SYSAM SP5 , langage pascal et option MPI en classe de seconde

Introduction

Certains collègues enseignant l'option MPI en classe de seconde et disposant dans leur établissement de la centrale d'acquisition EXAO SYSAM SP5 de la société Eurosmart, ont souhaité pouvoir réaliser des activités de programmation et d'acquisition simples avec cette centrale, du même type que celles qu'ils réalisaient avec ORPHY® et Visual Basic® : acquisition ponctuelle d'une tension, envoi d'une tension sur une sortie, exploitation des ports logiques pour commander des LED, etc.

EUROSMART, constructeur de l'interface SYSAM SP5, a fourni de la documentation et des exemples de programmation permettant d'accéder à toutes les fonctions d'acquisition, d'émission et d'entrées/sorties logiques, en langage pascal sous l'environnement de programmation DELPHI (anciennement BORLAND, racheté par CodeGear). Cependant, mettre en œuvre les informations de ces documents se révèle ardu, et dépasse largement les compétences des élèves inscrits à cette option (sauf petit génie de la programmation, sait-on jamais…). De plus, l'achat de DELPHI par les lycées peut se révéler onéreux au regard de l'utilisation limitée qui peut en être faite en option MPI.

J'ai donc créé un jeu de commandes simples, faciles à utiliser et correspondant exactement aux besoins de l'enseignement de l'option, que l'on peut mettre en œuvre avec l'équivalent de DELPHI dans la galaxie du logiciel libre, à savoir LAZARUS. Bien que moins performant en vitesse de compilation et autres paramètres, LAZARUS reste largement suffisant pour ce qui est demandé dans l'enseignement de cette option.

Cet article a pour objectifs

-de présenter sommairement la mise en œuvre de LAZARUS ;

-d'expliquer en détail chacune des commandes utiles pour SYSAM SP5 ;

-de décrire la réalisation d'un programme exemple simple utilisant les diverses commandes, à l'aide de LAZARUS.

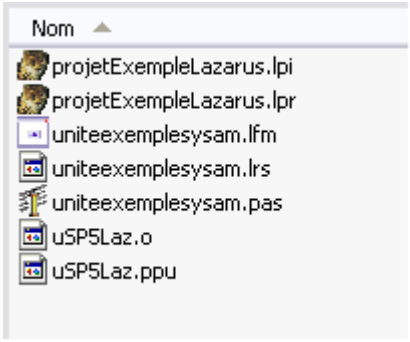
Remarques :

- *On voudra bien prendre en compte le fait qu'il est difficile d'écrire un article de taille raisonnable en ne connaissant pas les compétences du lecteur en informatique : sans doute nombre d'informations seront inutiles ou trop détaillées pour ceux qui ont programmé ORPHY avec VB, et insuffisamment détaillées pour ceux qui commencent ce genre d'activité.*
- *Un certain nombre des opérations décrites ci-dessous peut être réalisé de plusieurs façons sous LAZARUS. Sachant que par manque de temps je n'ai pas exploré de manière*

approfondie les possibilités de LAZARUS, et que cet article est destiné à la mise au point de séquences pédagogiques concernant SYSAM, j'ai systématiquement retenu la manière qui nécessite le moins de compétences, d'explications et d'accès aux options de LAZARUS. Toutes mes excuses aux spécialistes de LAZARUS qui lisent cet article et qui connaissent des astuces plus performantes ; ils peuvent bien sûr corriger les imperfections. Prière de me signaler les éléments à améliorer.

- *Tout ce qui suit n'est valable que sous un système d'exploitation WINDOWS, car l'interface SYSAM SP5 n'a pour l'heure pas de pilote pour d'autres systèmes.*
- *On suppose bien sûr que le pilote de l'interface a été installé.*
- *On suppose également que l'interface reste branchée en permanence. L'effet du débranchement au moment de l'appel des fonctions décrites dans cet article n'a pas été testé…*
- *Les captures d'écran sont celles de la première version de cet article, avec la version la plus récente de Lazarus à cette époque. Il peut y avoir quelques changements dans la version actuelle, généralement dans le sens d'ajouts de rubriques supplémentaires.*

Dossier et fichiers associés à cet article (parties intégrantes du fichier zip à télécharger) : « Projet 1 » avec les fichiers :



Première partie : utiliser LAZARUS.

1. Téléchargement de LAZARUS.

A l'aide de votre navigateur, rendez-vous à l'adresse :
http://sourceforge.net/project/showfiles.php?group_id=89339

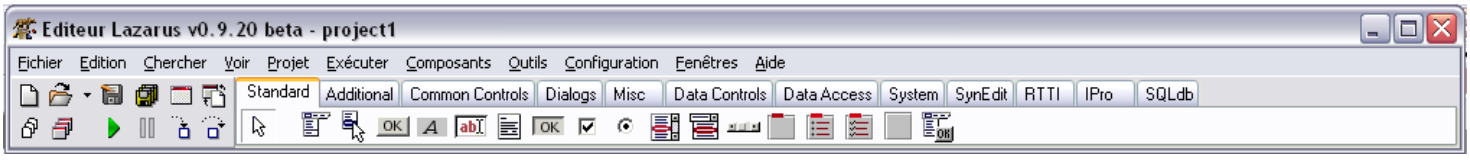
Sélectionnez la version de LAZARUS (Lazarus Windows) et téléchargez cette version. Réalisez la procédure d'installation classique (Attention : le dossier dans lequel se fera l'installation ne peut pas être dans le classique « Program files » car dans ce nom de dossier, il y a un espace… Acceptez simplement le dossier proposé par défaut lors du processus d'installation)

