
COMPILER DES
LIBRAIRIES .DLL
Octobre 2019

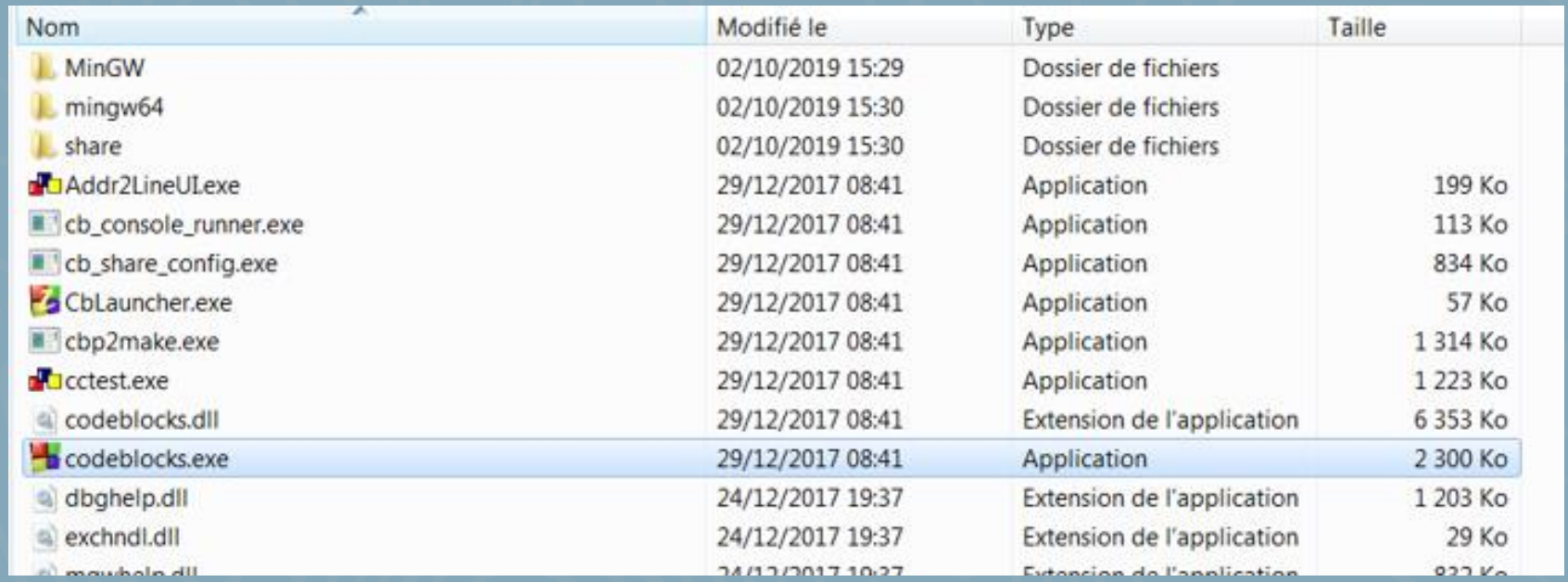
SOMMAIRE

1.1. Etape 1 | Installation & lancement de la plateforme IDE C/C++ CodeBlocks

- Télécharger en local les deux logiciels référencés ci-après:

Version portable v17.12 du logiciel Codeblocks	Version portable du compilateur GCC (32bits)
Codeblocks_v17.12.7z	TDM-GCC-64.7z

- Lancement de l'IDE CodeBlocks via l'icone codeblocks.exe (à la racine du dossier)

