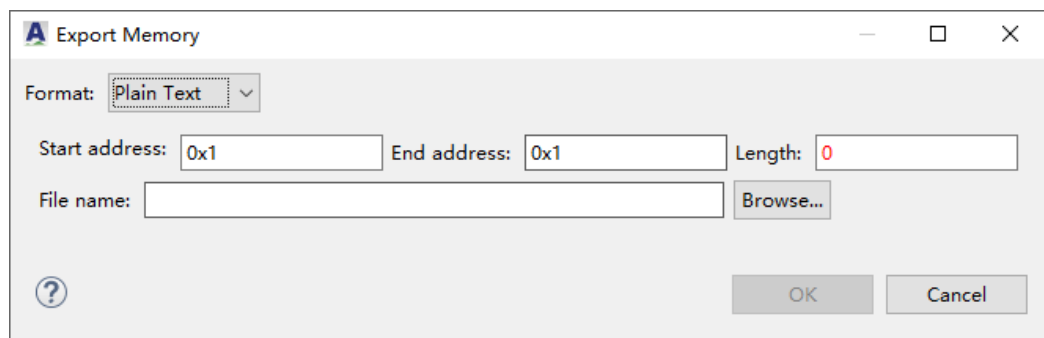
The functions of the Memory View toolbar include:

- "🔍" (switch source): Specifies the source of the memory contents, CPU or BUS. CPU mode provides the memory contents of the system or cache. BUS mode provides only the memory contents of system.
- "1010" (Import): Imports a specified set of memory values, as shown in Figure 4-17.

Figure 4-17 Import Memory Values

- "1010" (Export): Exports the memory values in the Memory View to a file, as shown in Figure 4-18.

Figure 4-18 Export Memory Values



The RDS software supports 3 formats for importing and exporting memory data: S-record, plain text and raw binary. Plain text and raw binary formats are generally more commonly used. Plain text data contains hexadecimal values, and one byte stores two hexadecimal characters. Raw binary data contains binary values.

- "🔄" (Auto Refresh): Used to refresh the memory contents at regular intervals. For RiscV_AE350_SOC with Auto Refresh debugging module, Auto Refresh can be enabled.

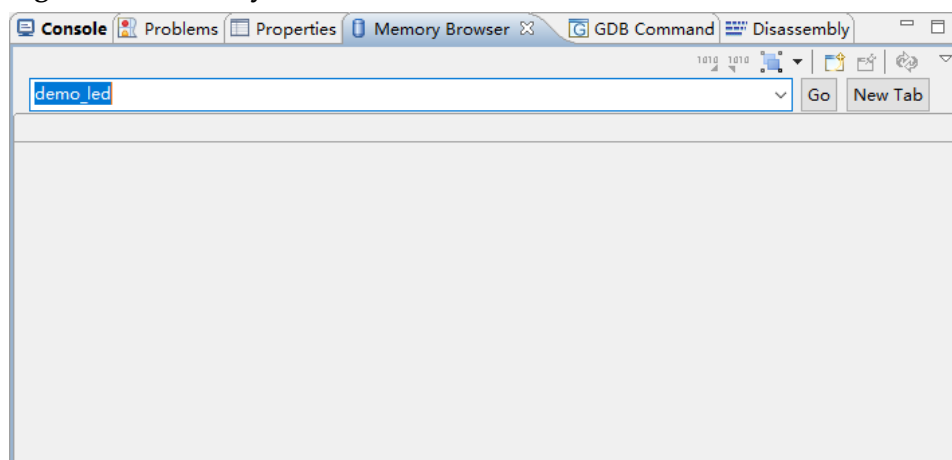
4.2.3 Memory Browser View

Memory Browser View is an alternative to Memory View for checking and modifying process memory.

Create a Memory Monitor

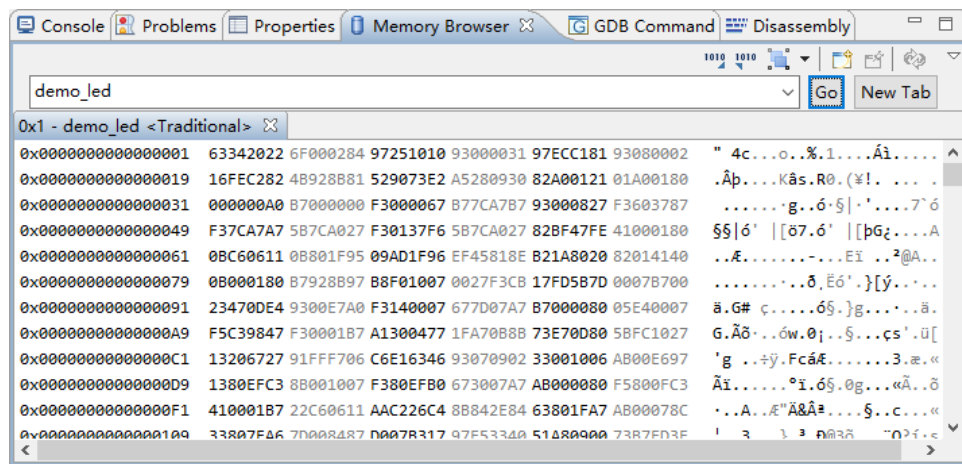
1. After the program is paused, enter the memory address or expression you want to view in the text box of the Memory Browser view and click "Go", as shown in Figure 4-19.

Figure 4-19 Memory Browser View



2. The memory browser view shows a traditional memory rendering by default, as shown in Figure 4-20.

Figure 4-20 Traditional Memory Rendering



Click the "▼" (View Menu) on the toolbar of Memory Browser View, and select "Default Rendering > Floating Point" from the drop-down list, and the click "New Tab", as shown in Figure 4-21 and Figure 4-22.

Figure 4-21 Floating Point Memory Rendering

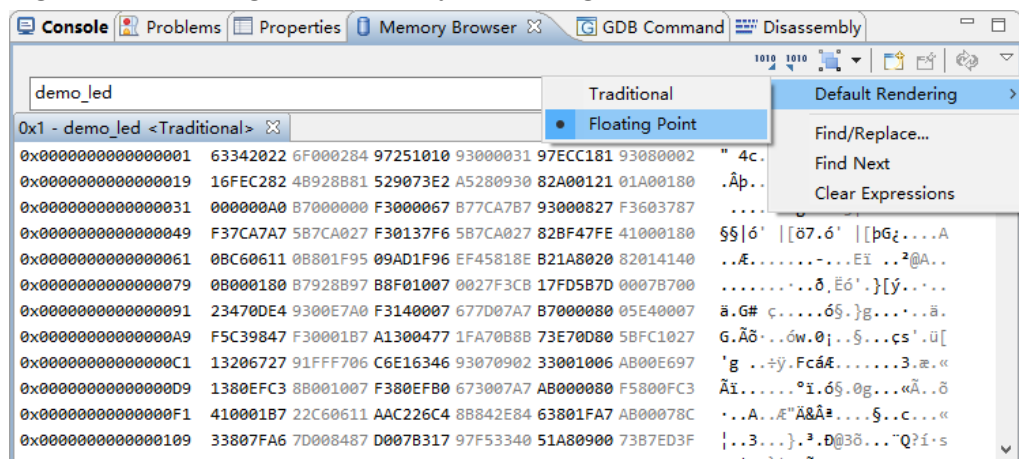
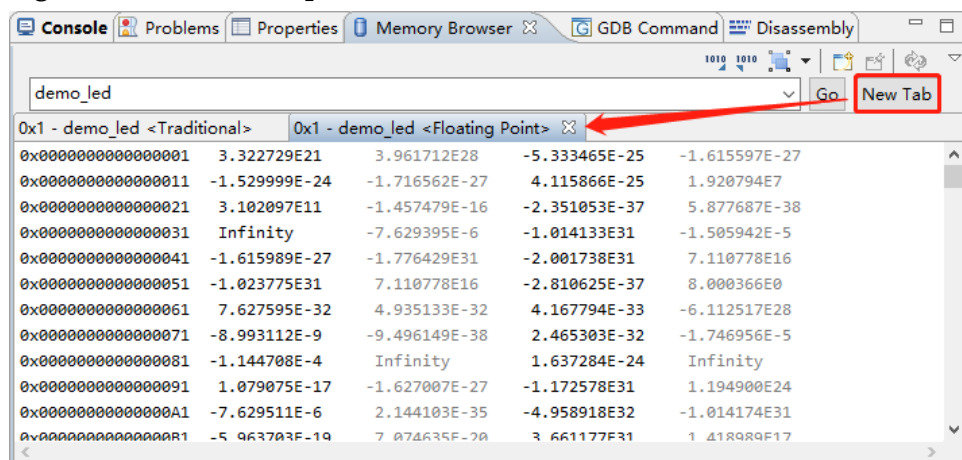


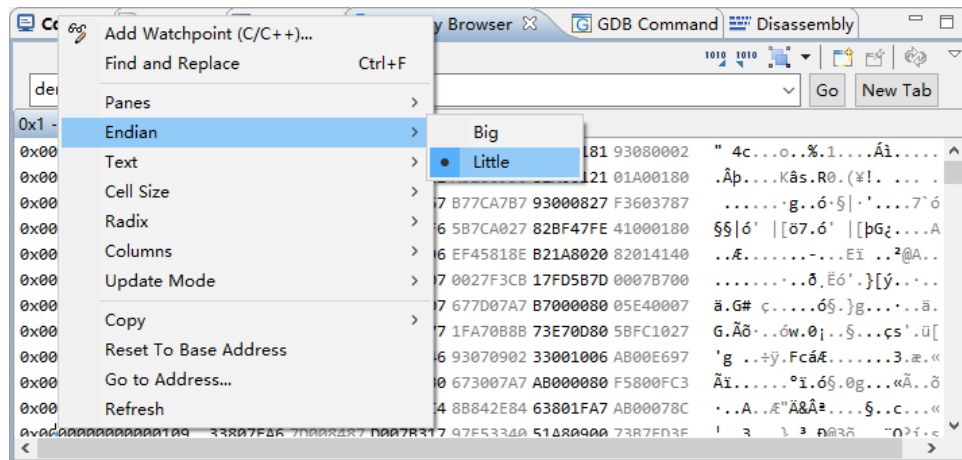
Figure 4-22 New Tab Option



3. Right-click on Memory Rendering and select "Endian" from the drop-

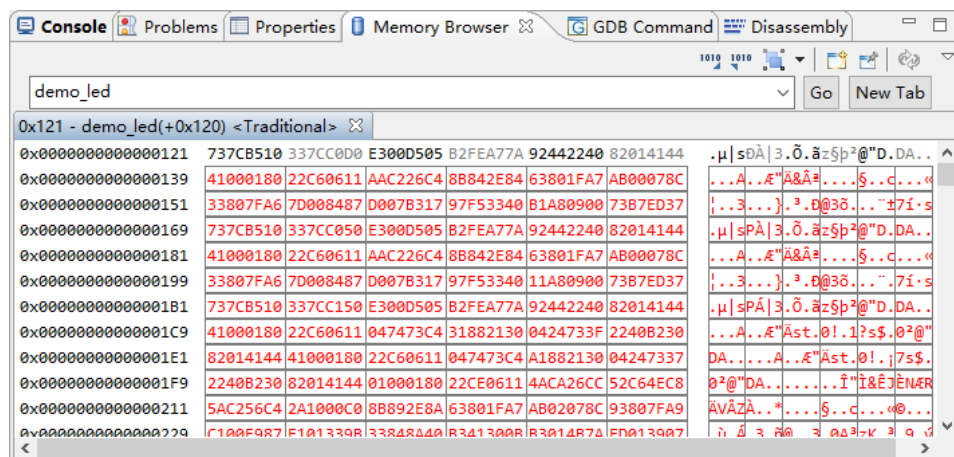
down list to change the byte order or addressable size of the data display, as shown in Figure 4-23.

Figure 4-23 Byte Order Setting



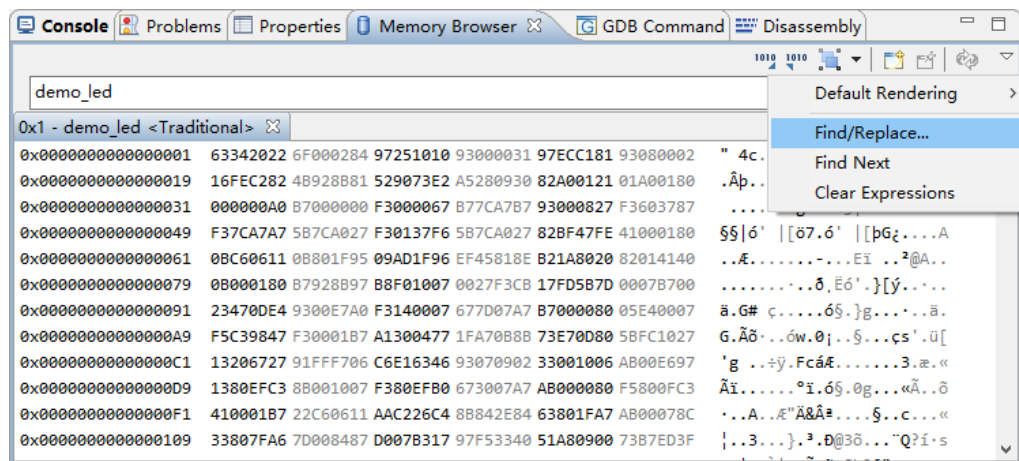
- Deviations from the contents of the memory are marked in red during program execution, as shown in Figure 4-24.

Figure 4-24 Contents of Memory



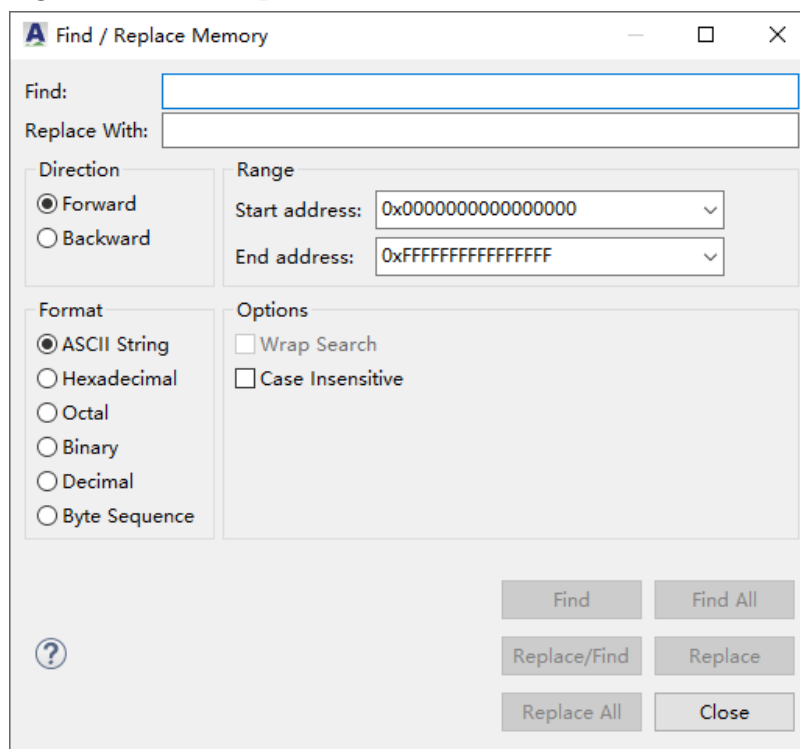
- Click the "▼" (View Menu) on the toolbar of Memory Browser View and select "Find/Replace..." from the drop-down list to find and replace the specified data for the selected memory monitor, as shown in Figure 4-25.

Figure 4-25 Select Find/Replace...



In the "Find / Replace Memory" dialog box, enter the data to be found and replaced, click "Find" or "Replace", as shown in Figure 4-26.

Figure 4-26 Find/Replace

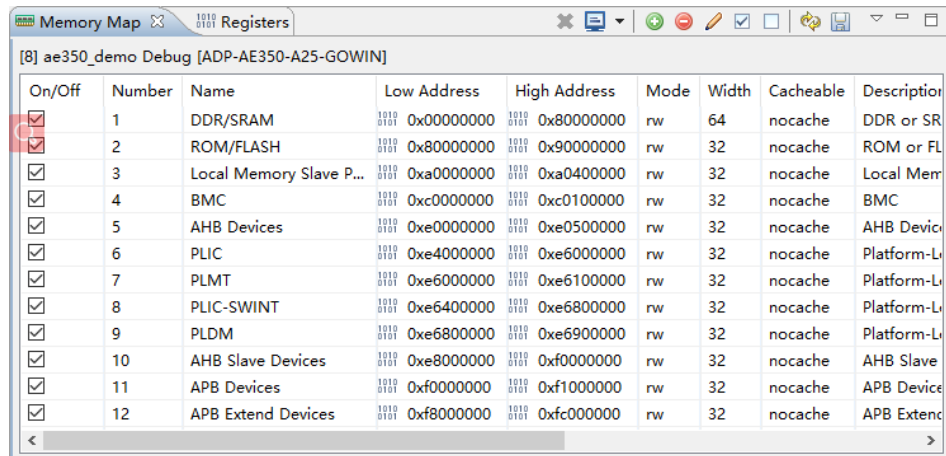


- For target systems with debug modules, the Auto Refresh is supported; click "🔄" (Auto Refresh) on the Memory Browser View toolbar to periodically refresh the memory contents.

4.2.4 Memory Map View

Memory Map View is an interface to check or add/modify/delete memory regions, memory attributes (read-only, write-only, read/write), and descriptions of memory mapped devices, as shown in Figure 4-27.

Figure 4-27 Memory Map View



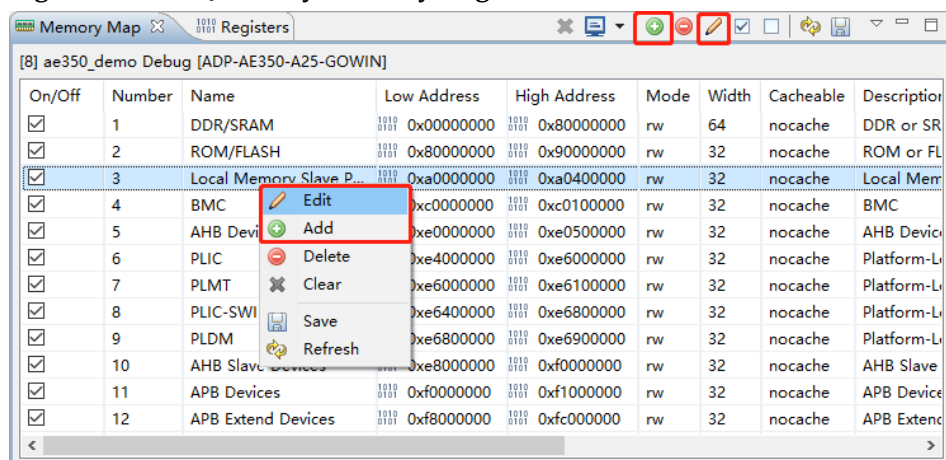
The screenshot shows the 'Memory Map' window with a toolbar at the top and a table of memory regions. The table has columns: On/Off, Number, Name, Low Address, High Address, Mode, Width, Cacheable, and Description. The 'On/Off' column has checkboxes, and the 'Number' column has a red box around the first two rows (1 and 2).

On/Off	Number	Name	Low Address	High Address	Mode	Width	Cacheable	Description
☒	1	DDR/SRAM	0x00000000	0x80000000	rw	64	nocache	DDR or SR
☒	2	ROM/FLASH	0x80000000	0x90000000	rw	32	nocache	ROM or FL
☒	3	Local Memory Slave P...	0xa0000000	0xa0400000	rw	32	nocache	Local Mem
☒	4	BMC	0xc0000000	0xc0100000	rw	32	nocache	BMC
☒	5	AHB Devices	0xe0000000	0xe0500000	rw	32	nocache	AHB Device
☒	6	PLIC	0xe4000000	0xe6000000	rw	32	nocache	Platform-L
☒	7	PLMT	0xe6000000	0xe6100000	rw	32	nocache	Platform-L
☒	8	PLIC-SWINT	0xe6400000	0xe6800000	rw	32	nocache	Platform-L
☒	9	PLDM	0xe6800000	0xe6900000	rw	32	nocache	Platform-L
☒	10	AHB Slave Devices	0xf0000000	0xf0000000	rw	32	nocache	AHB Slave
☒	11	APB Devices	0xf0000000	0xf1000000	rw	32	nocache	APB Device
☒	12	APB Extend Devices	0xf8000000	0xfc000000	rw	32	nocache	APB Extend

Refer to the following steps to add/modify/delete user-defined memory regions:

1. You can click "+" (Add Memory Regions) on the Memory Map View toolbar to add memory regions, or right-click anywhere in the Memory Map View and select "Add" from the drop-down list. You can select the row you want to modify in the Memory Map view, and click "Edit" (Modify Memory Regions) in the Memory Map view toolbar, or right-click anywhere in the Memory Map view, and select "Edit" in the drop-down menu to modify memory regions. Add/Modify Memory Regions as shown in Figure 4-28.

Figure 4-28 Add/Modify Memory Regions



The screenshot shows the 'Memory Map' window with a context menu open over row 3. The menu options are: Edit, Add, Delete, Clear, Save, and Refresh. The 'Add' option is highlighted with a red box.

On/Off	Number	Name	Low Address	High Address	Mode	Width	Cacheable	Description
☒	1	DDR/SRAM	0x00000000	0x80000000	rw	64	nocache	DDR or SR
☒	2	ROM/FLASH	0x80000000	0x90000000	rw	32	nocache	ROM or FL
☒	3	Local Memory Slave P...	0xa0000000	0xa0400000	rw	32	nocache	Local Mem
☒	4	BMC	0xc0000000	0xc0100000	rw	32	nocache	BMC
☒	5	AHB Devi	0xe0000000	0xe0500000	rw	32	nocache	AHB Device
☒	6	PLIC	0xe4000000	0xe6000000	rw	32	nocache	Platform-L
☒	7	PLMT	0xe6000000	0xe6100000	rw	32	nocache	Platform-L
☒	8	PLIC-SWI	0xe6400000	0xe6800000	rw	32	nocache	Platform-L
☒	9	PLDM	0xe6800000	0xe6900000	rw	32	nocache	Platform-L
☒	10	AHB Slave Devices	0xf0000000	0xf0000000	rw	32	nocache	AHB Slave
☒	11	APB Devices	0xf0000000	0xf1000000	rw	32	nocache	APB Device
☒	12	APB Extend Devices	0xf8000000	0xfc000000	rw	32	nocache	APB Extend

2. Double-click a memory mapped region, and in the "Memory Map

Region Wizard" dialog box, you can configure the relevant options, click "Add" or "Change", as shown in Figure 4-29.

Figure 4-29 Memory Map Region Wizard Dialog Box

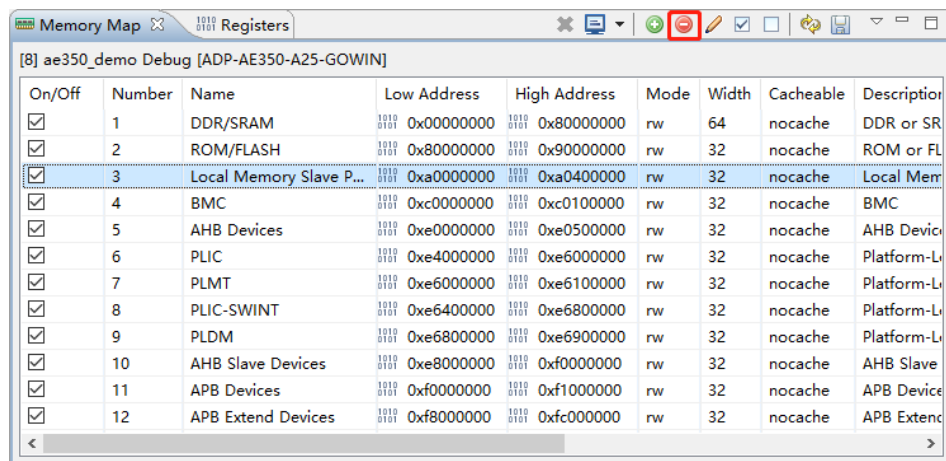
Options for the memory map region are as shown in Table 4-1.

