GPIB Toolbox

Enables communications between VISA for Windows (either Agilent or National Instrument) or Linux (NI-VISA) and Scilab in order to control your instruments with the serial port, USB or the GPIB bus.

Contact: Tibault Reveyrand
Website: http://www.reveyrand.fr/GPIB.html

Objective: control Tektronix TDS2002B (USBTMC compliant) oscilloscope trough USB port

Programs: Microsoft Windows XP, Scilab 4.1.2, GPIB toolbox v. 1.46 September 2009, NI-VISA 3.4 (as installed by Tektronix TDS2002B provided software)

Installation:

- Scilab

  version 4.1.2 – default options ("Source of XML" option is not needed)

- GPIB toolbox

  Download it from http://www.reveyrand.fr/GPIB.html

  Unzip the file in your Scilab contrib directory (SCI+\contrib)

  Execute the GPIB loader located at SCI+\contrib\GPIB\loader.sce

  If everything is correct this is going to be the output from Scilab:

```
                  ___________________________________________
                                scilab-4.1.2

                         Copyright (c) 1989-2007
                     Consortium Scilab (INRIA, ENPC)
                  ___________________________________________


Startup execution:
  loading initial environment

-->@@ Loading GPIB Toolbox (v. 1.46 September 2009)
-- DLL functions loaded
shared archive loaded
Link done
-- macro functions loaded
-- online help loaded
-- gpib init
```

  and we can use functions (`help "GPIB toolbox"`) defined in GPIB toolbox.

Usage:

VISA_open – Open a VISA session

Calling Sequence

```
VISA_open(id,name)
```

Parameters

- `id` : An arbitrary number between 1 and 32 to identify the VISA session.

- `name`: String. The VISA address of the instrument

Description

This function opens a VISA session. It should be used if the interface is different that "GPIB0::xx::INSTR". For GPIB0, you can use GPIB_write() and GPIB_read().

VISA_write – Send string trough VISA to a given instrument

Calling Sequence

```
VISA_write(id,command)
```

Parameters

- `id`: An number between 1 and 32 to identify the VISA session. This number was assign by `VISA_open()`.
- `command`: ASCII sequence of commands to be send. Can be a scalar, vector or array of type 'string'.

Description

Command is a string to which carriage return, a newline and an end instruction (EOI) is automatically added when sending to GPIB bus.
???Command is usually 'string' but may be also an array of 'string'. Constitute a vector control GPIB is handy to send several commands in sequence to an instrument with a single statement 'Scilab'.???

VISA_read – Read text from selected instrument trough VISA. Character 0 ends reading of the text.
Note: It is not possible to use this function for binary reading of the data. See `VISA_read_bin()`.

Calling Sequence

```
rep = VISA_read(id,buffer_size)
```

Parameters

- `id`: An number between 1 and 32 to identify the VISA session. This number was assign by `VISA_open()`.
- `buffer_size`: A buffer size (number of characters in ASCII) for receiving the expected response. Maximum size is 4096.

Description

`rep` is a string to which carriage return, a newline and an end instruction (EOI) is automatically added.

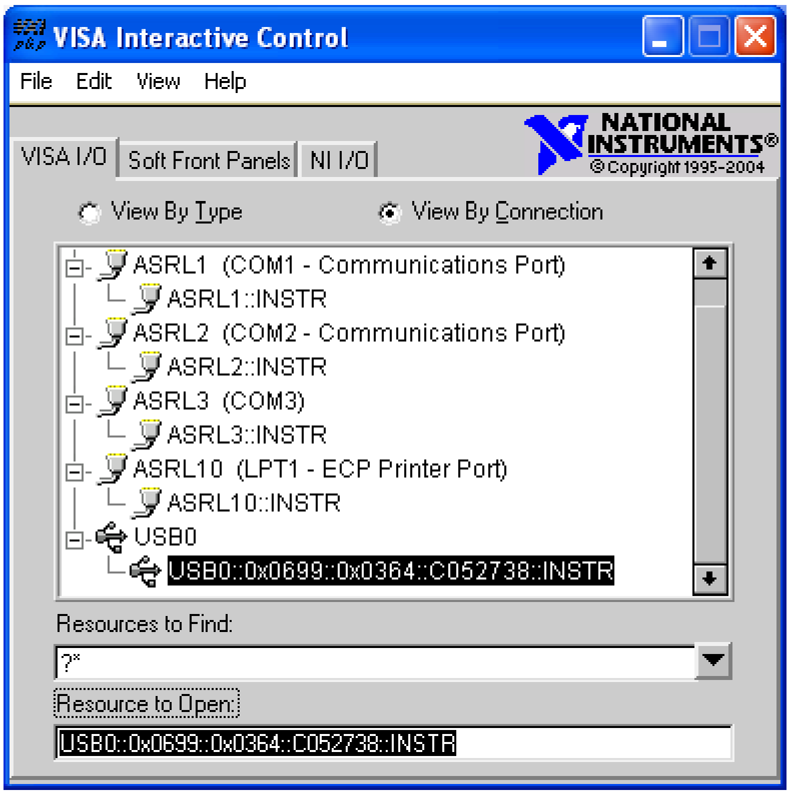
Example

```
// Get oscilloscope identification
//
VISA_open(1,"USB0::0x0699::0x0364::C052738::INSTR")
VISA_write(1,"*IDN?")
VISA_read(1,50)
 ans  =

 TEKTRONIX, TDS 2002B,C052738,CF:91.1CT FV:v22.11
```

How to find out VISA address of the instrument?

The VISA address of the instrument which is used as a `name` parameter in `VISA_open()` function can be interactively find out by using the "VISA Interactive Control" program (`NIvisaic.exe`, `NIvisaic` under GNU/Linux). In our case it is in the Programs folder -> National Intrsuments -> VISA -> VISA Interactive Control ("`C:\VXIPNP\WinNT\NIvisa\NIvisa.exe`").

For more informations please have a look into: "USB instrument control tutorial" by NI (http://zone.ni.com/devzone/cda/tut/p/id/4478).



```
ASRL1  (COM1 - Communications Port)
  ASRL1::INSTR
ASRL2  (COM2 - Communications Port)
  ASRL2::INSTR
ASRL3  (COM3)
  ASRL3::INSTR
ASRL10  (LPT1 - ECP Printer Port)
  ASRL10::INSTR
USB0
  USB0::0x0699::0x0364::C052738::INSTR
```

Resource to Open: USB0::0x0699::0x0364::C052738::INSTR

Problems during installation

If the script causes error 236 "shared archive not loaded" you might need to recompile the DLL library. The problem is due to the paths of libraries `visa.h`, `visatype.h` and `visa32.lib`, which are probably different from those entered in the time of compilation of the DLL. The next section explains how.

```
                ___________________________________________
                                  scilab-4.1.2

                           Copyright (c) 1989-2007
                        Consortium Scilab (INRIA, ENPC)
                ___________________________________________


Startup execution:
  loading initial environment
@@ Loading GPIB Toolbox (v. 1.46 September 2009)
-- DLL functions loaded
link failed for dll C:\PROGRA~1\SCILAB~1.2\contrib\GPIB\sci_gateway\GPIB_NI.dll
                "GPIB_write_binary_file"],"c");
                                 !--error 236
link: the shared archive was not loaded
at line      26 of exec file called by :
exec('loader.sce');
line      7 of exec file called by :
exec(SCI+'/contrib/GPIB/loader.sce');
line     10 of exec file called by :
  exec(SCI+'/contrib/loader.sce');
line    201 of exec file called by :
exec('SCI/scilab.star',-1);;scipad(getlongpathname('C:/PROGRA~1/SCILAB~1
```

Source: Guide de la GPIB ToolBox This document (in French) written by Julien Coudrat from Thales explain how to build the toolbox with a free C compiler (Dev C++ and MingWin).

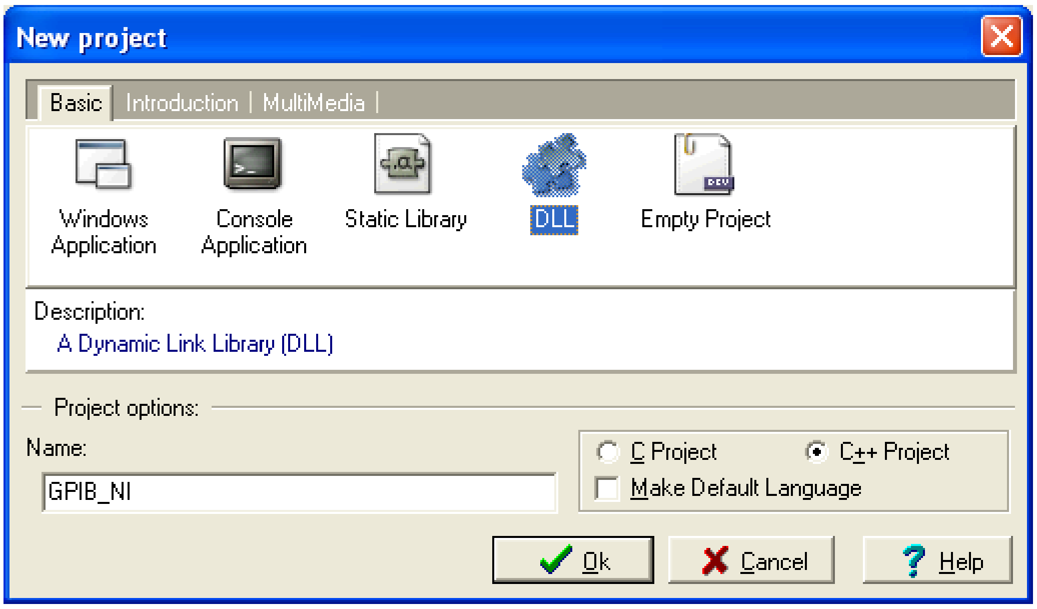
Programs: Dev-C++ (distributed under the terms of the GNU General Public License), MinGW (*MinGW runtime:* The MinGW base runtime package has been placed in the public domain, and is not governed by copyright. *w32api:* You are free to use, modify and copy this package. *MinGW profiling code:* MinGW profiling code is distributed under the terms of the GNU General Public License)

Installation:

- Dev-C++ 5.0 beta 9.2 (4.9.9.2) with Mingw/GCC 3.4.2
  We recommend to use Dev-C++ with Mingw/GCC compiler.

