# 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

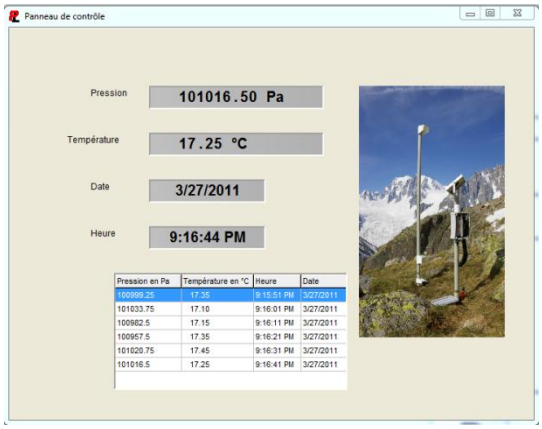
Activité : Autonomie – IOWarrior - Commande du capteur de pression et de la température par bus SPI

**IO-Warrior**

**Generic universal I/O Controller for USB**


# Sommaire

# 1 Présentation




Dans le but de répondre au cahier des charges fixé pour la station météorologique et le central, le StarterKit IOWarrior est utilisé avec un module complémentaire Capteur SCP1000.

La communication entre ces modules va se faire via une liaison SPI gérée par le composant IOW24.

La carte de prototypage est commandée par le PC (port USB).

L'objectif est d'obtenir l'IHM suivante. Il faut pouvoir :

- ☒ afficher la pression
- ☒ afficher la température.

Pour cela, il faut créer une DLL qui sera utilisée dans un programme graphique sous ProfiLab Expert.

# 2 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

Or le mode spécial n'est pas simplement accessible sous ProfiLab Expert. Il faut donc utiliser une DLL.

## 2.1 La liaison SPI

Le Composant IOW24 permet de générer une liaison SPI. Il est toujours l'unique maître de la communication.

La liaison SPI utilise les broches P0.7 (SCK), P0.6 (MISO), P0.5 (MOSI), P0.4 (/SS) et P0.3 (/DRDY) du composant IOW24.

Pour accéder à cette fonction, il faut utiliser le composant IOW24 dans le mode spécial. Le mode spécial est accessible via l'interface 1 de la liaison USB entre le PC et la carte de prototypage.

Pour communiquer sur la liaison SPI (comme sur les autres fonctions spéciales), un rapport (données sous forme de structure en C) composé de 8 octets est utilisé.

## 2.2 Ouverture, fermeture de la liaison SPI

Pour ouvrir ou fermer une liaison SPI, il faut envoyer le rapport suivant :

| ReportID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0x08 out | enable | mode | | 0x00 | 0x00 | 0x00 | 0x00 |

- ☒ Enable : 0x01, liaison initialisée ; 0x00, liaison fermée.
- ☒ Mode : Cet octet permet la configuration du mode SPI
  Bits 7, 6, 5, 4 : mis à 0.
  Bit 3 : CPOL défini le niveau logique de SCK au repos (CPOL=0 SCK au NL0 au repos, CPOL=1 SCK au NL1 au repos)
  Bit 2 = CPHA défini le front à partir duquel on déclenche l'acquisition des données (CPHA =0 déclenchement au 1er front, CPHA =1 déclenchement au second front de SCK).
  Bit 1 (MSB) et Bit 0 (LSB): ils définissent la valeur de la fréquence du bus SPI
  00 - 2MBit/sec
  01 - 1MBit/sec
  10 - 0.5MBit/sec
  11 - 0.0625MBit/sec
- ☒ . Les autres octets sont non utilisés

Remarque : A partir du moment où la liaison SPI est initialisée, les broches P0.7 à P0.3 ne sont plus accessibles en entrées/sorties (interface 0 de la liaison USB). Incompatible avec la gestion de l'affichage LCD

## 2.3 Initialisation du transfert sur la liaison SPI

Pour initialiser la transmission, il faut envoyer le rapport suivant :

| ReportID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0x09 out | flags | data | data | data | data | data | data |

- ☒ Flags : contient 8 bits :
  - ✓ Bit7 : use DRDY : l'esclave signale si et quand il est disponible pour recevoir ou envoyer des données
    use DRDY au niveau logique 1, le maître vérifie que le signal /DRDY est au niveau 0 avant d'envoyer une données si l'esclave souhaite un pause dans la transmission il met /DRDY au niveau logique 1 avant la fin de transmission de l'octet courant.
  - ✓ Bit6 : SSactive : au niveau logique 1, le signal /SS reste actif durant la transmission
  - ✓ Bit5 : ignoreDRDY: niveau logique 1 le /DRDY est non utilisé,
  - ✓ Bit4 : non utilisé, niveau logique 0
  - ✓ Bit3 : non utilisé, niveau logique 0
  - ✓ Bit2: MSB du nb d'octet à transmettre
  - ✓ Bit1
  - ✓ Bit0: LSB du nb d'octet à transmettre

  } Nombre d'octets de données à transmettre (6 au maximum), les suivants seront ignorés

Remarque : Si la transmission porte sur plus de 6 octets et que SSactive est au niveau logique 1 le maître laissera le signal /SS actif pour que l'ensemble des octets transmis soient considérés dans un seul et même transfert.

