Configuring MSVC2019 in Code::Blocks

A quick summary on how to use the

Microsoft Visual Studio 2019 Enterprise compiler (MSVC2019)

in Code::Blocks

Carsten Arnholm - January 2020

https://github.com/arnholm/cpde_utils

Table of Contents

Introduction

The basic idea is to let Code::Blocks project files be independent of the MSVC compiler version. We therefore define a user defined compiler in Code::Blocks and call in ‘MSVC’. All project files refer to this compiler only.

Also, we define the MSVC compiler in terms of Code::Blocks global variables with user defined fields. An overview is given in the table below.

Code::Blocks Global variable	Description
MSVC	A set of user defined fields used by other global fields. This reduces the changes between the variable sets to a minimum
MSVC_TOOLCHAIN	Defines MSVC, IDE, compiler, linker and related tools
NET_SDK	Defines .NET SDK include and library paths
WIN_SDK	Defines Windows SDK include paths as well as binary path (binary path not used for now).
WIN_SDK_LIB	Defines Windows SDK library paths

Global variable settings

MSVC

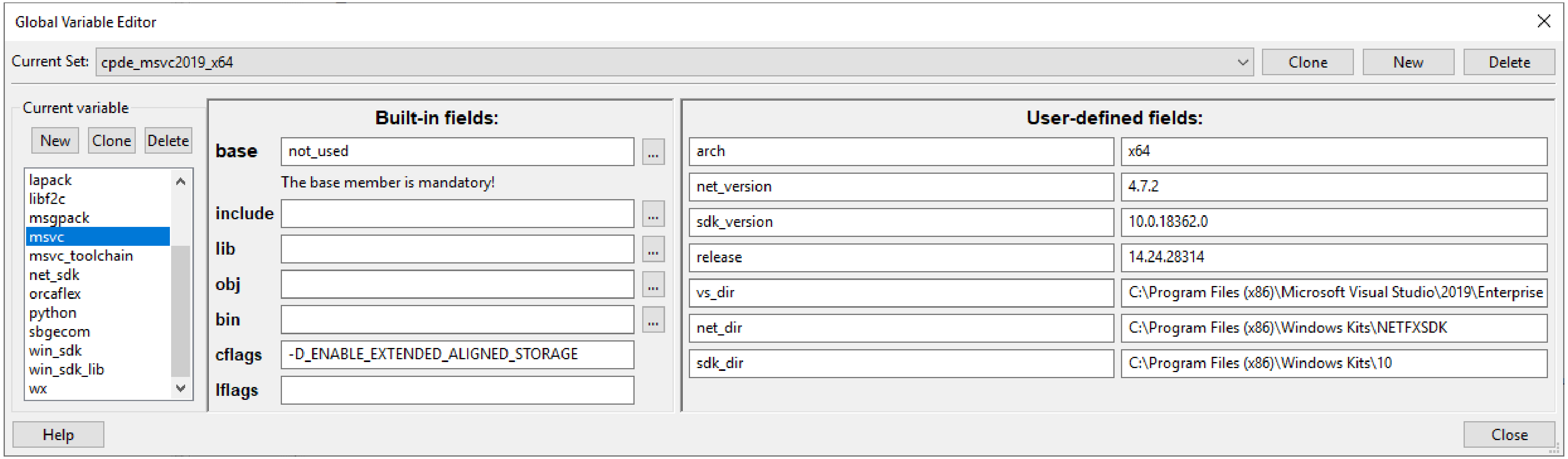
The ‘release’ field corresponds to the currently installed Visual studio version, as identified in e.g.

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.24.28314
```

The ‘sdk_version’ field corresponds to the “Windows Kits” version you are using, e.g.