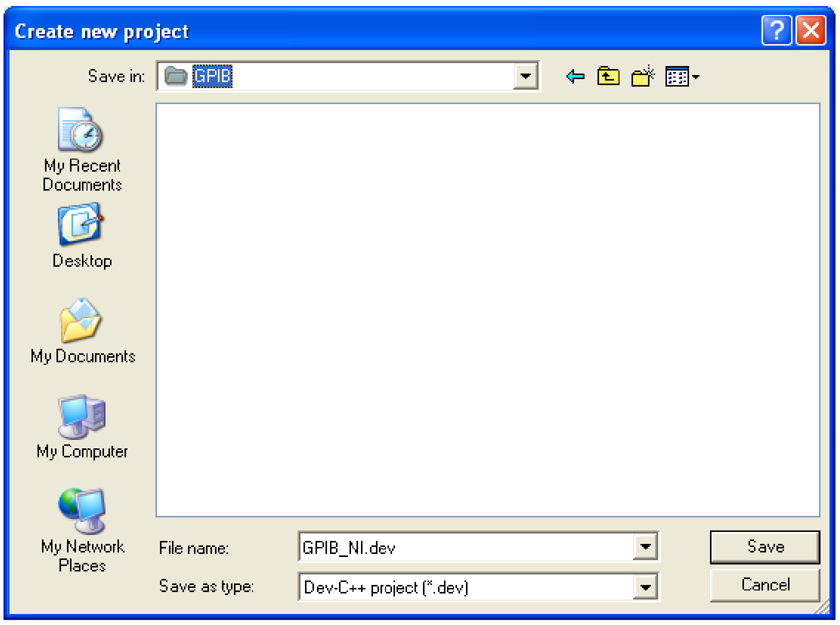
Compilation:

1. Locate the file `visa.h`, `visatype.h` and `visa32.lib`. (example: "`C:\VXIPNP\WinNT\include`" and "`C:\VXIPNP\WinNT\lib\msc`").
2. Open Dev-C++, create a new project (File -> New -> Project...) "DLL" with a name "GPIB_NI" and save it in some folder (for example "`C:\Dev-Cpp\GPIB\`").



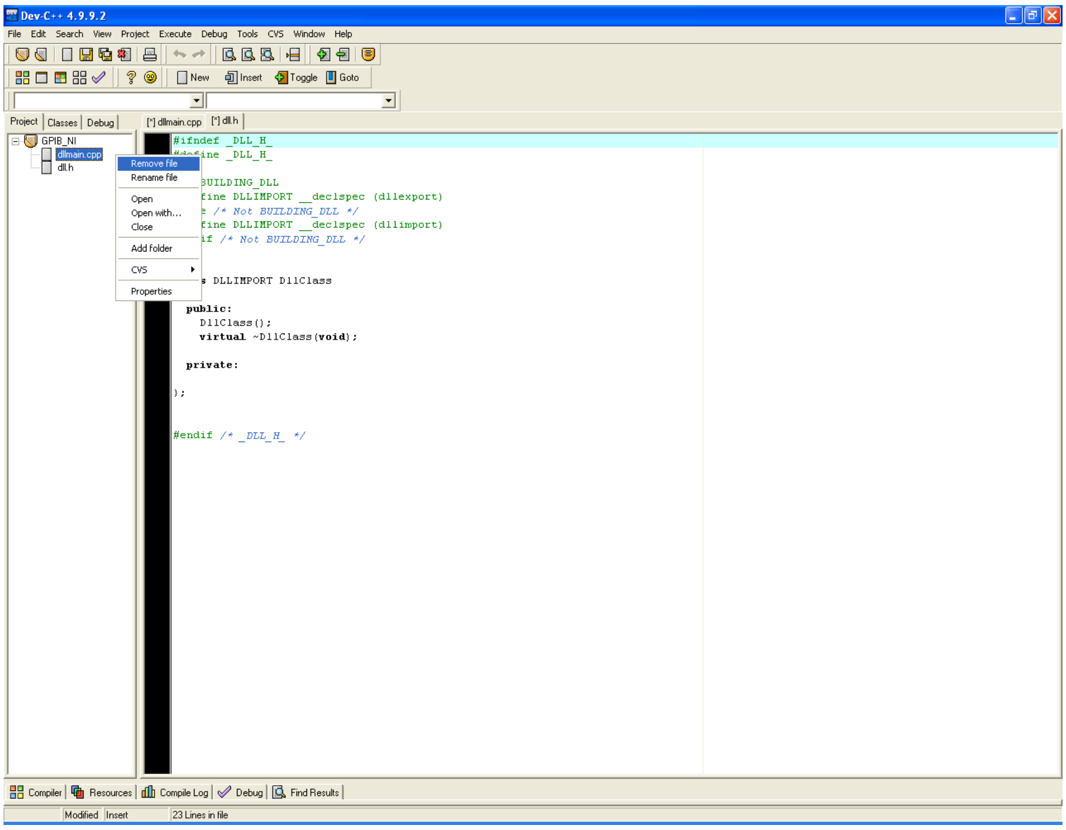
Description:
A Dynamic Link Library (DLL)

Project options:
Name: GPIB_NI
C Project [✓ C++ Project]
Make Default Language



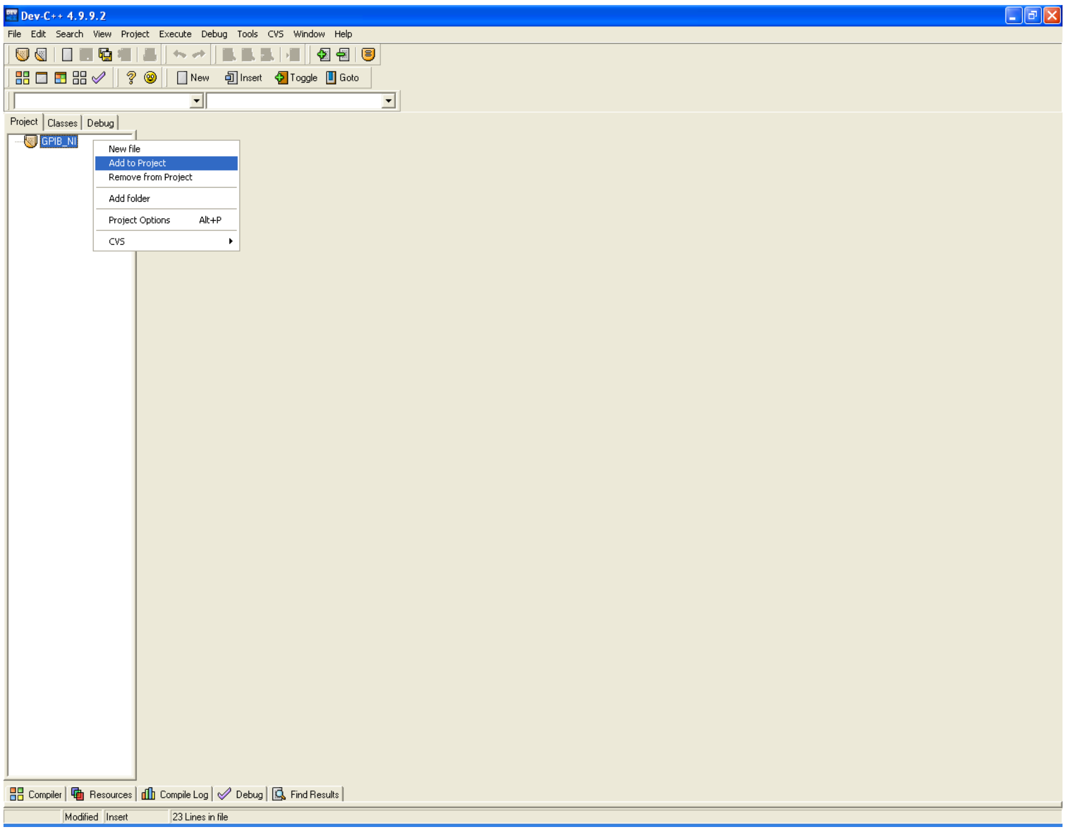
Save in: GPIB
File name: GPIB_NI.dev
Save as type: Dev-C++ project (*.dev)

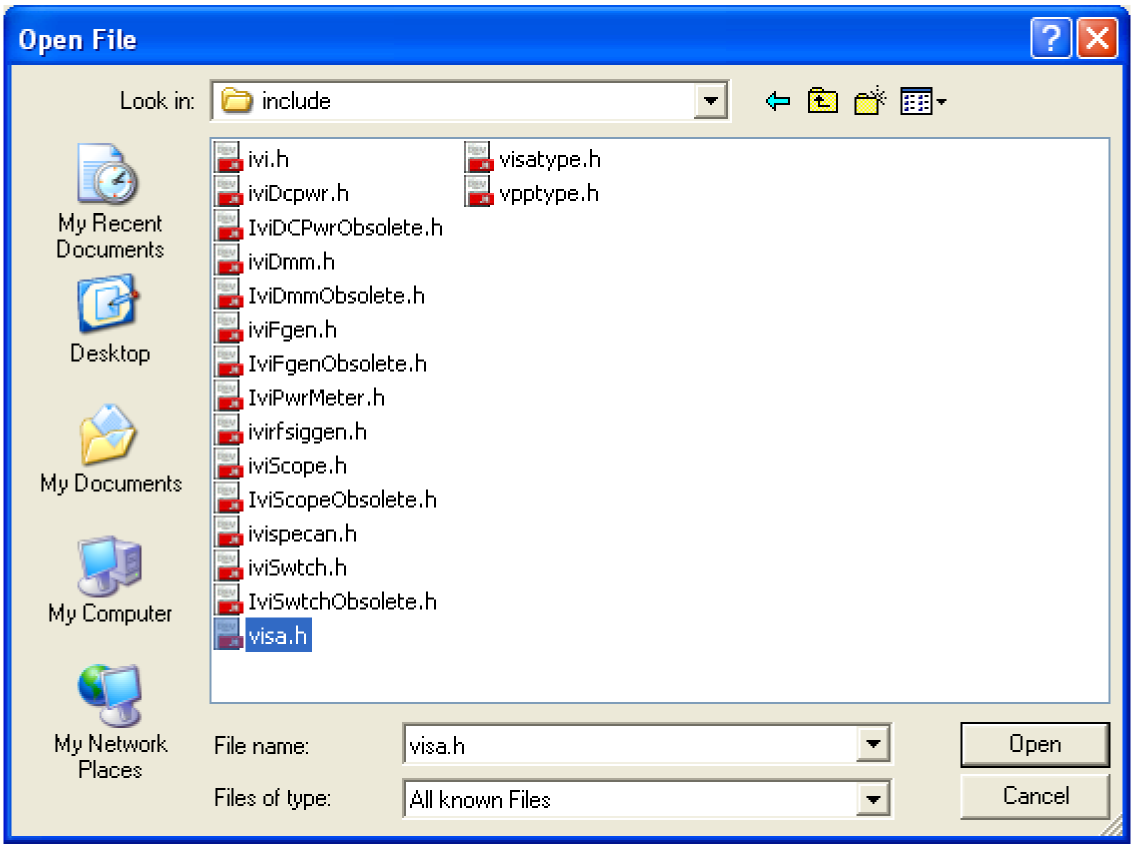
3. Delete files "`dllmain.cpp`" and "`dll.h`" from the project tree (right click -> Remove file) without saving changes.


