



Sciences et technologies de l'Industrie et du développement durable

SIN 1 : Maquettage d'une solution en réponse à un cahier des charges

Document ressource : Utilisation du composant DLL dans ProfiLab Expert



Sommaire

| | | |
|----------|---------------------------------------------------------------|-----------|
| 1 | Introduction..... | 3 |
| 2 | Réalisation d'une DLL | 3 |
| 2.1 | Création du projet | 3 |
| 2.2 | Programmation | 3 |
| 2.3 | Fonctions exportées | 4 |
| 3 | Utilisation d'une DLL | 5 |
| 4 | Réalisation d'une DLL pour ProfiLab Expert | 5 |
| 4.1 | Introduction..... | 5 |
| 4.2 | Réalisation de la DLL..... | 5 |
| 4.2.1 | Fonction NumInputs | 6 |
| 4.2.2 | Fonction NumOutputs | 6 |
| 4.2.3 | Fonction GetInputName | 7 |
| 4.2.4 | Fonction GetOutputName | 7 |
| 4.2.5 | Fonction CSimStart | 8 |
| 4.2.6 | Fonction CCalculate | 9 |
| 4.2.7 | Fonction CSimStop..... | 9 |
| 4.2.8 | Fonction CConfigure | 9 |
| 4.3 | Création de la DLL..... | 10 |
| 5 | Utilisation du composant DLL sous ProfiLab Expert..... | 10 |

1 Introduction

Une DLL (Dynamic Link Library) est un fichier contenant des fonctions logicielles (programmation dans différents langages possibles) qui peuvent être utilisées par plusieurs applications.

Son utilisation est très intéressante. Elle permet la mise en commun de bibliothèques de fonctions dans plusieurs applications.

Ce document décrit la réalisation et l'utilisation d'une DLL en C++ à l'aide du logiciel DevC++.

2 Réalisation d'une DLL

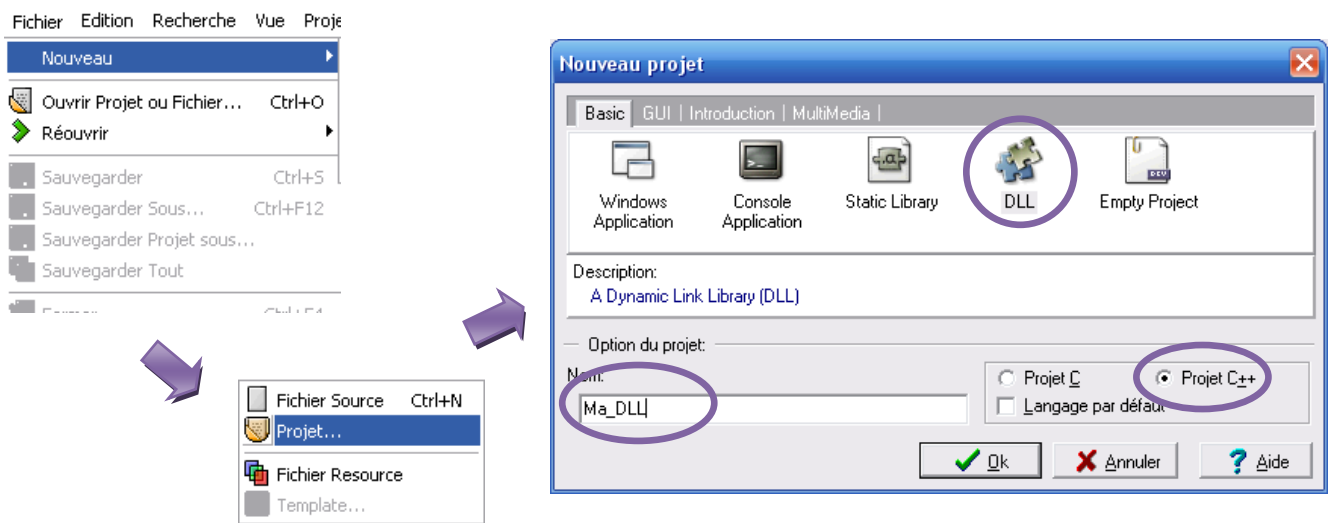
2.1 Création du projet

Pour créer une DLL, il faut d'abord créer un projet et configurer le logiciel.

