AN0033
Application Note

Eclipse with GCC

Introduction

As a popular integrated development environment (IDE), Eclipse supports various computer languages including C/C++ through a wide range of plug-ins, adding to the flexibility of Eclipse platform, on which the user is able to conduct software IDE.

This Application Note describes how to use Eclipse plug-ins to debug AT32 series chips.

Applicable products:

Part number	AT32F series

Contents

List of Tables

List of Figures

1 Overview

This Application Note gives a detailed description of how to debug AT32 series chips using Eclipse, ARM-GCC compiler, GNU-ARM plug-ins, JLink and ATLink.

It mainly covers the following contents:

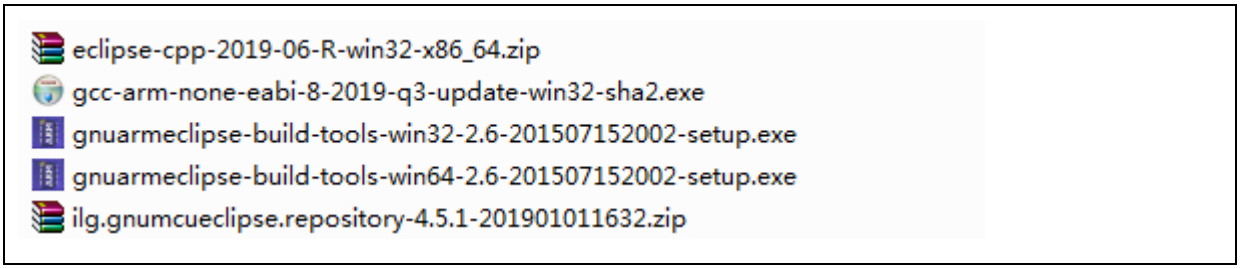
- Eclipse debug environment preparation
- Eclipse template project
- Eclipse compile configuration
- Eclipse debug configuration

Note: This installation manual is based on WINDOWS 7 x64 system, and projects in AT32Fxx_Firmware_Library\project\at_start_xx\templates\eclipse_gcc are used for illustration.

All the software kits in this document can be found in AT32_Eclipse_Packet.zip, which is unzipped to install and run.

AT32_Eclipse_Packet.zip contains the following files:

Figure 1. Files in AT32_Eclipse_Packet.zip



2 Eclipse debug environment preparation

First of all, the following software needs to be installed:

- Eclipse IDE for C/C++ Developers
- GNU ARM Eclipse plug-ins
- GCC ARM compiler
- GNU ARM Eclipse Build Tools (make, rm and others)

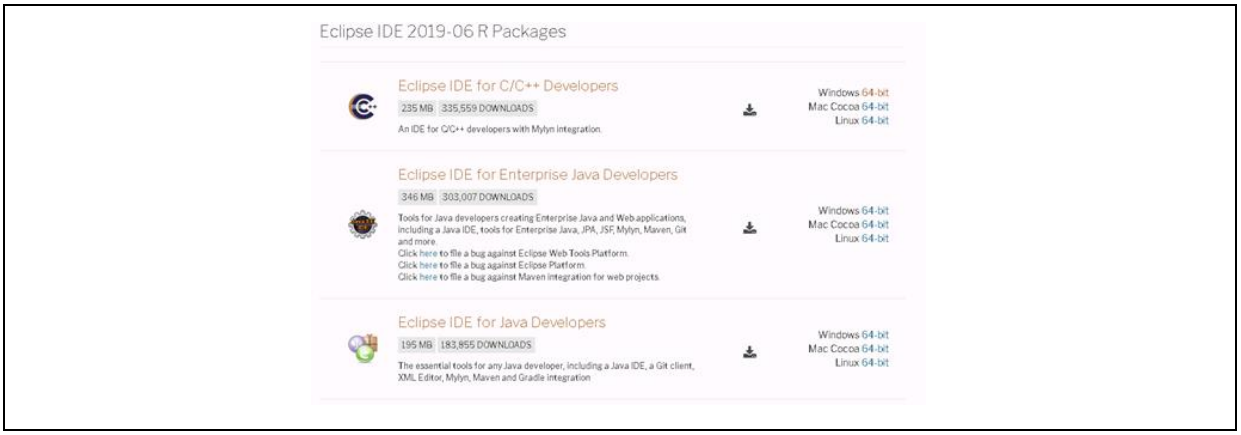
The subsequent sections introduce how to install these software.

2.1 Eclipse IDE for C/C++ Developers

Download the latest version of Eclipse IDE (for C/C++ developers). There is one version available in AT32_Eclipse_Packet.zip, that is, eclipse-cpp-2019-06-R-win32-x86_64.zip.

Download link: http://www.eclipse.org/downloads/eclipse-packages/

Figure 2 Eclipse download page



Download and unzip the eclipse-cpp-2019-06-R-win32-x86_64.zip. Click “eclipse.exe” to run Eclipse. Note that users need to install plug-ins before debugging code.

2.2 GNU ARM Eclipse plug-ins installation

Download and unzip the latest version of GNU ARM Eclipse plug-ins: ilg.gnumcueclipse.repository-4.5.1-201901011632.zip.

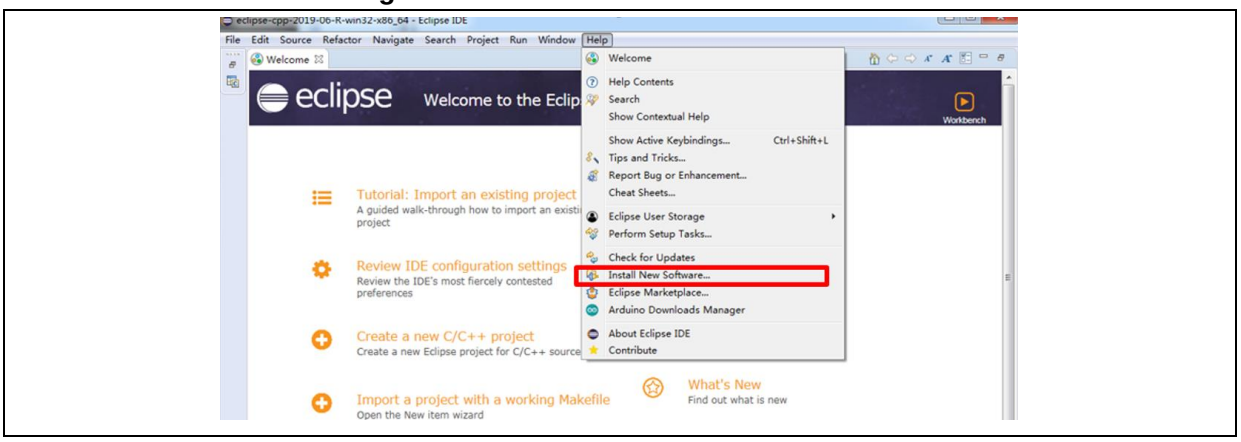
AT32_Eclipse_Packet.zip contains an available version of ilg.gnumcueclipse.repository-4.5.1-201901011632.zip.