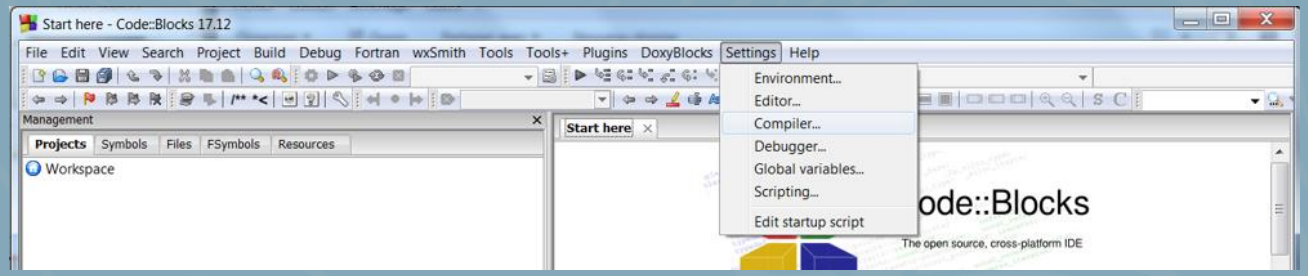


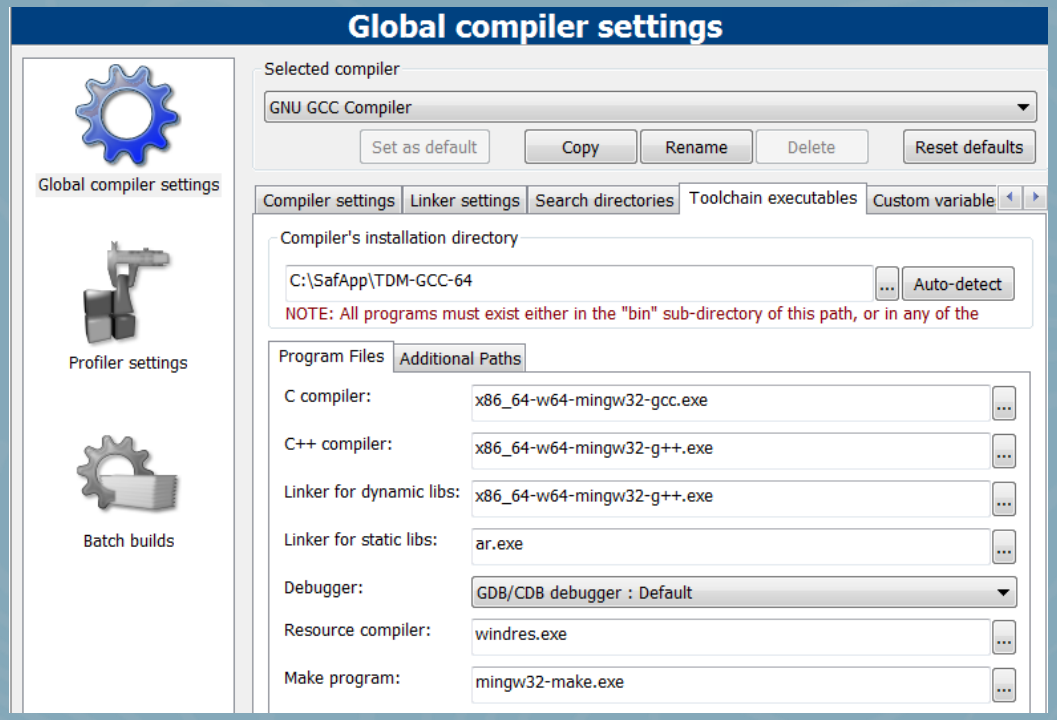
Nom	Modifié le	Type	Taille
MinGW	02/10/2019 15:29	Dossier de fichiers	
mingw64	02/10/2019 15:30	Dossier de fichiers	
share	02/10/2019 15:30	Dossier de fichiers	
Addr2LineUI.exe	29/12/2017 08:41	Application	199 Ko
cb_console_runner.exe	29/12/2017 08:41	Application	113 Ko
cb_share_config.exe	29/12/2017 08:41	Application	834 Ko
CbLauncher.exe	29/12/2017 08:41	Application	57 Ko
cbp2make.exe	29/12/2017 08:41	Application	1 314 Ko
cctest.exe	29/12/2017 08:41	Application	1 223 Ko
codeblocks.dll	29/12/2017 08:41	Extension de l'application	6 353 Ko
codeblocks.exe	29/12/2017 08:41	Application	2 300 Ko
dbghelp.dll	24/12/2017 19:37	Extension de l'application	1 203 Ko
exchndl.dll	24/12/2017 19:37	Extension de l'application	29 Ko

1.2. Etape 2 | Mise à jour des settings du logiciel Codeblocks

- ☞ Aller dans le menu « Settings » de la toolbar puis sélectionner « Compiler … » …
- ☞ Paramétrer l'onglet « Toolchain executables » conformément à l'exemple ci-contre …