- Lancer le logiciel DevC++ :



- Dans le menu 'Fichier', cliquer sur 'Nouveau' puis sur 'Projet...', ou bien sur l'icône .



- Sélectionner l'application 'DLL', sélectionner 'projet C++', puis entrer un nom de fichier pour la DLL et cliquer OK.
- Sélectionner le dossier de travail et un nom de projet.

Le projet est créé ainsi que 2 fichiers : 'dll.h' et 'dllmain.cpp'. Sur la fenêtre à gauche de l'écran apparaît l'arborescence du projet.

2.2 Programmation

- Dans l'arborescence du projet, cliquer sur le fichier 'dll.h'.

Par défaut, le logiciel a écrit une architecture de programme.

Ce fichier contient le prototype des fonctions, les types personnalisés, les structures...

- Inclure dans ce fichier la ligne suivante :

```
#define DLL_EXPORT extern "C" __declspec(dllexport)
```

- Inclure également dans ce fichier la définition de toutes les fonctions de la DLL.


Cette ligne définit la commande DLLEXPORT

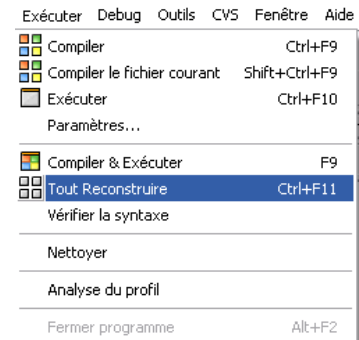
La fonction ‘`__declspec(dllexport)`’ permet au compilateur de générer les noms d’exportation des fonctions (dans un fichier avec l’extension ‘.a’ et ‘.def’).

Le code ‘externe «C»’ permet de conserver les noms des fonctions (sans décoration) à la compilation. En effet certains compilateurs modifient les noms des fonctions créées en ajoutant des chiffres ou des symboles tel que @.

- Dans l’arborescence du projet, cliquer sur le fichier ‘`maindll.cpp`’ et réaliser les fonctions souhaitées.

Lorsque le programme est termine. Il faut compiler le projet pour obtenir le fichier ‘.dll’.

- Dans le menu ‘*Exécuter*’, cliquer sur ‘*Tout Reconstruire*’ ou sur l’icône .
- Si des erreurs sont présentes dans le programme, elles apparaissent en bas de l’écran.



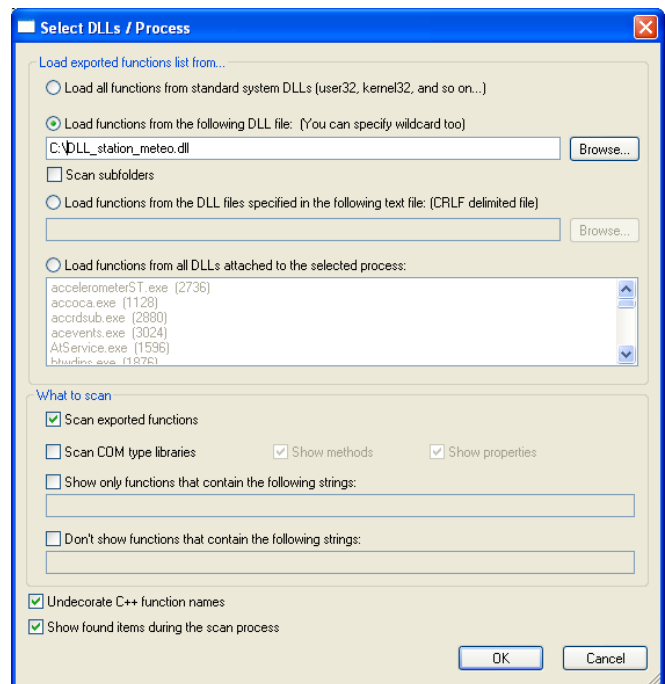
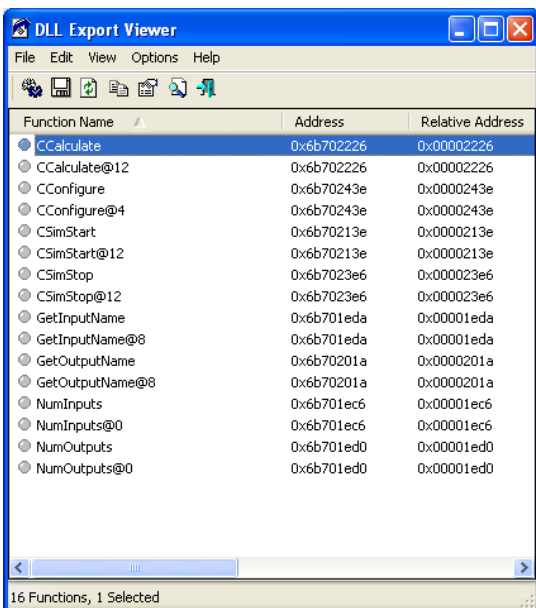
2.3 Fonctions exportées

Il est intéressant de vérifier si les fonctions exportées sont bien celles désirées, sans décoration.

Il existe pour cela des logiciels tel que *DIExp*.

- Sélectionner la fonction ‘*Load functions from the following DLL file*’ et indiquer le fichier puis OK.

Le résultat est le suivant :




Le compilateur a exporté les fonctions sous deux noms différents : les fonctions avec le nom correct et avec le nom décoré.

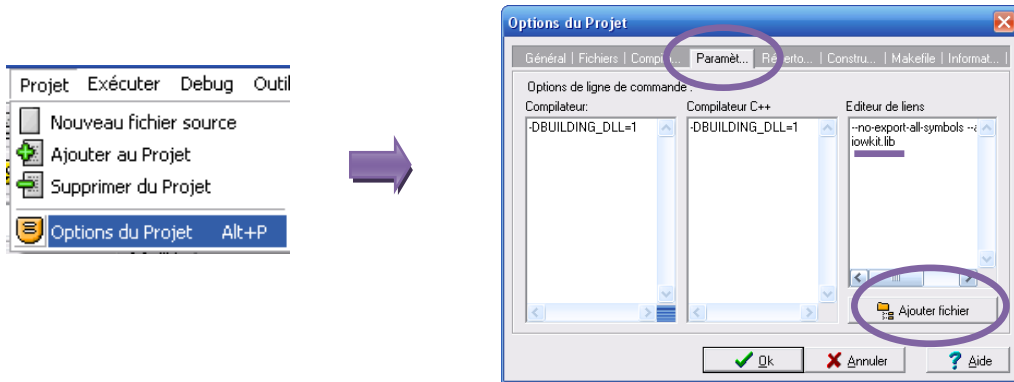
Par exemple : CCalculate et CCalculate@12 correspondent à la même fonction (même adresse).

3 Utilisation d'une DLL

Lors de la réalisation du projet, il faut intégrer la DLL. Cela permet d'utiliser les fonctions de la DLL comme si on avait fait un 'include'.

Pour cela :

- Dans le menu 'Projet', cliquer sur 'Option du projet' ou sur l'icone 



- Sélectionner l'onglet 'Paramètres' puis sur 'Ajouter fichier'. Ajouter le fichier 'iowkit.lib' et cliquer OK.

4 Réalisation d'une DLL pour ProfiLab Expert

4.1 Introduction

ProfiLab Expert est un logiciel de programmation graphique. Il intègre l'utilisation d'un nombre important de matériels tels que composant d'interfaçage, matériels de mesures...

Mais il est possible de gérer des matériels non intégrés au logiciel, de réaliser des calculs très complexes en utilisant le composant DLL fourni dans les bibliothèques du logiciel

Pour cela il faut concevoir une DLL dans un langage de programmation C, C++, Basic, Delphi.

4.2 Réalisation de la DLL

La réalisation de la DLL se fait par la création d'un projet (voir paragraphe 2.1).

La différence réside dans la réalisation du fichier 'dllmain.cpp'. L'architecture proposée ne convient pas. Il faut tout effacer.