Download link: https://github.com/gnu-mcu-eclipse/eclipse-plugins/releases

Installation steps:

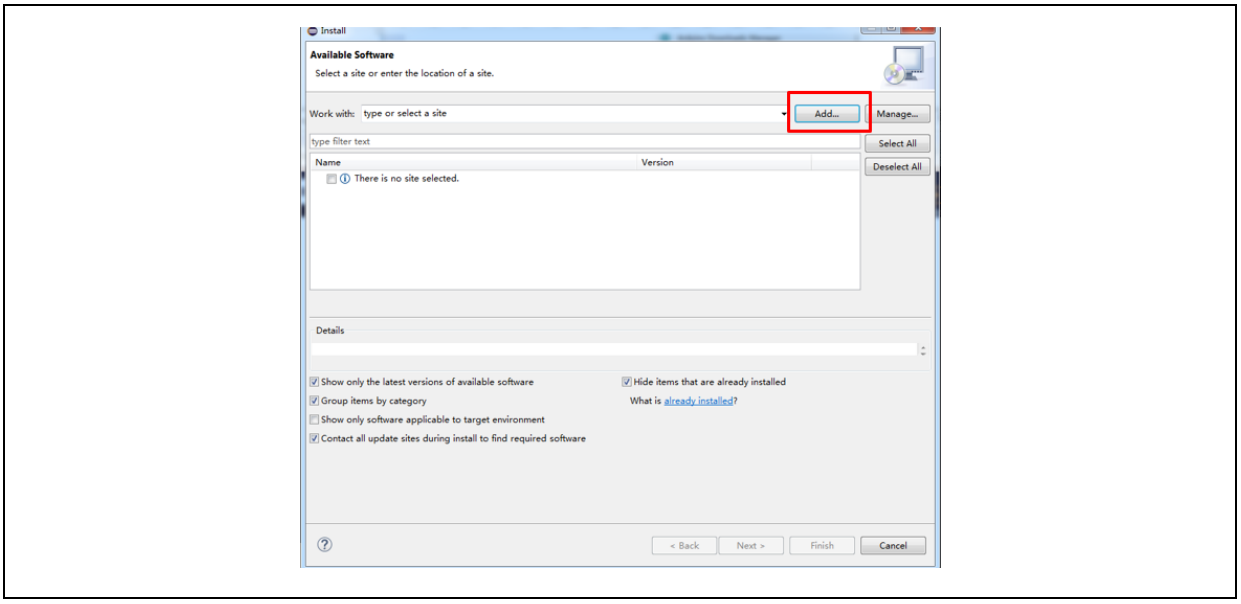
1. Go to Eclipse Help->Install New Software.

Figure 2. Click on Install New Software



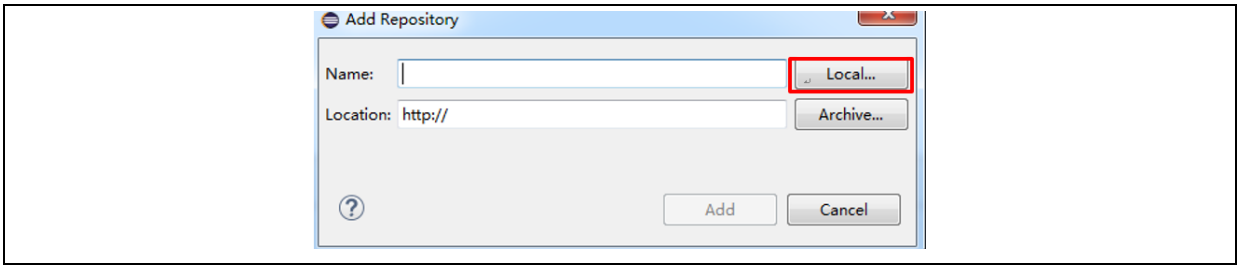
2. Click on “Add...”

Figure 3. Click on Add



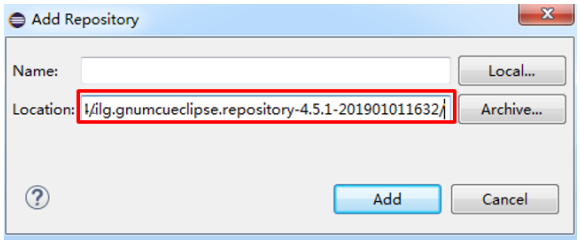
3. Add a local plug-in, or automatically download and install through an Internet path.

Figure 4. Add Repository



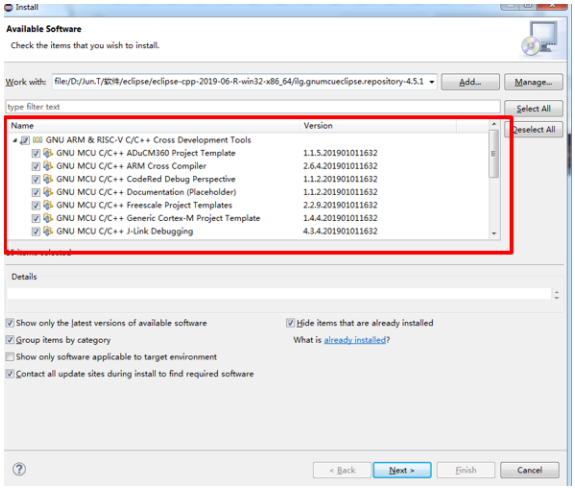
4. Select a local plug-in directory, and click on “Add”.

Figure 5. Select plug-in directory



5. Tick all plug-ins, and click on “Next”.

Figure 6. Tick plug-ins



6. Installation is completed, and click on “Next”.

Figure 7. Installation completed

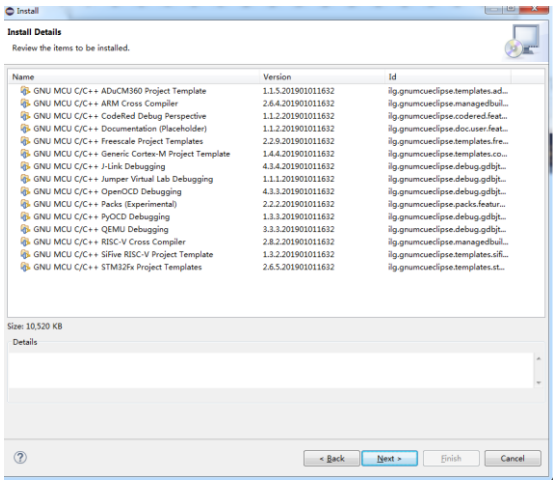
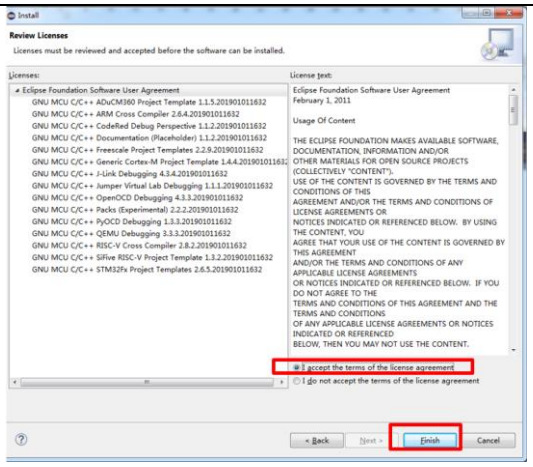
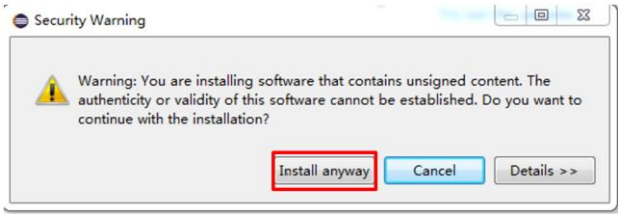


Figure 8. Accept the license agreement



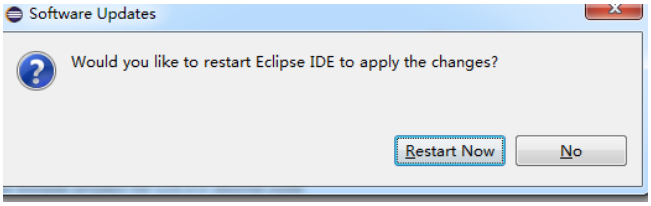
7. Go to “Install anyway”.