2. Le projet de programme sous LAZARUS.

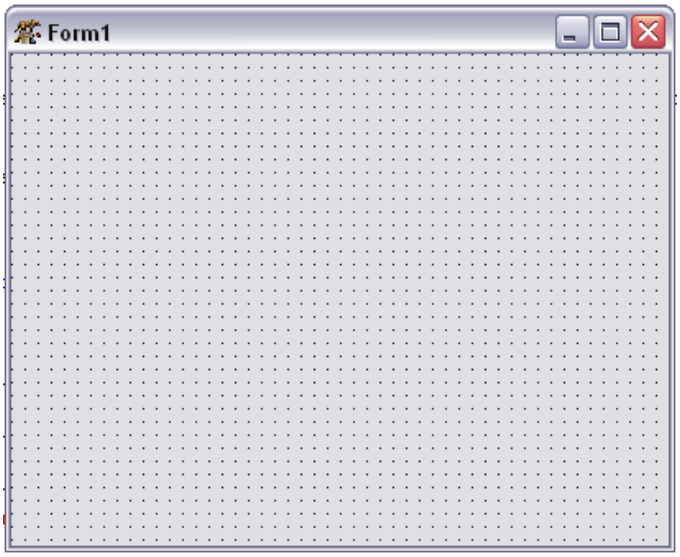
Lorsqu'on veut réaliser un programme, on travaille avec un ensemble de fichiers qui constituent le « projet » et qui doivent être placés dans un dossier unique (un dossier différent pour chaque projet)

a. Le projet LAZARUS et l'EDI (environnement de développement intégré)

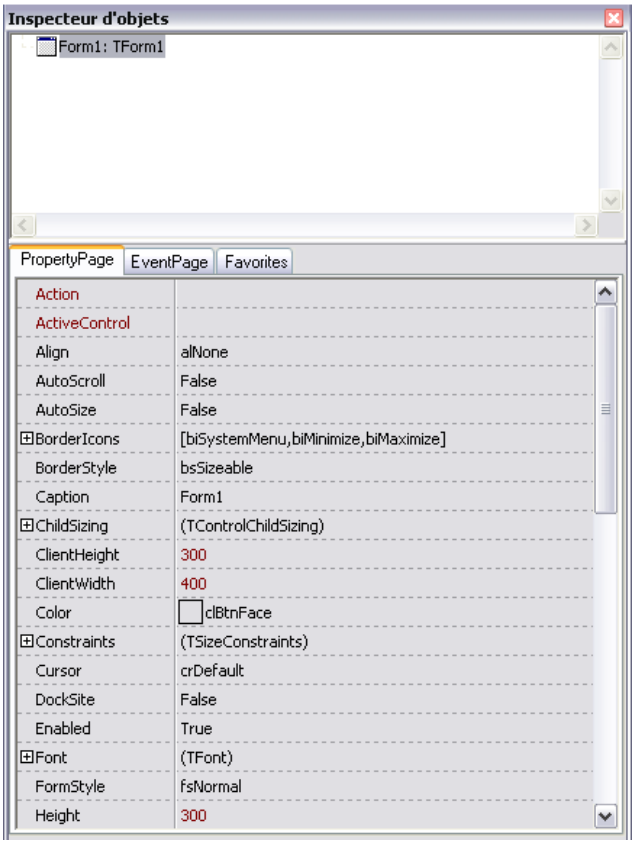
Au (premier) lancement on voit apparaître quatre fenêtres **La fenêtre principale de Lazarus,** avec le menu, le bouton de compilation/lancement du programme et les palettes de composants.



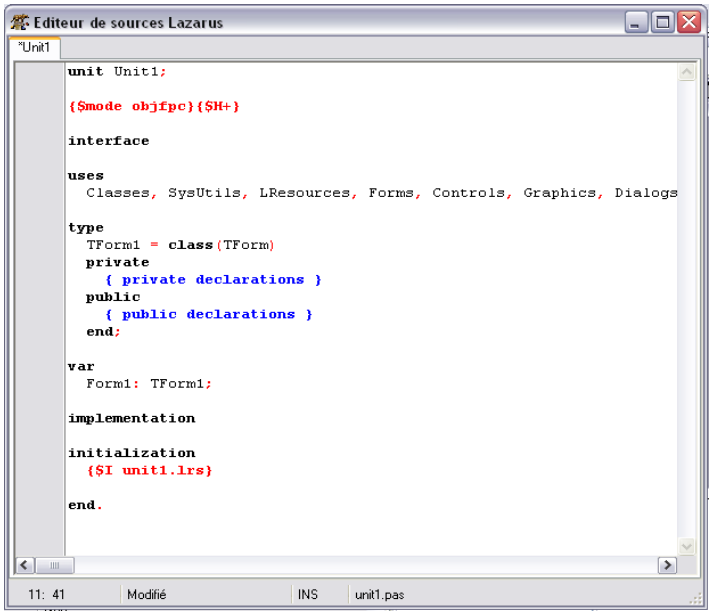
La fiche chantier, qui est une image de ce que sera la fenêtre du programme, sur laquelle on place des composants « prélevés » dans la palette des composants de la fiche principale



L'inspecteur d'objets dont le rôle est de choisir ou régler les propriétés de la fiche et des composants qu'elle contient, au futur démarrage du programme.



L'éditeur de code, où on placera les instructions que devra exécuter le programme.
Lors de la sauvegarde du projet, le contenu de l'éditeur est enregistré dans un fichier d'extension .pas.



Lors de la sauvegarde du projet, les propriétés des objets définies dans l'inspecteur d'objet sont enregistrées dans deux fichiers d'extensions respectives .lfm et .lrs.

Les trois fichiers cités ne diffèrent que par leur extension : la racine du nom est choisie lors du premier enregistrement du projet; ce choix est suivi du choix du nom commun des deux fichiers-projets, dont les extensions sont lpi et lpr.

Le projet est donc constitué au minimum de cinq fichiers.

Touches utiles (il est recommandé de les mémoriser):

F9 : compilation et lancement du programme : LAZARUS utilise les informations de l'inspecteur d'objet et le code de l'éditeur pour créer sur le disque dur un fichier-programme exécutable (.exe) et ensuite lance le fichier-programme.
F10 : passage en avant plan de la fenêtre principale de Lazarus (menu, barre d'outils, palette de composants…)
F11 : passage en avant plan de l'inspecteur d'objets
F12 : passage alterné en avant plan de la fiche chantier et de l'éditeur de code.