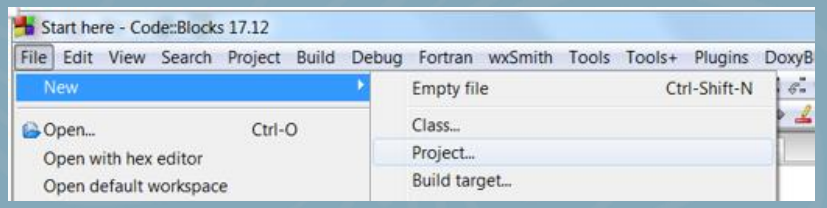
- */!\ Nota: l'installation directory devra bien entendu pointer sur votre compilateur TDM-GCC-64 préalablement copié - cf. étape précédente (exemple donné ici en cohérence avec mon environnement)*

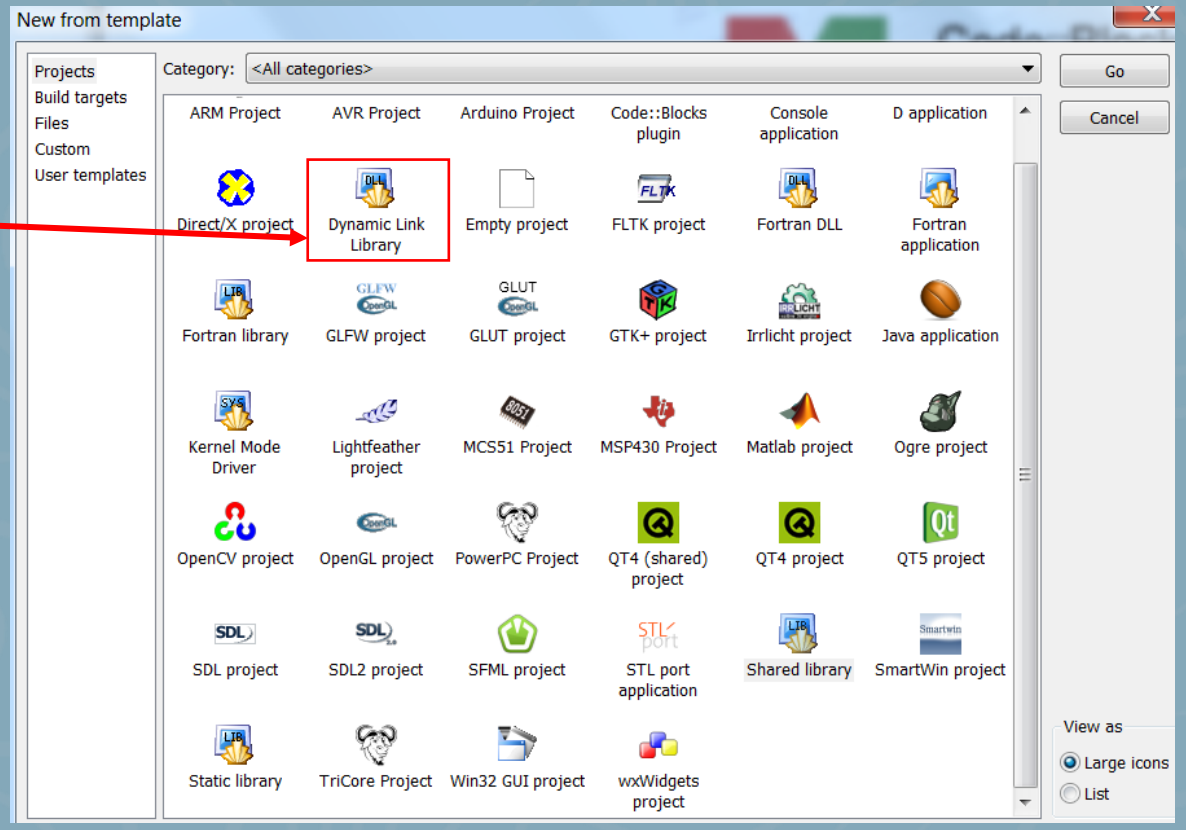




1.3. Etape 3 | Création du projet (1/2)

- Définir un projet .dll via la combinaison File/New/project …
- Puis sélectionner l'option Dynamic Link Library …





