Ecplise CDT setup guide for darktable in Windows (and Linux)

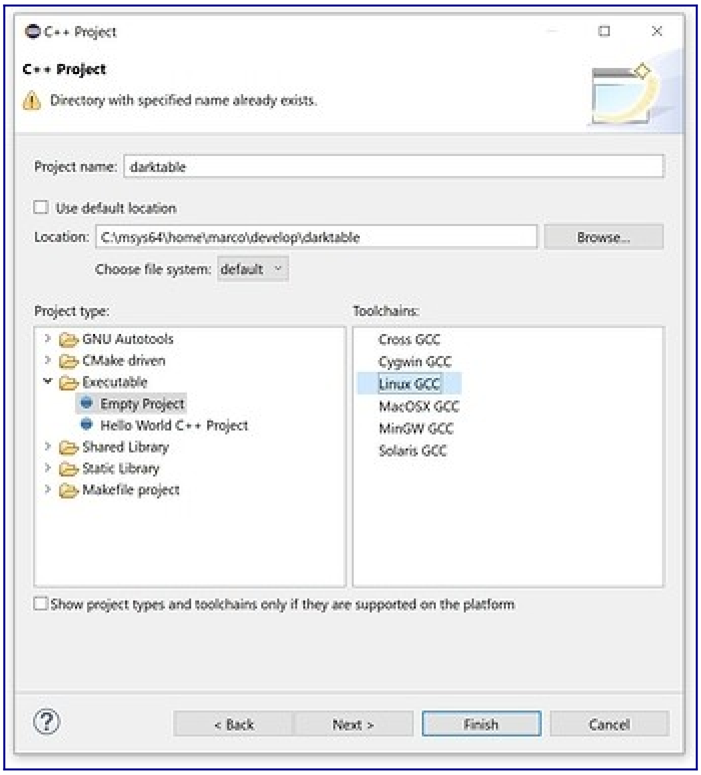
I have been requested to post a guide about setting Eclipse CDT IDE for darktable, so here it is.
This guide assumes that you are already able to build darktable from sources (in Windows through MSYS2), by following the build guide on GitHub.
Basic knowledge of Eclipse CDT is also required.
If you want to contribute to darktable project, normally you need to create your own fork of the project in GitHub.
We assume that darktable sources are in C:\msys64\home\<username>\develop\darktable (/home/<username>/develop/darktable in Linux)

Step 1. Install Ecplise CDT from website

Step 2. From the Ecplise Marketplace, search and install the addon cmake4eclipse

Step 3. Import Darktable source code

In eclipse: File→ New → Project → C/C++ → C++ Project
In the dialog, choose “Linux GCC toolchain” and point to the source code folder. See below



C++ Project

C++ Project

Directory with specified name already exists.

Project name: darktable

□ Use default location

Location: C:\msys64\home\marco\develop\darktable

Choose file system: default

Project type: Executable → Empty Project

Toolchains: Linux GCC

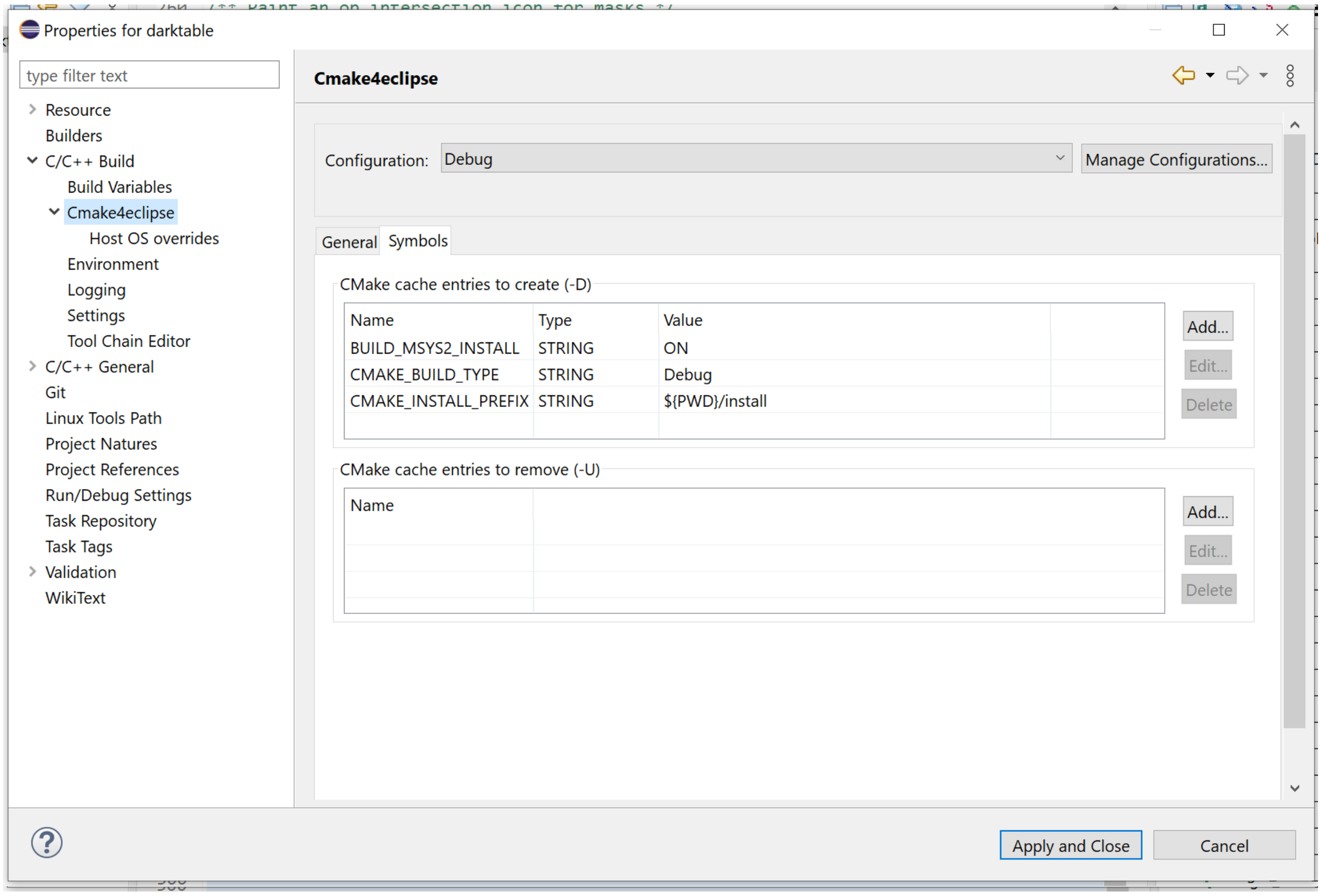
□ Show project types and toolchains only if they are supported on the platform