Si vous compilez maintenant le projet (touche F9), vous obtiendrez dans le programme une fenêtre que vous pourrez déplacer, réduire, maximiser et fermer, mais votre logiciel ne fera rien…

Pour créer un programme original, on sélectionne des composants en cliquant avec la souris (fenêtre principale de Lazarus dans la palette standard par exemple) et on les place sur la fiche chantier (en cliquant sur cette fiche, un composant à la fois). On peut ensuite les « saisir », les placer où on veut sur la fenêtre, et leur donner les dimensions que l'on veut…

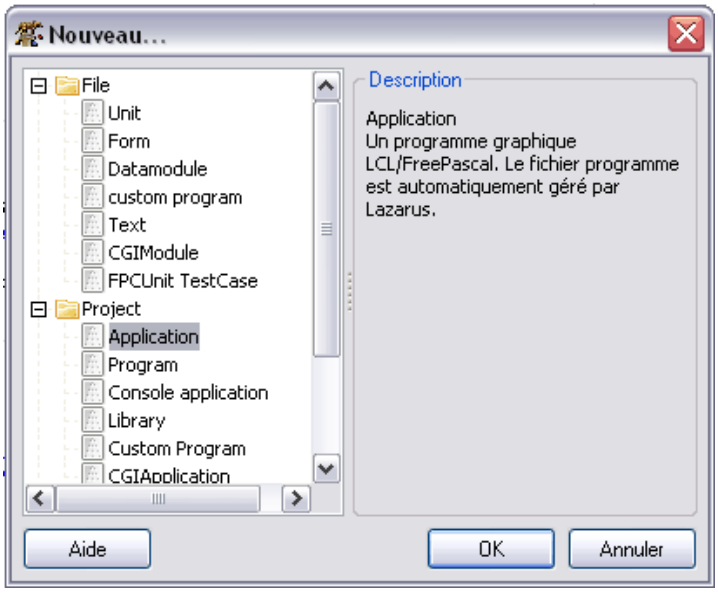
Puis on écrit du code (suite d'instructions) qui est exécuté lors d'évenements
→ provoqués par le futur utilisateur du programme :
-appuis sur des touches (en rapport avec le composant détenant la focalisation),
-clics avec la souris (en rapport avec les composants sur lesquels on clique)

→ ayant lieu obligatoirement et indépendamment de la volonté de l'utilisateur (celui ci n'ayant alors rien à faire) pour une bonne gestion de la mémoire et des fichiers :
-démarrage du programme
-fin du programme, pour ne citer que les plus simples.

La troisième partie de cet article vous guide pas à pas dans toutes ces opérations pour la réalisation d'un exemple.

b. Créer un nouveau projet

Pour créer un nouveau projet, allez dans Fichier / Nouveau… et choisissez « Project/Application » dans la boîte de dialogue, puis cliquez sur OK.



Enregistrez immédiatement le projet dans un dossier accessible en écriture en choisissant Fichier/Tout enregistrer. Lazarus vous demandera de donner deux noms : un premier pour le groupe des trois fichiers mémorisant la fiche chantier (extensions .pas, .lfm, .lrs) un second pour mémoriser les deux fichiers projet (extensions .lpi et .lpr). Cette opération évitera des problèmes d'accès au disque lors des compilations.

c. Modifier/ compléter un projet existant

On peut également partir d'un projet existant. Cet article est accompagné d'un dossier contenant un exemple de projet complet, avec tous les composants nécessaires et le code de démonstration des possibilités de SYSAM en MPI.

Nous supposons que vous avez téléchargé/copié ce dossier sur votre disque dur.
Pour ouvrir ce projet dans LAZARUS, deux moyens :
1-Lancer LAZARUS. Utiliser le menu :Fichier/Ouvrir, puis sélectionner un fichier projet d'extension lpi ou lpr dans le dossier du projet.
Ou bien :
2-LAZARUS étant encore fermé, cliquer dans le poste de travail sur l'un de ces deux fichiers.

On peut par exemple imaginer que les élèves disposent lors de la séance d'un projet adapté à l'activité du jour, avec les composants déjà placés sur la fiche, et avec un code comportant des commentaires, où il suffira d'ajouter les instructions intéressantes ayant un lien direct avec l'acquisition avec SYSAM. Il leur suffira d'ouvrir ce projet dans LAZARUS, d'ajouter les quelques instructions adéquates, puis de compiler et lancer le programme.

Deuxième partie : le détail du jeu de commandes

Les constantes prédéfinies, objets et commandes qui suivent sont regroupées dans une unité, sous forme d'un fichier d'extension .PAS, appelé uSP5Laz.pas qui se trouve sur mon disque dur personnel. Vous disposez en fait de deux fichiers obtenus par la compilation de cette unité : uSP5Laz.o et uSP5Laz.ppu (ils se trouvent dans le dossier de l'exemple). Il est indispensable de mettre ces deux fichiers dans le dossier de tout projet en cours (donc il faut les placer dans le dossier de tout nouveau projet). Accéder au code de cette unité est impossible et inutile.

AVERTISSEMENT IMPORTANT : l'accessibilité à cette unité dans les programmes que l'on crée ci-dessous suppose que l'on utilise la version de LAZARUS avec laquelle cette unité a été compilée. A la date de mise à jour de cet article (9/12/2009), la version de LAZARUS utilisée est la v0.9.28.2 beta. Si vous utilisez une version antérieure de LAZARUS, il se peut que le compilateur vous avertisse qu'il ne trouve pas l'unité uSP5LAZ bien que vous ayez copié les fichiers afférents dans le dossier de votre projet. Ce n'est en fait qu'un conflit de versions. Si, après mise à jour de LAZARUS, le problème devait persister, contactez l'auteur : contact@logisciences.fr**.**

A. Constantes prédéfinies

Ces constantes ne sont pas modifiables : les valeurs qu'elles contiennent sont fixes. Les identificateurs de ces constantes sont en italique gras ; ils peuvent être utilisés tels quels dans le code.

