Steps to setup and debug generated Papyrus-RT models on Windows using CMake & Cygwin

Contents

Overview

This pictorial guide describes a basic Cygwin configuration for building and debugging generated Papyrus-RT model projects. The developer should be familiar with generating Papyrus-RT model projects, and a basic understanding of Cygwin and CMake. This document does not address installation procedures nor provide methods for troubleshooting related code generation and build issues.

Requirements

The steps below depend on the following Windows configuration:

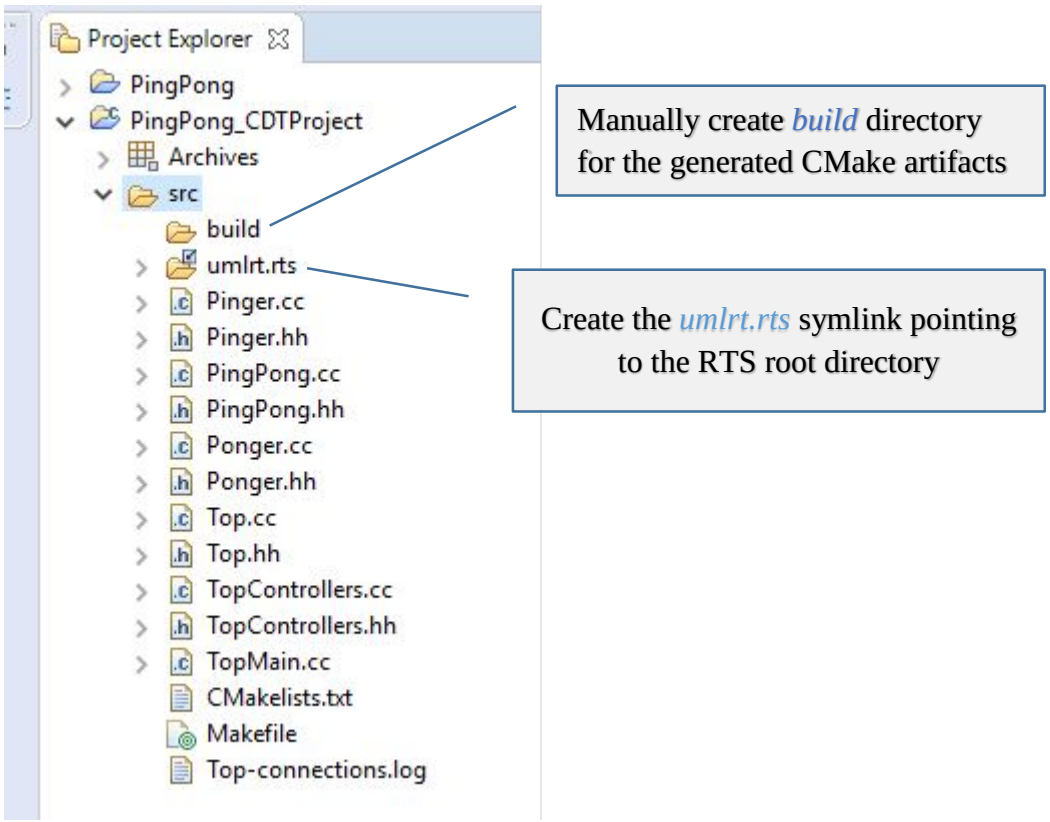
- The Neon developer environment for Papyrus-RT
- A current 32-bit installation of Cygwin including CMake and related dev packages
- The PingPong project tutorial

Steps

1. Setup required directories

The generated CDT project requires an additional build directory, and a symlink to the RTS root directory. The additional *build* directory has no particular naming constraints. The symlink, however, should be named *umlrt.rts* to avoid the necessity of undocumented configuration changes. For example:

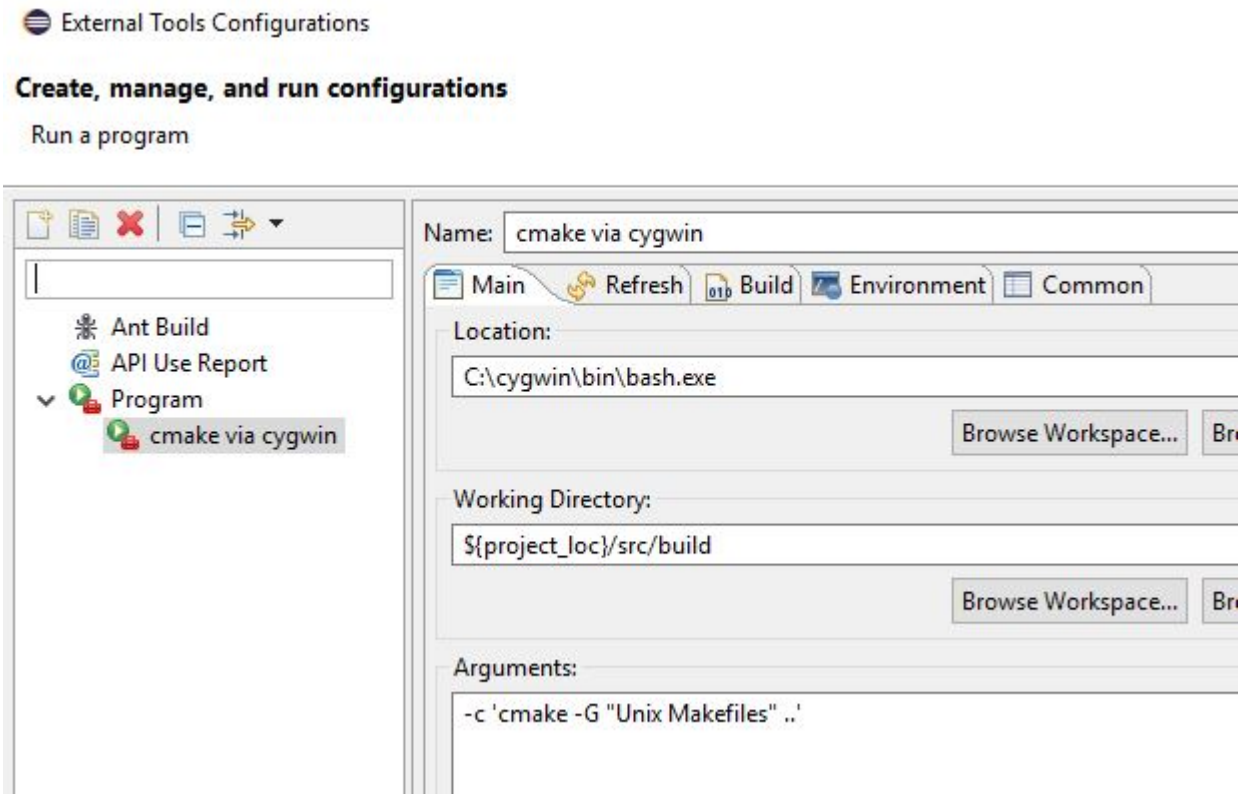
```
umlrt.rts -> [home]\papyrus-rt-master\git\org.eclipse.papyrus-rt\plugins\umlrt\runtime\rts
```



2. Create an external CMake tool

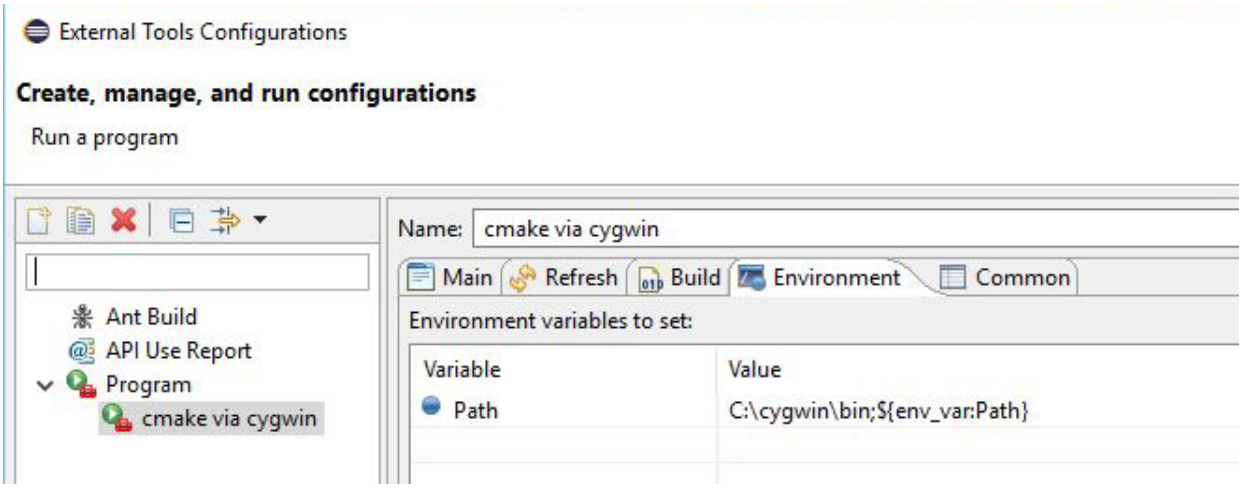
CMake will be executed in the context of Cygwin. The following tool configuration shows how this can be accomplished as a one-line command. The Working Directory specifies the build directory created in the previous step. For clarity, the arguments to *bash*:

```
-c 'cmake -G "Unix Makefiles" ..'
```