Hit “Finish”

Step 4. Set Project Properties

Go to the properties of the newly created Eclipse project. For example in Project explorer window, right click on the project → Properties

Setup as follows (hit “Apply” at every step)



Configuration: Debug

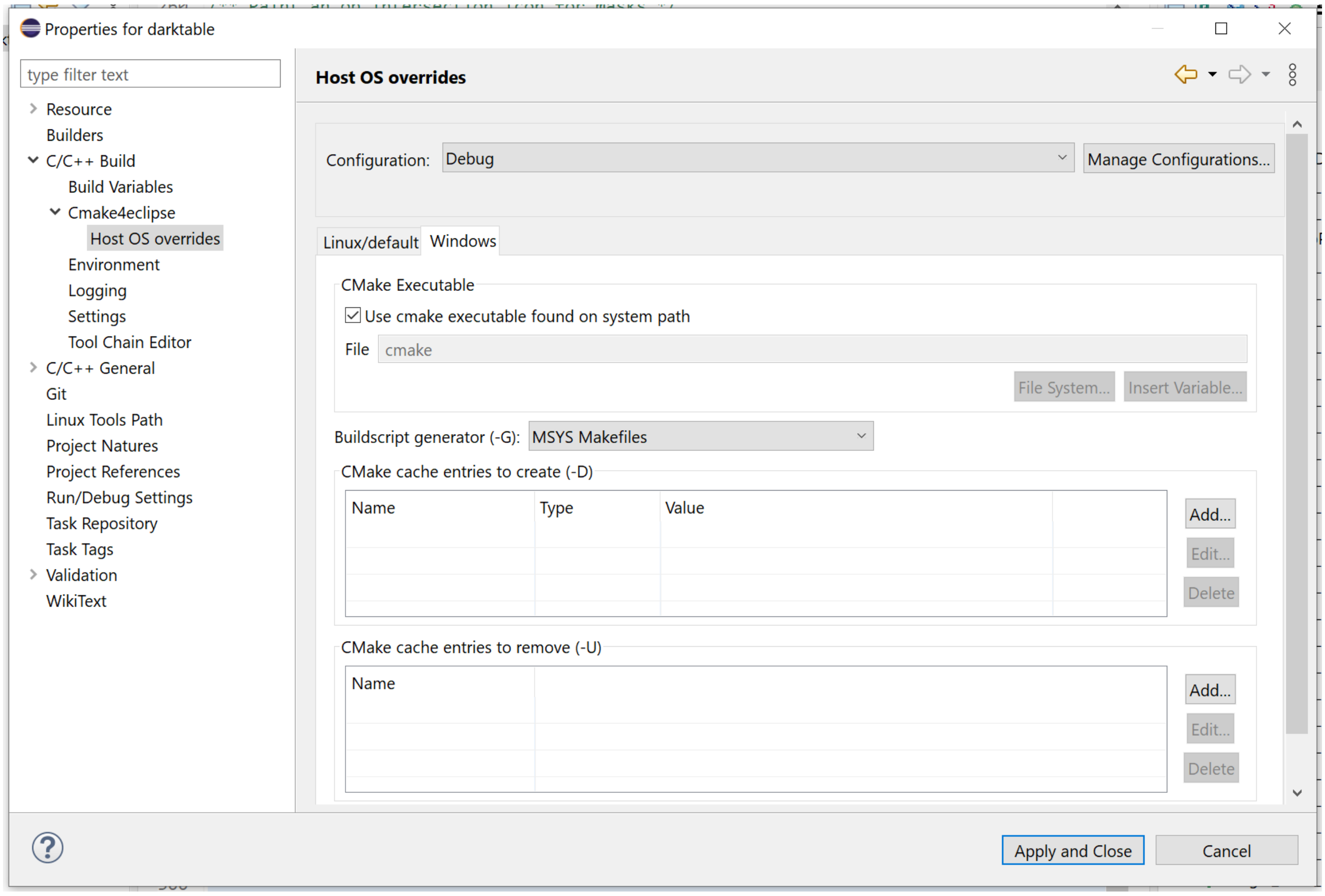
CMake cache entries to create (-D)