### 2.4 Lecture sur la liaison SPI

Pendant la transmission on peut lire les valeurs des données dans le rapport suivant :

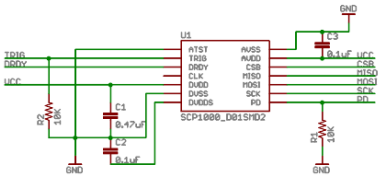
| ReportID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0x09 in | count | data | data | data | data | data | data |

- ☒ count : contient le nombres d'octets de données valides transmis à lire.
- ☒ les octets suivant contiennent les données transmisses

Remarque : On prendra soin de faire suivre les phases d'écriture de phases de lecture afin de ne pas être confronté à des problèmes inhérents aux stockage des données dans le ReportID9.

## 3 Le module capteur de pression et température

Le module utilise le composant SCP1000.


Caractéristiques :

- ☒ Mesure de température de -20°C à +70°C.
- ☒ Résultat sur 14 bits.
- ☒ Différents mode de mesures configurable par programmation
- ☒ Alimentation en 3.3v

Il dispose de plusieurs registres contenant, notamment ceux contenant les valeurs de la température et de la pression du lieu où il est installé.

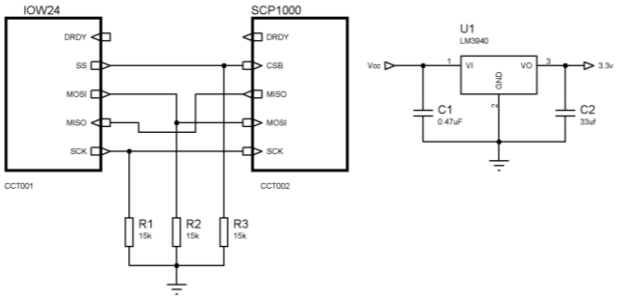
| Adress | Name | Description | Mode | Register acess | With (bits) |
|---|---|---|---|---|---|
| 0x03 | OPERATION | Operation register | RW | Direct | 8 |
| 0x07 | STATUS | Asic top-level status | R | Direct | 8 |
| 0x1F | DATARD8 | Pressure data output MSB | R | Direct | 8 |
| 0x20 | DATARD16 | Pressure data output LSB | R | Direct | 16 |
| 0x21 | TEMPOUT | 14 bit température output data | R | Direct | 16 |

Schéma d'interfaçage:

Compte tenu du niveau de tension du SCP3.3v il faut adapter le 5v de la carte IOwarrior24 en 3.3v pour le SCP 1000. Le Schéma proposé est le suivant:

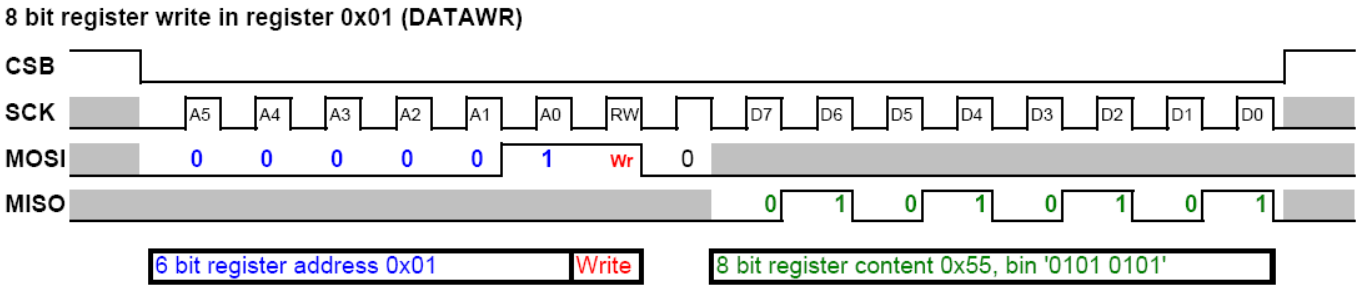
L'IOwarrior24 dispose en interne d'une résistance de pull-up d'environ 8KΩ. L'emploi en externe d'une résistance de pull-down permet de ramener un niveau de tension de 3.26v sur le SCP1000.

Remarque: Pas d'adaptation sur la MISO.



### 3.1 Ecriture dans un registre du SCP1000

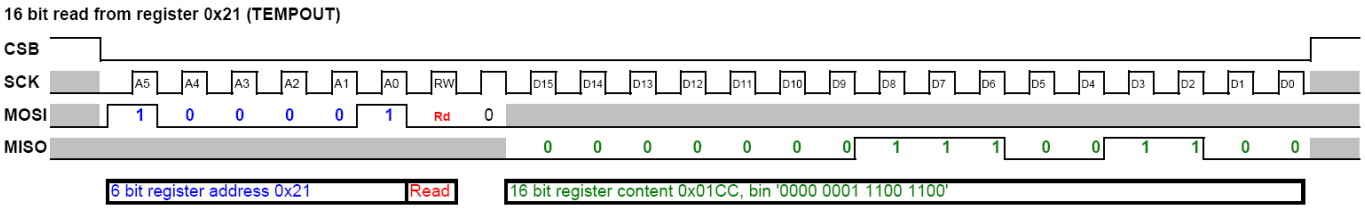
Le fabricant donne les informations présentes sur le bus SPI pendant l'écriture :




La l'écriture se fait à partir

### 3.2 Lecture des données du SCP1000

Le fabricant donne les informations présentes sur le bus SPI pendant la lecture :




## 4 Manipulation

La programmation de la DLL va se faire de manière progressive, en parallèle avec la programmation sous ProfiLab Expert et les relevés de trame.

### 4.1 Définition des entrées et des sorties de la DLL

Pour respecter l'IHM, il faut :

- ☒ 1 entrée :
  - ✓ Pour déclencher l'acquisition de température et de pression [ACQ].
- ☒ 2 sorties :
  - ✓ Pour l'affichage de la pression [PRESS].
  - ✓ Pour l'affichage de la température [TEMP].

- Ouvrir le projet '*DLL_SPI.dev*'. Il contient les fichiers '*dll.h*' et '*dllmain.cpp*'. Ces deux fichiers sont en partie écrits. Les entrées et les sorties sont déjà définies.

- Compiler le projet.

#### 4.1.1 Programmation graphique

- Ouvrir un nouveau document sous ProfiLab Expert.

- Ajouter un composant DLL dans la zone de programmation graphique, double cliquer dessus et importer le fichier dll créé précédemment et cliquer OK.

Les entrées et sorties définies dans apparaissent dans le composant DLL.

Remarque : il ne faut pas oublier de sauvegarder le document et fermer ProfiLab Expert avant de recompiler le projet DLL en C++.

## 4.2 Initialisation et arrêt de la DLL

Il faut définir les conditions de départ et d'arrêt de la DLL. Il faut donc écrire les fonctions '*CsimStart*' et '*Csimstop*'.

### 4.2.1 Fonction CSimStart de la DLL

La fonction '*CSimStart*' doit :

- ☒ Initialiser la communication entre le PC et le composant IOW24 avec la fonction '*IowKitOpenDevice()*' (voir ressource).
- ☒ Ouvrir la liaison SPI en envoyant un rapport (voir paragraphe 2.2) au composant IOW24 avec la fonction suivante :

| Nom de la fonction | Rôle |
|---|---|
| ULONG IOWKIT_API IowKitWrite(IOWKIT_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Ecrire sur les ports d'entrées/sorties du composant IOW24 via l'USB. |

- Compléter la fonction '*CSimStart*' avec le programme suivant :

```
// 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);
            InitSPI(devHandle);

        }
}
```

Elle utilise une fonction '*InitSPI*'.

- Ouvrir le fichier 'fonction_vide.cpp' dans le dossier du projet. Ce fichier est vide. Il contiendra les différentes fonctions créées pour l'application.
- Créer une fonction '*InitSPI*' contenant le programme suivant :

```
BOOLEAN InitSPI(IOWKIT_HANDLE IOWarrior)
{
        IOWKIT_SPECIAL_REPORT rapport;

    memset(&rapport, 0, 8);
    rapport.ReportID = 0x08;          //fonction SPI de IOW24
    rapport.Bytes[0] = 0x01;          //enable SPI
    rapport.Bytes[1] = 0x07;          // /CPOL CPHA freq 0.0625Mhz
    IowKitWrite(IOWarrior, 1, (char *) &rapport, 8);
    memset(&rapport, 0, 8);
    rapport.ReportID = 0x09;          //transfert SPI
    rapport.Bytes[0] = 0x02;          // config envoi de 2 octets
    rapport.Bytes[1] = 0x0E;          // @ du registre sur 6 bits  03 +1 pour write et zero
    rapport.Bytes[2] = 0x0A;          // high resolution continu
    IowKitWrite(IOWarrior, 1, (char *) &rapport, 8);
    IowKitRead(IOWarrior, 1, (char *) &rapport, 8);
}
```

- Expliquer les différentes instructions de ces deux fonctions. On utilisera ces instructions dans les autres fonctions à réaliser.