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
[*] dllmain.cpp [*] dll.h
GPIB_NI
dllmain.cpp
dll.h
Remove file
Rename file
Open
Open with...
Close
Add folder
CVS
Properties
Compiler Resources Compile Log Debug Find Results
Modified Insert 23 Lines in file

```
#ifndef _DLL_H_
#define _DLL_H_

BUILDING_DLL
fine DLLIMPORT __declspec (dllexport)
e /* Not BUILDING_DLL */
fine DLLIMPORT __declspec (dllimport)
if /* Not BUILDING_DLL */


s DLLIMPORT DllClass

  public:
    DllClass();
    virtual ~DllClass(void);

  private:

};


#endif /* _DLL_H_ */
```

4. Add the files `visa.h`, `visatype.h` (as previously located, see item 1.) and `main.c` (`SCI+\contrib\GPIB\sci_gateway\main.c`) in the project tree (right click -> “Add to project”)


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
GPIB_NI
New file
Add to Project
Remove from Project
Add folder
Project Options Alt+P
CVS
Compiler Resources Compile Log Debug Find Results
Modified Insert 23 Lines in file

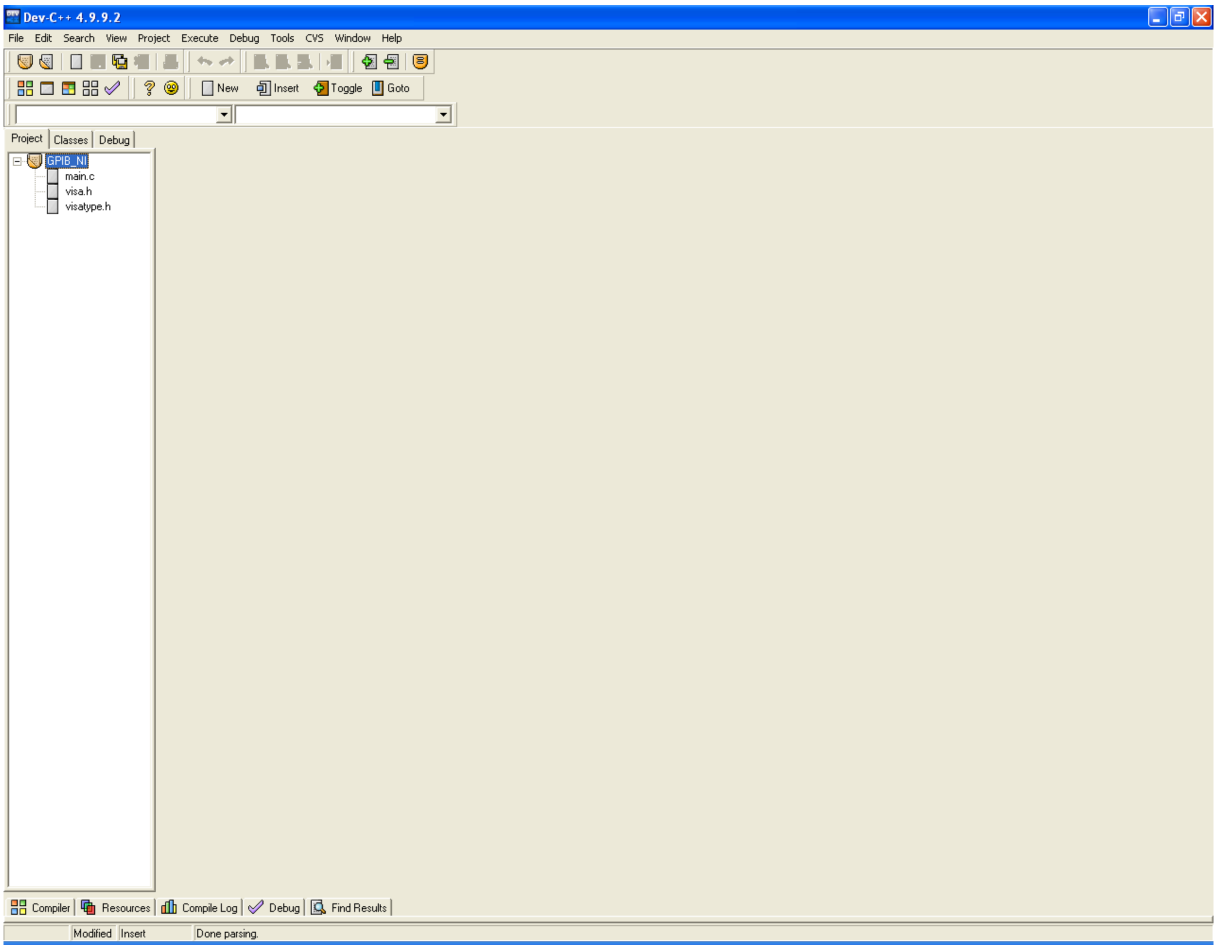


Look in: include

ivi.h
iviDcpwr.h
IviDCPwrObsolete.h
iviDmm.h
IviDmmObsolete.h
iviFgen.h
IviFgenObsolete.h
IviPwrMeter.h
ivirfsiggen.h
iviScope.h
IviScopeObsolete.h
ivispecan.h
iviSwtch.h
IviSwtchObsolete.h
visa.h
visatype.h
vpptype.h

File name: visa.h

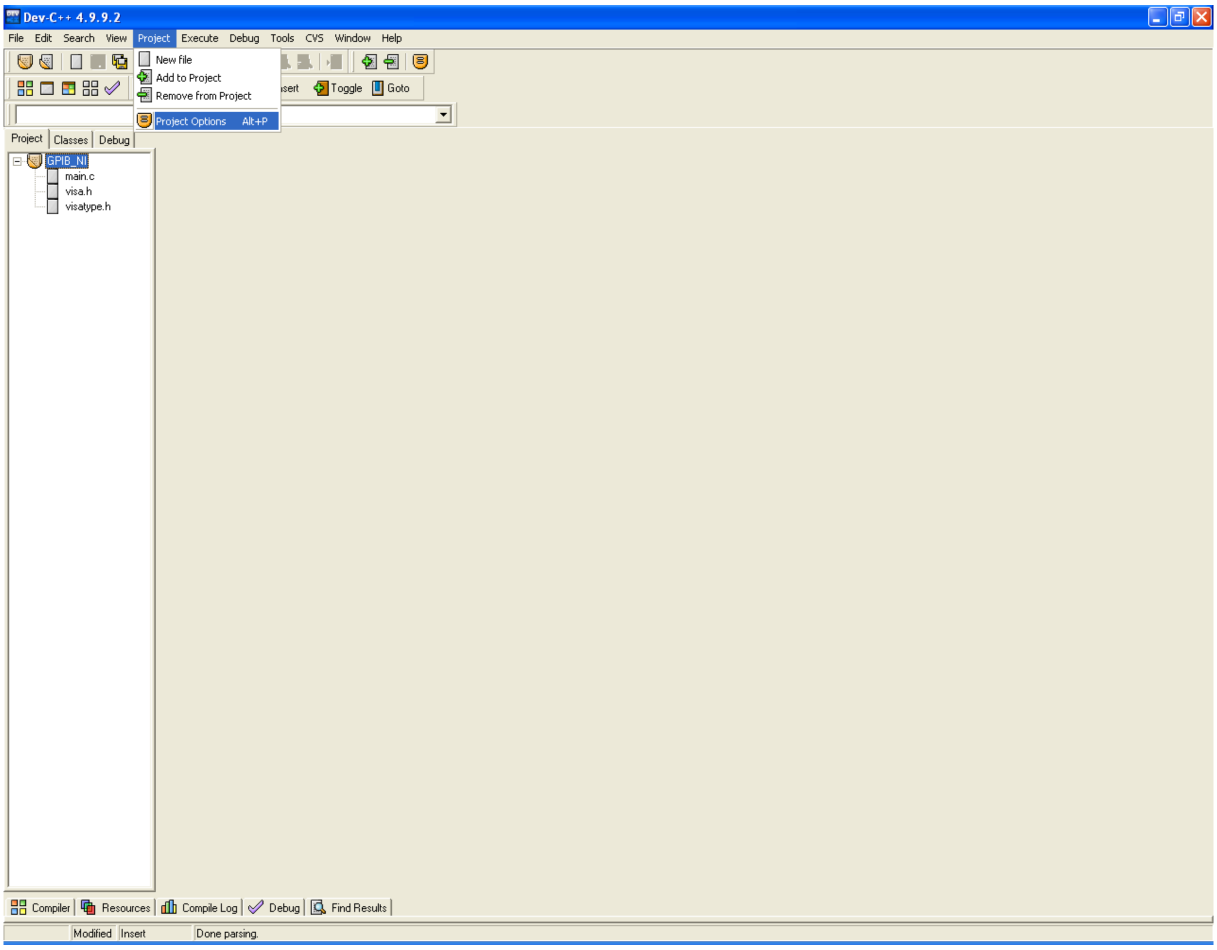
Files of type: All known Files

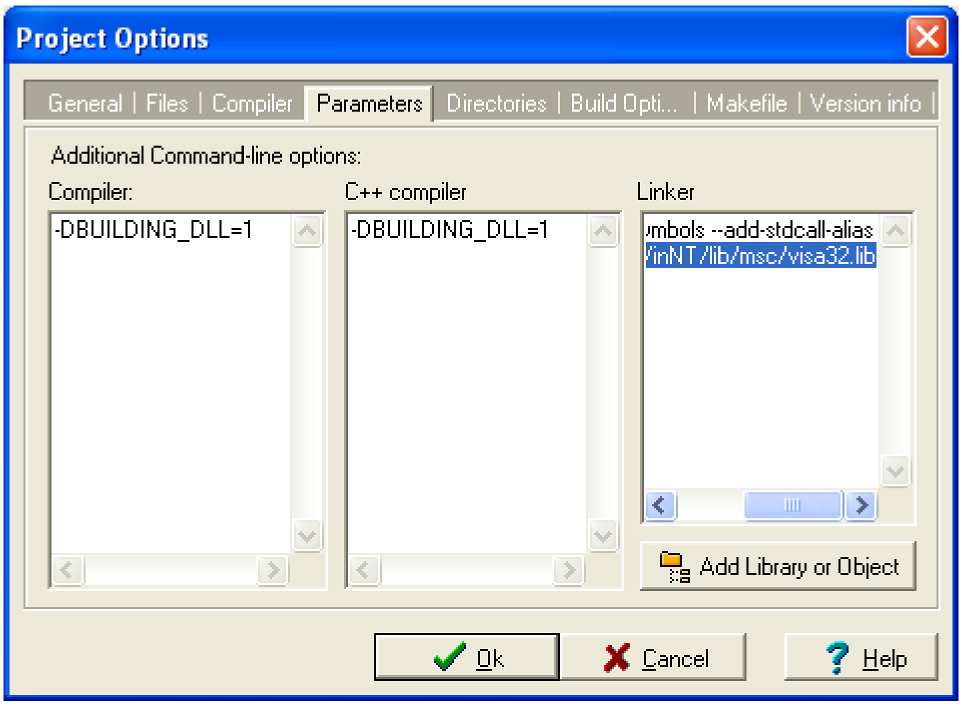


GPIB_NI
main.c
visa.h
visatype.h

Modified | Insert | Done parsing.

5. Choose menu "Project" -> "Project Options". In the "Parameters" tab, press "Add Library or Object" and add `visa32.lib`. (as previously located, see item 1.).


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New file
Add to Project
Remove from Project
Project Options Alt+P
Toggle
Goto
Project Classes Debug
GPIB_NI
main.c
visa.h
visatype.h
Compiler Resources Compile Log Debug Find Results
Modified Insert Done parsing.