```
C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0
```

The ‘arch’ field should be ‘x64’ for 64-bit or ‘x86’ for 32-bit



Current Set: cpde_msvc2019_x64

Built-in fields:

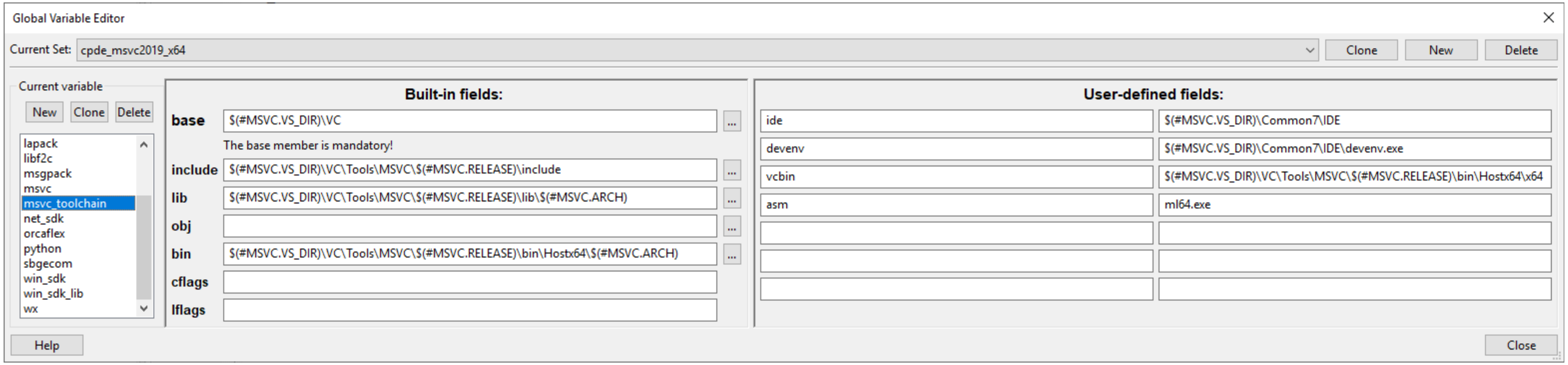
base	not_used
	The base member is mandatory!
include	
lib	
obj	
bin	
cflags	-D_ENABLE_EXTENDED_ALIGNED_STORAGE
lflags	

User-defined fields:

arch	x64
net_version	4.7.2
sdk_version	10.0.18362.0
release	14.24.28314
vs_dir	C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise
net_dir	C:\Program Files (x86)\Windows Kits\NETFXSDK
sdk_dir	C:\Program Files (x86)\Windows Kits\10

MSVC_TOOLCHAIN

Defines MSVC, IDE, compiler, linker and related tools



Current Set: cpde_msvc2019_x64

lapack
libf2c
msgpack
msvc
msvc_toolchain
net_sdk
orcaflex
python
sbgecom
win_sdk
win_sdk_lib
wx

Built-in fields:

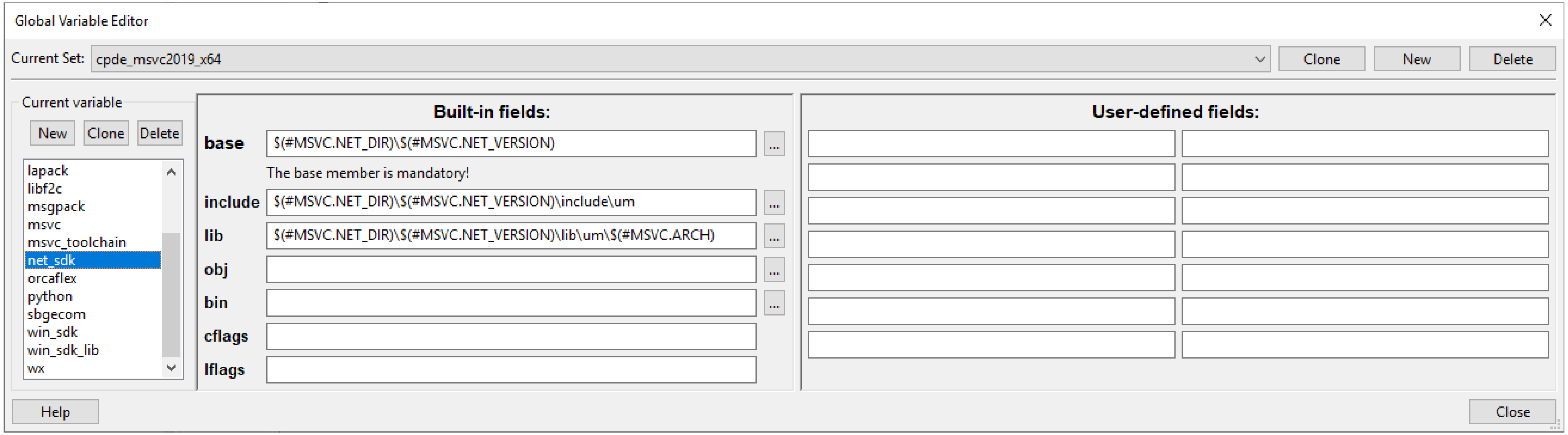
base	$(#MSVC.VS_DIR)\VC
	The base member is mandatory!
include	$(#MSVC.VS_DIR)\VC\Tools\MSVC\$(#MSVC.RELEASE)\include