Table 4-1 Memory Mapping Region Options

Option	Description
Name	Name of the memory region
Low Address	Specify the starting address of the memory region
High Address	Specify the end address of the memory region
Description	Descriptions of memory region
Mode	Specify the mode in which GDB accesses memory regions, including: ro: Read-only wo: Write-only rw: Read/Write (default mode)
Cacheable	Specify whether GDB caches target memory

3. The created/modified memory regions are immediately displayed in the Memory Map View and can be turned on/off by checking/unchecking the checkboxes in the On/Off column.
4. You can select a row of memory regions and click "⊖" (Delete Memory Regions) on the Memory Map View toolbar to delete a memory region, as shown in Figure 4-30.

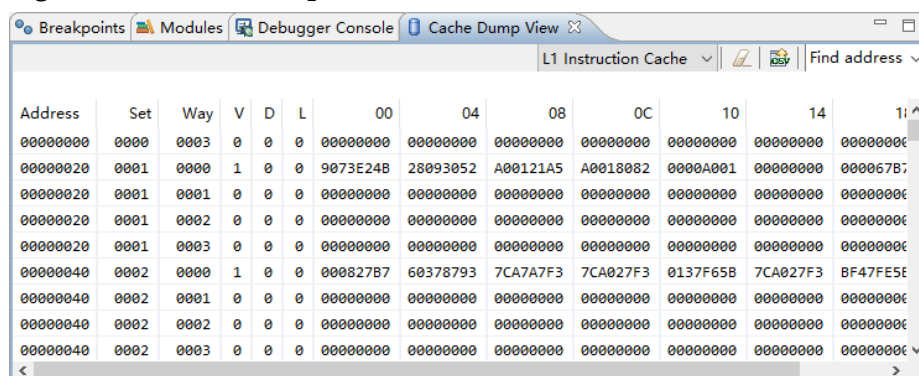
Figure 4-30 Delete Memory Regions



4.2.5 Cache Dump View

The Cache Dump View is used to monitor the target system's CPU cache after a pause in program execution. For each cache line, this view displays its current information, including "Address", "Set", "Way", "V" (Valid state), "D" (Dirty state), "L" (Lock state), and data, as shown in Figure 4-31.

Figure 4-31 Cache Dump View

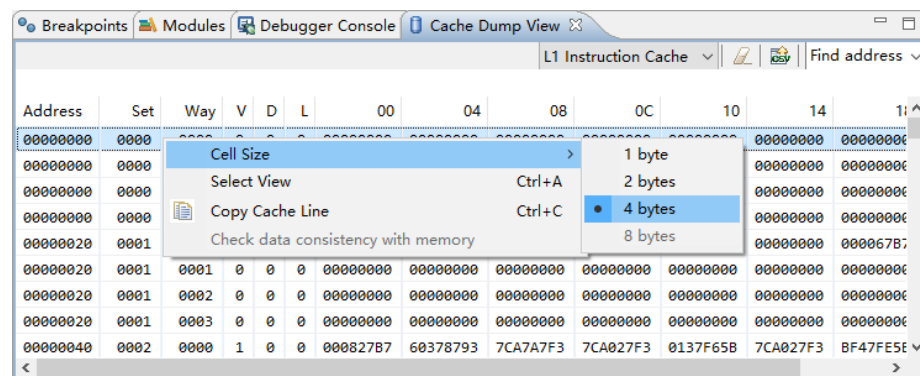


When the program executes, or after a cache operation, if there is a difference from the previous cache, the cell change is marked with a different color.

Cell Size Setting

By changing the cell size (1/2/4/8 bytes), you can adjust how the cache is displayed. Right-click on a line of cache, and select "Cell Size" from the drop-down list, as shown in Figure 4-32.

Figure 4-32 Cell Size Setting



Check Data Consistency with Memory

The Cache Dump view allows you to check data consistency with memory by right-clicking on a valid cache line and selecting "Check data consistency with memory" from the drop-down list, as shown in Figure 4-33. The top of this view immediately shows whether the data in the cache line is consistent with the data in memory, as shown in Figure 4-34.

Figure 4-33 Check Data Consistency with Memory

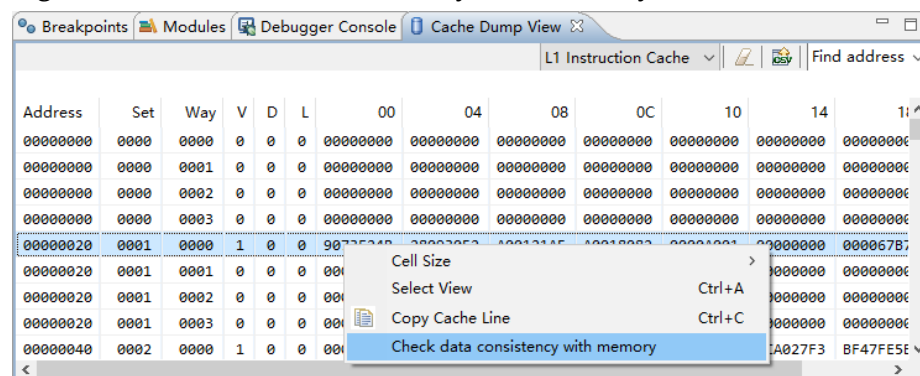
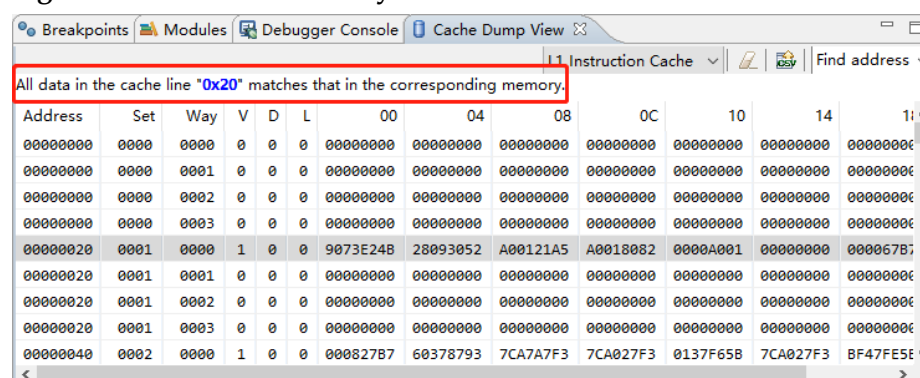


Figure 4-34 Data and Memory Matches

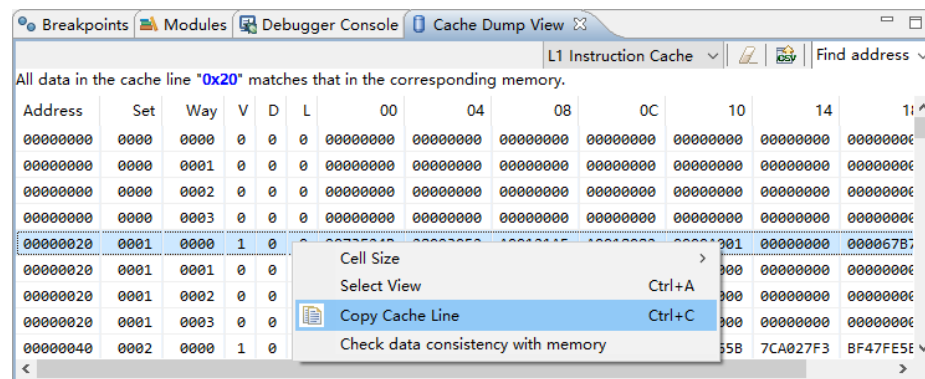


Copy Cache Line

If you need to save the information displayed in this view, right-click on a cache line and select "Copy Cache Line" from the drop-down list to

paste it into a file, as shown in Figure 4-35.

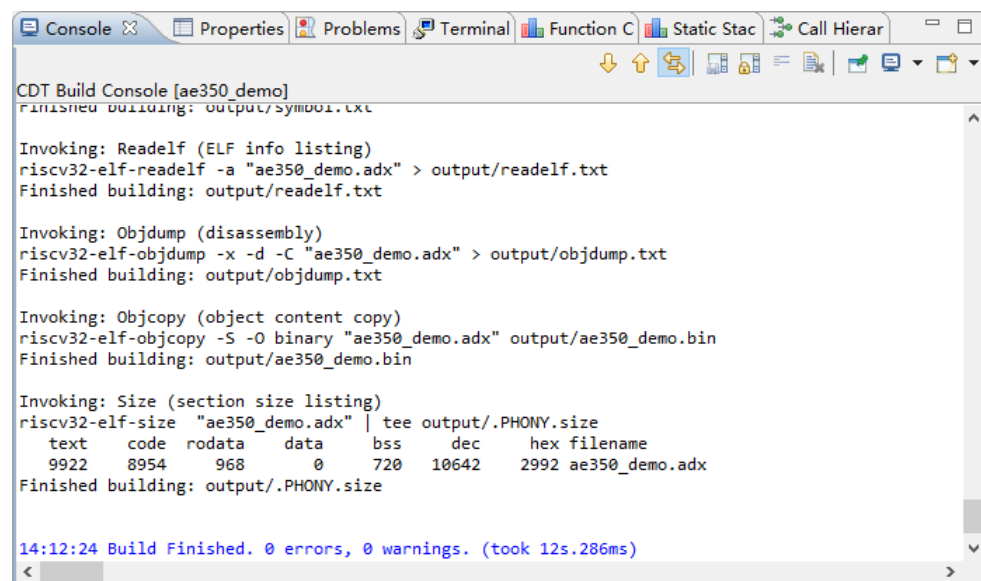
Figure 4-35 Copy Cache Line



4.2.6 Console View

The console view displays process output as well as providing a command line interface for the command input. For example, a CDT build console that displays the build results of a software project is shown in Figure 4-36.

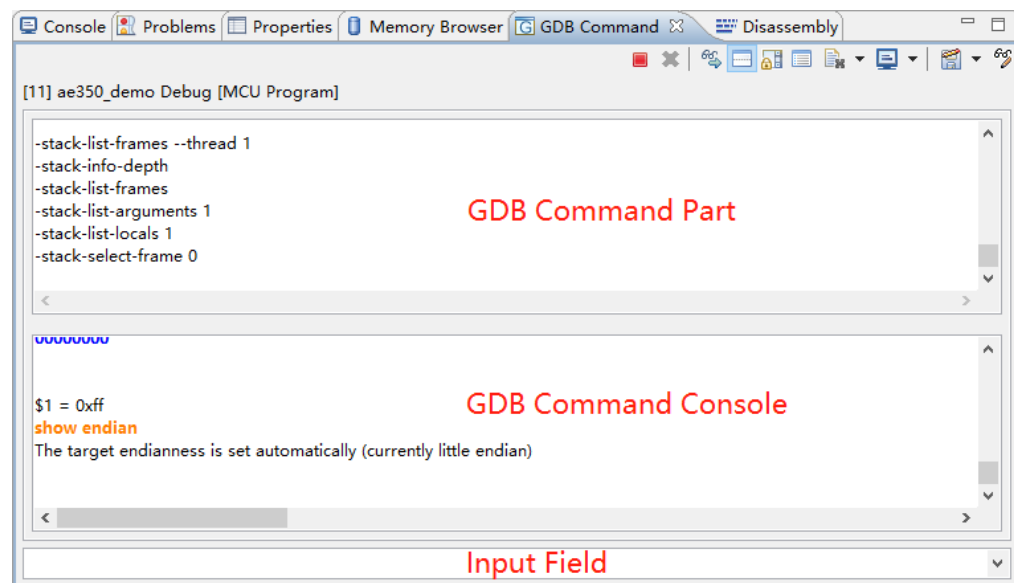
Figure 4-36 Console View



4.2.7 GDB Command View

GDB Command View serves as an interface for GDB commands, as shown in Figure 4-37.

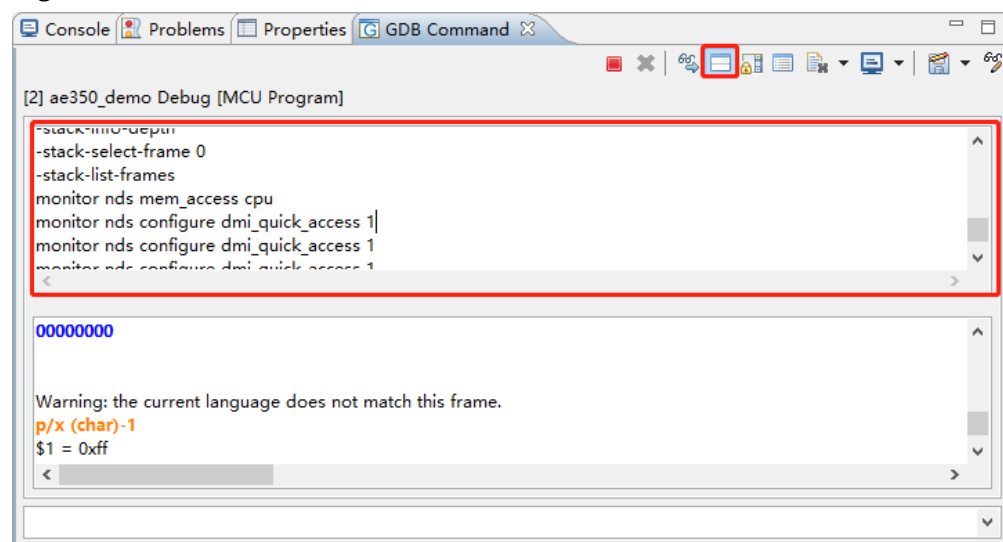
Figure 4-37 GDB Command View



The GDB Command View consists of three sections: the GDB Command Part, the GDB Command Console, and the GDB Command Input.

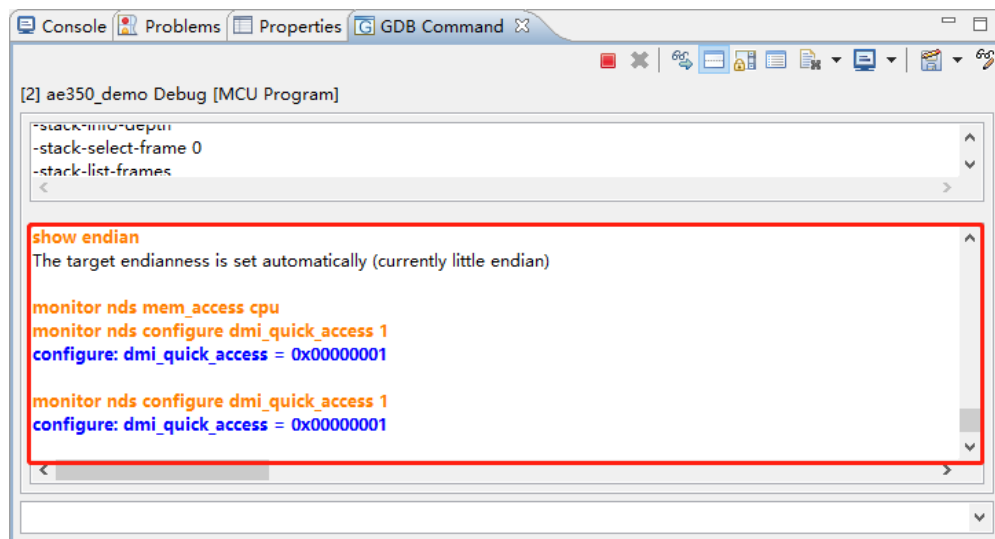
The top of this view is a read-only log view showing all the GDB commands that can be sent in a debugging session, and this is hidden by default and can be invoked by clicking "Show Command Part" on the GDB Command View toolbar, as shown in Figure 4-38.

Figure 4-38 GDB Command Part



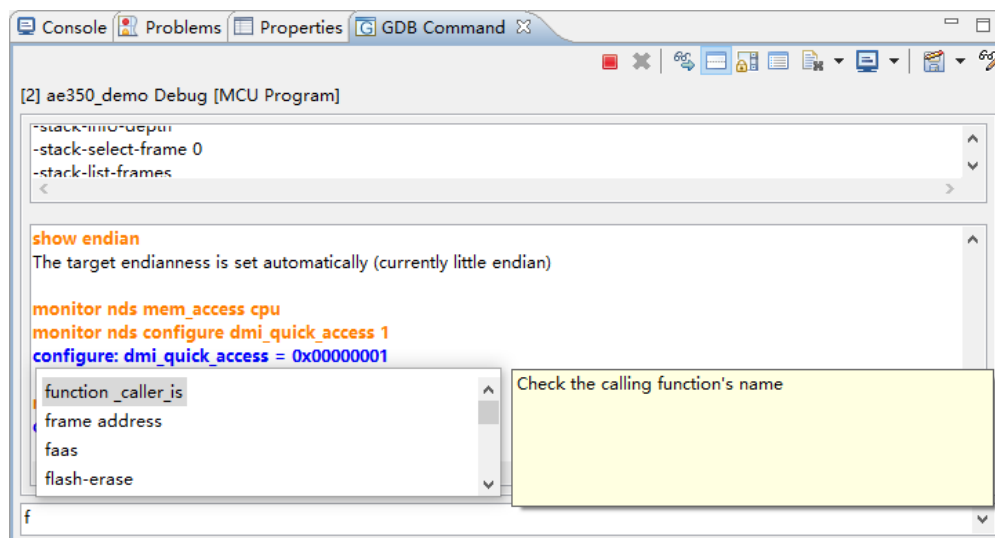
The middle part of this view is the GDB Command Console, which displays GDB commands and results. The console displays different types of information in different colors: orange for GDB commands, blue for program output, and black for GDB command results, as shown in Figure 4-39.

Figure 4-39 GDB Command Console



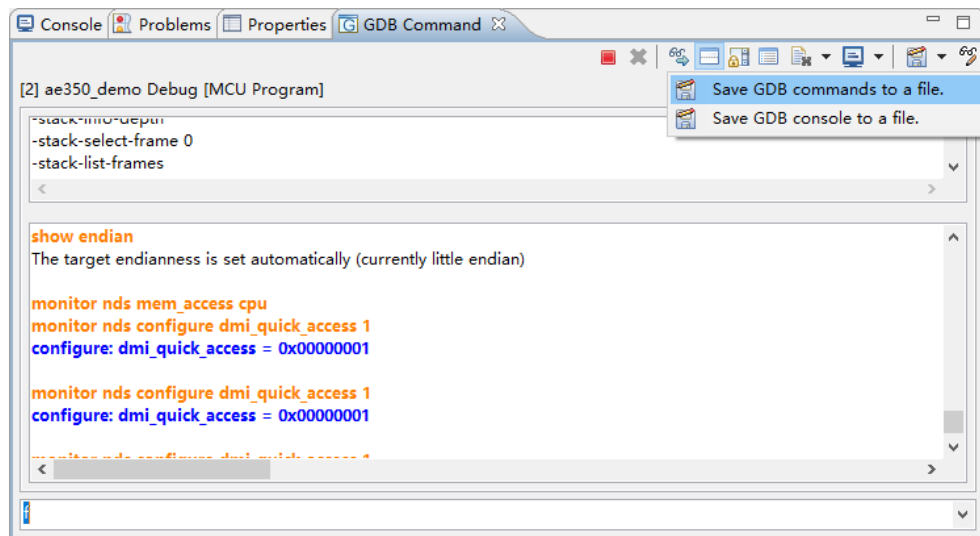
The bottom portion of this view is the Command Input Interface, which is used to send GDB commands. This section provides suggestions and descriptions of the input commands and can store up to 20 GDB command records, as shown in Figure 4-40.

Figure 4-40 GDB Command Input



The GDB Command Part or GDB Command Console information can be exported to a file by clicking " " drop-down arrow (Save GDB Context) on the GDB Command View toolbar and selecting "Save GDB commands to a file." or "Save GDB console to a file.", as shown in Figure 4-41.

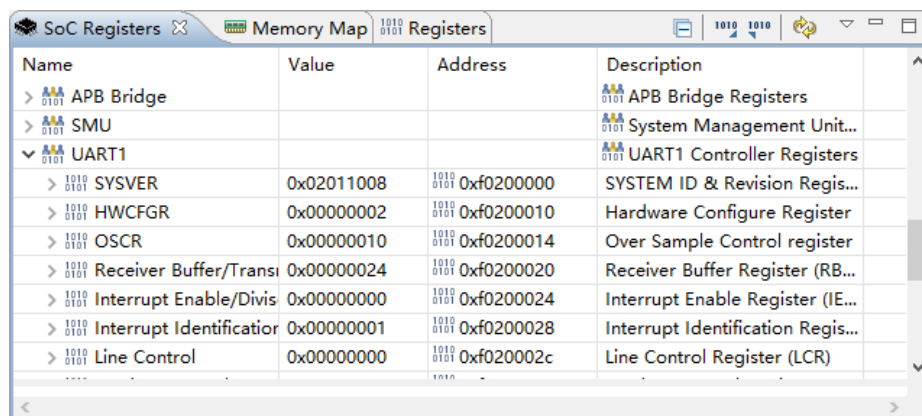
Figure 4-41 Export GDB Command



4.2.8 SoC Registers View

The SoC Registers View is used to read or write SoC register values at the bit-field level. This view is automatically triggered during a debug session as part of the debug perspective, as shown in Figure 4-42.

Figure 4-42 SoC Registers View



Import/Export Register Values

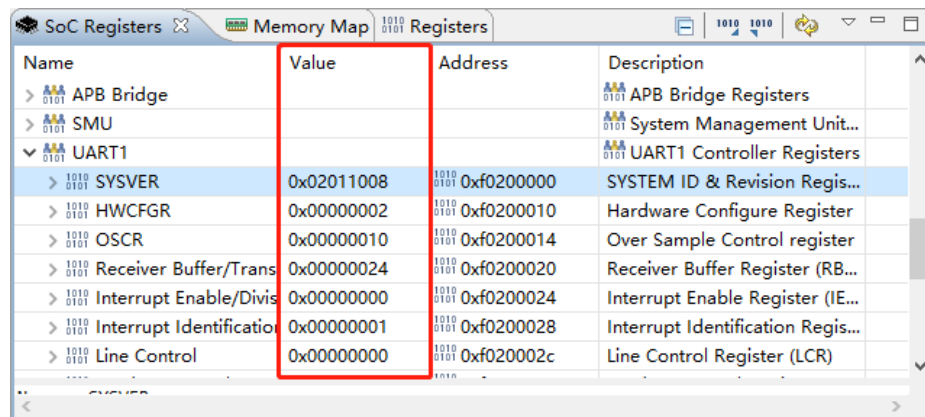
The value of a specified SoC register can be exported at program runtime by selecting the SoC component or register that you want to export (you can select multiple times by holding down "Ctrl"), and then clicking "1010" (Export) on the SoC Registers View toolbar.

SoC register data can be imported by clicking "1010" (Import) on the SoC register view toolbar.

Modify Register Values

You can directly modify the value of a register or bit field in a cell, as shown in Figure 4-43.

