# How config OneAPI Intel C Compiler, 32 and 64 bits into CodeBlocks

Name of tutorial : How config OneAPI Intel C\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
```

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\C++ on Windows : OneAPI Intel C\C++ compiler (version 32 and 64 bits).

How to install OneAPI Intel C\C++ Compiler (32 and 64 bits) ?

This compiler is available on site :
https://www.intel.com/content/www/us/en/developer/tools/oneapi/dpc-compiler-download.html

You must download many kits if you want used version 32 bits more version 64 bits (files available mid 2024) :

Intel OneAPI C/C++ compiler : "w_dpcpp-cpp-compiler_p_2024.2.0.491.exe"
Runtime Intel OneAPI C/C++ compiler : "w_dpcpp_cpp_runtime_p_2024.2.0.980.exe"
Intel Performance Primitives : "w_ipp_oneapi_p_2021.12.0.557.exe"
Math Kernel Primitives : "w_onemkl_p_2024.2.0.662.exe"
Base kit 32 bits : "w_BaseKit_32bit_p_2024.2.0.222.exe"
(only if you want generation 32 bits with this compiler)

One important question about use of this compiler : "free" or not ?

In all case, all of these kits are available without "identification" of user. Yes, during installation of these kits, you must accept "use rules" from Intel Corporation.
But in many "free" software, copyrigths appear in code source by example. Then "free" or not ?
You can consider that this compiler (last version based from LLVM architecture) is "free use", but all
kits are and "rest" proprietary's right of Intel Corporation. Not a problem ... like MSVC ...

One precision, this compiler (like "clang") must to be in "front end" of another full development environment like MSVC
or MinGW32/64. For next configurations, the choice is to "back end" with Visual Studio 2002 Community + SDK Windows 11.

In consequence, the configuration of this compiler under IDE CB is not ... simply, but after many inspections of
"configuration" commands available, it is possible to understand how configure correctly this compiler into CB.

First, you must access into IDE CB at menu "Settings" and submenu "Compilers" to select "Intel C/C++ compiler".
To distinguish version 32 bits or 64 bits into CB, you must rename this ident of compiler in "Intel C/C++ compiler (64 bits)".
An after, you must position "Compiler's installation directory" to "C:\Program Files (x86)\Intel\oneAPI\compiler\latest".

This directory is the same beetween 32 or 64 bits, but the rest of configuration is different in 32 or 64 bits.
You must select list of tools like this (in tab "Program Files") :
C Compiler : icx.exe
C++ Compiler : icxp.exe
Linker for dynamic libs : xilink.exe
Linker for statics libs : xilib.exe
Debugger : GDB/CDB (choice by default, because Intel provide special version GDB)
Resource compiler : rc.exe (yes, this of SDK Windows 11)
Make program : nmake.exe (yes, this of MSVC included in Visual Studio 2002 Community)

But with this configuration, you must add multiple directories (in another tab "Additional Paths") :
C:\Program Files (x86)\Intel\oneAPI\tbb\latest\env..\bin
C:\Program Files (x86)\Intel\oneAPI\ocloc\latest\bin
C:\Program Files (x86)\Intel\oneAPI\dev-utilities\latest\bin
C:\Program Files (x86)\Intel\oneAPI\debugger\latest\opt\debugger\bin
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib\ocloc
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\bin
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\host\windows64\bin
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\bin
C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.40.33807\bin\Hostx64\x64
C:\Program Files (x86)\Windows Kits\10\bin\10.0.22621.0\x64
C:\Program Files (x86)\Windows Kits\10\bin\x64

C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Current\Bin\amd64

WARNING : After update, all number included into name of directories of MSVC/SDK can change !!!
To integrate these evolutions, it's seem rational to define environment variables on system like this :
VS_VERSION define to 2022 (at date)
VS_NUM define to 14.40.33807 (at date) and
KIT_VERSION define to 10 (at date)
KIT_NUM define to 10.0.22621.0 (at date)
Then, you can use these variables into configuration of CB with use of %var% to be independant of evolutions.

And, it's not all, you must configure in tab "Search Directories" and select "Compiler", or "Linker" or "Resource Compiler"
subtabs.

For compiler, search directories of "include files" are (and it's the same for "Resource Compiler") :
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib\clang\18\include
C:\Program Files (x86)\Intel\oneAPI\tbb\latest\include
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\include
C:\Program Files (x86)\Intel\oneAPI\dev-utilities\latest\include
C:\Program Files (x86)\Intel\oneAPI\ocloc\latest\include
C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\include
C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Auxiliary\VS\include
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\ucrt
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\um
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\shared
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\winrt
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\cppwinrt

For linker, search directories of "lib files" are :
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib
C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x64
C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64\store
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x64

Remark : If you wan't use environment variables, you can translate these by "real value" of directory into CB.

It's some tedious, but with these "full" configurations, you can build an program on Windows 11 64 bits with success.

Now, you must copy "Intel C/C++ compiler (64 bits)" in list of available compilers into CB (menu "Settings" submenu
"Compilers"), and rename it by "Intel C/C++ compiler (32 bits)" by example.

New configurations for this compiler in version 32 bits are next :

In subtab "Additional Paths" :
C:\Program Files (x86)\Intel\oneAPI\tbb\latest\env..\bin32
C:\Program Files (x86)\Intel\oneAPI\ocloc\latest\bin
C:\Program Files (x86)\Intel\oneAPI\dev-utilities\latest\bin
C:\Program Files (x86)\Intel\oneAPI\debugger\latest\opt\debugger\bin
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib\ocloc
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\bin32
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\host\windows64\bin
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\bin
C:\Program Files\Microsoft Visual
Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\bin\Hostx86\x86
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin%KIT_NUM%\x86
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin\x86
C:\Program Files\Microsoft Visual
Studio%VS_VERSION%\Community\MSBuild\Current\Bin\amd64

In tab "Search Paths", only changes are mandatory in subtab "Linker" ("include" directory files are same) :
C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib32
C:\Program Files\Microsoft Visual
Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x86
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x86
C:\Program Files\Microsoft Visual
Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x86\store
C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x86

And, to terminate, you must "force" an option in tab "Compiler Settings" and in subtab "Other compiler options" to "-m32",

it's option to force 32 bits generation with this compiler (Intel inform that this option will disappear in future version).

And, 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 "Intel C/C++ compiler (64 bits)".
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).

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 : You can also use compiler OneAPI Intel CC++ compiler in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set PATH=C:\Program Files (x86)\Intel\oneAPI\compiler\latest;C:\Program Files (x86)\Intel\oneAPI\tbb\latest\env..\bin;C:\Program Files (x86)\Intel\oneAPI\ocloc\latest\bin;C:\Program Files (x86)\Intel\oneAPI\debugger\latest\opt\debugger\bin;C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib\ocloc;%PATH%
set PATH=C:\Program Files (x86)\Intel\oneAPI\compiler\latest\bin;C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\host\windows64\bin;C:\Program Files (x86)\Intel\oneAPI\compiler\latest\opt\oclfpga\bin;%PATH%
set PATH=C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\bin\Hostx64\x64;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin%KIT_NUM%\x64;%PATH%
```

