Sciences et technologies
de l'Industrie et du développement durable

SIN : 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 : TP2 – IOWarrior - Commande de l'horloge temps réel

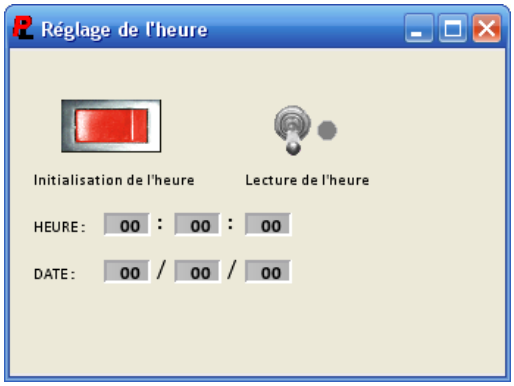
IO-Warrior

Generic universal I/O Controller for USB

Code Mercenaries

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 des modules complémentaires RTC et Capteur.

La communication entre ces modules va se faire via une liaison I2C gérée par le composant IOW24.

Au cours de cette activité, seul le module RTC est utilisé. La carte de prototypage est commandée par le PC (port USB).

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

- ☒ initialiser l'heure
- ☒ lire l'heure
- ☒ afficher l'heure.

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 I2C

Le Composant IOW24 permet de générer une liaison I2C. Il est toujours l'unique maître de la communication.

La liaison I2C utilise les broches P0.1 (SCL) et P0.2 (SDA) 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 I2C (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 I2C

Pour ouvrir ou fermer une liaison I2C, il faut envoyer le rapport suivant :

ReportID	1	2	3	4	5	6	7
0x01	enable	flags	timeout	0x00	0x00	0x00	0x00

- ☒ Enable : 0x01, liaison initialisée ; 0x00, liaison fermée.
- ☒ Flags : le bit de poids fort de cet octet permet de connecter des résistances de pull-up internes au composant IOW24 sur les broches SDA et SCL (au niveau logique 0)
- ☒ Timeout : temps qu'attend le IOW24 pour que l'esclave libère le signal SCL après l'acquittement.
- ☒ Les autres octets n'interviennent pas.

Remarque : A partir du moment où la liaison I2C est initialisée, les broches P0.1 et P0.2 ne sont plus accessibles en entrées/sorties (interface 0 de la liaison USB).

2.3 Ecriture sur la liaison I2C

Pour écrire sur la liaison I2C, il faut envoyer le rapport suivant :

ReportID	1	2	3	4	5	6	7
0x02 out	flags	data	data	data	data	data	data

- ☒ Flags : contient 8 bits :
  - ✓ Bit7 : start : au niveau logique 1, un bit de start est généré avant l'envoi des données
  - ✓ Bit6 : stop : au niveau logique 1, un bit de stop est généré après l'envoi des données
  - ✓ Bit5 : non utilisé, niveau logique 0
  - ✓ Bit4 : non utilisé, niveau logique 0
  - ✓ Bit3 : non utilisé, niveau logique 0
  - ✓ Bit2
  - ✓ Bit1
  - ✓ Bit0

  } Nombre d'octets de données à transmettre (6 au maximum)

Remarque : il est possible d'envoyer plus de 6 octets. Pour cela envoyer un premier rapport avec uniquement le bit start généré. Envoyer autant de rapport, sans bits start et stop générés, que nécessaire. Envoyer un dernier rapport avec uniquement le bit stop généré.

Après l'écriture, il est possible d'obtenir l'acquittement de l'opération par un rapport d'erreur, sur 8 octets :

ReportID	1	2	3	4	5	6	7
0x02 in	flags	0x00	0x00	0x00	0x00	0x00	0x00

- ☒ Flags : contient 8 bits :
  - ✓ Bit7 : drapeau d'erreur : au niveau logique 1 s'il y a eu erreur (pas d'acquittement de l'esclave sur la liaison I2C).
  - ✓ Bit6 : on utilisé, niveau logique 0
  - ✓ Bit5 : non utilisé, niveau logique 0
  - ✓ Bit4 : non utilisé, niveau logique 0
  - ✓ Bit3 : non utilisé, niveau logique 0
  - ✓ Bit2
  - ✓ Bit1
  - ✓ Bit0