- Il faut tout d'abord inclure le fichier d'en-tête précédent avec l'instruction : #include "dll.h"

L'utilisation de la DLL sous ProfiLab Expert impose la présence des fonctions suivantes :

| Nom de la fonction | Rôle |
|-------------------------------------------------------------------------|--------------------------------------------------|
| unsigned char _stdcall NumInputs() | Définir le nombre d'entrées du composant DLL |
| unsigned char _stdcall NumOutputs() | Définir le nombre de sortie du composant DLL |
| void _stdcall GetInputName(unsigned char Channel, unsigned char *Name) | Définir le nom de chaque entrée du composant DLL |
| void _stdcall GetOutputName(unsigned char Channel, unsigned char *Name) | Définir le nom de chaque sortie du composant DLL |

| | |
|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <code>void _stdcall CSimStart(double *PInput, double *POutput, double *PUser)</code> | Définir les conditions initiales du composant DLL lors de l'exécution du programme Profilab Expert |
| <code>void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)</code> | Définir le fonctionnement interne du composant DLL lors de l'exécution du programme Profilab Expert |
| <code>void _stdcall CSimStop(double *PInput, double *POutput, double *PUser)</code> | Définir l'arrêt du programme Profilab Expert |
| <code>void _stdcall CConfigure(double *PUser)</code> | Définir les configurations du composant DLL (lors de l'élaboration du programme sous Profilab Expert) |

Ce sont ces fonctions de la DLL qui doivent être accessibles par le logiciel Profilab Expert. Il faut donc les exporter.

- Compléter le fichier 'maindll.cpp' de la façon suivante pour créer les fonctions précédentes :

Remarques :

- L'ajout de la commande 'DLLEXPORT', définie dans le fichier d'en-tête permet l'export des fonctions.
- La commande '__stdcall' permet de gérer les paramètres de la pile lors de l'appel de la fonction.

```

/*Inclure le fichier d'en-tête*/
#include "dll.h"

// Retourne le nombre d'entrées
DLLEXPORT unsigned char _stdcall NumInputs()
{
}

// Retourne le nombre de sorties
DLLEXPORT unsigned char _stdcall NumOutputs()
{
}

// Retourne le nom des entrées
DLLEXPORT void _stdcall GetInputName(unsigned char Channel,unsigned char *Name)
{
}

// Retourne le nom des sorties
DLLEXPORT void _stdcall GetOutputName(unsigned char Channel,unsigned char *Name)
{
}

// Retourne les conditions au départ
DLLEXPORT void _stdcall CSimStart(double *PInput, double *POutput, double *PUser)
{
}

// Retourne l'état des sorties en fonction des entrées
DLLEXPORT void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)
{
}

// Arrêt du programme
DLLEXPORT void _stdcall CSimStop(double *PInput, double *POutput, double *PUser)
{
}

// Configuration du programme
DLLEXPORT void _stdcall CConfigure(double *PUser)
{
}

```

4.2.1 Fonction NumInputs

Cette fonction doit retourner un octet dont la valeur détermine le nombre d'entrées de la DLL. Cela se matérialisera par autant de broches d'entrées sur le symbole DLL dans Profilab Expert.

Exemple pour 1 entrée :

```

// Retourne le nombre d'entrées
DLLEXPORT unsigned char _stdcall NumInputs()
{
    return 1;
}

```

4.2.2 Fonction NumOutputs

Cette fonction doit retourner un octet dont la valeur détermine le nombre de sorties de la DLL. Cela se matérialisera par autant de broches de sorties sur le symbole DLL dans Profilab Expert.