lib	$(#MSVC.VS_DIR)\VC\Tools\MSVC\$(#MSVC.RELEASE)\lib\$(#MSVC.ARCH)
obj	
bin	$(#MSVC.VS_DIR)\VC\Tools\MSVC\$(#MSVC.RELEASE)\bin\Hostx64\$(#MSVC.ARCH)
cflags	
lflags	

User-defined fields:

ide	$(#MSVC.VS_DIR)\Common7\IDE
devenv	$(#MSVC.VS_DIR)\Common7\IDE\devenv.exe
vcbin	$(#MSVC.VS_DIR)\VC\Tools\MSVC\$(#MSVC.RELEASE)\bin\Hostx64\x64
asm	ml64.exe

NET_SDK

Defines .NET SDK include and library paths. Not required for native C++ only.



Global Variable Editor

Current Set: cpde_msvc2019_x64

Current variable: net_sdk

Built-in fields:

base: $(#MSVC.NET_DIR)\$(#MSVC.NET_VERSION)

The base member is mandatory!

include: $(#MSVC.NET_DIR)\$(#MSVC.NET_VERSION)\include\um

lib: $(#MSVC.NET_DIR)\$(#MSVC.NET_VERSION)\lib\um\$(#MSVC.ARCH)

obj:

bin:

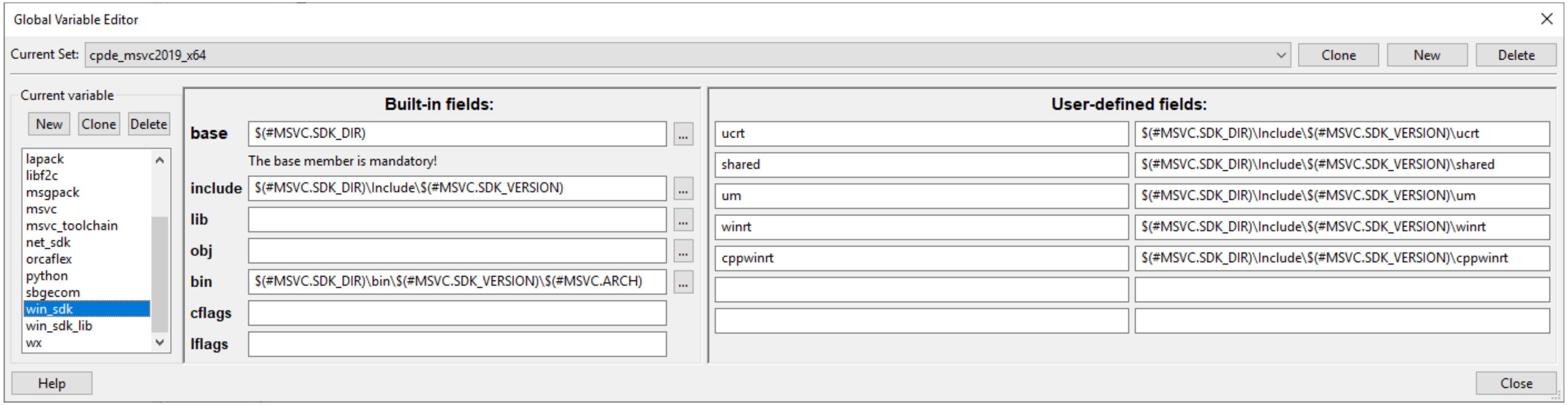
cflags:

lflags:

User-defined fields:

WIN_SDK

Defines Windows SDK include paths as well as binary path (binary path not used for now).



Current Set: cpde_msvc2019_x64

