Application Note – Eclipse Environment Setup

1. Version

v0.1	Initial version
v0.2	More details added to Project Import & Compilation
v0.3	Typos corrections.
v1.0	Release with AmbiqSuite SDK

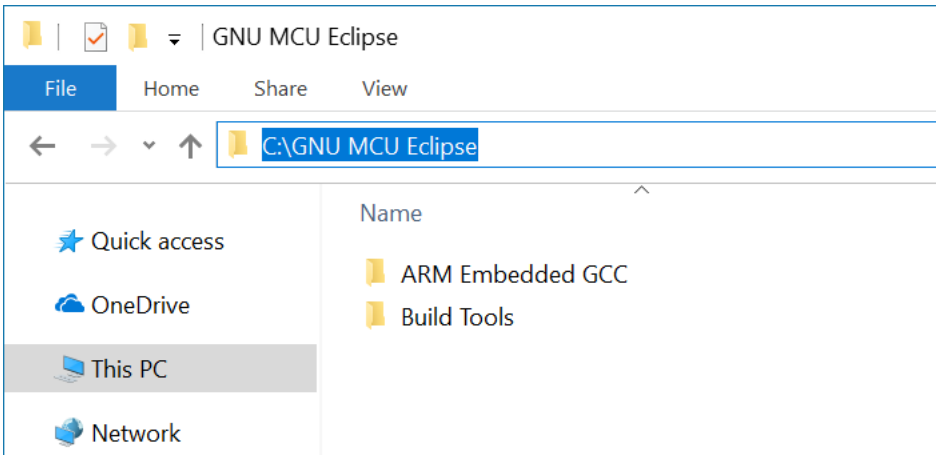
2. Scope

The open source Eclipse IDE is not formally supported in AmbiqSuite SDK and only limited tests are performed. AmbiqSuite does support GCC makefiles and all examples compiled and tested in the environment. This application note provides a step-by-step procedure to setup Eclipse development and debugging environment on machines running Windows operating systems. In this document, Windows 10 64-bit is used an example. It outlines the open source tools that need to be downloaded, but the user should keep in mind that these tools change quite rapidly and some research may be required to get the latest versions.

3. Installation

1. Java Runtime Environment (JRE) or Java Development Kit (JDK)
   - JRE is sufficient for our usage.
   - https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
   - Make sure JRE is in Windows environment variable *Path*.
2. GNU MCU Eclipse ARM Embedded GCC
   - Download from https://github.com/gnu-mcu-eclipse/arm-none-eabi-gcc/releases/
   - Extract and place it to a proper location and add this path to Windows environment variable *Path*.
3. GNU MCU Eclipse Windows Build Tools
   - Download from https://github.com/gnu-mcu-eclipse/windows-build-tools/releases
   - Extract and place it to a proper location.

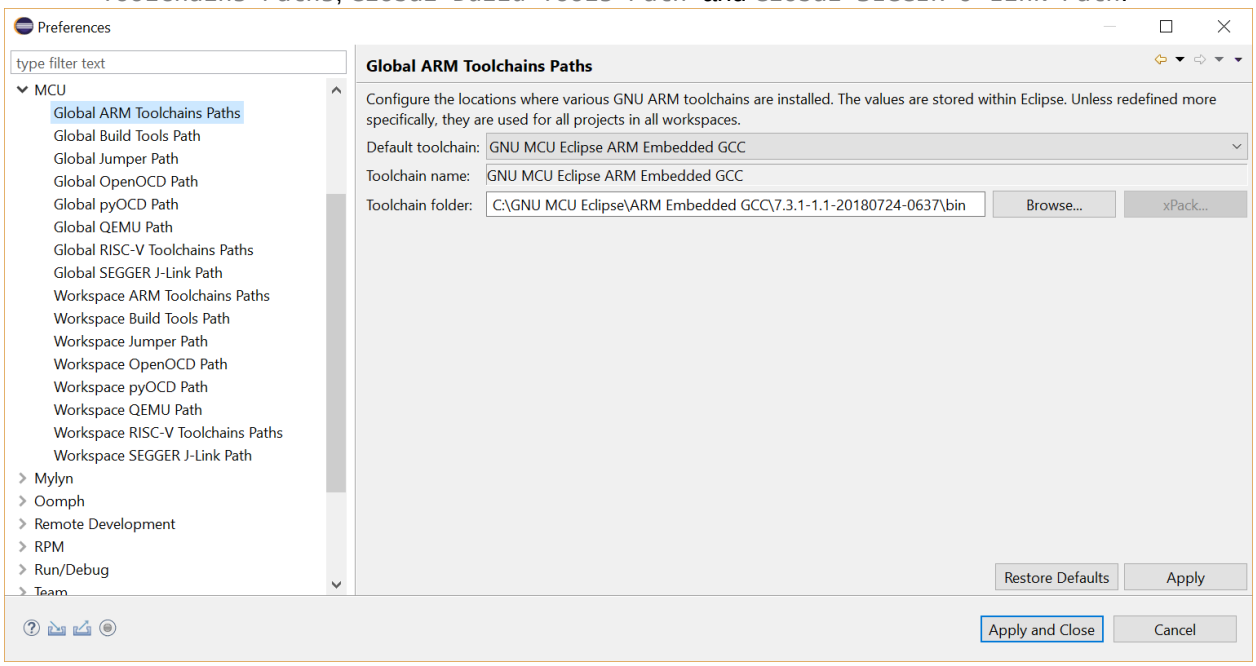
** For example:

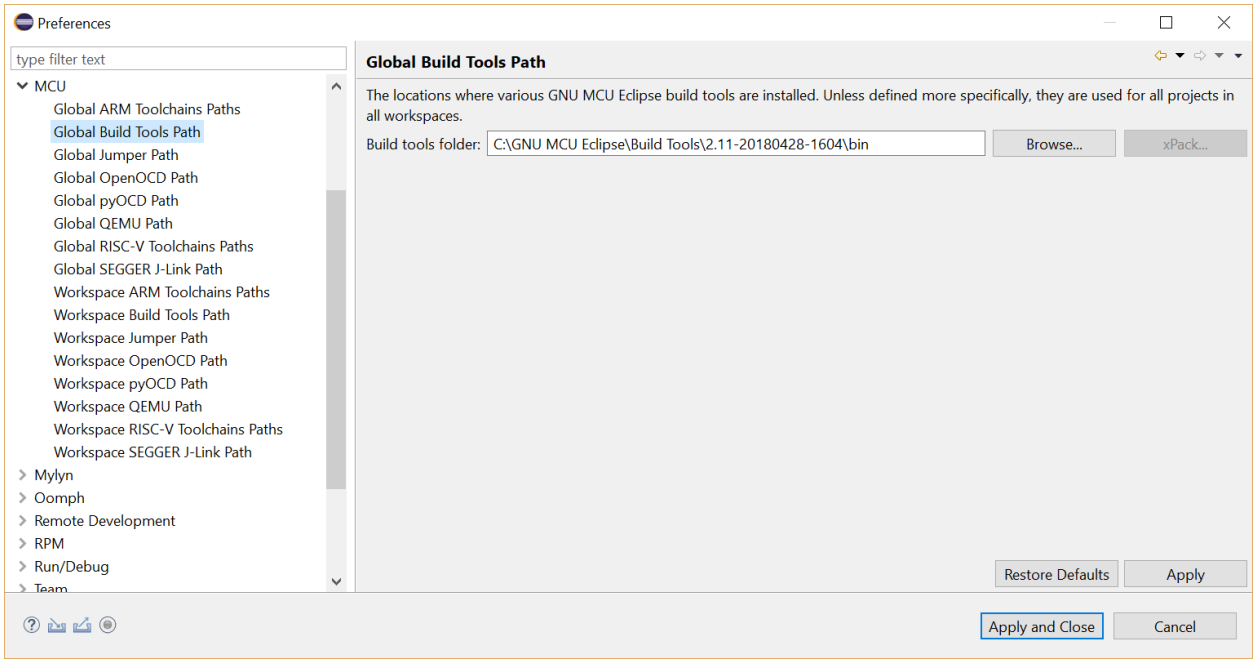


4. GNU MCU Eclipse IDE for C/C++ Developers
   - Download from https://github.com/gnu-mcu-eclipse/org.eclipse.epp.packages/releases. Make sure the version matches the installed JRE, both 32-bit version or 64-bit.
   - Extract and place it to a proper location.
5. J-Link Software and Documentation pack for Windows
   - Download from https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack and install it.

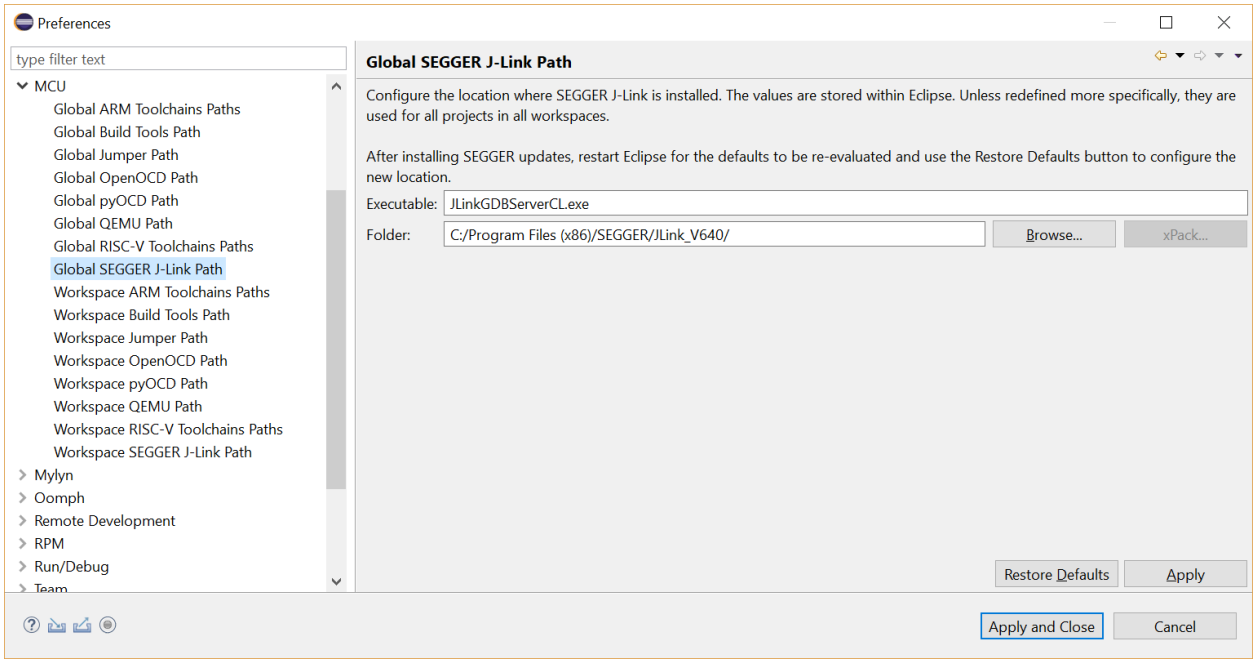
4. Eclipse Setup

1. Launch Eclipse and you will be asked to setup a workspace which can be anywhere.
2. In Eclipse, navigate to `Window`>`Preferences`. In the left panel, unfold MCU and configure `Global ARM Toolchains Paths`, `Global Build Tools Path` and `Global SEGGER J-Link Path`.



