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

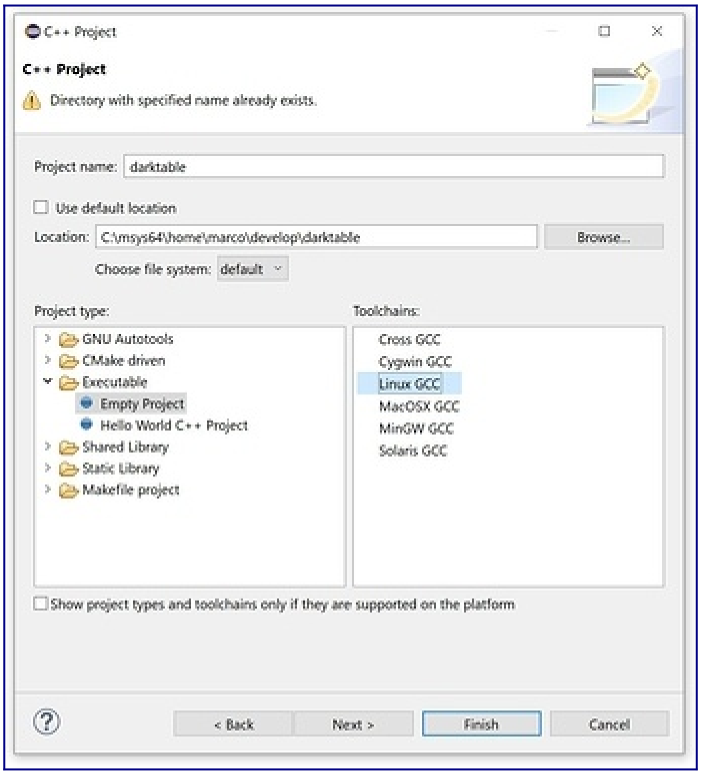
I have been requested to post a guide about setting Eclipse CDT IDE for darktable, so here it is.
This guide assumes that you are already able to build darktable from sources (in Windows through MSYS2), by following the build guide on GitHub.
Basic knowledge of Eclipse CDT is also required.
If you want 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

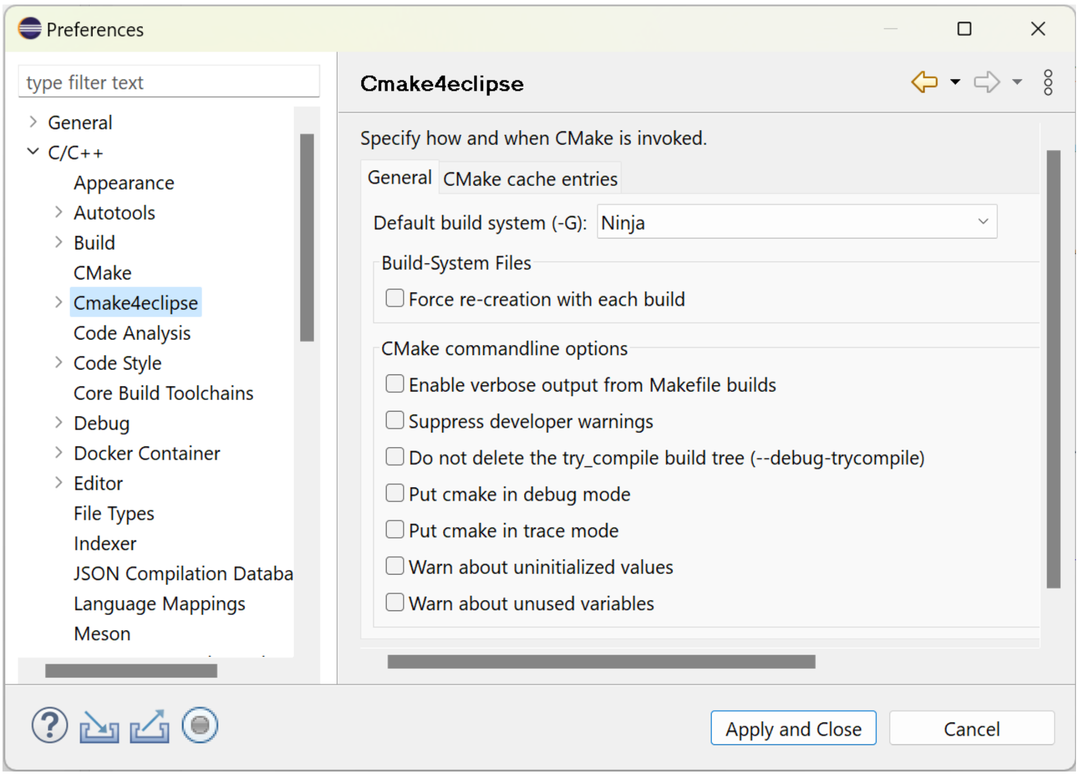
Toolchains: Linux GCC

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

Hit "Finish"

Step 4. Set Cmake4eclipse default builder

Go to Windows → Preferences → C/C++ → Cmake4eclipse, and set the default build system to Ninja



Cmake4eclipse

Specify how and when CMake is invoked.

General | CMake cache entries

Default build system (-G): Ninja

Build-System Files

□ Force re-creation with each build

CMake commandline options

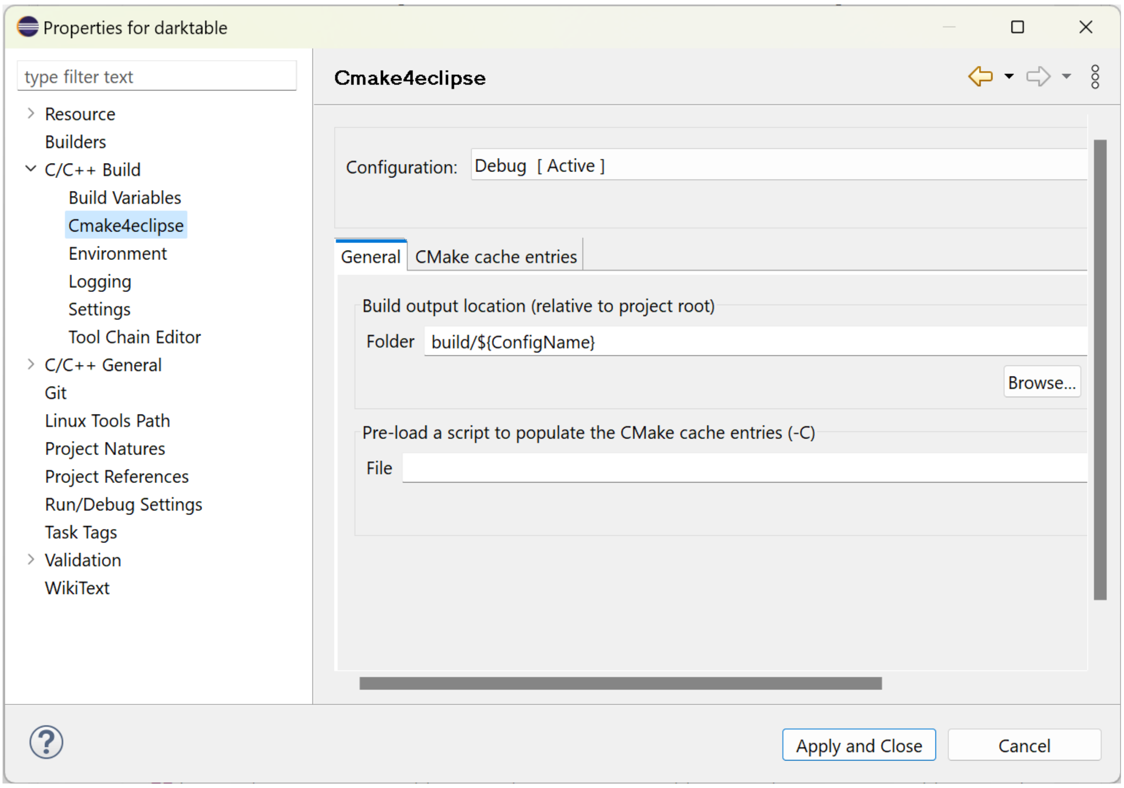
□ Enable verbose output from Makefile builds
□ Suppress developer warnings
□ Do not delete the try_compile build tree (--debug-trycompile)
□ Put cmake in debug mode
□ Put cmake in trace mode
□ Warn about uninitialized values
□ Warn about unused variables