Built-in fields:

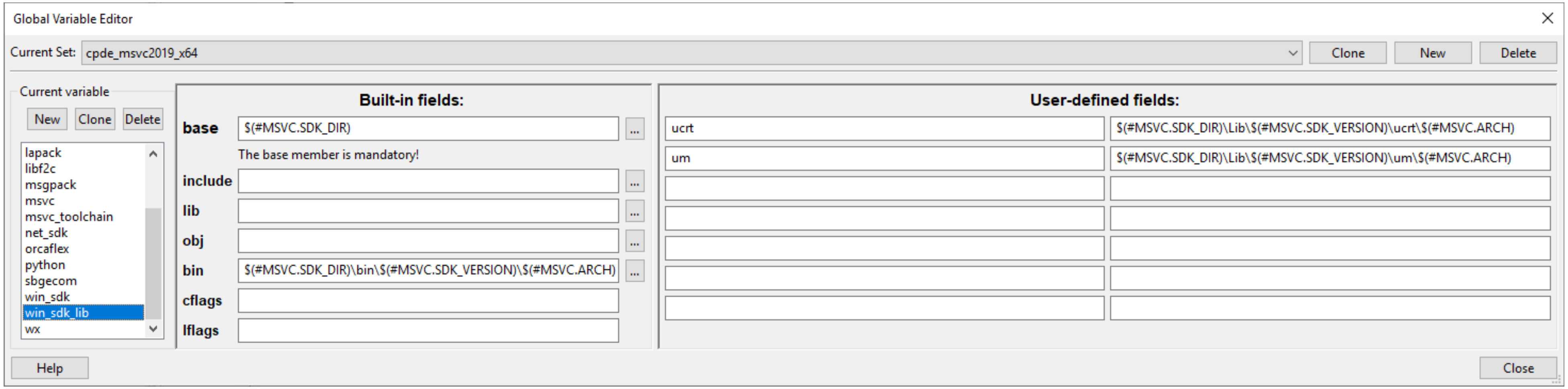
base	$(#MSVC.SDK_DIR)
	The base member is mandatory!
include	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)
lib	
obj	
bin	$(#MSVC.SDK_DIR)\bin\$(#MSVC.SDK_VERSION)\$(#MSVC.ARCH)
cflags	
lflags	

User-defined fields:

ucrt	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)\ucrt
shared	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)\shared
um	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)\um
winrt	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)\winrt
cppwinrt	$(#MSVC.SDK_DIR)\Include\$(#MSVC.SDK_VERSION)\cppwinrt

WIN_SDK_LIB

Defines Windows SDK library paths



Current Set: cpde_msvc2019_x64

Built-in fields:

base	$(#MSVC.SDK_DIR)
include	
lib	
obj	
bin	$(#MSVC.SDK_DIR)\bin\$(#MSVC.SDK_VERSION)\$(#MSVC.ARCH)
cflags	
lflags	

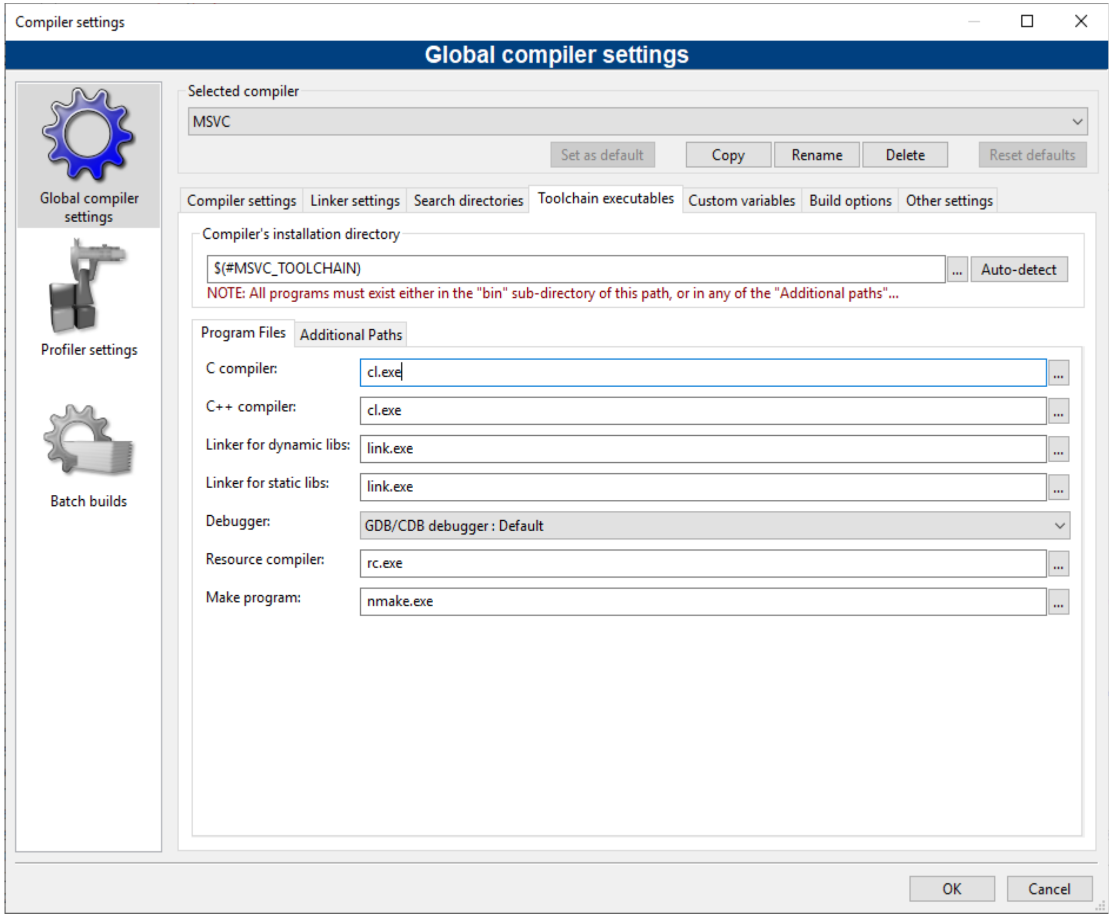
The base member is mandatory!

User-defined fields:

ucrt	$(#MSVC.SDK_DIR)\Lib\$(#MSVC.SDK_VERSION)\ucrt\$(#MSVC.ARCH)