set PATH=C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin\x64;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\MSBuild\Current\Bin\amd64;%PATH%
set LIBSAV=%LIB%
set LIB=C:\Program Files (x86)\Intel\oneAPI\compiler\latest\lib;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64\store
set LIB=C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x64;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x64;%LIB%

REM Compile + link in one pass
icx hellowworld.c /Fe:hellowworld.exe
REM Compile + link in two pass
icx /c hellowworld.c /Fo:hellowworld.obj
xilink hellowworld.obj /OUT:hellowworld.exe /MACHINE:X64 /SUBSYSTEM:CONSOLE

Continue with use of OneAPI Intel CC++ compiler, and don't forgive, at the end of your work, to return to initial state :

set PATH=%PATHSAV%
set LIB=%LIBSAV%

And, with precedent example, you can also generate version 32 bits with "icx" but, you must change values of PATH and LIB like
with good directories before (don't forgive "-m32" during run of "icx" and "/MACHINE:X86" during call of "xilink" if two pass).

But, it's much easy to use OneAPI Intel C/C++ compiler directly into CB IDE especially with complex C program
(many C sources and many subdirectories ...) -)

PS3 : Syntax of intel compiler "icx" on command line is very "verbose". It's much simply to redirect
"help" of this compiler into text file with next command : "icx /help > command_icx.txt"

But to resume principal and useful options of command "icx", you can use by example :

```
"-m32"                          Generate i386 32 bits object or executable.
"-m64"                          Generate x64 64 bits object or executable.
"/c"                Compile and assemble, but do not link.
"/debug"                        Enable debug information.
```

```
"/D <macro[=value]>" Define macro to preprocessor, and optionnally attribute a
value.
"/Fo:<file>"        Place the object output into <file> (with "/c" option)
"/Fe:<file>"        Place the executable/DLL output into <file>
"/I <directory>"        Add directory to search include files.
"/LD"               Option to create DLL ("/LDd" if create debug DLL)
"/link <options>    Forward options to the linker. Example of useful options
in the case of use linker of MSVC :
                                                    /DEBUG[:{FASTLINK|FULL|NONE}]
Link with Debug option, and you can choose type of Debug.
                                                    /DEF:<file>
create a def file of all exported sysmbols of shared library
                                                    /DLL
create DLL on Win32 systems
                                                    /IMPLIB:<file>
create a library file of all exported sysmbols of shared library
                                                    /LIBPATH:<directory>
Add directory to search library files.
                                                    /MACHINE:
{ARM|ARM64|ARM64EC|ARM64X|EBC|X64|X86} Choose architecture of target (ex : X86
32 bits, X64, 64 bits, by example)
                                                    /OUT:<file>
Place the output into <file>
                                                    /PDB:<file>
Place the pdb file into <file>
                                                    /RELEASE
Generate "Release" version (and with no option /DEBUG[:{FASTLINK|FULL|NONE}])
                                                    /SUBSYSTEM:ident
create target based on "ident" that can be valued by

{BOOT_APPLICATION|CONSOLE|EFI_APPLICATION|

EFI_BOOT_SERVICE_DRIVER|EFI_ROM|EFI_RUNTIME_DRIVER|

NATIVE|POSIX|WINDOWS|WINDOWSCE}[,#[.##]]
```

Same link options can be positioned with command "xilink" but without "/link" like with tool "link" of MSVC.

You can consult many documentation on site :
https://www.intel.com/content/www/us/en/developer/tools/oneapi/base-toolkit-documentation.html