Hit “Apply and Close”

Step 5. 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)



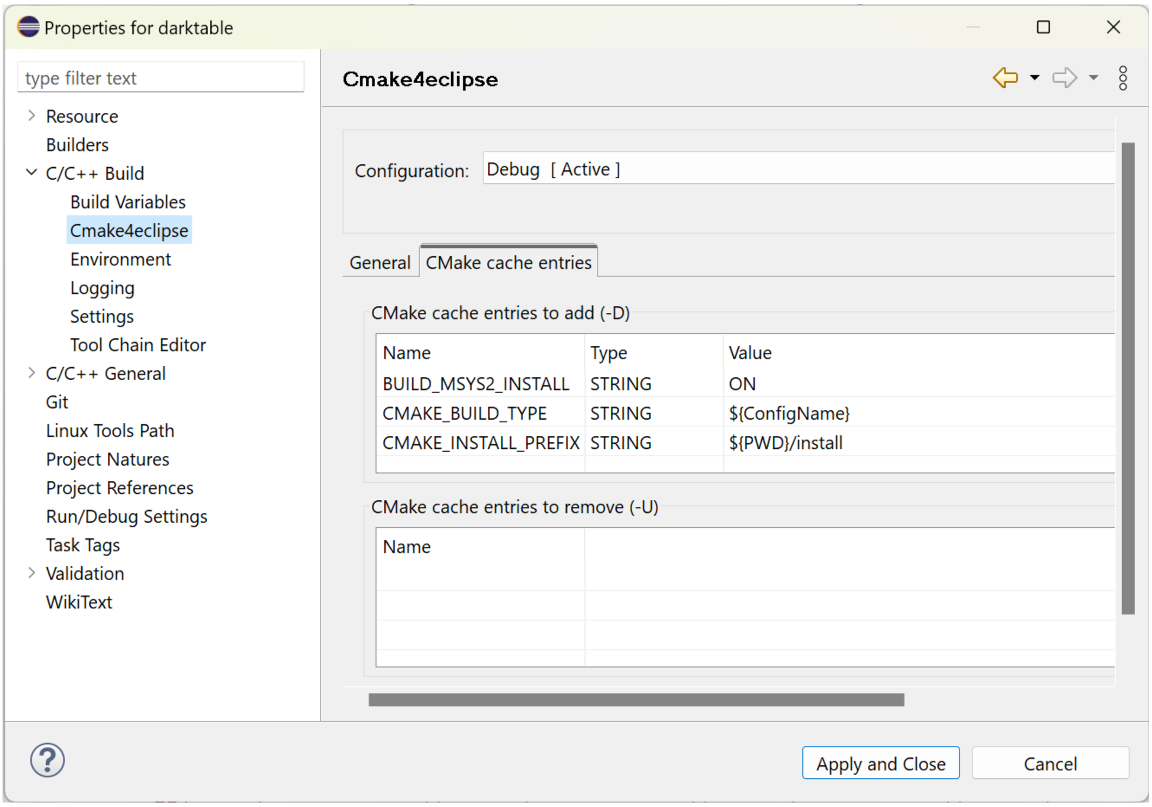
Configuration: Debug [Active]

Build output location (relative to project root)

Folder build/${ConfigName}

Pre-load a script to populate the CMake cache entries (-C)

File



Configuration: Debug [Active]

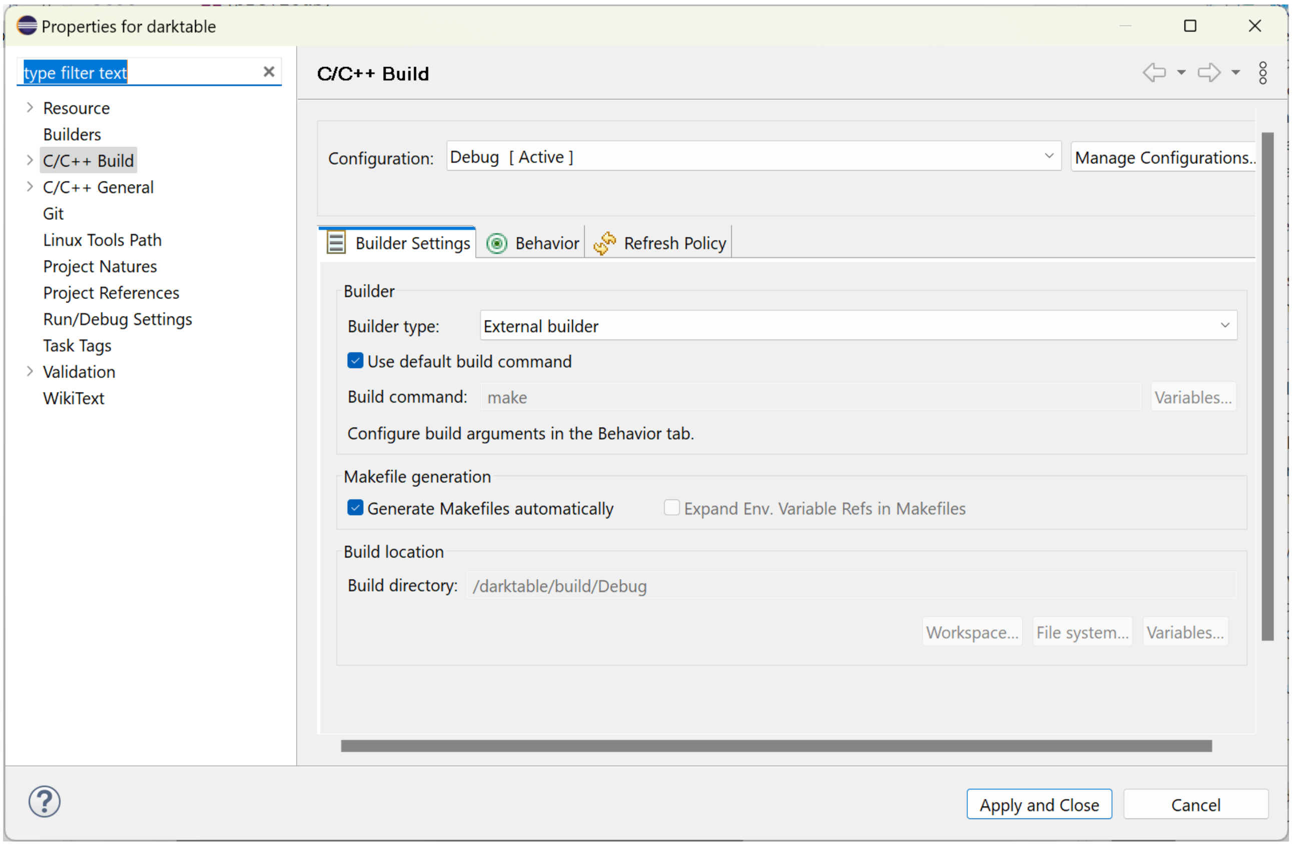
CMake cache entries to add (-D)

