GUIDE DE LA GPIB TOOLBOX

Tables des matières

Table des figures

1 PRESENTATION

La GPIB ToolBox regroupe une série de fonctions, écrites en C/C++ par Tibault Reveyrand et compilées sous forme de DLL (dynamic link library). Elle permet d'utiliser le protocole de communication VISA sur Windows afin de communiquer, par l'intermédiaire de Scilab et au travers d'une liaison GPIB ou série, avec vos instruments de mesure. Pour fonctionner, la ToolBox suppose que soit installés, au préalable, les drivers NI-VISA développés par National Instruments. Ce rapport explique comment procéder lors de l'installation, comment réagir aux problèmes qui peuvent être rencontrés durant cette installation et comment utiliser la GPIB ToolBox.

2 INSTALLATION DE NI-VISA

NI-VISA est, comme nous l'avons dis, un ensemble de drivers, permettant d'utiliser le protocole VISA sur Windows. C'est un pack payant qui est fournit lors de l'installation de NI-488.2, un logiciel jouant le rôle d'interface entre le PC et les instruments de mesure. Le logiciel, étant un produit commercial, ne pose généralement pas de problème à l'installation, il faut juste être vigilant sur les composants installés (l'assistant d'installation n'installera pas forcément NI-VISA par défaut !).

3 INSTALLATION DE LA GPIB TOOLBOX

3.1 Installation standard

1- Désarchivez « GPIB.zip »sous « <dossier Scilab>\contrib »
2- Exécuter le script « <dossier Scilab>\contrib\GPIB\loader.sce »
3- Si l'installation se passe bien vous verrez apparaître le message :

« shared archive loaded
Link done
Loading : GPIB Toolbox - v1.41 - 6th April 2007 »

Si l'exécution du script provoque l'erreur 236 « shared archive not loaded ». La section suivante explique comment réagir.

3.2 Recompilation de la DLL

Le problème est dû aux chemins des bibliothèques visa.h, visatype.h et visa32.lib, qui sont probablement différents de ceux entrés lors de la compilation de la DLL. Il va donc nous falloir recompiler cette DLL, en fournissant les chemins d'accès appropriés. La DLL est, à la base, compilée sous microsoft visual C++, nous allons, toutefois, utiliser **DEV-C++** (logiciel gratuit).

1- Repérez les fichiers visa.h, visatype.h et visa32.lib. (exemple : « C:\VXIPNP\WinNT\include » et « C:\VXIPNP\WinNT\lib\msc »).
2- Ouvrez DEV-C++, créer un nouveau projet « DLL » nommé « Sci_HPIB » et sauvez le dans le dossier que vous souhaitez.

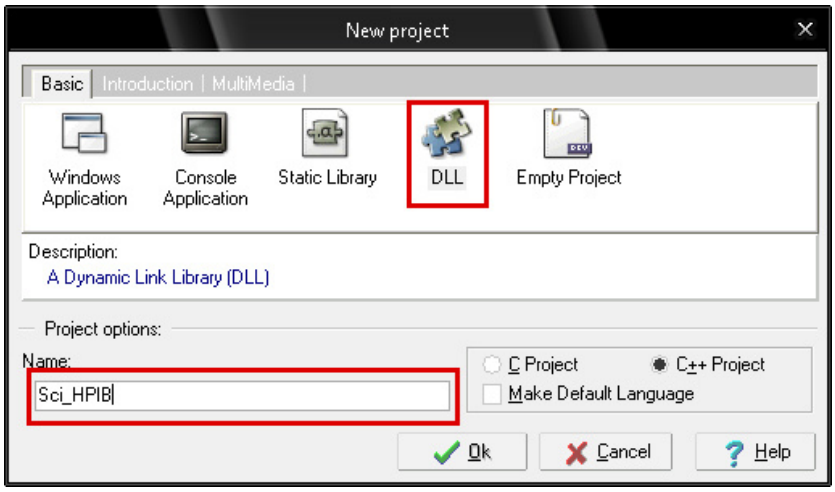

figure 1. Création du projet

3- Supprimez les fichiers « dllmain.cpp » et « dll.h » de l'arborescence (clic droit>Remove file), sans les sauver.
4- Ajoutez les fichiers « visa.h », « visatype.h » et « main.cpp » (<dossier Scilab>\contrib\GPIB\routines\main.cpp) dans l'arborescence (clic droit>Add to project).

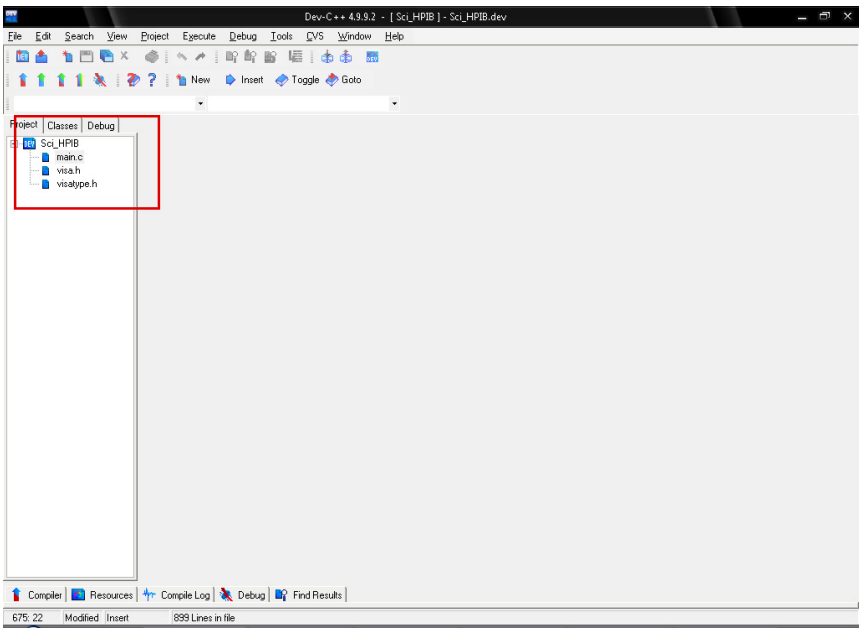

figure 2. Création de l'arborescence.

5- Faites Project>Project Options. Dans l'onglet Parameters, appuyez sur « Add library or object » et ajoutez visa32.lib.

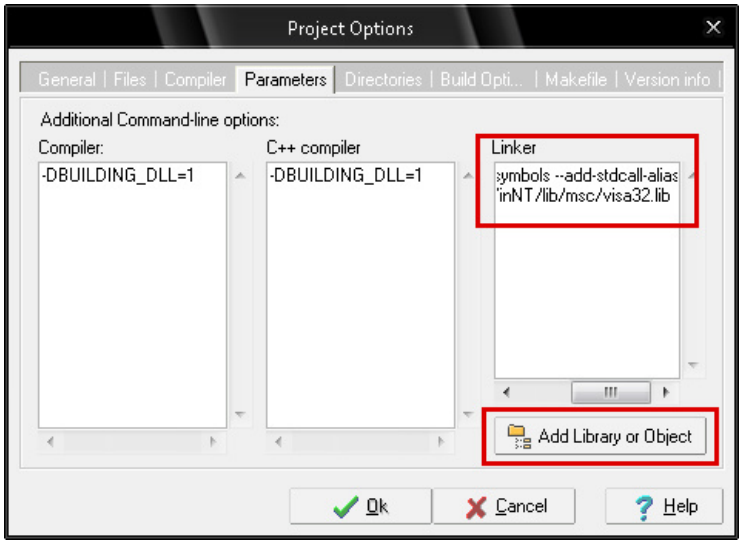

figure 3. Ajout de visa32.lib.

6- Dans l'onglet Directories, ajoutez le chemin de « visa.h » et « visatype.h » à la sous-catégorie Include directories et le chemin de « visa32.lib », à la sous-catégorie Library directories.