Figure 9. Install anyway



8. Restart Eclipse.

Figure 10. Restart Eclipse



2.3 ARM GCC compiler tool chains installation

Download the latest compiler tool chain: gcc-arm-none-eabi-8-2019-q3-update-win32-sha2.exe.

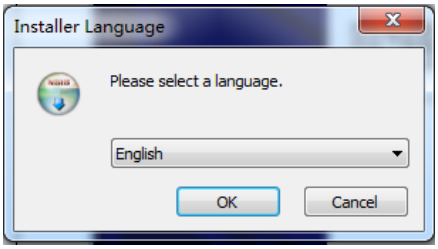
AT32_Eclipse_Packet.zip contains an available version of gcc-arm-none-eabi-8-2019-q3-update-win32-sha2.exe.

Download link: https://launchpad.net/gcc-arm-embedded/+download

Installation steps:

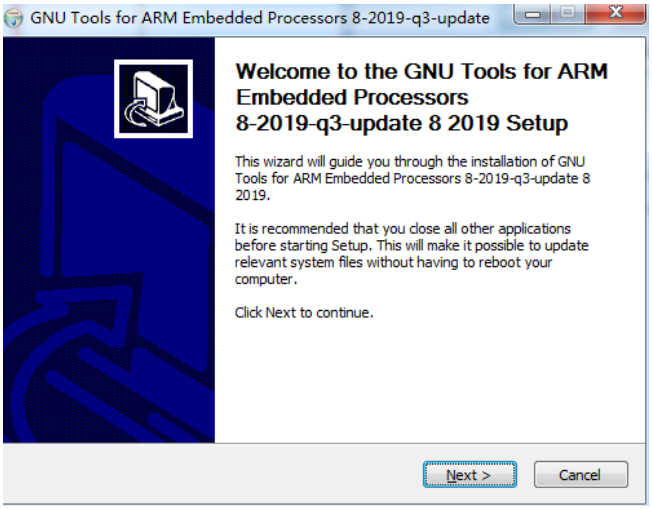
1. Select a language.

Figure 11. Installer language



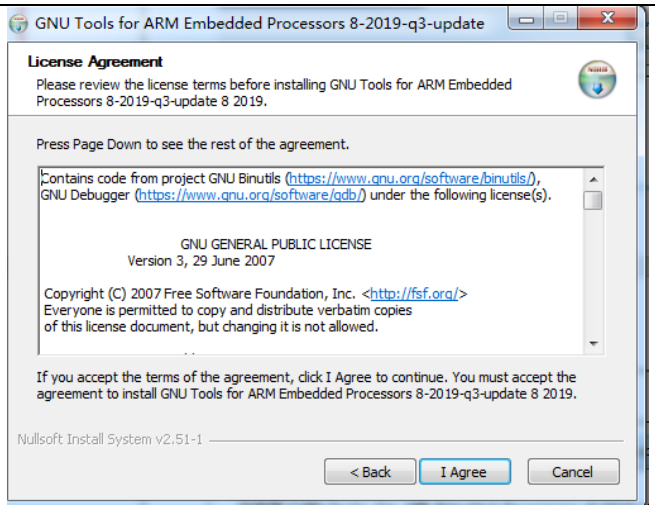
2. Go to Setup wizard, and click on “Next”.

Figure 12. Setup wizard



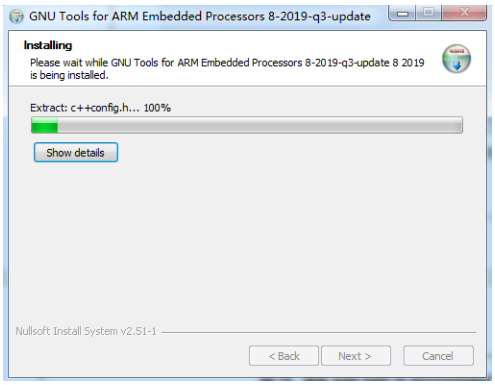
3. Click on “Accept” for the license agreement.

Figure 13. Accept license agreement



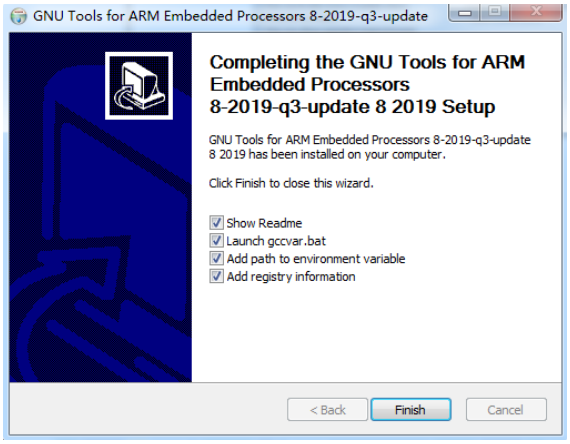
4. Select the default installation location, and click on “Install”.

Figure 14. Installation progress



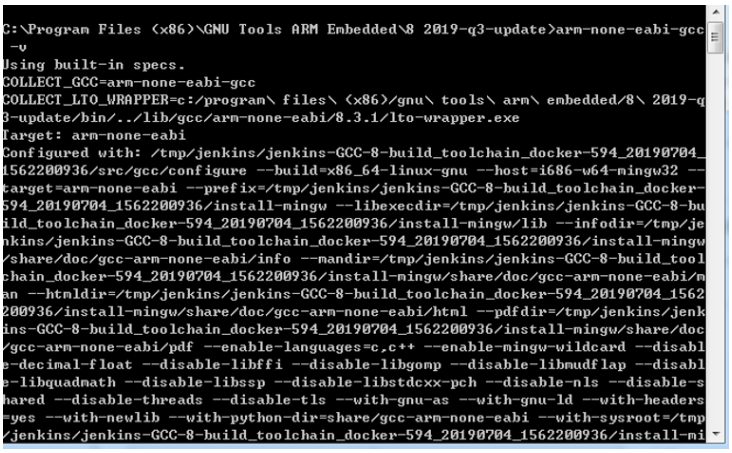
5. In the pop-up installation window, tick “Add path to environment variable” for auto add, or else, you need to do so by manual operation.

Figure 15. Tick “Add path to environment variable”



6. After the installation is completed, enter “arm-none-eabi-gcc –v” in the pop-up command window, and some information including version code will be displayed, indicating that it is a successful installation.

Figure 16. Installation result displayed



2.4 GNU ARM Eclipse Build Tools installation

This section provides information about the setup of such commands as “make” and “rm”.

Download link: https://sourceforge.net/projects/gnuarmeclipse/files/Build Tools/

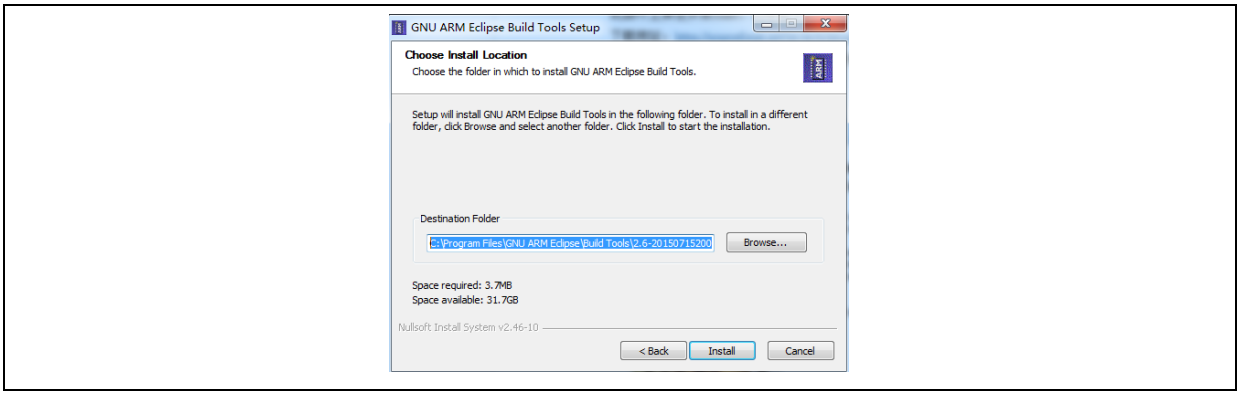
AT32_Eclipse_Packet.zip contains an available version of gnuarmeclipse-build-tools-win64-2.6-201507152002-setup.exe, or you may download other applicable versions.