1.3. Etape 3 | Création du projet (2/2)

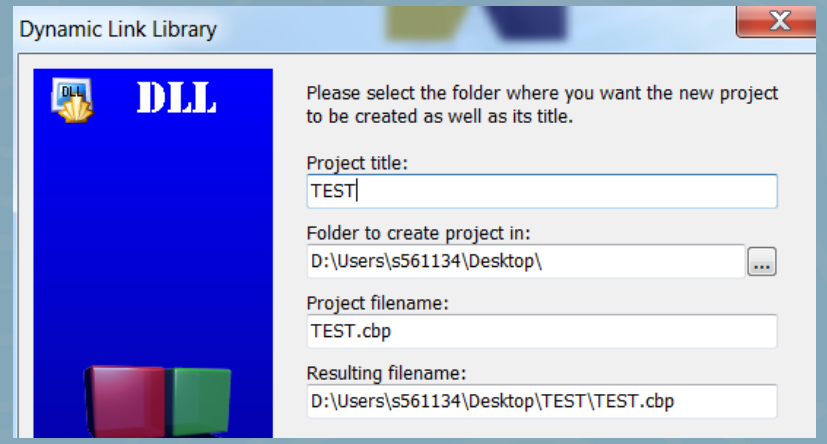
☞ Après avoir cliqué sur Go, puis Next, définir le « Project title » et le « répertoire de travail » …

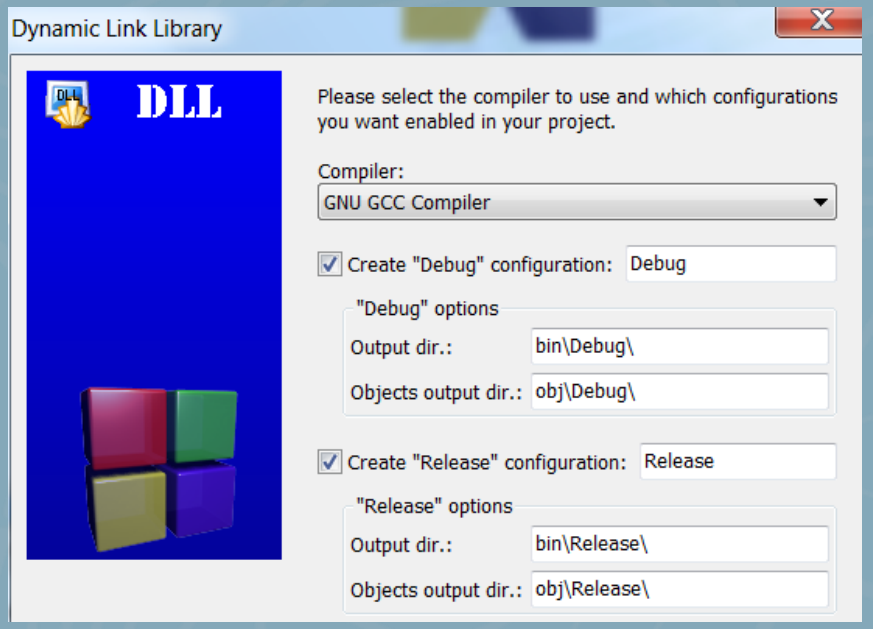
- */!\ Nota: les 2 derniers champs sont automatiquement mis à jour*

☞ Puis cliquer sur Next pour accéder au menu suivant (compiler et configurations) à définir comme dans l'exemple ci-contre … (

- */!\ Nota: sélectionner a minima l'option « Release » (indispensable à la compilation de la dll)*

☞ Cliquer sur finish pour lancer la création du dossier de travail et de l'arborescence dans le logiciel Codeblocks