Constantes numériques :
Numéros des sorties :
SA1 = 1
SA2 = 2

Index des calibres disponibles sur les entrées analogiques:
cal_10V = 0
cal_05V = 1
cal_02V = 2
cal_01V = 3
cal_002V = 4 (calibre 0,2 V)
cal_001V = 5 (calibre 0,1 V)

Mode des voies :
mvDiff = true;
mvNormal = false;

Entrées et canaux :
nbEntreesA = 8;
nbCanauxA = 4;

Codes d'erreur de la fonction acquisition(voie1, mesure1) :
eAucune = 0 ;
eVoieNonActive = 1;

Chaînes de caractères :
Noms des entrées analogiques de SYSAM SP5 :
nomsEA : array[0 .. nbEntreesA - 1] of string
= ('EA0', 'EA1', 'EA2', 'EA3', 'EA4', 'EA5', 'EA6', 'EA7');

Calibres des entrées :

sCalibresEA : array[0..5] of string
= ('-10/+10', '-5/+5', '-2/+2', '-1/+1', '-0,2/+0,2','-0,1/+0,1');
Ce tableau utile pour afficher des résultats.

Valeurs des calibres :

calibresEA : array[0..5] of double
= (10.0000, 5.0000, 2.000, 1.000, 0.2000, 0.1000);

Ces trois tableaux sont pratiques pour les boucles de code…

B. La classe TSP5 et ses méthodes

L'objet ou classe de type TSP5 comporte les procédures nécessaires pour l'utilisation de SYSAM.

1. Variables de TSP5 :

Tableau des valeurs renvoyées par l'acquisition, huit cases :

mesures: array[0..7] of word;
valeurs comprises entre 0 et 4095

Tableau mémorisant les voies activées, huit cases :

EAActives: array[0..nbEntreesA - 1] of boolean;
Mémorise les voies actives. **Valeurs** True ou **False**

Tableau des valeurs renvoyées par l'acquisition sur les ports logiques, huit cases :

vpB : array[0..7] of byte;
- Valeurs utilisées : 0 ou 1.
- Utilisé pour la lecture simultanée de 4 ou 8 bits ou l'écriture simultanée de 4 ou 8 bits du port logique de SYSAM.

2. Procédures et fonctions d'acquisition sur les entrées analogiques EA0 à EA7

procedure activeVoieA (numVoie : byte);
Active (sans mesure) l'entrée analogique numVoie pour que le moment venu elle puisse faire une mesure. Le paramètre numVoie doit être compris entre 0 et 7 (numVoie = 0 correspond à EA0) .

procedure desactiveVoieA (numVoie : byte);
Désactive l'entrée analogique numVoie pour que le moment venu, elle ne fasse plus de mesure (numVoie compris entre 0 et 7, 0 <-> EA0)

procedure CalibreVoieA (numVoie, calibre : byte);
Prépare le calibre de la voie numVoie.
Calibre est l'index du calibre. Utiliser les constantes prédéfinies cal_10V, cal_05V...

procedure ModeDiff (canal1 : integer; mode1 : boolean);
Place le canal1 en mode différentiel si mode1 = true et sur mode non différentiel si mode1 = false.

En mode différentiel, on utilise les deux entrées d'un même canal, par exemple, pour le canal 1 les entrées EA1-EA5. La masse de SYSAM n'a alors pas besoin d'être reliée au circuit.

procedure Acquisition;

Réalise une mesure simultanée sur toutes les voies actives.
Lorsque la procédure « acquisition » est appelée,

1. toutes les valeurs de ce tableau sont mises à 2047, ce qui correspond à une tension nulle, puis
2. les mesures sur les voies actives sont réalisées, puis
3. les valeurs entières obtenues sont placées dans le tableau, à la position correspondant au numéro de la voie sur laquelle la valeur a été acquise

function Acquisition(voie1: integer; var mesure1: word): integer;

Réalise une mesure sur la voie « voie1 ». Le résultat entier est mesure1. La valeur renvoyée par la fonction est un code d'erreur : il vaut 0 (zéro = eAucune) si tout s'est bien passé (voie1 activée...) et une valeur différente de zéro sinon.

function Ux(voie1 :Integer; vEntiere1 :integer):double;

Fonction de conversion permettant d'obtenir à partir de la valeur entière renvoyée par SYSAM la valeur de tension mesurée sur la voie « voie1 », en tenant compte de l'étalonnage de SYSAM, et avec limitation de la valeur obtenue au calibre de voie1

Fonction équivalente:
Ux := calibre * (2 * vEntiere1 / calibre - 1)

Cette fonction équivalente donne un résultat moins précis car elle n'utilise pas les données de calibrage de l'interface. Mais elle est sans doute plus pédagogique…

function ValEntiere(voie1:integer; U1: single):integer;

Fonction de conversion inverse de la fonction précédente : renvoie la valeur sur 12 bits (entre 0 et 4095) correspondant à la tension U1, en tenant compte des données de calibrage de l'interface.

3. Emission sur les sorties SA1 et SA2

function UxSA(SA :Integer; vNum:integer):double;

Fonction de conversion de la valeur numérique envoyée sur la sortie SA, en valeur de tension, tenant compte du calibrage de SYSAM

function ValUxSA(SA:integer; U1: single):integer;

Fonction inverse de la précédente : renvoie la valeur numérique correspondant à la tension à émettre.

procedure EmissionValContinue(SAi: integer; vNum: integer);

Emission d'une valeur entière vNum sur la sortie SAi. La valeur vNum correspond affinement à la tension souhaitée en sortie de SYSAM ; elle doit être comprise entre 0 (correspondant à –10,0 V) et 4095 (correspondant à + 10,0 V).

procedure EmissionUContinue(SAi: integer; U1: single);

Après l'appel de cette procédure, la sortie SAi fournit une tension continue de valeur U1 (entre –10,0 V et +10,0 V). Cette fonction tient compte du calibrage de SYSAM.

procedure ArretEmission(SAi : integer);

Après l'appel de cette procédure, la sortie SAi fournit une tension continue nulle.