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

CMake cache entries to remove (-U)

Name

(in Linux, omit BUILD_MSYS2_INSTALL)



C/C++ Build

Configuration: Debug [Active]

Builder

Builder type: External builder

☑ Use default build command

Build command: make

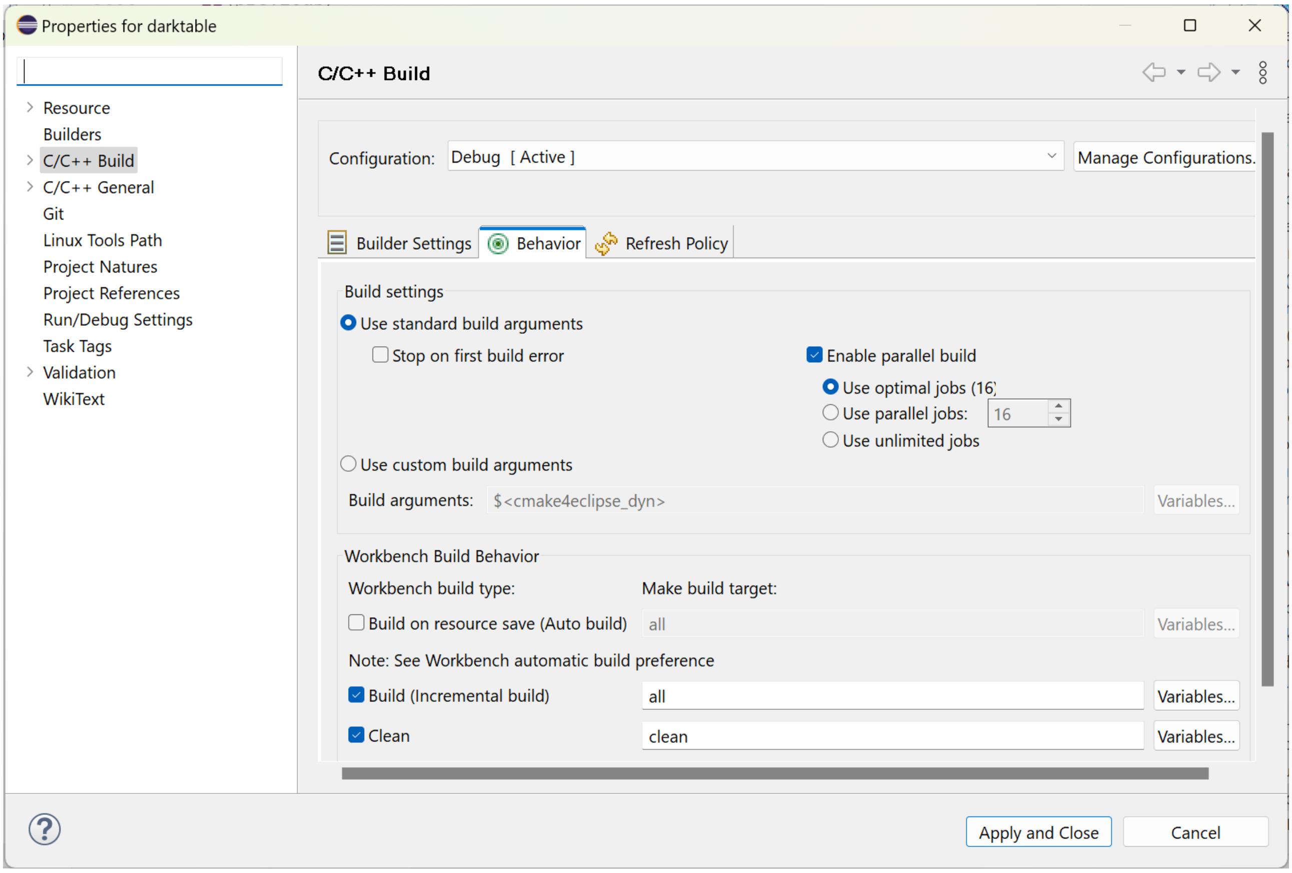
Configure build arguments in the Behavior tab.

Makefile generation

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

Build location

Build directory: /darktable/build/Debug



C/C++ Build

Configuration: Debug [Active]

Build settings

● Use standard build arguments

☐ Stop on first build error ☑ Enable parallel build

● Use optimal jobs (16)

○ Use parallel jobs: 16

○ Use unlimited jobs

○ Use custom build arguments

Build arguments: $<cmake4eclipse_dyn>

Workbench Build Behavior

Workbench build type: Make build target:

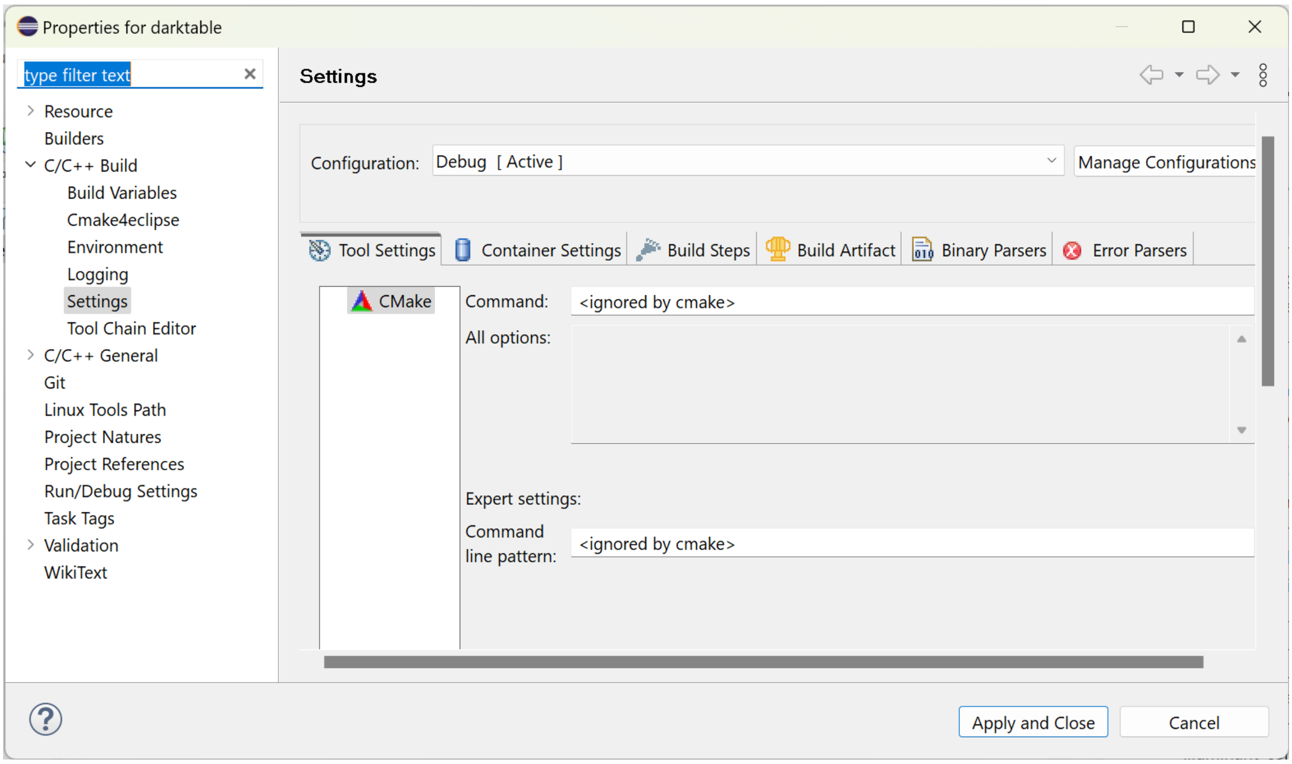
☐ Build on resource save (Auto build) all

Note: See Workbench automatic build preference

☑ Build (Incremental build) all

☑ Clean clean

Specifying “install” instead of “all” as a build target, darktable will be installed in the directory specified by CMAKE_INSTALL_PREFIX



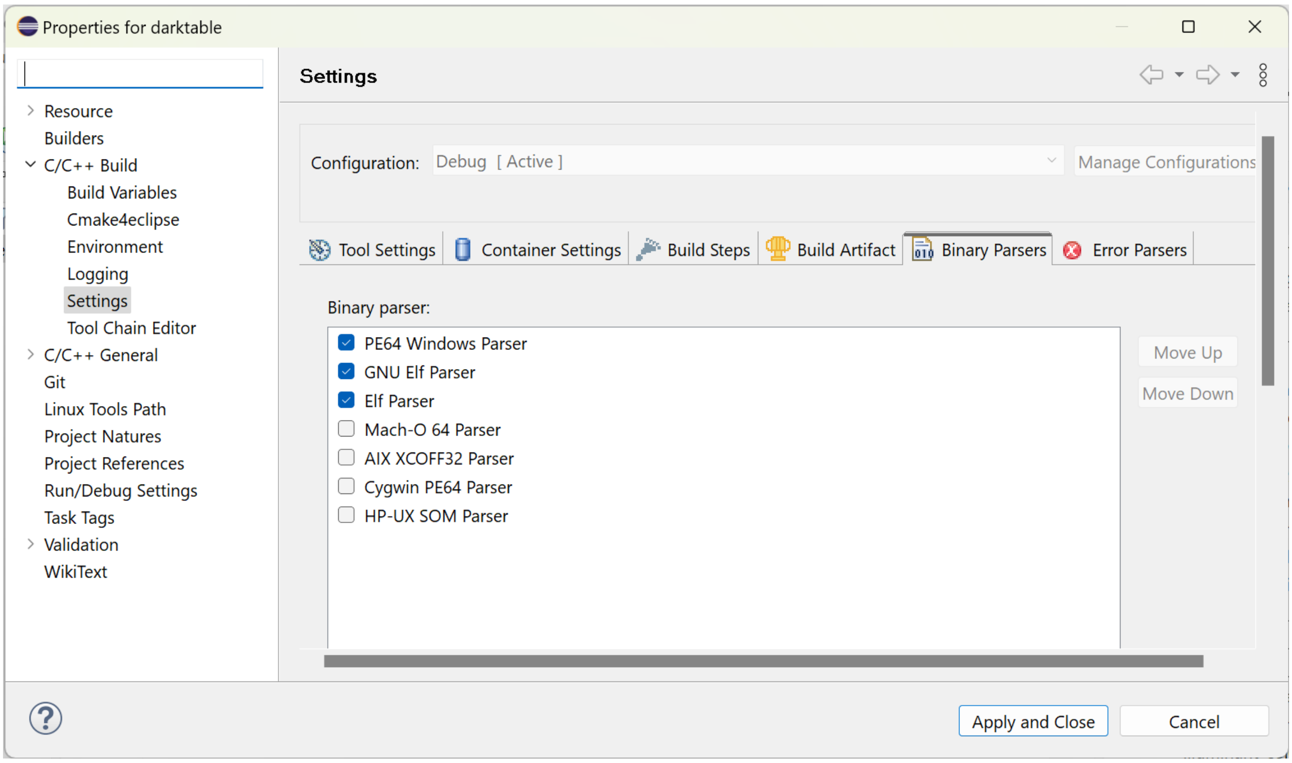
Configuration: Debug [Active]

Command: <ignored by cmake>

All options:

Expert settings:

Command line pattern: <ignored by cmake>

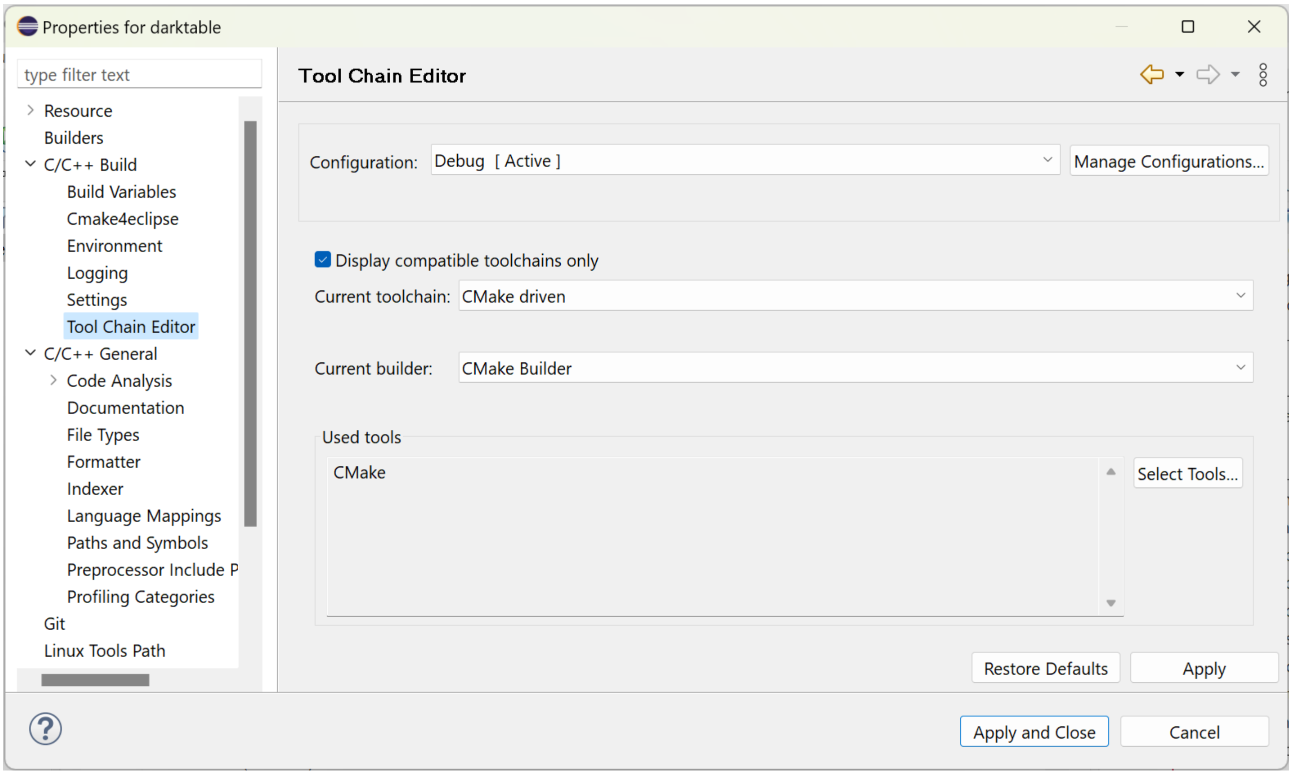


Configuration: Debug [Active]

Binary parser:

- ☑ PE64 Windows Parser
- ☑ GNU Elf Parser
- ☑ Elf Parser
- ☐ Mach-O 64 Parser
- ☐ AIX XCOFF32 Parser
- ☐ Cygwin PE64 Parser
- ☐ HP-UX SOM Parser

(for Linux, just use GNU Elf Parser)



Tool Chain Editor

Configuration: Debug [Active]

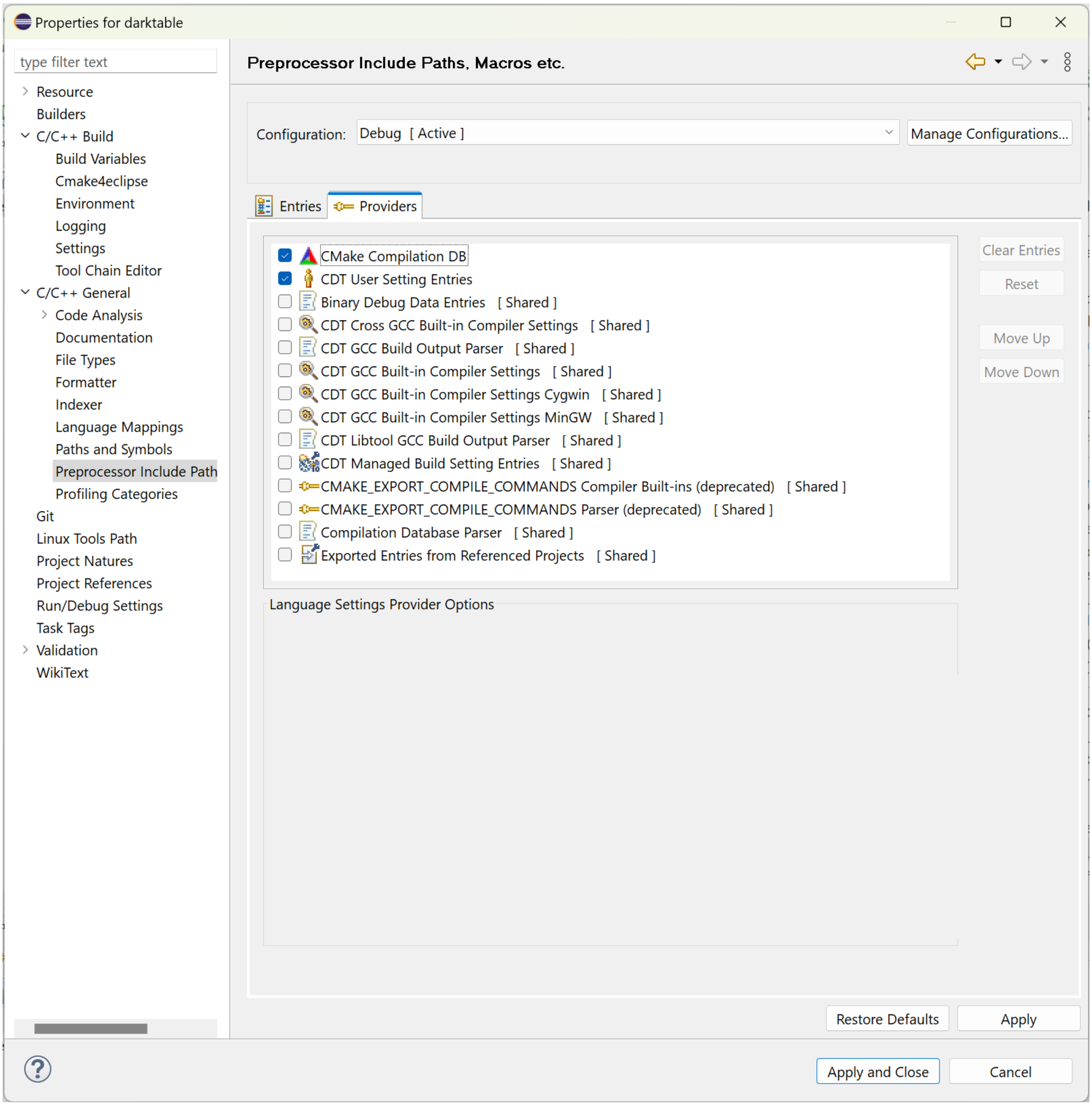
☑ Display compatible toolchains only

Current toolchain: CMake driven

Current builder: CMake Builder

Used tools

CMake

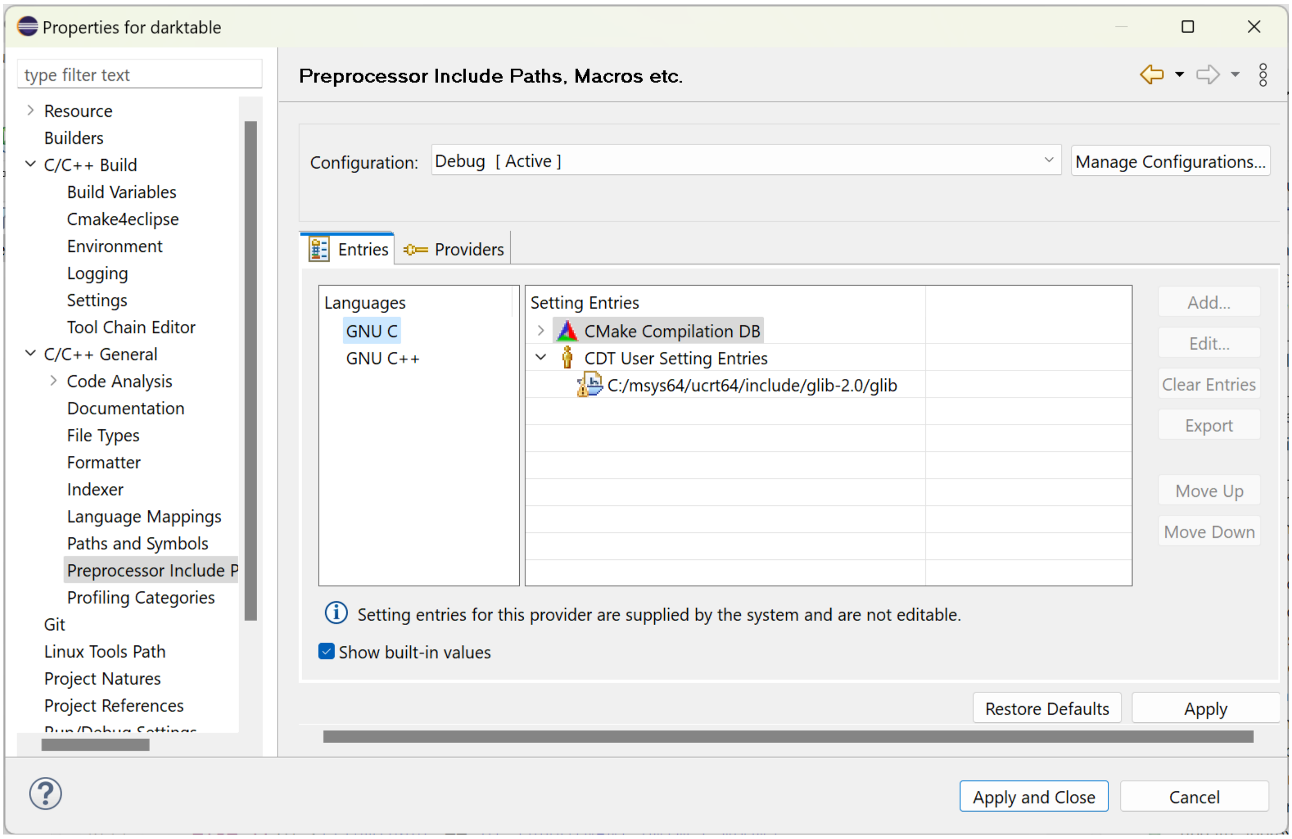


Preprocessor Include Paths, Macros etc.

Configuration: Debug [Active]

☑ CMake Compilation DB
☑ CDT User Setting Entries
☐ 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]
☐ CMAKE_EXPORT_COMPILE_COMMANDS Compiler Built-ins (deprecated) [Shared]
☐ CMAKE_EXPORT_COMPILE_COMMANDS Parser (deprecated) [Shared]
☐ Compilation Database Parser [Shared]
☐ Exported Entries from Referenced Projects [Shared]

Language Settings Provider Options



Preprocessor Include Paths, Macros etc.

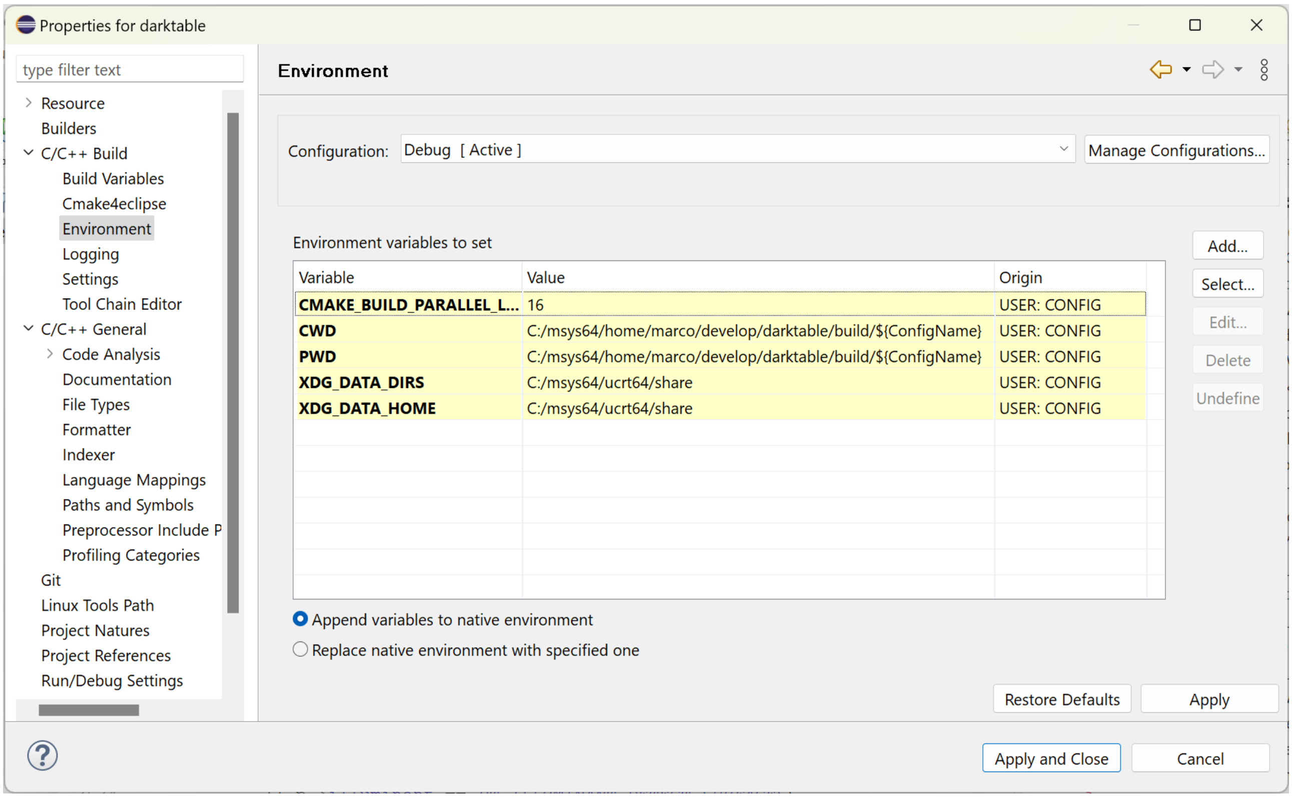
Configuration: Debug [Active] Manage Configurations...

Languages:
- GNU C
- GNU C++

Setting Entries:
- CMake Compilation DB
- CDT User Setting Entries
  - C:/msys64/ucrt64/include/glib-2.0/glib

Setting entries for this provider are supplied by the system and are not editable.

☑ Show built-in values



Environment

Configuration: Debug [Active] Manage Configurations...

Environment variables to set

Variable	Value	Origin
CMAKE_BUILD_PARALLEL_L...	16	USER: CONFIG
CWD	C:/msys64/home/marco/develop/darktable/build/${ConfigName}	USER: CONFIG
PWD	C:/msys64/home/marco/develop/darktable/build/${ConfigName}	USER: CONFIG
XDG_DATA_DIRS	C:/msys64/ucrt64/share	USER: CONFIG
XDG_DATA_HOME	C:/msys64/ucrt64/share	USER: CONFIG

● Append variables to native environment
○ Replace native environment with specified one

(for Linux, use the corresponding path /home/<username>/develop/darktable/build/${ConfigName} and omit XDG_DATA_DIRS and XDG_DATA_HOME)

Hit “Apply and Close”

Step 6. Build the project

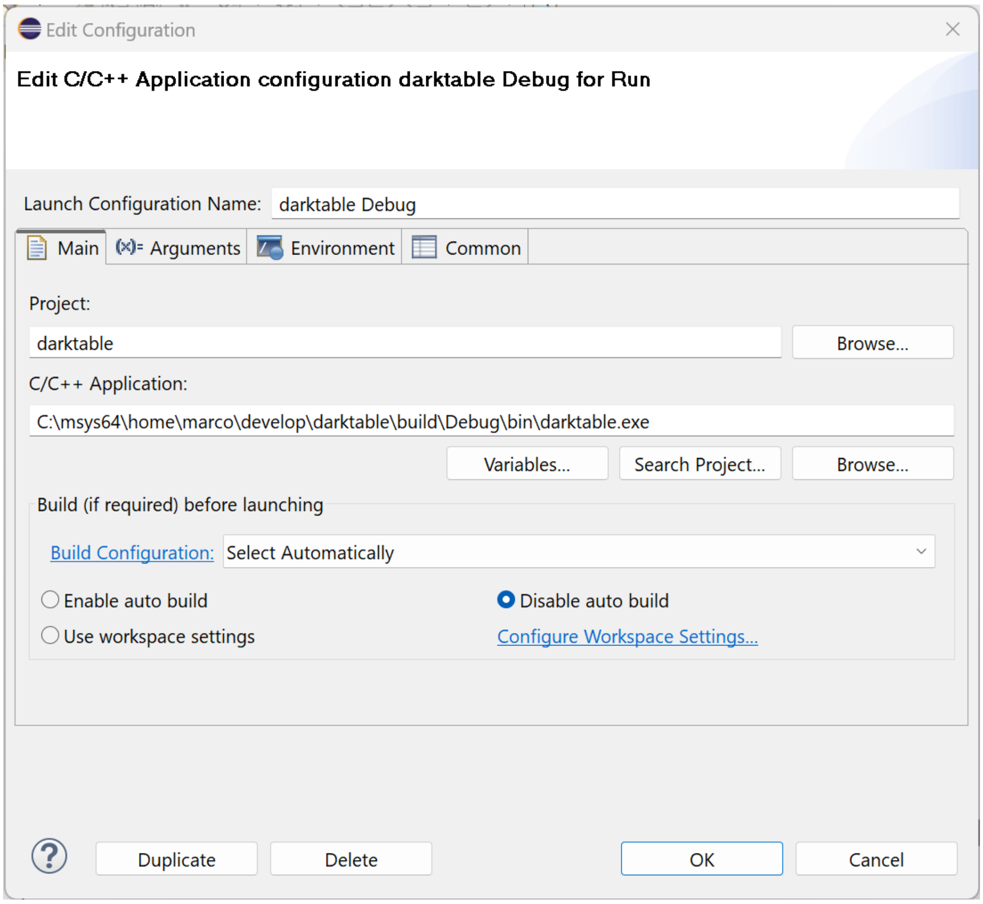
Eclipse should run CMake first and then Ninjia, you can see the output in the Eclipse console.

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 7. Create a run configuration

a) Run in place



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\bin\darktable.exe

Build (if required) before launching

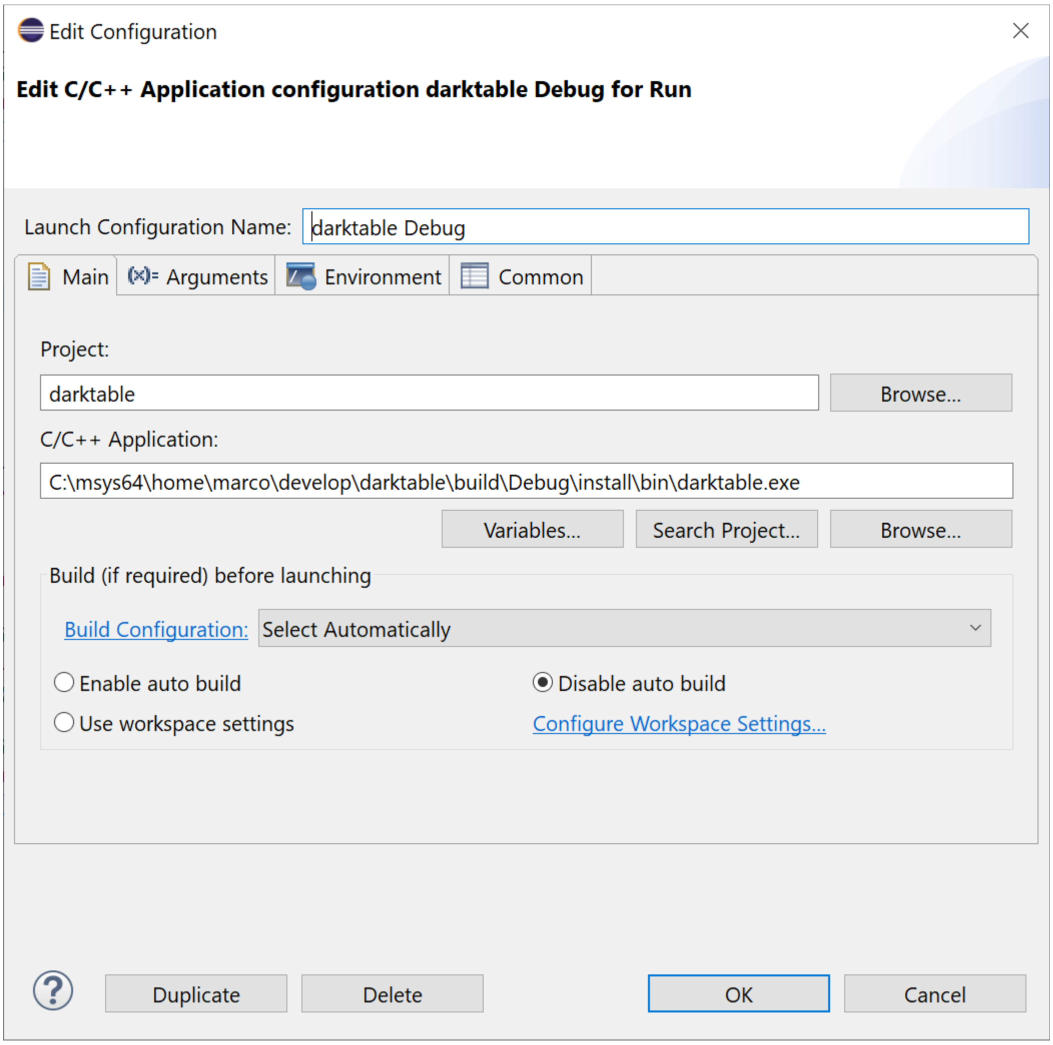
Build Configuration: Select Automatically