Name	Type	Value
BUILD_MSYS2_INSTALL	STRING	ON
CMAKE_BUILD_TYPE	STRING	Debug
CMAKE_INSTALL_PREFIX	STRING	${PWD}/install

CMake cache entries to remove (-U)

Name

(in Linux, omit BUILD_MSYS2_INSTALL)



Host OS overrides

Configuration: Debug

CMake Executable

☑ Use cmake executable found on system path

File cmake

Buildscript generator (-G): MSYS Makefiles

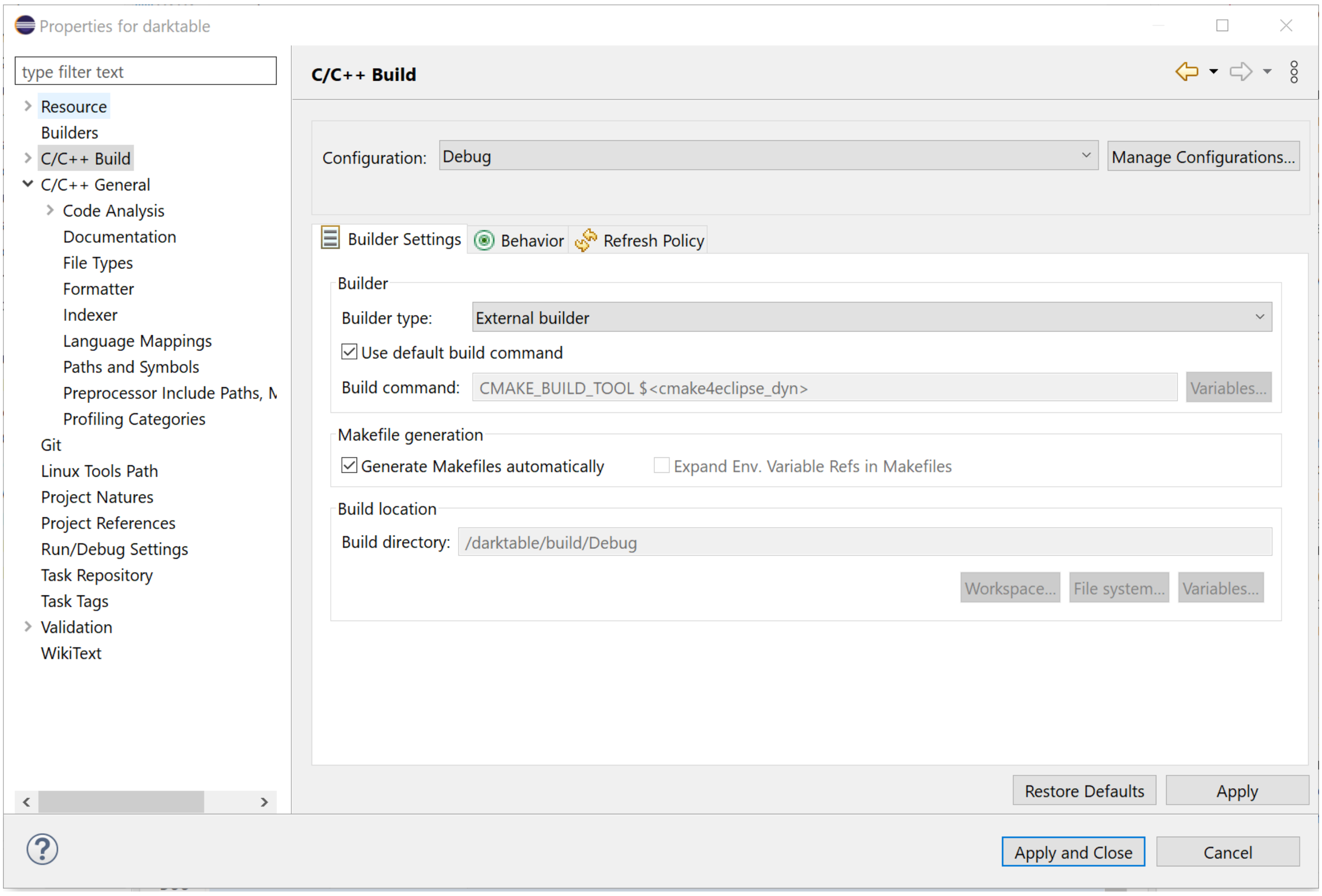
CMake cache entries to create (-D)

Name	Type	Value

CMake cache entries to remove (-U)

Name

For newer releases of darktable you can also use Ninja as Buildscript generator, which is faster, provided you installed in MSYS2 the relative toolchain.
(for Linux, use the Linux/Default tab and put Unix Makefiles or Ninja as buildscript generator)