4. Gestion du port logique B (boîtier BOLOGIC)

Les 8 bits du port logique B peuvent fonctionner en entrée ou en sortie. Cependant, le port logique est conçu pour fonctionner par groupe de 4 bits : les entrées/sorties 0 à 3 fonctionnent dans le même sens (entrée ou sortie), de même que les entrées/sorties 4 à 7. Si par exemple on émet une valeur sur B2 (0 ou 1), tout le groupe B0-B3 est mis en mode émission.

procedure EmetPLB0_3;

Met en une opération les quatre premiers bits (0 à 3) du port logique (BOLOGIC) aux valeurs binaires du tableau plb (plb[0] à plb[3]). Les LED de BOLOGIC correspondant aux bits 0 à 3 de valeur 1 doivent s'allumer.

procedure EmetPLB4_7;

Met en une opération les quatre derniers bits(4 à 7) du port logique B (BOLOGIC) aux valeurs binaires du tableau plb (plb[4] à plb[7]). Les LED de BOLOGIC correspondant aux bits 4 à 7 de valeur 1 doivent s'allumer.

procedure litPLB0_3;

Lit en une opération les valeurs sur le port B et les place dans plb (de plb[0] à plb[3]).

procedure litPLB4_7;

Lit en une opération les valeurs sur le port B et les place dans plb (de plb[4] à plb[7])

procedure emetPLB(numBit, val1 : byte);

Emet sur le port numBit la valeur val1 (0 ou 1). La LED du numBit bit correspondant doit s'allumer si val1 = 1. Les quatre bits du groupe correspondant à numBit sont mis en mode émission.

function litPLB(numBit: byte): byte;

Lit sur le port numBit et renvoie la valeur lue (0 ou 1).
Les quatre bits du groupe correspondant à numBit sont mis en mode lecture.

Troisième partie : créer un programme.

Rappels :

A. On compose la fenêtre (fiche) du programme en y plaçant des composants que l'on prend dans la palette des composants. L'inspecteur d'objet permet de fixer les propriétés initiales de ces composants (couleurs, taille, position sur la fenêtre, texte qu'il contient...)

B. On fait de la "programmation orientée objet et événementielle": le programme lancé réagit à des événements (clics de souris sur des boutons, appuis de touches...) en exécutant les procédures correspondantes appelées gestionnaires d'événements. Tout l'art consiste à mettre dans ces gestionnaires un code bien adapté, ce qui reste assez facile pour des petits programmes.

Commençons :

- Remettez LAZARUS dans son « état initial », avec un nouveau projet vierge (menu : Fichier/Nouveau, puis sélectionnez Project/Application, cliquez sur OK).

Après le mot réservé « type » apparaît une déclaration de type Tform1, qui est le type de la fiche-fenêtre que nous créons en ajoutant des composants.

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  uSP5Laz, Classes, SysUtils, LResources,
  Forms, Controls, Graphics, Dialogs;

type
  TForm1 = class(TForm)
  private
    { private declarations }
  public
    { public declarations }
    interfaceSP5 : TSP5;
  end;

var
  Form1: TForm1;

implementation

initialization
  {$I unit1.lrs}

end.
```

1. Ajout de l'unité uSP5LAZ.

- Dans l'éditeur de code (touche F12), positionnez le curseur après le mot réservé « uses », après la série d'unités standard déclarées, et ajoutez uSP5Laz.
- **Attention : un bug dans LAZARUS oblige à placer la déclaration de cette unité en première position.**

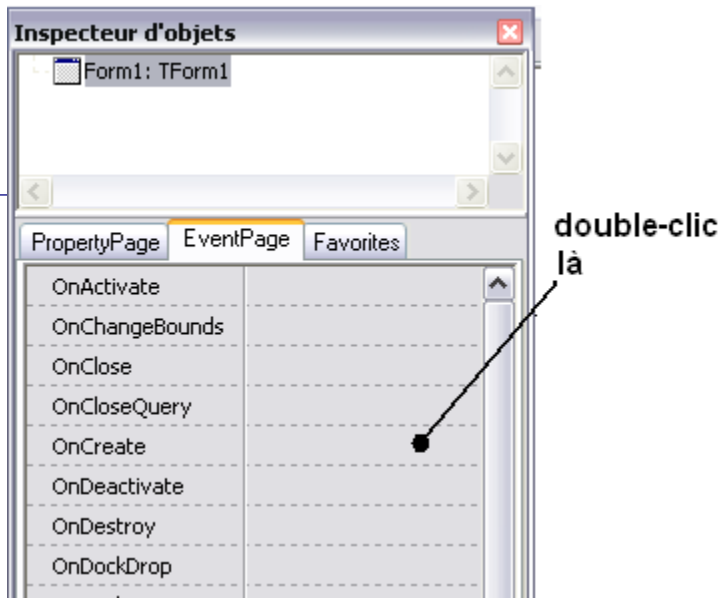
2. Déclaration d'un objet de type TSP5, nommé « interfaceSP5 » :

- ajoutez dans la partie des déclarations publiques { public declaration } de l'objet Tform1, la déclaration : « interfaceSP5 : TSP5 ; »

(InterfaceSP5 est l'identificateur de l'objet dans tout le code que nous créerons ci-dessous)

3. La création de l'objet InterfaceSP5.

A ce stade, si nous compilons et lançons le programme (touche F9), aucune mémoire vive ne sera réservée pour le stockage des propriétés de l'objet interfaceSP5, et aucune de ses procédures ne sera disponible.



Il faut donc qu'au démarrage du programme, avant même que n'apparaisse quoi que ce soit sur l'écran, cet objet soit « créé ». Pour tout programme, il existe un événement particulier qui a lieu à ce moment-là, à la faveur duquel nous pouvons réaliser cette création, grâce à son gestionnaire.

•Allez dans l'inspecteur d'objet (IO, touche F11)

Comme il n'y a pas encore de composants sur la fiche, c'est elle-même qui sera automatiquement traitée par l'inspecteur d'objet.

•Dans l'IO, sélectionnez l'onglet « EventPage »

•Double-cliquez à droite de la rubrique « OnCreate », qui correspond à l'événement de création de la fiche.

Instantanément, LAZARUS rajoute dans le code la procédure FormCreate, dans la partie des déclarations de Tform1, puis, dans la partie implémentation, ajoute l'enveloppe du gestionnaire de l'évenement correspondant. Précisons qu'il ne faut jamais ajouter manuellement du code avant le mot réservé « private » : cette partie de Tform1 est gérée automatiquement par LAZARUS, et sert à la déclaration des composants ajoutés à l'aide de la palette de composants. Vous la verrez s'étoffer au fur et à mesure de la construction de notre programme.