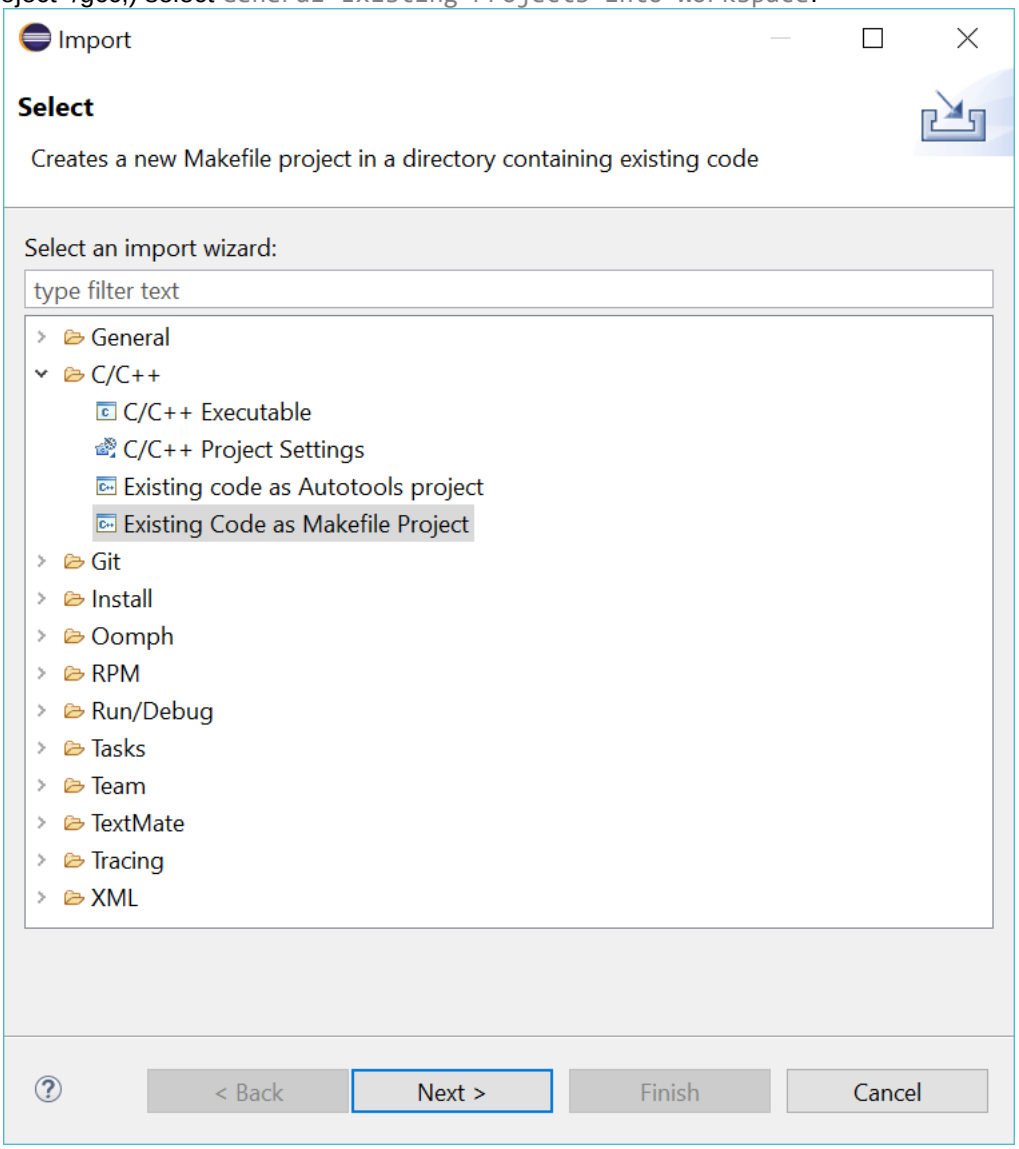


Application Note – Eclipse Environment Setup

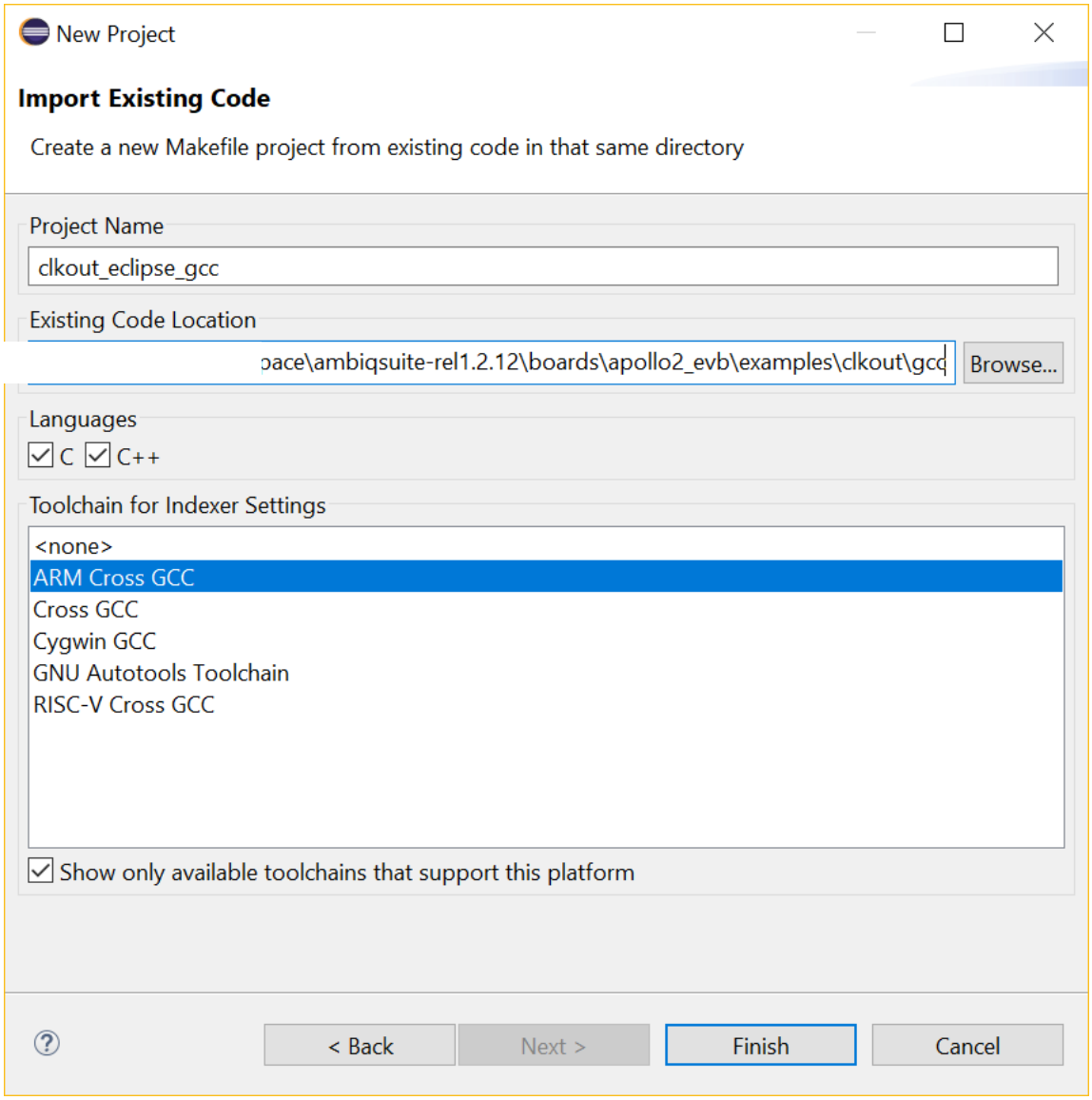


5. Project Import & Compilation

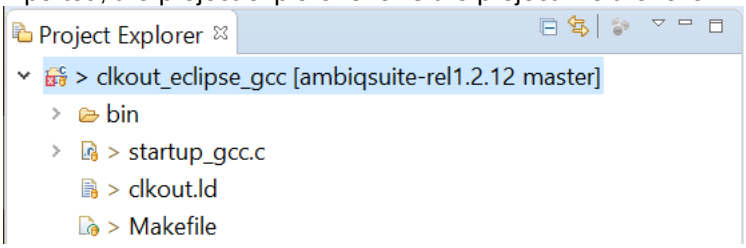
1. In Eclipse, navigate to `File`>`Import`. Select `C/C++`>`Existing Code as Makefile Project`. Click `Next`. For projects which are to be imported for the first time use this option. For those projects which have previously been imported to Eclipse (check if the files, .project and .cproject, and the folder .settings exist in <project>/gcc,) select `General`>`Existing Projects into Workspace`.