  } Indique le dernier octet transmis correctement

2.4 Lecture sur la liaison I2C

Pour lire sur la liaison I2C, il faut envoyer le rapport suivant :

ReportID	1	2	3	4	5	6	7
0x03 out	count	commande	0x00	0x00	0x00	0x00	0x00

- ☒ Count : contient le nombre d'octets à lire
- ☒ Commande : contient l'adresse de l'esclave et l'opération
- ☒ Les autres octets sont non utilisés

Après l'envoi de ce rapport, les données sont accessibles en lisant le rapport suivant :

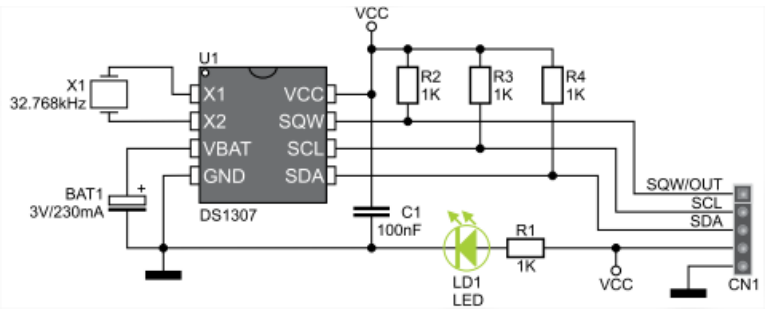
ReportID	1	2	3	4	5	6	7
0x02 in	flags	data	data	data	data	data	data

- ☒ Flags : contient 8 bits :
  - ✓ Bit7 : drapeau d'erreur : au niveau logique 1 s'il y a eu erreur (pas d'acquittement de l'esclave
  - ✓ Bit6 : on utilisé, niveau logique 0
  - ✓ Bit5 : non utilisé, niveau logique 0
  - ✓ Bit4 : non utilisé, niveau logique 0
  - ✓ Bit3 : non utilisé, niveau logique 0
  - ✓ Bit2
  - ✓ Bit1
  - ✓ Bit0

  } Indique le dernier octet lu correctement
- ☒ Les autres octets contiennent les données lues.

3 Le module horloge temps réel

Le module horloge temps réel ou RTC utilise le composant DS1307.



Il a pour adresse I2C : 0b1101000

Il dispose de plusieurs registres contenant, notamment, l'heure la date et le jour codée en BCD :

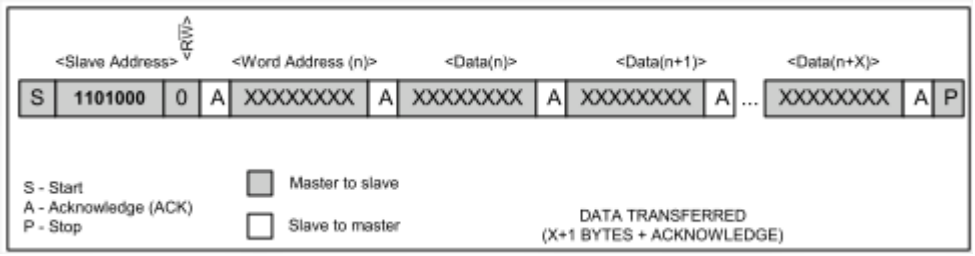
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12 24	10 Hour PM/ AM	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Le registre '*Control*' gère la sortie 'SQW'. Elle n'aura pas d'utilité dans l'application.

Un registre pointeur mémorise l'adresse interne du registre accessible.