Figure 4-43 Modify Register Values

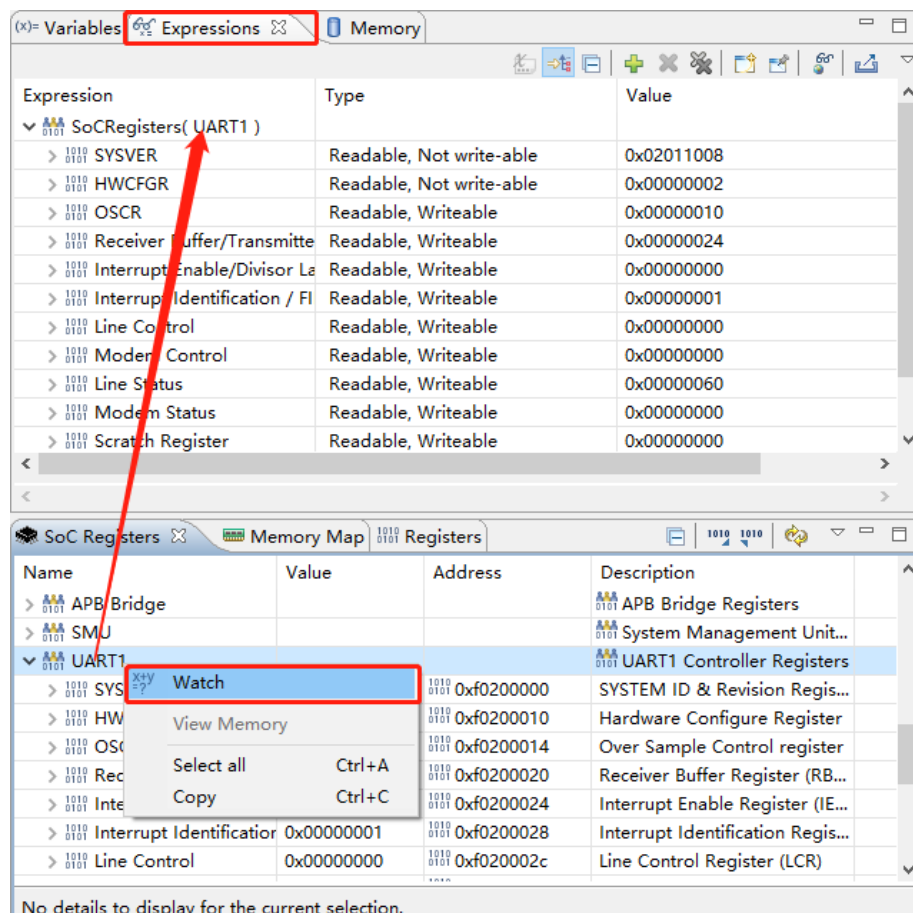


Name	Value	Address	Description
> APB Bridge			APB Bridge Registers
> SMU			System Management Unit...
> UART1			UART1 Controller Registers
> SYSVER	0x02011008	0xf0200000	SYSTEM ID & Revision Regis...
> HWCFGR	0x00000002	0xf0200010	Hardware Configure Register
> OSCR	0x00000010	0xf0200014	Over Sample Control register
> Receiver Buffer/Trans	0x00000024	0xf0200020	Receiver Buffer Register (RB...
> Interrupt Enable/Divis	0x00000000	0xf0200024	Interrupt Enable Register (IE...
> Interrupt Identification	0x00000001	0xf0200028	Interrupt Identification Regis...
> Line Control	0x00000000	0xf020002c	Line Control Register (LCR)

Watch SoC Registers

You can watch an individual SoC registers or groups of SoC registers. In the SoC Registers view, right-click on the register group or individual register you want to view, and select "Watch" from the drop-down list. The Expressions View displays information about the selected SoC registers, as shown in Figure 4-44.

Figure 4-44 Watch SoC Registers



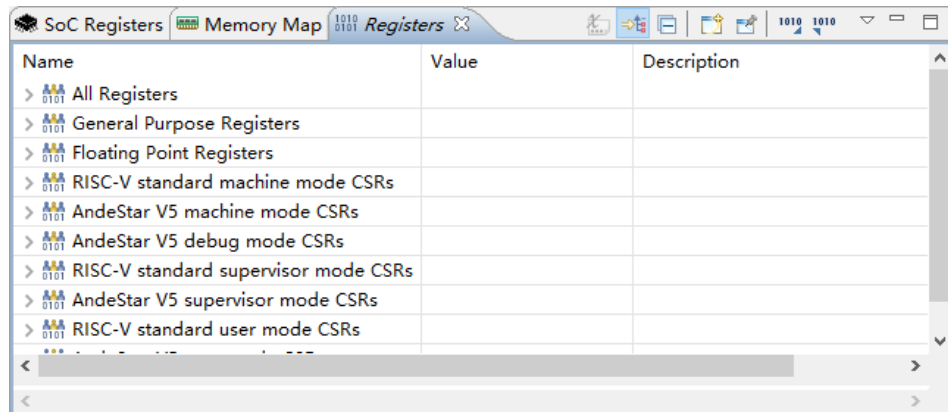
Expression	Type	Value
> SoCRegisters(UART1)		
> SYSVER	Readable, Not write-able	0x02011008
> HWCFGR	Readable, Not write-able	0x00000002
> OSCR	Readable, Writeable	0x00000010
> Receiver Buffer/Transmitte	Readable, Writeable	0x00000024
> Interrupt Enable/Divisor La	Readable, Writeable	0x00000000
> Interrupt Identification / FI	Readable, Writeable	0x00000001
> Line Control	Readable, Writeable	0x00000000
> Modem Control	Readable, Writeable	0x00000000
> Line Status	Readable, Writeable	0x00000060
> Modem Status	Readable, Writeable	0x00000000
> Scratch Register	Readable, Writeable	0x00000000

Name	Value	Address	Description
> APB Bridge			APB Bridge Registers
> SMU			System Management Unit...
> UART1			UART1 Controller Registers
> SYSVER		0xf0200000	SYSTEM ID & Revision Regis...
> HWCFGR		0xf0200010	Hardware Configure Register
> OSCR		0xf0200014	Over Sample Control register
> Receiver Buffer/Trans		0xf0200020	Receiver Buffer Register (RB...
> Interrupt Enable/Divis		0xf0200024	Interrupt Enable Register (IE...
> Interrupt Identification	0x00000001	0xf0200028	Interrupt Identification Regis...
> Line Control	0x00000000	0xf020002c	Line Control Register (LCR)

4.2.9 Registers View

The Registers View displays CPU information at the level of the register bit field, and you can modify the value of the register or bit field directly in the cell, as shown in Figure 4-45. You can click "1010" (Export) or "1010" (Import) on Registers View toolbar to export or import the value of the register.

Figure 4-45 Registers View



4.2.10 Variables View

The Variables View lists information about the variables associated with the stack structure selected in the debug view. When the program execution is paused, the changes in the values of the variables are highlighted, as shown in Figure 4-46.

Figure 4-46 Variables View

Name	Type	Value
(x)= num	uint8_t	0x2
(x)= led_pin	uint32_t	0x2
(x)= begin_time	long	0x60816
(x)= end_time	long	0xbd961
(x)= use_time	long	0x5d14b

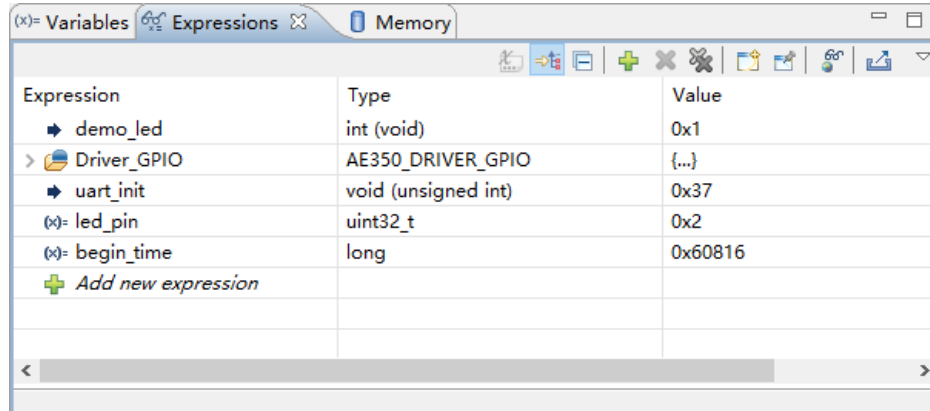
4.2.11 Expressions View

The Variables View does not display global variables, but which are instead displayed by the Expression View or the Global Variables Live View; for the details, you can see [4.2.12 Global Variables Live View](#). If the value of a monitored global variable is updated, it is highlighted in yellow in

this view. For the Expression View, the updated value of the global variable is displayed only after program execution is paused. For the Global Variables Live View, the values of global variables are displayed in real time as long as the program is running on a target system with a RISC-V core.

For example, an expression view of an application in a debugging session is shown in Figure 4-47.

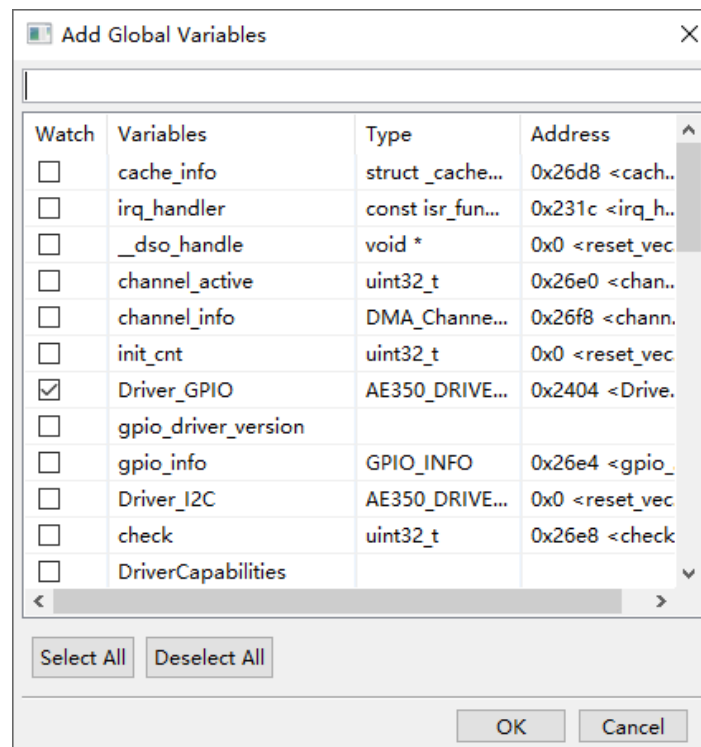
Figure 4-47 Expressions View



The toolbar of the Expression view includes:

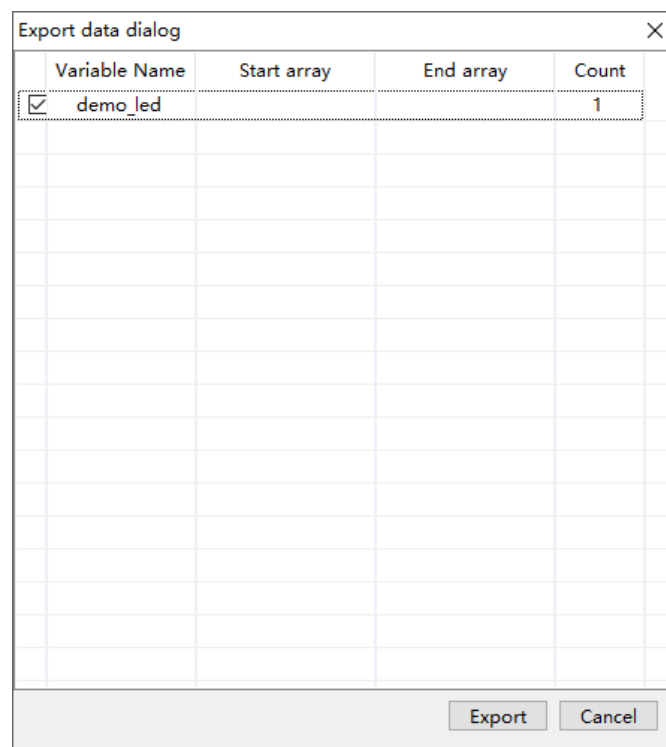
- "  " (add global variables): After program execution is paused, click "  ", and in "Add Global Variables" dialog box, select the global variables you want to monitor, then the Expression View displays the selected global variables and their values, as shown in Figure 4-48.

Figure 4-48 Add Global Variables



-  (Export): You can export the desired global variable; select a global variable and click "", as shown in Figure 4-49.

Figure 4-49 Export a Global Variable



In the "Export data dialog", specify the start and end arrays for the

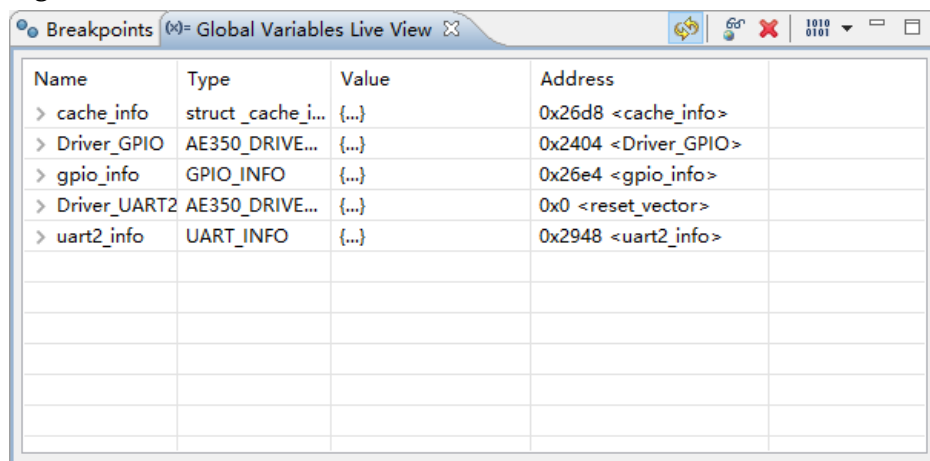
variables to be exported, and then click "Export".

4.2.12 Global Variables Live View

Global Variables Live View can monitor global variables and check the values of global variables in real time as long as the program is running on a target system with a RISC-V core, as shown in Figure 4-50. After a pause in program execution, all variable members can be checked in the expanded list. The values of global variables or structure members that are updated during program execution are highlighted in yellow in this view.

If you are using an simulator target system, this view shows the updated values of global variables only after the program is paused.

Figure 4-50 Global Variables Live View

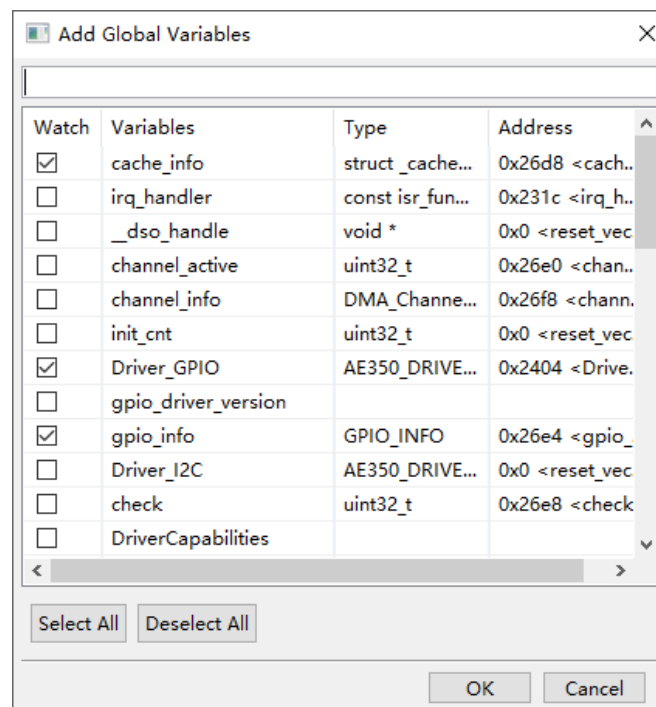


Name	Type	Value	Address
> cache_info	struct_cache_i...	{...}	0x26d8 <cache_info>
> Driver_GPIO	AE350_DRIVE...	{...}	0x2404 <Driver_GPIO>
> gpio_info	GPIO_INFO	{...}	0x26e4 <gpio_info>
> Driver_UART2	AE350_DRIVE...	{...}	0x0 <reset_vector>
> uart2_info	UART_INFO	{...}	0x2948 <uart2_info>

The toolbar of the global variables live view includes:

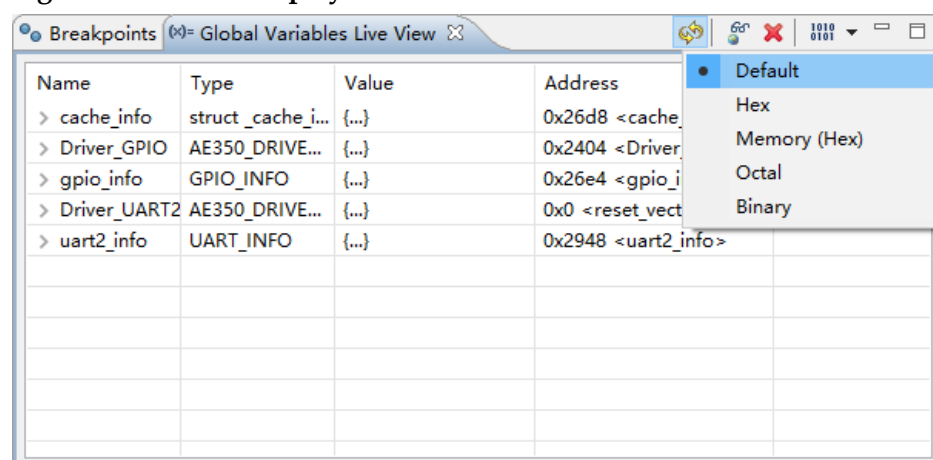
- "🔄" (Auto Refresh): Refresh the global variables in real time.
- "🔍" (add global variables): After program execution is paused, click "🔍", and in "Add Global Variables" dialog box, select the global variables you want to monitor, then the Global Variables Live View displays the selected global variables and their values, as shown in Figure 4-51.

Figure 4-51 Add Global Variables



- "✖" (Delete the selected variables): Deletes the monitored global variables.
- "1010 0101" (Select the format for values): Click the drop-down arrow and select a display format for the monitored global variables from the drop-down list, including "Default", "Hex", "Memory (Hex)", "Octal", and "Binary", as shown in Figure 4-52. Note that "Memory (hex)" refers to the hexadecimal value stored in memory, and the byte order is consistent with the byte order of the target system.

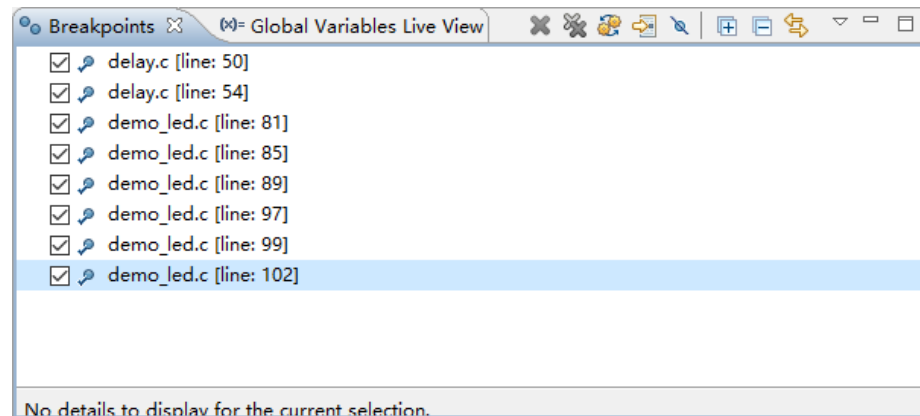
Figure 4-52 Select Display Format



4.2.13 Breakpoints View

The Breakpoints View lists all the breakpoints set in the software project, allowing you to add, cancel, turn on/off breakpoints, as shown in Figure 4-53.

Figure 4-53 Breakpoints View



4.2.14 Disassembly View

The disassembly view shows the high-level language and its related assembly code, which is used to find instruction problems during debugging, as shown in Figure 4-54.

Figure 4-54 Disassembly View



4.3 Debugging Session

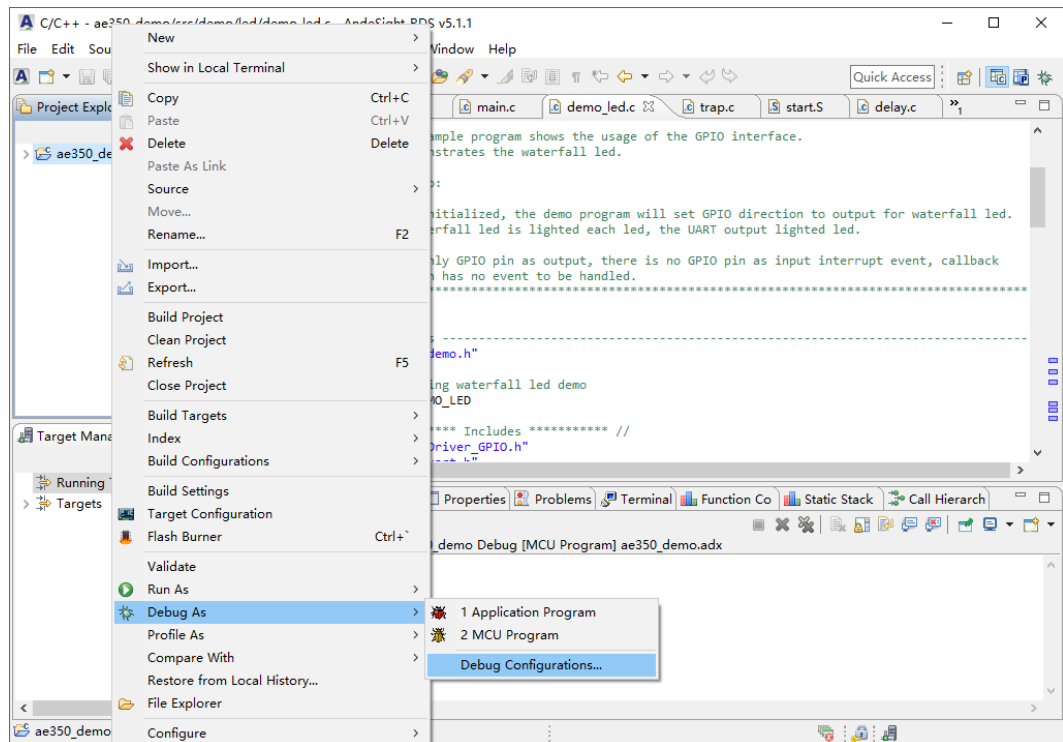
A debugging session is a debugging process that uses a specified set of program executable files, debugger behavior, related debugging tools, and other configuration settings. Once the software project has created the debug session configuration, click "  " (Debug) on the RDS software toolbar to start the debug session.

4.3.1 Create Debug Configurations

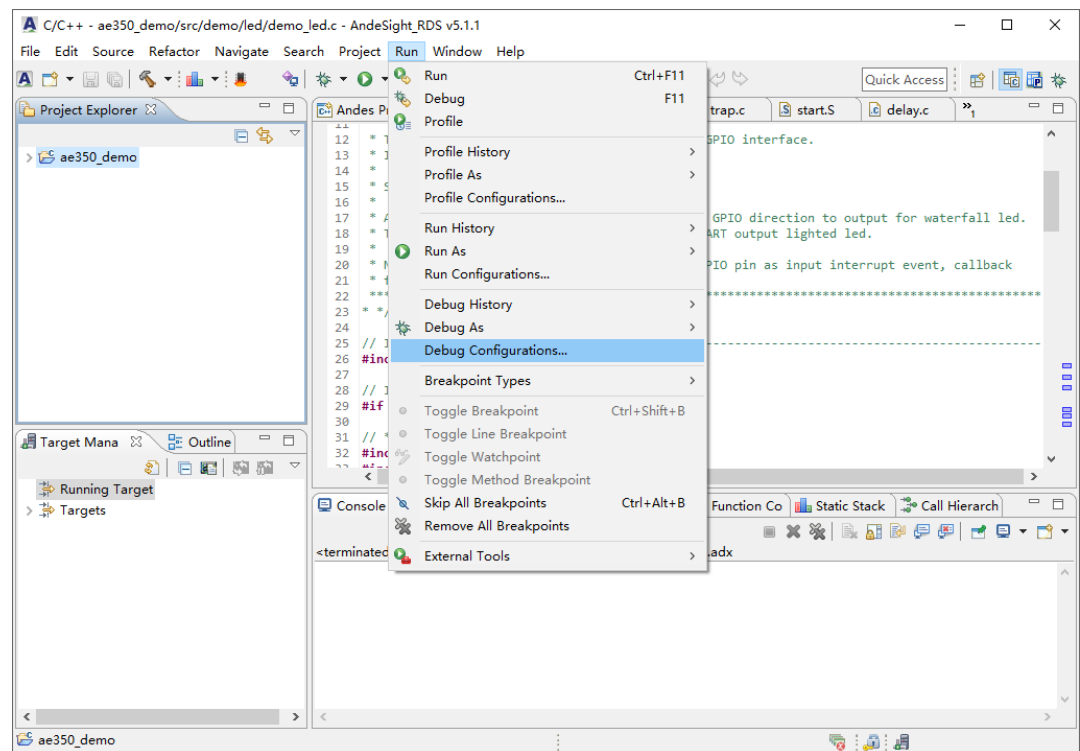
You can refer to the following steps to create debug configurations:

1. In the Project Explorer view, right-click to select a software project, and select "Debug As > Debug Configurations..." from the drop-down list, as shown in the Figure 4-55.

