How config Pelles C Compiler (32 and 64 bits) into CodeBlocks

Full name of tutorial : How config Pelles C Compiler (32 and 64 bits) into Code::Blocks on Windows 11 64 bits.

```
Code::Blocks : the best and great free IDE for Windows, Linux and ... Mac OS
```

Presentation of Pelles C Compiler

During first run of CB on Windows, this IDE detect automatically some compilers, or present one list of them pre-configured.
It's very good functionnality, but, sometimes, you must "force" these configurations proposed by default to run correctly.
This tuto describe how configure one compiler C on Windows : Pelles C Compiler (version 32 and 64 bits) that is not detected automatically by CB ...

Pelles C is a complete development kit for Desktop Windows. It contains among other things an optimizing C compiler, a macro assembler, a linker, a resource compiler, a message compiler, **a code signing utility**, a make utility, an debbuger and an install builder.
Last version of this compiler support C99/C11/C17/C2x recommandations of C. Pelles Orinius is creator and major maintener of this fork of LCC compiler (by Chris Fraser and David Hanson).

How to install Pelles C Compiler (32 and 64 bits) ?

You can download it from Internet site : http://www.smorgasbordet.com/pellesc/

Then, you select link "setup" and donwload file "setup_pellesC_v12.exe" (version v12 dated may 2023, maybe v13 appear soon !)
After, click on this file to install Pelles C compiler on "C:\PellesC" directory (by default), with no forget to select before "Add-in SDK" in choices proposed.

Configuration of Pelles C Compiler into Code::Blocks

Normally, after that, next run of CB detect presence of these compilers and proposed it in list of available compiler in main menu "Settings" and after submenu "Compiler...", but in these case,

no automatic detection... Then how configure this ?
Arbitrarily, you can choose into list proposed by CB, "Microsoft Visual C++ 2022" compiler because Pelles C is very closed to syntax of MSVC, and select "Copy" button, and after "Rename" button to change reference of new compiler like "Pelles C (32b)" by example.

Then you must force many parameters positioned by default with choice of initial compiler "Microsoft Visual C++ 2022".

First, in tab "Toolchain executable", you must write in field "Compiler installation directory" :
C:\PellesC (subdirectory "\bin" automatically searched after this "top" directory),
and in subtab "Program Files", type new list next, field by field :
- C compiler : pocc.exe
- C++ compiler : pocc.exe (unused here, because only C compiler)
- linker for dynamic lib : polink.exe
- linker for static lib : polib.exe
- debugger :
- resource compiler : porc.exe
- make program : pomake.exe

NB : All name of executable begin by "po" in reference at Pelles Orinius.
NB1 : Pelles C compiler support only C language (and very strict conformance of C99/C11/C17/C2x recommandations).

After, you select tab "Search directories", and into each subtab, you write with "add" button :
- to compiler : C:\PellesC\include\Win and C:\PellesC\include
- to linker : C:\PellesC\lib\Win and C:\PellesC\lib
- to resource compiler : C:\PellesC\include\Win and C:\PellesC\include

It's recommended then to select an option in tab "Compiler Settings" and in subtab "Other compiler options" to write "-Tx86-coff", and in subtab "#define" to type "*X86*" in first step. And in the last step, in tab "Linker Settings", select windows "Other linker
options" ans type "/MACHINE:X86", if you want generate and test your target in version 32 bits.

You can also define version 64 bits of Pelles Ccompiler, into CB. You must first return in main menu "Settings" and after submenu "Compiler..." to choose "Pelles C (32b)", after click on button "Copy" and to terminate rename it with type "Pelles C (64b)" by button "Rename", by example (=> force identification of "new" compiler into CB).