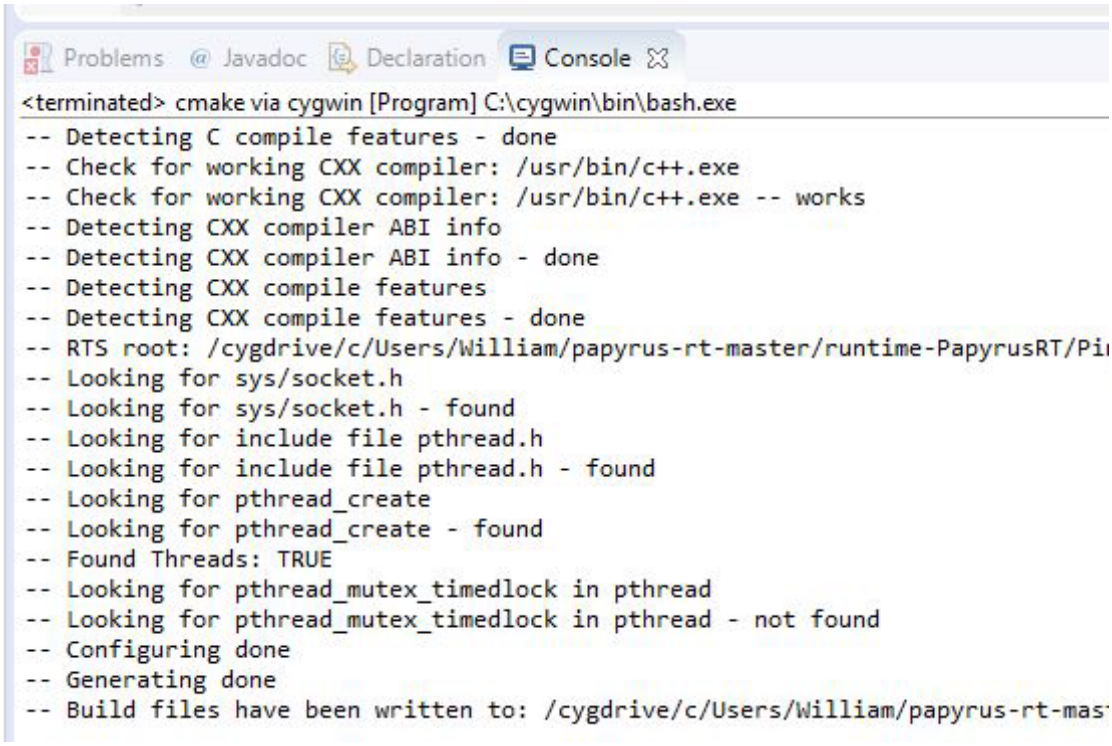
In cases where Cygwin's bin is excluded from the session path, the Path parameter can be updated as shown below. The recommended character case and value for the Path variable is as follows:

```
Path => C:\cygwin\bin;${env_var:Path}
```

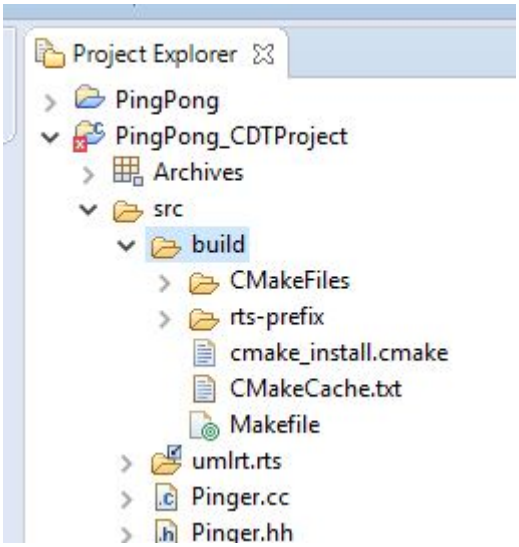


3. Generate the CMake structure

With the CDT project focused, run the *cmake via cygwin* external tool to generate the CMake artifacts. The CMake results can be viewed in the Console window:

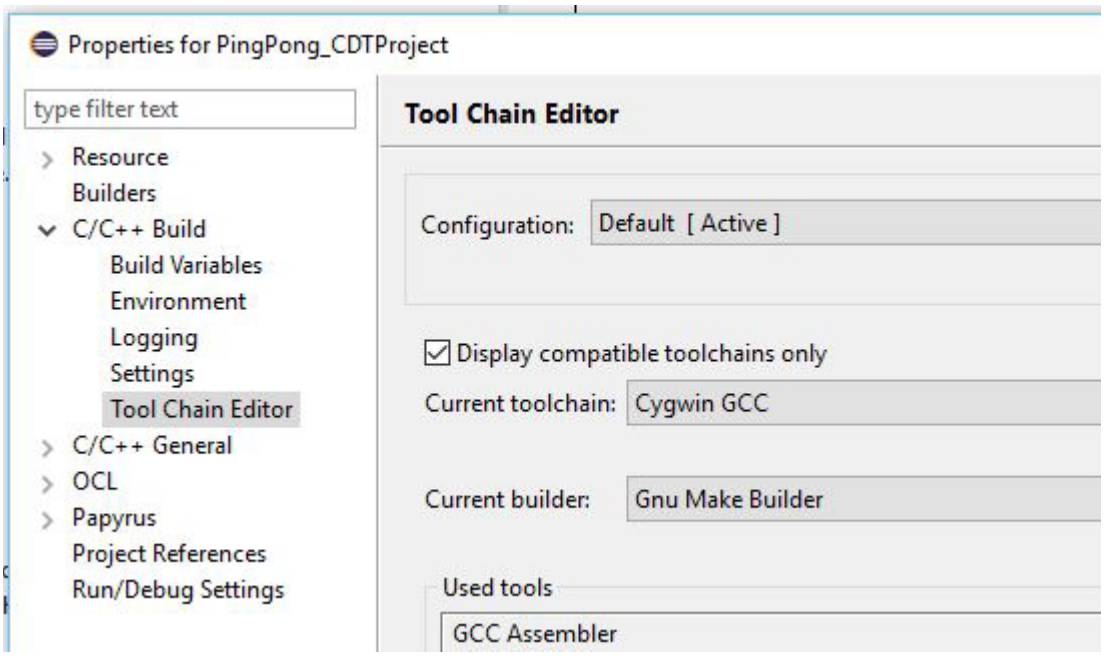


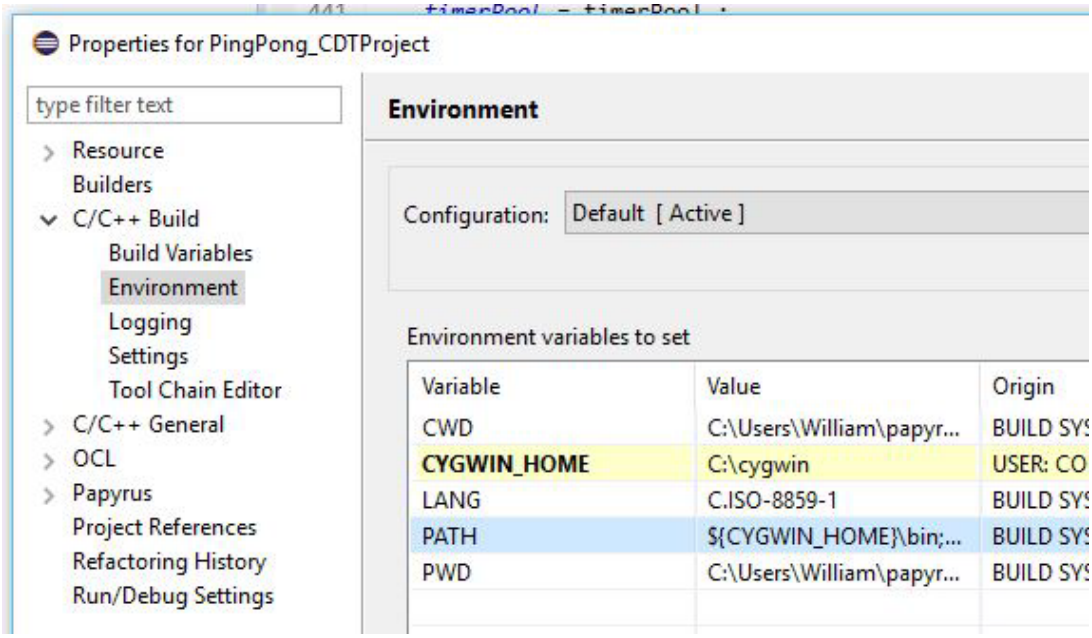
Refresh the build folder to see the artifacts:



4. Verify the CDT Project's Toolchain

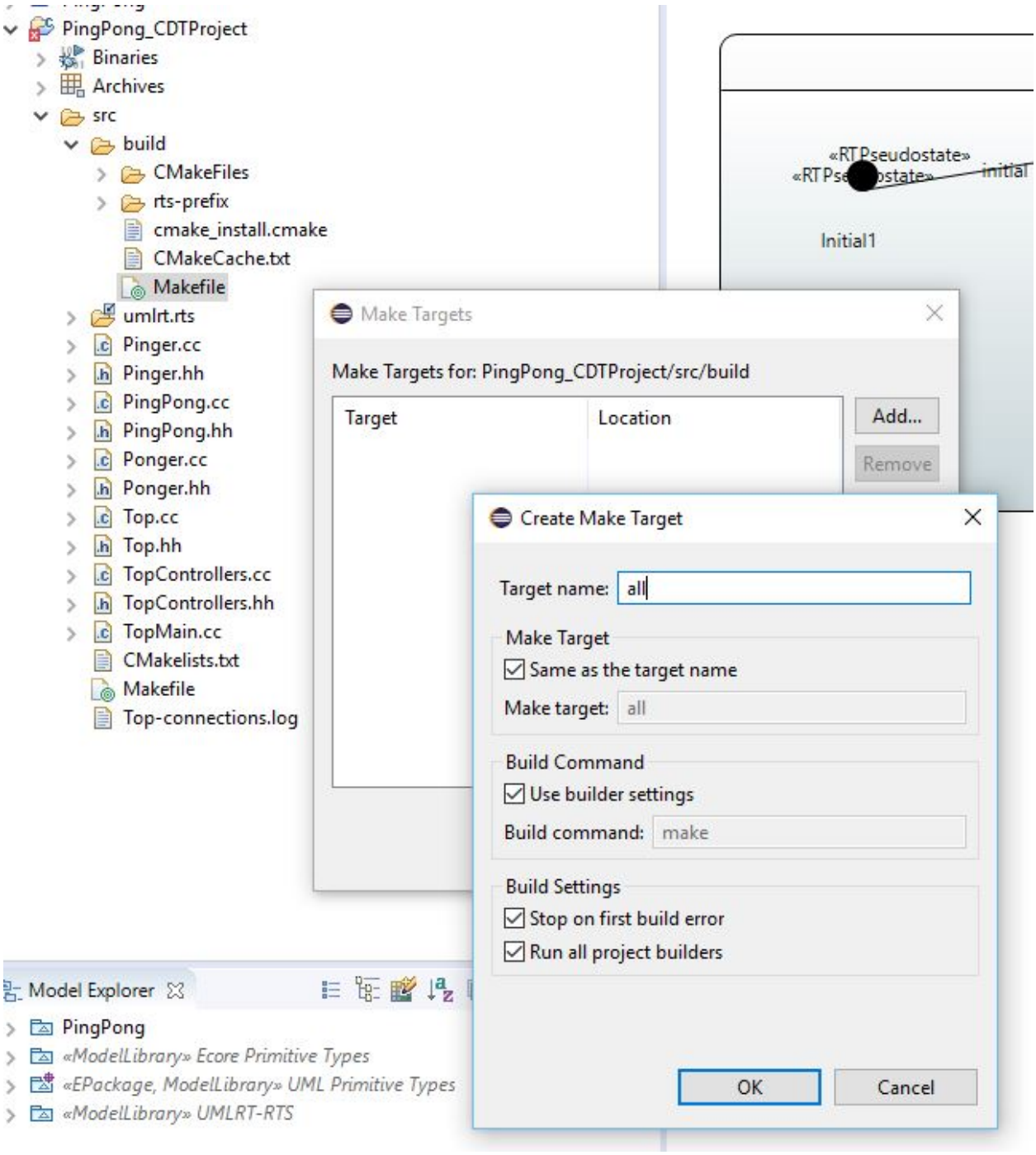
Make sure *Cygwin GCC* is the project's active toolchain.