C/C++ Build

Configuration: Debug

Builder

Builder type: External builder

☑ Use default build command

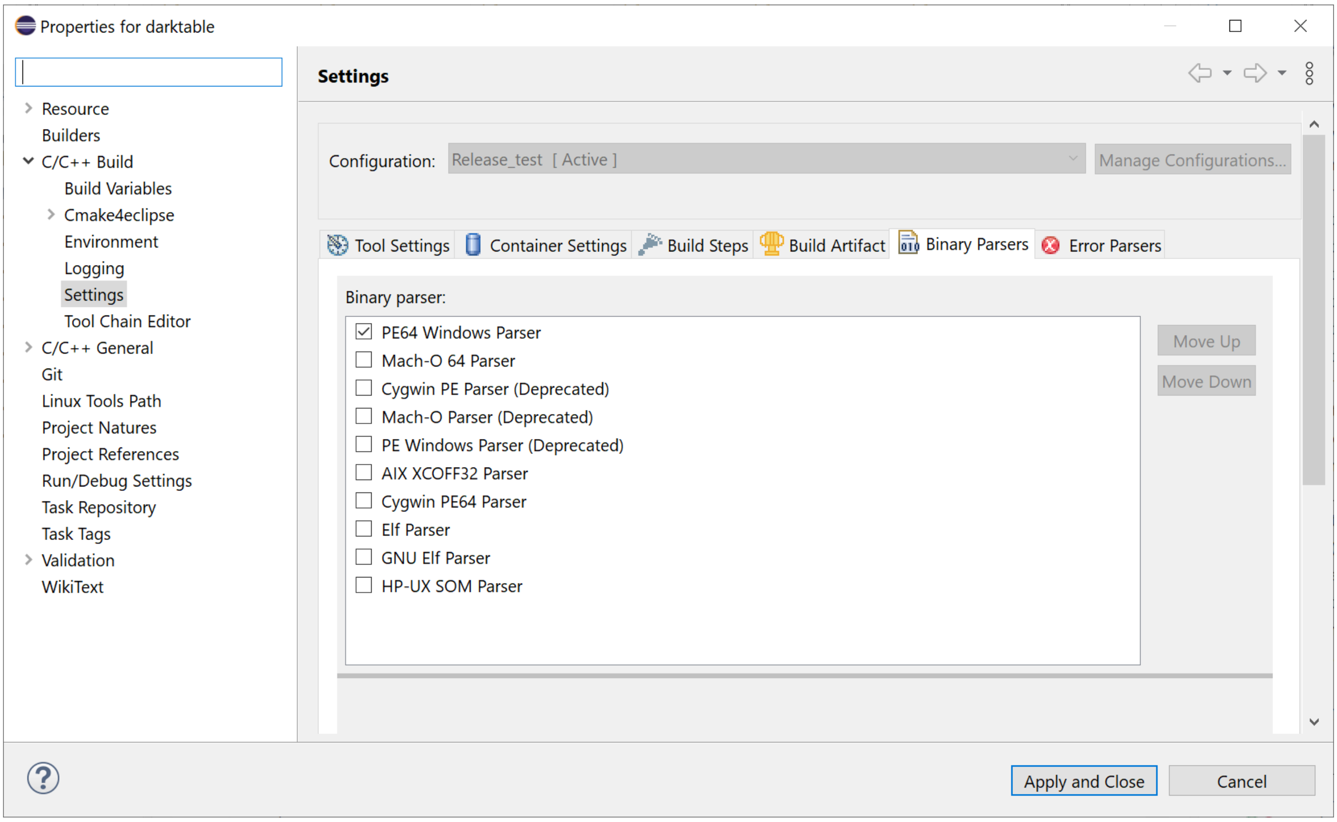
Build command: CMAKE_BUILD_TOOL $<cmake4eclipse_dyn>