3.1 Ecriture dans le module RTC

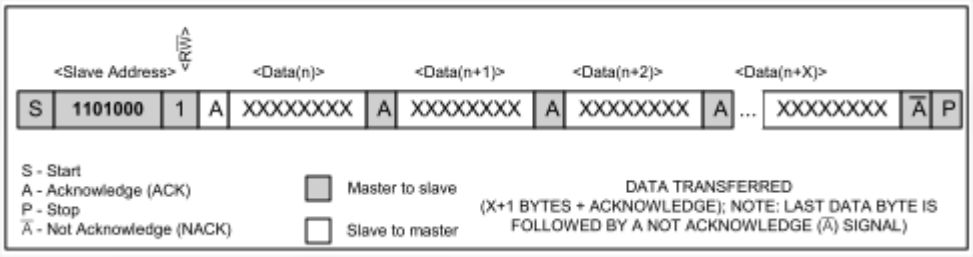
Le fabricant donne les informations présentes sur le bus I2C pendant l'écriture :



Avant l'envoi des données, il faut préciser à partir de quelle adresse, interne au composant DS1307, débute la mémorisation.

3.2 Lecture des données du module RTC

Le fabricant donne les informations présentes sur le bus I2C pendant la lecture :



La lecture se fait à partir de l'adresse contenue dans le registre pointeur. Il convient donc d'initialiser ce registre pointeur avant de faire la lecture.

Pout initialiser le registre pointeur, il faut écrire dans le module RTC. L'octet repéré '*Word Address*' est placé dans le registre pointeur.

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 :

- ☒ 8 entrées :
  - ✓ Seconde, minute, heure, date, mois et année pour initialiser le module RTC.
  - ✓ Init pour effectuer l'initialisation du module RTC.
  - ✓ RD pour la lecture de la date et de l'heure du module RTC.
- ☒ 6 sorties :
  - ✓ Seconde, minute, heure, date, mois et année pour afficher.

- Ouvrir le projet '*DLL_RTC.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 I2C en envoyant un rapport (voir paragraphe 1.2) 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)
{
      BOOLEAN rc;

      // Open device
      IOWarrior = IowKitOpenDevice();
      if (IOWarrior == NULL)
          MessageBox(NULL,TEXT("Echec lors de l'ouverture de IOW24"),TEXT("Recherche IOW24"), MB_OK);

      else
          MessageBox(NULL,TEXT("IOW24 détecté : initialisation de la liaison I²C"),TEXT("Recherche IOW24"), MB_OK);

      rc = InitI2C(IOWarrior);
      if (!rc)
          MessageBox(NULL,TEXT("Echec pour initialiser l'I2C"),TEXT("Ecriture"), MB_OK);
}
```

Elle utilise une fonction '*InitI2C*'.

- Ouvrir le fichier 'fonction.cpp' dans le dossier du projet. Ce fichier est vide. Il contiendra les différentes fonctions créées pour l'application.
- Créer une fonction '*InitI2C*' contenant le programme suivant :

```
// Initialisation de la liaison I²C
BOOLEAN InitI2C(IOWKIT_HANDLE IOWarrior)
{
     IOWKIT_SPECIAL_REPORT rapport;

    memset(&rapport, 0, 8);
    rapport.ReportID = 0x01;
    rapport.Bytes[0] = 0x01;
    if (IowKitWrite(IOWarrior, 1, (char *) &rapport, 8)  == 8)
       return TRUE;
    else
       return FALSE;
}
```

- 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 I2C 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 '*CloseI2C*', ayant la même structure (mais des valeurs différentes) que la fonction '*InitI2C*'.

- 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 I2C 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 :

☒ Initialiser la RTC si l'entrée '*Init*' est au niveau logique haut. Les valeurs présentes sur les entrées de la DLL sont envoyées vers la RTC (voir paragraphe 3).
☒ Lire la RTC si l'entrée RD est au niveau logique haut. Il faut lire le contenu de la RTC et envoyer les données sur les sorties de la DLL.
☒ Vérifier s'il y a erreur à chaque transmission.

Pour cela, il faut utiliser les fonctions suivantes :

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.
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, trois fonctions '*InitRTC*' et '*Lecture_heureRTC*' et '*Lecture_dateRTC*' sont créées dans le fichier '*fonction.cpp*'.

- Créer ces trois fonctions à partir des trames suivantes :