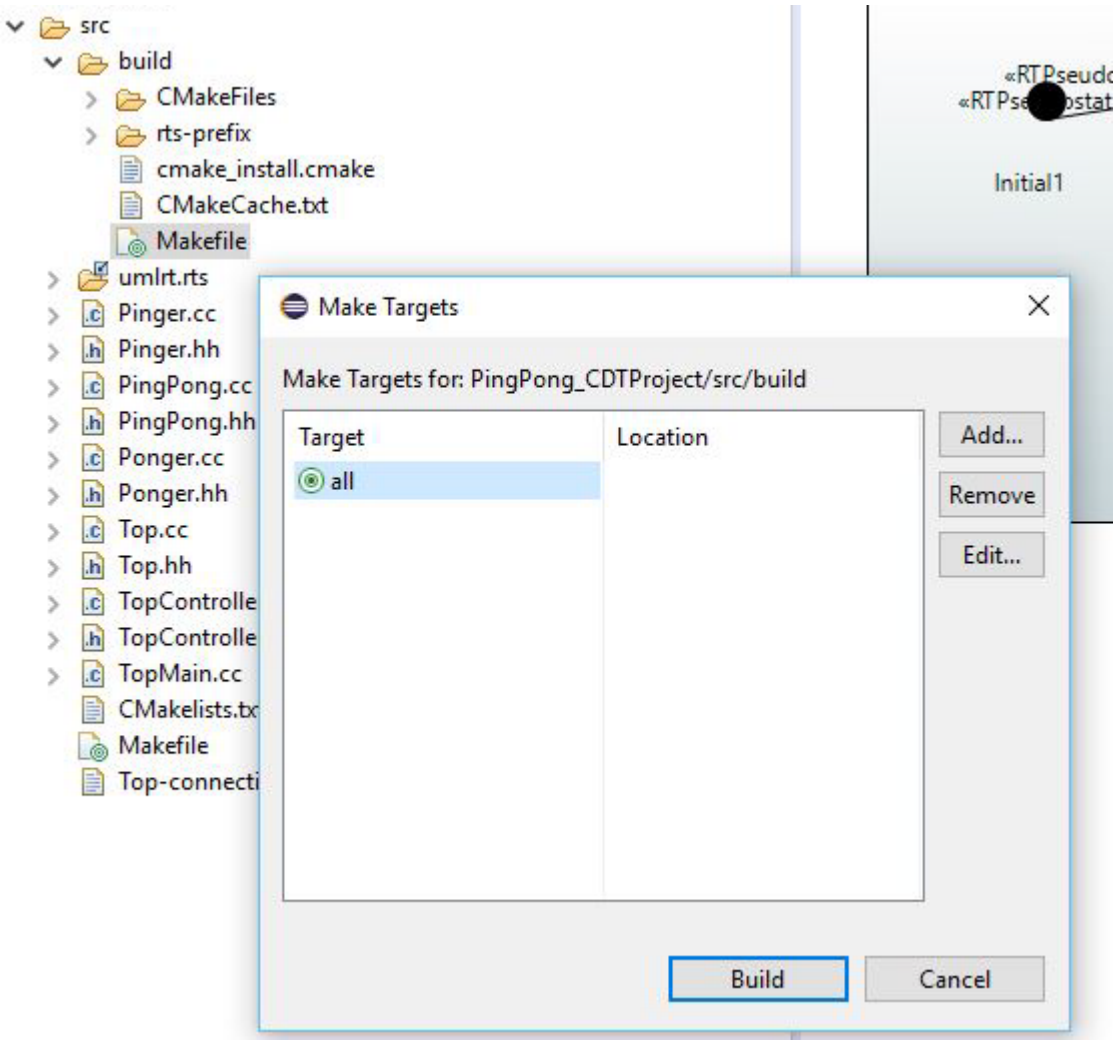
5. Update the Make Target

Select the Makefile generated by CMake and create the Make Target, *all*. The dialog is available via the context menu, Make Targets / Build…



6. Build the CDT Project

Select the target, *all*, and run the build.