Makefile generation

☑ Generate Makefiles automatically □ Expand Env. Variable Refs in Makefiles

Build location

Build directory: /darktable/build/Debug



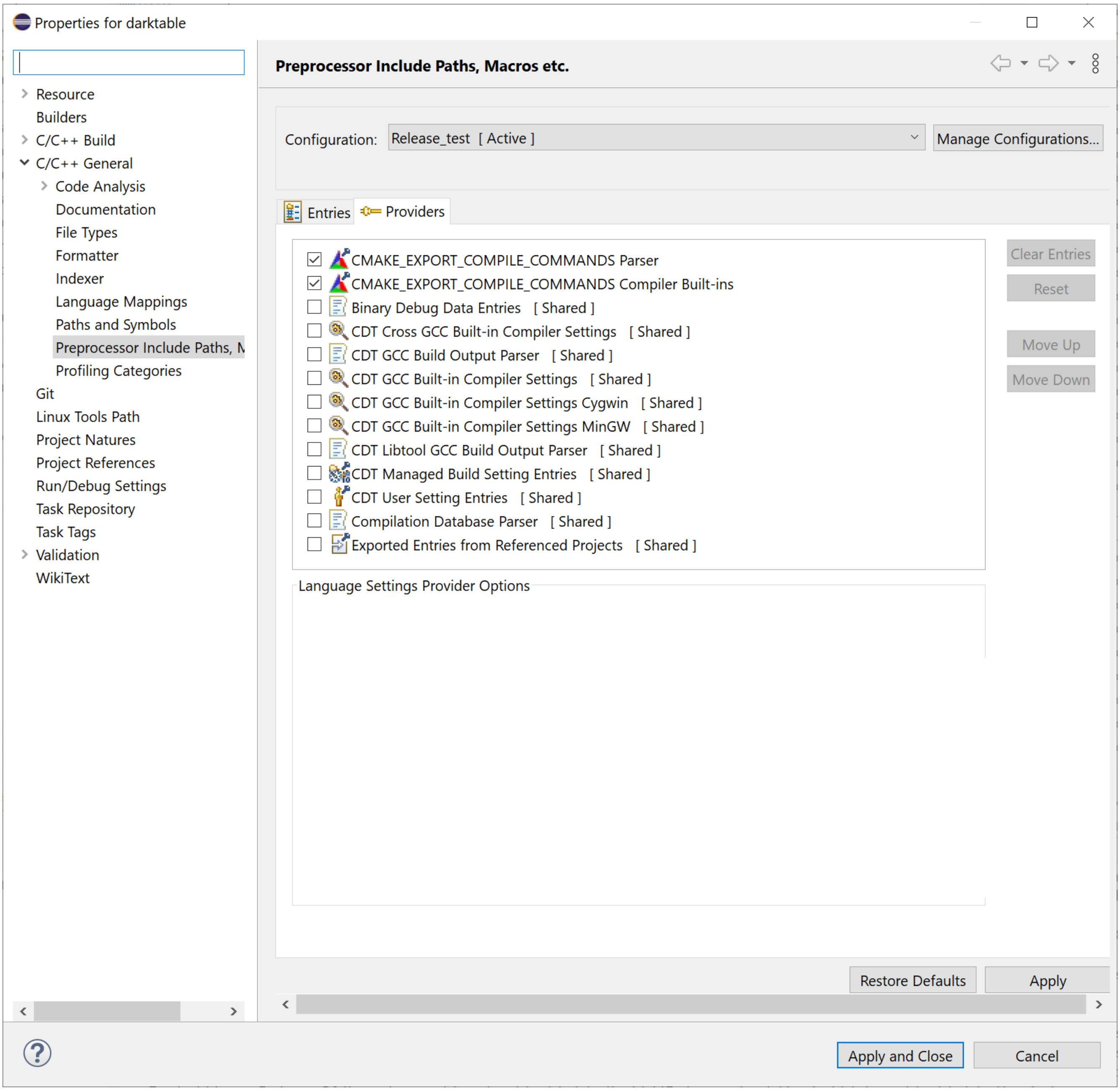
Settings

Configuration: Release_test [Active]

Binary parser:

☑ PE64 Windows Parser
□ Mach-O 64 Parser
□ Cygwin PE Parser (Deprecated)
□ Mach-O Parser (Deprecated)
□ PE Windows Parser (Deprecated)
□ AIX XCOFF32 Parser
□ Cygwin PE64 Parser
□ Elf Parser
□ GNU Elf Parser
□ HP-UX SOM Parser

(for Linux, use GNU Elf Parser)