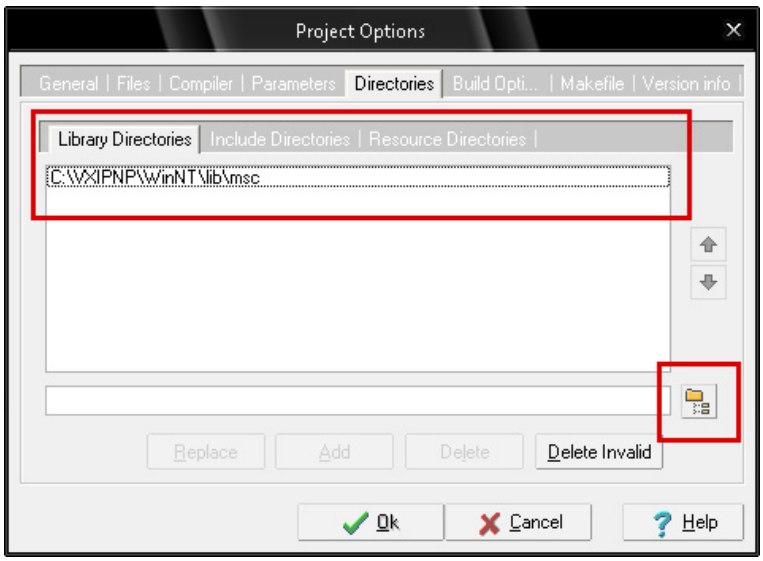

figure 4. Ajout des chemins d'accès.

7- Dans l'onglet Build Options, rajoutez le chemin d'accès de votre projet, dans les champs « executable output directory » et « object file output directory », et cochez l'option « override output filename ».

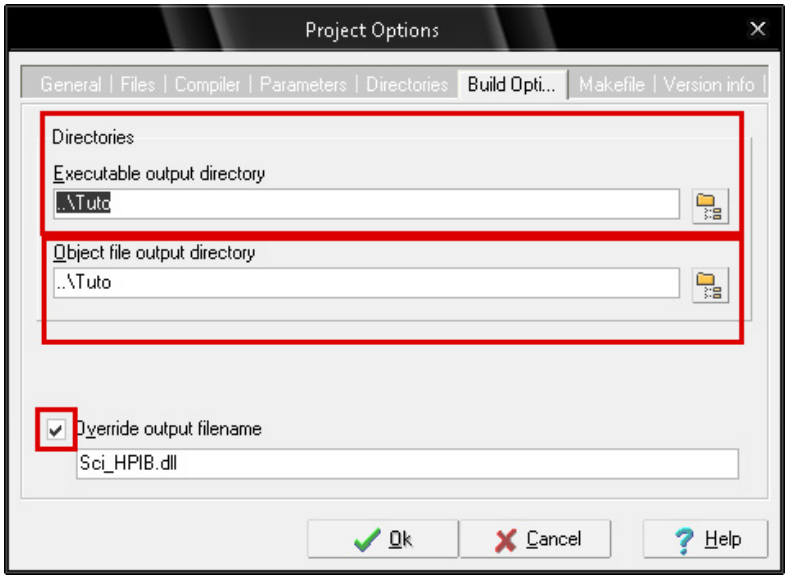

figure 5. Ajout du répertoire de sortie.

8- Appliquer alors les changements en cliquant sur « Ok ».

Un autre fichier, fourni avec la toolbox à son utilité : main.def. Ce fichier est utilisé lors de la compilation pour définir les points d'entrée de la dll. DEV-C++ ne reconnait, toutefois, pas ce type de fichier. Il va donc falloir rajouter un bout de code au début de chaque déclaration de fonction pour qu'elle puisse être appelée par des programmes externes.

9- Rajoutez le code suivant avant la déclaration de chaque fonction :

Extern « C » __declspec(dllexport)