Nothing to change into "Toolchain executables" menu, but you must change values in "Search directories" menu, for each subtab you
must write with "edit" button to type :
- to compiler : C:\PellesC\include\Win64 and C:\PellesC\include

- to linker : C:\PellesC\lib\Win64 and C:\PellesC\lib
- to resource compiler : C:\PellesC\include\Win64 and C:\PellesC\include

After, you must change "Compiler settings" and "Linker settings". select an option in tab "Compiler Settings" and in subtab "Other compiler options" to write "-Tx64-coff", and suppress "*X86*" in subtab "#define" in first step.
And in the last step, in tab "Linker Settings", select windows "Other linker options" ans type "/MACHINE:X64", if you want generate and test your target in version 64 bits.

You can see that the only difference beetween 32 or 64 bits with this compiler is not based on different binaries, but only with different options during generations, positioned during compilation but also during linkage.

Test of "simple" code with Pelles C into CB

With simply source "hellowworld.c", you can test generation of program into IDE CB, choosing "create new project" in main windows of CB, and choose "console application" with no source proposed by default, because named "main.c" by default, and choose compiler "Pelles C (32b)".
You can select good directory/source with option "add file" after first creation of project into CB.

One time project created, you can generate it with selecting main menu "Build" and choose submenu "Rebuild..." (or CTRL-F11), and save your project.

To test 64 bits, return into main menu "Project" and in submenu "Build options" to change the compiler to "OpenWatcom Compiler (64b)".

You can generate it again with selecting main menu "Build" and choose submenu "Rebuild..." (or CTRL-F11).

If you apply all of precedent instructions, compile and link of your program must be succeeded.

Pleasure of programming is open for you, your imagination is illimited, at your keyboard ! Enjoy !

PS : source file "hellowworld.c" :

```
/* Basic example in language C : hellowworld.c */

#include <stdio.h>
```

```
int main(int argc, char *argv[]) {
/* printf() displays the string inside quotation */
printf("Hello, World!");
return 0;
}
```

PS2 : Use of PellesC compiler in command line (just to illustrate)

Pelles C compiler can be used directly into command console of Windows (CMD.EXE) and next command lines configure it :

SET PATHSAV=%PATH%
SET INCSAV=%INCLUDE%
SET LIBSAV=%LIB%
SET PELLESC=C:\PellesC
REM Choose if you use binary directory of Pelles C beetween 32 bits or 64 bits, by suppress "[64]" after or conserve only "64" beetween []
SET PATH=%PELLESC%\bin;%PATH%
SET INCLUDE=%PELLESC%\include\Win[64];%PELLESC%\include;%INCLUDE%
SET LIB=%PELLESC%\lib\Win[64];%PELLESC%\lib;%LIB%

REM No "one pass" available to generate target with this compiler. Only two passes are supported :
pocc /Fohellowworld.obj hellowworld.c
polink /OUT:hellowworld.exe hellowworld.obj /MACHINE:X64 /SUBSYSTEM:CONSOLE

After work with Pelles C compiler, but, at the end of your work, think to return in initial state ... to avoid difficulties :

SET PATH=%PATHSAV%
SET INCLUDE=%INCSAV%
SET LIB=%LIBSAV%

But, it's much easy to use Pelles C compiler directly into CB IDE especially with complex C program (many C sources and many subdirectories ...) , and multiple targets by example : main DLL and console program to test this DLL, ... -)

PS3 : Options of command line utilities "pocc" and "polink" for Pelles C compiler :

Options of command line "pocc"
Pelles ISO C Compiler, Version 12.00.1

Syntax:
POCC [options] srcfile{.c|.asm}

Options :