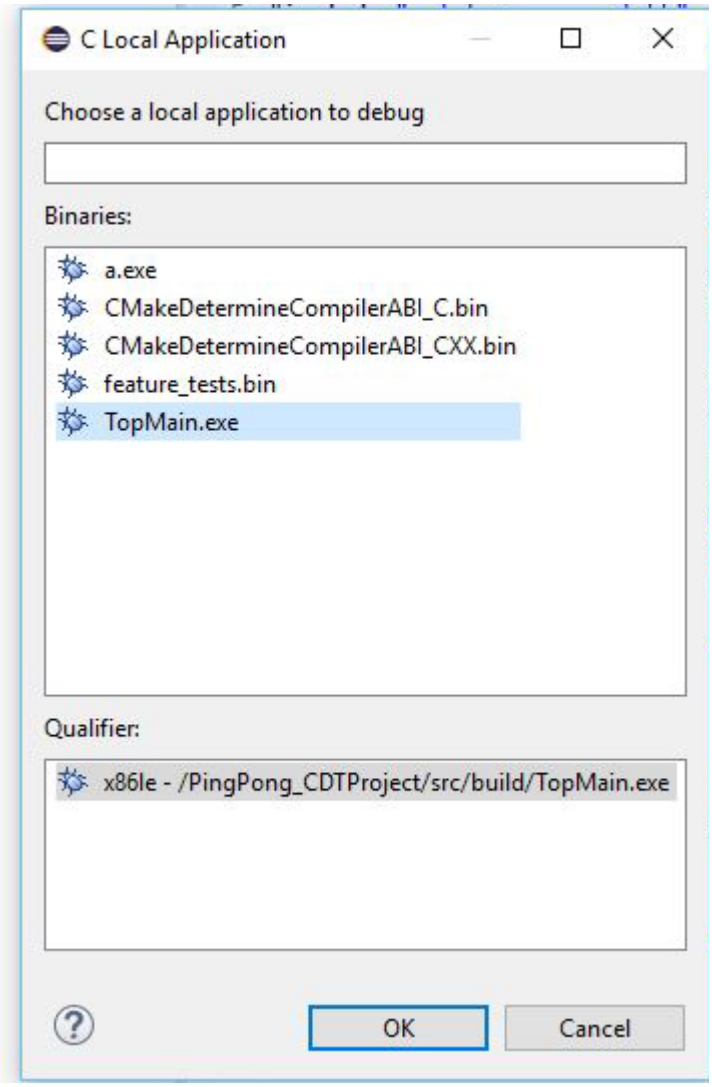


The build results can be viewed in the Console window:

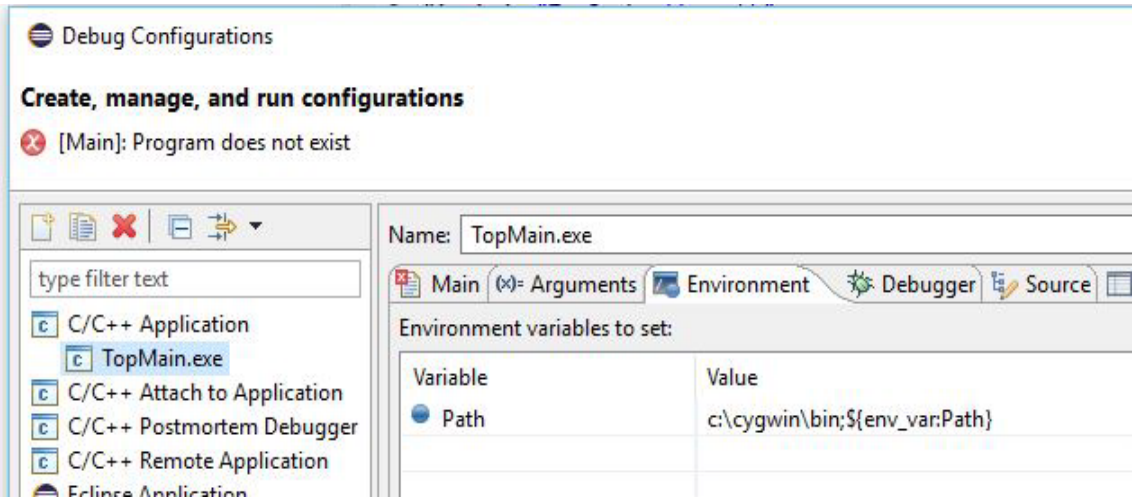
```
[100%] Linking CXX static library librtsd.a
[100%] Built target rts
Install the project...
-- Install configuration: "Debug"
-- Installing: /cygdrive/c/Users/William/papyrus-rt-master/runtime
[ 53%] Completed 'rts'
[ 53%] Built target rts
Scanning dependencies of target TopMain
[ 60%] Building CXX object CMakeFiles/TopMain.dir/TopMain.cc.o
[ 66%] Building CXX object CMakeFiles/TopMain.dir/PingPong.cc.o
[ 73%] Building CXX object CMakeFiles/TopMain.dir/Pinger.cc.o
[ 80%] Building CXX object CMakeFiles/TopMain.dir/Ponger.cc.o
[ 86%] Building CXX object CMakeFiles/TopMain.dir/Top.cc.o
[ 93%] Building CXX object CMakeFiles/TopMain.dir/TopControllers.c
[100%] Linking CXX executable TopMain.exe
[100%] Built target TopMain

22:48:52 Build Finished (took 1m:1s.668ms)
```

7. Setup a Debug Configuration

Create a C/C++ Application configuration. Select TopMain.exe if prompted.