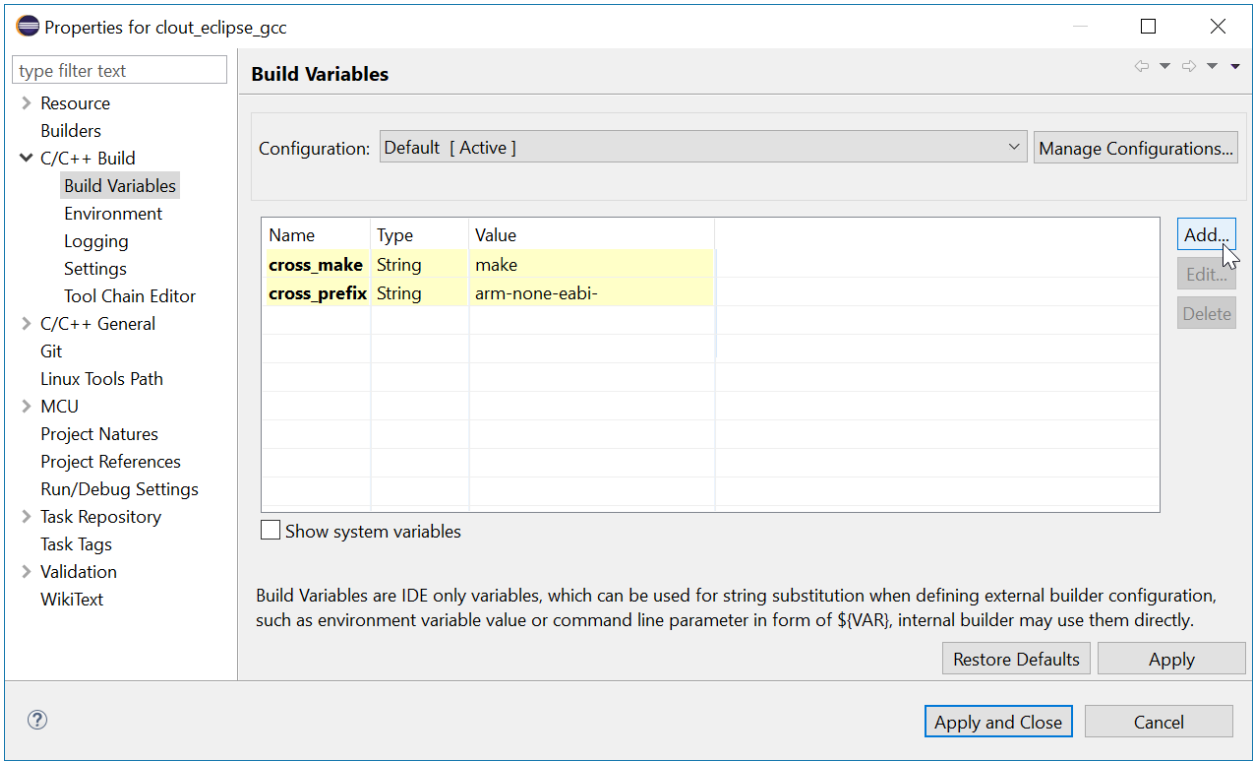
2. Select the targeting project. Take the project clkout of R2.0 for Apollo3 as an example.
   - [Optional] Change the project name.
   - Select `ARM Cross GCC` in `Toolchain for Indexer Settings`.



3. After the project is imported, the project explorer shows the project like the following screenshot.



4. Right click on the project and go to `Properties`>`C/C++ Builds`>`Build Variables`. Add two variables as highlighted in below. Click `Apply and Close`.



5. Right click on the project and select `Clean Project`.

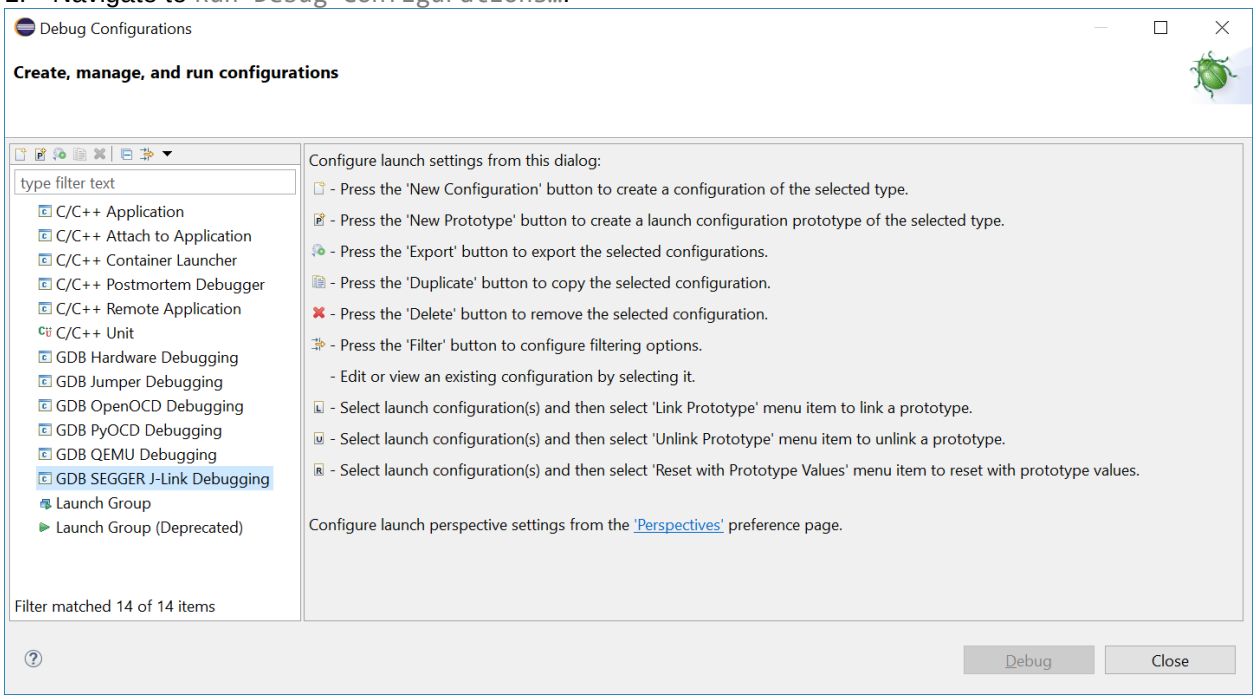
```
Problems  Tasks  Console  Properties
CDT Build Console [clout_eclipse_gcc]
17:14:13 **** Clean-only build of configuration Default for project clout_eclipse_gcc ****
make clean
Cleaning...

17:14:13 Build Finished. 0 errors, 0 warnings. (took 317ms)
```