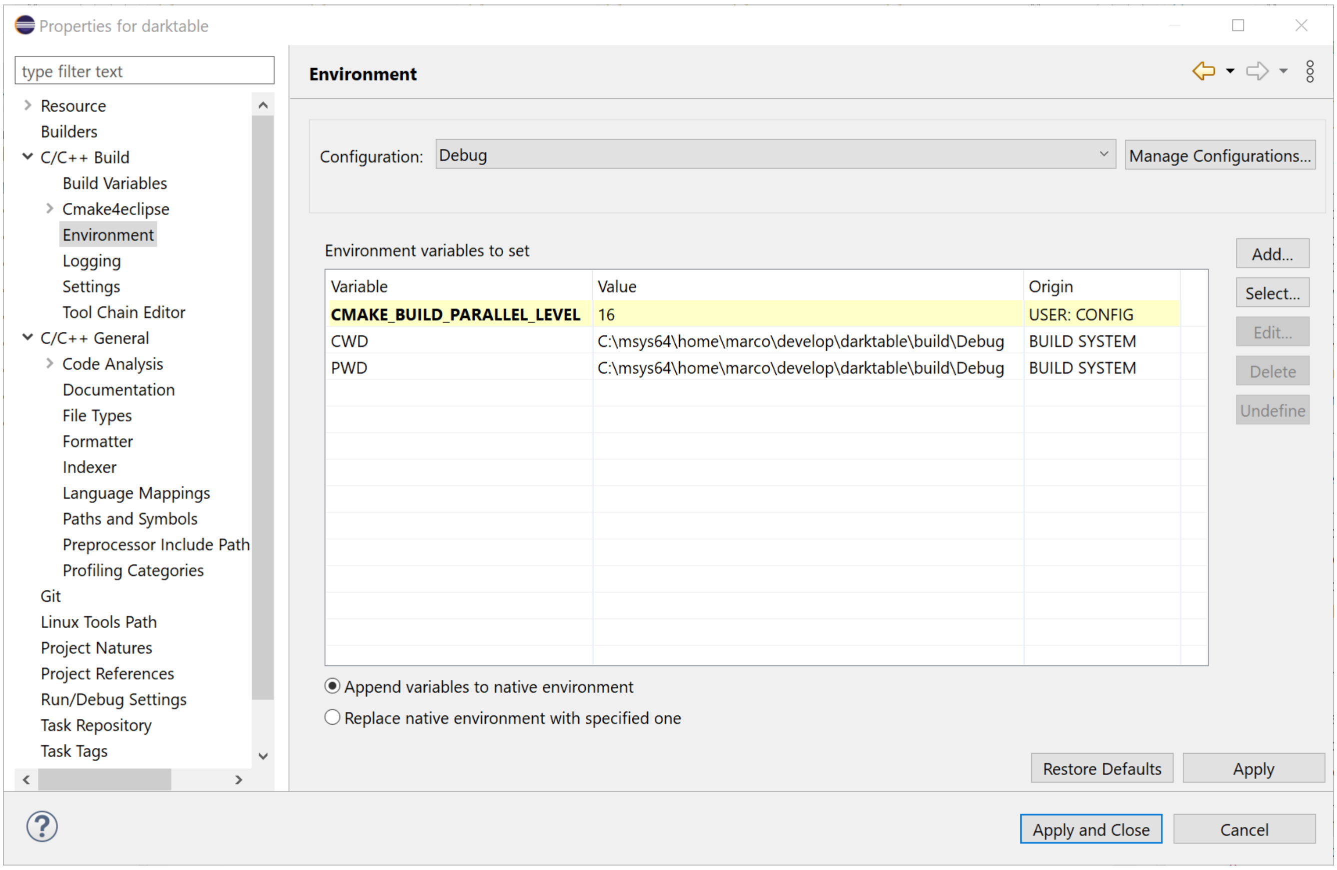


Preprocessor Include Paths, Macros etc.

Configuration: Release_test [Active]

☑ CMAKE_EXPORT_COMPILE_COMMANDS Parser
☑ CMAKE_EXPORT_COMPILE_COMMANDS Compiler Built-ins
□ Binary Debug Data Entries [Shared]
□ CDT Cross GCC Built-in Compiler Settings [Shared]
□ CDT GCC Build Output Parser [Shared]
□ CDT GCC Built-in Compiler Settings [Shared]
□ CDT GCC Built-in Compiler Settings Cygwin [Shared]
□ CDT GCC Built-in Compiler Settings MinGW [Shared]
□ CDT Libtool GCC Build Output Parser [Shared]
□ CDT Managed Build Setting Entries [Shared]
□ CDT User Setting Entries [Shared]
□ Compilation Database Parser [Shared]
□ Exported Entries from Referenced Projects [Shared]

Language Settings Provider Options



Environment

Configuration: Debug

Environment variables to set

Variable	Value	Origin
CMAKE_BUILD_PARALLEL_LEVEL	16	USER: CONFIG
CWD	C:\msys64\home\marco\develop\darktable\build\Debug	BUILD SYSTEM
PWD	C:\msys64\home\marco\develop\darktable\build\Debug	BUILD SYSTEM

● Append variables to native environment

○ Replace native environment with specified one

(fol Linux use the corresponding path /home/<username>/develop/darktable/build/Debug)

Hit “Apply and Close”