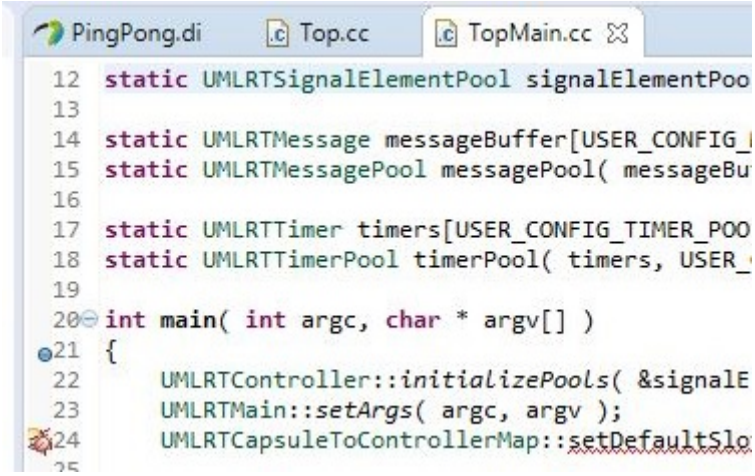


Update the Path variable required to locate Cygwin DLLs.

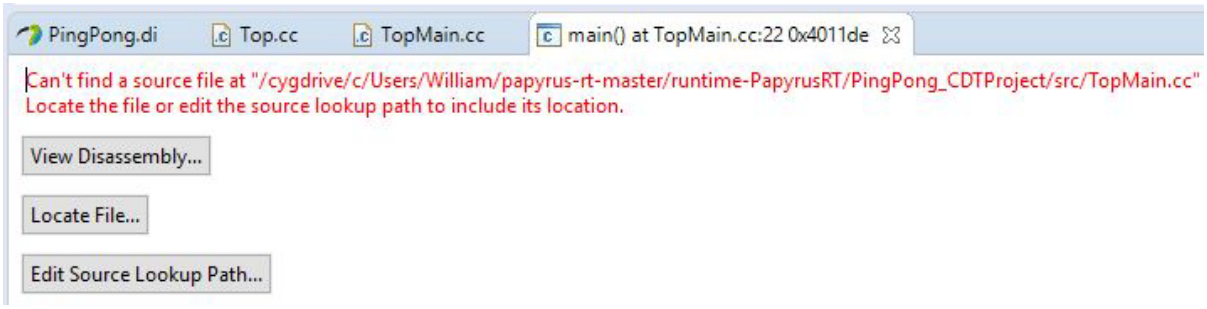


8. Debug the Modeled Application

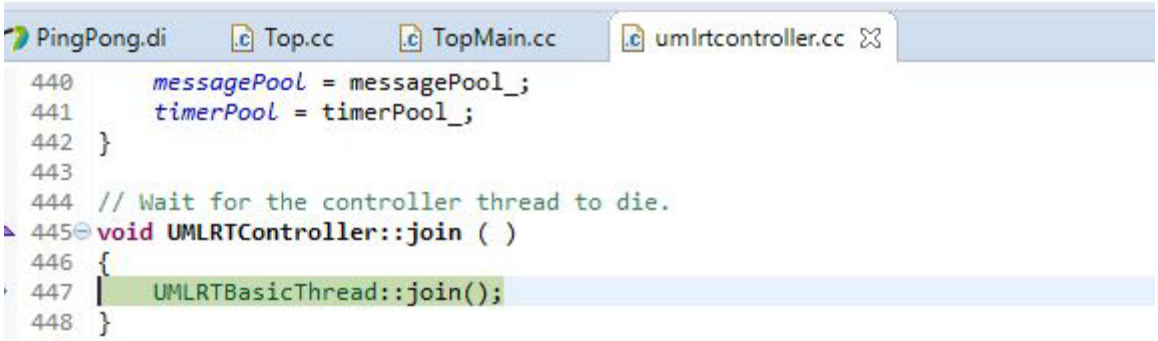
Set a breakpoint at the beginning of main, and start debugging.



The initial debug session may ask for the location of the source. Navigate to the source file using *Locate File…*



The RTS library can be stepped into as shown:



The results: