Sciences et technologies
de l'Industrie et du développement durable

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

Module SIN 1.1 : Concevoir un système local et permettre le dialogue entre l'homme et la machine

Activité : TP1 – IOWarrior - Commande de la Led

IO-Warrior

Generic universal I/O Controller for USB

Code Mercenaries

Sommaire

1 Starter Kit IOWarrior 24

1.1 Le composant IOWarrior

Le composant IOWarrior est un contrôleur d'entrées/sorties. Il intègre un certain nombre de fonctions. Ces fonctions dépendent de la version du composant.

Nous allons utiliser le composant IOW 24. Cette version intègre :

- ☒ Une interface d'entrées/sorties
- ☒ Une liaison I2C
- ☒ Une liaison SPI
- ☒ Un décodeur infrarouge (code RC5)
- ☒ Une gestion de matrice de LED
- ☒ Un afficheur LCD compatible avec le HD44780
- ☒ 2 registres timers

Toutes ces fonctions se sont pas accessibles simultanément. Le IOW24 dispose de 2 modes de fonctionnement :

- ☒ Mode normal : accès aux entrées/sorties
- ☒ Mode spécial : accès aux autres fonctions

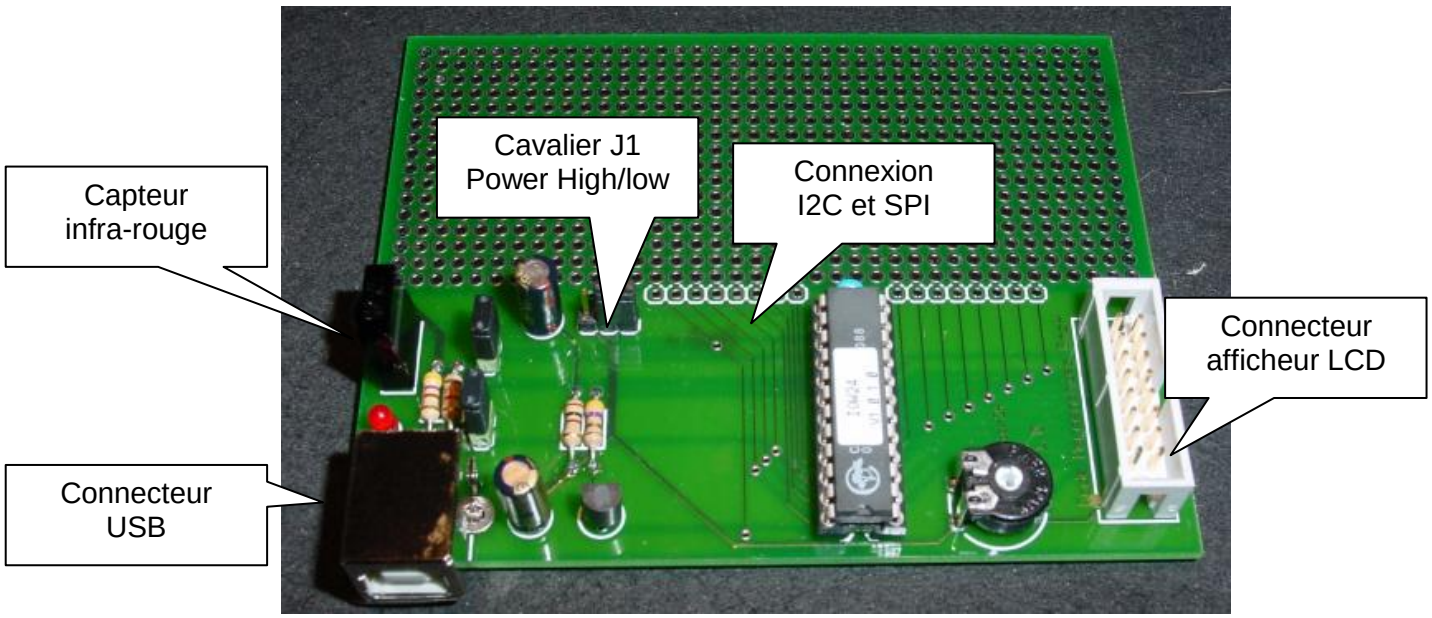
1.2 Carte de prototypage

Le composant est utilisé sur une carte de prototypage.

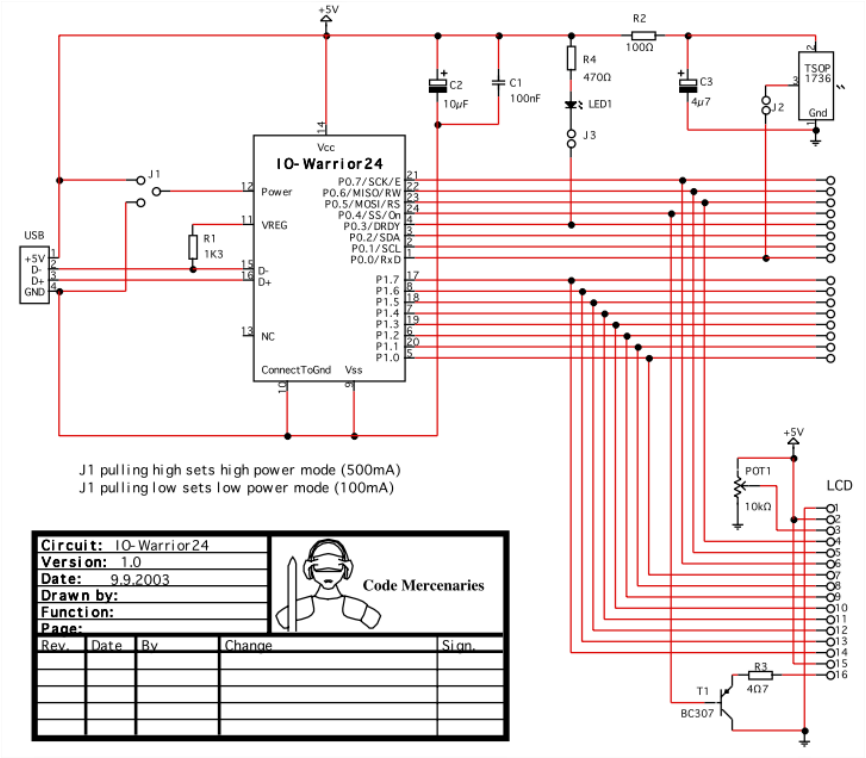
L'utilisation de la carte peut nécessiter un courant important ou non. Il est donc possible de régler la puissance demandée au port USB par l'intermédiaire d'un cavalier J1 :

- ☒ En position High : courant maxi jusqu'à 500mA
- ☒ En position Low : courant maxi de 100mA

Pour la suite, on placera J1 sur High.



1.3 Schéma structurel de la carte de prototypage



1.4 Connexion au PC

La connexion se fait par port USB. Il faut placer J1 avant de relier la carte et le PC.

Le composant IOW24 est définit comme un composant de la classe HID (Human Interface Device), le PC le reconnait donc automatiquement.

- Relier la carte de prototypage au PC. Au bout de quelques instants, il est possible de travailler avec la carte.

2 Programmation graphique

Utilisation de la carte de prototypage se fait par l'intermédiaire d'un logiciel qui permet la programmation graphique : ProfiLab Expert.

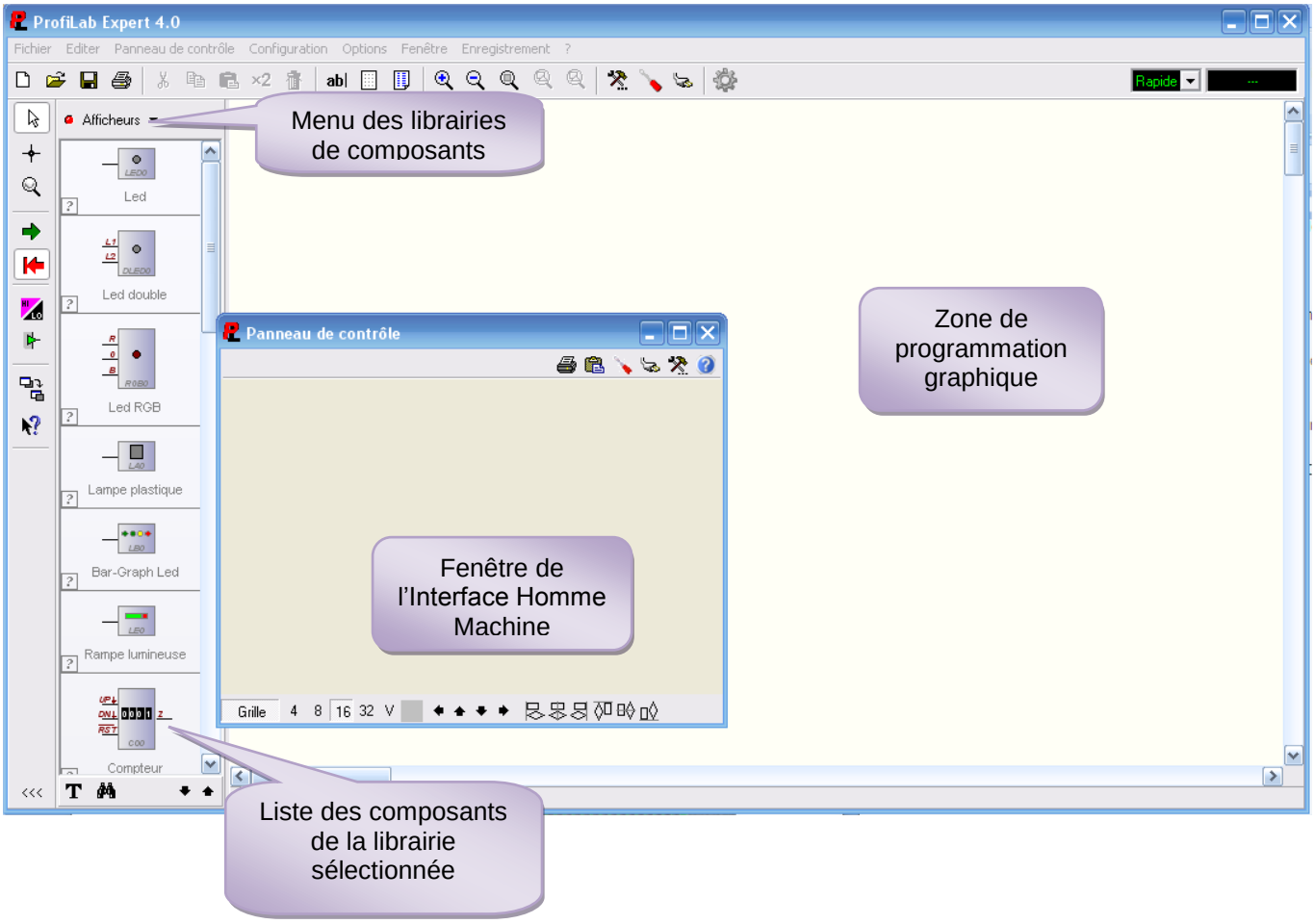
Il est tout à fait possible de piloter la carte par l'intermédiaire d'un programme C, C++.

2.1 ProfiLab Expert

C'est un logiciel de programmation graphique. Attention, il ne permet pas d'implanter un programme dans un composant.

- Lancer le logiciel à partir de l'icône sur le bureau du PC.

Il est impératif de relier d'abord la carte de prototypage au PC, puis de lancer le logiciel.



Le rôle des différents icones sera indiqué au fur et à mesure de leurs utilisations dans l'activité.

2.2 Utilisation de l'interface d'entrée/sortie

Les objectifs sont :

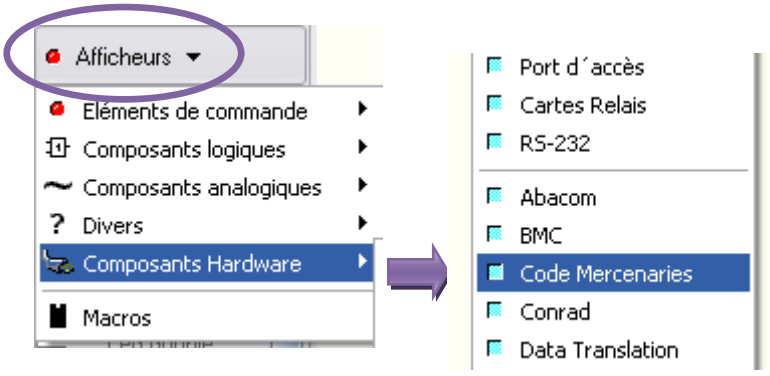
- ✓ La prise en main du logiciel de programmation (comprendre la manière de programmer)
- ✓ L'utilisation de l'interface entrées/sorties du composant IOW24

Pour cela, le programme qui va être fait permet l'utilisation d'un port d'entrée/sortie pour commander la Led présente sur la carte de prototypage.

2.2.1 Programmation graphique

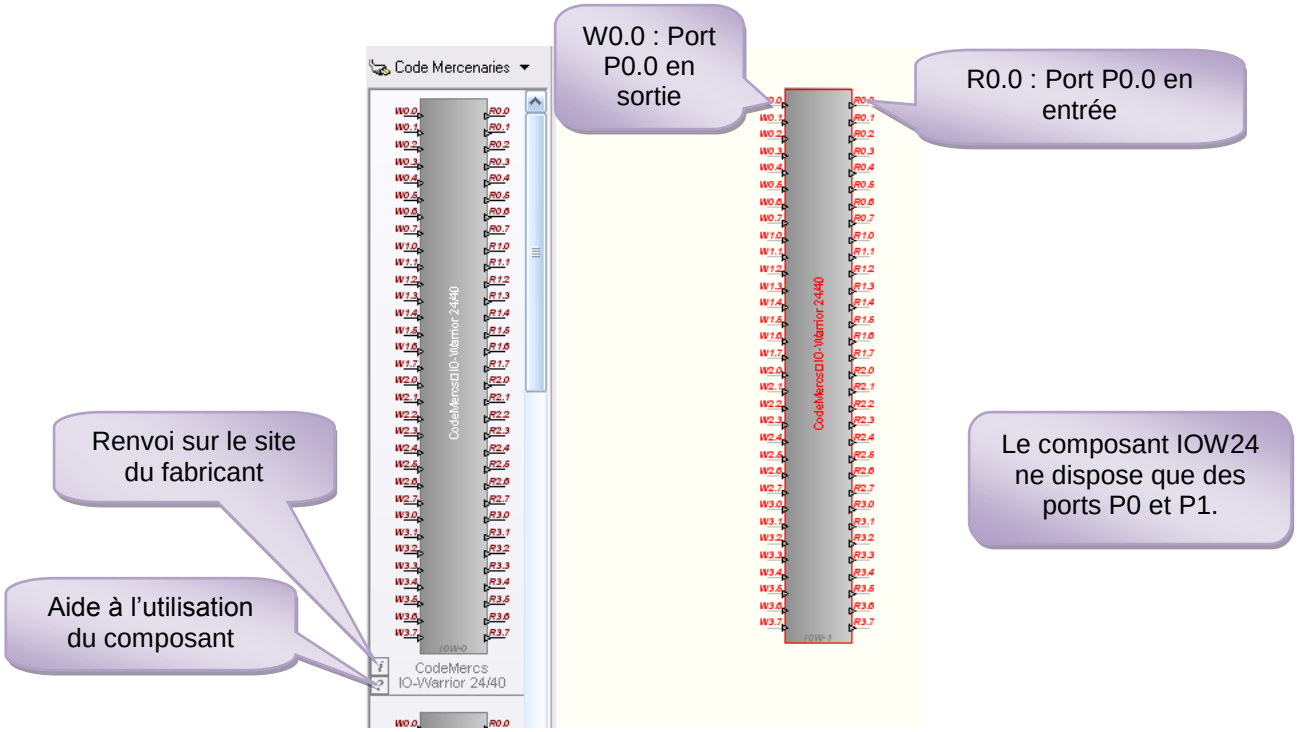
Le logiciel peut piloter un certain nombre de matériel grâce à des bibliothèques spéciales. L'une d'entre elles contient le composant IOW24.

- Dans le menu des librairies de composants, sélectionner '*Composants Hardware*', puis le fabricant *'Code Mercenaries'*.



- Sélectionner le composant '*CodeMercs IO-Warrior 24/40*' en cliquant dessus puis le positionner dans la zone de programmation graphique

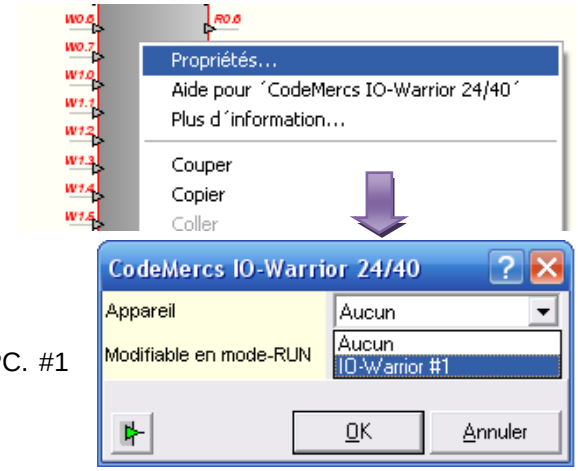
Chaque port peut-être une entrée ou une sortie. Pour matérialiser cela, sur le symbole chaque port est représenté par une entrée et une sortie.



Il faut ensuite lier le symbole du composant avec le composant physique.

- Cliquer droit sur le symbole du composant IOW24 et sélectionner '*Propriétés*'.

- Dans la rubrique '*Appareil*', sélectionner le composant '*IO-Warrior #1*'.

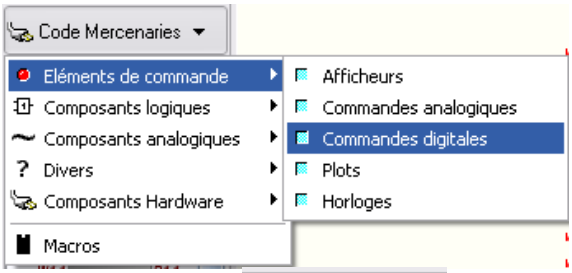


Remarque : Il est possible de connecter plusieurs cartes au PC. #1 permet alors de repérer le premier composant connecté.

Si aucun composant n'apparaît dans la rubrique, il faut alors quitter le logiciel, reconnecter la carte de prototypage au PC, puis relancer le logiciel.

Il faut maintenant placer la commande de la Led. Pour cela, on utilisera un interrupteur.

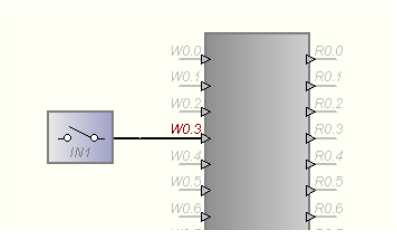
- Dans le menu des librairies de composants, sélectionner '*Eléments de commande'*, puis *'Commandes digitales'.*



- Sélectionner le composant '*Interrupteur*' en cliquant dessus puis le positionner dans la zone de programmation graphique.

Il ne reste plus qu'à relier l'interrupteur au composant IOW24.

- Cliquer sur l'icône  présent sur la colonne à gauche de l'écran. Il permet de réaliser les connexions entre composants.

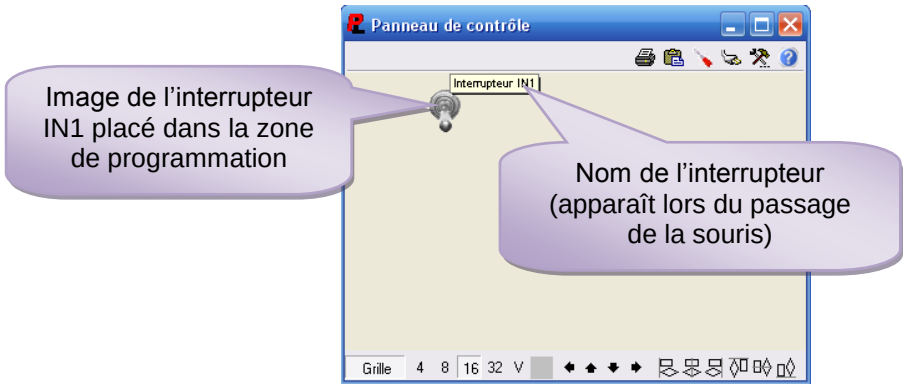


- Relier l'interrupteur à la bonne broche du composant IOW24.

2.2.2 L'interface Homme Machine

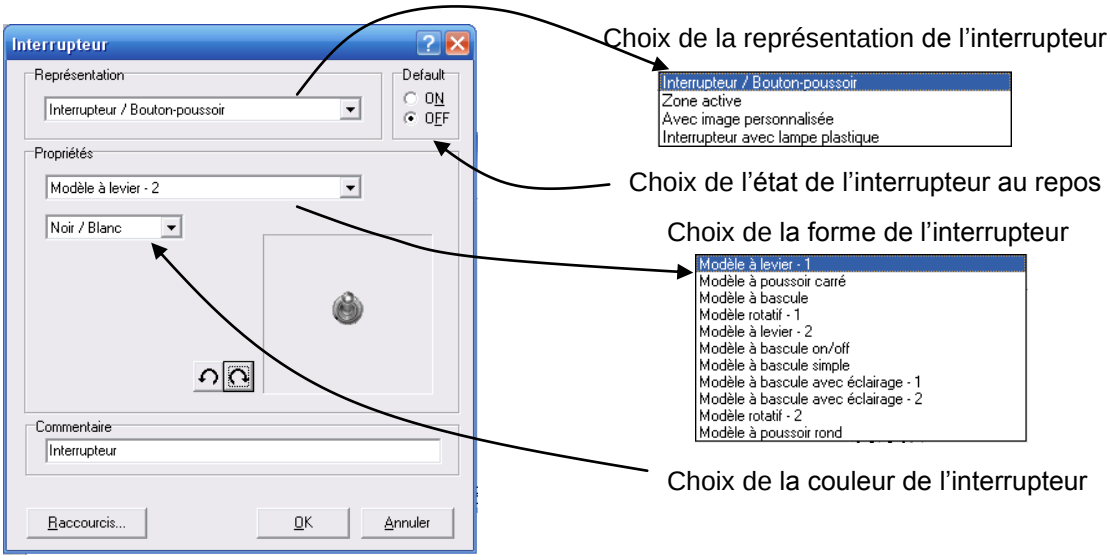
Lors du placement de l'interrupteur dans la zone de programmation graphique, apparaît automatiquement une image de l'interrupteur dans la fenêtre de l'Interface Homme Machine (IHM).

- Cliquer sur l'icône  présent sur la colonne à gauche de l'écran. Il permet de visualiser l'IHM.



Il est possible de modifier l'image de l'interrupteur :

- Double-cliquer sur l'image de l'interrupteur, et définir un modèle.



Remarque : Lorsque l'interrupteur est OFF, sa sortie est à un niveau bas. Sa sortie est au niveau haut lorsqu'il est ON.

2.2.3 Test du programme

- Cliquer sur l'icone  pour lancer l'exécution du programme. Pour modifier l'état de la Led, cliquer sur l'interrupteur de l'IHM.
- Pour arrêter le programme, cliquer sur  .

Pendant l'exécution du programme, il est possible de visualiser l'état des fils et des broches.

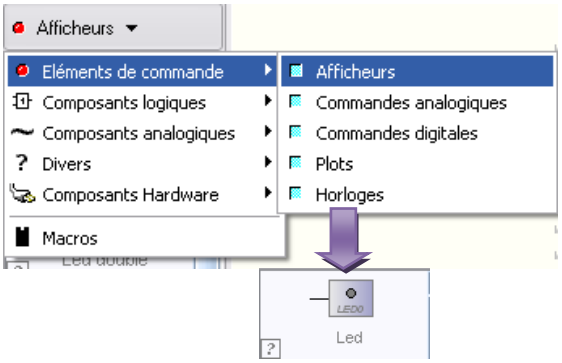
- Cliquer sur l'icône  présent sur la colonne à gauche de l'écran, pour visualiser l'état logique des fils (fil noir : NL0, fil rouge NL1).
- Cliquer sur l'icône  présent sur la colonne à gauche de l'écran, pour visualiser l'état logique des broches.

2.2.4 Amélioration du programme

Parce que l'état de la commande n'est pas toujours visible, on souhaite le faire apparaitre sur l'interface graphique.

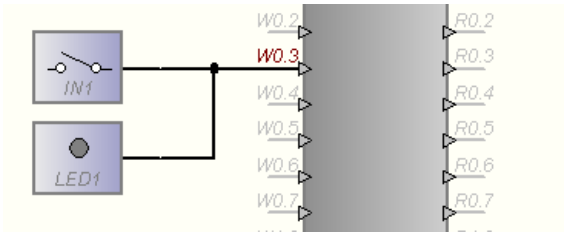
Pour cela, on utilisera un voyant représentatif de l'état de l'interrupteur.

- Dans le menu des librairies de composants, sélectionner '*Eléments de commande*', puis *'Afficheurs'.*
- Sélectionner le composant '*Led*' en cliquant dessus puis le positionner dans la zone de programmation graphique.
- Relier le voyant Led à l'interrupteur de commande.



Remarque : L'ajout du voyant Led dans le programme fait apparaître un voyant rond (vert ou gris) dans l'IHM.

Le programme est le suivant :



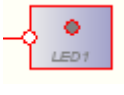
- Tester le programme. Justifier l'allumage du voyant Led sur l'IHM.

Visualiser l'état de la commande n'est pas toujours suffisant, il peut être intéressant de visualiser l'état de la Led (sur la carte de prototypage) sur l'IHM.

- Déplacer le voyant Led pour lire l'état de la sortie du composant IOW24.

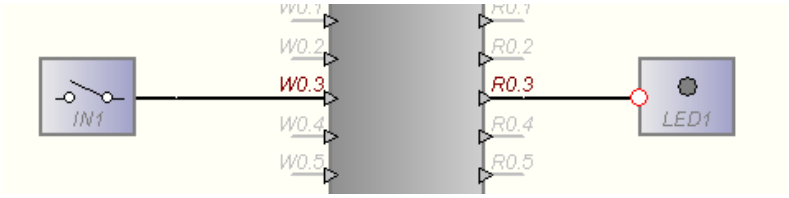
Remarque : Sur le schéma structurel de la carte (page 4), le port P0.3 est relié à la cathode de la led, il faut donc un NL0 pour allumer la led.

Donc si niveau bas permet d'allumer la Led, il doit aussi allumer le voyant Led. Il faut donc placer un inverseur entre la sortie P0.3 de l'IOW24 et l'entrée du voyant Led.



- Placer le curseur de la souris sur la broche de la Led, un inverseur apparaît. Cliquer à ce moment et l'entrée du voyant Led est inversée.

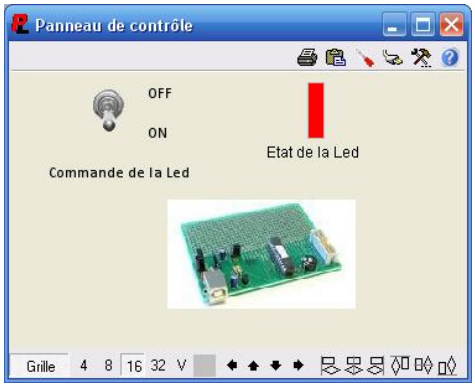
Le programme est le suivant :



- Tester le programme. Vérifier l'allumage du voyant Led sur l'IHM.

2.2.5 Amélioration de l'Interface Homme Machine

On souhaite réaliser une interface graphique du type suivant :



- Faire apparaître l'IHM.
- Pour ajouter du texte, cliquer droit et sélectionner 'Ajouter texte' dans le menu contextuel. Double cliquer sur le texte qui apparaît à l'écran et compléter.
- Pour ajouter une image, cliquer droit et sélectionner 'Ajouter image' dans le menu contextuel. Double cliquer sur le cadre qui apparaît et renseigner.

Remarque : ProfiLab Expert n'utilise que le format Bitmap pour les images.

- Pour modifier le voyant Led, double cliquer dessus et modifier la couleur et le style.

3 Réalisation d'une DLL pour ProfiLab Expert

Le composant IOW24 est fourni avec une bibliothèque de fonctions écrites en C.

Nous allons pouvoir réaliser la même opération que précédemment, toujours sous ProfiLab Expert, mais en utilisant cette fois cette bibliothèque. Pour cela, il faut créer une DLL (Dynamic Link Library).

La DLL est réalisée en C avec quelques commandes C++ à l'aide du logiciel DevC++ (bien entendu un autre logiciel peut-être utilisé).

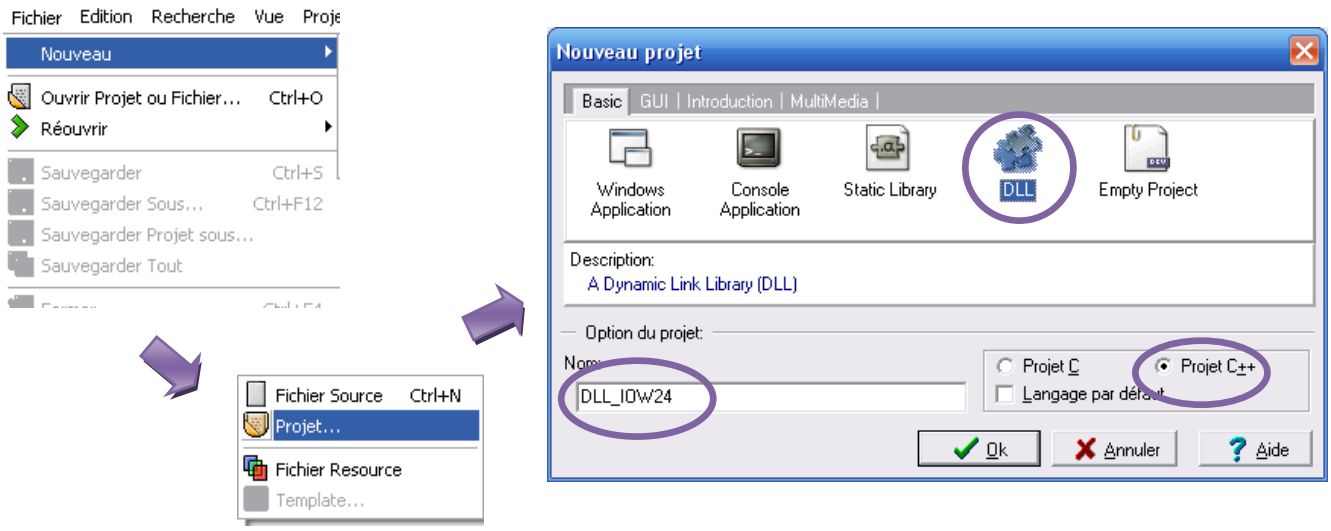
3.1 DevC++

- Avant la création de la DLL, créer un dossier de travail, nommé '*DLL_IOW24*' sur l'ordinateur. Ajouter les 3 fichiers suivants (fournis par le formateur) dans ce dossier : iowkit.lib, iowkit.h et iowkit.dll (bibliothèque de fonctions pour IOW24).

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 .

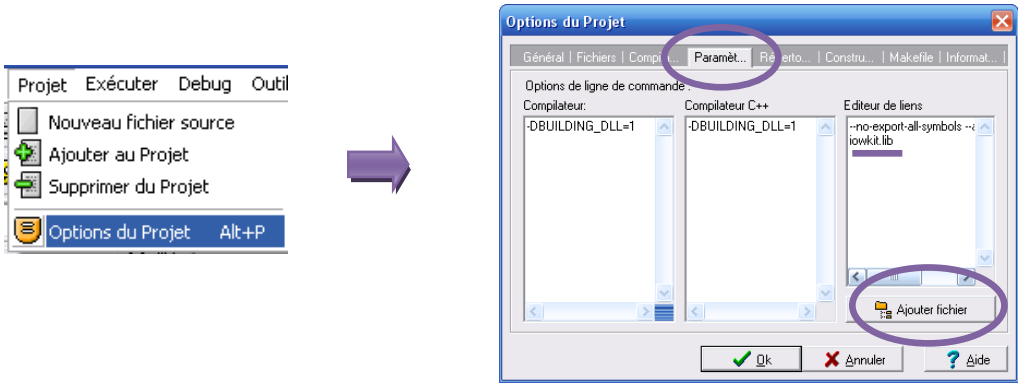


- Sélectionner l'application 'DLL', sélectionner 'projet C++', puis entrer un nom de fichier pour la DLL et cliquer OK.
- Sélectionner le dossier de travail '*DLL_IOW24*' et un nom de projet.

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.

Afin d'utiliser le composant IOW24, il convient d'intégrer sa bibliothèque de fonctions au projet.

- 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.

3.2 Programmation

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

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

- Remplacer le texte par défaut par celui-ci :

```
#include <windows.h>
#include "iowkit.h"
#include "fonction.cpp"

#ifndef DLL_IOW24
#define DLL_IOW24

#define DLLEXPORT extern"C" __declspec(dllexport)

#endif
```

Remarques :

- ☒ La fonction '*__declspec(dllexport)*' permet au compilateur de générer les noms d'exportation des fonctions (dans un fichier avec l'extension '*.a*'.
- ☒ Le code 'externe «C»' permet de conserver les noms des fonctions (sans modification) à la compilation.

- Dans l'arborescence du projet, cliquer sur le fichier '*maindll.cpp*'.

De même que pour le fichier précédent, le logiciel a créé une architecture qui ne convient pas à l'application. Il convient de 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 l'utilisation d'un certain nombre de fonctions bien définies :

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
void _stdcall CSimStart(double *PInput, double *POutput, double *PUser)	Définir les conditions initiales du composant DLL lors de l'exécution du programme ProfiLab Expert
void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)	Définir le fonctionnement interne du composant DLL lors de l'exécution du programme ProfiLab Expert
void _stdcall CSimStop(double *PInput, double *POutput, double *PUser)	Définir l'arrêt du programme Profilab Expert
void _stdcall CConfigure(double *PUser)	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 :
- Ajouter dans le fichier '*dll.h*' la liste des fonctions.

```
/*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)
{
}
```

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.

3.2.1 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.

Dans notre cas, il n'y a qu'une entrée.

- Compléter de la manière suivante le programme C de la fonction NumInputs :

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

3.2.2 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.

Dans notre cas, il n'y a qu'une sortie.

- Compléter de la manière suivante le programme C de la fonction NumOutupts :

```
// Retourne le nombre de sorties
DLLEXPORT unsigned char _stdcall NumOutputs()
{
          return 1;
}
```

3.2.3 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)

- Compléter de la manière suivante le programme C de la fonction GetInputName :

```
// 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 ;
      }
}
```

3.2.4 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)

- Compléter de la manière suivante le programme C de la fonction GetOutputName :

```
// Retourne le nom des sorties
DLLEXPORT void _stdcall GetOutputName(unsigned char Channel,unsigned char *Name)
{
            switch(Channel)
            {
    case 0: // Sortie 0
            {
            Name[0]= 'P';
            Name[1]= 'O';
            Name[2]= '.';
            Name[3]= '3';
            Name[4]= 0;
            break;
            }
    default:;
            }

}
```

3.2.5 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

3.2.5.1 Initialisation de la sortie

La sortie est au niveau logique 0 au départ. Elle est la sortie numéro 0. La valeur de cette sortie est accessible par la variable POutput[0].

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 mettre 5.

3.2.5.2 Initialisation du composant IOW24

Cette fonction doit également initialiser la communication avec le composant IOW24.

Pour cela, il faut utiliser la bibliothèque de fonctions du IOW24 fourni par le fabricant (Voir le document constructeur) :

La fonction qui va servir pour l'application est la suivante :

Nom de la fonction	Rôle
IOWKIT_HANDLE IOWKIT_API IowKitOpenDevice(void)	Ouvrir la communication avec le composant IOW24 détecté sur l'USB.

Remarques :

- ☒ Cette fonction renvoie une variable de type IOWKIT_HANDLE. Ce type est défini dans le fichier d'en-tête '*iowkit.h*'. Il correspond à un pointeur universel (sans type défini : void *). Cette variable permet de distinguer le composant IOW24 d'un autre périphérique.
- ☒ IOWKIT_API est défini comme la commande '*__stdcall*' vu précédemment.
- ☒ Si la fonction renvoie la valeur NULL, cela signifie que la communication n'est pas ouverte.

3.2.5.3 Programme

- Compléter de la manière suivante le programme C de la fonction CSimStart :

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

        // Open device
        devHandle = IowKitOpenDevice();
        if (devHandle == NULL)
            MessageBox(NULL,TEXT("Echec lors de l'ouverture de IOW24"),TEXT("Recherche IOW24"), MB_OK);
        else
        {
            MessageBox(NULL,TEXT("IOW24 détecté"),TEXT("Recherche IOW24"), MB_OK);
            POutput[0] = 0;
        }
}
```

- Attention, il faut déclarer la variable '*devHandle*' en variable globale de type '*IOWKIT_HANDLE*' :

```
// Définition des variables
IOWKIT_HANDLE devHandle;
```

3.2.6 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

3.2.6.1 Ecrire sur le port d'entrées/sorties du composant IOW24

La fonction 'CCalculate' doit pouvoir modifier le port P0.3 du IOW24. Il faut donc pouvoir écrire sur le port d'entrée/sortie.

Pour cela, il faut utiliser la bibliothèque de fonctions du IOW24 fourni par le fabricant (Voir le document constructeur) :

La fonction qui va servir pour l'application est la suivante :

Nom de la fonction	Rôle
ULONG IOWKIT_API IowKitWrite(IOWKIT_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length);	Ecrire des données au composant IOW24 via l'USB.

Cette fonction renvoie une valeur (de type ULONG : entier long non signé) contenant le nombre d'octets écrits.

Elle nécessite le passage de 4 paramètres :

Paramètre	Type	Rôle
devHandle	Pointeur universel (void *)	Permet d'identifier le composant IOW24
numPipe	Entier long non signé	Définit l'interface USB à utiliser pour communiquer
Buffer	Pointeur sur octet	Contient les données à transmettre (sous forme de rapport)
length	Entier long non signé	Indique le nombre d'octets à transmettre

Remarques :

- ☒ L'USB dispose de plusieurs interfaces de communication. Celle utilisée pour commander les ports d'entrée/sortie a pour valeur 0.
- ☒ Les données à transmettre sont représentées par un rapport, une structure de données.

```
typedef struct _IOWKIT24_IO_REPORT
{
        UCHAR ReportID;
        union
        {
            WORD Value;
            BYTE Bytes[2];
        };
}
IOWKIT24_IO_REPORT;
```

ReportID : Octet indiquant l'opération effectuée (NL 0 : lecture/écriture sur les ports)

2 octets contenant les niveaux logiques sur les ports P0 et P1.

- A partir de la structure, on peut déduire que la valeur du paramètre '*length*' est 3.

3.2.6.2 Programme

- Compléter de la manière suivante le programme C de la fonction CCalculate :

```
// Retourne l'état des sorties en fonction des entrées
DLLEXPORT void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)
{
      BYTE valeur;
      IOWKIT24_IO_REPORT rapport;

      if (PInput[0] < 2.5)                        //Si l'entrée de la DLL au NL0
      {
            POutput[0] = 0;                       //Mise à 0 de la sortie de la DLL
            valeur = 0x00;                        //Mise à 0 de la sortie P0.3
      }
      else                                        //Sinon
      {
            POutput[0] = 5;                       //Mise à 1 de la sortie de la DLL
            valeur = 0x08;                        //Mise à 1 de la sortie P0.3
      }

      // Initaliser rapport
      memset(&rapport, 0xff, 3);                  //Mise à 255 des 3 octets de la variable rapport
      rapport.ReportID = 0;                       //Opération écriture sur ports
      rapport.Bytes[0] = valeur;                  //Octet correspondant au Port0 chargé à valeur

      if (!(IowKitWrite(devHandle, 0,(PCHAR) &rapport, 3) == 3))        //Ecriture et test si les
                                                                        //3 octets ont été envoyés
            MessageBox(NULL,TEXT("Echec lors de l'écriture"),TEXT("Ecriture"), MB_OK);
}
```

3.2.7 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

3.2.7.1 Fermeture de la communication avec le composant IOW24

La fonction CSimStop doit terminer la communication USB entre le PC et le composant IOW24.

Pour cela, il faut utiliser la bibliothèque de fonctions du IOW24 fourni par le fabricant. La fonction qui va servir pour l'application est la suivante :

Nom de la fonction	Rôle
void IOWKIT_API IowKitCloseDevice(IOWKIT_HANDLE devHandle)	Fermer la communication avec le composant IOW24.

3.2.7.2 Programme

- Compléter de la manière suivante le programme C de la fonction CSimStop :

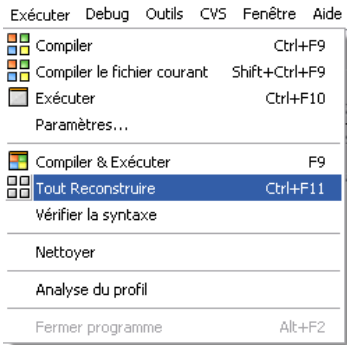
```
// Arrêt du programme
DLLEXPORT void _stdcall CSimStop(double *PInput, double *POutput, double *PUser)
{
     // Close device
     IowKitCloseDevice(devHandle);
}
```

3.2.8 Fonction CConfigure

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

Dans l'application proposée, cette fonction n'a pas d'utilité.

3.3 Création de la DLL



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.

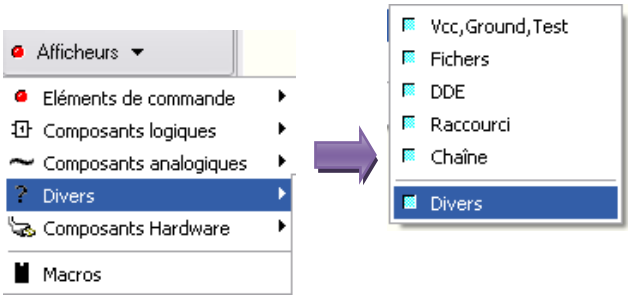
Remarque : Si ProfiLab Expert est ouvert et utilise la DLL, il n'est alors pas possible de recompiler. Il faut absolument le fermer.

4 Programmation en utilisant une DLL

- Sous ProfiLab Expert, créer un nouveau fichier

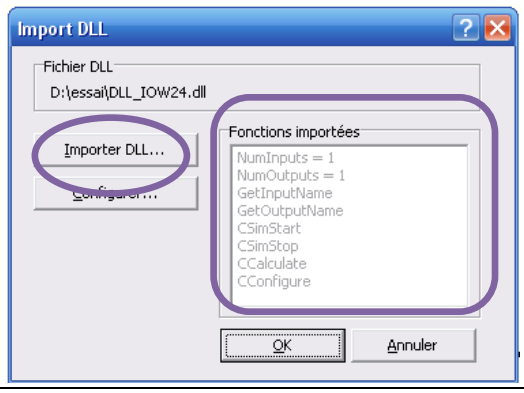
Le symbole du composant IOW24 est maintenant remplacé par le symbole DLL.

- Dans le menu des librairies 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.
- Lorsque l'opération est réussie, la liste des fonctions créées dans la DLL apparaissent dans la fenêtre '*Fonctions importées*'.



- Placer l'interrupteur et le voyant Led dans la zone de programmation graphique.
- Lancer l'exécution du programme et vérifier le fonctionnement.

Annexe : Programme complet de la DLL

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

// Définition des variables
IOWKIT_HANDLE devHandle;


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

// Retourne le nombre de sorties
DLLEXPORT unsigned char _stdcall NumOutputs()
{
	return 1;
}

// 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 ;
	}
}

// Retourne le nom des sorties
DLLEXPORT void _stdcall GetOutputName(unsigned char Channel,unsigned char *Name)
{
	switch(Channel)
	{
		case 0:					// Sortie 0
		{
			Name[0]= 'P';
			Name[1]= '0';
			Name[2]= '.';
			Name[3]= '3';
			Name[4]= 0;
			break;
		}
		default:;
	}

}

// Retourne les conditions au départ
DLLEXPORT void _stdcall CSimStart(double *PInput, double *POutput, double *PUser)
{
	// Open device
	devHandle = IowKitOpenDevice();
	if (devHandle == NULL)
		MessageBox(NULL,TEXT("Echec lors de l'ouverture de IOW24"),TEXT("Recherche IOW24"), MB_OK);
	else
	{
```

```
            MessageBox(NULL,TEXT("IOW24 détecté"),TEXT("Recherche IOW24"), MB_OK);
            POutput[0] = 0;
        }
}

// Retourne l'état des sorties en fonction des entrées
DLLEXPORT void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)
{
        BYTE valeur;
        IOWKIT24_IO_REPORT rapport;

        if (PInput[0] < 2.5)                    //Si l'entrée de la DLL au NL0
        {
            POutput[0] = 0;                     //Mise à 0 de la sortie de la DLL
            valeur = 0x00;                      //Mise à 0 de la sortie P0.3
        }
        else                                    //Sinon
        {
            POutput[0] = 5;                     //Mise à 1 de la sortie de la DLL
            valeur = 0x08;                      //Mise à 1 de la sortie P0.3
        }

        // Initaliser rapport
        memset(&rapport, 0xff, 3);              //Mise à 255 des 3 octets de la variable rapport
        rapport.ReportID = 0;                   //Opération écriture sur ports
        rapport.Bytes[0] = valeur;              //Octet correspondant au Port0 chargé à valeur

        if (!(IowKitWrite(devHandle, 0,(PCHAR) &rapport, 3) == 3))      //Ecriture et test si les
                                                                        //3 octets ont été envoyés
            MessageBox(NULL,TEXT("Echec lors de l'écriture"),TEXT("Ecriture"), MB_OK);
}

// Arrêt du programme
DLLEXPORT void _stdcall CSimStop(double *PInput, double *POutput, double *PUser)
{
        // Close device
        IowKitCloseDevice(devHandle);
}

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