um	$(#MSVC.SDK_DIR)\Lib\$(#MSVC.SDK_VERSION)\um\$(#MSVC.ARCH)

Compiler settings

The following pages show the settings required to define the MSVC user defined compiler, based on the global variable settings.



Selected compiler: MSVC

Compiler's installation directory: $(#MSVC_TOOLCHAIN)

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional paths"...

C compiler: cl.exe

C++ compiler: cl.exe

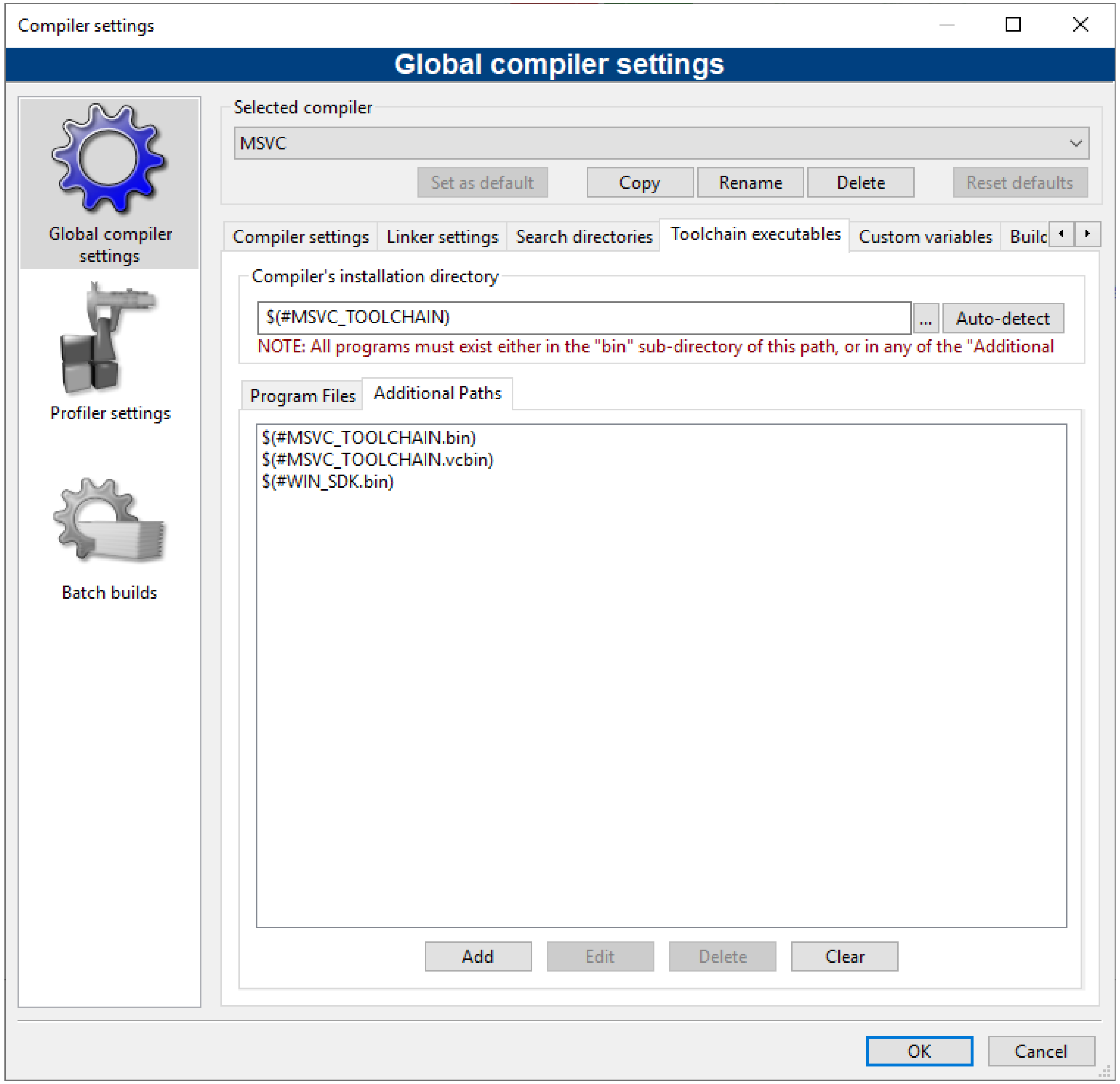
Linker for dynamic libs: link.exe

Linker for static libs: link.exe

Debugger: GDB/CDB debugger : Default

Resource compiler: rc.exe

Make program: nmake.exe

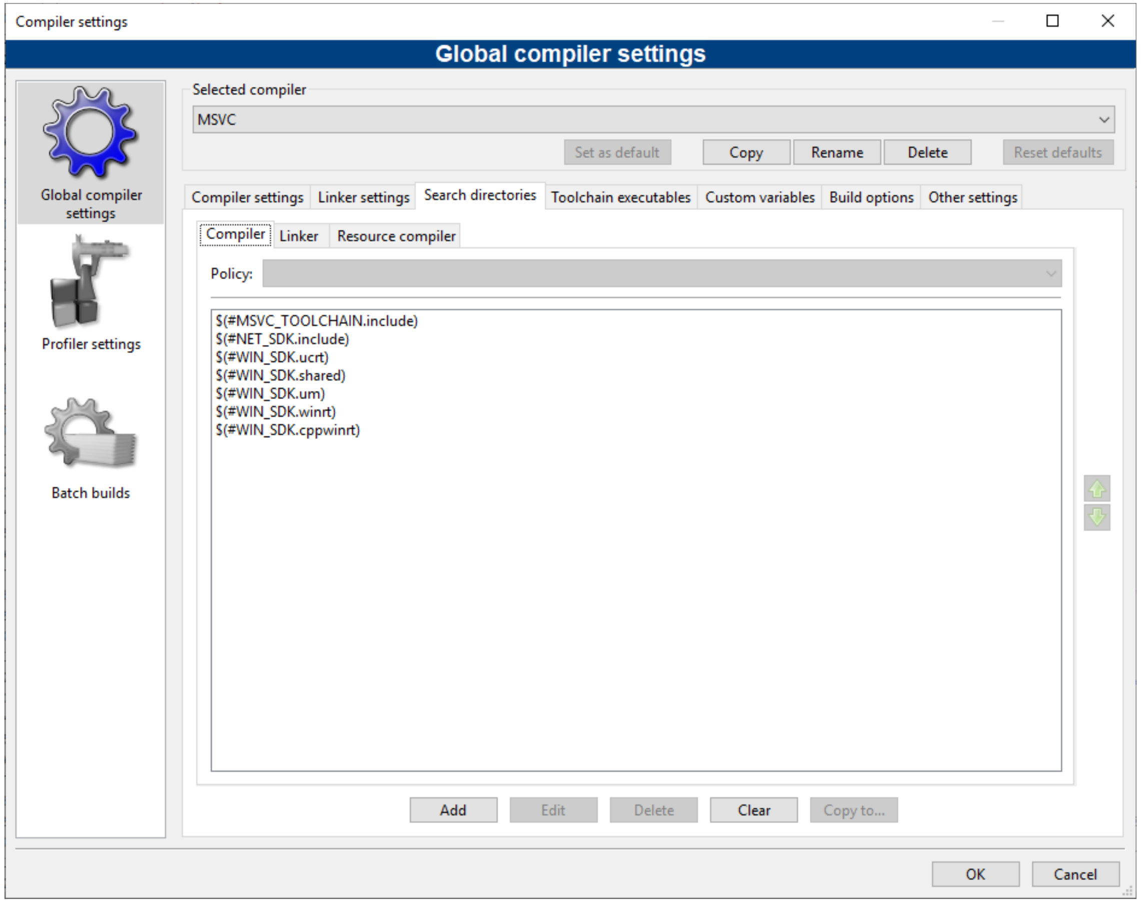


Selected compiler: MSVC

Compiler's installation directory: $(#MSVC_TOOLCHAIN)

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional

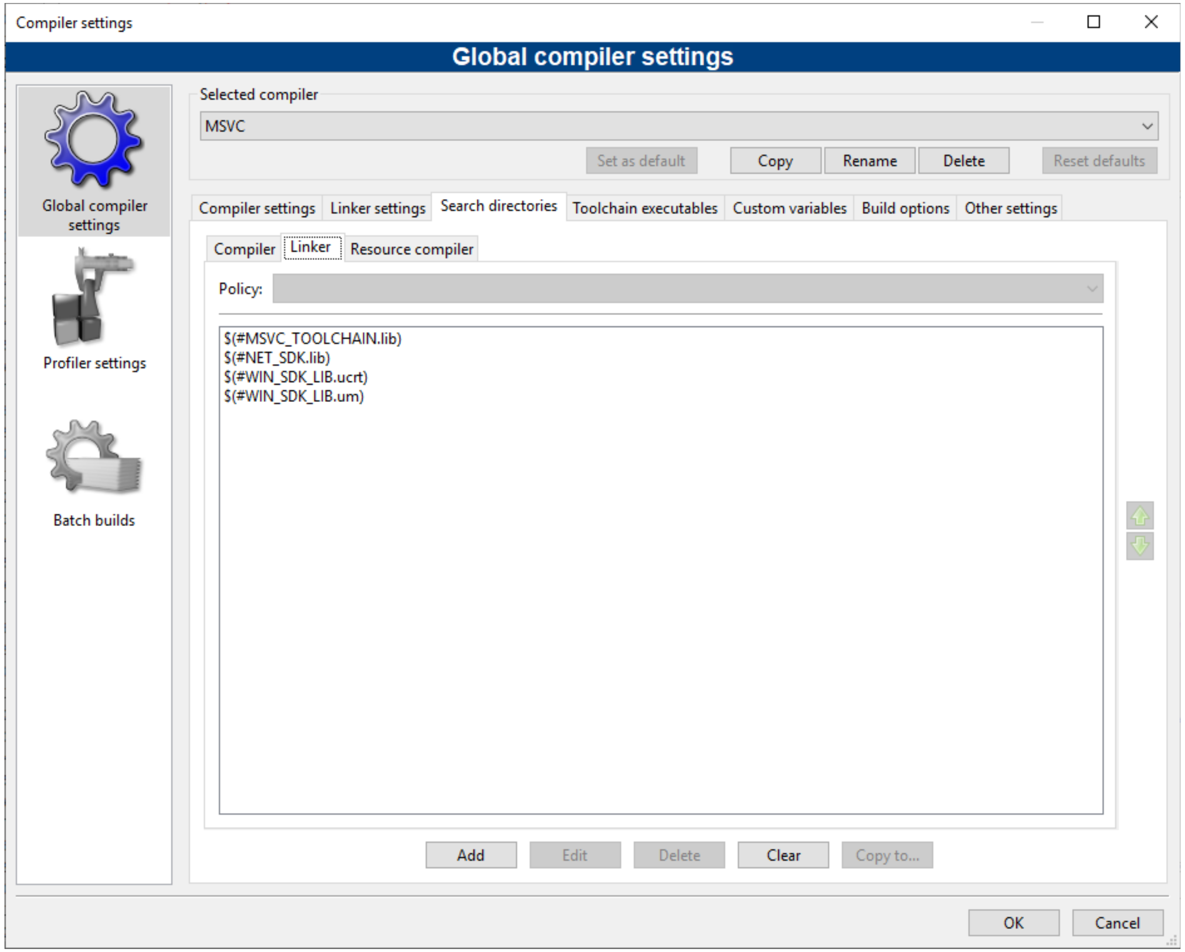
```
$(#MSVC_TOOLCHAIN.bin)
$(#MSVC_TOOLCHAIN.vcbin)
$(#WIN_SDK.bin)
```



Selected compiler: MSVC

Policy:

```
$(#MSVC_TOOLCHAIN.include)
$(#NET_SDK.include)
$(#WIN_SDK.ucrt)
$(#WIN_SDK.shared)
$(#WIN_SDK.um)
$(#WIN_SDK.winrt)
$(#WIN_SDK.cppwinrt)
```



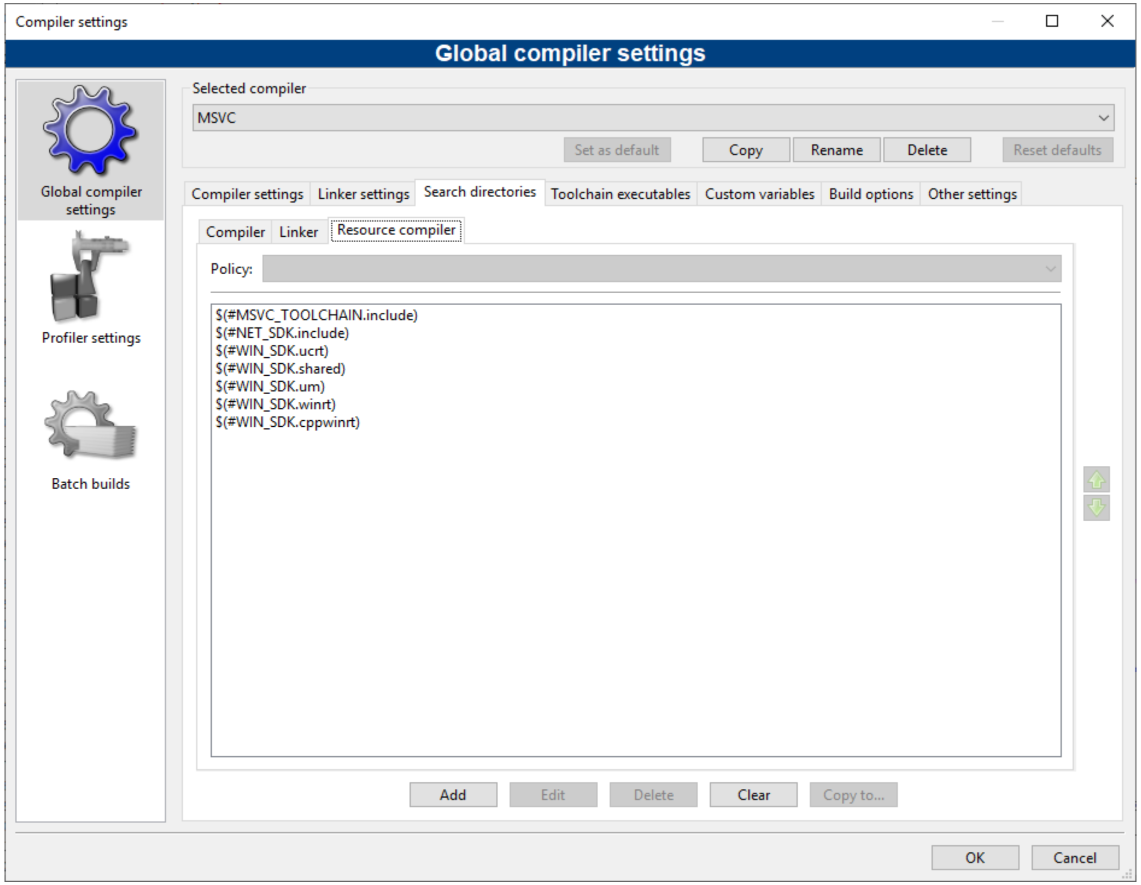
Compiler settings

Global compiler settings

Selected compiler: MSVC

Policy:

```
$(#MSVC_TOOLCHAIN.lib)
$(#NET_SDK.lib)
$(#WIN_SDK_LIB.ucrt)
$(#WIN_SDK_LIB.um)
```



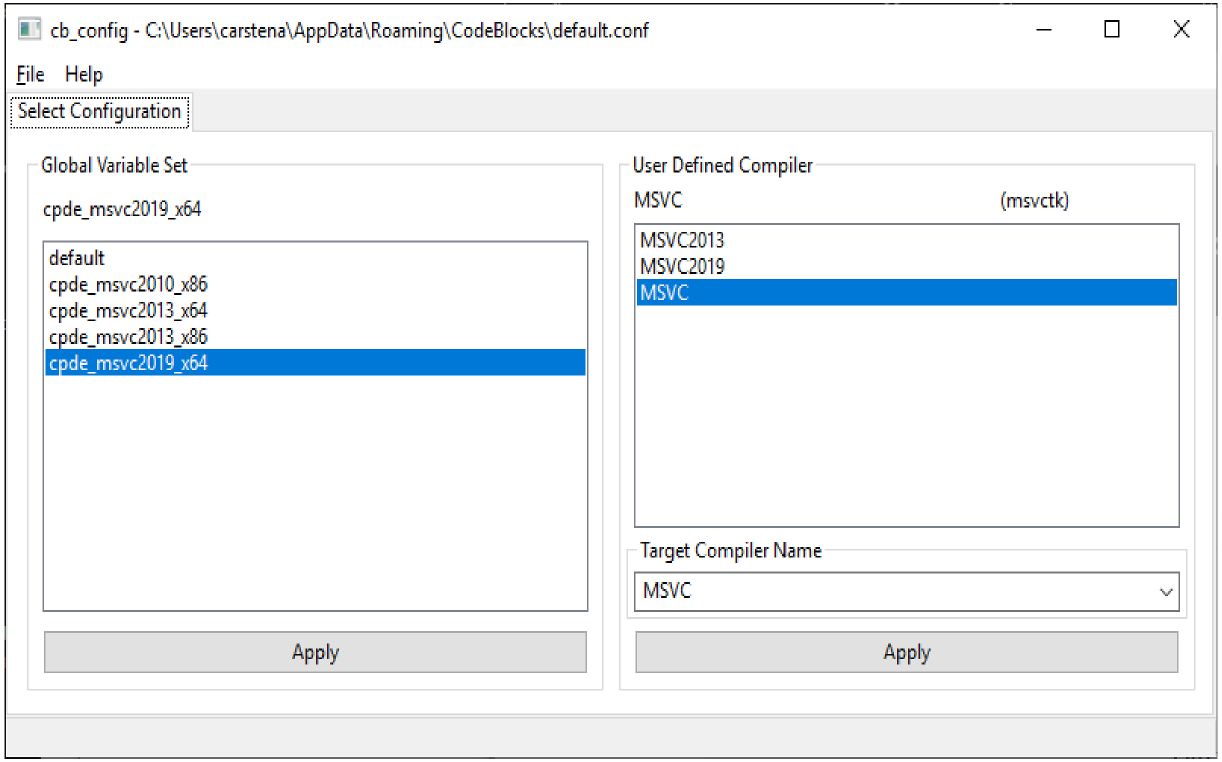
Selected compiler: MSVC