Figure 4-55 Select Debug As > Debug



Or select "Run > Debug Configurations..." on the main menu of RDS software, as shown in Figure 4-56.

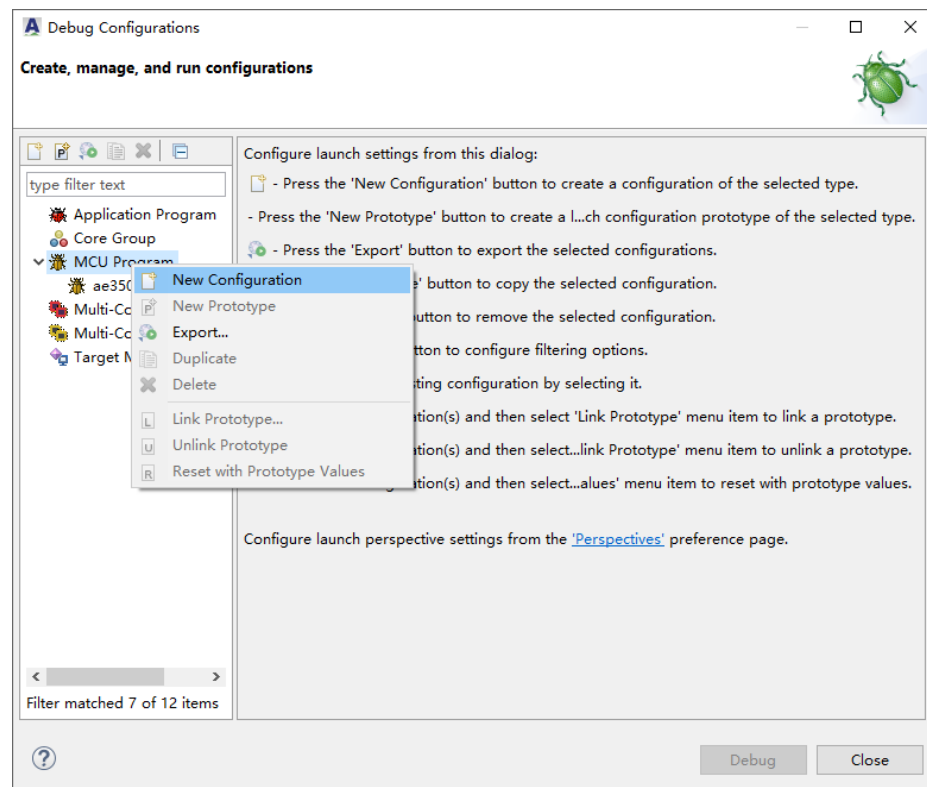
Figure 4-56 Select Run > Debug Configurations...

2. In the "Debug Configurations" dialog box, you can refer to the debug configuration type, as shown in Table 4-2; right-click on the selected configuration type, such as "MCU Program", and select "New Configuration" in the drop-down list to create debug configurations, as shown in Figure 4-57.

Table 4-2 Debugging Configuration Type

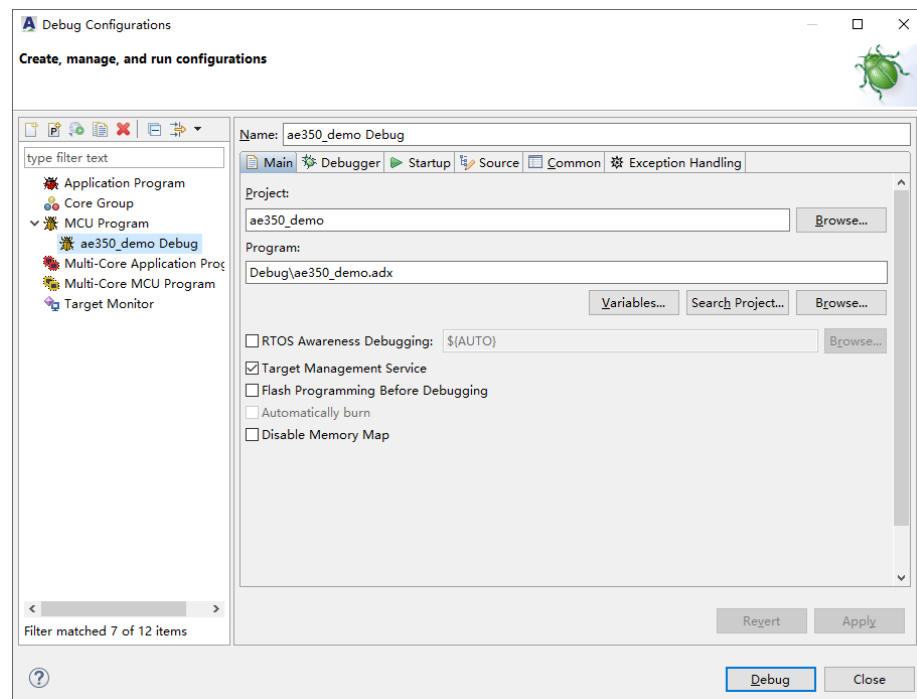
Type	Description
Application Program	Used for the target system with system service
MCU Program	Used for the target system without system service
Core Group	Used to initiate multiple debugging configurations, and send debugging commands to the core group
Multi-Core Application Program	Used for the multi-core target system with system service
Multi-Core MCU Program	Used for the multi-core target system without system service
Target Monitor	Used to check the default status of the target

Figure 4-57 Create New Debugging Configuration



3. After completing the configuration settings in each tab, click "Apply" and "Debug" to start the debugging session, as shown in Figure 4-58.

Figure 4-58 Debugging Configurations



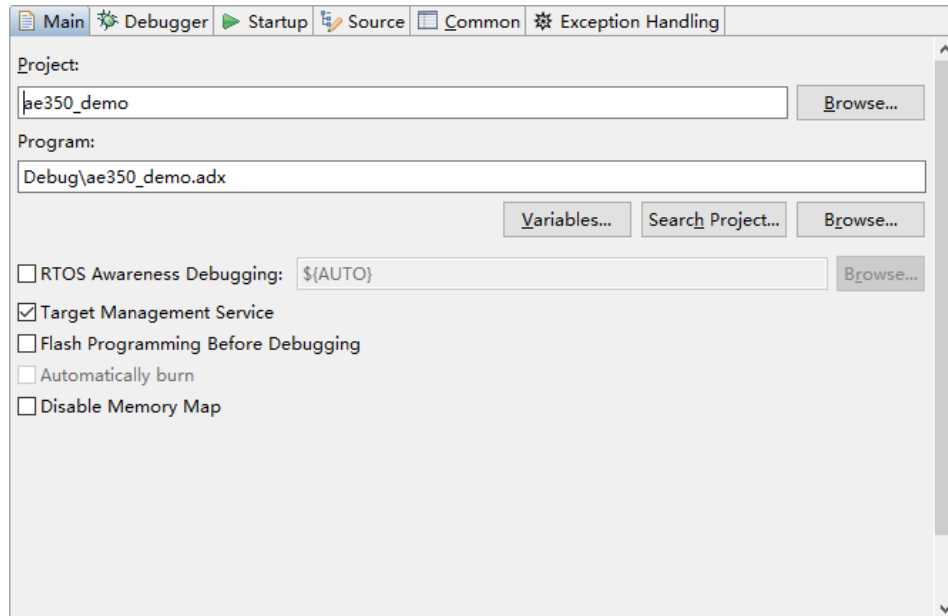
4.3.2 Debugging Session Settings

The following sections describe how to configure the "MCU Program" debug in details.

Main Tab

The "Main" tab is as shown in Figure 4-59.

Figure 4-59 Main Tab



The descriptions of Main options are as shown in Table 4-3.

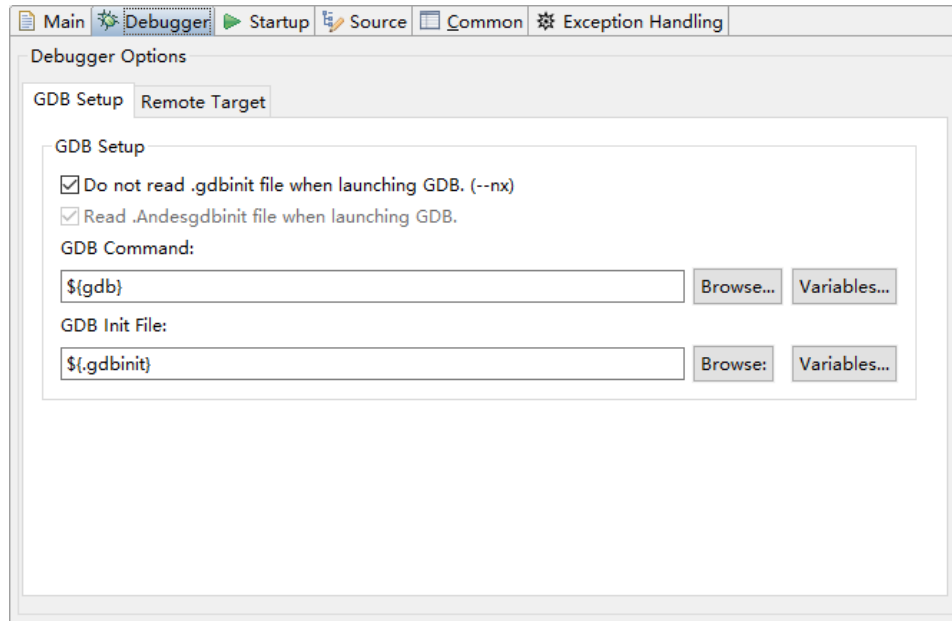
Table 4-3 Main Options

Option	Description
Project	The software project to be debugged
Program	The software project executables
RTOS Awareness Debugging	Create a table for displaying RTOS information. This function can automatically detect the RTOS core version based on symbols defined in the source code, or you can manually specify a Python script with GDB commands by clicking "Browse..."
Target Management Service	Target management service
Disable Memory Map	Load program executables without mapping to predefined memory areas

Debugger Tab

The "Debugger" tab is as shown in Figure 4-60.

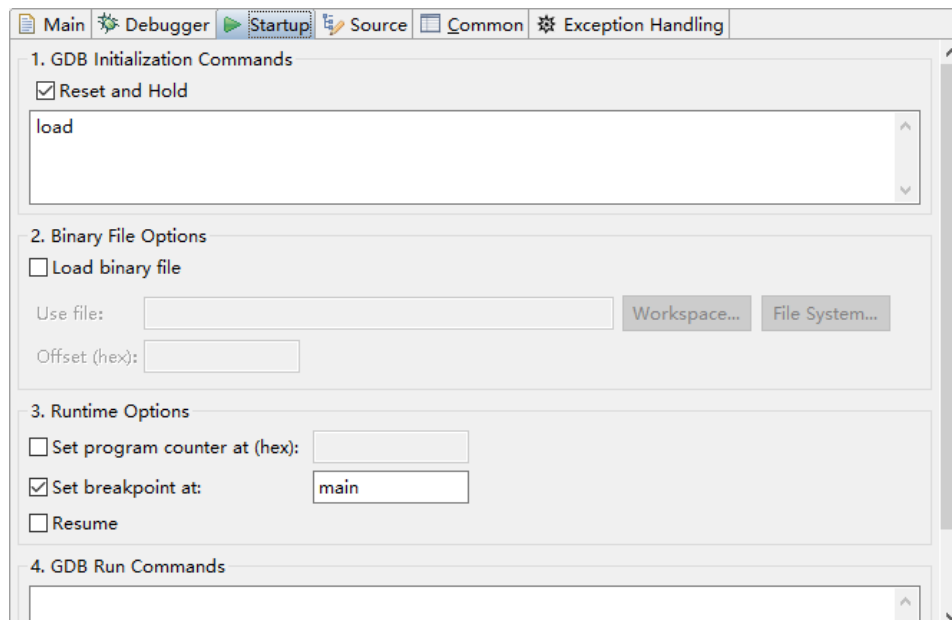
Figure 4-60 Debugger Tab



Startup Tab

The "Startup" tab is as shown in Figure 4-61.

Figure 4-61 Startup Tab



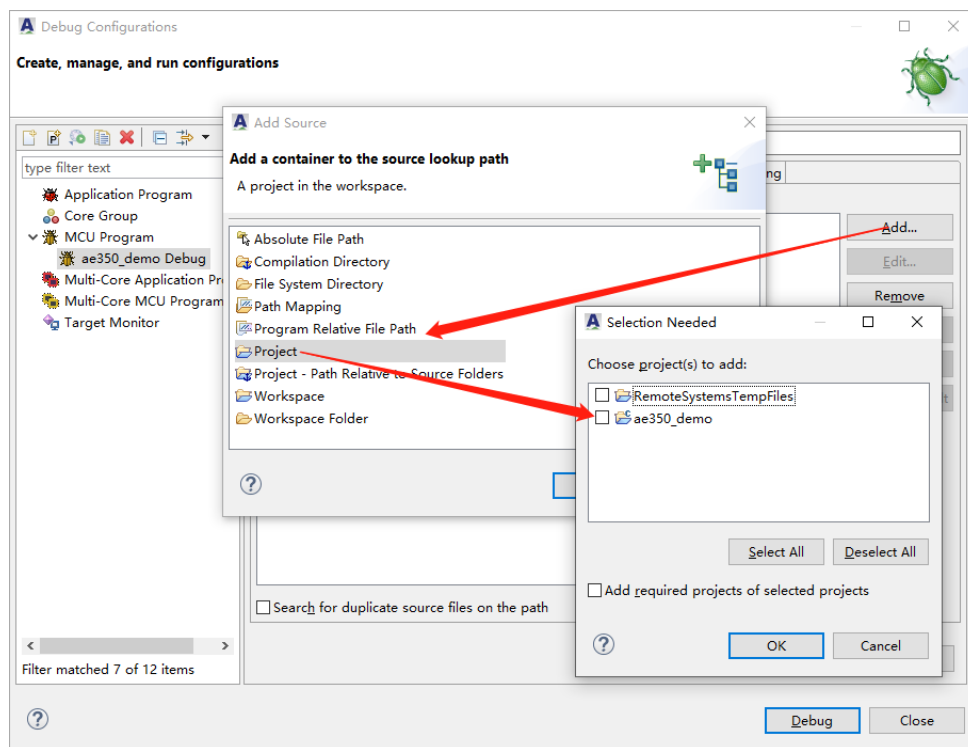
The descriptions of Startup options are as shown in Table 4-4.

Table 4-4 Startup Options

Option	Description
Reset and Hold	Immediately hold at "_start" after a reset load: GDB loading program
Set breakpoint at	Set the breakpoint position, e.g. "main"
Resume	Run to breakpoint position after reset
GDB Run Commands	GDB run commands

Source Tab

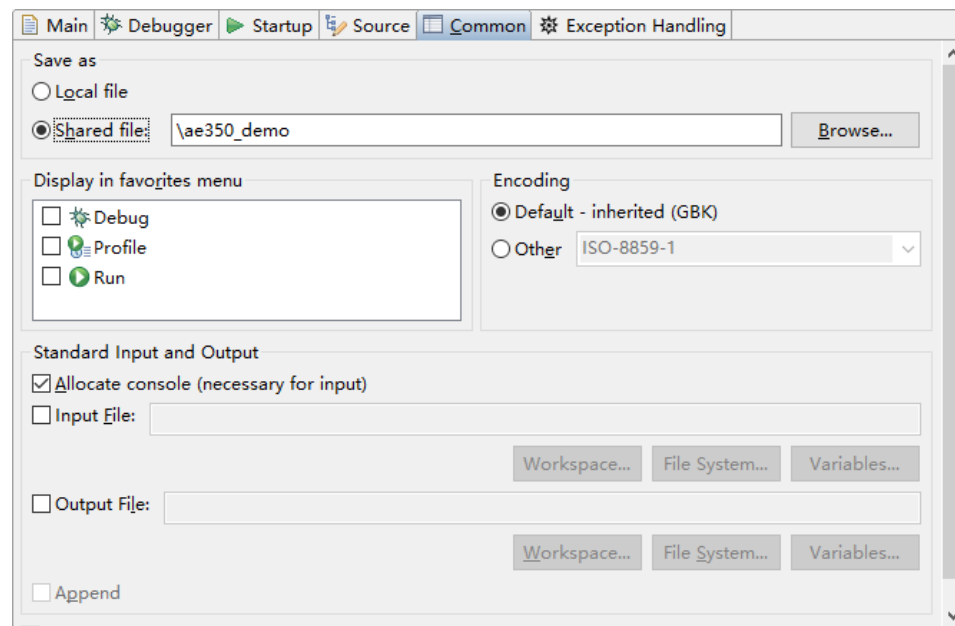
The "Source" tab is as shown in Figure 4-62.

Figure 4-62 Source Tab

This tab allows you to specify the source files to be used for debugging a C or C++ application, e.g., you can specify a software project that contains shared libraries and add a search path.

Common Tab

The "Common" tab is as shown in Figure 4-63.

Figure 4-63 Common Tab

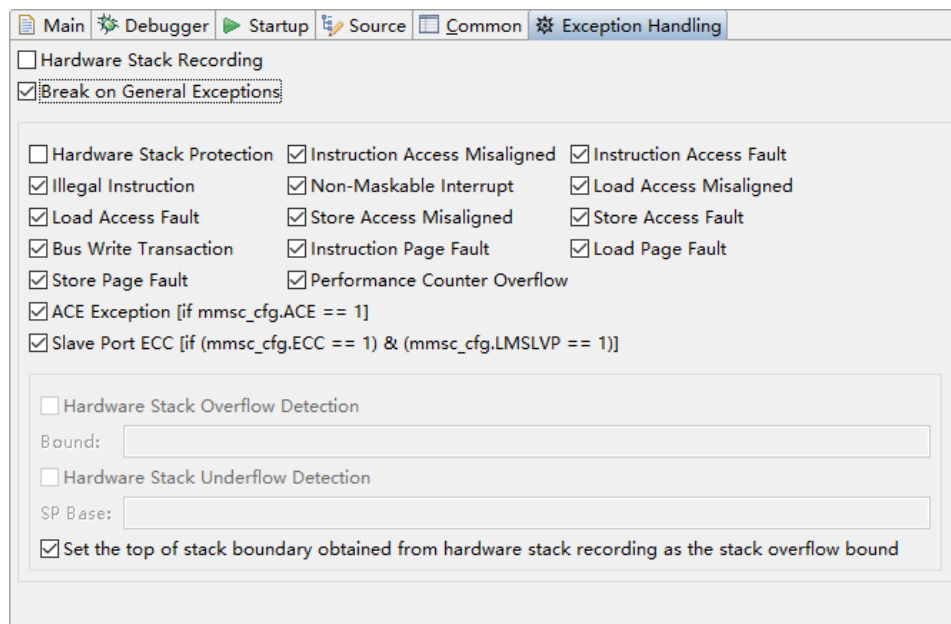
The descriptions of Common options are as shown in Table 4-5.

Table 4-5 Common Options

Option	Description
Save as	The launch configuration file is saved as a file (*.launch)
Standard Input and Output	Allocate console (necessary for input): Set the console view as an input or output prompt, as described in the 4.2.6 Console View Input File: Allocate standard inputs from project resources (click "Workspace...") or files (click "File system...") Output File: Allocate standard outputs from project resource (click "Workspace...") or file (click "File system...")

Exception Handling Tab

The "Exception Handling" tab is as shown in Figure 4-64.

Figure 4-64 Exception Handling Tab

This tab is used to enable hardware stack recording, which specifies the types of exceptions that must be captured during program execution, as described in the [Chapter 6 Generic Exception Handling and Stack Recording/Protection](#). Note that you cannot select "Hardware Stack Recording" and "Hardware Stack Protection" at the same time.

4.4 Breakpoints and Watchpoints

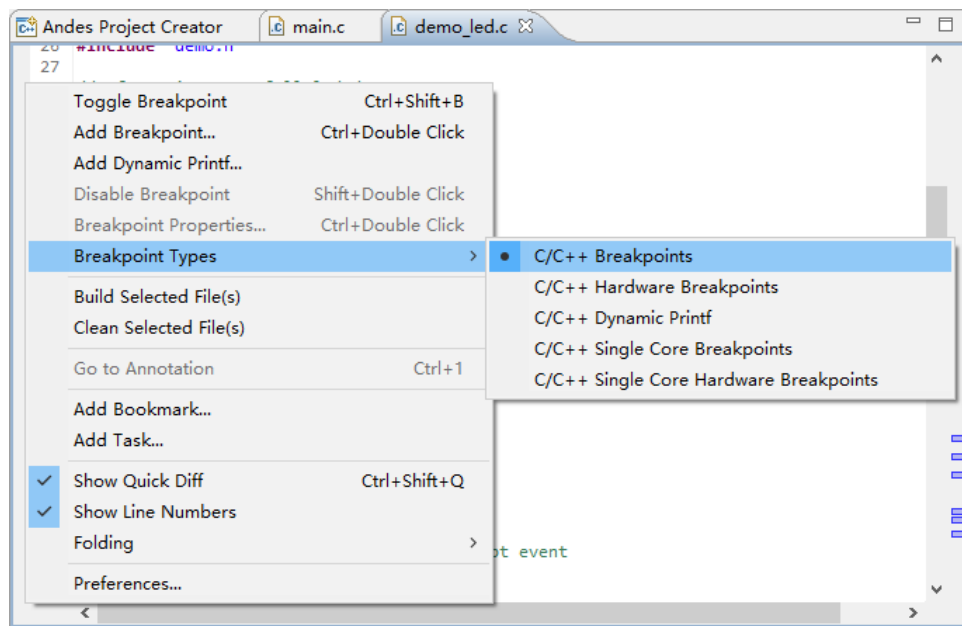
The number of breakpoints and watchpoints that can be set by the RDS software depends on the number of hardware breakpoints supported by the target; you can refer to the target datasheet to determine the number of hardware breakpoints or watchpoints supported.

