How config Digital Mars Compiler C_C++ (32 bits only) into CodeBlocks

Full name of tutorial : How config Digital Mars Compiler C/C++ (32 bits only) into Code::Blocks on Windows 11 64 bits.

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

Presentation of Digital Mars Compiler C/C++ included into CB

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 two compilers on Windows : LCC 32 bits and LCC 64 bits.

How to install Digital Mars Compiler C/C++ 32 bits ?

You can download it from Internet site : https://digitalmars.com/download/freecompiler.html

Then, you select link "Digital Mars C/C++ Compiler Version 8.57 (3662658 bytes) (NEW!)" to donwload file "dm857c.zip", and select link "Basic Utilities (262,000 bytes)" to download file "bup.zip" (mandatory to compile resource file).
Decompress these two files into "C:\dm" directory (by example).

Configuration of Digital Mars Compiler C/C++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..." : "Digital Mars Compiler" (autodetect by CB normally). If not, you can select this compiler with same choices of menus, and in the subtab "Toolchain executables" and click on "..." button to search installation's directory of DMC.
After, you must verify that fields describe next are parametered into CB.

In tab "Toolchain executable", you must find in field "Compiler installation directory" :
C:\dm (subdirectory "\bin" automatically searched after this "top" directory),
and in subtab "Program Files", list next :
compilateur C : dmc.exe
compilateur C++ : dmc.exe
linker for dynamic lib : link.exe
linker for static lib : lib.exe
debugger :
resource compiler : rcc.exe
make program : make.exe

If CB propose different values of fields described below, you can change/force it.

After, you select tab "Search directories", and into each subtab, you write with "add" button, if not searched by default :
to compiler : C:\dm\include\win32 and C:\dm\include
to linker : C:\dm\lib
to resource compiler : C:\dm\include\win32 and C:\dm\include

Test of "simple" code with Digital Mars Compiler C/C++ into CB

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 "Digital Mars Compiler".
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).

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 Digital Mars Compiler C/C++ in command line (just to illustrate)

Digital Mars Compiler C/C++ can be used directly into command console of Windows (CMD.EXE) and next command lines configure it :

```
SET PATHSAV=%PATH%
SET DMC=C:\dm
SET PATH=%DMC%\bin;%PATH%
```

By example, to generate "console" executable :

```
REM Generate console executable in one pass
dmc hellowworld.c -ohellowworld.exe
REM Generate console executable in two pass
dmc -c hellowworld.c -ohellowworld.obj
link hellowworld.obj /SUBSYSTEM:console
```

After, work with Digital Mars Compiler C/C++, but, at the end of your work, think to return in initial state ... to avoid difficulties :

```
.....
SET PATH=%PATHSAV%
```

But, it's much easy to use Digital Mars Compiler C/C++ 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 : Syntax of tools of DMC

Syntax and options of "dmc.exe" :

```
Copyright (C) Digital Mars 2000-2004. All Rights Reserved.
Written by Walter Bright www.digitalmars.com/ctg/sc.html
DMC is a one-step program to compile and link C++, C and ASM files.
Usage ([] means optional, ... means zero or more):
```

```
DMC file... [flags...] [@respfile]
```

file... .CPP, .C or .ASM source, .OBJ object or .LIB library file name
@respfile... pick up arguments from response file or environment variable
flags... one of the following:

-a[1|2|4|8] alignment of struct members
-A strict ANSI C/C++
-Aa enable new[] and delete[]
-Ab enable bool
-Ae enable exception handling
-Ar enable RTTI
-B[e|f|g|j] message language: English, French, German, Japanese
-c skip the link, do compile only
-cpp source files are C++
-cod generate .cod (assembly) file
-C no inline function expansion
-d generate .dep (make dependency) file
-D #define DEBUG 1
-Dmacro[=text] define macro
-e show results of preprocessor
-EC do not elide comments
-EL #line directives not output
-f IEEE 754 inline 8087 code
-fd work around FDIV problem
-ff fast inline 8087 code
-g generate debug info
-gf disable debug info optimization
-gg make static functions global
-gh symbol info for globals
-gl debug line numbers only
-gp generate pointer validations
-gs debug symbol info only
-gt generate trace prolog/epilog
-GTnnnn set data threshold to nnnn
-H use precompiled headers (ph)
-HDdirectory use ph from directory
-HF[filename] generate ph to filename
-HHfilename read ph from filename
-HIfilename #include "filename"