- /arch: Select X64 architecture AVX, AVX2, or SSE2
- /D[=
- /diag: Select diagnostic kind CLASSIC or CARET
- /E Preprocess only (to stdout)
- /Fo Name the output file
- /fp: Set floating-point model PRECISE or FAST
- /Gh Enable hook function call
- /Gi Enable trap for signed integer overflow
- /Go Define compatibility names and use OLDNAMES.LIB
- /GA Assume _tls_index is zero for thread local storage
- /GT Generate fiber-safe access for thread local storage
- /GX Enable 64-bit time_t for X64
- /hotpatch[:] Enable Microsoft hotpatching
- /I
- /J Default char type is unsigned
- /M Preprocess only; write MAKE dependencies to stdout
- /MD Enable dynamic C runtime library (POCRT*.DLL)
- /MT Enable multi-threading support (CRTMT*.LIB)
- /openmp Enable OpenMP 3.1 extensions
- /O1 Same as /Os
- /O2 Same as /Ot
- /Ob Set inline expansion model 0, 1 or 2
- /Os Optimize, favor code space
- /Ot Optimize, favor code speed
- /Ox Perform maximum optimizations
- /P Preprocess only (to srcfile.i)
- /std: Select language mode C17, C11 or C99
- /std:C2X Select experimental/unfinished language mode C2X
- /T Select output target (/T? displays a list), list of possible targets (* = current):
  - /Tx86-coff

  - /Tx86-asm
  - /Tx64-coff
  - /Tx64-asm
  - /Tamd64-coff
  - /Tamd64-asm
- /U Undefine a preprocessor symbol
- /utf-8 Set source and execution character set to UTF-8
- /V Set verbosity level 0, 1 or 2
- /W Set warning level 0, 1 or 2
- /Wd Disable warning `#n`
- /X Don't search standard places for `#include` files
- /Zd Enable line number debugging information
- /Ze Enable Microsoft extensions
- /Zg Write function prototypes to stdout
- /Zi Enable full debugging information
- /Zl Omit default library name in object file
- /Zs Syntax check only
- /Zx Enable Pelle's C extensions

Options of command line "polink"
Pelles Linker, Version 12.00.0

Syntax:
POLINK [{ option | file | @commandfile } ...]

Options:

- /ALIGN:#
- /ALLOWBIND[:NO]
- /ALLOWISOLATION[:NO]
- /ALTERNATENAME:symbol=symbol
- /BASE:address
- /DBG:filename
- /DEBUG[:NO]
- /DEBUGTYPE:{PO|COFF|BOTH}
- /DEF:filename
- /DEFAULTLIB:filename
- /DELAY:{NOBIND|UNLOAD}

- /DELAYLOAD:filename
- /DEPENDENTLOADFLAG:flag
- /DLL
- /DRIVER[:{UPONLY|WDM}]
- /DYNAMICBASE[:NO]
- /ENTRY:symbol
- /EXPORT:symbol[=[module.]symbol][,@ordinal[,NONAME]][,DATA]
- /FIXED[:NO]
- /FORCE:MULTIPLE
- /HEAP:reserve[,commit]
- /HIGHENTROPYVA[:NO]
- /IMPLIB:filename
- /INCLUDE:symbol
- /INTEGRITYCHECK[:NO]
- /LARGEADDRESSAWARE[:NO]
- /LIBPATH:path
- /MACHINE:{X64|X86}
- /MANIFEST[:{NO|EMBED[,ID=#]}]
- /MANIFESTDEPENDENCY:dependency
- /MANIFESTFILE:filename
- /MANIFESTUAC[:{NO|fragment}]
- /MAP[:filename]
- /MAPINFO:{EXPORTS|FIXUPS|LINES|PATHS}
- /MERGE:from=to
- /NODEFAULTLIB
- /NOENTRY
- /NOIMPLIB
- /NXCOMPAT[:NO]
- /OPT:{REF|NOREF}
- /OSVERSION:#[.##]
- /OUT:filename
- /RELEASE
- /SAFESEH[:NO]
- /SECTION:name,[!][E][R][W][S][D][K][P][,ALIGN=#]
- /STACK:reserve[,commit]
- /STUB:filename
- /SUBSYSTEM:{CONSOLE|NATIVE|WINDOWS}[,#[.##]]

- /SWAPRUN:{CD|NET}
- /TSAWARE[:NO]
- /VERBOSE
- /VERSION:#[.##]

Pelles C is very good C compiler on Windows systems, it "make the job", it support "openmp" and recent standards of C language, and generate very small executable with option "/Os". But it's only C compiler, not C++ ...