### 4.2.2 Fonction CSimStop de la DLL

La fonction 'CSimStop' doit :

☒ Fermer la liaison SPI en envoyant un rapport au composant IOW24.
☒ Clore la communication entre le PC et le composant IOW24 avec la fonction '*IowKitCloseDevice()*' (voir TP1).

- En prenant modèle sur la fonction précédente, compléter la fonction '*CSimStop*'. Il faut utiliser une fonction '*CloseSPI*', ayant la même structure (mais des valeurs différentes) que la fonction '*InitSPI* ' (voir datasheet IOWarrior page 16).

```
BOOLEAN CloseSPI(IOWKIT_HANDLE IOWarrior)
{
        IOWKIT_SPECIAL_REPORT rapport;

    memset(&rapport, 0, 8);
    rapport.ReportID = 0x??;          //fonction SPI de IOW24
    rapport.Bytes[0] = 0x??;          //disable SPI
    if (IowKitWrite(IOWarrior, 1, (char *) &rapport, 8) == 8)
       return TRUE;
    else
       return FALSE;
}
```

- Compiler le projet

### 4.2.3 Programmation graphique

- Ouvrir le document de programmation graphique précédent.

- Exécuter le programme et vérifier que le composant IOW24 est détecté et que la liaison SPI est initialisée.

Remarque : Il ne faut pas oublier de connecter la carte de prototypage avant de lancer le logiciel ProfiLab Expert.

## 4.3 Comportement de la DLL

Cette partie définit le comportement de la DLL en fonctionnement.

### 4.3.1 Fonction CCalculate

La fonction '*CCalculate*' doit :

☒ Faire l'acquisition de la pression et de la température si un niveau logique 1 est présent sur la broche ACQ.
☒ Attendre que le SCP1000 est une conversion achevée avec des données prêtes à être lues (voir document SCP1000 D01 page 19/37 registre STATUS bit 5).
☒ Lire la valeur de la pression (voir document SCP1000 D01 pages 19/37 et 20/37 registres DATARD8 et DATARD16).
☒ Lire la valeur de la température (voir document SCP1000 D01 page 20/37 registre TEMPOUT).

Pour cela, il faut utiliser les fonctions suivantes :

| Nom de la fonction | Rôle |
|---|---|
| ULONG IOWKIT_API IowKitWrite(IOWKIT_HANDLE devHandle, ULONG numPipe, PCHAR buffer, | Ecrire sur les ports d'entrées/sorties du composant IOW24 via l'USB. |

| Nom de la fonction | Rôle |
|---|---|
| ULONG length); | |
| ULONG IOWKIT_API IowKitRead(IOWKIT_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Lire des données du composant IOW24 via l'USB. |

Dans ce but, 3 fonctions *'attente SCP'*, '*Lecture_press*' et '*Lecture_temp*' sont créées dans le fichier '*fonction.cpp*'.

- Créer ces 3 fonctions à partir des trames suivantes :

```
BOOLEAN attenteSCP(IOWKIT_HANDLE IOWarrior, double *POutput)
{
// attente données de conversion valide
   do
    {
    memset(&rapport, 0, 8);
    rapport.ReportID = 0x09;          //report ID 09
    rapport.Bytes[0] = 0x02;          //recup 2 octets
    rapport.Bytes[1] = 0x1C;          // adresse du registre 07 puis 0 (pour lecture) puis 0  puis 2 octects envoyés
    IowKitWrite(IOWarrior, 1, (char *) &rapport, 8);
    IowKitRead(IOWarrior, 1, (char *) &rapport, 8);
    }
    while ((rapport.Bytes[2] && 0x20) ==0); //masque pour ne tester que le bit DRDY (b5) du registre  STATUS $07
}
```

```
//Mesure de la pression
    memset(&rap_press_1, 0, 8);
    rap_press_1.ReportID = 0x??;          //report ID 09
    rap_press_1.Bytes[0] = 0x??;          //recup 2 octets
    rap_press_1.Bytes[1] = 0x??;          //adresse 1F registre DATARD8 puis lecture puis 3 octects
    IowKitWrite(IOWarrior, 1, (char *) &rap_press_1, 8);
    memset(&rap_press_1, 0, 8);
    IowKitRead(IOWarrior, 1, (char *) &rap_press_1, 8);
    pression_1 = ( rap_press_1.Bytes[?]<<16); //decalage du MSB de la pression

    memset(&rap_press_2, 0, 8);
    rap_press_2.ReportID = 0x??;          //report ID 09
    rap_press_2.Bytes[0] = 0x??;          //recup 3 octets
    rap_press_2.Bytes[1] = 0x??;          //adresse 20 registre DATARD16 puis lecture puis 3 octects
    IowKitWrite(IOWarrior, 1, (char *) &rap_press_2, 8);
    memset(&rap_press_2, 0, 8);
    IowKitRead(IOWarrior, 1, (char *) &rap_press_2, 8);
    pression_2 = ( rap_press_2.Bytes[?]<<8 |  rap_press_2.Bytes[?]); // decalage des LSB
    pression = (pression_1 + pression_2)/4; //recomposition de la valeur de la pression
    return (pression);
}
```

```
double lecture_temp(IOWKIT_HANDLE IOWarrior, double *POutput)
{

    //Mesure de la temperature
    memset(&rap_temp, 0, 8);
    rap_temp.ReportID = 0x??;          //report ID 09
    rap_temp.Bytes[0] = 0x??;          //recup 3 octets
    rap_temp.Bytes[1] = 0x??;          //21+1+1+3 octect
    IowKitWrite(IOWarrior, 1, (char *) &rap_temp, 8);
    memset(&rap_temp, 0, 8);
    IowKitRead(IOWarrior, 1, (char *) &rap_temp, 8);
    temperature = (double)(( rap_temp.Bytes[?]<<8 |  rap_temp.Bytes[?]))/20;
    return (temperature);

}
```

- Compléter la fonction '*CCalculate*' du fichier 'dllmain.cpp' à partir de la trame suivante :

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

 if(PInput[???]>??)
          {
          attenteSCP(devHandle, POutput);
          lecture_???(devHandle, POutput);
          POutput[PRESS] = ????;
          lecture_?????(devHandle, POutput);
          POutput [TEMP] = ?????;
          }
 else
 {

 }
}
```

- Compiler le projet pour obtenir la DLL.

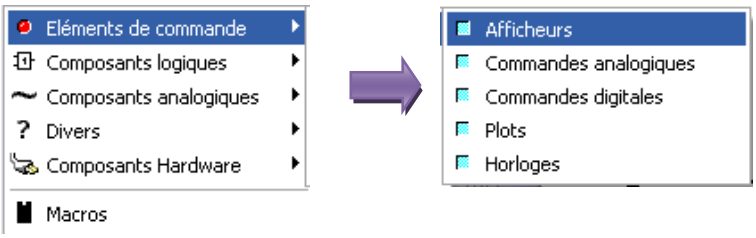
### 4.3.2 Programmation graphique

- Ouvrir le document de programmation graphique précédent.

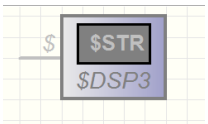
Il convient de rajouter des composants pour pouvoir tester la DLL :

☒ Un bouton poussoir connecté à l'entrée ACQ.
☒ Un afficheur connecté à la sortie PRESS
☒ Un afficheur connecté à la sortie TEMP.

- Pour visualiser les résultats de la lecture, il faut ajouter des composants afficheurs pour chaque sortie.
- Sélectionner la librairie '*Eléments de commande*' puis '*Afficheurs*' :

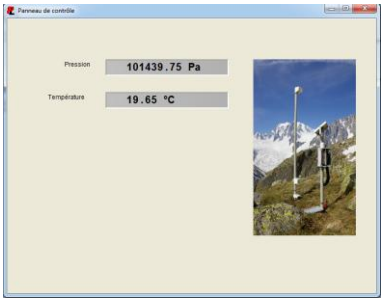



- Sélectionner le composant '$*Afficheur* '.




A partir de ces informations, connecter ces deux composants à un afficheur.

- Arranger l'IHM pour qu'elle ressemble à celle proposée ci-après.




Il est possible de modifier les paramètres de l'IHM :

- Cliquer droit sur l'IHM et sélectionner '*Propriétés...*' dans le menu contextuel.

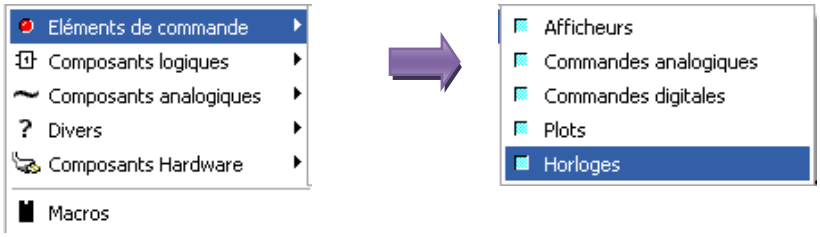
- Modifier le titre, ajouter une image, fermer la barre d'outils...
-

- Exécuter le programme et vérifier que sur un appui "bouton poussoir" on procède bien à une acquisition et un affichage de la pression et de la température.

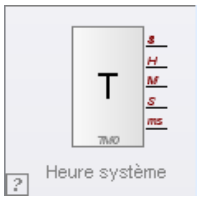
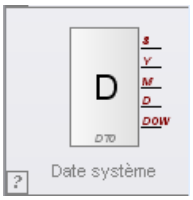
Afin d'horodater les mesures de pression et de température nous allons utiliser les possibilités offertes par le logiciel ProfiLabExpert.

Pour prendre en compte l'heure et la date, le logiciel dispose de deux composants fournissant l'heure et la date du PC .

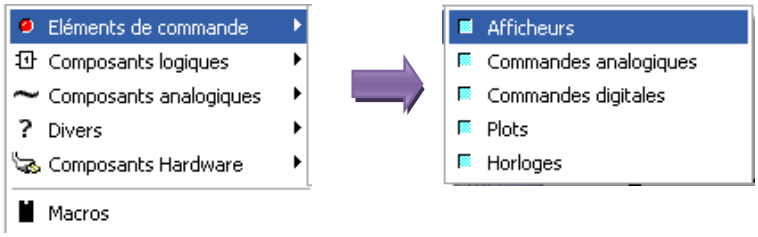
- Sélectionner la librairie '*Eléments de commande*' puis '*Horloges*' :



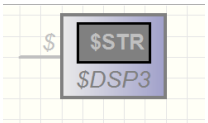

- Sélectionner les composants '*Heure système*' et '*Date système*'.

| | |
|---|---|
|   |   |
| Fournit :<br>☒ H : valeur numérique des heures<br>☒ M : valeur numérique des minutes<br>☒ S : valeur numérique des secondes<br>☒ ms : valeur numérique des millisecondes<br>☒ $ : chaîne de caractères contenant H, M, S et ms. | Fournit :<br>☒ Y : valeur numérique de l'année<br>☒ M : valeur numérique du mois<br>☒ D : valeur numérique de la date<br>☒ DOW : valeur numérique du jour de la semaine<br>☒ $ : chaîne de caractères contenant Y, M, D et DOW. |

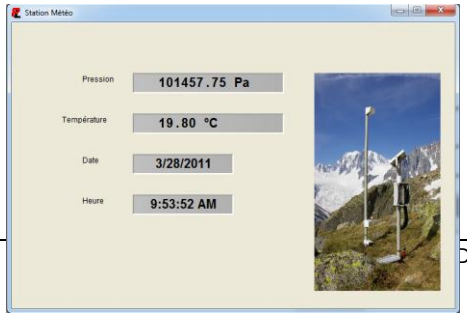
- A partir de ces informations, connecter ces deux composants à un afficheur..
- Sélectionner la librairie '*Eléments de commande*' puis '*Afficheurs*' :




- Sélectionner le composant '$*Afficheur* '.




- Arranger l'IHM pour qu'elle ressemble à celle proposée ci-contre.



### 4.3.3 Relevés des trames SPI

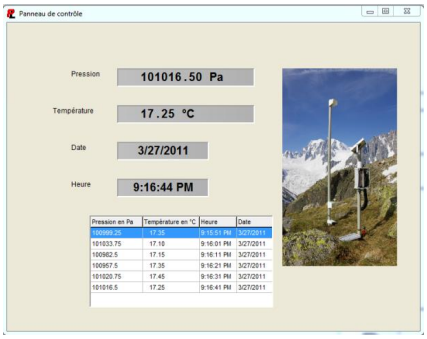
Selon le matériel à disposition, on pourra utiliser un oscilloscope ou un analyseur logique

- Configurer le matériel pour relever et décoder une trame SPI en mode monocoup.
- Relever la trame SPI lors de l'initialisation du mode SPI. Avec l'analyseur logique, on pourra faire la mesure pour différentes fréquences d'échantillonnage.
- Mesurer la fréquence de l'horloge SCK et vérifier sa valeur.
- Relever la trame lors de la lecture de la valeur de la température.

# 5 Amélioration de l'IHM

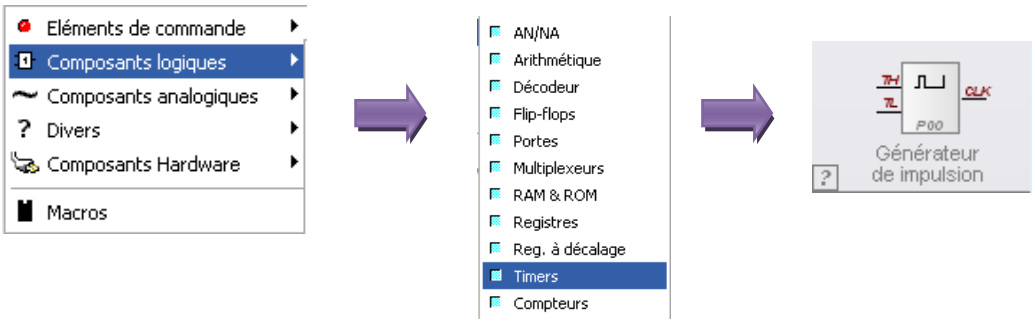
Jusqu'à présent, l'acquisition est faite par appui sur un bouton poussoir.

On souhaite remplacer ce mode de fonctionnement par un mode de "relevés automatique" toutes le 5s (on imposera dans le cahier des charges 30s mais pour éviter d'attendre inutilement nous prendrons 5s pour ce TP) et de mémoriser ces relevés dans une table de valeurs. L'IHM souhaitée est celle proposé ci-contre:



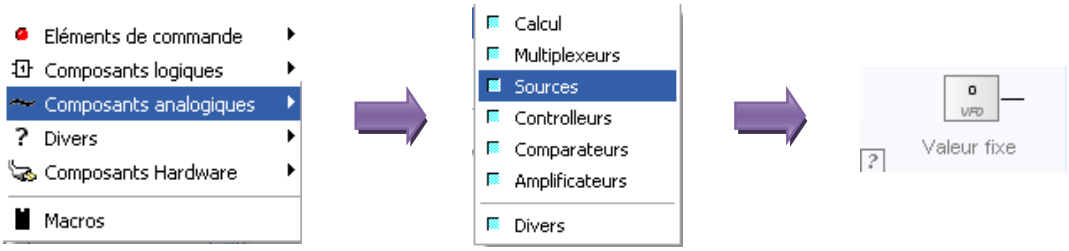

## 5.1 Ajouter un générateur de signaux

- Sélectionner la librairie '*Composants logiques*' puis '*Timers*'. Sélectionner le composant '*Générateur d'impulsion'* :




Ce composant génère un signal périodique dont les durées à l'état bas (TL) et à l'état haut(TH) sont programmables. Pour obtenir une période de 5 secondes, on prendra TL = 4.9s et TH = 0.1s

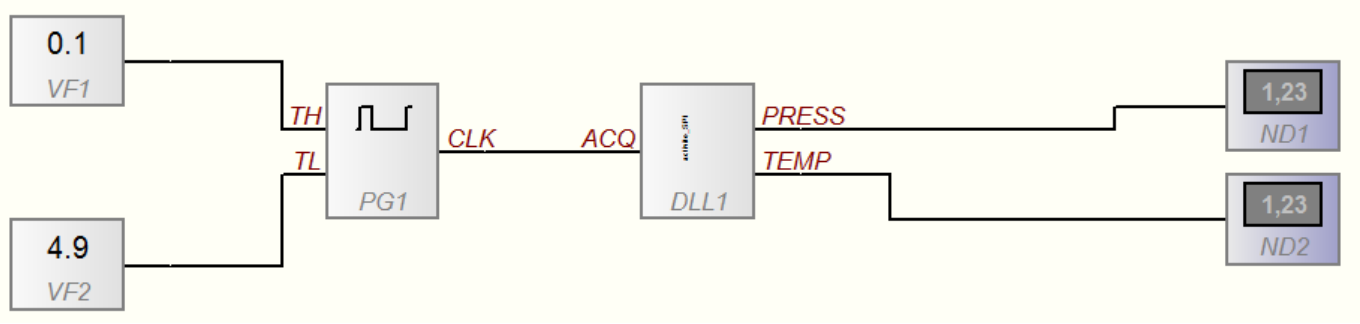
- Sélectionner la librairie '*Composants analogiques*' puis '*Sources*'. Sélectionner le composant '*Valeur fixe'* :




- Connecter ce composant sur les entrées du générateur d'impulsion et configurer les valeurs TH et TL en double cliquant dessus.

- Compléter les connexions entre composants et modifier l'IHM.

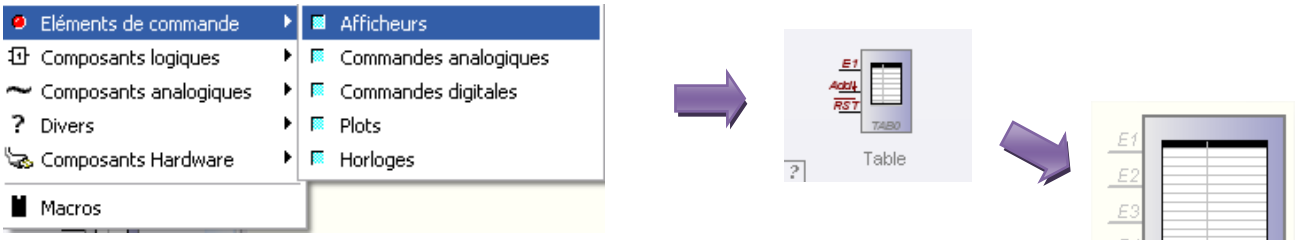
- Tester et valider le programme.




## 5.2 Ajouter une table de valeurs

Le cahier des charges impose un enregistrement des mesures horodatées. Il est possible d'enregistrer les mesures dans un fichier de type texte, Excel, Word.

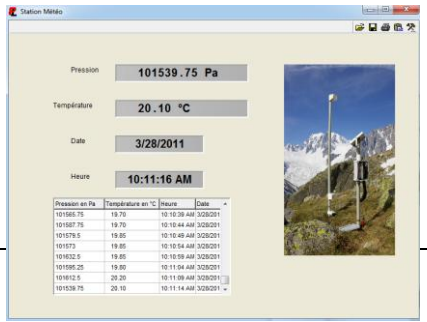
- Sélectionner la librairie '*Eléments de commande*' puis '*Afficheurs*'. Sélectionner le composant '*Table'* :




- Double cliquer sur le composant et sélectionner le nombre d'entrée à mémoriser. Taper 2 pour mémoriser la pression et la température. L'heure et la date du PC sont automatiquement insérées dans la table.

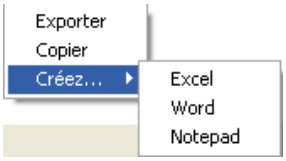
- 2 Entrées 'E' apparaissent. Les relier aux différentes sorties de la DLL.

- L'entrée '*Add*' du composant permet de mémoriser les entrées sur front descendant. Relier cette entrée à l'entrée '*ACQ'* de la DLL.
- Modifier l'IHM. Double cliquer sur la table et nommer les colonnes, définir la largeur des



colonnes, ainsi que le format des informations.

Pendant l'exécution du programme, cliquer droit sur la table un menu contextuel apparaît. Cliquer sur '*Créez...*' pour enregistrer les données de la table sous le format désiré.

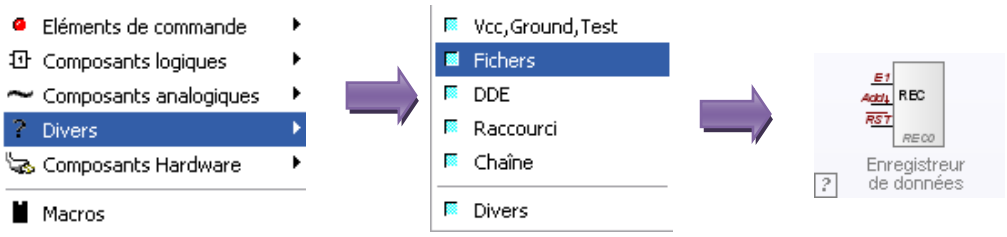



Remarques :

- ☒ Les données ne sont par enregistrées dans le fichier au fur et à mesure.
- ☒ La table a une contenance de 16000 valeurs par colonne.

Il est possible de mémoriser les données dans un fichier automatiquement.

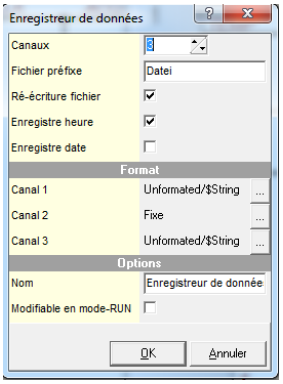
- Sélectionner la librairie '*Divers*' puis '*Fichiers*'. Sélectionner le composant '*Enregistreur de données'*. Le mettre à la place du composant '*Table'*.




- Double cliquer sur le composant et sélectionner le nombre d'entrée à mémoriser. Taper 3 pour mémoriser la date, la pression et la température. L'heure du PC est automatiquement présente dans le fichier généré.

- Définir le format de l'information pour chaque entrée, le nom di fichier d'enregistrement. Pour la pression et la date on choisira le format Unformated/$String .

- 3 Entrées 'E' apparaissent. Les relier aux 2 sorties de la DLL et à la sortiedu composant '*Date système*' .




L'entrée '*Add*' du composant permet de mémoriser les entrées sur front descendant. Relier cette entrée à l'entrée '*ACQ'* de la DLL.

La table a disparu de l'IHM.

Lors de l'exécution du programme, les données sont enregistrées dans le fichier à chaque front descendant sur l'entrée '*Add'*. Le fichier est fermé dès l'arrêt du programme.

- Retrouver le fichier créé dans le dossier suivant :

C:\Program Files\ProfiLab-Expert40\Data