```
implementation

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);
begin
     interfaceSP5 := TSP5.Create;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
     interfaceSP5.free;
end;

initialization
  {$I unit1.lrs}

end.
```

•Entre les mots « begin » et « end » de la procédure Tform1.FormCreate, ajoutez l'instruction :

interfaceSP5 := TSP5.create;

Ainsi une règle de base de la programmation orientée objet est respectée: un objet doit être créé avant d'être utilisable.

4. La destruction de l'objet interfaceSP5.

Lorsque l'utilisateur quitte le programme, il est nécessaire de libérer la mémoire vive occupée par l'objet interfaceSP5. On utilise pour cela l'événement ayant lieu à la destruction de la fiche, c'est-à-dire juste avant que le programme libère la mémoire occupée par la fenêtre-fiche, et avant que le programme ne s'arrête.

•Dans l'onglet « EventPage » de l'inspecteur d'objets, double-cliquez à droite de la rubrique onDestroy.

De la même façon que précédemment, LAZARUS crée l'enveloppe du gestionnaire de l'événement.

•Placez dans le bloc l'instruction « interfaceSP5.free ; ».

Cette opération est importante car si elle n'est pas réalisée, le système risque de perdre un peu de mémoire vive à chaque arrêt du programme.

5. Première acquisition.

Le déclenchement de l'acquisition peut être provoqué suite à l'événement de clic sur un bouton de la fenêtre du programme (action la plus courante des utilisateurs de logiciels…). Nous allons donc ajouter sur notre fiche-fenêtre un bouton et utiliser son gestionnaire d'événement de clic pour provoquer une acquisition.

•Dans la fenêtre principale de LAZARUS, allez dans la palette de composants, onglet « Standard ». Cliquez sur le bouton du composant Tbutton :

•Cliquez ensuite sur la fiche chantier, là où vous voulez déposer ce composant. En principe, à ce stade, le composant est géré dans l'inspecteur d'objets.

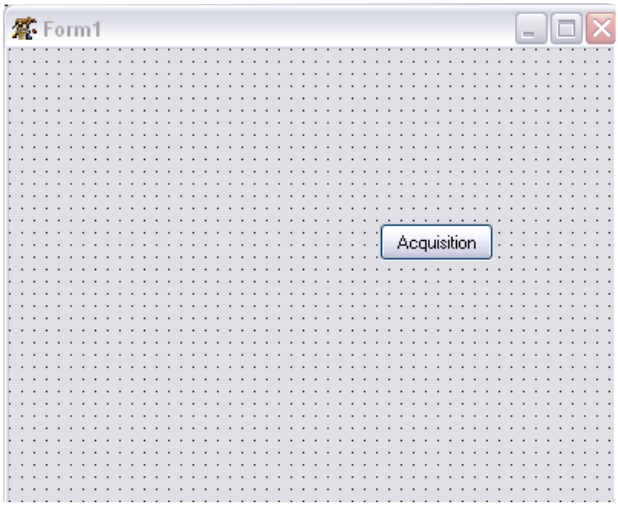
•Renommez le bouton : recherchez dans l'onglet PropertyPage la rubrique « Name », et changer « Button1» en « Bacquisition » (cette propriété doit respecter la syntaxe du pascal, à savoir

-le premier caractère est une lettre (pas un chiffre)

-ni espace ni caractères accentués ou spéciaux. Le caractère _ peut être utilisé.

•Texte du bouton : recherchez la propriété « Caption » et changez-la en « Acquisition ».

•Allez dans l'onglet EventPage de l'inspecteur (F11) et double-cliquez à droite de la rubrique « OnClick », ce qui fait apparaître dans le code (F12) l'enveloppe de la procédure-gestionnaire de l'événement de clic sur le bouton.



•Dans l'entête de la procédure (avant le mot réservé « begin ») placez les déclarations de variables :

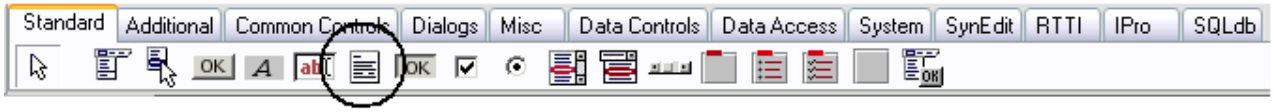
```
Var   mesure1 : word ;
      voie : byte ;
```

•Ajoutez entre le « begin » et le « end » les instructions permettant de réaliser une mesure unique sur l'entrée analogique EA0 avec le calibre 5V (colonne de gauche du tableau suivant).

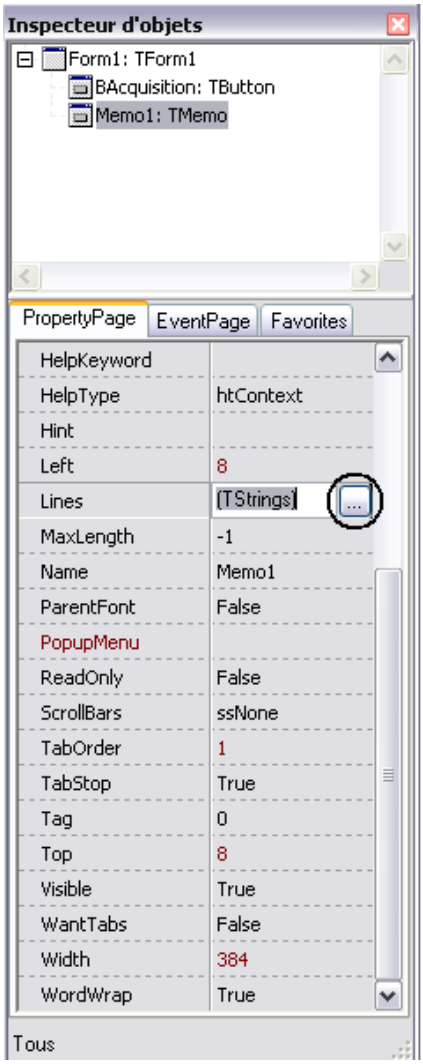
Instructions	Actions correspondantes
Voie := 0 ;	Choix de la voie EA0
interfaceSP5. calibreVoieA (voie, cal_05V);	Réglage du calibre de l'entrée EA0 sur +/-5V (notez l'usage de la constante prédéfinie cal_05V)
interfaceSP5. activeVoieA (voie);	Activation de l'entrée EA0 (la seule qui fera une mesure pour le moment).
interfaceSP5. acquisition (voie, mesure1) ;	Réalisation de l'acquisition, dont le résultat est récupéré dans la variable mesure1

A ce stade de l'exécution, (i.e. lorsque l'utilisateur a cliqué sur le bouton acquisition), la valeur entière mesurée est « stockée » dans la variable entière mesure1. Il faut maintenant ajouter les composants et les instructions qui permettent d'afficher cette valeur à l'écran. Un composant adapté à cet usage (parmi d'autres) est le memo.

•Dans la fenêtre principale de LAZARUS, allez dans la palette de composants, onglet « Standard ». Cliquez sur le bouton du composant Tmemo.



•Cliquez ensuite sur la fiche chantier, là où vous voulez déposer ce composant. Agrandissez ou réduisez la taille du memo en saisissant un de ses coins avec la souris, de manière à ce qu'il ne recouvre pas le bouton. Déplacer éventuellement le bouton. A l'aide de l'inspecteur d'objets, vous pouvez également changer la police de caractères… Notez que le nom du memo est le nom par défaut « memo1 » ; c'est ce nom qui sera utilisé dans le code ci-dessous.



•De plus, dans la rubrique « lines », cliquez sur les (Tstrings) puis sur les trois points du petit bouton, et dans l'éditeur de texte qui apparaît, supprimez le texte (« memo1 »).

•Dans l'éditeur de code, retournez à la suite de la déclaration de la variable « mesure1 » et ajoutez :

s : string ; (chaîne de caractères qui servira à la mise en forme de l'affichage du résultat)

U : single ; (valeur flottante destinée à stocker la valeur physique de la tension mesurée).

•A la suite des quatre instructions précédentes, ajoutez, en respectant strictement les caractères (en particulier la double apostrophe) les instructions de la colonne de gauche du tableau, dans l'ordre:

Instructions	Actions
memo1.lines.add('Mesure sur l''entrée '+ nomsEA[voie]);	Titre de l'affichage des résultats dans le memo1 (notez là encore l'usage du tableau prédéfini nomsEA[]).
s := intToStr(mesure1);	Conversion de la valeur entière en chaîne de caractères.
s := ' valeur entière: v = ' + s;	Ajoute d'un titre.
memo1.lines.add(s);	Ajout de cette chaîne dans le memo1.
u := InterfaceSP5. ux(voie, mesure1);	Récupération de la tension correspondant à la valeur entière, en tenant compte du calibrage de l'interface (seule instruction non standard Pascal/Lazarus).
S := floatToStrF(u, ffgeneral, 4 , 2);	Conversion de la valeur de la tension en chaîne de caractères (4 chiffres significatifs)
s := ' tension : u = ' + s + ' V';	Ajoute d'un titre.
memo1.lines.add(s);	Ajout de la chaîne dans le memo1.

Ci-dessous, l'éditeur de code avec ces instructions.