4.4.1 Set Breakpoints

You can refer to the following steps to set breakpoints:

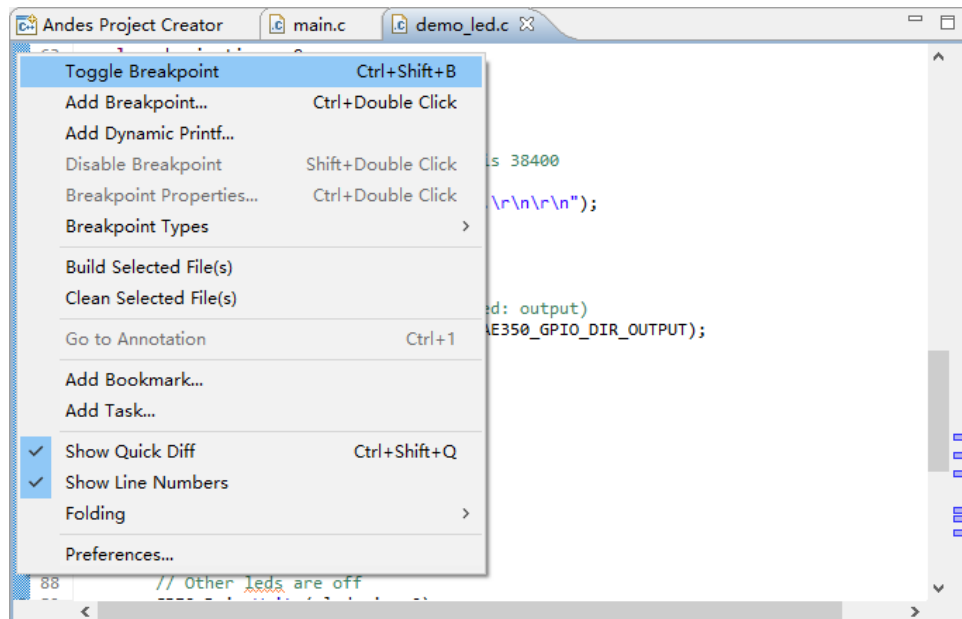
1. Open a source file in the Code Editor.
2. Right-click the ruler on the left edge of the code editor and select "Breakpoint Types > C/C++ Hardware Breakpoints" or "C/C++ Breakpoints" from the drop-down list to specify the type of breakpoint, as shown in Figure 4-65.

Figure 4-65 Select Breakpoint Types



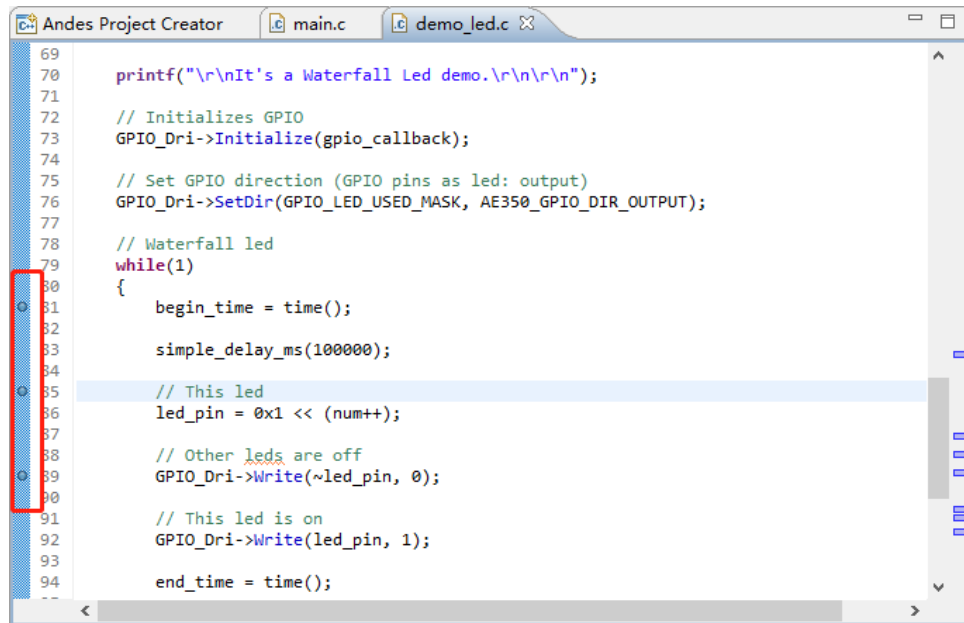
3. On the left side of the line where you need to set the breakpoint, double-click the ruler, or right-click the ruler, and select "Toggle Breakpoint" from the drop-down list, as shown in Figure 4-66.

Figure 4-66 Select Toggle Breakpoint



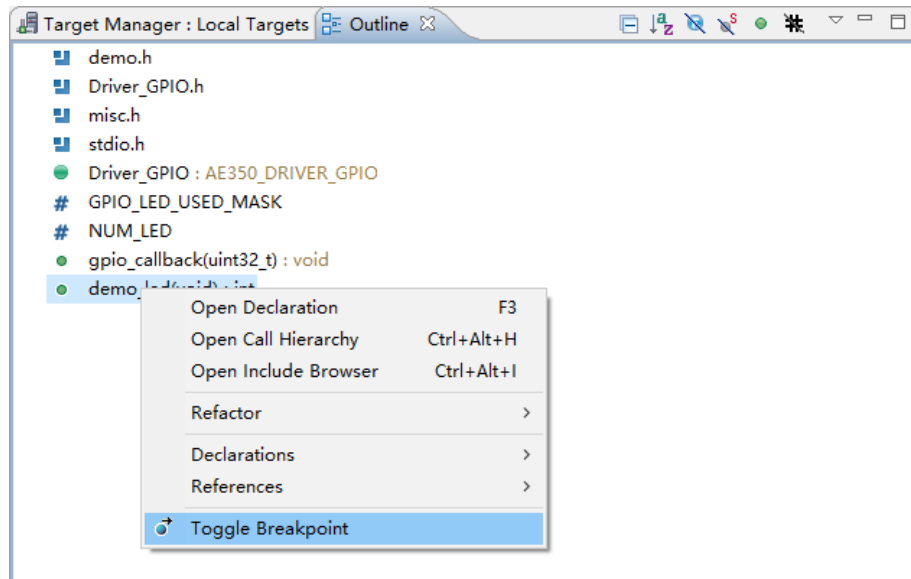
The breakpoint markers displayed on the ruler indicate where you want program execution to pause, as shown in Figure 4-67.

Figure 4-67 Breakpoints



You can also set breakpoints in the Outline View; right-click a function, and select "Toggle Breakpoint" in the drop-down list, as shown in Figure 4-68.

Figure 4-68 Select Toggle Breakpoint in Outline View



4. To specify the triggering conditions for a breakpoint, right-click on a breakpoint in the breakpoint view or on the code editor ruler and select "Breakpoint Properties..." in the drop-down menu, as shown in the following example.

Figure 4-69 Select Breakpoint Properties... in Breakpoints View

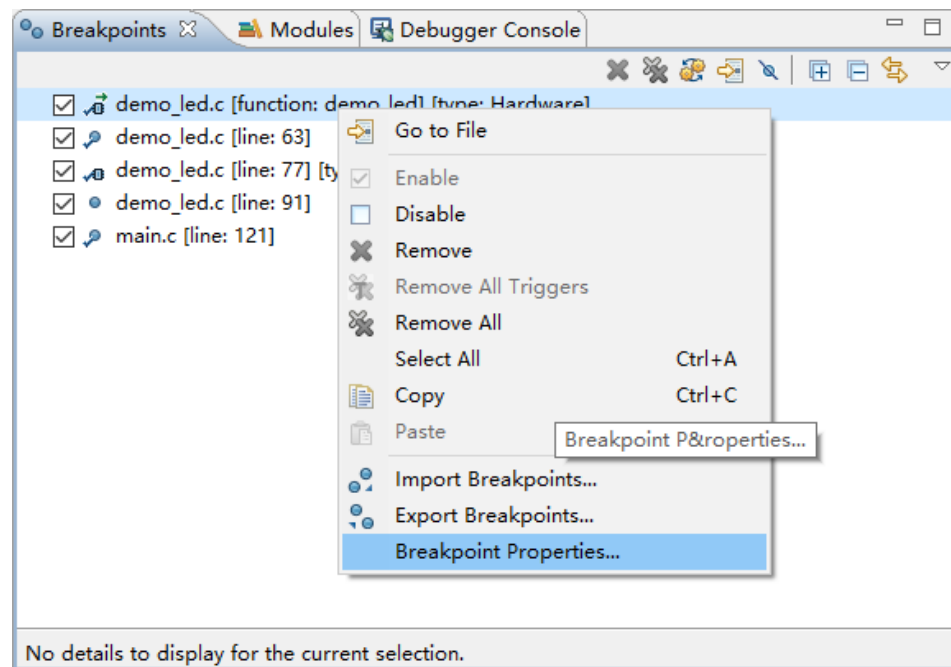
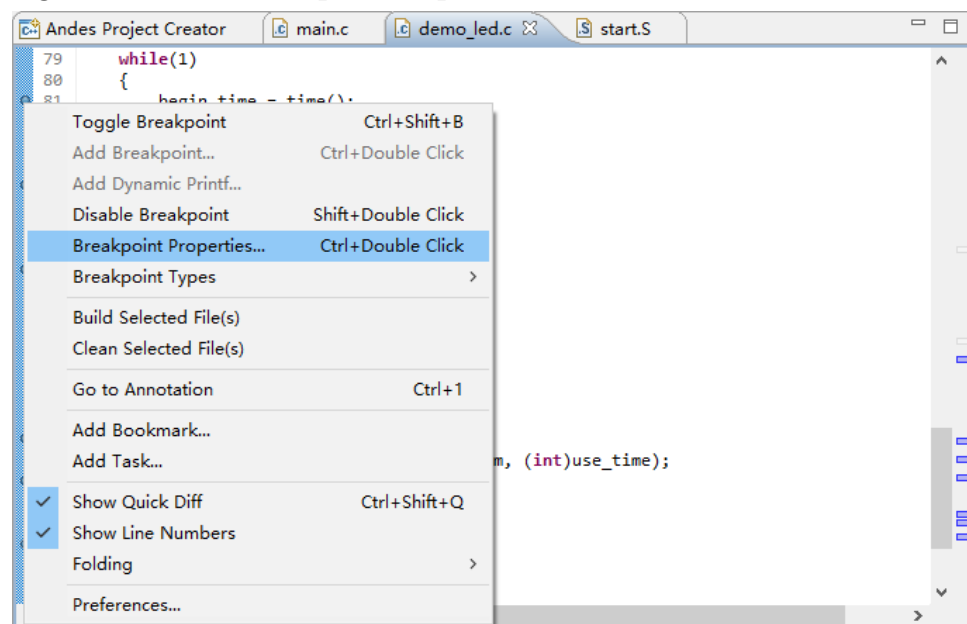
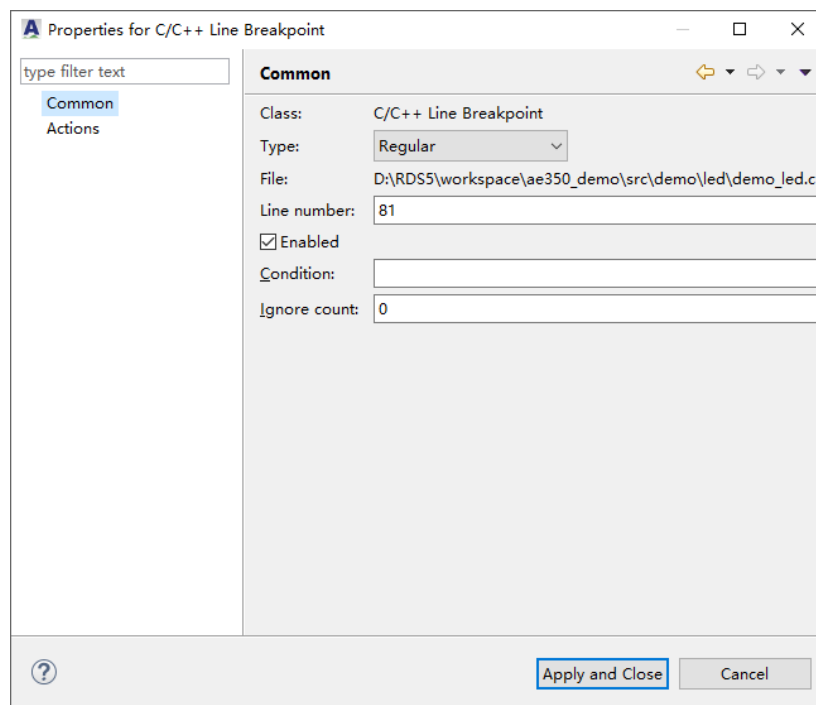


Figure 4-70 Select Breakpoint Properties... in Code Editor



In the "Properties for C/C++ Line Breakpoint" dialog box, select "Common", and enter the condition expression in "Condition" text box, then click "Apply and Close", as shown in Figure 4-71.

Figure 4-71 Properties for C/C++ Line Breakpoint

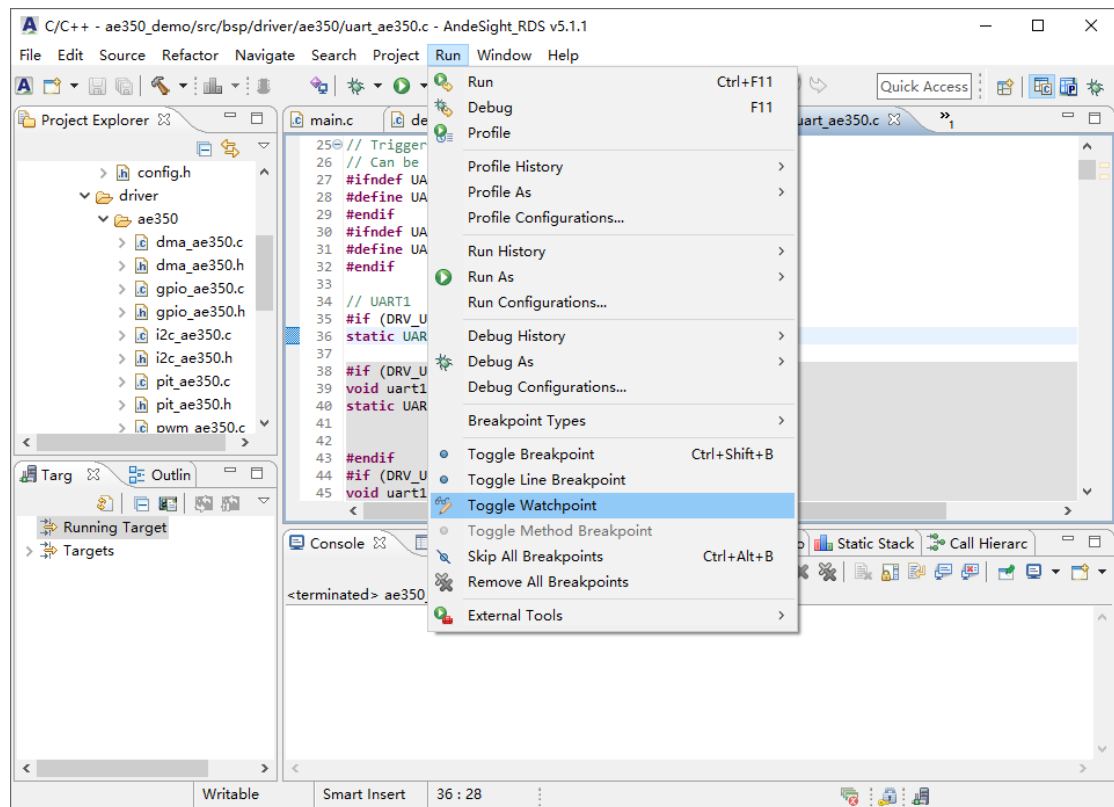
5. To clear a breakpoint, you can double-click the breakpoint marker in the code editor.

4.4.2 Set Watchpoints

Unlike breakpoints, watchpoints only pause program execution when the value of a variable or expression changes.

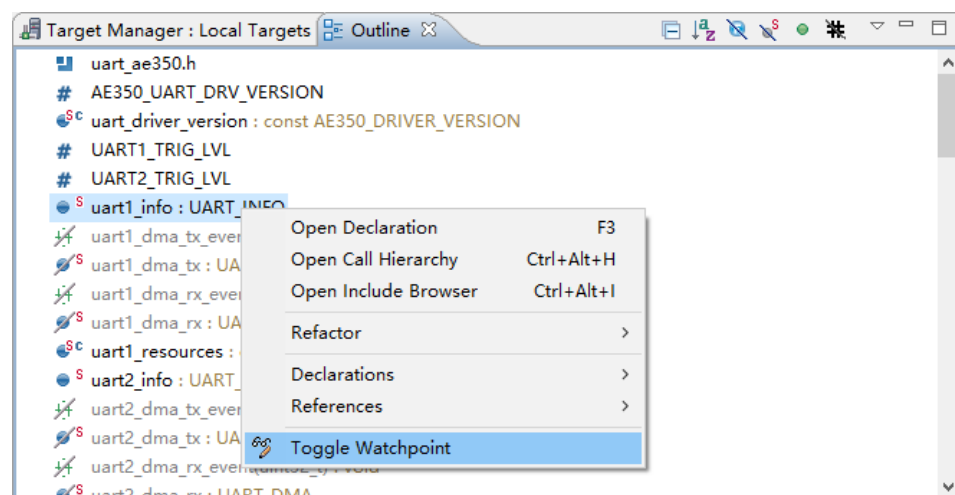
1. You can refer to the following methods to set watchpoints:
 - Code Editor: In the code editor, select the global variable you want to watch, and select "Run > Toggle Watchpoint" on the main menu of RDS software, as shown in Figure 4-72.

Figure 4-72 Select Run > Toggle Watchpoint



- Outline View: In the outline view, select the global variable you want to watch, and right-click on the global variable and select "Toggle Watchpoint" in the drop-down list, as shown in Figure 4-73.

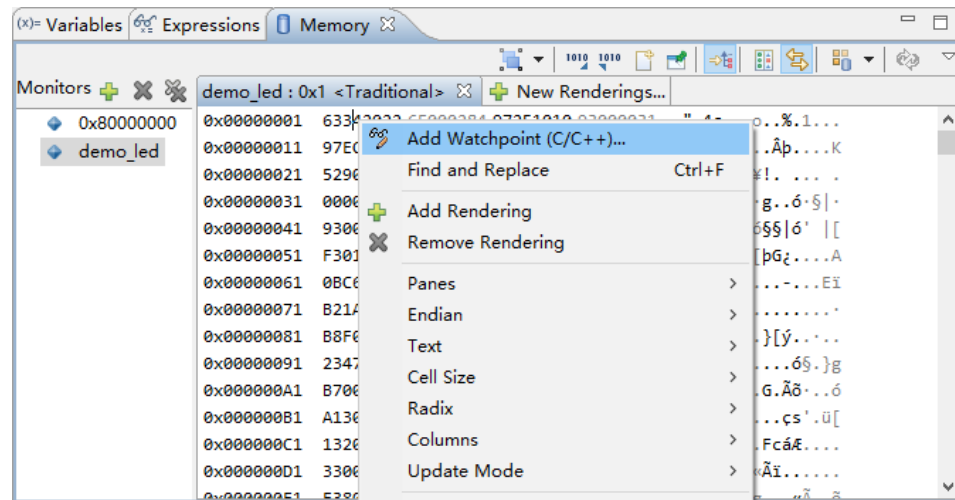
Figure 4-73 Select Toggle Watchpoint in Outline View



- Memory View: In the memory view, right-click on a addressable cell or range of cells in the Memory Rendering, and select "Add Watchpoint (C/C++)..." from the drop-down list, as shown in Figure

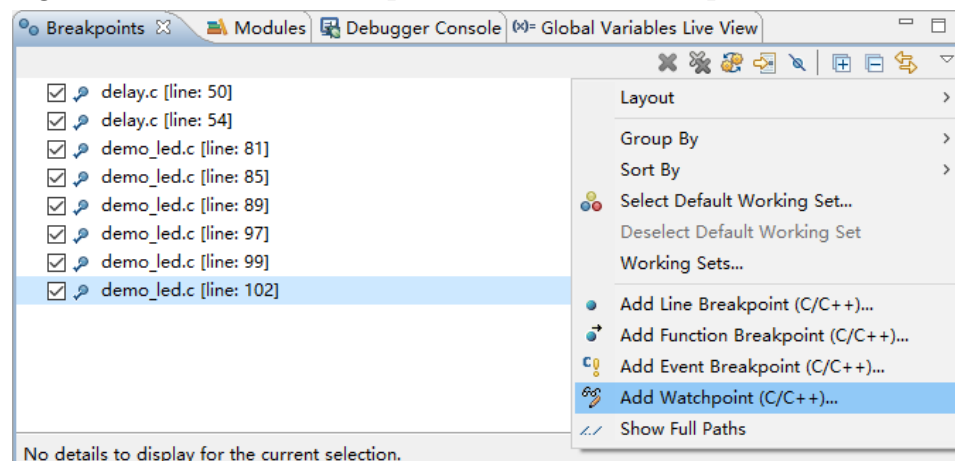
4-74.

Figure 4-74 Select Add Watchpoint (C/C++)... in Memory View

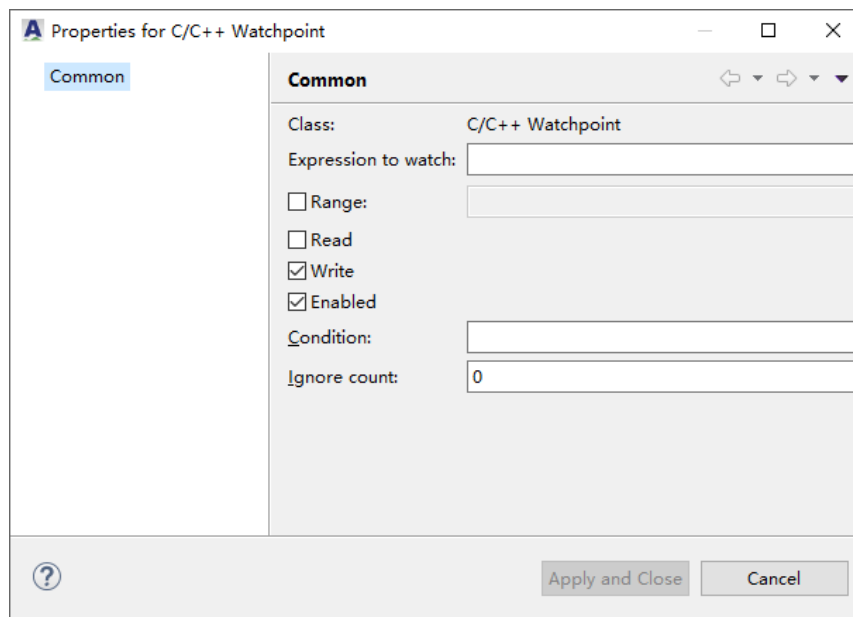


- Breakpoint View: In the Breakpoint View, click "▼" on the toolbar, and select "Add Watchpoint (C/C++)..." from the drop-down list, as shown in Figure 4-75.

Figure 4-75 Select Add Watchpoint (C/C++)... in Breakpoint View



2. In the "Properties for C/C++ Watchpoint" dialog box, specify the various settings for the Watchpoints, and click "Apply and Close", as shown in Figure 4-76.

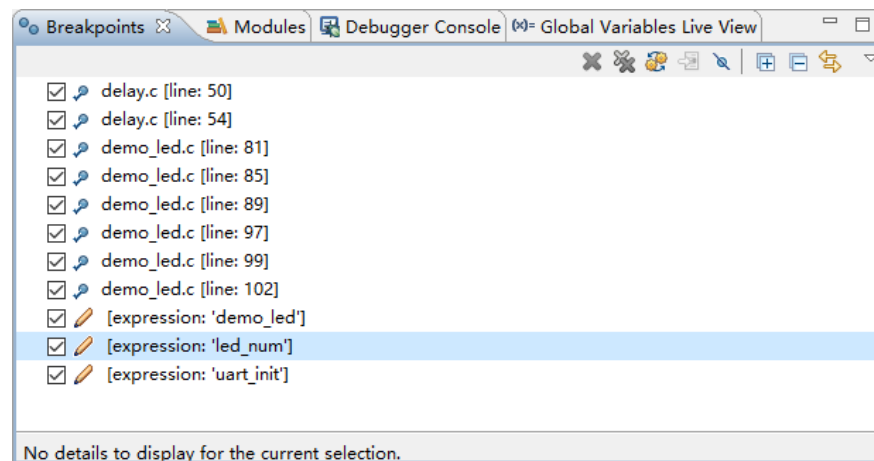
Figure 4-76 Properties for C/C++ Watchpoint

The descriptions of watchpoints options are as shown in Table 4-6.