```
// Initialisation de la RTC
void InitRTC(IOWKIT_HANDLE IOWarrior, double *PInput)
{
    IOWKIT_SPECIAL_REPORT rapport;

    memset( ??, ??, ??);                // Mise à 0 de rapport
    rapport.ReportID = ???;             // fonction écriture
    rapport.Bytes[0] = ???;             //Start + 6 bytes
    rapport.Bytes[1] = ???;             //Adresse + ecriture (plus de 6 octets à envoyer donc pas de stop)
    rapport.Bytes[2] = ???;             //Adresse du 1er octet de la rtc
    rapport.Bytes[3] = decimal_vers_bcd((UCHAR)PInput[????]);  //seconde
    rapport.Bytes[4] = decimal_vers_bcd((UCHAR)PInput[????]);  //minutes
    rapport.Bytes[5] = decimal_vers_bcd((UCHAR)PInput[????]);  //heures
    rapport.Bytes[6] = ???;                                    //jour
    IowKitWrite( ???, ???, (char *) ???, ???);                 //Ecriture des données
    IowKitRead( ???, ???, (char *) ???, ???);                  //Leture des erreurs
    if (rapport.Bytes[0] > ???)                                //Test si erreur
       MessageBox(NULL,TEXT("Erreur de transmission"),TEXT("InitialisationRTC"), MB_OK);
    memset(???, ???, ???);            // Mise à 0 de rapport
    rapport.ReportID = ???;           //Fonction écriture
    rapport.Bytes[0] = ???;           //3 bytes + stop  (envoi des dernières données)
    rapport.Bytes[1] = decimal_vers_bcd((UCHAR)PInput[???]);   //date
    rapport.Bytes[2] = decimal_vers_bcd((UCHAR)PInput[???]);   //mois
    rapport.Bytes[3] = decimal_vers_bcd((UCHAR)(PInput[???]-2000));  //année
    IowKitWrite( ???, ???, (char *) ???, ???);                 //Ecriture des données
    IowKitRead( ???, ???, (char *) ???, ???);                  //Lecture des erreurs
    if (rapport.Bytes[0] > ???)                                //Test si erreur
       MessageBox(NULL,TEXT("Erreur de transmission"),TEXT("InitialisationRTC"), MB_OK);
}
```

,

```
// Lecture de l'heure de la RTC
IOWKIT_SPECIAL_REPORT Lecture_heureRTC(IOWKIT_HANDLE IOWarrior)
{
     IOWKIT_SPECIAL_REPORT rapport;

     //initialisation du pointeur
                                              // Mise à 0 de rapport
                                              //Fonction écriture
                                              //Start + stop + 2 bytes
                                              //Adresse + ecriture
                                              //Adresse du 1er octet de la rtc
                                              //Ecriture des données
                                              //Lecture des erreurs
                                              //Test si erreur


     //demande de lecture des 3 octets
                                              // Mise à 0 de rapport
                                              //Opération lecture
                                              //nombre d'octets à lire
                                              //Adresse + lecture
                                              //Ecriture des données
                                              //Lecture des erreurs et données


     return rapport;                          //Retourner les données lues
}
```

```
// Lecture de la date de la RTC
IOWKIT_SPECIAL_REPORT Lecture_dateRTC(IOWKIT_HANDLE IOWarrior)
{
    IOWKIT_SPECIAL_REPORT rapport;


    //initialisation du pointeur
                                                // Mise à 0 de rapport
                                                //Fonction ecriture
                                                //Start + stop + 2 bytes
                                                //Adresse + ecriture
                                                //Adresse du 1er octet de la date de la rtc
                                                //Ecriture des données
                                                //Lecture des erreurs
                                                //Test si erreur


    //demande de lecture des 3 octets
                                                // Mise à 0 de rapport
                                                //Fonction lecture
                                                //nombre d'octets à lire
                                                //Adresse + lecture
                                                //Ecriture des commandes
                                                //Lecture des erreurs et données


                                                //Retourner les données lues
}
```

- Compléter la fonction '*CCalculate*' à partir de la trame suivante :