6. Right click on the project and select `Build Project`.

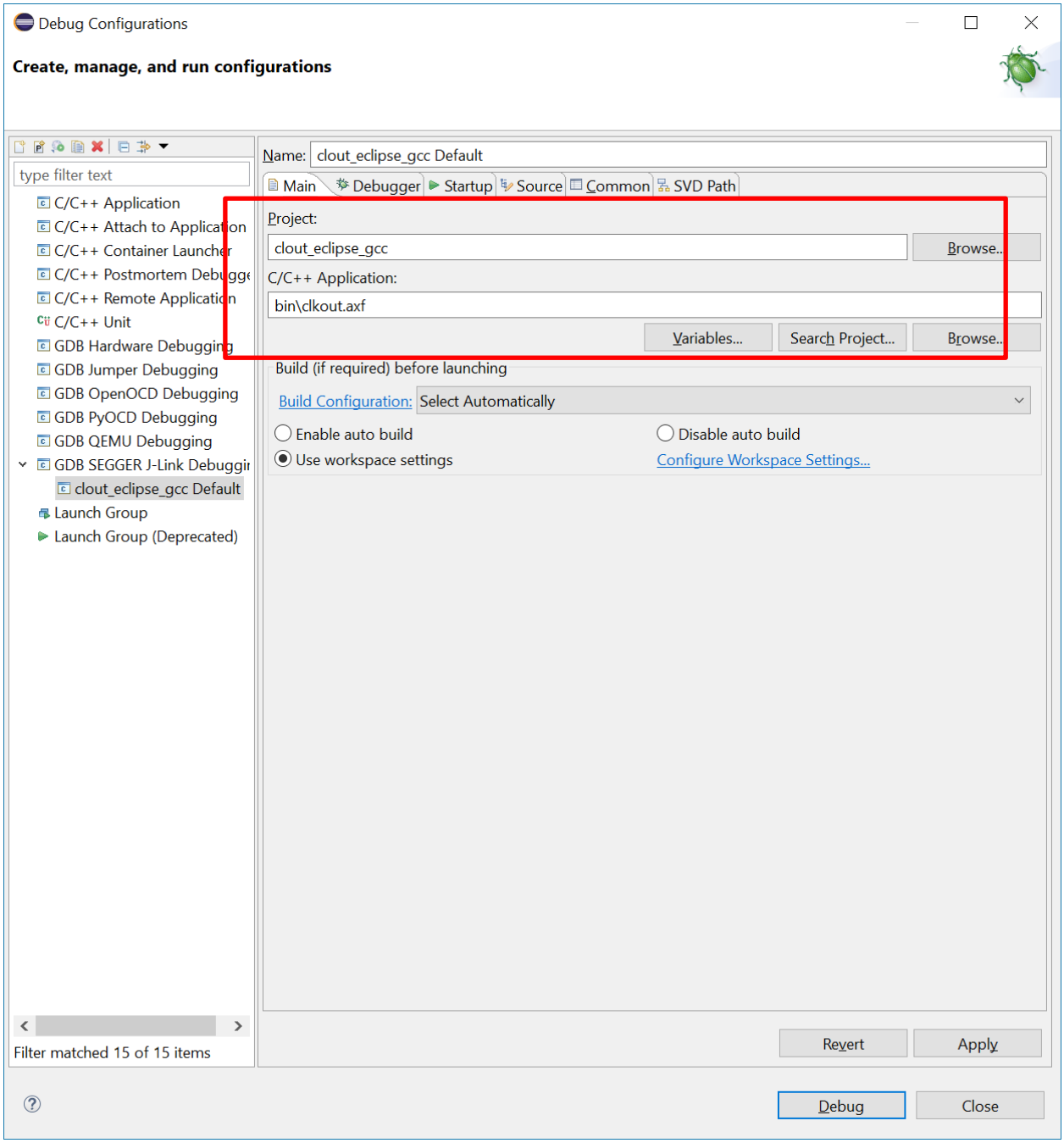
```
Problems  Tasks  Console  Properties
CDT Build Console [clout_eclipse_gcc]
17:22:10 **** Build of configuration Default for project clout_eclipse_gcc ****
make all
 Compiling gcc ../src/clkout.c
 Compiling gcc ../../../../../utils/am_util_delay.c
 Compiling gcc ../../../../../utils/am_util_faultisr.c
 Compiling gcc ../../../../../utils/am_util_stdio.c
 Compiling gcc ../../../../../devices/am_devices_led.c
 Compiling gcc startup_gcc.c
 Linking gcc bin/clkout.axf
 Copying gcc bin/clkout.bin...

17:22:11 Build Finished. 0 errors, 0 warnings. (took 1s.376ms)
```