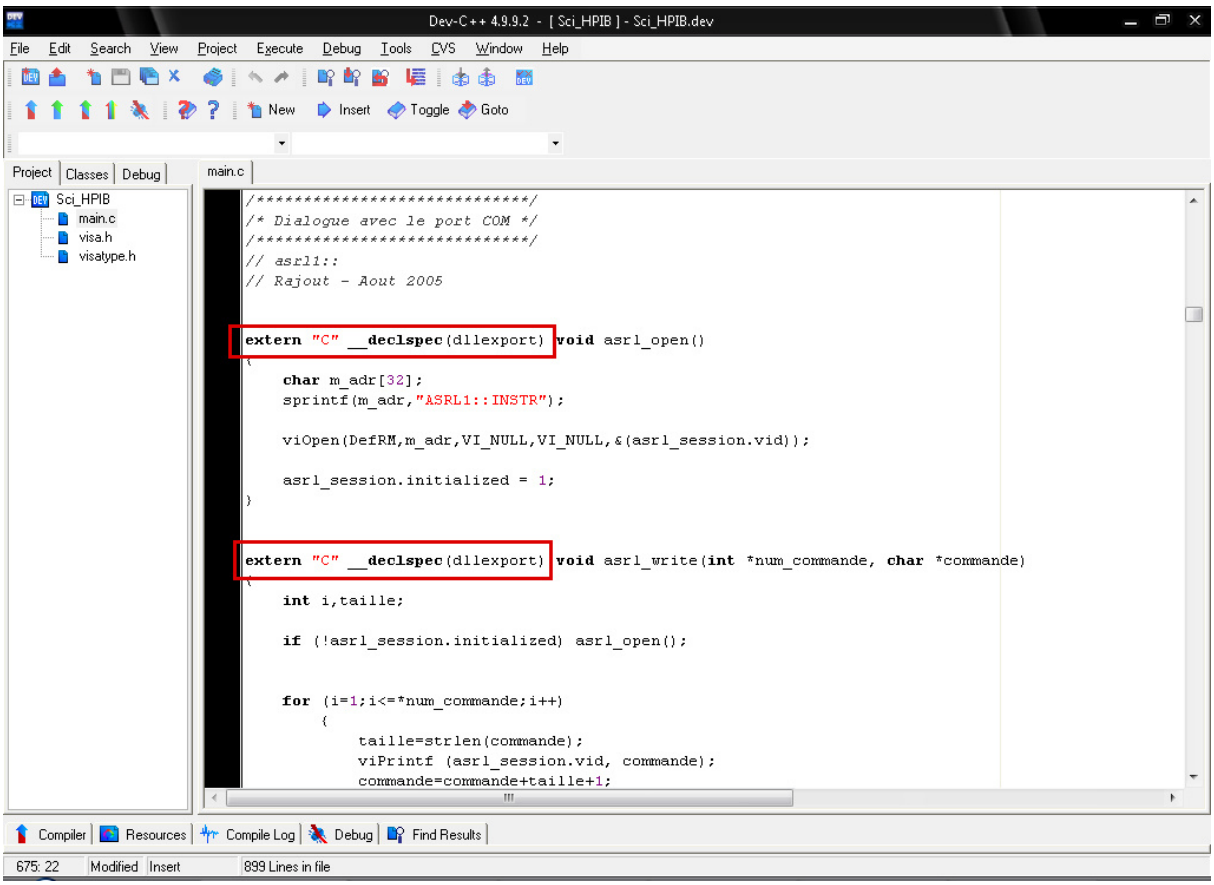

figure 6. Exportation des points d'entrée.

10- Compilez enfin la DLL (Execute>Compile) et remplacer la DLL ainsi produite par l'ancienne dans le dossier <dossier Scilab>\contrib\GPIB\routines (si une erreur de conversion de type apparaît au niveau de la fonction GPIB_read_block_short, remplacez **unsigned char *buffer** par **char *buffer**).

A ce niveau, il suffit de relancer « loader.sce », pour charger la GPIB ToolBox.

4 UTILISATION DE LA GPIB TOOLBOX

La toolbox contient plusieurs fonctions permettant de fixer la fréquence d'un instrument, sa puissance de sortie, … Ces fonctions ne fonctionnent, toutefois pas avec tout appareil doté d'une interface GPIB, on en revient donc très vite à utiliser les fonctions élémentaires que sont **GPIB_write** et **GPIB_read**.

La syntaxe à utiliser, dans ce cas, est : **GPIB_write(id,ordre)** et **GPIB_read(id,Nrep,Nbuf)**. Avec **id** l'identifiant de l'instrument (de 1 à 32), **ordre** une chaine de caractères type string représentant l'ordre à envoyer (la syntaxe dépend de l'appareil), **Nrep** le nombre de réponses attendues et **Nbu**f la taille du buffer (i.e. le nombre de caractères ASCII).

Exemple :

*GPIB_write(16, « *IDN ? »);*
Reponse=GPIB_read(16,1,30);

//Retourne la marque et le type d'instrument (plus quelques informations complémentaires).

Les autres fonctions de bas niveau (utilisable sur tout instrument GPIB) sont **GPIB_init()**, **GPIB_close()**, **GPIB_timeout(id,time)** (time en ms) et **GPIB_pause(time)** (time en s).

Deux instructions supplémentaires existent lorsque l'on souhaite adresser un ordre à un instrument doté d'une sous-adresse GPIB : **GPIB_write_sub(di1,id2,ordre)** et **GPIB_read_sub(id1,id2,Nrep,Nbuf)**.