```
// Lire les entrées et écrire sur la sortie
DLLEXPORT void _stdcall CCalculate(double *PInput, double *POutput, double *PUser)
{
      IOWKIT_SPECIAL_REPORT rapport_RTC;
      BOOLEAN rc;

      if(PInput[???] > ???)                         //Si entrée Init au niveau haut
                                                    //Initialisation de la RTC
      else                                          //Sinon
      {
          if(PInput[???] > ???)                     //Si entrée RD au niveau haut
          {
              rapport_RTC = ???;                     // Lecture de l'heure de la RTC
              if( ??? )                              //Si erreur de transmission
                    MessageBox(NULL,TEXT("Erreur de lecture"),TEXT("LECTURE"), MB_OK);
              else                                  //Sinon
              {
                                                    //Affecter les valeurs lues
                                                    //aux sorties correspondantes de la DLL


              }
              rapport_RTC = ???;                    //Lecture de la date de la RTC
              if( ??? )                             //Si erreur de transmission
                    MessageBox(NULL,TEXT("Erreur de lecture"),TEXT("LECTURE"), MB_OK);
              else                                  //Sinon
              {
                                                    //Affecter les valeurs lues
                                                    //aux sorties correspondantes de la DLL


              }
          }
      }
}
```

- Compiler le projet pour obtenir la DLL.

4.3.2 Programmation graphique

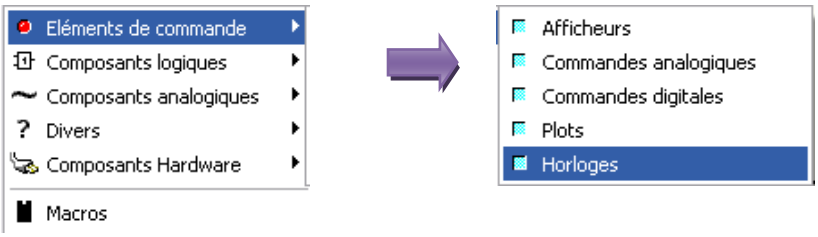
- Ouvrir le document de programmation graphique précédent.

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

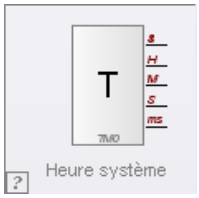
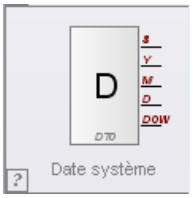
- ☒ Un interrupteur et un voyant Led connecté à l'entrée RD.
- ☒ Un interrupteur à l'entrée Init.

Pour initialiser 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*' :



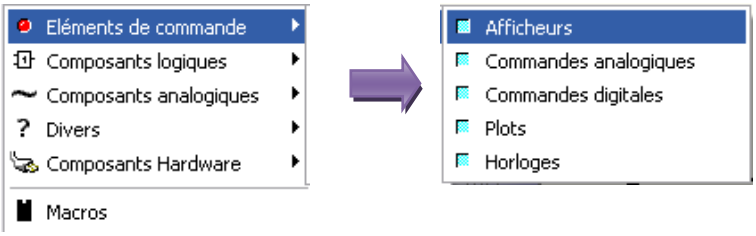
- Sélectionner les composants '*Heure système*' et '*Date système*'.

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

- A partir de ces informations, connecter ces deux composants au composant DLL.

Pour visualiser les résultats de la lecture, il faut ajouter des composants afficheurs pour chaque sortie.

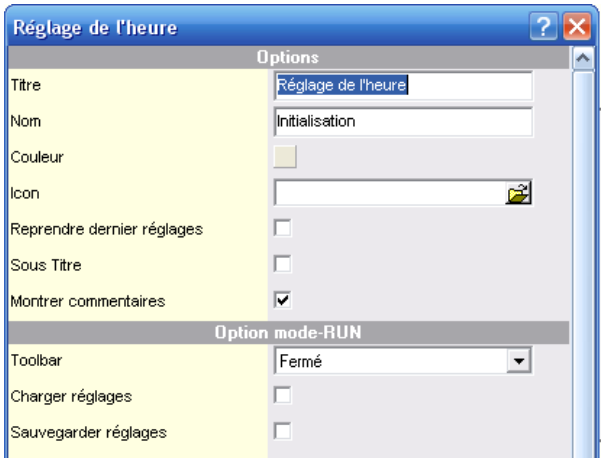
- Sélectionner la librairie '*Eléments de commande*' puis '*Afficheurs*' :



- Sélectionner le composant '*Afficheur numérique*'.

- Arranger l'IHM pour qu'elle ressemble à celle du cahier des charges.

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, fermer la barre d'outils...

4.3.3 Relevés des trames I2C

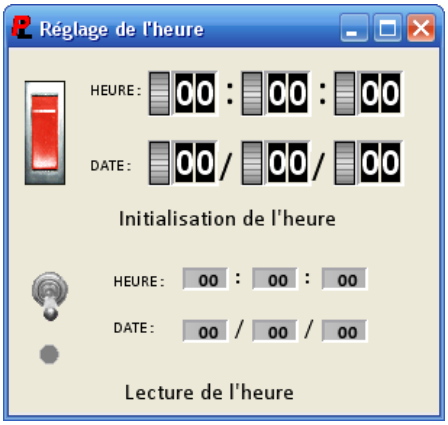
Selon le matériel à disposition, on pourra utiliser un oscilloscope ou un analyseur logique

- Configurer le matériel pour relever et décoder une trame I2C en mode monocoup.
- Relever la trame I2C lors de l'initialisation de l'horloge temps réel. Avec l'analyseur logique, on pourra faire la mesure pour différentes fréquences d'échantillonnage.
- Mesurer la fréquence de l'horloge SCL et vérifier sa valeur.
- Relever la trame lors de la lecture de l'heure et de la date.

5 Amélioration de l'IHM

Jusqu'à présent, l'initialisation de la RTC est réalisée à partir de l'heure du PC.

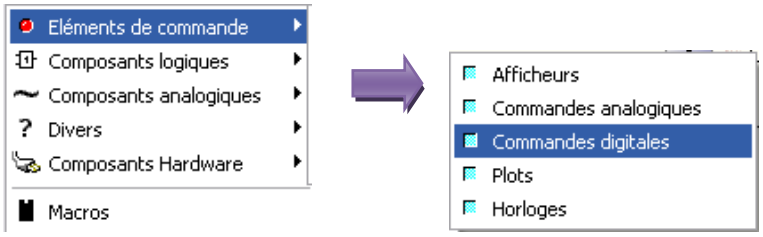
On souhaite remplacer les composants '*Heure système*' et '*Date système*' par des molettes pour pouvoir régler l'heure et la date initiales. L'IHM souhaitée est :



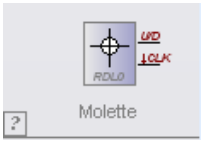
5.1 Réglage manuelle de la date et de l'heure

Ajout des molettes :

- Sélectionner la librairie '*Eléments de commande*' puis '*Commandes digitales*' :



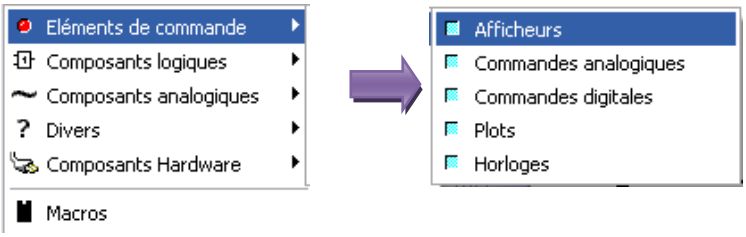
- Sélectionner le composant '*Molette*'. Placer les 6 molettes correspondant aux 6 réglages.

	☒ UD : sens de rotation de la molette (NL1 : incrémentation ; NL0 : décrémentation) ☒ CLK : signal d'horloge avec la présence d'un front descendant pour chaque pas de la molette.

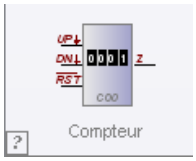
Ces molettes permettent d'augmenter et diminuer la valeur des réglages. Il faut ajouter des compteurs-décompteurs. Dans ProfiLab Expert, il existe un composant compteur avec affichage de la valeur du compteur.