```
implementation

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);
begin
     interfaceSP5 := TSP5.Create;
end;

procedure TForm1.BAcquisitionClick(Sender: TObject);
var mesure1 : word ;
    voie : byte ;
    s : string ;
    U : single;

begin
  voie := 0;
  interfaceSP5. calibreVoieA ( voie, cal_05V );
  interfaceSP5. activeVoieA ( voie );
  interfaceSP5. acquisition ( voie, mesure1) ;
  memo1.lines.add('Mesure sur l''entrée '+ nomsEA[voie]);
  s := intToStr(mesure1);
  s := ' valeur entière:  v = ' + s;
  memo1.lines.add(s);
  u := InterfaceSP5. ux(voie, mesure1);
  S := floatToStrF(u, ffgeneral, 4 , 2);
  s := ' tension :        u = ' + s + ' V';
  memo1.lines.add(s);

end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
     interfaceSP5.free;
end;
```

•Le premier essai :

Reliez un générateur continu à l'entrée EA0, ainsi qu'un voltmètre, réglez sur une tension comprise entre –5V et +5 V, puis compilez et lancez le programme (F9). Cliquez sur le bouton acquisition. Votre première mesure est réalisée et s'affiche dans le mémo ; comparez avec l'indication du voltmètre.

6. Une autre acquisition

L'acquisition ne remet pas à zéro les paramètres. Si par exemple, on a activé une voie pour une acquisition, et qu'on ne souhaite pas l'utiliser lors de l'acquisition suivante, il faut la désactiver avant de lancer l'acquisition suivante.

Dans ce second exemple, nous allons réaliser une acquisition sur les huit entrées analogiques en même temps, en utilisant le calibre +/-10 V sur les quatre premières et +/-2 V sur les quatre dernières .

•Cette seconde acquisition sera déclenchée par un autre bouton que vous placerez à votre convenance sur la fiche-chantier. Renommez ce bouton (Bacquisition2), cliquez dans l'inspecteur d'objets à droite de la rubrique onClick de l'onglet « EventPage ». Vous obtenez dans le code l'enveloppe du gestionnaire de l'événement de clic sur ce bouton, du style:

procedure Tform1.Bacquisition2OnClick(sender : Tobject).

•En en-tête, ajoutez les déclarations de variables :