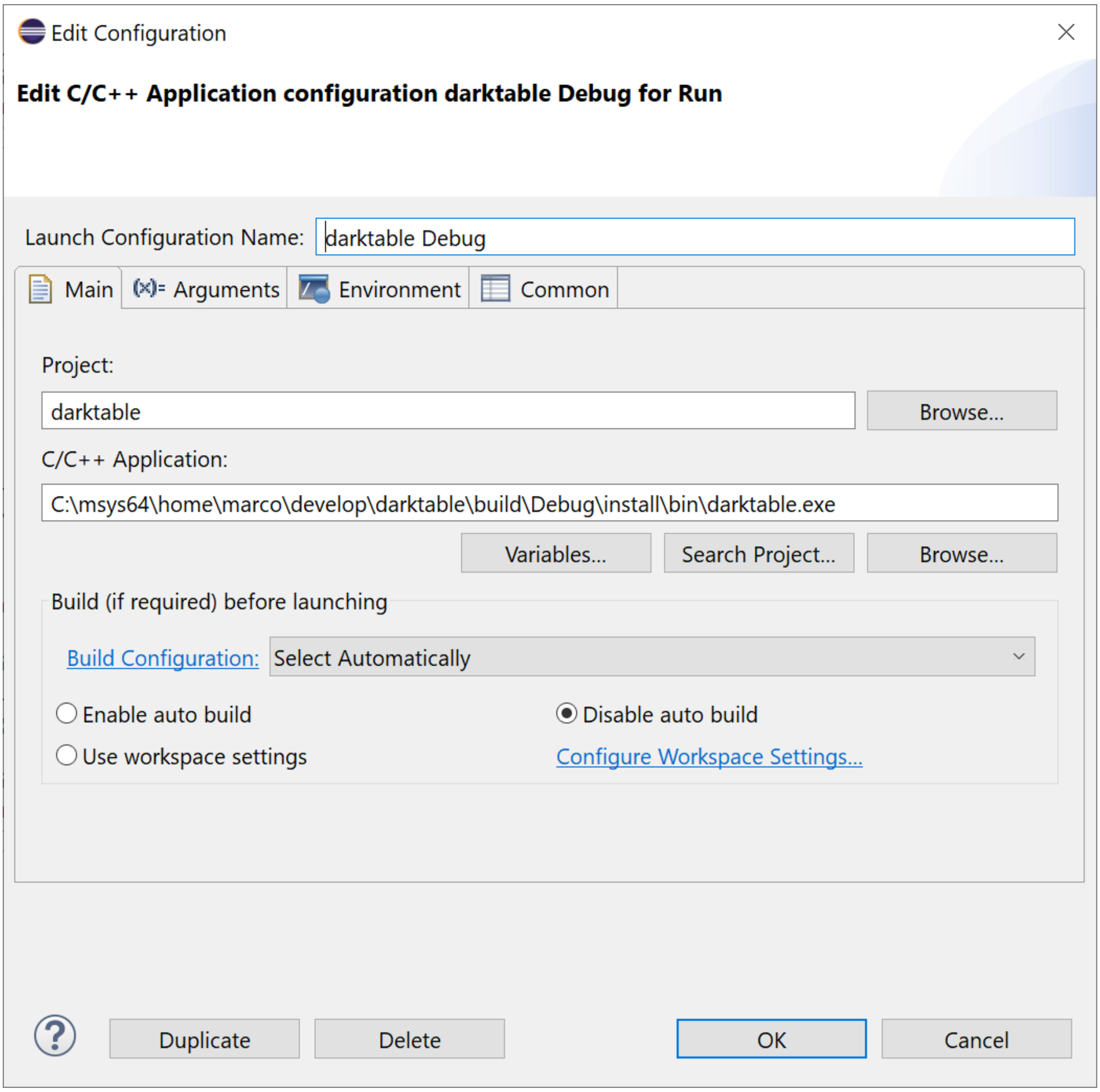
Step 5. Build the project

Eclipse should run CMake first and then make (or ninjia) and install, you can see the output in the Eclipse console.

The initial build could give some errors related to failing build of user manual and/or translations, that should not be a problem.

Incremental builds shall not give errors.

Step 6. Create a run configuration



Edit Configuration

Edit C/C++ Application configuration darktable Debug for Run

Launch Configuration Name: darktable Debug

Project: darktable

C/C++ Application: C:\msys64\home\marco\develop\darktable\build\Debug\install\bin\darktable.exe

Build (if required) before launching

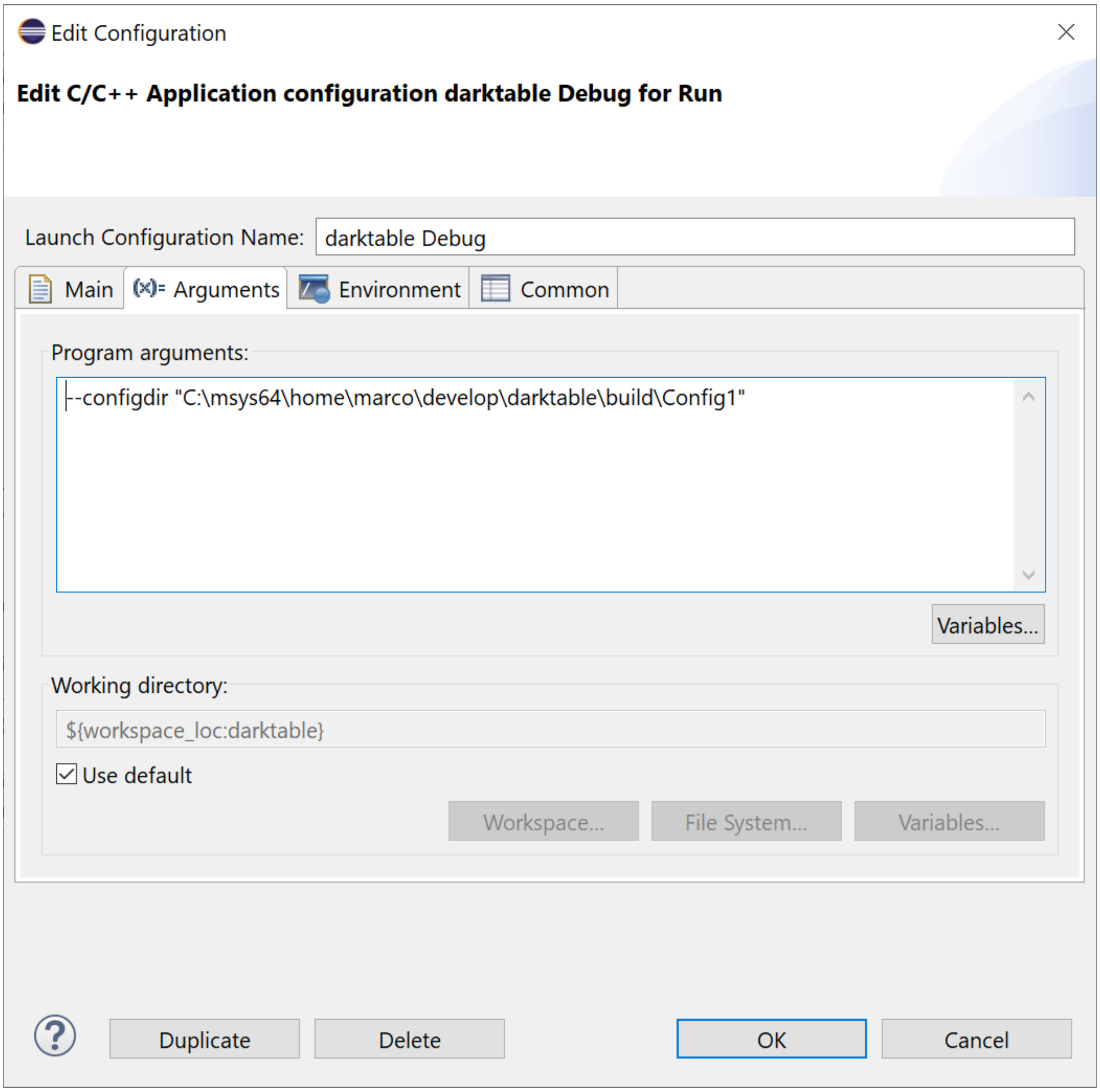
Build Configuration: Select Automatically