Ajout des compteurs/afficheurs :

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



- Sélectionner le composant '*Afficheur numérique*'.

	☒ UP : incrémentation du compteur sur front descendant. ☒ DN : décrémentation du compteur sur front descendant. ☒ RST : mise à zéro du compteur sur niveau bas. ☒ Z : valeur numérique du compteur.

- Réaliser les connexions entre les différents composants. Il est possible d'ajouter des portes logiques, comparateurs, ...
- Arranger l'IHM pour qu'elle ressemble à celle du cahier des charges.
- Tester et valider le bon fonctionnement de l'IHM.

5.2 Utilisation de macro

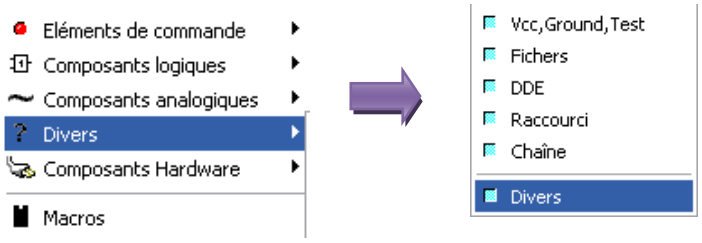
Le programme graphique devient important. Il est plus difficile de s'y retrouver.

Les molettes et les afficheurs représentent une fonction du programme. Nous allons regrouper toute cette fonction et réaliser une macro.

- Ne conserver du programme précédent que la partie permettant de fixer les valeurs numériques des entrées (molettes, compteurs-afficheurs et autres composants associés).

Il faut maintenant fixer les sorties de cette macro :

- Sélectionner la librairie '*Divers*' puis '*Divers*'

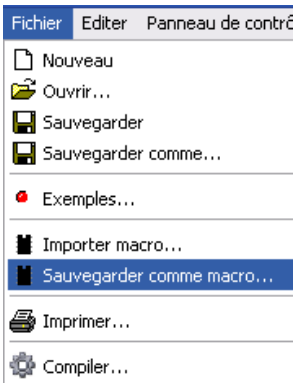


- Sélectionner le composant '*Broche macro*' et le placer sur les 6 sorties de la macro. En double cliquant sur ce composant, on pourra le renommer.

Il faut modifier l'IHM de la macro.

- Sélectionner l'IHM du programme en cours et la modifier pour obtenir ceci :

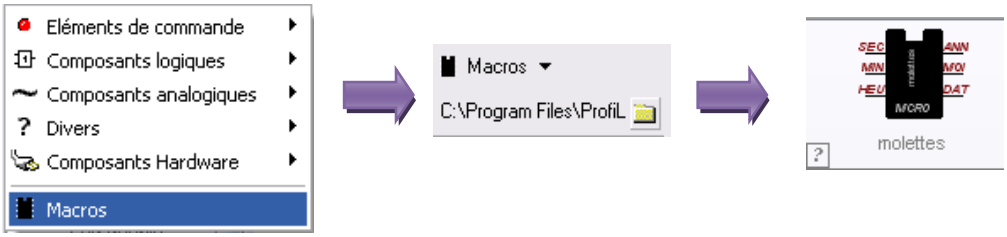
- Pour créer la macro, dans le menu '*Fichier*', cliquer sur '*Sauvegarder comme macro...*'.



- Sélectionner le dossier de travail et nommer la macro.

Lorsque la macro est créée, il faut la placer dans le programme.

- Reprendre le programme graphique complet.
- Enlever tous les composants qui ont été placés dans la macro.
- Sélectionner la librairie '*Macros*' puis rechercher le dossier de travail.



- Compléter les connexions entre composants et modifier l'IHM.
- Tester et valider le programme.

Il est possible d'accéder au programme graphique de la macro en cliquant droit dessus et en sélectionnant '*Editer macro*'.