1.4. Etape 4 | Update des syntaxes pour la création d'une dll

- Remplacer respectivement l'intégralité des fichiers main.cpp et main.h créés par défaut par la totalité du contenu des fichiers templates équivalents disponibles ci-joint:


template_main.cpp


template_main.h

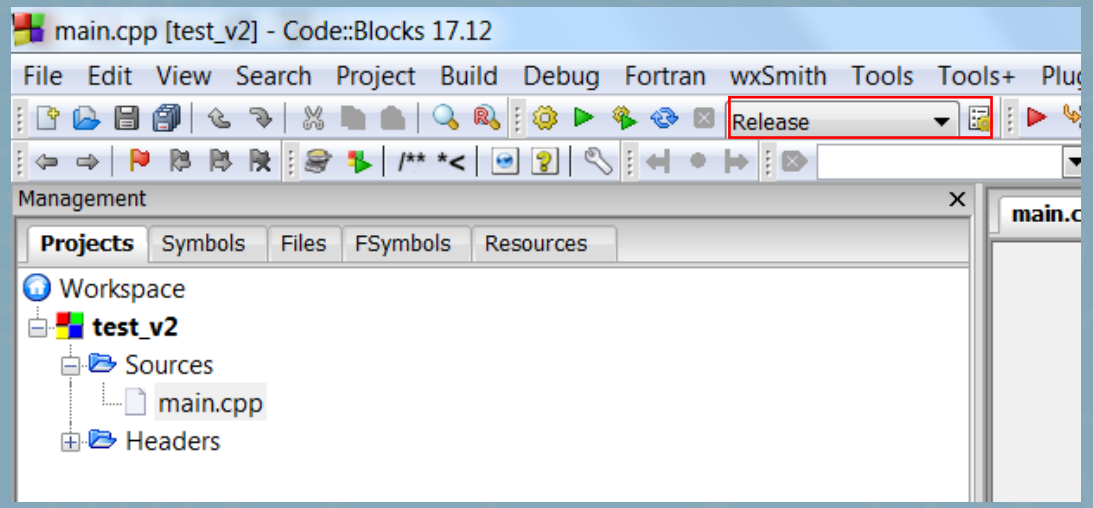
- Procédure d'ajout d'une nouvelle fonction à la libraire …
  - **Dans le fichier main.cpp:** dupliquer le bloc type (cf. ci-après) puis coder la fonction souhaitée, sans oublier de la renommer.

```
double DLL_EXPORT SomeFunction2(double a, double b)
{
    double r;
    r = (a + b) / 2;
    return r;
}
```

  - **Dans le fichier main.h:** compléter en y insérant la nouvelle fonction préalablement définie dans le fichier main.cpp à l'instar des fonctions exemples proposées (cf. lignes 21 et 22).
- Compilation de la dll en allant dans Build/Build (ou raccourci ctrl+F9)

1.5. Etape 5 | Compilation de la dll

- Sélectionner « Release » dans le menu « Build Target » (cf. zone encadrée en rouge ci-dessous) :



- Après avoir sélectionner le fichier main.cpp dans l'arborescence projet, lancer la compilation de la dll en allant dans Build/Build (ou raccourci ctrl+F9)

➔ La dll sera stockée sous bin/Release dans le répertoire de travail défini lors de l'étape 3.2

1.6. Etape 6 | Appel des librairies .dll

- En environnement Python …
  - En PJ ci-contre, un exemple des quelques lignes de codes à intégrer pour charger & appeler les fonctions d'une librairie dll (à réadapter) :
  - */!\ Nota WARNING: le compilateur C/C++ étant en 32bits, cela contraint à l'utilisation d'une version python 32bits (tests effectuées sur la v2.7, sans doute des révisions à prévoir si passage sur une v3.xx)*

template_dll_loading.py

- Sous Matlab (tests effectués en R2013b) …
  a. Copier le fichier main.h source dans le même dossier que la dll associée
  b. Ouvrir le dossier de travail dans l'interface Matlab
  c. Taper la commande mex -setup pour définir le compilateur C++ compatible de votre version matlab
  d. Rentrer la succession des commandes suivantes:
     - « loadlibrary XXXX.dll main.h mfilename visaprototype.m » → création du fichier visaprototype.m
     - « loadlibrary('XXXX.dll', @visaprototype) » → chargement de la dll via le fichier visaprototype.m
     - « calllib('XXXX','SomeFunction', int8('job')) » → appel de la fonction 'SomeFunction'
     - « calllib('XXXX','SomeFunction2',1, 2) » → appel de la fonction 'SomeFunction2'
     - « unloadlibrary XXXX » → déchargement de la dll une fois tous les appels de fonction terminés

Enjoy