```
$(#MSVC_TOOLCHAIN.include)
$(#NET_SDK.include)
$(#WIN_SDK.ucrt)
$(#WIN_SDK.shared)
$(#WIN_SDK.um)
$(#WIN_SDK.winrt)
$(#WIN_SDK.cppwinrt)
```

Switching between MSVC compilers

Sometimes it is required to have more than one MSVC compiler installed. In order to switch between such compilers it is typically required to have multiple global variable sets and also multiple compiler definitions. In Code::Blocks, you can select a global variable set, but redefining the MSVC compiler is cumbersome. Therefore a tool (cb_config, part of cpde_utils) has been developed that modifies the Code::Blocks config.conf file to do this more easily (it requires all instances of Code::blocks to be closed before it is used).

For example, you can have MSVC2013 and MSVC2019 installed at the same time, with corresponding global variable sets for each compiler. Switching between compilers means to redefine the MSVC compiler based on MSVC2013 or MSVC2019. As the two compilers are very different it is usually required to use different global variable sets. After selecting and saving (File → Save), Code::Blocks is configured accordingly on next startup.



cb_config - C:\Users\carstena\AppData\Roaming\CodeBlocks\default.conf

Global Variable Set: cpde_msvc2019_x64

default
cpde_msvc2010_x86
cpde_msvc2013_x64
cpde_msvc2013_x86
cpde_msvc2019_x64

User Defined Compiler: MSVC (msvctk)

MSVC2013
MSVC2019
MSVC

Target Compiler Name: MSVC