Project Options
General Files Compiler Parameters Directories Build Opti... Makefile Version info
Add Library or Object
Ok
Cancel
Help

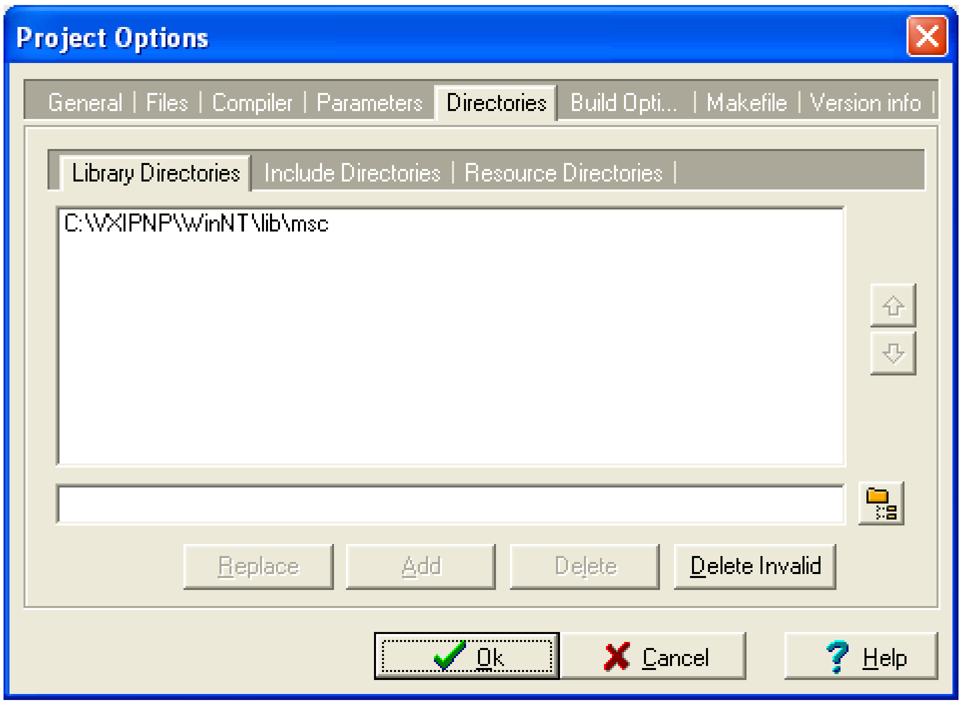
Additional Command-line options:

Compiler:
```
-DBUILDING_DLL=1
```

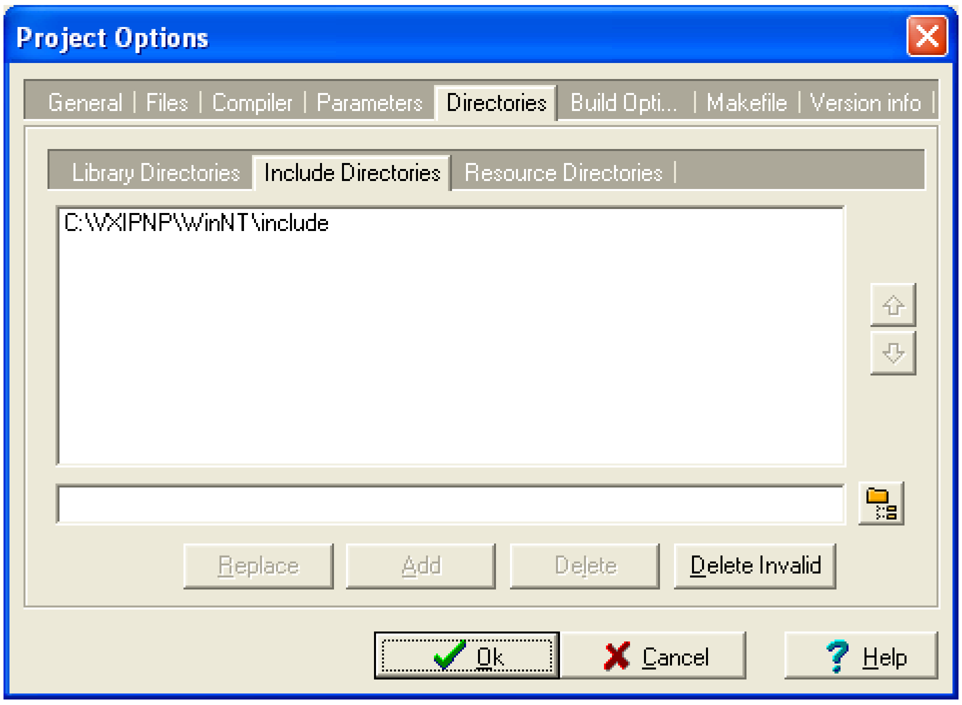
C++ compiler
```
-DBUILDING_DLL=1
```

Linker
```
mbols --add-stdcall-alias
/inNT/lib/msc/visa32.lib
```

6. In the “Directories” tab add the path to `visa32.lib` in the subcategory “Library Directories” and the path to `visa.h` and `visatype.h` in the subcategory “Include Directories”.

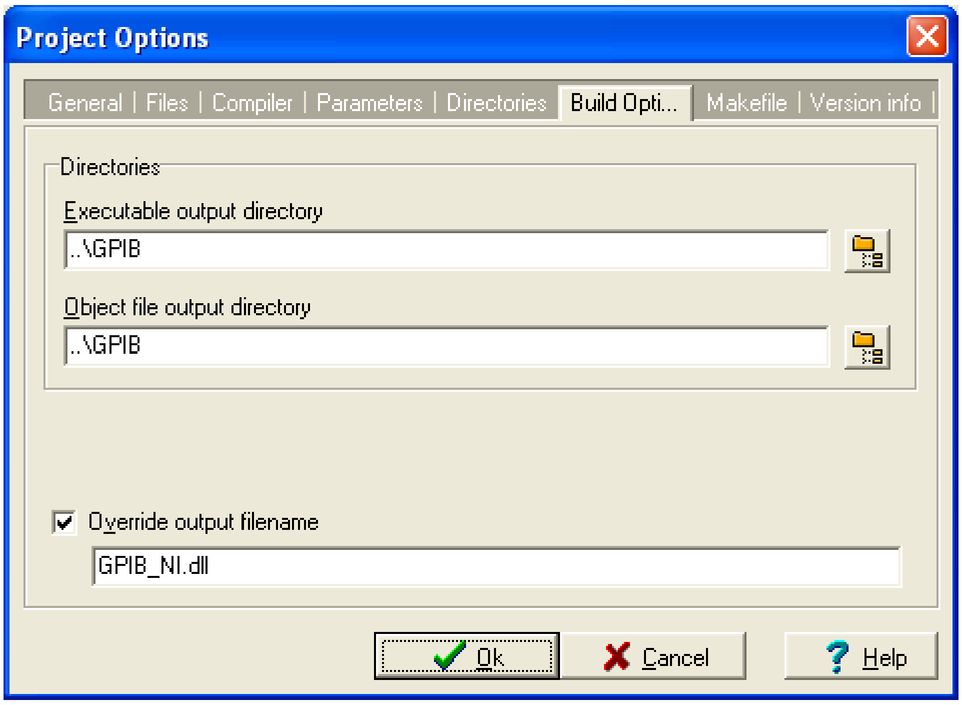

Project Options
General | Files | Compiler | Parameters | Directories | Build Opti... | Makefile | Version info
Library Directories | Include Directories | Resource Directories
Replace
Add
Delete
Delete Invalid
Ok
Cancel
Help

C:\VXIPNP\WinNT\lib\msc


Project Options
General | Files | Compiler | Parameters | Directories | Build Opti... | Makefile | Version info
Library Directories | Include Directories | Resource Directories
Replace
Add
Delete
Delete Invalid
Ok
Cancel
Help

C:\VXIPNP\WinNT\include

7. In the “Build Options” tab, add the path of your project in the field “Executable output directory” and “Object file output directory” (for example `"C:\Dev-Cpp\GPIB\"` as in item 2.) and check the option “Override output filename”.


Project Options
General | Files | Compiler | Parameters | Directories | Build Opti... | Makefile | Version info
Ok
Cancel
Help

Directories

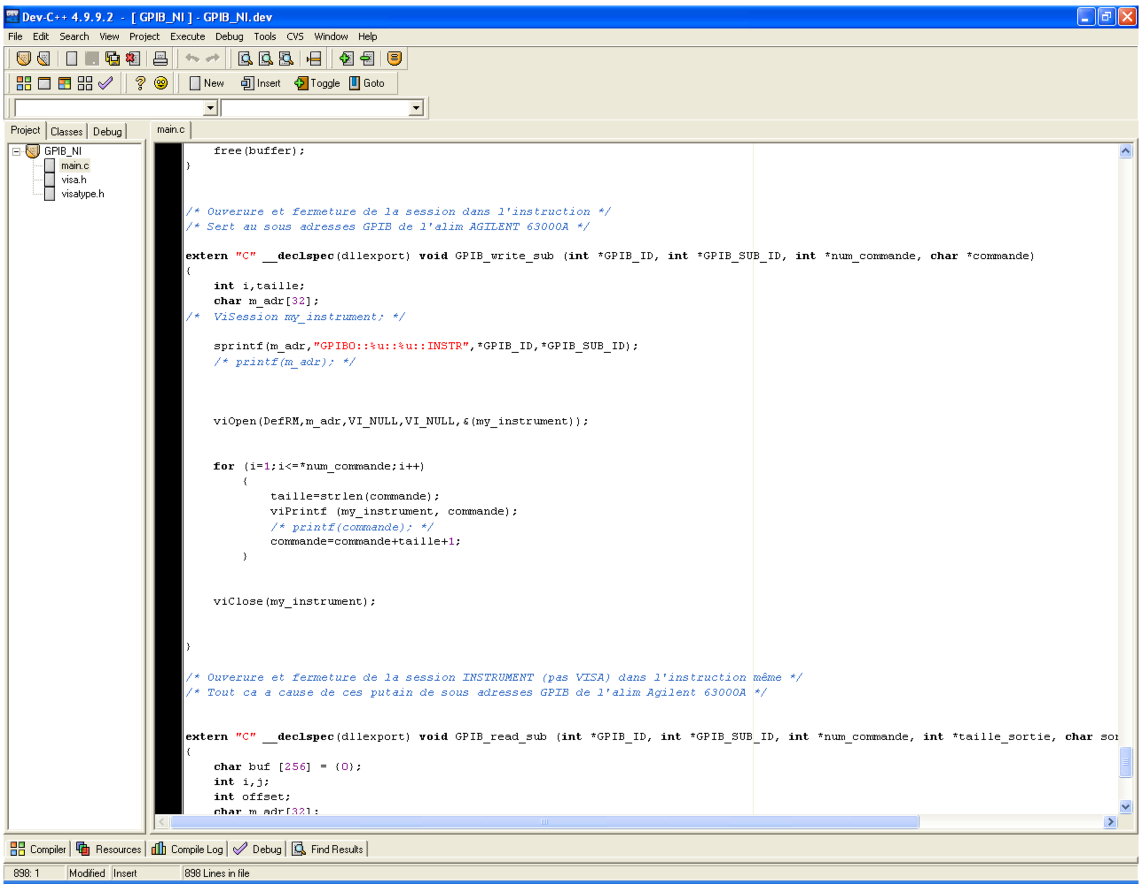
Executable output directory: ..\GPIB

Object file output directory: ..\GPIB

[✓] Override output filename: GPIB_NI.dll

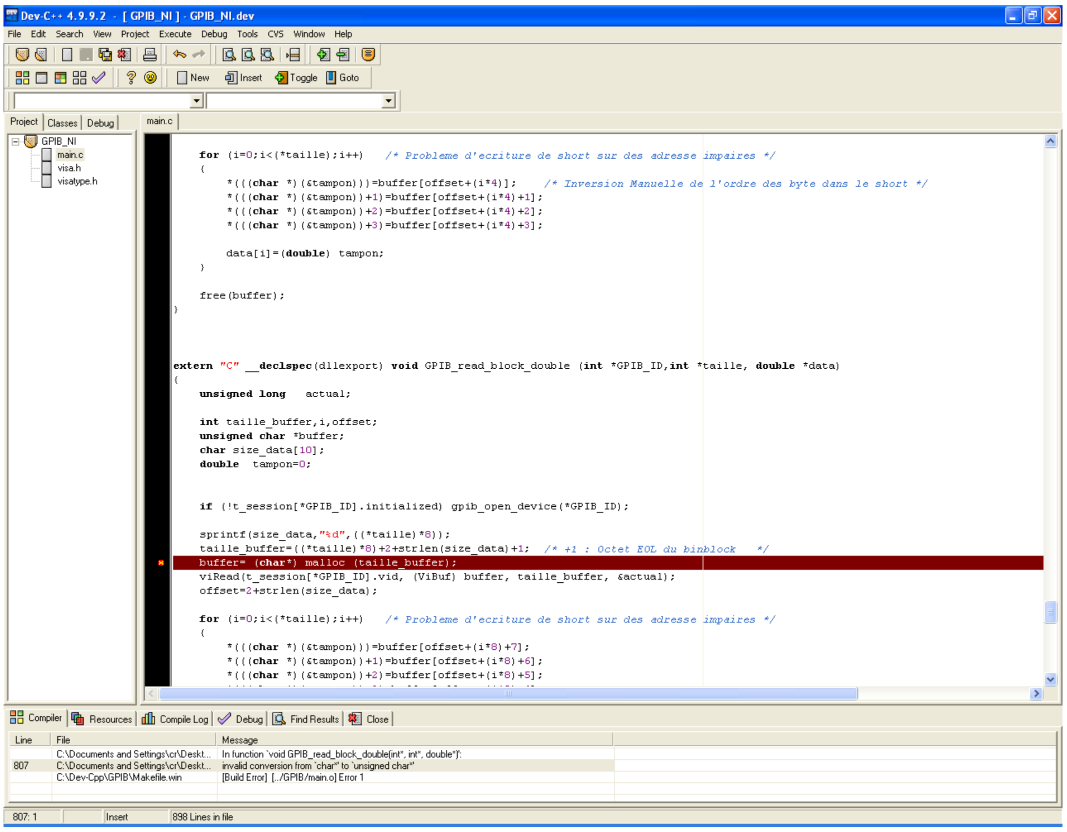
8. Apply the changes by clicking on "OK".
9. Another file included with the GPIB toolbox `main.def` is used to define DLL's functions when compiling the DLL. DEV-C++ does recognize this file and it is therefore necessary to define them in the source code by **`extern "C" __declspec(dllexport)`** keyword. It will be therefore necessary to add a bit of code at the beginning of each function declaration. Add the following code before the declaration of each function in the `main.c` file:

```
extern "C" __declspec(dllexport)
```


Dev-C++ 4.9.9.2 - [GPIB_NI] - GPIB_NI.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
GPIB_NI
main.c
visa.h
visatype.h
main.c
Compiler Resources Compile Log Debug Find Results
898: 1 Modified Insert 898 Lines in file

```
    free(buffer);
}


/* Ouverure et fermeture de la session dans l'instruction */
/* Sert au sous adresses GPIB de l'alim AGILENT 63000A */

extern "C" __declspec(dllexport) void GPIB_write_sub (int *GPIB_ID, int *GPIB_SUB_ID, int *num_commande, char *commande)
{
    int i,taille;
    char m_adr[32];
/*  ViSession my_instrument; */

    sprintf(m_adr,"GPIB0::%u::%u::INSTR",*GPIB_ID,*GPIB_SUB_ID);
    /* printf(m_adr); */



    viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(my_instrument));


    for (i=1;i<=*num_commande;i++)
        {
            taille=strlen(commande);
            viPrintf (my_instrument, commande);
            /* printf(commande); */
            commande=commande+taille+1;
        }


    viClose(my_instrument);


}

/* Ouverure et fermeture de la session INSTRUMENT (pas VISA) dans l'instruction même */
/* Tout ca a cause de ces putain de sous adresses GPIB de l'alim Agilent 63000A */


extern "C" __declspec(dllexport) void GPIB_read_sub (int *GPIB_ID, int *GPIB_SUB_ID, int *num_commande, int *taille_sortie, char sor
{
    char buf [256] = {0};
    int i,j;
    int offset;
    char m_adr[32];
```

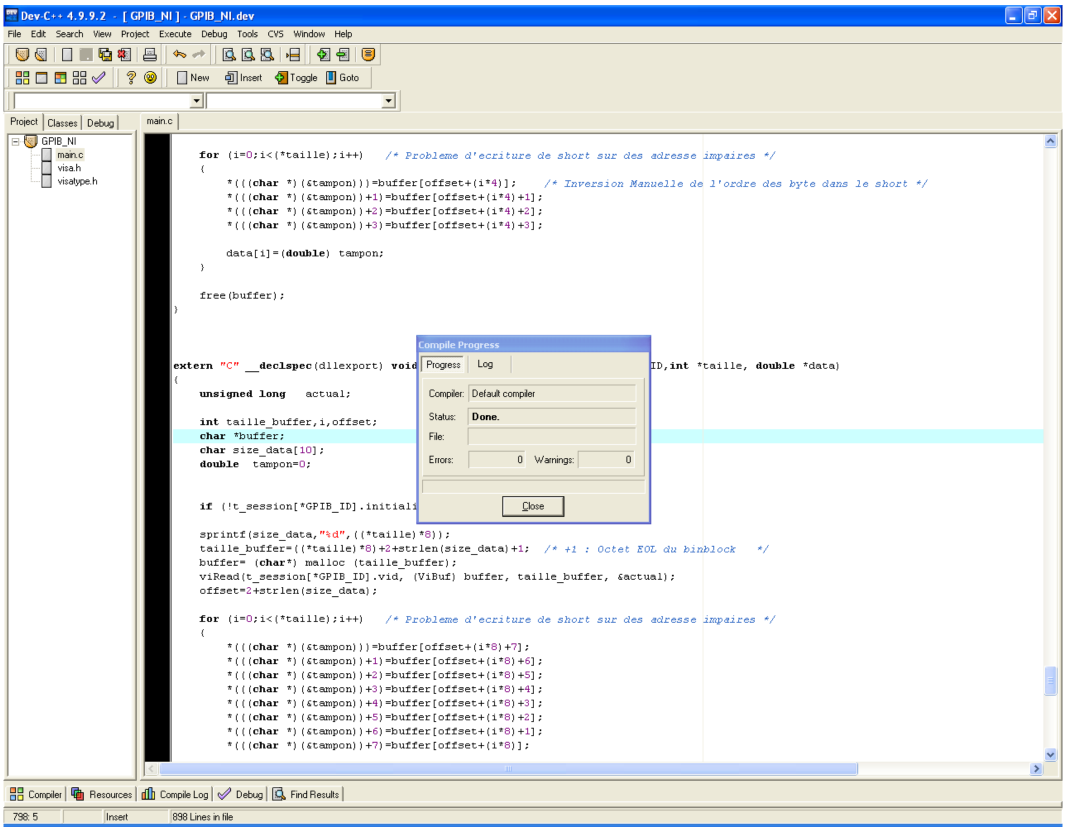
10. Finally compile the DLL (menu “Execute” -> “Compile”) and replace original DLL file ("SCI+\contrib\GPIB\sci_gateway\GPIB_NI.dll") by newly created DLL file ("C:\Dev-Cpp\GPIB\GPIB_NI.dll"). If there is a compilation error in the function GPIB_read_block_short replace the code `unsigned char *buffer` by `char *buffer` and compile the DLL once more.


Dev-C++ 4.9.9.2 - [GPIB_NI] - GPIB_NI.dev

```
    for (i=0;i<(*taille);i++)   /* Probleme d'ecriture de short sur des adresse impaires */
    {
        *(((char *)(&tampon)))=buffer[offset+(i*4)];    /* Inversion Manuelle de l'ordre des byte dans le short */
        *(((char *)(&tampon))+1)=buffer[offset+(i*4)+1];
        *(((char *)(&tampon))+2)=buffer[offset+(i*4)+2];
        *(((char *)(&tampon))+3)=buffer[offset+(i*4)+3];

        data[i]=(double) tampon;
    }

    free(buffer);
}



extern "C" __declspec(dllexport) void GPIB_read_block_double (int *GPIB_ID,int *taille, double *data)
{
    unsigned long   actual;

    int taille_buffer,i,offset;
    unsigned char *buffer;
    char size_data[10];
    double  tampon=0;


    if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

    sprintf(size_data,"%d",((*taille)*8));
    taille_buffer=((*taille)*8)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock   */
    buffer= (char*) malloc (taille_buffer);
    viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
    offset=2+strlen(size_data);

    for (i=0;i<(*taille);i++)   /* Probleme d'ecriture de short sur des adresse impaires */
    {
        *(((char *)(&tampon)))=buffer[offset+(i*8)+7];
        *(((char *)(&tampon))+1)=buffer[offset+(i*8)+6];
        *(((char *)(&tampon))+2)=buffer[offset+(i*8)+5];
```

Line	File	Message
	C:\Documents and Settings\cr\Deskt...	In function `void GPIB_read_block_double(int*, int*, double*)':
807	C:\Documents and Settings\cr\Deskt...	invalid conversion from `char*' to `unsigned char*'
	C:\Dev-Cpp\GPIB\Makefile.win	[Build Error] [../GPIB/main.o] Error 1

807: 1 | Insert | 898 Lines in file

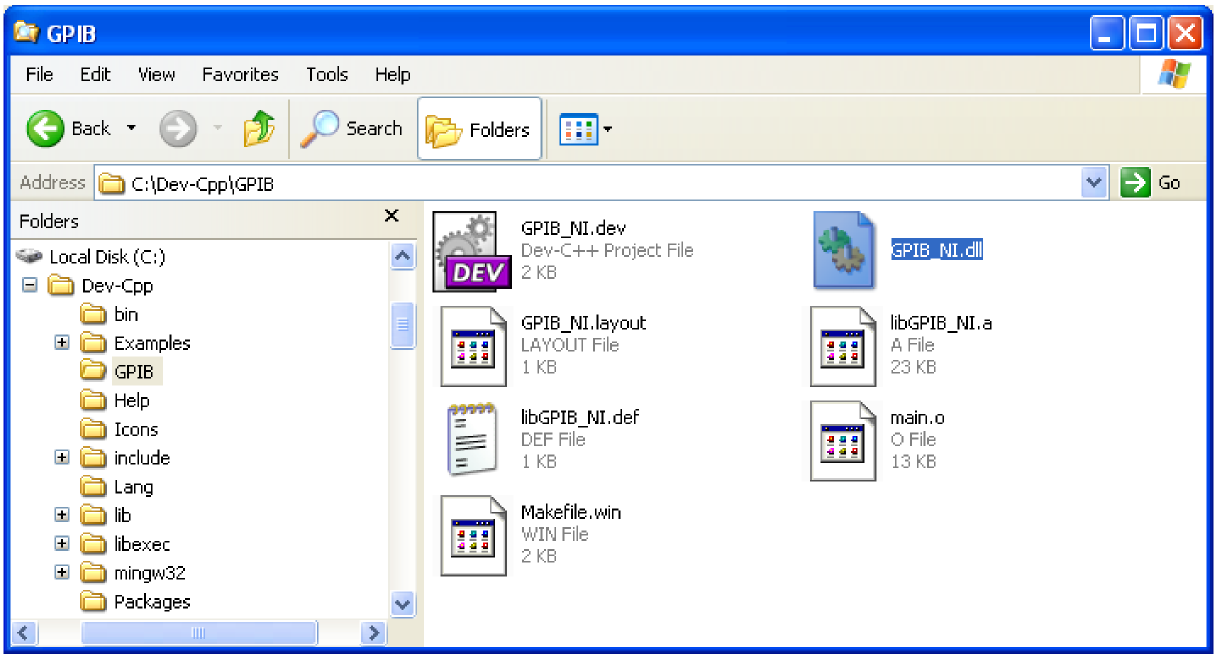

Dev-C++ 4.9.9.2 - [GPIB_NI] - GPIB_NI.dev

```
    for (i=0;i<(*taille);i++)   /* Probleme d'ecriture de short sur des adresse impaires */
    {
        *(((char *)(&tampon)))=buffer[offset+(i*4)];    /* Inversion Manuelle de l'ordre des byte dans le short */
        *(((char *)(&tampon))+1)=buffer[offset+(i*4)+1];
        *(((char *)(&tampon))+2)=buffer[offset+(i*4)+2];
        *(((char *)(&tampon))+3)=buffer[offset+(i*4)+3];

        data[i]=(double) tampon;
    }

    free(buffer);
}



extern "C" __declspec(dllexport) void                                ID,int *taille, double *data)
{
    unsigned long   actual;

    int taille_buffer,i,offset;
    char *buffer;
    char size_data[10];
    double  tampon=0;


    if (!t_session[*GPIB_ID].initiali

    sprintf(size_data,"%d",((*taille)*8));
    taille_buffer=((*taille)*8)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock   */
    buffer= (char*) malloc (taille_buffer);
    viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
    offset=2+strlen(size_data);

    for (i=0;i<(*taille);i++)   /* Probleme d'ecriture de short sur des adresse impaires */
    {
        *(((char *)(&tampon)))=buffer[offset+(i*8)+7];
        *(((char *)(&tampon))+1)=buffer[offset+(i*8)+6];
        *(((char *)(&tampon))+2)=buffer[offset+(i*8)+5];
        *(((char *)(&tampon))+3)=buffer[offset+(i*8)+4];
        *(((char *)(&tampon))+4)=buffer[offset+(i*8)+3];
        *(((char *)(&tampon))+5)=buffer[offset+(i*8)+2];
        *(((char *)(&tampon))+6)=buffer[offset+(i*8)+1];
        *(((char *)(&tampon))+7)=buffer[offset+(i*8)];
```

Compile Progress

Compiler: Default compiler
Status: Done.
File:
Errors: 0 Warnings: 0

798: 5 | Insert | 898 Lines in file



11. Go to Scilab and once more execute the GPIB loader located at `SCI+\contrib\GPIB\loader.sce`. If everything is correct this is going to be the output from Scilab:

```
              ___________________________________________
                               scilab-4.1.2

                       Copyright (c) 1989-2007
                   Consortium Scilab (INRIA, ENPC)
              ___________________________________________


Startup execution:
  loading initial environment

-->@@ Loading GPIB Toolbox (v. 1.46 September 2009)
-- DLL functions loaded
shared archive loaded
Link done
-- macro functions loaded
-- online help loaded
-- gpib init
```

VISA read binary function

1. We need to define a function which will read binary data trough VISA bus. We added this function into main.c file:

```
extern "C" __declspec(dllexport) void visa_read_bin (int *VISA_ID,int
*taille, double *data, int *delka)
{
 int   i;
 unsigned long       actual;
 char *buffer;

 buffer= (char*) malloc (*taille);
```

```
        if (v_session[*VISA_ID].initialized)
        {
        viRead (v_session[*VISA_ID].vid, (ViBuf) buffer, *taille, &actual);


            for (i=0;i<(actual);i++)
                   data[i]=(double) buffer[i];

        free(buffer);
        *delka=(int) actual;
        }
}
```

2. Moreover we need to create a Scilab function `VISA_read_bin.sci` in `SCI+\GPIB\macros\` directory:

```
function mat=VISA_read_bin(id,buffer_size)
  data=linspace(0,0,buffer_size);
  delka=0;
  [mat,delka]=fort("visa_read",id,1,"i",buffer_size,2,"i",data,3,"d",delka
,4,"i","out",[buffer_size,1],3,"d",[1,1],4,"i");
  mat=mat(1:delka);
endfunction
```

3. We need also to modify file `loader.sci` (add `"VISA_read_bin";` line) in `SCI+\GPIB\sci_gateway\` directory:

```
mode(-1);
pathL=get_absolute_file_path('loader.sce');
printf('-- DLL functions loaded\n');
link(pathL+"GPIB_NI.dll",[  "visa_open";
              "visa_close";
              "visa_write";
              "visa_read";
              "visa_read_bin";
              "gpib_init";
              "gpib_close";
              "gpib_set_timout";
              "GPIB_write";
              "GPIB_read";
              "asrl_write";
              "asrl_read";
              "AWG2021_save_signal";
              "AWG520_save_signal";
              "GPIB_read_block_short";
              "GPIB_read_block_byte";
              "GPIB_write_sub";
              "GPIB_read_sub";
              "GPIB_read_block_float";
              "GPIB_read_block_floatNI";
              "GPIB_read_block_double";
              "GPIB_read_data";
              "GPIB_read_stb";
              "GPIB_write_binary_file"],"c");
clear pathL
```

4. Then run `SCI+\contrib\GPIB\builder.sci` and `SCI+\contrib\GPIB\loader.sci` files.

VISA_read_bin – Read binary data from selected instrument trough VISA.

Calling Sequence

```
bin_data = VISA_read_bin(id,buffer_size)
```

Parameters

- `id`: An number between 1 and 32 to identify the VISA session. This number was assign by `VISA_open()`.
- `buffer_size`: A buffer size (number of bytes to read). If the real buffer contains less then bytes, `bin_data` will contain only data in a buffer. When there is more data in a buffer only `buffer_size` is read and the rest of the data stays in a buffer queue.

Description

`bin_data` contains binary data from the VISA bus assigned to given instrument.