Exemple pour 3 sorties :

```
// Retourne le nombre de sorties
DLLEXPORT unsigned char _stdcall NumOutputs()
{
    return 3;
}
```

4.2.3 Fonction GetInputName

Cette fonction permet de donner un nom aux différentes entrées. Elle est exécutée autant de fois qu'il y a d'entrées. Elle nécessite le passage de 2 paramètres :

| Paramètre | Type | Rôle |
|-----------|------------------------------|----------------------------------------------------------------------------------|
| Channel | Octet non signé | Désigne le numéro de l'entrée (la 1ère entrée a le n°0) |
| *Name | Pointeur sur octet non signé | Désigne une suite d'octets contenant le nom de l'entrée (doit se terminer par 0) |

Exemple pour 1 entrée :

```
// Retourne le nom des entrées
DLLEXPORT void _stdcall GetInputName(unsigned char Channel, unsigned char *Name)
{
    if (Channel == 0)
    {
        Name[0] = 'I';
        Name[1] = 'N';
        Name[2] = '1';
        Name[3] = 0 ;
    }
}
```

4.2.4 Fonction GetOutputName

Cette fonction permet de donner un nom aux différentes sorties. Elle est exécutée autant de fois qu'il y a de sorties. Elle nécessite le passage de 2 paramètres :

| Paramètre | Type | Rôle |
|-----------|------------------------------|----------------------------------------------------------------------------------|
| Channel | Octet non signé | Désigne le numéro de l'entrée (la 1ère entrée a le n°0) |
| *Name | Pointeur sur octet non signé | Désigne une suite d'octets contenant le nom de l'entrée (doit se terminer par 0) |

Exemple pour 2 sorties :

```

// Retourne le nom des sorties
DLLEXPORT void _stdcall GetOutputName(unsigned char Channel,unsigned char *Name)
{
    switch(Channel)
    {
        case 0: // Sortie 0
        {
            Name[0]= 'O';
            Name[1]= 'U';
            Name[2]= 'T';
            Name[3]= '1';
            Name[4]= 0;
            break;
        }
        case 1: // Sortie 1
        {
            Name[0]= 'O';
            Name[1]= 'U';
            Name[2]= 'T';
            Name[3]= '2';
            Name[4]= 0;
            break;
        }
        default:;
    }
}

```

4.2.5 Fonction CSimStart

Cette fonction permet d'initialiser le composant DLL (par exemple : fixer les valeurs initiales de variables internes et (ou) des sorties de la DLL). Elle nécessite le passage de 3 paramètres :

| Paramètre | Type | Rôle |
|-----------|-------------------------------------|---------------------------------------------------------------------------------|
| *PInput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des entrées |
| *POutput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des sorties |
| *PUser | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où il est possible de stocker des variables |

La première entrée est l'entrée numéro 0. La valeur de cette entrée est accessible par la variable PInput[0].

La première sortie est la sortie numéro 0. La valeur de cette sortie est accessible par la variable POutput[0].

La seconde sortie est la sortie numéro 1. La valeur de cette sortie est accessible par la variable POutput[1].

Profilab Expert prend en compte la tension et non le niveau logique. Pour un niveau logique 0, il faudra mettre la variable à 0. Pour un niveau logique 1, il faudra la mettre à 5.

Exemple d'initialisation de 2 sorties :

```

// Retourne les conditions au départ
DLLEXPORT void _stdcall CSimStart(double *PInput, double *POutput, double *PUser)
{
    POutput[0] = 0;
    POutput[1] = 5;
}

```


4.2.6 Fonction CCalculate

Cette fonction permet de définir le fonctionnement interne du composant DLL (par exemple : comment varie la sortie de la DLL). Elle nécessite le passage de 3 paramètres :

| Paramètre | Type | Rôle |
|-----------|-------------------------------------|---------------------------------------------------------------------------------|
| *PInput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des entrées |
| *POutput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des sorties |
| *PUser | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où il est possible de stocker des variables |

```
// Retourne l'état des sorties en fonction des entrées
DLLEXPORT void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)
{
    if (PInput[0] < 2.5)
    {
        POutput[0] = 0;
        POutput[1] = 5;
    }
    else
    {
        POutput[0] = 5;
        POutput[1] = 0;
    }
}
```

Exemple avec 1 entrée et 2 sorties :

4.2.7 Fonction CSimStop

Cette fonction est exécutée lors de l'arrêt du programme Profilab Expert. Elle permet par exemple de fermer un fichier, clore une communication. Elle nécessite le passage de 3 paramètres :

| Paramètre | Type | Rôle |
|-----------|-------------------------------------|---------------------------------------------------------------------------------|
| *PInput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des entrées |
| *POutput | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où sont stockées les valeurs des sorties |
| *PUser | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où il est possible de stocker des variables |

S'il n'y a aucune opération à effectuer à l'arrêt du programme, cette fonction doit être vide, mais présente dans la DLL.