6. Project Debugging

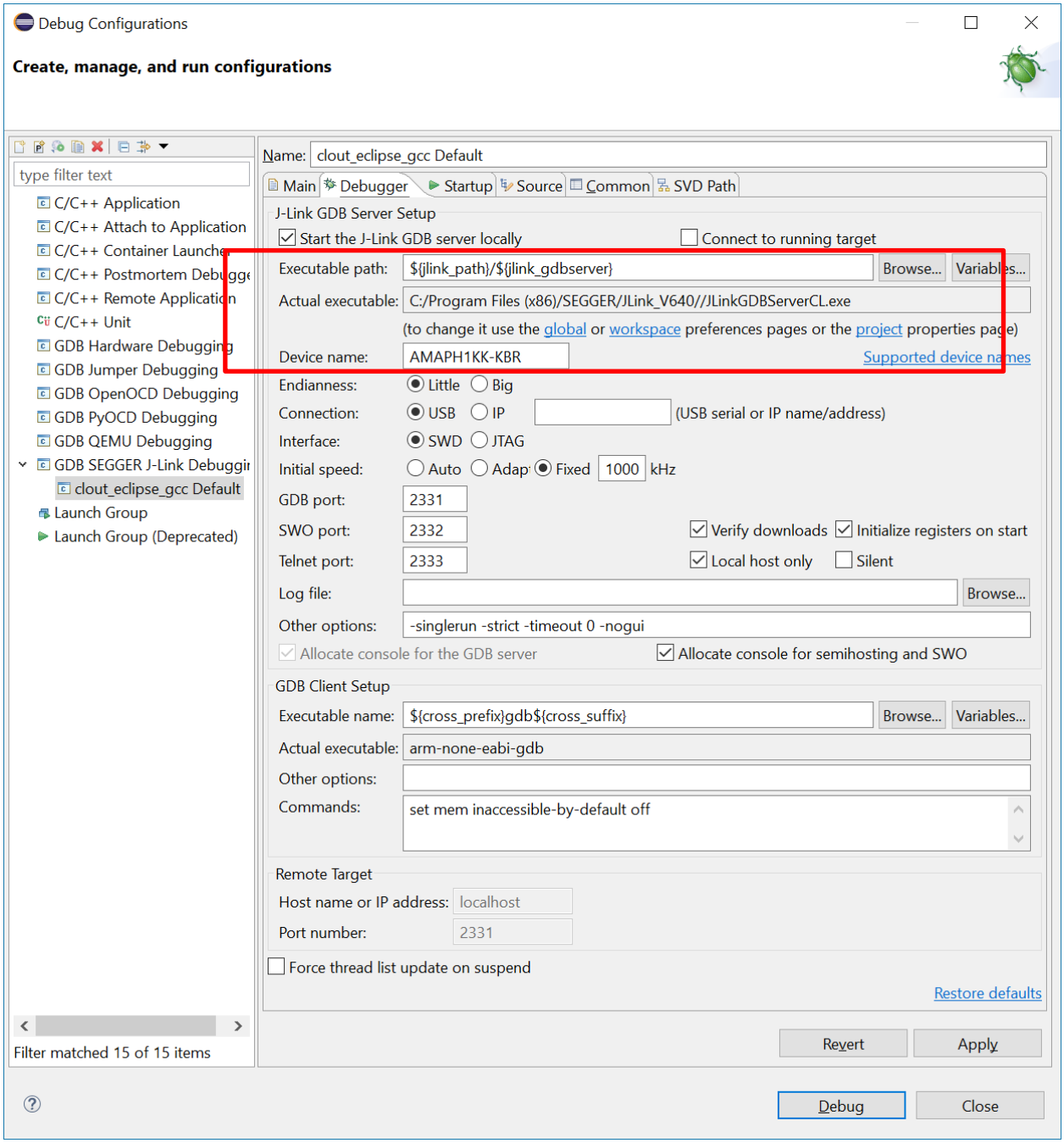
1. Navigate to `Run`>`Debug Configurations…`.



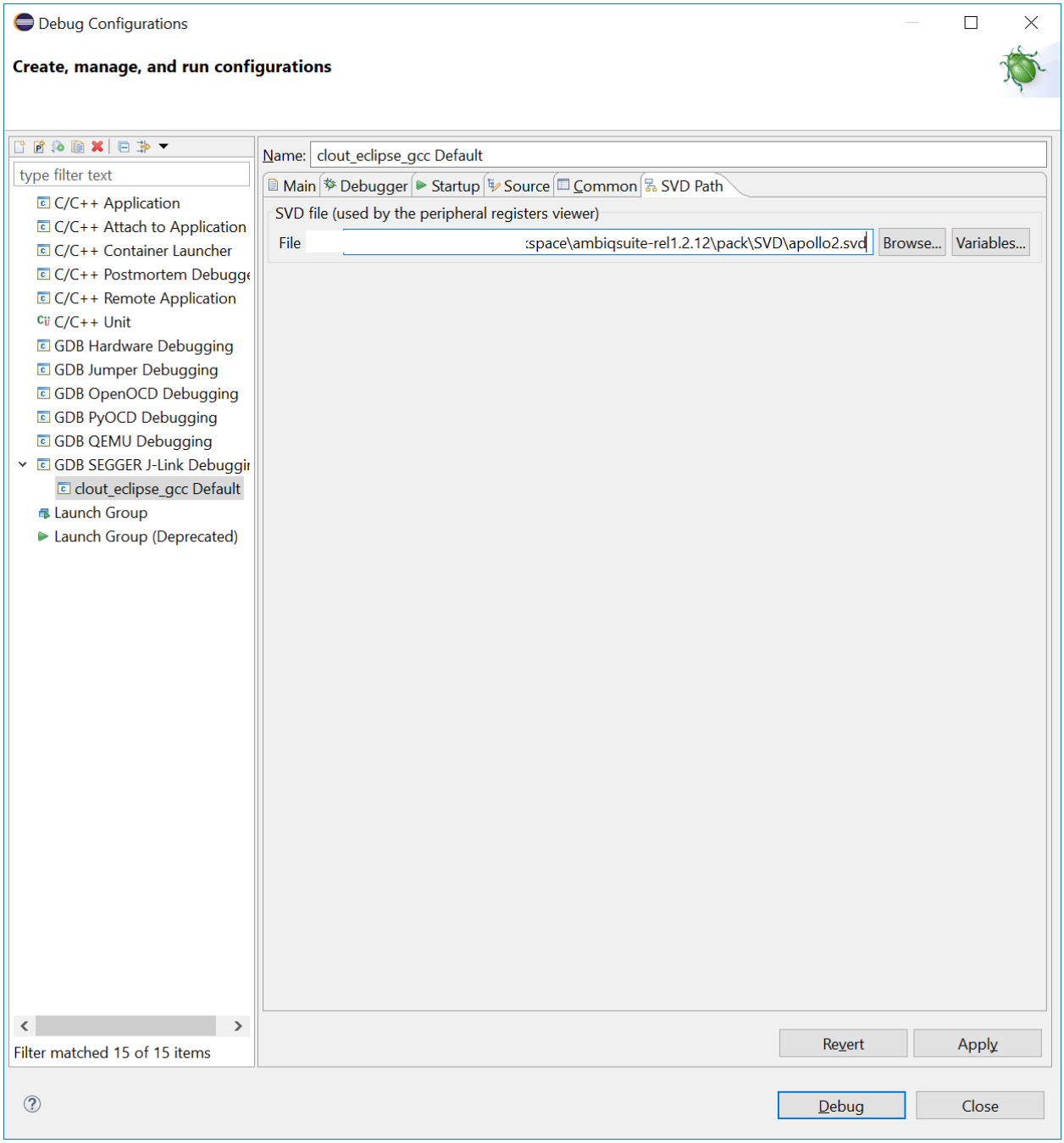
2. Right click on `GDB SEGGER J-Link Debugging` and select `New Configuration`.
3. In `Main` page, make sure `Project` has the name identical to the one set in project import and `C/C++ Application` pointed to the corresponding .axf file.