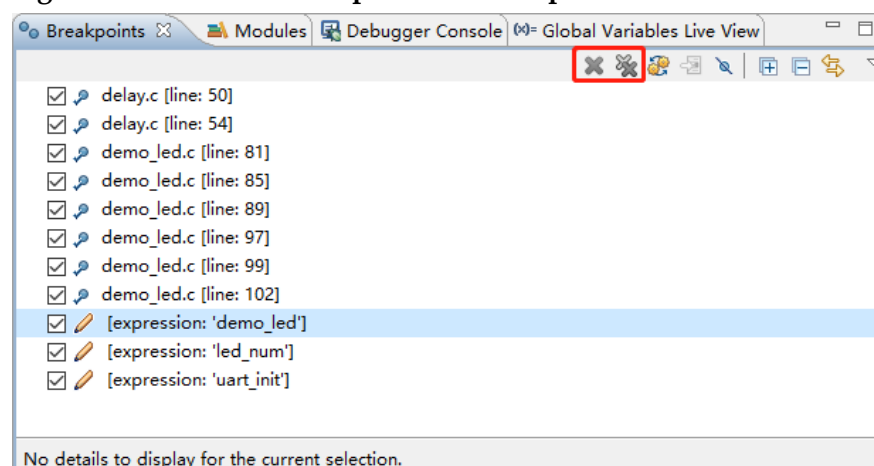
Table 4-6 Watchpoints Options

Option	Description
Expression to watch	To specify the area to be watched in the "Range" option, enter a constant or variable address (&i) here
Range	Specify an addressable cell to be watched by a watchpoint
Read	Stop program execution while reading a watched expression
Write	Stop program execution while writing a watched expression
Enabled	Enable a watchpoint
Conditions	Specify an expression when the watchpoint is hit
Ignore count	Specify the number changes in the value of the expression

3. The Breakpoints view lists all set watchpoints, with the watchpoint marker "🔍" for writing watchpoints and the "🔍" for reading watchpoints, as shown in Figure 4-77.

Figure 4-77 Watchpoints in Breakpoints View

4. To remove the watchpoints, select the watchpoints to be deleted in the Breakpoints view and click "✕" on the toolbar, as shown in Figure 4-78.

Figure 4-78 Remove Watchpoints in Breakpoints View

5 Download Software Project

Follow the steps below to download the software project:

1. Follow the steps in 2.4 Create a Build System and 2.5 Build a Software Project to create and build the software project, generating the software programming binary file.
2. Use Gowin FTDI USB programmer to connect the PC host to the target board (constrained JTAG interface of Gowin RiscV_AE350_SOC) to create the target system.
3. Use Gowin USB Cable to connect the PC host to the target board (FPGA Programming JTAG interface) for downloading the software programming binary file.
4. Click on the " (Flash Burner) on the RDS software toolbar, or right-click on the selected software project; choose "Flash Burner" from the drop-down menu to launch Gowin Programmer, as shown in Figure 5-1 and Figure 5-2.

Note!

CPU debugging and FPGA/CPU downloading can now simultaneously use Gowin FTDI USB programmer and WinUSB driver. Please prepare two Gowin FTDI USB cables, and both are installed with WinUSB driver. One is connected to the CPU JTAG interface for CPU debugging, and the other is connected to the FPGA JTAG interface for FPGA/CPU downloading.

Figure 5-1 Launch Flash Burner

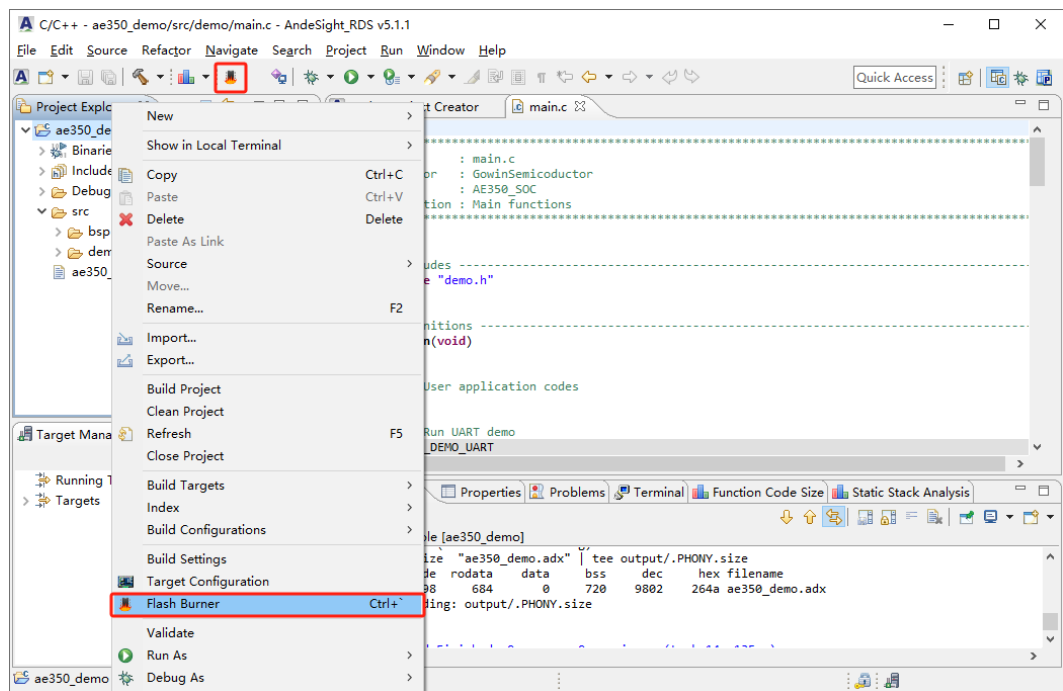
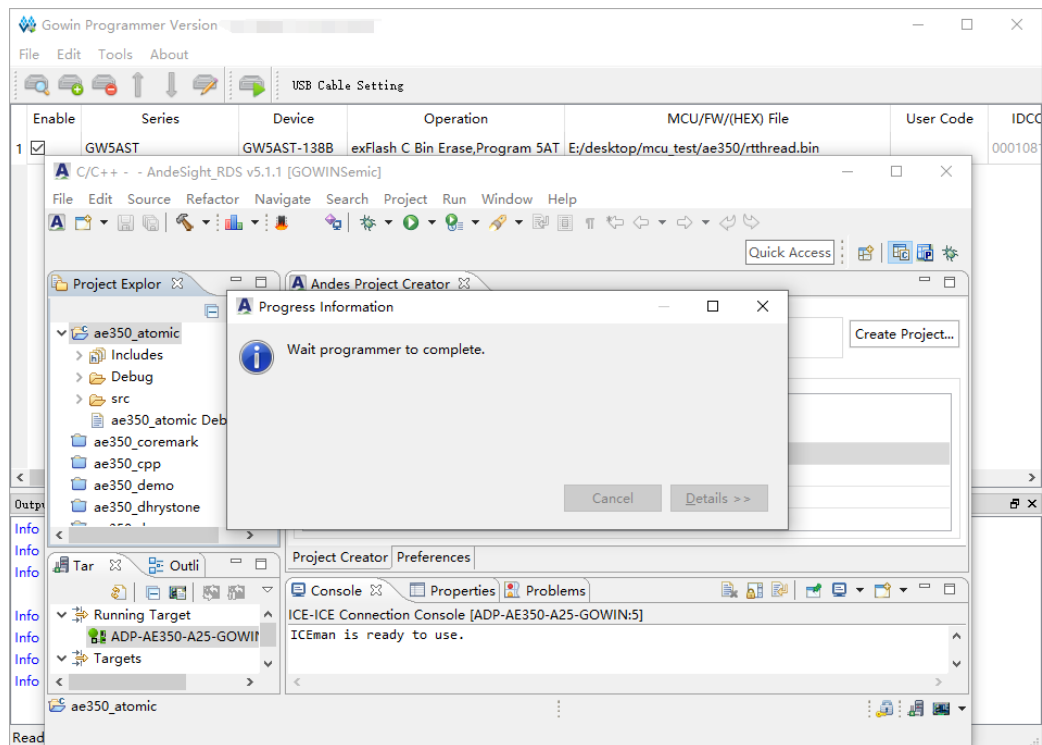
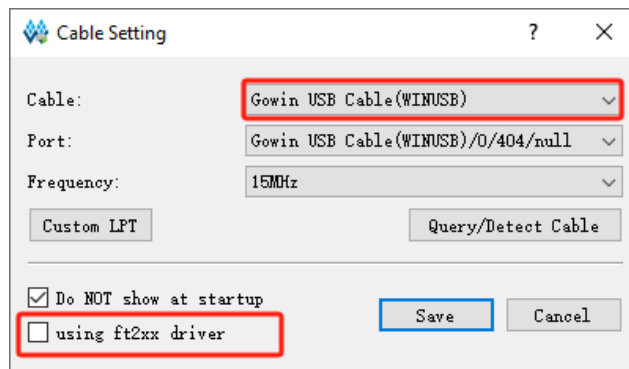


Figure 5-2 Launch Gowin Programmer



5. In Gowin Programmer, if using the WinUSB driver, first select "Cable SETTING" from the toolbar and uncheck the "using ft2xx driver option", as shown in Figure 5-4. Then select "Edit > Configure Device" from the menu or click "Configure Device" on the toolbar to configure the download options, as shown in Figure 5-4.

Figure 5-3 Cable Setting



An example of the download option configuration is as follows.

Access Mode: External Flash Mode 5A

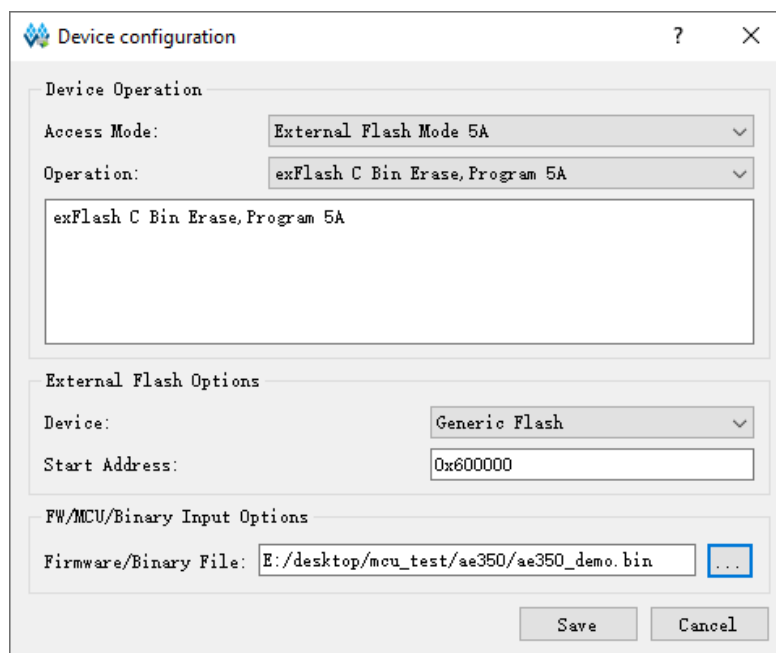
Operation: exFlash C Bin Erase, Program 5A

External Flash Options > Device: Generic Flash

External Flash Options > Start Address: 0x600000

FW/MCU/Binary Input Options > Firmware/Binary File:
ae350_demo.bin

Figure 5-4 Device Configuration



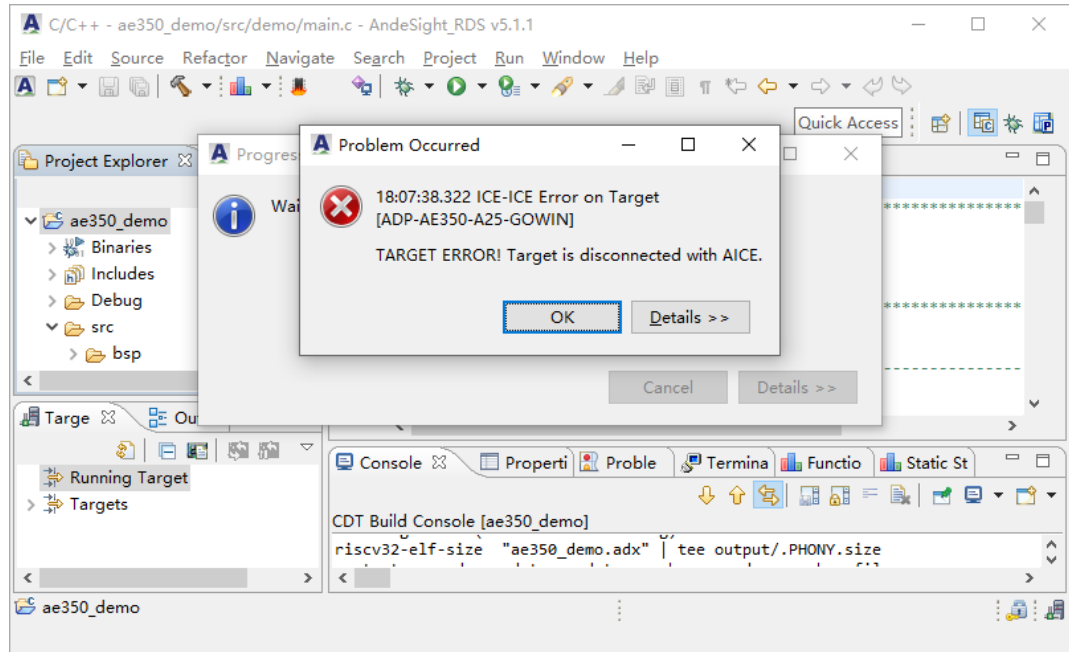
Click "Save" to complete the download configuration.

Click on the "🔧" (Program/Configure) on the toolbar to download the software programming binary file.

Note!

After the download is complete, the RDS software may display the message "TARGET ERROR! Target is disconnected with AICE", indicating that the RDS software is disconnected from the target board, as shown in Figure 5-5. This happens because after completing the download, the FPGA reloads the downloaded software programming binary file, causing the RDS software to disconnect from the target board. Click "OK" and close Gowin Programmer, then exit Flash Burner to resolve this.

Figure 5-5 Reload



6 Generic Exception Handling and Stack Recording/Protection

6.1 Stack Recording/Protection

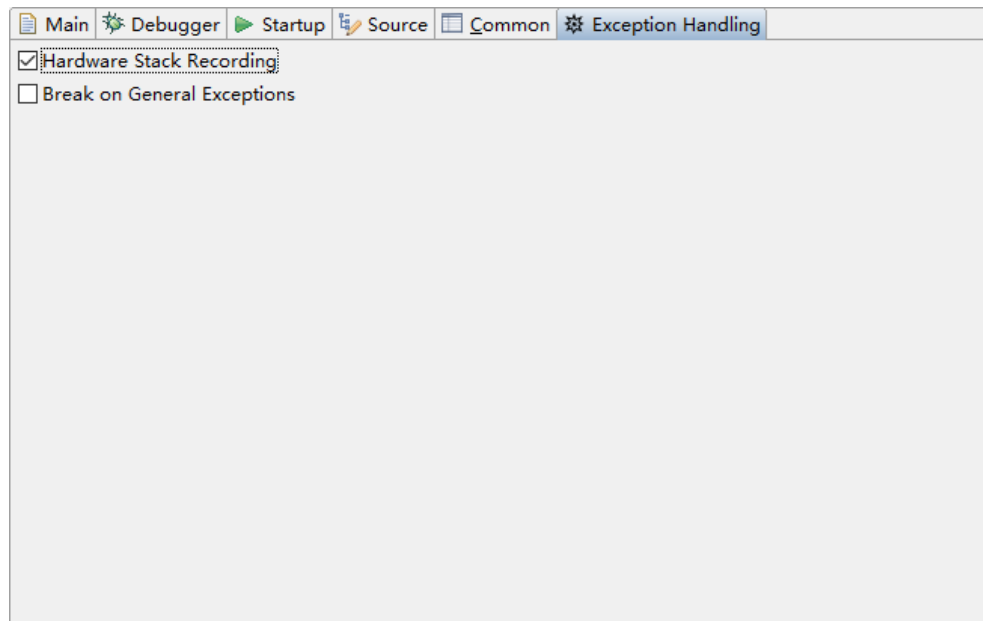
For CPU cores that support hardware stack protection, the RDS software allows hardware top-of-stack recording or hardware stack protection, but these two functions cannot be used at the same time. Hardware top-of-stack recording function can record the top-of-stack boundary while the program is running, and help the user to find the amount of stack memory required by the application. Hardware stack protection function can report stack overflow or underflow problems.

The following conditions must be met:

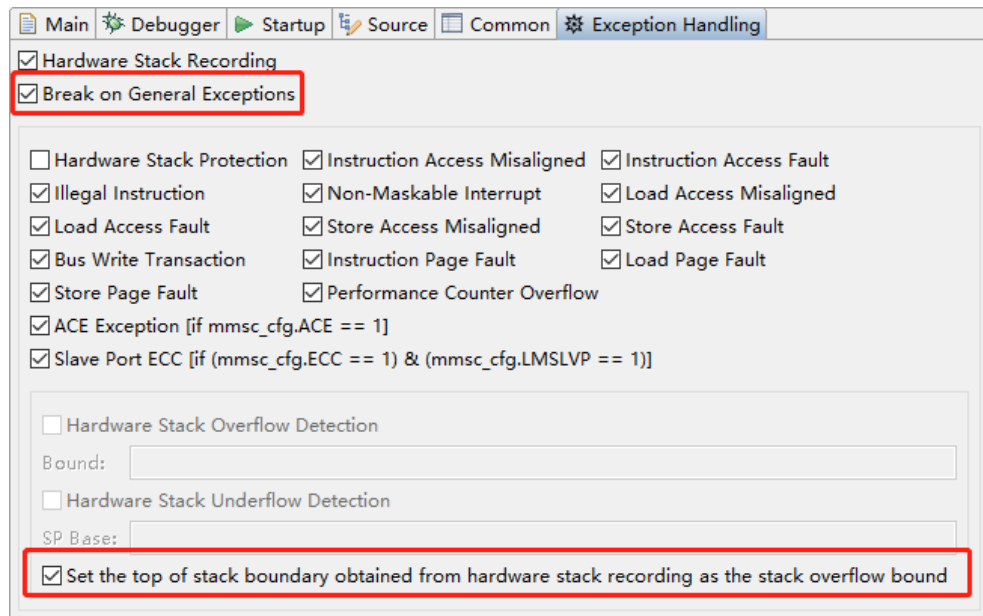
- The values of system registers used for hardware stack protection cannot be modified using program code.
- The number of available hardware breakpoints is limited and one must be reserved for use by the stack protection handler.

You can refer to the following steps to enable the hardware stack recording function:

1. Refer to steps 1 and 2 in [4.3.1 Create Debug Configurations](#) to create a debug session configuration; in the "Debug Configurations" dialog box, click the "Exception Handling" tab and select "Hardware Stack Recording", as shown in Figure 6-1.

Figure 6-1 Select Hardware Stack Recording

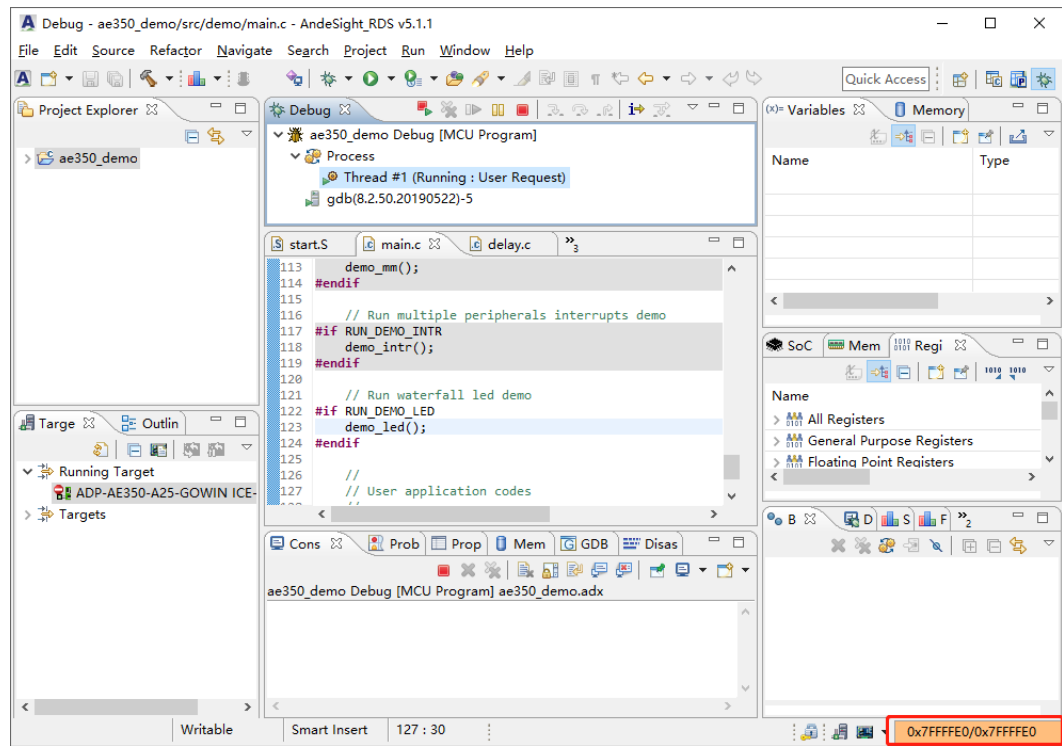
If you want to set the top of stack boundary recorded in the debugging session as the stack overflow boundary, select "Break on General Exceptions > Set the top of stack boundary obtained from hardware stack recording as the stack overflow bound", as shown in Figure 6-2.

Figure 6-2 Set the Top of Stack Boundary

2. In the "Debug Configurations" dialog box, click "Apply" and "Debug" to start the debugging session.
3. After starting a debugging session, the RDS software automatically enters the debugging perspective; click "▶" on the toolbar to resume

executing the program. The status bar in the lower right corner of the RDS software displays the top value of the stack boundary when the program is running, as shown in Figure 6-3.

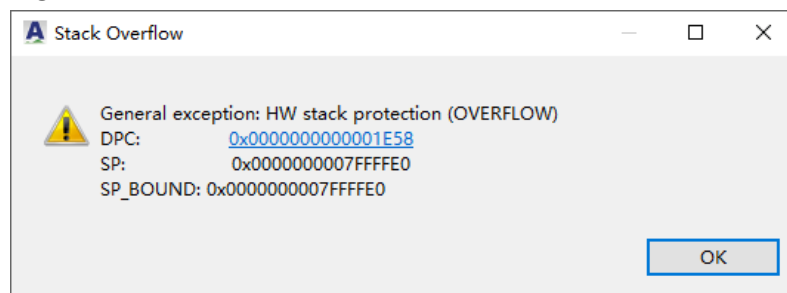
Figure 6-3 Display Stack Boundary



6.2 Generic Exception Handling

The RDS software allows program execution to be stopped when a selected generic exception occurs, and the "Stack Overflow" dialog box prompts exception information, as shown in Figure 6-4.