○ Enable auto build ● Disable auto build

○ Use workspace settings

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

b) Run an installed build



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

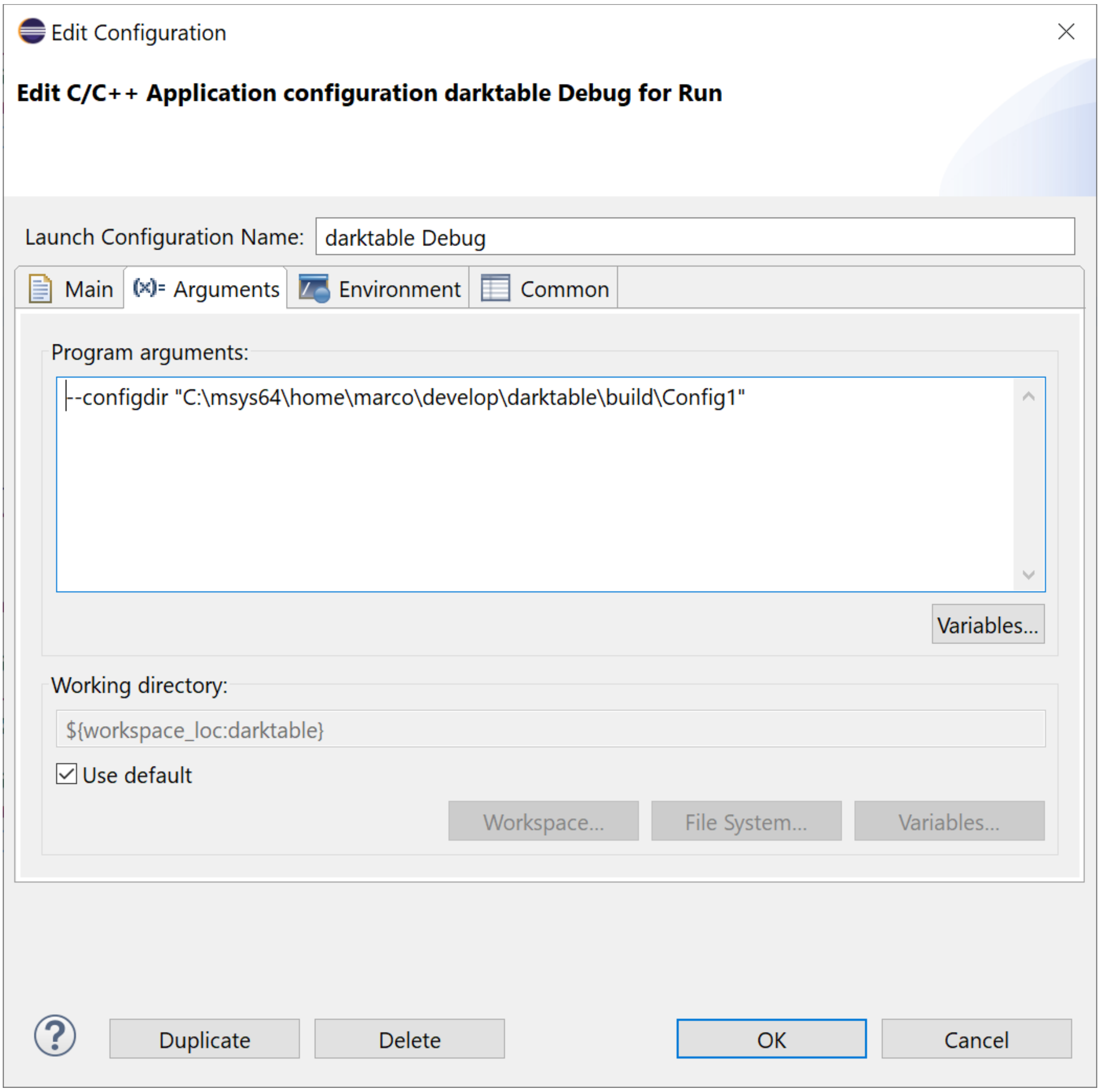
Build Configuration: Select Automatically

○ 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"
```

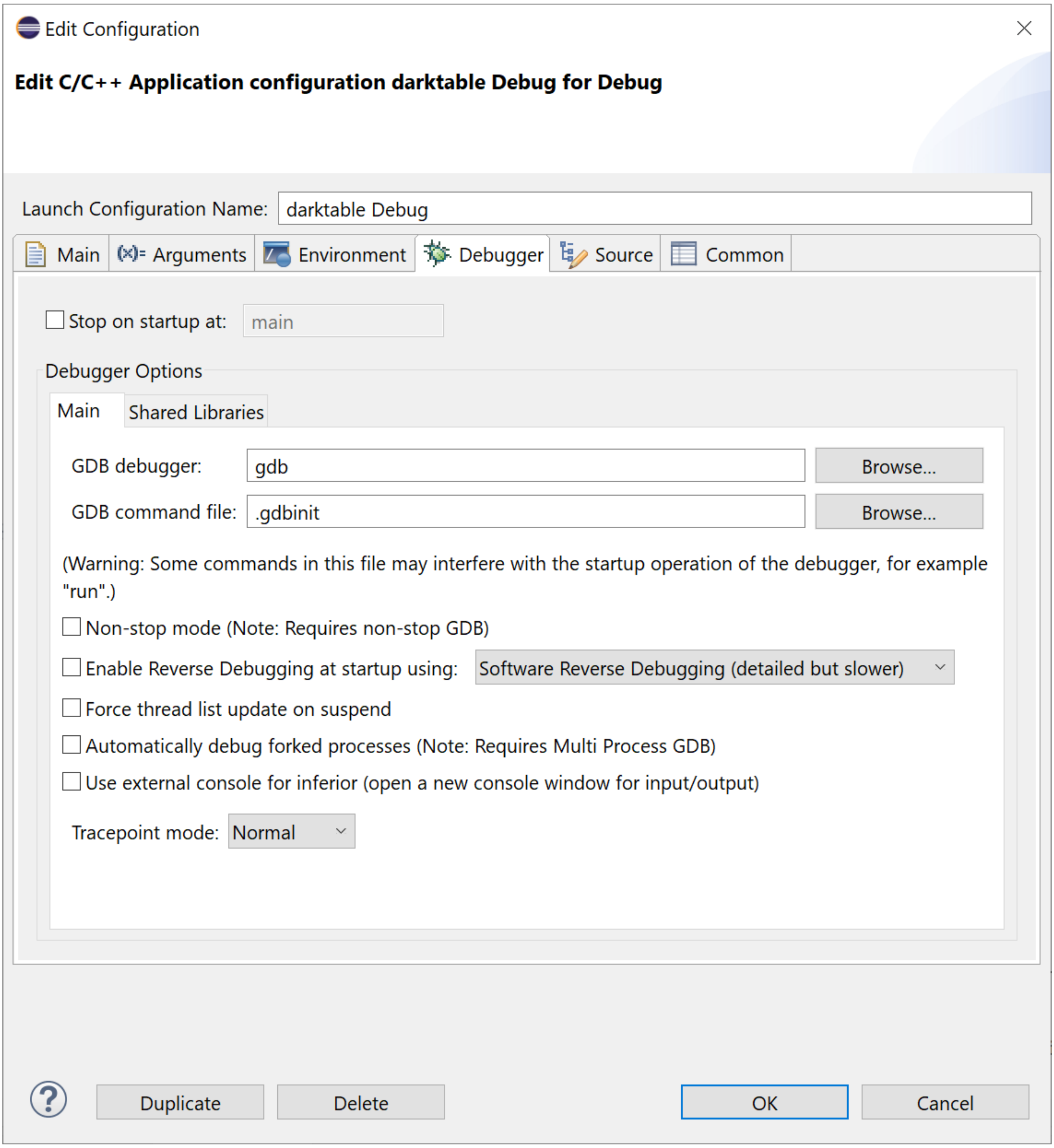
Working directory: ${workspace_loc:darktable}

☑ 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 build configurations, like Release or RelWithDebInfo, with the same parameters.
- In general, building and debugging in Windows is slower than in Linux.

Feel free to comment / modify / improve.