-HO include files only once

-HS only search -I directories

-HX automatic precompiled headers

-Ipath #include file search path

-j[0|1|2] Asian language characters

0: Japanese 1: Taiwanese and Chinese 2: Korean

-Jm relaxed type checking

-Ju charunsigned char

-Jb no empty base class optimization

-J chars are unsigned

-llistfile generate list file

-L using non-Digital Mars linker

-Llink specify linker to use

-L/switch pass /switch to linker

-Masm specify assembler to use

-M/switch pass /switch to assembler

-mtsmclvfnrpxzdowu set memory model

s: small code and data m: large code, small data

c: small code, large data l: large code and data

v: VCM r: Rational 16 bit DOS Extender

p: Pharlap 32 bit DOS Extender x: DOSX 32 bit DOS Extender

z: ZPM 16 bit DOS Extender f: OS/2 2.0 32 bit

t: .COM file n: Windows 32s/95/98/NT/2000/ME/XP

d: DOS 16 bit o: OS/2 16 bit

w: SS != DS u: reload DS

-Nc function level linking

-NL no default library

-Ns place expr strings in code seg

-NS new code seg for each function

-NTname set code segment name

-NV vtables in far data

-o-+flag run optimizer with flag

-ooutput output filename

-p turn off autoprototyping

-P default to pascal linkage

-Pz default to stdcall linkage

-r strict prototyping

-R put switch tables in code seg

-s stack overflow checking

-S always generate stack frame

-u suppress predefined macros
-v0|1|2 verbose compile
-w suppress all warnings
-wc warn on C style casts
-wn suppress warning number n
-wx treat warnings as errors
-W{0123ADabdefmrstuvwx-+} Windows prolog/epilog
-WA Windows EXE
-WD Windows DLL
-x turn off error maximum
-XD instantiate templates
-XItemp instantiate template class temp
-XIfunc(type) instantiate template function func(type)
-[0|2|3|4|5|6] 8088/286/386/486/Pentium/P6 code

Syntax and options of "link.exe" (or "optlink.exe") (much complex) :

Command line operation of OPTLINK uses the following syntax:
LINK obj[,out[,map[,lib[,def[,res]]]]]

OPTLINK (R) for Win32 Release 8.00.16
Copyright (C) Digital Mars 1989-2013 All rights reserved.
http://www.digitalmars.com/ctg/optlink.html
A[lignment] BAS[e] B[atch]
BI[nary] BYO[rdinal] CHECKA[bort]
CHECKE[xe] CHECKS[um] CO[deview]
COM[defsearch] CP[armaxalloc] CVV[ersion]
CVW[arnings] DEB[ug] DEBUGA[pploader]
DEBUGB[rowser] DEBUGC[overages] DEBUGF[iles]
DEBUGLI[nes] DEBUGLO[cals] DEBUGM[odules]
DEBUGP[ublics] DEBUGR[eferences] DEBUGT[ypes]
DEF[aultlibrarysearch] DE[lexecutable] DET[ailedmap]
DO[sseg] EC[hoindirect] EMSM[axsize]
EMSP[ageframeio] EMSU[se40] EN[try]
ER[rorflag] E[xepack] EXET[ype]
F[arcalltranslation] FI[xds] FIXE[d]
G[roupassociation] GROUPS[tack] HEA[p]
H[elp] IG[norecase] IMPD[ef]
IMPL[ib] IMPLIBC[off] I[nformation]
LA[RGEADDRESSAWARE] L[inenumbers] LO[wercase]
MAC[hine] M[ap] NOB[atch]

NOCHECKA[bort] NOCHECKE[xe] NOCHECKS[um]
NOCO[deview] NOCOM[defsearch] NOCV[pack]
NODEB[ug] NODEBUGA[pploader] NODEBUGB[rowser]
NODEBUGC[overages] NODEBUGLI[nes] NODEBUGLO[cals]
NODEBUGP[ublics] NODEBUGR[eferences] NODEBUGT[ypes]
NOD[efaultlibrarysearch] NODEL[executable] NODET[ailedmap]
NODO[sseg] NOEC[hoindirect] NOEMSP[ageframeio]
NOEMSU[se40] NOER[rorflag] NOEXE[pack]
NOE[xtdictionary] NOF[arcalltranslation] NOFI[xds]
NOG[roupassociation] NOGROUPS[tack] NOI[gnorecase]
NOLI[nenumbers] NOL[ogo] NOM[ap]
NONA[mes] NONT[host] NON[ullsdosseg]
NOP[ackcode] NOPACKD[ata] NOPACKF[unctions]
NOPACKI[fnosegments] NOPAU[se] NOPR[ompt]
NOR[elocationcheck] NOREO[rdersegments] NOSCANLIB
NOSCANLINK NOWARND[ups] NOWI[npack]
NOX[ref] NT[host] NU[llsdosseg]
ON[error] OPT PAC[kcode]
PACKD[ata] PACKF[unctions] PACKI[fnosegments]
PACKS[ize] PADC[ode] PADD[ata]
PAG[esize] PAU[se] PM[type]
PR[ompt] RC RELOC[ationcheck]
REO[rdersegments] SCANLIB SCANLINK
SE[gments] SEGP[ack] SI[lent]
ST[ack] STU[b] SU[bsystem]
T[iny] U[ppercase] VERS[ion]
W[arnfixup] WARND[ups] WI[npack]
XM[smaxsize] X[ref] XN[oignorecase]
XU[ppercase]

You can consult documentation of Digital Mars Compiler on this site https://digitalmars.com/dmc/pdf.html