○ Enable auto build ● Disable auto build

○ Use workspace settings

(for Linux use the path build/Debug/install/bin/darktable)

It is recommended you specify a configuration folder (you can have more than one)



Edit Configuration

Edit C/C++ Application configuration darktable Debug for Run

Launch Configuration Name: darktable Debug

Program arguments:

```
--configdir "C:\msys64\home\marco\develop\darktable\build\Config1"
```

Working directory:

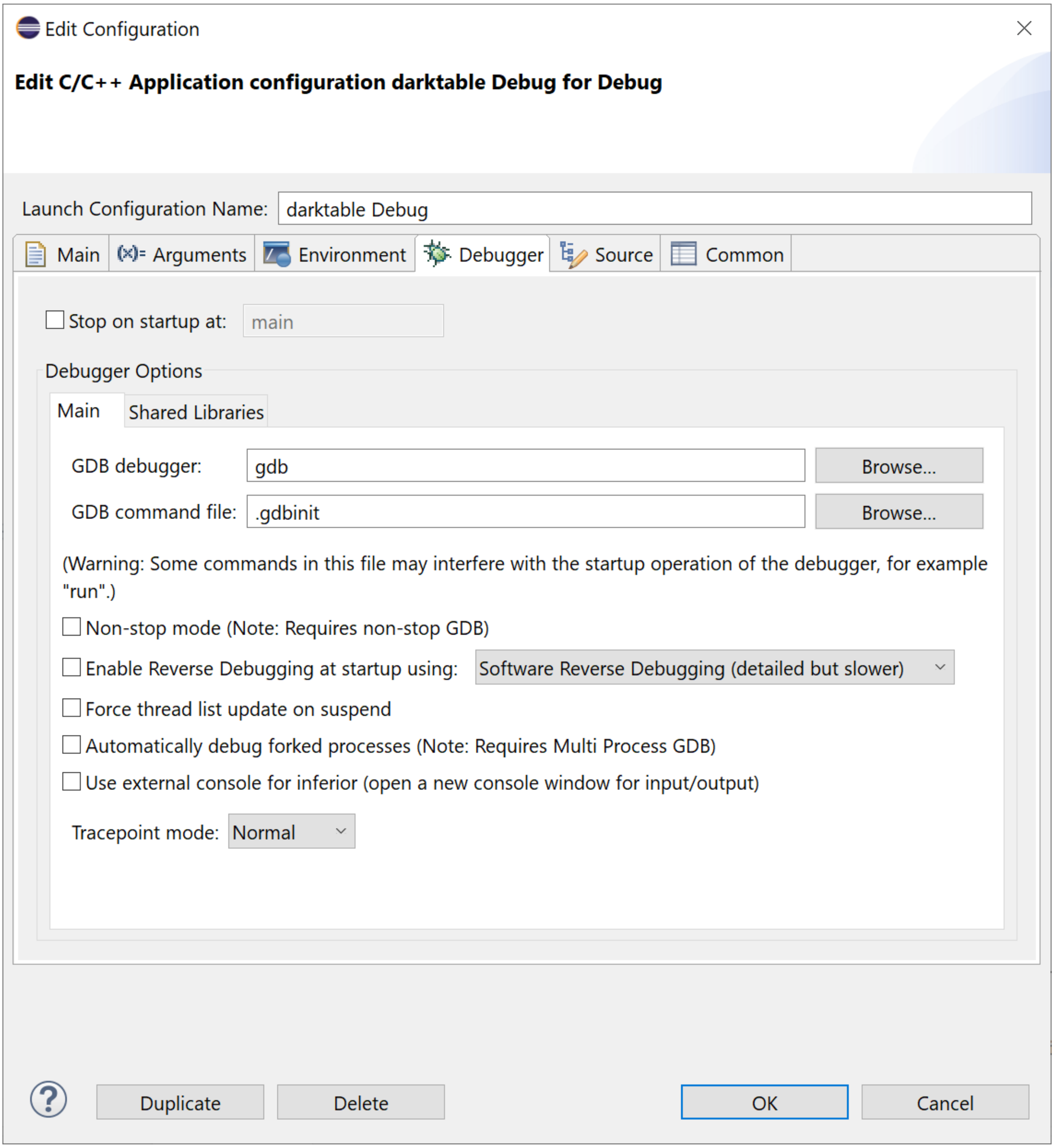
${workspace_loc:darktable}

☑ Use default

(for Linux, use the path /home/<username>/develop/darktable//build/Config1)

In this dialog you can also specify additional command line arguments for darktable, like debugging options (-d <debug option>).

Step 7. Create a debug configuration



Edit Configuration

Edit C/C++ Application configuration darktable Debug for Debug

Launch Configuration Name: darktable Debug

□ Stop on startup at: main

Debugger Options

GDB debugger: gdb

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

□ Non-stop mode (Note: Requires non-stop GDB)

□ Enable Reverse Debugging at startup using: Software Reverse Debugging (detailed but slower)

□ Force thread list update on suspend

□ Automatically debug forked processes (Note: Requires Multi Process GDB)

□ Use external console for inferior (open a new console window for input/output)

Tracepoint mode: Normal

Now Eclipse should be able to run and debug darktable.

Final remarks

- You can define more configurations, like Release for example by changing CMAKE_BUILD_TYPE or other parameters.
- In general, building and debugging in Windows is slower than in Linux.

Feel free to comment / modify / improve.