1. Run the installation package.

Figure 17. Run installation package

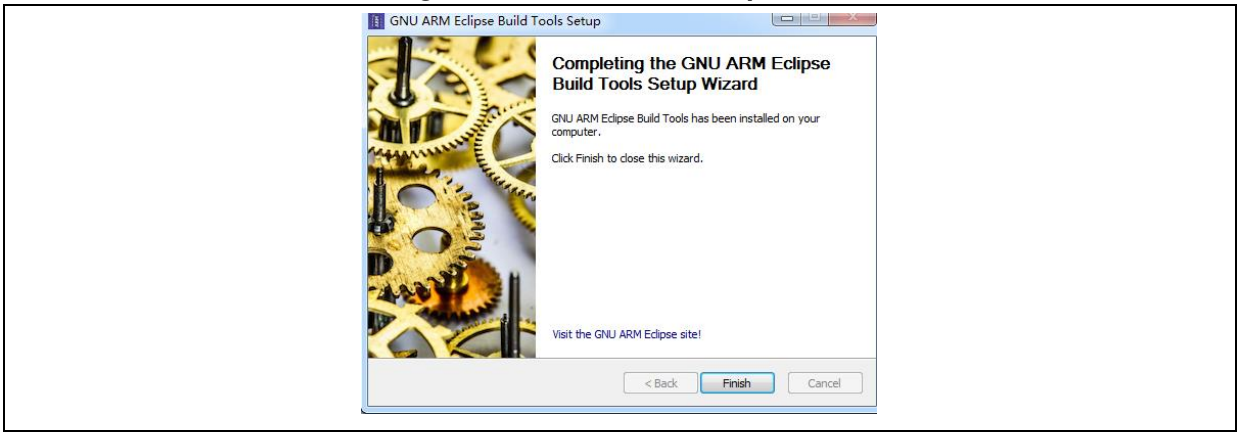
2. Select destination folder.

Figure 18. Select destination folder



3. Restart Eclipse after finishing installation.

Figure 19. Installation completed



2.5 Install JLink

It is necessary to copy AT32 series chips to JLink directory through ICP.

1. JLink installation (omitted)

   Download and install the latest version of JLink.

2. Copy algorithm file

To recognize and download program to AT32 series chips through JLink, the AT32 algorithm file should be downloaded to JLink directory through ICP tool (run ICP directly, and the corresponding AT32 algorithm will be copied to JLink directory).

3 Template project configuration and compiling

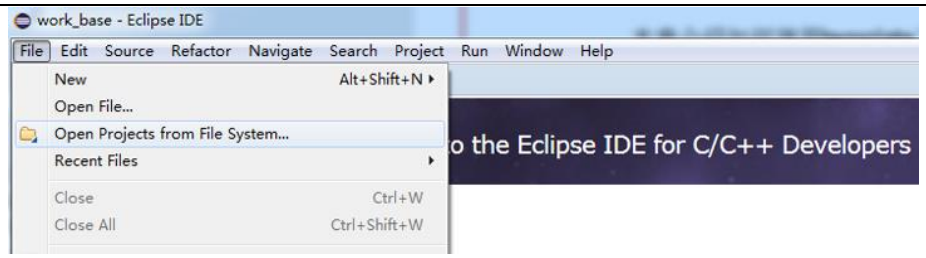
This section demonstrates how to use template projects.

Project path: AT32Fxx_Firmware_Library\project\at_start_xx\templates\eclipse_gcc.

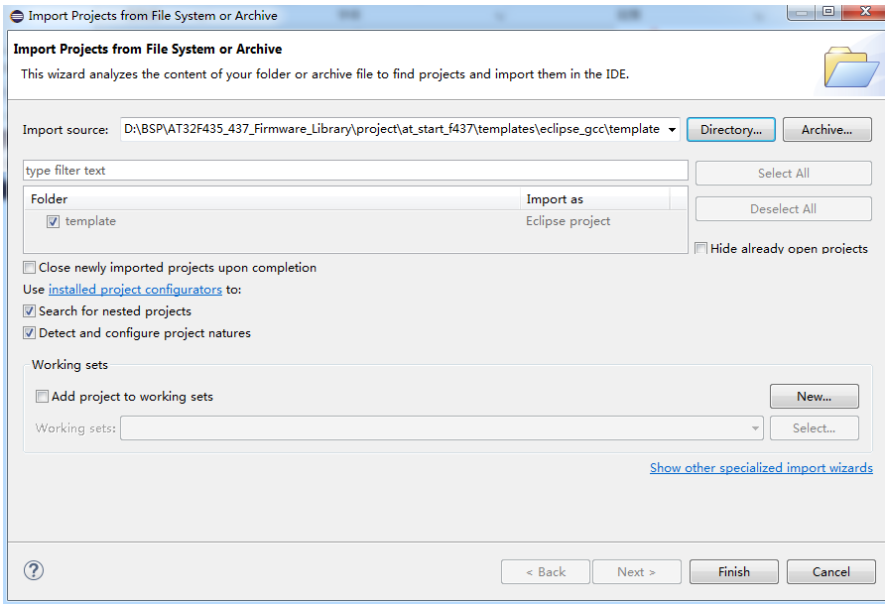
This section takes AT32F437 as an example to illustrate the project configuration and compilation.

3.1 Open template project

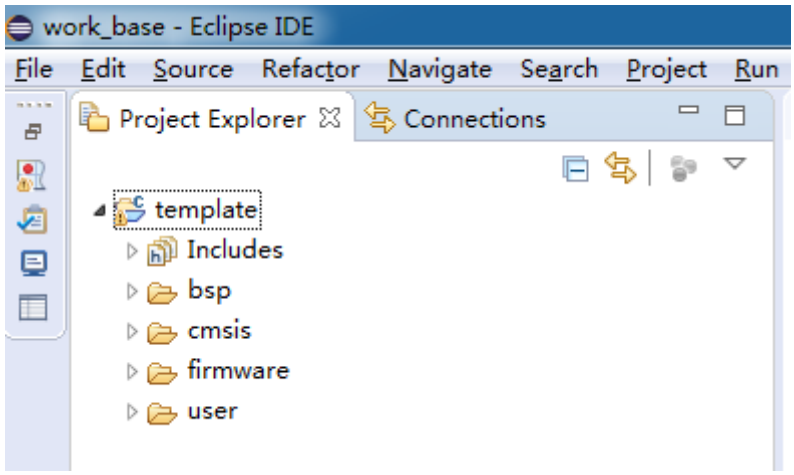
1. Click on File→Open Projects from File System.



2. Select a path in “Import source”, and click on “Finish”. AT32F437xx template path: xxx\AT32F435_437_Firmware_Library\project\at_start_f437\templates\eclipse_gcc\template (similar paths for other series).



3. Open the project, and you will see the following template.

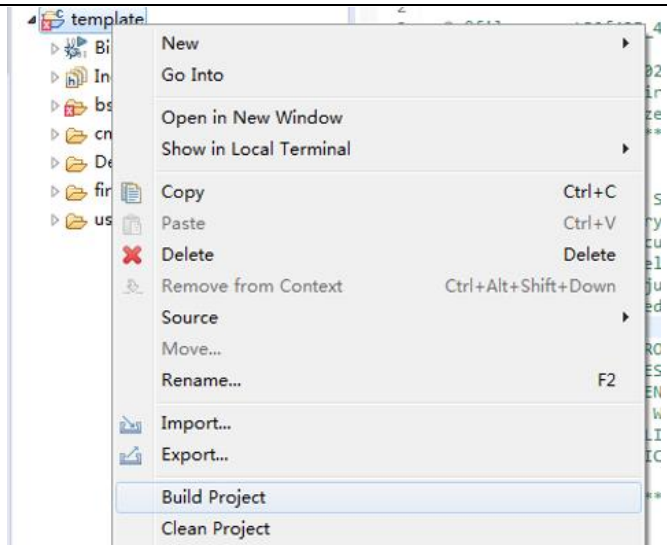


3.2 Compile

During the compiling process, the following configurations should be completed.

- Chip configuration
- Header file path configuration
- Macro configuration
- Script file configuration (ld files of different series)

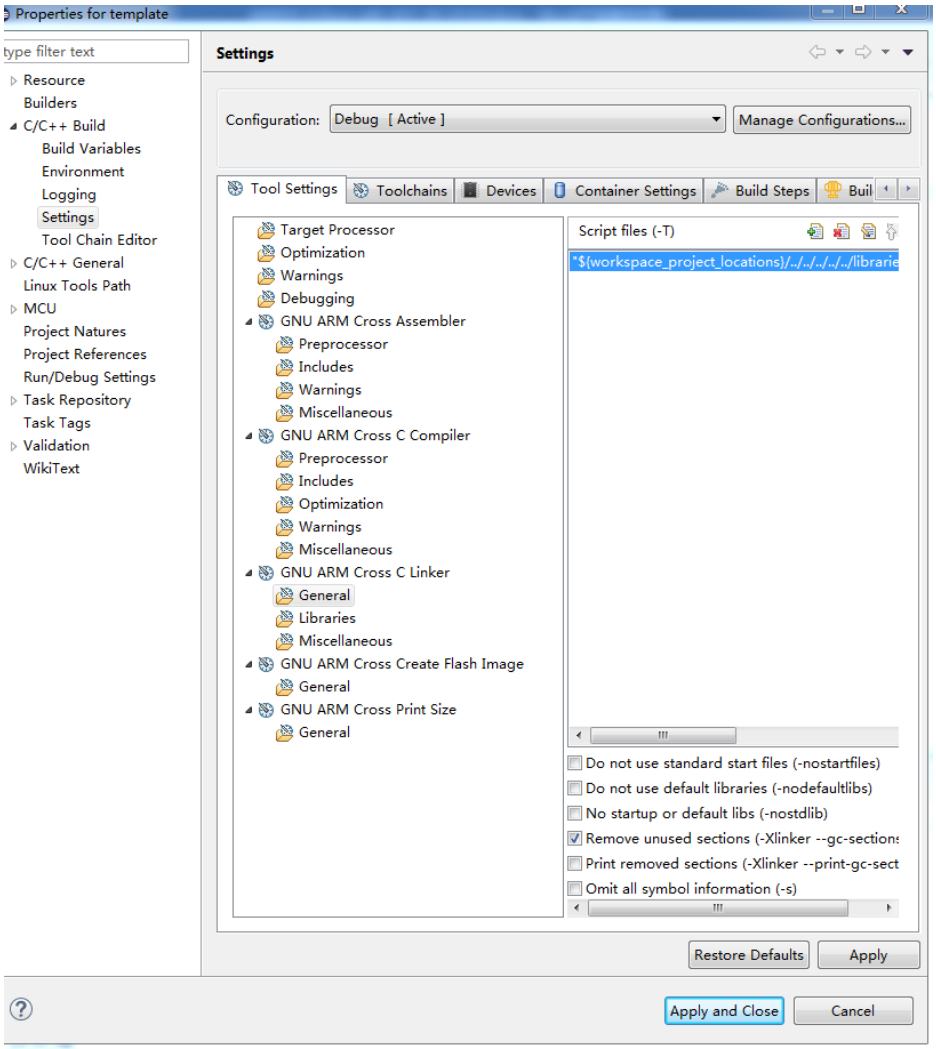
1. Right click on “Template” and select “Build Project”.

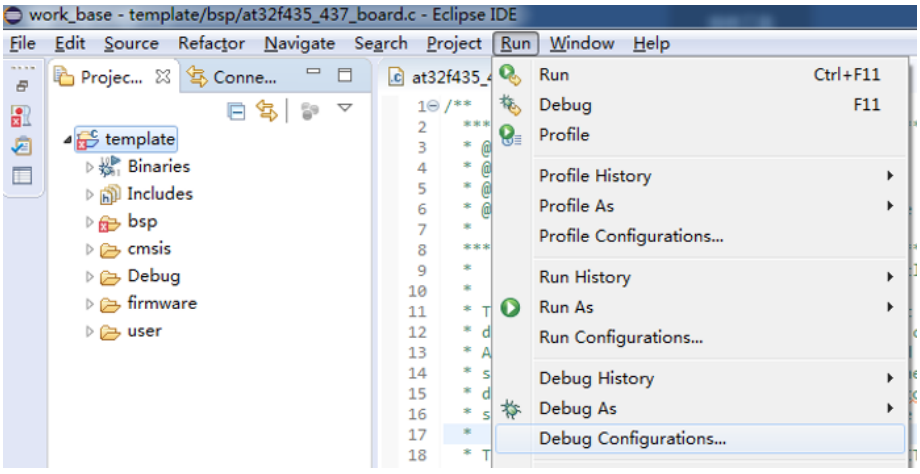


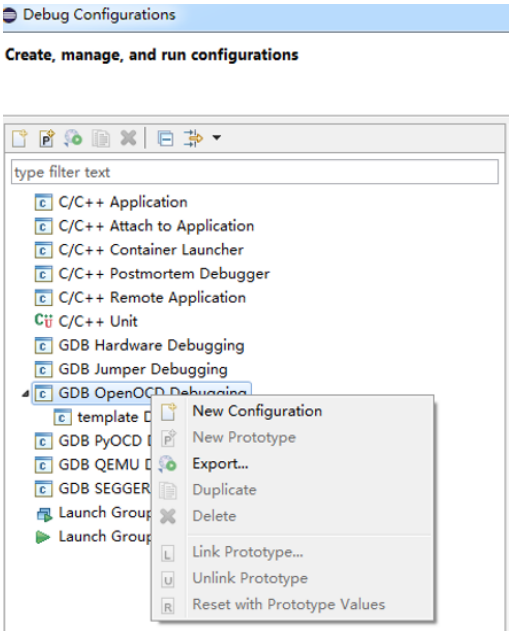
2. After completion of compiling, the “template.elf” is generated.

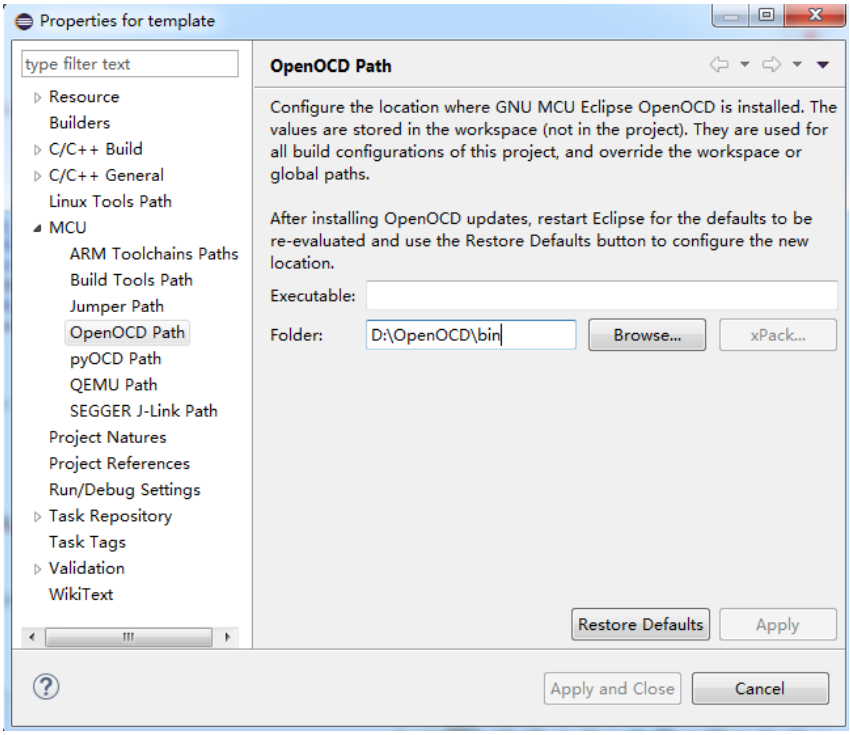
```
CDT Build Console [template]
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -ffunction-sections  -g -DAT_START_F437_V1 -DAT32F437ZMT7 -DUSE_STDPERIPH_DRIV
Finished building: D:/BSP/AT32F435_437_Firmware_Library/libraries/cmsis/cm4/device_support/system_at32f435_437.c

Building file: D:/BSP/AT32F435_437_Firmware_Library/project/at32f435_437_board/at32f435_437_board.c
Invoking: GNU ARM Cross C Compiler
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -ffunction-sections  -g -DAT_START_F437_V1 -DAT32F437ZMT7 -DUSE_STDPERIPH_DRIV
Finished building: D:/BSP/AT32F435_437_Firmware_Library/project/at32f435_437_board/at32f435_437_board.c

Building target: template.elf
Invoking: GNU ARM Cross C Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -ffunction-sections  -g -T "D:\BSP\AT32F435_437_Firmware_Library\project\at_st
Finished building target: template.elf


15:09:24 Build Finished. 0 errors, 0 warnings. (took 11s.303ms)
```

3. For configurations of different models (of the same series), you only need to modify ld files in “Settings” as shown below. You can also modify other settings such as header file path in “Settings” if necessary.






Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Container Launcher
C/C++ Postmortem Debugger
C/C++ Remote Application
C/C++ Unit
GDB Hardware Debugging
GDB Jumper Debugging
GDB OpenOCD Debugging
template D
GDB PyOCD D
GDB QEMU D
GDB SEGGER
Launch Group
Launch Group
New Configuration
New Prototype
Export...
Duplicate
Delete
Link Prototype...
Unlink Prototype
Reset with Prototype Values


Properties for template
type filter text
Resource
Builders
C/C++ Build
C/C++ General
Linux Tools Path
MCU
ARM Toolchains Paths
Build Tools Path
Jumper Path
OpenOCD Path
pyOCD Path
QEMU Path
SEGGER J-Link Path
Project Natures
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText
OpenOCD Path
Configure the location where GNU MCU Eclipse OpenOCD is installed. The values are stored in the workspace (not in the project). They are used for all build configurations of this project, and override the workspace or global paths.
After installing OpenOCD updates, restart Eclipse for the defaults to be re-evaluated and use the Restore Defaults button to configure the new location.
Executable:
Folder:
D:\OpenOCD\bin
Browse...
xPack...
Restore Defaults
Apply
Apply and Close
Cancel

4 Debug

This chapter contains JLink debug and ATLink debug of AT32 series chips.

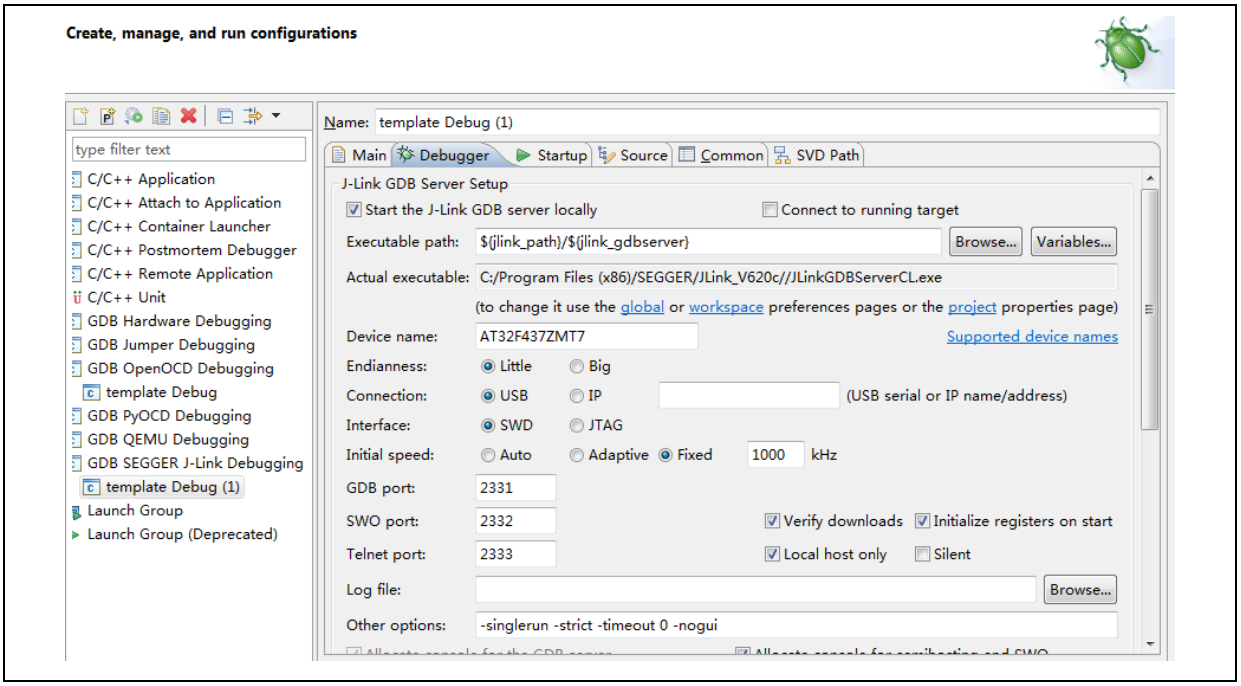
4.1 JLink debug

The following configurations are needed for JLink debug.

- JLink configuration
- GDB configuration
- SVD peripheral register configuration

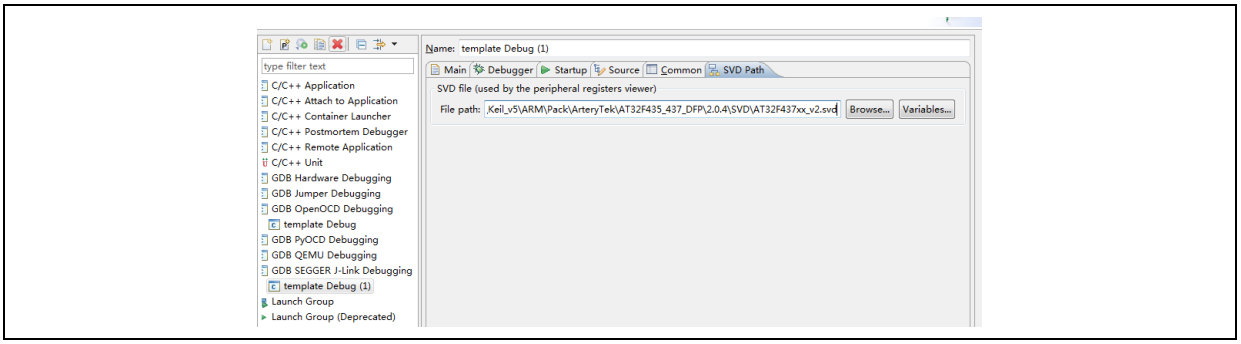
4.1.1 Debug configuration

1. Go to “Run” → “Debug Configurations” → “GDB SEGGER J-Link Debugging”→ “New Configuration”. Create a new Debug configuration, configure JlinkGDBServerCL, and fill in device name, such as AT32F437ZMT7, AT32F413RCT7 or AT32F415RCT7.

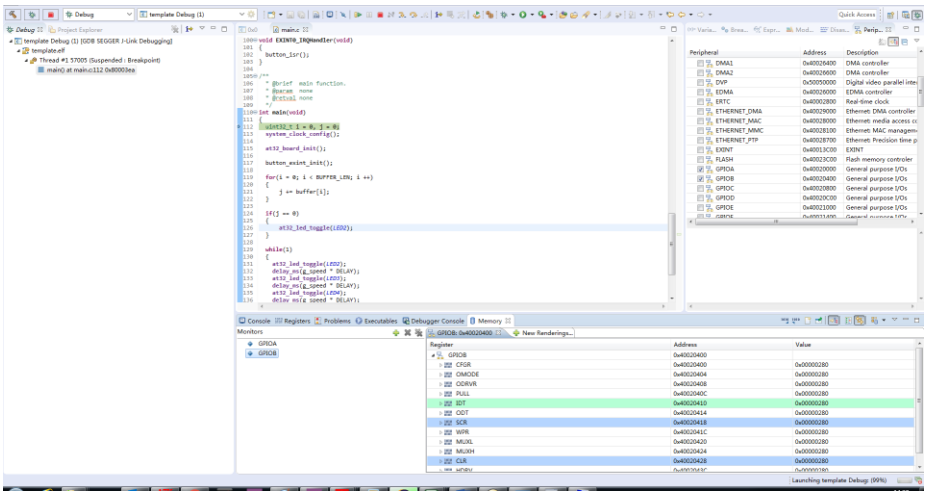


2. To set up GDB, select arm-none-eabi-gdb.exe under GCC directory.

3. Select SVD Path for debug register description. You can use the svd file in keil. When AT32 keil Packet is installed, the svd file can be automatically copied to keil directory.



4. Debug configuration is done, then go to “Apply”→”Debug”.



4.2 ATLink debug

This section demonstrates how to debug AT32 using OpenOCD + Eclipse + ATLink. For details on ATLink, please refer to AT-Link_User_Manual_SC.pdf.

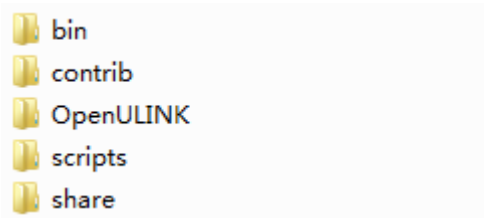
This section contains the following contents:

- Eclipse Openocd configuration
- GDB configuration
- SVD peripheral register configuration

Unzip the OpenOCD package file <<OpenOCD_V2.x.x.zip>>.

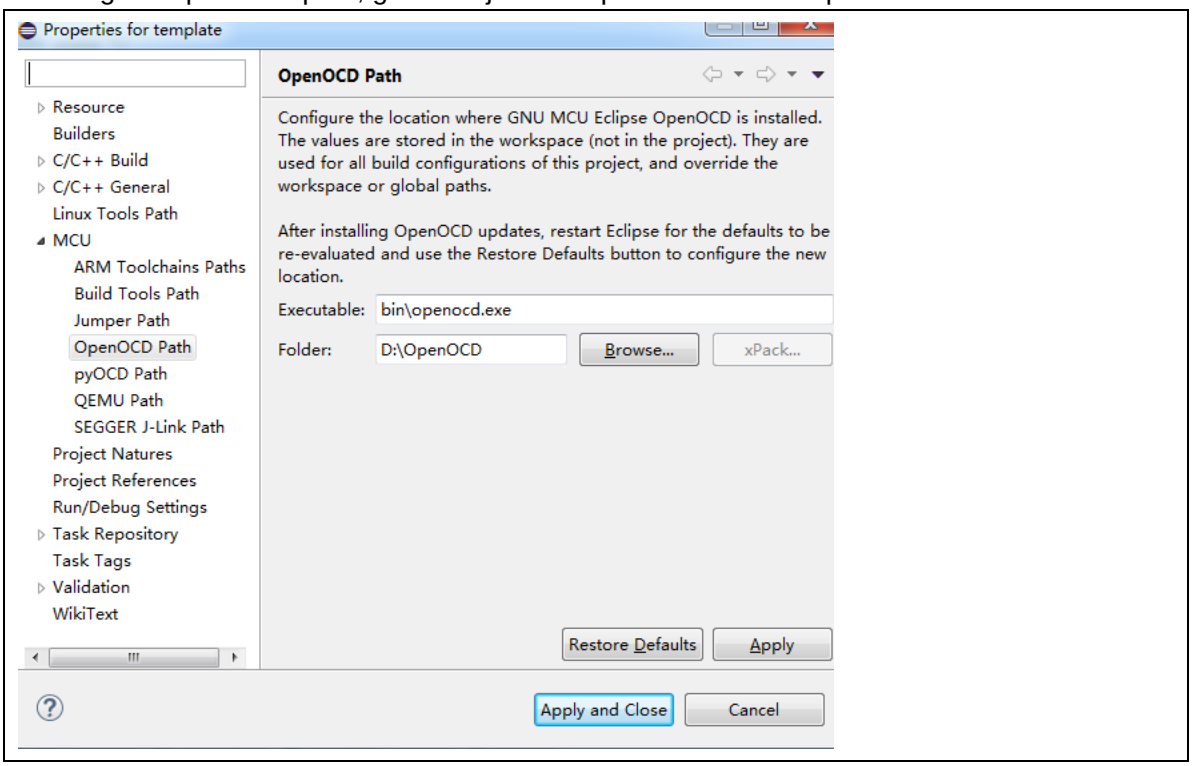
It includes five directories. The bin file is executable, and the scripts is for configuration files.

OpenOCD includes the following directories.



4.2.1 Debug configuration

1. To configure OpenOCD path, go to Project→Properties→MCU→OpenOCD Path.



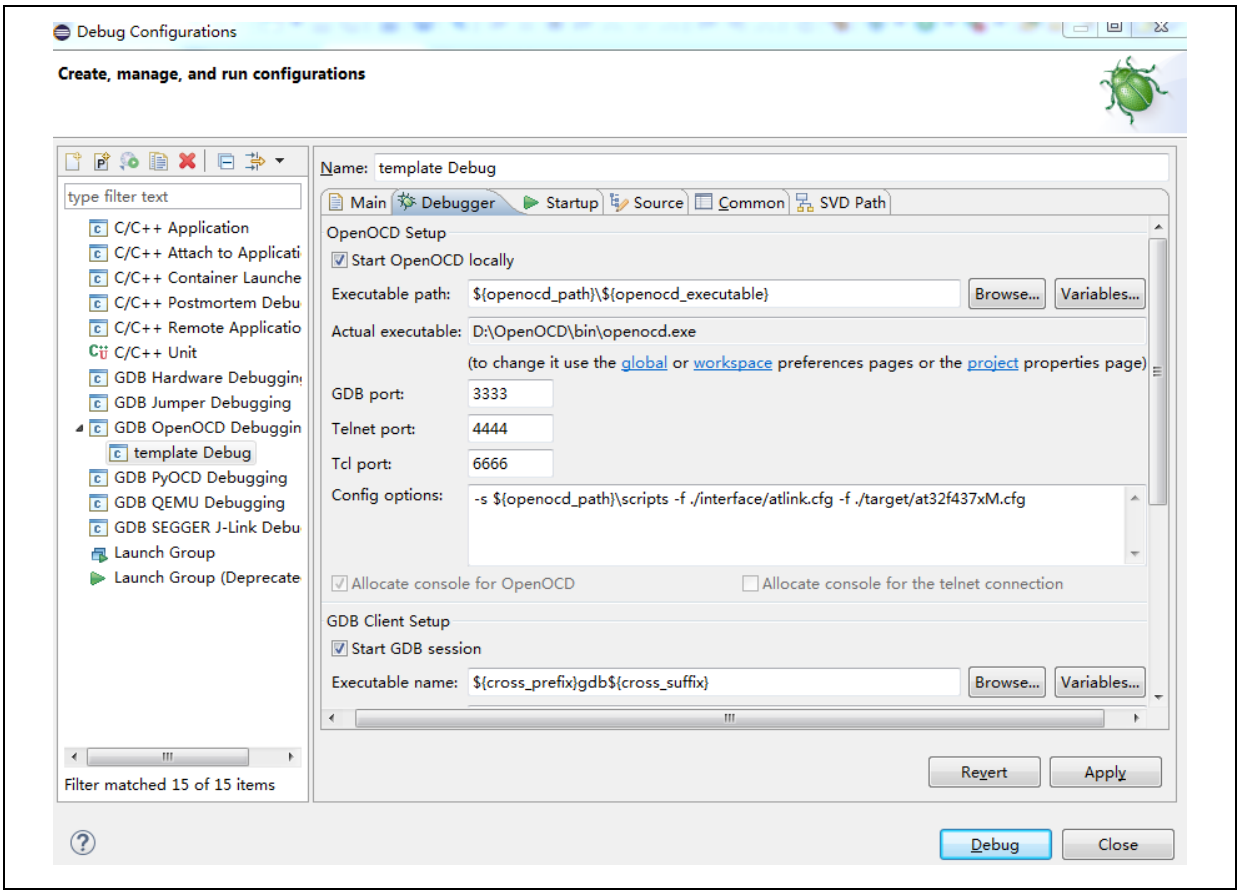
2. To configure Debug, go to “Run” → ” Debug Configurations” →”GDB OpenOCD Debugging”→“New Configuration”.

   The configuration items are as follows:

   Openocd executable path: D:\OpenOCD\bin\openocd.exe.

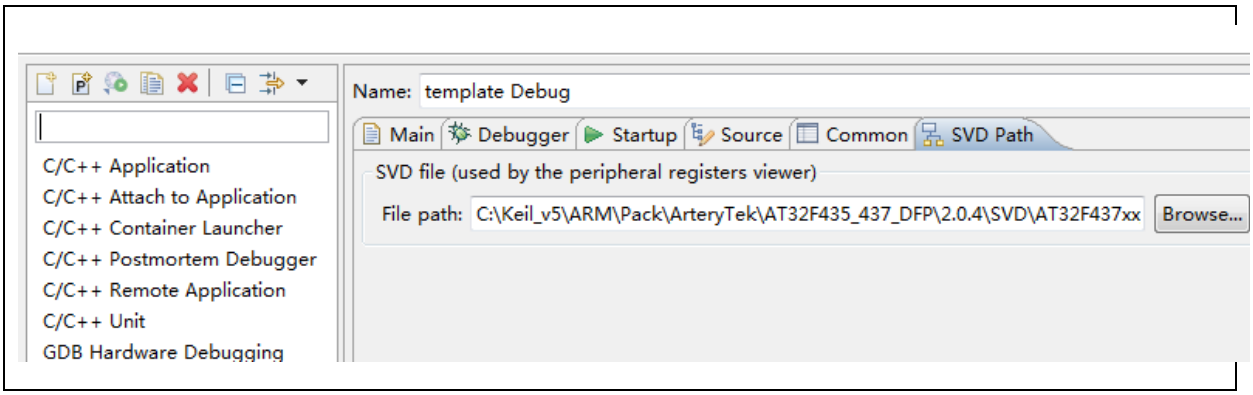
   Config options: -s ${openocd_path}\scripts -f ./interface/atlink.cfg -f ./target/at32f437xM.cfg.

   The “atlink.cfg” refers to ATLink debug tool, and the “at32f437xM.cfg” means that AT32F437 has a 4032 KB FLASH (for other AT32F437 Flash sizes, use at32f437xx.cfg). For other series such as AT32F403A and AT32F415, the “target/xxx.cfg” should be modified accordingly.

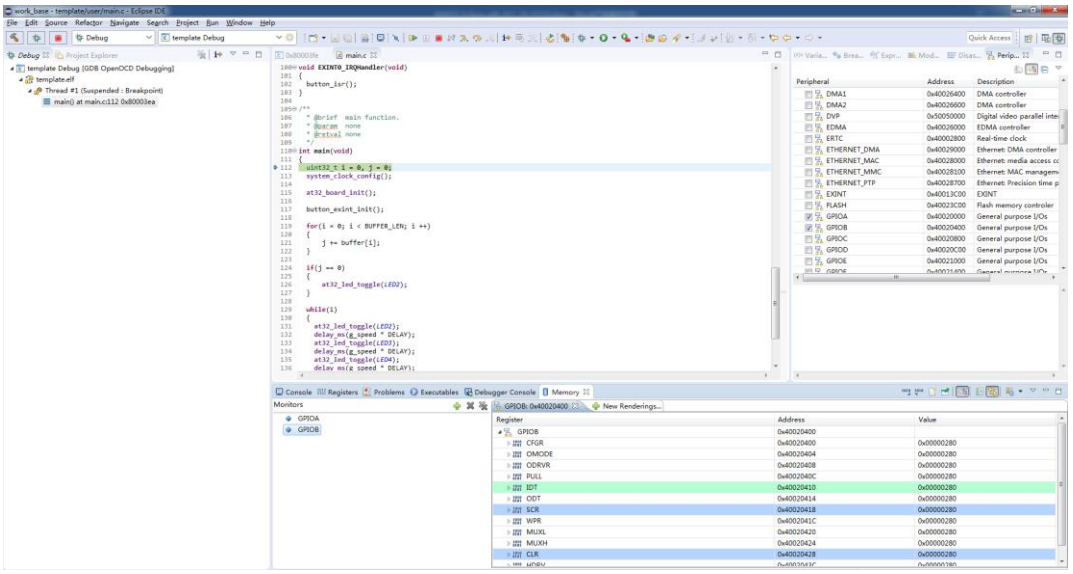


3. Configure SVD files

Download corresponding SVD files for Debug.



4. After completing Debug configuration, go to “Apply”→”Debug” to start debugging.



5 Revision history

Table 1. Document revision history

Date	Version	Revision note
2021.12.13	2.0.0	Initial release