4.2.8 Fonction CConfigure

Cette fonction permet de configurer le composant DLL avant l'exécution du programme Profilab Expert. Elle est optionnelle.

Elle permet par exemple de choisir le nombre d'entrées, le nombre de sorties, des conditions initiales...


Elle nécessite le passage d'un paramètre

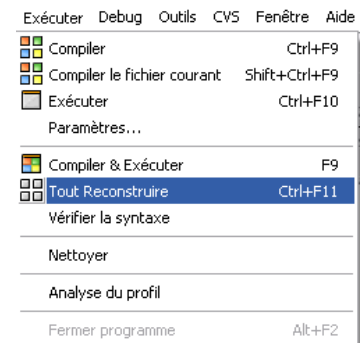
| Paramètre | Type | Rôle |
|-----------|-------------------------------------|---------------------------------------------------------------------------------|
| *PUser | Pointeur sur réel double (8 octets) | Permet d'accéder à une zone mémoire où il est possible de stocker des variables |

Un exemple, pour un compteur, est donné dans le dossier d'installation du logiciel.

4.3 Création de la DLL

Lorsque le programme est terminé. Il faut compiler le projet pour obtenir le fichier '.dll'.

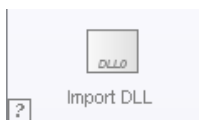
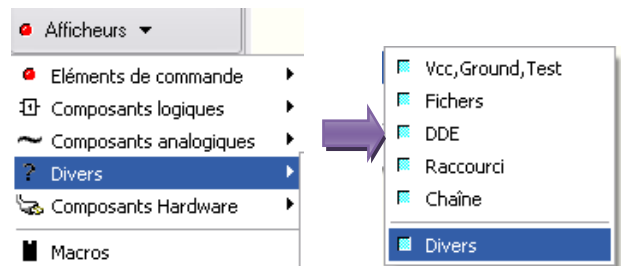
- Dans le menu 'Exécuter', cliquer sur 'Tout Reconstruire' ou sur l'icône .
- Si des erreurs sont présentes dans le programme, elles apparaissent en bas de l'écran.



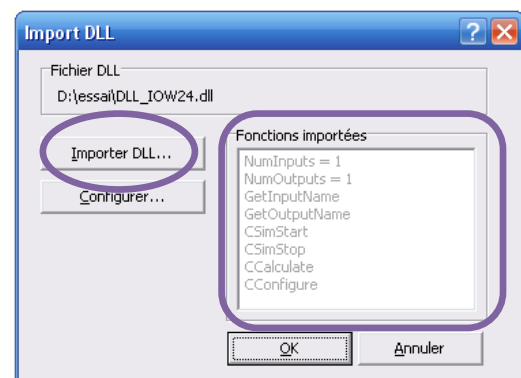
Remarque : Si Profilab Expert est ouvert et utilise la DLL, il n'est alors pas possible de recompiler. Il faut absolument le fermer.

5 Utilisation du composant DLL sous Profilab Expert

- Sous Profilab Expert, créer un nouveau fichier
- Dans le menu des bibliothèques de composants, sélectionner 'Divers', puis le menu 'Divers'.
- Sélectionner le composant 'Import DLL' en cliquant dessus puis le positionner dans la zone de programmation graphique



- Double-cliquer sur le symbole puis sur 'Importer DLL...' et sélectionner le fichier '.dll' créé.
- Lorsque l'opération est réussie, la liste des fonctions créées dans la DLL apparaît dans la fenêtre 'Fonctions importées'.



Si la fonction 'CConfigure' existe dans la DLL, le bouton 'Configurer...' est accessible. Lors d'un clic dessus, cette fonction est exécutée.

Cliquer sur OK et le symbole DLL est mis a jour en tenant compte des entrées et sorties définie dans la DLL.