4. In `Debugger` page, make sure `Actual executable` is correctly interpreted and `Device name` is added.

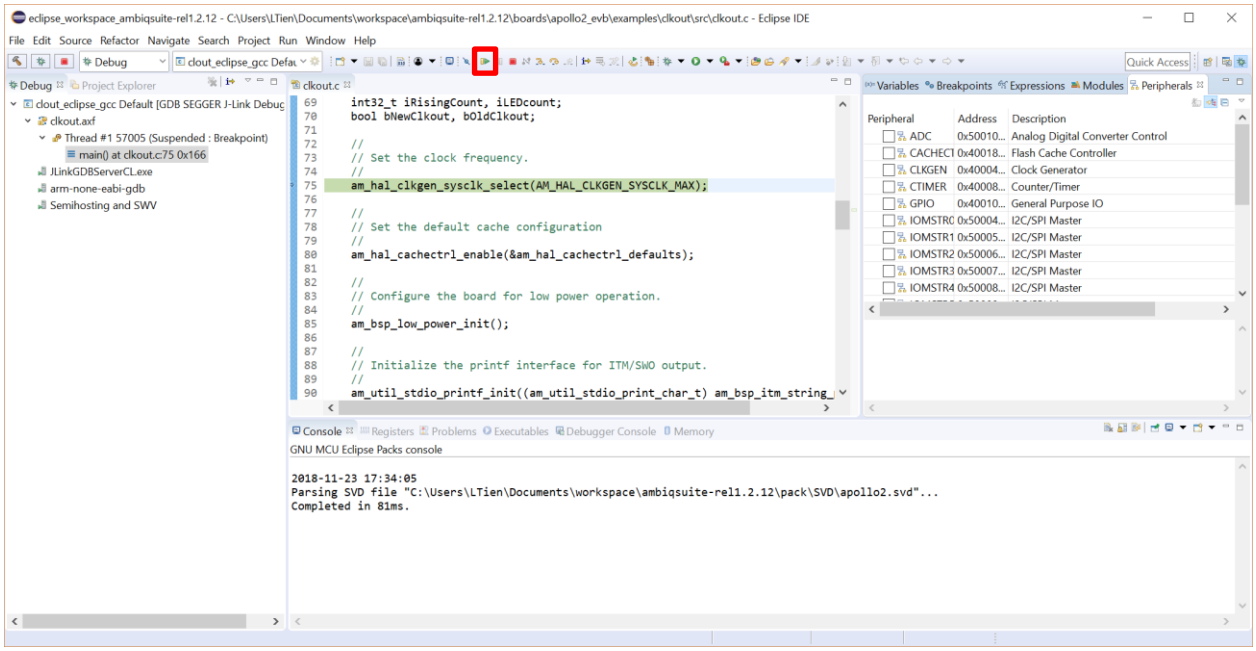


5. In SVD page, add the path to the targeting board SVD file located in <SDK>/pack. Click Apply and Debug.



6. After the debugger is launched, the program stops at the main function. Click the run icon highlighted in the red frame below. The LEDs on EVB shall start to blink.

Application Note – Eclipse Environment Setup



7. Trouble Shooting

1. Make sure Windows Environment variable *Path* is configured correctly. For Windows 10, right click on `This PC` and navigate to `Properties>Advanced system settings>Environment Variables…`. Check the variable *Path*.
2. Please contact regional Ambiq Micro FAEs if you encounter further questions.

Application Note – Eclipse Environment Setup

Contact Information

Address	Ambiq Micro, Inc. 6500 River Place Blvd. Building 7, Suite 200 Austin, TX 78730
Phone	+1 (512) 879-2850
Website	http://www.ambiqmicro.com
General Information	info@ambiqmicro.com
Sales	sales@ambiqmicro.com
Technical Support	support@ambiqmicro.com

Legal Information and Disclaimers