Example

```
// Get oscilloscope identification and read data from channel 1
//
VISA_open(1,"USB0::0x0699::0x0364::C052738::INSTR")
VISA_write(1,"*IDN?")
VISA_read_bin(1,100)
 ans  =

 TEKTRONIX, TDS 2002B,C052738,CF:91.1CT FV:v22.11

VISA_write(1, ':DATA:ENCDG SRI;SOURCE CH1;START 1;STOP 2500;WIDTH 1;');
VISA_write(1, ':DATA:SOURCE CH1;:curve?');
ch1=VISA_read_bin(1,2507);

ascii(ch1)
 ans  =

 TEKTRONIX, TDS 2002B,C052738,CF:91.1CT FV:v22.11
```

```
/* ********************************** */
/* *                                * */
/* * Programme C                    * */
/* * ===========                    * */
/* *                                * */
/* * DLL de communication entre     * */
/* * scilab et la couche VISA       * */
/* *                                * */
/* *                                * */
/* * Derniere modif - Fevrier 2007  * */
/* ********************************** */


// Version 1.1 (Sept 2005): Gestion de l'interface VISA (incl. TCP IP)
// Version 1.2 (Fev 2007) : Lecture des binblocks flottants
// 1.4 (Mars 2007)

/* ********************************** */
/* Création de la librairie           */
/* ---------------------------------- */

/*************************************/
/* Sous Windows :                     */
/*
Localiser les fichiers visa.h et visa.lib
ils sont fournis avec NI-VISA et Agilent VISA pour les version Windows

Pour la création de la DLL, sous Microsoft Visual C

* Project > Project Setting > Onglet "Link" > Object / library modules : AJOUTER VISA32.lib

* Tools > Option > Onglet Directories > Show directories for Include files : Mettre le repertoire ou
se trouve VISA.h (ex: C:\Program Files\VISA\Winnt\include


* Tools > Option > Onglet Directories > Show directories for Librairy files : Mettre le repertoire
ou se trouve VISA32.lib (ex: C:\Program Files\VISA\Winnt\lib\msc

/ *********************************** /


/*************************************/
/* Sous Linux :                     */
/*
Localiser les fichiers visa.h et visa.lib
ils sont fournis avec NI-VISA

Pour la création de la librairie, avec GCC


* Comilation de la librairie
gcc -c -fPIC main.c

* Edition des liens
gcc -shared main.o /usr/local/vxipnp/linux/bin/libvisa.so -o Sci_HPIB.so

/ *********************************** /




/*  Pour Windows, remplacer cette ligne par */
#include <visa.h>

/* Pour Linux */
/*  #include "/usr/local/vxipnp/linux/include/visa.h" */


#include <stdio.h>
#include <string.h>
#include <stdlib.h>


//Session VISA par défaut
static ViSession DefRM;
static long ltimout;

//Tableau des sessions gpib
```

```
#define NB_ADR_GPIB 32
struct visa_gpib_device_session
{
        ViSession vid;
        short initialized;
};


static struct visa_gpib_device_session t_session[NB_ADR_GPIB];

static struct visa_gpib_device_session asrl_session;

/* ====================
    Version 1.1
        -----------
        ADD ON : VISA direct (incl Socket Raw)
        */

//Tableau des sessions visa
#define NB_ADR_VISA 32
struct visa_device_session
{
        ViSession vid;
        char name[200];
        short initialized;
};
static struct visa_device_session v_session[NB_ADR_GPIB];

/* Cette variable ne sert que pour Write_sub et Read_sub (instrument à 2 adresses) */
static ViSession my_instrument;



/***************/
/*  GPIB_INIT  */
/***************/
extern "C" __declspec(dllexport) void gpib_init(char *a)
{
        short i;

        viOpenDefaultRM(&DefRM);

        printf("\n*=====================*\n");
        printf("* Scilab GPIB Toolbox *\n");
        printf("*              v. 1.4 *\n");
        printf("*---------------------*\n");
        printf("*  Tibault Reveyrand  *\n");
        printf("*     CNES / IRCOM    *\n");
        printf("*=====================*\n");
        printf("\n");
        printf("http://membres.lycos.fr/treveyrand");
        printf("\n");

        /* Tableau des Sessions GPIB */
        for (i=0; i<NB_ADR_GPIB; i++)
        {
                t_session[i].initialized = 0;
                t_session[i].vid = VI_NULL;
        }

        /* 1 Session COM */
        asrl_session.initialized = 0;
        asrl_session.vid = VI_NULL;

        /* Tableau des Sessions VISA (autre que GPIB0:...) */
        for (i=0; i<NB_ADR_VISA; i++)
        {
                v_session[i].initialized = 0;
                v_session[i].vid = VI_NULL;
        }




//      atexit(gpib_close());
}
```

```
/*****************************/
/* Dialogue avec le port COM */
/*****************************/
// asrl1::
// Rajout - Aout 2005


extern "C" __declspec(dllexport) void asrl_open()
{
        char m_adr[32];
        sprintf(m_adr,"ASRL1::INSTR");

        viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(asrl_session.vid));

        asrl_session.initialized = 1;
}


extern "C" __declspec(dllexport) void asrl_write(int *num_commande, char *commande)
{
        int i,taille;

        if (!asrl_session.initialized) asrl_open();


        for (i=1;i<=*num_commande;i++)
                {
                        taille=strlen(commande);
                        viPrintf (asrl_session.vid, commande);
                        commande=commande+taille+1;
                }


}

extern "C" __declspec(dllexport) void asrl_read(int *num_commande, int *taille_sortie, char
sortie[])
{
        // char buf [256] = {0};
        char buf [4096] = {0};
        int     i,j;
        int     offset;

        if (!asrl_session.initialized) asrl_open();

        i=0;
        offset=0;
        for (j=1;j<=(*num_commande);j++)
        {

                viScanf (asrl_session.vid, "%t", &buf);
                // printf ("Instrument : %s\n", buf);

                i=0;
                while ((i<(*taille_sortie))&&(buf[i]!=0))
                {
                        sortie[i+offset]=buf[i]; i++;
                }
            sortie[i]=0;i++;
                offset=offset+strlen(sortie+offset)+1;

        }
}


extern "C" __declspec(dllexport) void asrl_close()
{
        if (asrl_session.initialized)
                        viClose(asrl_session.vid);
}

/* ***************************** */

extern "C" __declspec(dllexport) void visa_open(int *VISA_ID, char *name)
{
        // ex : name = "TCPIP::10.0.0.2::23::SOCKET"
        char m_adr[100];
        sprintf(m_adr,name);
        viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(v_session[*VISA_ID].vid));

        v_session[*VISA_ID].initialized = 1;
}

extern "C" __declspec(dllexport) void visa_close(char *a)
{
        short i;
        for (i=0; i<NB_ADR_VISA; i++)
        {
                if (v_session[i].initialized)
                        viClose(v_session[i].vid);
        }
}

extern "C" __declspec(dllexport) void visa_write (int *VISA_ID, char *commande)
{
        if (v_session[*VISA_ID].initialized)
        {
                viPrintf (v_session[*VISA_ID].vid, commande);
        }

}

extern "C" __declspec(dllexport) void visa_read (int *VISA_ID, int *taille_sortie, char *sortie)
{
        char buf [4096] = {0};
        int     i;

        if (v_session[*VISA_ID].initialized)
        {
                viScanf (v_session[*VISA_ID].vid, "%t", &buf);
                i=0;
                while ((i<(*taille_sortie))&&(buf[i]!=0))
                {
                        sortie[i]=buf[i]; i++;
                }
            sortie[i]=0;i++;
        }
}

extern "C" __declspec(dllexport) void visa_read_bin (int *VISA_ID,int *taille, double *data, int
*delka)
{
        int     i;
        unsigned long    actual;
        char *buffer;

        buffer= (char*) malloc (*taille);

        if (v_session[*VISA_ID].initialized)
        {
        viRead (v_session[*VISA_ID].vid, (ViBuf) buffer, *taille, &actual);

            for (i=0;i<(actual);i++)                                        /*
Probleme d'ecriture de short sur des adresse impaires */
                    data[i]=(double) buffer[i];                             /*
Conversion en double pour Scilab */

        free(buffer);
        *delka=(int) actual;
        }
}

/* ****************************** */




/****************/
/*  GPIB_CLOSE  */
/****************/
extern "C" __declspec(dllexport) void gpib_close(char *a)
{
        short i;
        for (i=0; i<NB_ADR_GPIB; i++)
        {
                if (t_session[i].initialized)
                        viClose(t_session[i].vid);
        }
        asrl_close();
        viClose(DefRM);
}




extern "C" __declspec(dllexport) void gpib_open_device(int GPIB_ID)
{
        char m_adr[32];
        sprintf(m_adr,"GPIB0::%u::INSTR",GPIB_ID);

        viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(t_session[GPIB_ID].vid));

        t_session[GPIB_ID].initialized = 1;
}



extern "C" __declspec(dllexport) void gpib_set_timout(int *GPIB_ID, int *value_ms)
{

        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        viSetAttribute(t_session[*GPIB_ID].vid,VI_ATTR_TMO_VALUE,*value_ms);
        return;
}


extern "C" __declspec(dllexport) void GPIB_write (int *GPIB_ID, int *num_commande, char *commande)
{
        int i,taille;

        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);


        for (i=1;i<=*num_commande;i++)
                {
                        taille=strlen(commande);
                        viPrintf (t_session[*GPIB_ID].vid, commande);
                        commande=commande+taille+1;
                }

}


extern "C" __declspec(dllexport) void GPIB_write_block (int *GPIB_ID, char *buffer, int
*taille_buffer)
{
        unsigned long    actual;

        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);


        viWrite(t_session[*GPIB_ID].vid, (ViBuf) buffer, *taille_buffer, &actual);
}



// #### COPIE D'UN FICHIER BINAIRE VERS LE BUS GPIB
extern "C" __declspec(dllexport) long  fsize(FILE* fd)
{
   long size;
   fseek(fd, 0, SEEK_END);       /* aller en fin */
   size = ftell(fd);             /* lire la taille */
   return size;
}

extern "C" __declspec(dllexport) void GPIB_write_binary_file (int *GPIB_ID, char *fichier)
{
        unsigned long    actual;

        FILE * pFile;
        long lSize;
        char * buffer;
        size_t result;

        pFile = fopen ( fichier , "rb" );
        if (pFile!=NULL)
        {

                // obtain file size:
                fseek (pFile , 0 , SEEK_END);
                lSize = ftell (pFile);
                rewind (pFile);

                // allocate memory to contain the whole file:
                buffer = (char*) malloc (sizeof(char)*lSize);
                if (buffer != NULL)
                {

                        // copy the file into the buffer:
                        result = fread (buffer,1,lSize,pFile);
                        if (result == lSize)
                        {

                                /* the whole file is now loaded in the memory buffer. */
                                if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);
                                viWrite(t_session[*GPIB_ID].vid, (ViBuf) buffer, lSize, &actual);
                        }
                }
        }

  // terminate
  fclose (pFile);
  free (buffer);

}



extern "C" __declspec(dllexport) void GPIB_read (int *GPIB_ID, int *num_commande, int
*taille_sortie, char sortie[])
{
        // char buf [256] = {0};
        char buf [4096] = {0};
        int     i,j;
        int     offset;

        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        i=0;
        offset=0;
        for (j=1;j<=(*num_commande);j++)
        {

                viScanf (t_session[*GPIB_ID].vid, "%t", &buf);
                // printf ("Instrument : %s\n", buf);

                i=0;
                while ((i<(*taille_sortie))&&(buf[i]!=0))
                {
                        sortie[i+offset]=buf[i]; i++;
                }
            sortie[i]=0;i++;
                offset=offset+strlen(sortie+offset)+1;

        }
}



extern "C" __declspec(dllexport) void GPIB_read_data (int *GPIB_ID, int *taille_sortie, char
sortie[])
{
        char buf [23*3201] ={0};
        int     i;
        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        viScanf (t_session[*GPIB_ID].vid, "%t", &buf);
        i=0;
        while ((i<(*taille_sortie))&&(buf[i]!=0))
        {
          sortie[i]=buf[i]; i++;
        }
}


extern "C" __declspec(dllexport) void GPIB_read_stb (int *GPIB_ID, unsigned short *result)
{
        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);
    viReadSTB (t_session[*GPIB_ID].vid, result);
}








extern "C" __declspec(dllexport) void AWG2021_save_signal (int *GPIB_ID,int *taille, double *data,
char *name)
{
        /* Version Simple : pas de gestion des markers */

        int      taille_buffer,offset,i,j;
        char *buffer;
        char size_data[10];

        short tampon;
        int nb_command;

        nb_command=1;

// Preparation de l'envoi
        taille_buffer=16+strlen(name);
        buffer= (char*) malloc (taille_buffer);
        sprintf(buffer,":DATA:DEST ");
        buffer[11]=34;
        sprintf(buffer+12,"%s",name);
        buffer[12+strlen(name)]=34;
        buffer[12+strlen(name)+1]=10;