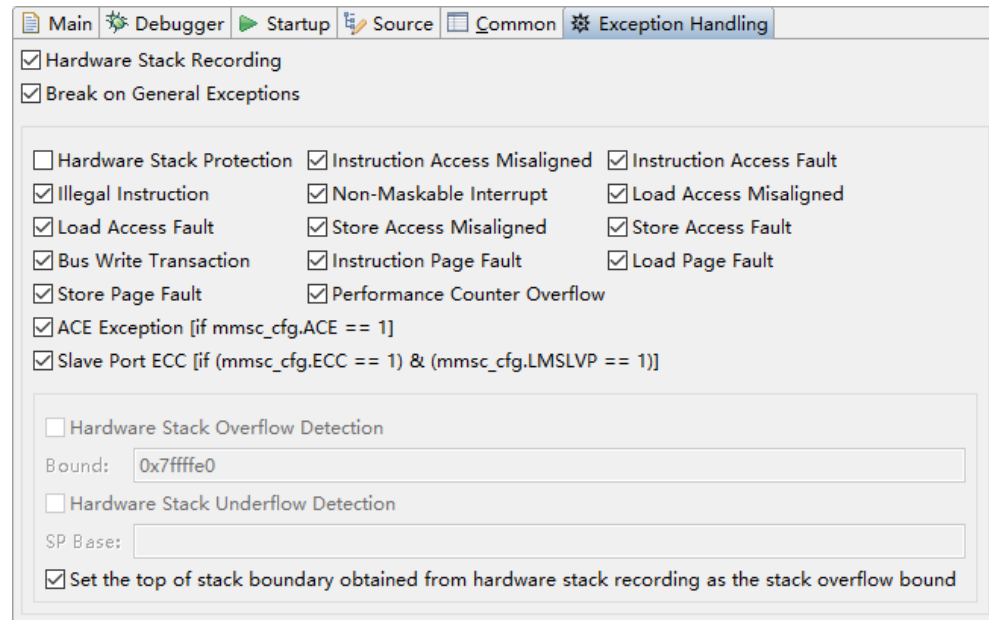
Figure 6-4 Stack Overflow



You can refer to the following steps to learn how to use the RDS software to catch the generic exceptions during a debugging session. If the interrupt vector is in ROM or Flash, the exception specified during program execution occupies a hardware breakpoint, in which case it is important to ensure that there are enough hardware breakpoints available in the target system.

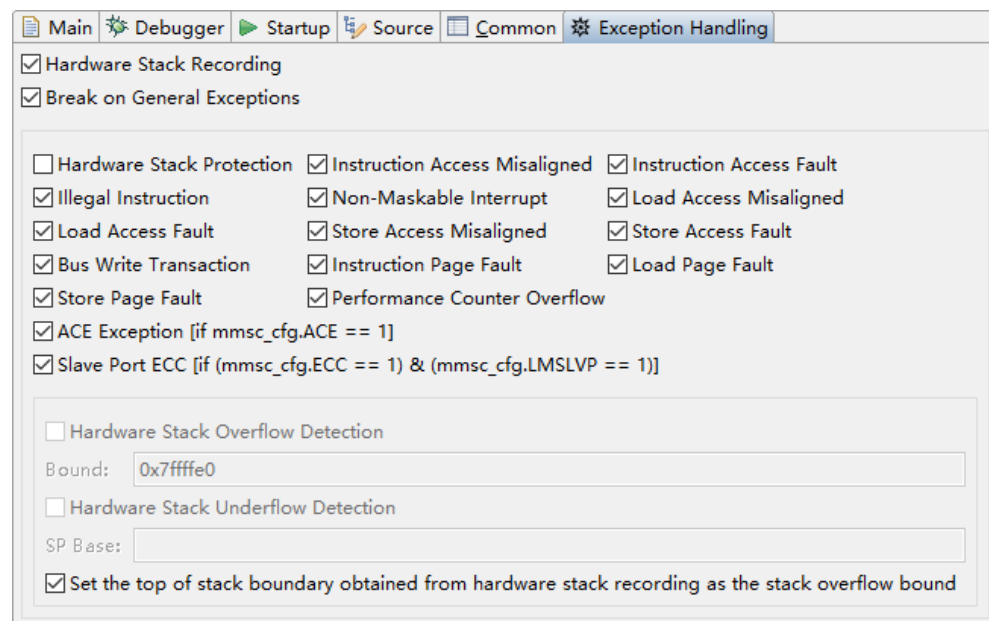
1. You can refer to steps 1 and 2 in [4.3.1 Create Debug Configurations](#) to create a debug session configuration; in the "Debug Configurations" dialog box, click the "Exception Handling" tab and select "Break on General Exceptions", as shown in Figure 6-5.

Figure 6-5 Select Break on General Exceptions



2. Specify one or more exceptions to be caught during program execution, as shown in Figure 6-6.

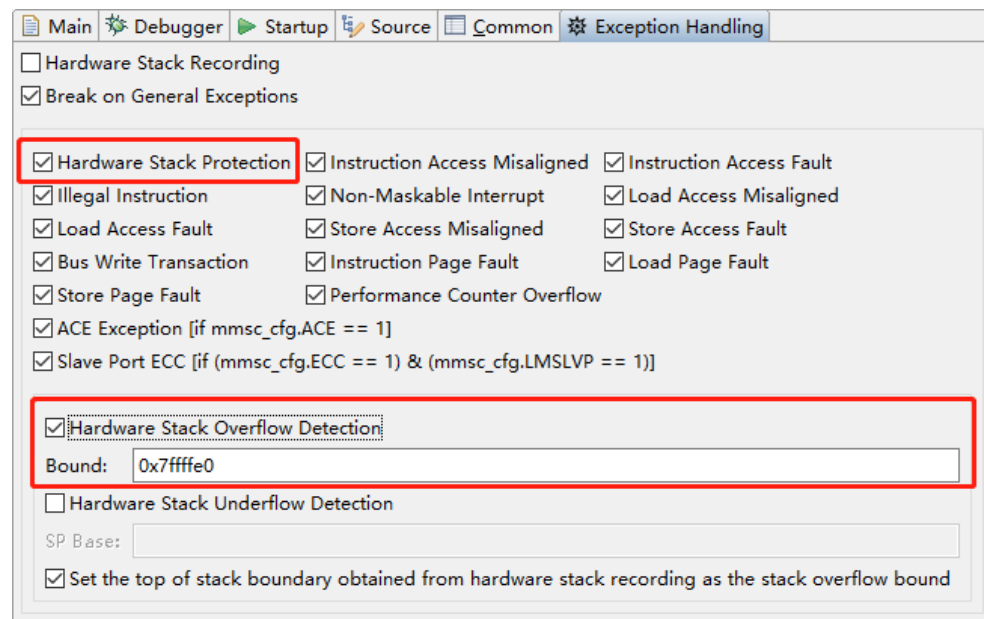
Figure 6-6 Specify Exceptions



For "Hardware Stack Protection", you must specify whether to perform "Hardware Stack Overflow Detection" and/or "Hardware Stack Underflow

Detection", as shown in Figure 6-7. For "Hardware Stack Overflow Detection", you can input the value of the stack bound register in "Bound:" text box, which is used to compare with the value of the Stack Pointer (SP) register. For "Hardware Stack low Detection", you can input the value of the stack base address register in "SP Base:" text box, which is used to compare with the value of the Stack Pointer (SP) register. If "Hardware Stack Recording" and "Set the top of stack boundary obtained from hardware stack recording as the stack overflow bound" have been selected, the "Bound:" of "Hard Stack Overflow Detection" will be automatically set to the maximum value of the recorded stack bound.

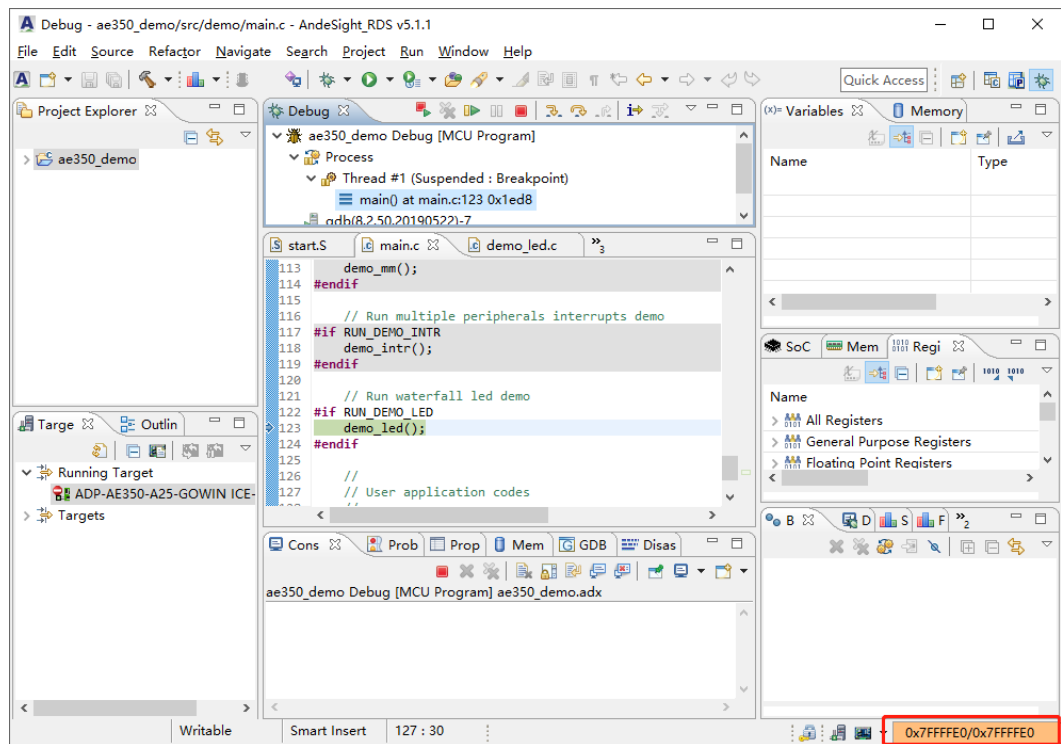
Figure 6-7 Select Hardware Stack Protection



3. In the "Debug Configurations" dialog box, click "Apply" and "Debug" to start a debugging session.
4. After starting a debugging session, the RDS software automatically enters the debugging perspective; set breakpoints in the source code, and click "▶" on the toolbar to resume executing the program.

If "hardware Stack Protection" is specified, the status bar at the bottom right corner of the RDS software displays the stack protection status of the running program, as shown in Figure 6-8.

Figure 6-8 Display the Stack Protection Status



5. Whenever a specified exception occurs, a dialog box pops up to prompt. To locate the location in the source code where the exception occurred, simply click on the IPC value in the dialog box.

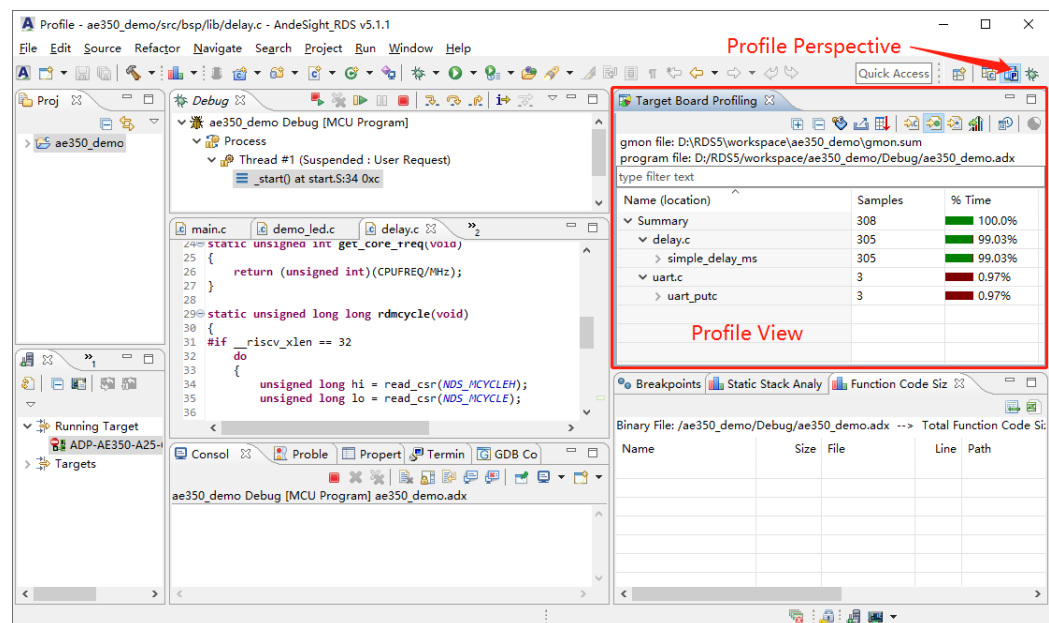
7 Profile Software Project

RDS software can perform real-time analysis of software projects built with the GCC compiler on the target board. When the program pauses due to breakpoints or single-stepping pause, it immediately collects, processes, and analyzes relevant information.

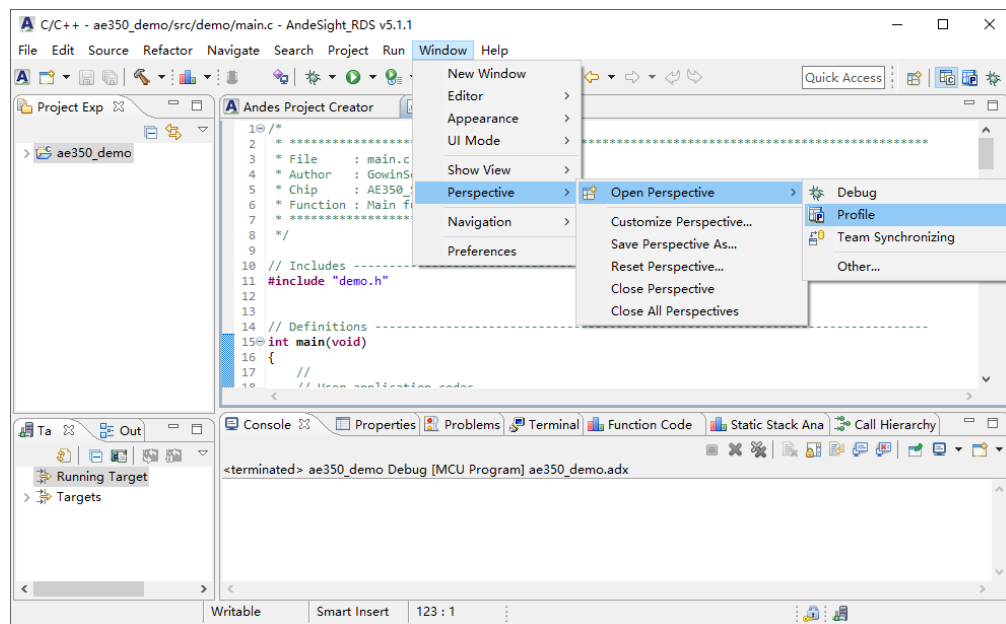
7.1 Profile Perspective

The Profile Perspective contains the Profile View, which is used to display the profile data, as shown in Figure 7-1.

Figure 7-1 Profile Perspective



After the Profile Session is started, the RDS software automatically enters the Profile Perspective, or manually enters it by selecting "Window > Perspective > Open Perspective > Profile" on the main menu of the RDS software, as shown in Figure 7-2.

Figure 7-2 Select Window > Perspective > Open Perspective > Profile

Alternatively, you can click on the "📊" (Profile) on the toolbar to enter the Profile Perspective.

7.2 Target Board Profiling View

The Target Board Profiling View is used for an analysis display of the target system, displaying which parts of the program take up the most execution time, providing information about calling each function, and allowing you to create graphs using the analysis data, as shown in Figure 7-3. This view provides a real-time interface for tracking and checking bottlenecks in program execution.

Figure 7-3 Target Board Profiling View

Name (location)	Samples	% Time
Summary	308	100.0%
delay.c	305	99.03%
simple_delay_ms	305	99.03%
> simple_delay_ms (delay.c:56)	305	99.03%
uart.c	3	0.97%
uart_putc	3	0.97%
> uart_putc (uart.c:59)	3	0.97%

As part of the Profile Perspective, the Target Board Profiling View is

automatically launched after starting a profile session. Alternatively, you can select "Windows > Show View > Other..." on the main menu of the RDS software, as shown in Figure 7-4. In the "Show View" dialog box, select "Profile > Target Board Profiling" to manually start the target board profiling view, as shown in Figure 7-5.

Figure 7-4 Select Windows > Show View > Other...

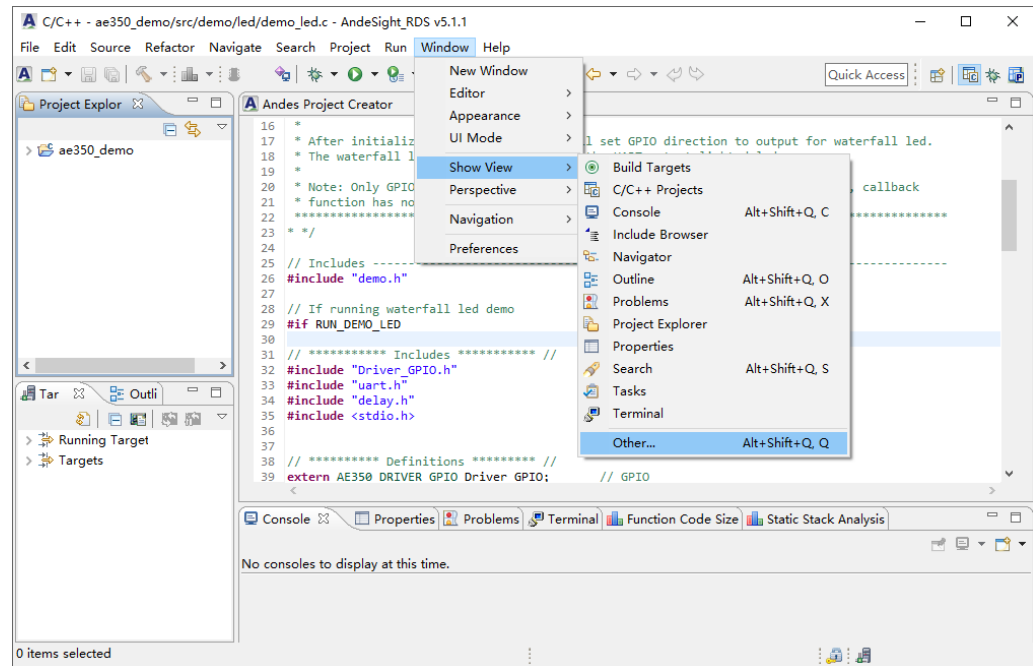
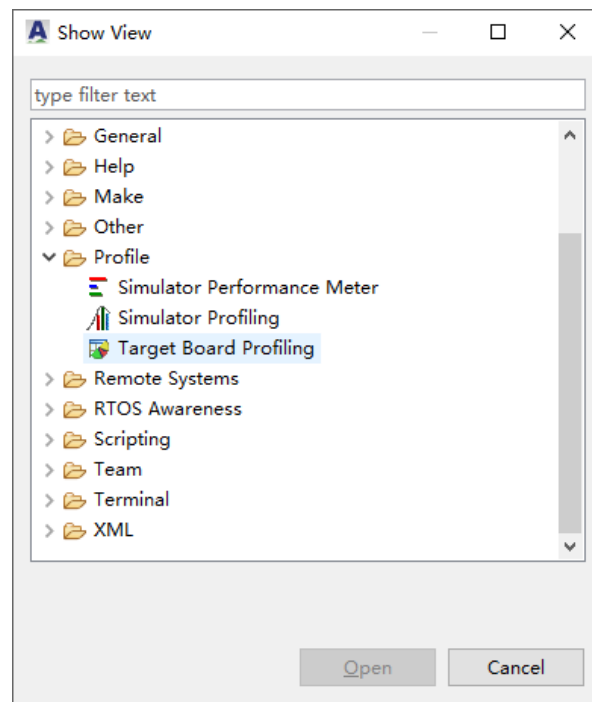


Figure 7-5 Select Profile > Target Board Profiling



The columns of the target board profiling view are shown in Table 7-1.

Table 7-1 Columns of the Target Board Profiling View

Column	Description
Name (location)	List of analyzed source files and their instructions
Samples	Instruction sampling counters, with an instruction call sampled every 10 milliseconds
%Time	Percentage of total program execution time occupied by the execution time of specific instructions

The toolbar of the target board profiling view includes:

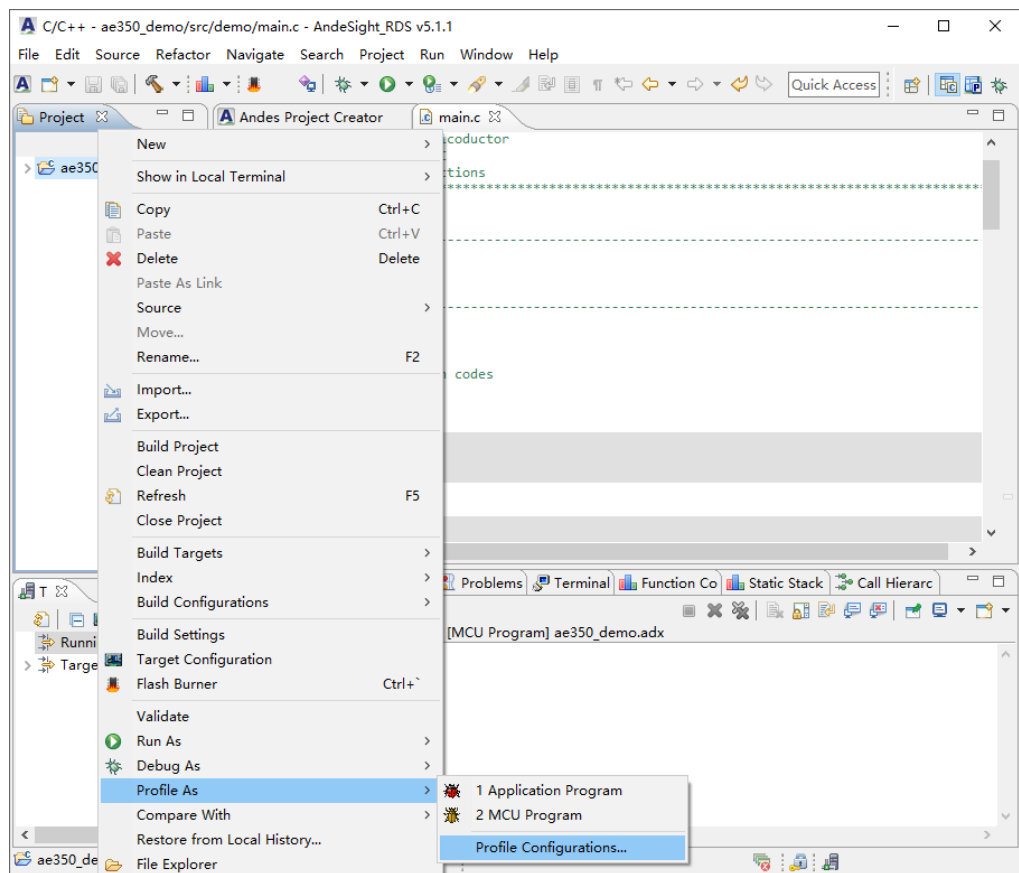
-  (Show/Hide columns): In the "Show/Hide columns" dialog box, specify the columns to be displayed in the Target Board Profiling View.
-  (Export to CSV): Export the analysis data to a .csv file.
-  (Sorting): In the Sorting dialog box, specify the order in which the analyzed data should be arranged in the Target Board Profiling View.
-  (Sort samples per file): Sort the analysis data by file.
-  (Sort samples per function): Sort the analysis data by function.
-  (Sort samples per line): Sort the analysis data by line.
-  (Reset): The analyzed data is set to zero.
-  (Switch samples/time): Data is switched from sample information to time information or vice versa.
-  (Create chart...): Create a bar chart, vertical bar chart, or pie chart using the specified analysis data.

7.3 Profile Session

You can refer to the following steps to create a profile session:

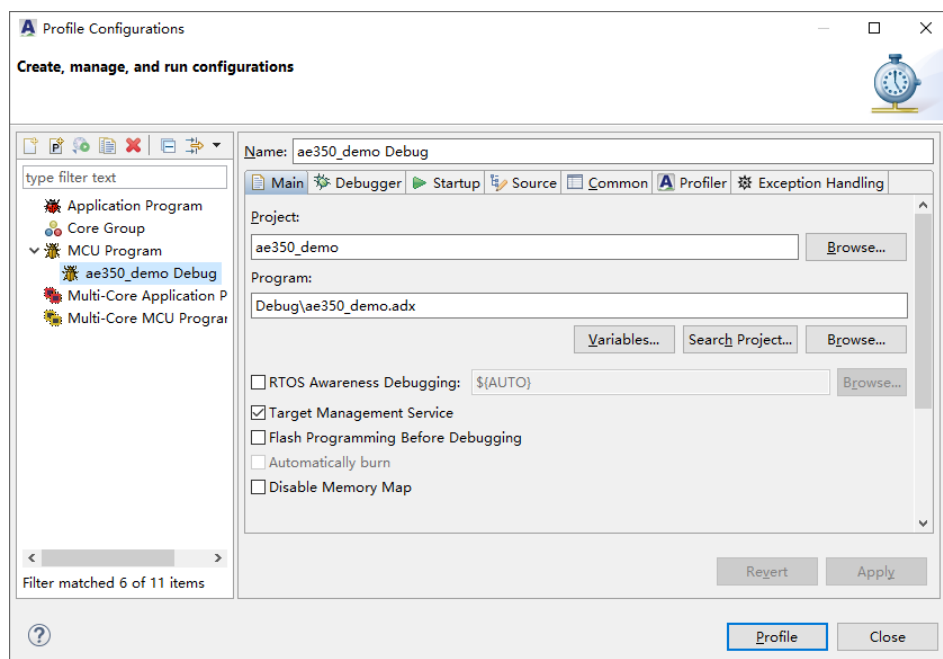
1. You can create and build a software project as described in Chapter 2 Create and Build a Software Project.
2. In the Project Explorer view, right-click to select a software project, and select "Debug As > Debug Configurations..." from the drop-down list, as shown in Figure 7-6.

Figure 7-6 Profile As > Profile Configurations...



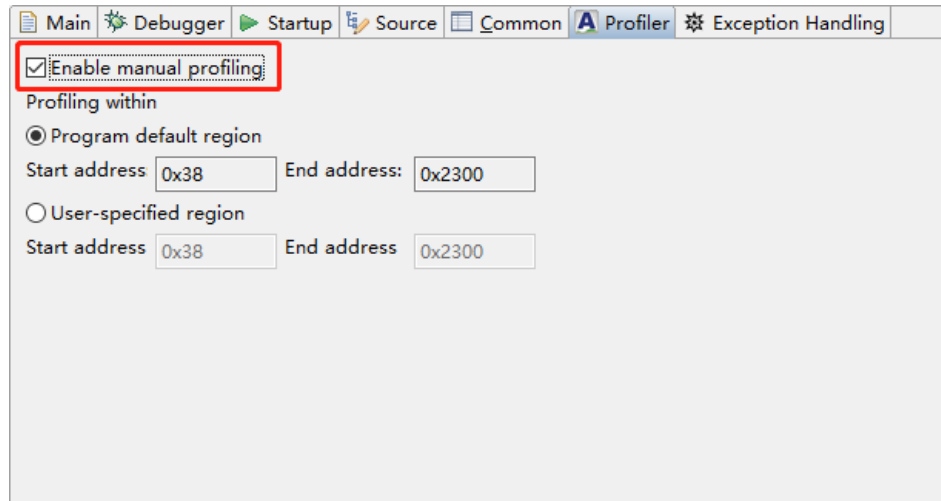
In the "Profile Configurations" dialog box, double-click the configuration type "MCU Program" to create a profile configuration, as shown in Figure 7-7.

Figure 7-7 Create Profile Configurations



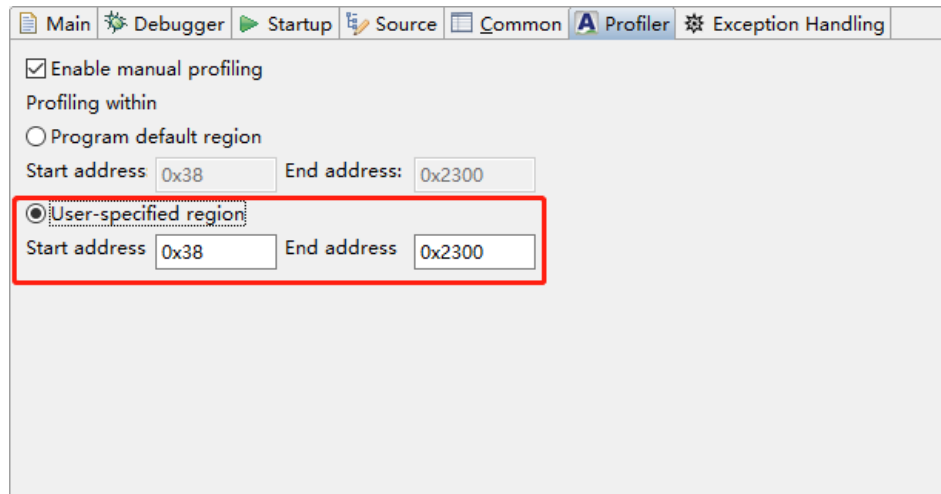
In the "Profiler" tab, if you want to manually start the profile, you can select "Enable manual profiling"; if it closes, RDS software automatically start the profile, and you can click "Profile", as shown in Figure 7-8.

Figure 7-8 Select Enable manual profiling

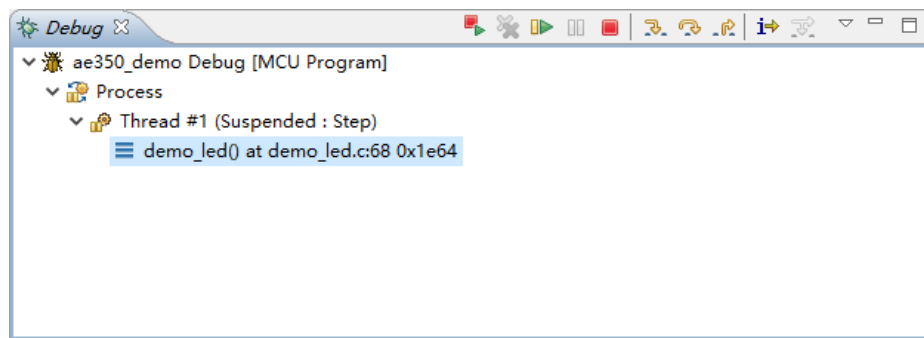


The ".text" segment is the default segment to be analyzed, if you want to change the region to be analyzed, select "User-specified region" and specify the start and end address of the region to be analyzed, as shown in Figure 7-9.

Figure 7-9 Select User-specified region



- When a profile session is started, the RDS software automatically enters the Profile Perspective and you can click on "▶" (Resume), "🔍" (Step Info), or "🔗" (Step Over) on the toolbar of Debug View to continue executing the program, as shown in Figure 7-10.

Figure 7-10 Debug View

For profile session in which "Enable manual profiling" is not selected, the Target Board Profiling View automatically generates and displays the analysis data immediately after program execution is paused or stopped, as shown in Figure 7-11.

Figure 7-11 Analysis Data in Target Board Profiling View

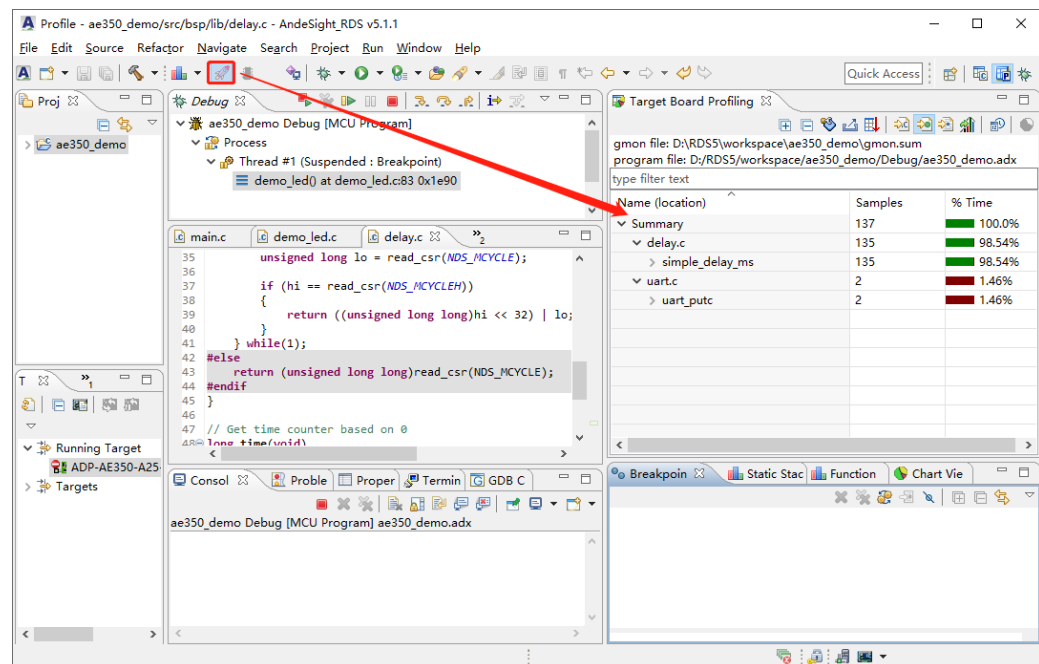
The screenshot shows the 'Target Board Profiling' window with the following data:

gmon file: D:\RDS5\workspace\ae350_demo\gmon.sum
 program file: D:\RDS5\workspace\ae350_demo\Debug\ae350_demo.adx
 type filter text

Name (location)	Samples	% Time
Summary	308	100.0%
simple_delay_ms	305	99.03%
simple_delay_ms (delay.c:56)	305	99.03%
0x1ac8	299	97.08%
0x1acc	6	1.95%
uart_putc	3	0.97%
uart_putc (uart.c:59)	3	0.97%
0x1e22	3	0.97%

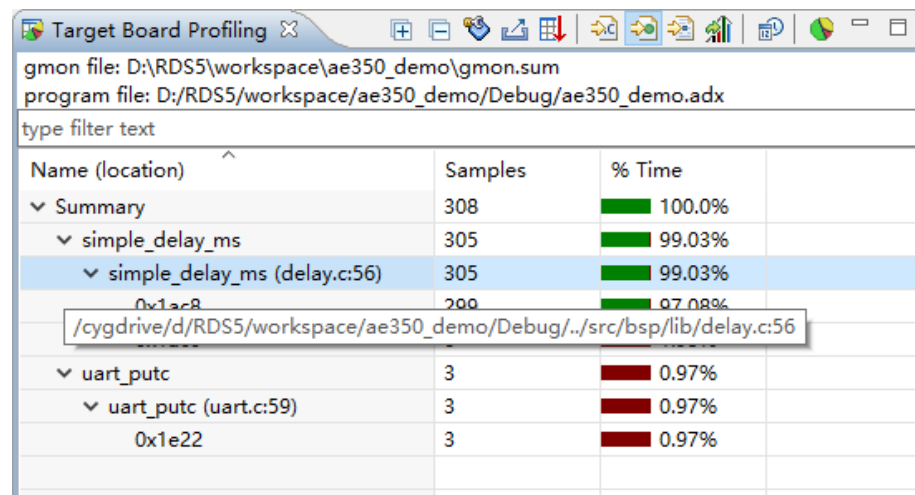
For profile session in which "Enable manual profiling" is selected, you can set breakpoints as described in [4.4.1 Set Breakpoints](#). Click "▶" on the toolbar to continue the execution of the program; when the program reaches the breakpoints, it pauses; at this time, click "🚀" (Start Profiling) on the toolbar, the target board profile view immediately displays the analysis data; if you click "🚀" again, the profile will be stopped, as shown in Figure 7-12.

Figure 7-12 Enable Manual Profiling



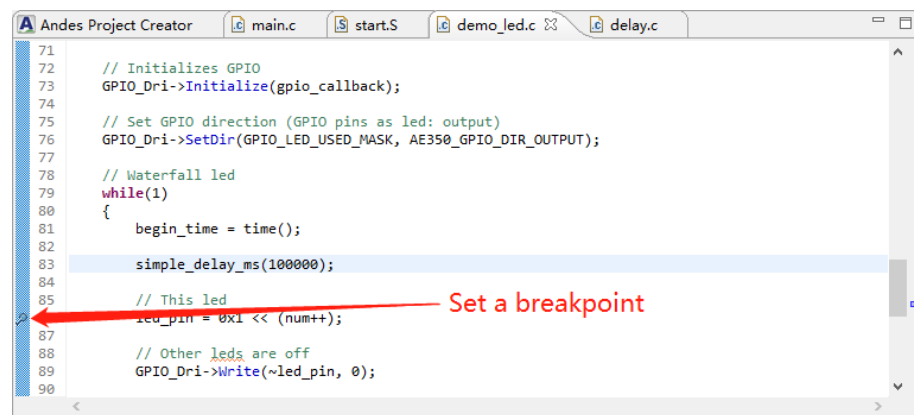
- According to the analyzed data, you can find the bottleneck of the program execution, double-click on the function you want to view, and directly locate the source code, as shown in Figure 7-13.

Figure 7-13 Locate Source Code



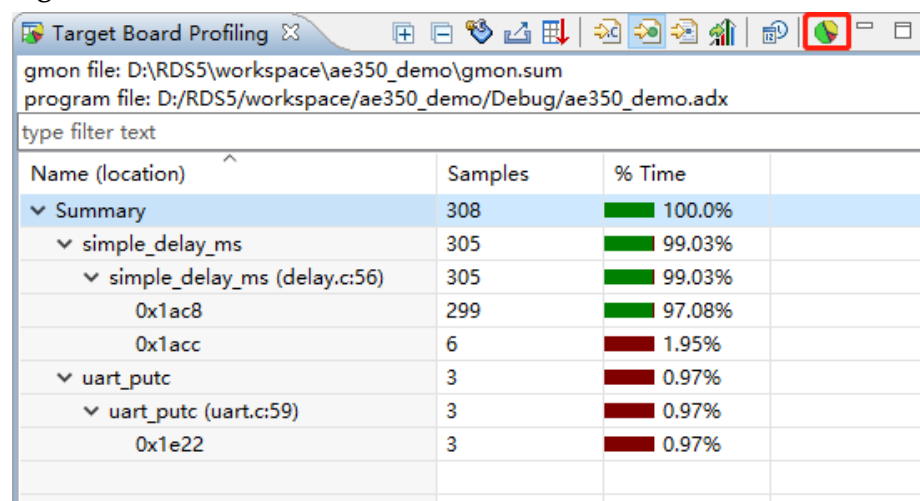
- Set breakpoints in the source code at the statements of the functions that you are interested in, as shown in Figure 7-14.

Figure 7-14 Set Breakpoints



6. Click "🔍" (Profile) on the toolbar of the RDS software to start the profile session and pause the program execution at the "main" function.
7. Click "▶" (Resume) on the toolbar of Debug View to continue the program execution and pause at the first set breakpoint.
8. Click "🔍" (Step Info) or "🔍" (Step Over) on the toolbar of Debug View to step through the statements of the function you are interested in.
9. In the target board profile view, review the analysis data again and refer to the following steps to generate a chart to view the analysis data:
 - In the Target Board Profile View, select the file or function you want to view and click "📊" (Create chart...) on the toolbar in target board profile view, as shown in Figure 7-15.

Figure 7-15 Select Create chart...



- In the "Create chart from selection..." dialog box, configure "Select

your char type" and "Select the column(s) to show", click "OK", then the chart is generated, as shown in Figure 7-16 and Figure 7-17.

Figure 7-16 Create chart from selection...

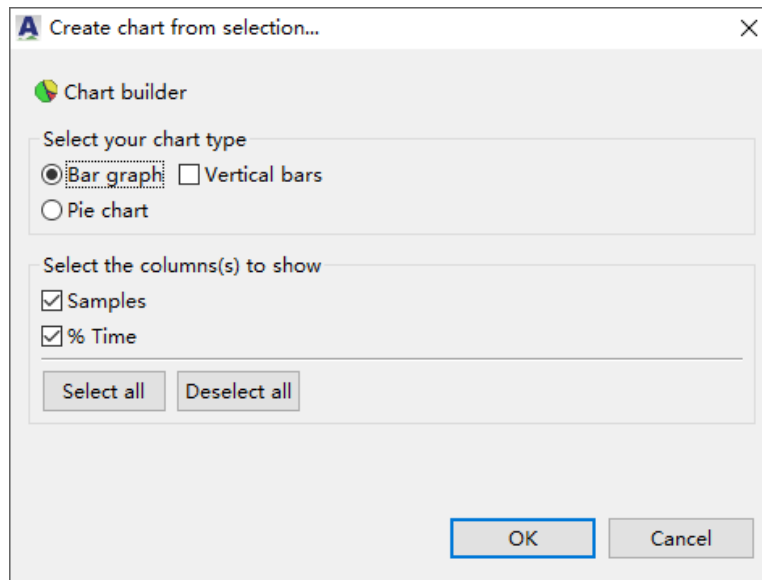
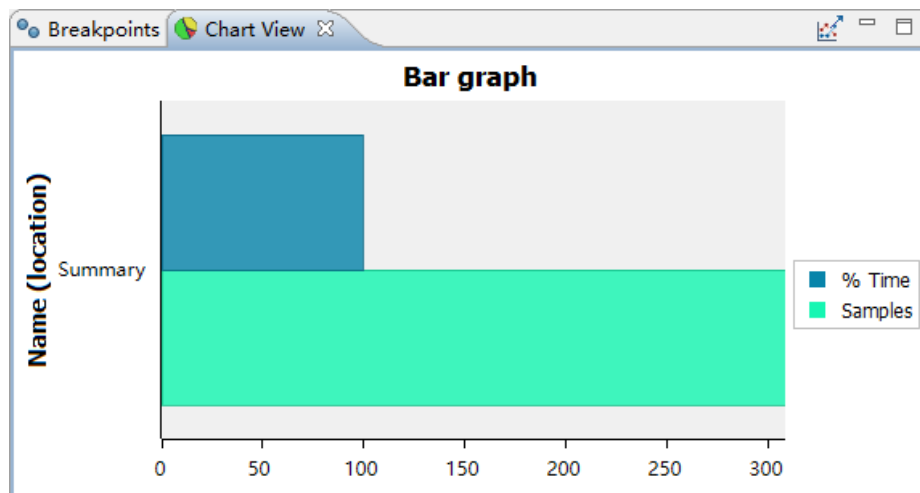


Figure 7-17 Chart View



- If there is assembly code in the program, click "i" (Instruction Stepping Mode) on the toolbar of debugging view to switch to the disassembly view and each assembly instruction is executed in a single step, as shown in Figure 7-18 and Figure 7-19.

Figure 7-18 Select Instruction Stepping Mode

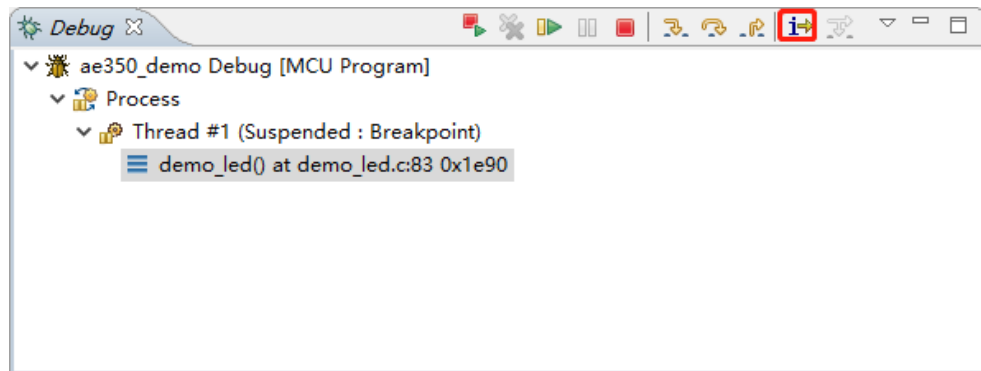
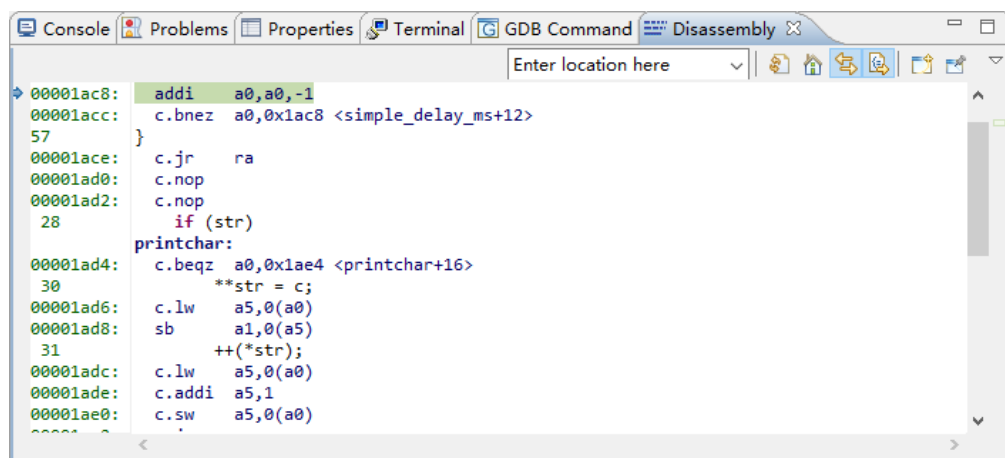


Figure 7-19 Disassembly View



8 ICEman Software

The ICEman software is located at "AndeSight_RDS_v511\ice\andes-iceman.bat" or can be accessed through the desktop shortcut "ICEman_AndeSight_RDS_v5.1.1 ", with options described as shown in Table 8-1.

Table 8-1 ICEman Options

Option	Description
-a, --reset-aice	Reset AICE as ICEman startup, for AICE only
-b, --bport	Socket port number for burner connection (default: 2354)
-c, --clock	Specify JTAG clock setting Usage: -c num num should be the following: 0: 30 MHz 1: 15 MHz 2: 7.5 MHz 3: 3.75 MHz 4: 1.875 MHz 5: 909.091 KHz 6: 461.538 KHz 7: 232.558 KHz 10: 10 MHz 11: 6 MHz 12: 3 MHz 13: 1.5 MHz 14: 750 KHz 15: 375 KHz Andes AICE-MINI+ only supports 11 ~ 15

Option	Description
-C, --check-times	Second to check DTM (default: 3 seconds) Example: 1. -C 100 to check 100 millisecond 2. -C 100s or -C 100S to check 100 seconds
-D, --larger-logfile	The maximum size of the log file is 1MBx2. The size is increased to 512MBx2 with this option.
-f, --log-output	output log file path
-h, --help	The usage is for ICEman
-H, --reset-hold	Reset-and-hold while ICEman startup
-I, --interface	Specify an interface config file in ice/interface
-I, --custom-srst	Use custom script to do SRST
-L, --custom-trst	Use custom script to do TRST
-N, --custom-restart	Use custom script to do RESET-HOLD
-o, --reset-time	Reset time of reset-and-hold (milliseconds) (default: 1000 milliseconds)
-p, --port	Socket port number for gdb connection
-s, --source	Show commit ID of this version
-S, --stop-seq	Specify the SOC device operation sequence while CPU stop
-R, --resume-seq	Specify the SOC device operation sequence before CPU resume Usage: --stop-seq A1:D1[:M1],A2:D2[:M2] --resume-seq A3:D3[:M3],A2:rst A*:address, D*:data, [:M*]:optional mask, rst:restore A*,D*,[M*] should be hex as following example Example: --stop-seq 0x500000:0x80,0x600000:0x20 --resume-seq 0x500000:0x80,0x600000:rst
-t, --tport	Socket port number for Telnet connection
-T, --boot-time	Boot time of target board (milliseconds) (default: 5000 milliseconds)
-v, --version	Version of ICEman
-x, --diagnosis	Diagnose connectivity issue Usage:

Option	Description
	--diagnosis[=address]
-X, --uncnd-reset-hold	Unconditional Reset-and-hold while ICEman startup (This implies -H)
-z, --ace-conf	Specify ACE file on each core Usage: --ace-conf <core#id>=<ace_conf>[,<core#id>=<ace_conf>]* Example: --ace-conf core0=core0.aceconf,core1=core1.aceconf
-Z, --target	Specify target type
--target-cfg	Specify the CPU configuration file for a complex multicore
--halt-on-reset	Enable/Disable halt-on-reset functionality