        buffer[12+strlen(name)+2]=13;
        buffer[12+strlen(name)+3]=0;
        // printf("%s",buffer);
        GPIB_write(GPIB_ID,&nb_command,buffer);
        free(buffer);

// Construction du binblock à partir d'un tableau de double normalisé
        taille_buffer=2*(*taille);
        sprintf(size_data,"%d",taille_buffer);
        // printf("taille buffer %d",taille_buffer);
        taille_buffer=7+2+taille_buffer+strlen(size_data);
        // 7 pour ":CURVE "
        // 2 pour "#x"

        buffer= (char*) malloc (taille_buffer);
        sprintf(buffer,":CURVE #");
        sprintf(buffer+8,"%d",strlen(size_data));
        sprintf(buffer+9,"%s",size_data);
        offset=9+strlen(size_data);

        for (i=0;i<*taille;i++) {
                tampon=(short)(((((double) data[i])+1)/2)*4095);
                for (j=0;j<2;j++)
                        buffer[offset+i*2+1-j]=*( ((char *)&tampon)+j);
        }

// Debugage
//      for (i=0;i<taille_buffer;i++) printf("%c",buffer[i]);

// Envoi via GPIB
        GPIB_write_block (GPIB_ID, buffer, &taille_buffer);


        free(buffer);


}
```

```
extern "C" __declspec(dllexport) void SMU200A_save_signal (int *GPIB_ID,int *taille, double *data,
char *name, double *clock)
{
        /* {TYPE: SMU-WV,0} {CLOCK: 10e6} {WAVEFORM-401: #ÿ𝒵

        */

//      char entete[50];
//      sprintf(entete,"{TYPE: SMU-WV,0} {CLOCK: %E} {WAVEFORM-",*clock);
//      taille_buffer=2*(*taille);



        /* Version Simple : pas de gestion des markers */

        int     taille_buffer,offset,i,j;
        char *buffer;
        char size_data[10];

        short tampon;
        int nb_command;

        nb_command=1;

// Preparation de l'envoi
        taille_buffer=16+strlen(name);
        buffer= (char*) malloc (taille_buffer);
        sprintf(buffer,":DATA:DEST ");
        buffer[11]=34;
        sprintf(buffer+12,"%s",name);
        buffer[12+strlen(name)]=34;
        buffer[12+strlen(name)+1]=10;
        buffer[12+strlen(name)+2]=13;
        buffer[12+strlen(name)+3]=0;
        // printf("%s",buffer);
        GPIB_write(GPIB_ID,&nb_command,buffer);
        free(buffer);

// Construction du binblock à partir d'un tableau de double normalisé
        taille_buffer=2*(*taille);
        sprintf(size_data,"%d",taille_buffer);
        // printf("taille buffer %d",taille_buffer);
        taille_buffer=7+2+taille_buffer+strlen(size_data);
        // 7 pour ":CURVE "
        // 2 pour "#x"

        buffer= (char*) malloc (taille_buffer);
        sprintf(buffer,":CURVE #");
        sprintf(buffer+8,"%d",strlen(size_data));
        sprintf(buffer+9,"%s",size_data);
        offset=9+strlen(size_data);

        for (i=0;i<*taille;i++) {
                tampon=(short)(((((double) data[i])+1)/2)*4095);
                for (j=0;j<2;j++)
                        buffer[offset+i*2+1-j]=*( ((char *)&tampon)+j);
        }

// Debugage
//      for (i=0;i<taille_buffer;i++) printf("%c",buffer[i]);

// Envoi via GPIB
        GPIB_write_block (GPIB_ID, buffer, &taille_buffer);


        free(buffer);




}




extern "C" __declspec(dllexport) void AWG520_save_signal (int *GPIB_ID,int *taille, double *data,
char *name, double *clock)
```

```
{
        /* Version Simple : pas de gestion des markers */

        int      taille_buffer,offset,i,j;
        char *buffer;
        char size_data[10];
        char size_data_2[10];
        char clock_line[50];
        float tampon;


// Preparation de l'envoi
        sprintf(clock_line,"CLOCK %E\r\n",*clock);

        // printf("\n %s \n",clock_line);
        taille_buffer=5*(*taille);
        sprintf(size_data,"%d",taille_buffer);
        i=12+strlen(size_data)+taille_buffer+strlen(clock_line)+1;
        sprintf(size_data_2,"%d",i);

        taille_buffer=i+2+strlen(size_data_2)+14+strlen(name);

        buffer= (char*) malloc (taille_buffer);

        offset=12+strlen(name);

        sprintf(buffer,":MMEM:DATA ");
        buffer[11]=34;                                      /* " */
        sprintf(buffer+12,"%s",name);      /* nom de fichier */
        buffer[offset]=34;                                  /* " */
        buffer[offset+1]=44;                        /* , */
        buffer[offset+2]=35;                        /* # */
        sprintf(buffer+offset+3,"%d",strlen(size_data_2));           /* index */
        sprintf(buffer+offset+4,"%s",size_data_2 );                          /* taille en byte */

        offset=offset+4+strlen(size_data_2);
        sprintf(buffer+offset,"MAGIC 1000%c%c%c",13,10,35);
        offset=offset+13;

        sprintf(buffer+offset,"%d",strlen(size_data));                       /* index */
        sprintf(buffer+offset+1,"%s",size_data );                            /* taille en byte */

        offset=offset+strlen(size_data)+1;

        for (i=0;i<(*taille);i++)
        {
                tampon=(float) data[i];
                for (j=0;j<4;j++)
                        buffer[offset+i*5+j]=*( ((char *)&tampon)+j);
                buffer[offset+i*5+4]=0;
        }
        buffer[offset+4]=1;

        offset=offset+(*taille)*5;

        sprintf(buffer+offset,"%s",clock_line );


// Debugage
//      printf("\r\n Taille = %d \r\n",taille_buffer);
//      for (i=0;i<taille_buffer;i++) printf("%c",buffer[i]);
//      printf("\r\n");
//      for (i=0;i<taille_buffer;i++) printf("<%d>",buffer[i]);


// Envoi via GPIB

        GPIB_write_block (GPIB_ID, buffer, &taille_buffer);


        free(buffer);


}


extern "C" __declspec(dllexport) void GPIB_read_block_short (int *GPIB_ID,int *taille, double *data)
{
        unsigned long    actual;

        int      taille_buffer,i,offset;
        char *buffer;
        char size_data[10];
        short    tampon=0;

        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        sprintf(size_data,"%d",((*taille)*2));
        taille_buffer=((*taille)*2)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock      */
        buffer= (char*) malloc (taille_buffer);
        viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
        offset=2+strlen(size_data);

        // for (i=0;i<taille_buffer;i++) printf("<%c : %d> \r\n",buffer[i],buffer[i]);

        for (i=0;i<(*taille);i++)                                                   /* Probleme
d'ecriture de short sur des adresse impaires */
        {
                        /* On passe donc par une variable tampon déclarée en Short */

        *((char *)(&tampon))=buffer[offset+(i*2)+1];        /* Inversion Manuelle de l'ordre des
byte dans le short */
        *(((char *)(&tampon))+1)=buffer[offset+(i*2)];
        // printf ("%d",tampon);
        data[i]=(double) tampon;                                            /* Conversion en
double pour Scilab */
        }


        free(buffer);
}



extern "C" __declspec(dllexport) void GPIB_read_block_byte (int *GPIB_ID,int *taille, double *data)
{
        unsigned long    actual;

        int      taille_buffer,i,offset;
        char *buffer;
        char size_data[10];


        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        sprintf(size_data,"%d",((*taille)));
        taille_buffer=((*taille))+2+strlen(size_data)+1;    /* +1 : Octet EOL du binblock      */
        buffer= (char*) malloc (taille_buffer);
        viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
        offset=2+strlen(size_data);

        for (i=0;i<(*taille);i++)                                                   /* Probleme
d'ecriture de short sur des adresse impaires */
                data[i]=(double) buffer[i+offset];                                            /*
Conversion en double pour Scilab */

        free(buffer);
}


extern "C" __declspec(dllexport) void GPIB_read_block_float (int *GPIB_ID,int *taille, double *data)
{
        unsigned long    actual;

        int      taille_buffer,i,offset;
        char *buffer;
        char size_data[10];
        float    tampon=0;


        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        sprintf(size_data,"%d",((*taille)*4));
        taille_buffer=((*taille)*4)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock      */
        buffer= (char*) malloc (taille_buffer);
        viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
        offset=2+strlen(size_data);

        for (i=0;i<(*taille);i++) /* Probleme d'ecriture de short sur des adresse impaires */
        {
                *(((char *)(&tampon)))=buffer[offset+(i*4)+3];      /* Inversion Manuelle de
l'ordre des byte dans le short */
                *(((char *)(&tampon))+1)=buffer[offset+(i*4)+2];
                *(((char *)(&tampon))+2)=buffer[offset+(i*4)+1];
                *(((char *)(&tampon))+3)=buffer[offset+(i*4)];

                data[i]=(double) tampon;
        }

        free(buffer);
}


extern "C" __declspec(dllexport) void GPIB_read_block_floatNI (int *GPIB_ID,int *taille, double
*data)
{
        unsigned long    actual;

        int      taille_buffer,i,offset;
        char *buffer;
        char size_data[10];
        float    tampon=0;


        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        sprintf(size_data,"%d",((*taille)*4));
        taille_buffer=((*taille)*4)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock      */
        buffer= (char*) malloc (taille_buffer);
        viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
        offset=2+strlen(size_data);

        for (i=0;i<(*taille);i++) /* Probleme d'ecriture de short sur des adresse impaires */
        {
                *(((char *)(&tampon)))=buffer[offset+(i*4)];        /* Inversion Manuelle de
l'ordre des byte dans le short */
                *(((char *)(&tampon))+1)=buffer[offset+(i*4)+1];
                *(((char *)(&tampon))+2)=buffer[offset+(i*4)+2];
                *(((char *)(&tampon))+3)=buffer[offset+(i*4)+3];

                data[i]=(double) tampon;
        }

        free(buffer);
}



extern "C" __declspec(dllexport) void GPIB_read_block_double (int *GPIB_ID,int *taille, double
*data)
{
        unsigned long    actual;

        int      taille_buffer,i,offset;
        char *buffer;
        char size_data[10];
        double   tampon=0;


        if (!t_session[*GPIB_ID].initialized) gpib_open_device(*GPIB_ID);

        sprintf(size_data,"%d",((*taille)*8));
        taille_buffer=((*taille)*8)+2+strlen(size_data)+1;  /* +1 : Octet EOL du binblock      */
        buffer= (char*) malloc (taille_buffer);
        viRead(t_session[*GPIB_ID].vid, (ViBuf) buffer, taille_buffer, &actual);
        offset=2+strlen(size_data);

        for (i=0;i<(*taille);i++) /* Probleme d'ecriture de short sur des adresse impaires */
        {
                *(((char *)(&tampon)))=buffer[offset+(i*8)+7];
                *(((char *)(&tampon))+1)=buffer[offset+(i*8)+6];
                *(((char *)(&tampon))+2)=buffer[offset+(i*8)+5];
                *(((char *)(&tampon))+3)=buffer[offset+(i*8)+4];
                *(((char *)(&tampon))+4)=buffer[offset+(i*8)+3];
                *(((char *)(&tampon))+5)=buffer[offset+(i*8)+2];
                *(((char *)(&tampon))+6)=buffer[offset+(i*8)+1];
                *(((char *)(&tampon))+7)=buffer[offset+(i*8)];

                data[i]= tampon;
        }

        free(buffer);
}


/* Ouverure et fermeture de la session dans l'instruction */
/* Sert au sous adresses GPIB de l'alim AGILENT 63000A */

extern "C" __declspec(dllexport) void GPIB_write_sub (int *GPIB_ID, int *GPIB_SUB_ID, int
*num_commande, char *commande)
{
        int i,taille;
        char m_adr[32];
/*      ViSession my_instrument; */

        sprintf(m_adr,"GPIB0::%u::%u::INSTR",*GPIB_ID,*GPIB_SUB_ID);
        /* printf(m_adr); */



        viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(my_instrument));


        for (i=1;i<=*num_commande;i++)
                {
                        taille=strlen(commande);
                        viPrintf (my_instrument, commande);
                        /* printf(commande); */
                        commande=commande+taille+1;
                }


        viClose(my_instrument);


}

/* Ouverure et fermeture de la session INSTRUMENT (pas VISA) dans l'instruction même */
/* Tout ca a cause de ces putain de sous adresses GPIB de l'alim Agilent 63000A */


extern "C" __declspec(dllexport) void GPIB_read_sub (int *GPIB_ID, int *GPIB_SUB_ID, int
*num_commande, int *taille_sortie, char sortie[])
{
        char buf [256] = {0};
        int     i,j;
        int     offset;
        char m_adr[32];
        ViSession my_instrument;


        sprintf(m_adr,"GPIB0::%u::%u::INSTR",*GPIB_ID,*GPIB_SUB_ID);

        viOpen(DefRM,m_adr,VI_NULL,VI_NULL,&(my_instrument));

        i=0;
        offset=0;
        for (j=1;j<=(*num_commande);j++)
        {

                viScanf (my_instrument, "%t", &buf);
        /*      printf ("Instrument : %s\n", buf); */

                i=0;
                while ((i<(*taille_sortie))&&(buf[i]!=0))
                {
                        sortie[i+offset]=buf[i]; i++;
                }
            sortie[i]=0;i++;
                offset=offset+strlen(sortie+offset)+1;

        }
        viClose(my_instrument);
}
```