```
var   voie : integer;
      s, s1: string;
      U : single;
```

•Entre le « begin » et le « end » de ce gestionnaire, ajoutez ce qui suit (les commentaires sont entre accolades et les instructions en gras (utilisez le copier/coller):

{Ecriture d'un titre :}

memo1.lines.add('Acquisition d''une valeur sur les 8 entrées, avec le calibre +/-10 V sur ' + 'les quatre premières et +/-2 V sur les quatre dernières');

{Calibre +/- 10V sur les 4 premières voies :}
for voie := 0 to 3 do interfaceSP5. calibreVoieA(voie, cal_10V);

{calibre +/- 2V sur les 4 dernières voies :}
for voie := 4 to 7 do interfaceSP5. calibreVoieA(voie, cal_02V);

{activation de toutes les voies :}
for voie := 0 to 7 do interfaceSP5. activeVoieA(voie);

{réalisation de l'acquisition :}
interfaceSP5. acquisition;
{lors de l'exécution du programme et après clic sur le bouton, à ce stade, le tableau "mesures" contient une mesure entière dans chaque case}

{affichage des résultats dans le memo1 : }
for voie := 0 to 7 do {on répète pour les 8 entrées les instructions du bloc suivant}
begin {début du bloc}

{construction d'une chaîne de caractères :}
s := 'u(' + nomsEA[voie] + ') = ';

{récupération de la tension :}
u := interfaceSP5. ux(voie, interfaceSP5. mesures[voie]);

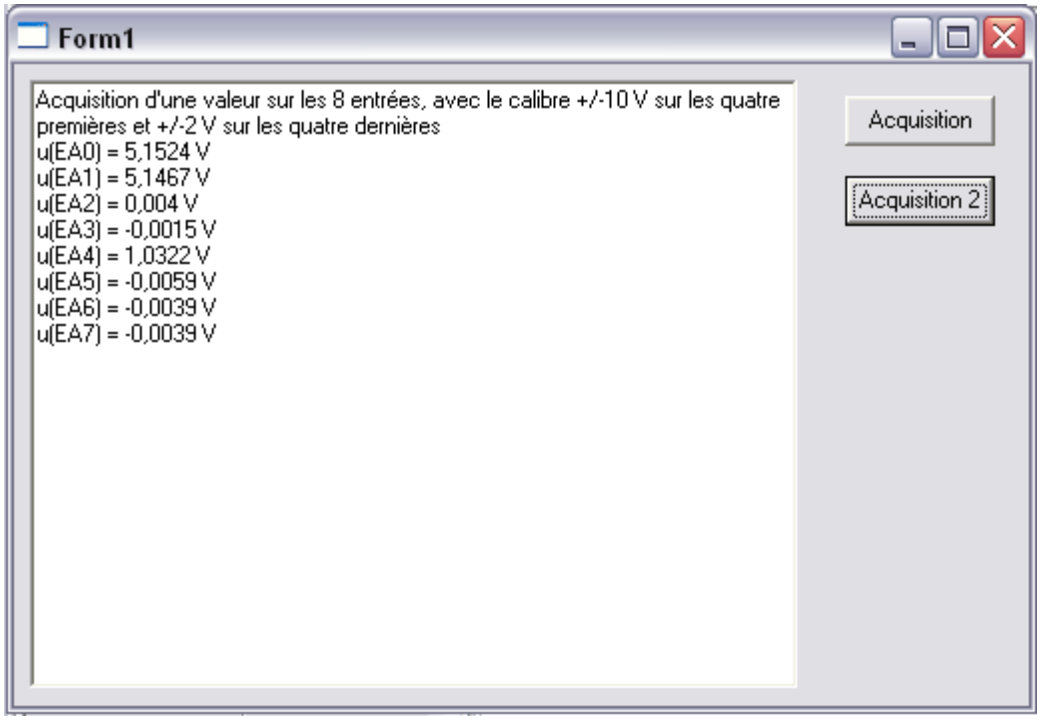
{conversion de la valeurs de la tension en chaîne de caractères :}
s1 := floatToStrF(u , ffgeneral, 4, 2);
s := s + s1 + ' V';

{ajout de la chaîne dans le memo1 :}
memo1.lines.add(s);
end; { fin du bloc }

Voici le code obtenu (sans les commentaires):

```
procedure TForm1.BAcquisition2Click(Sender: TObject);
var voie: integer;
    s,s1: string;
    U : single;
begin
   memo1.lines.add('Acquisition d''une valeur sur les 8 entrées, avec le calibre +/-10 V sur ' +
                   'les quatre premières et +/-2 V sur les quatre dernières');

   for voie := 0 to 3 do interfaceSP5. calibreVoieA(voie,cal_10V);

   for voie := 4 to 7 do interfaceSP5. calibreVoieA(voie,cal_02V);

   for voie := 0 to 7 do interfaceSP5. activeVoieA(voie);

   interfaceSP5. acquisition;

    for voie := 0 to 7 do
         begin
              s := 'u(' + nomsEA[voie] + ') = ';

              u := interfaceSP5.ux(voie, interfaceSP5.mesures[voie]);

              s1 := floatToStrF(u , ffgeneral, 4, 2);

              s := s + s1 + ' V';

              memo1.lines.add(s);
         end;


end;
```

Ainsi qu'une capture d'écran après un clic sur Acquisition2 et trois tensions non nulles en entrée :



7. Une émission.

Ajoutez un nouveau bouton sur la fiche, changez sa propriété name en « BEmission », créez l'enveloppe de son gestionnaire d'événement OnClick, puis placez dans ce gestionnaire la seule instruction :

```
interfaceSP5.EmissionUContinue( SA1 , 2.00 );
```

Lancez le programme, cliquez sur le bouton ; la sortie SA1 de l'interface doit alors fournir une tension de 2.0 V. Vérifiez avec un voltmètre…
Cependant cette procédure n'a peut-être pas une portée pédagogique suffisante. On peut donc la remplacer par les instructions suivantes (en gras), après avoir déclaré V1 comme single et vNum1 comme word dans l'entête du gestionnaire :

var V1 : single ; vNum1 : word;

```
begin
        {Choix de la valeur de la tension continue à émettre}
        V1 := 2.00 ;

        {Calcul de la valeur entière correspondante }
        Vnum1 := round( 2047 * (1 + V1 / 10) );

        {Emission de la valeur :}
        interfaceSP5. EmissionValContinue( SA1 , vNum1 );
end ;
```

Il ne reste plus qu'à tester…

8. Le port logique B et le boîtier BOLOGIC

Le boîtier BOLOGIC permet d'accéder individuellement aux huit bits du port logique B.

a. Emission sur les sorties logiques.

Ajoutez un bouton, renommez-le, choisissez un « caption » significatif, créez l'enveloppe de son gestionnaire d'événement de clic.

Voici le code du gestionnaire du bouton déclenchant l'émission de la valeur 1 sur les ports 0 et 3.

```
begin
        memo1.lines.add(' ');
        memo1.lines.add('Mise à 1 des ports logiques B0 et B3');
        interfaceSP5.  emetPLB(0,1);
        interfaceSP5.  emetPLB(3,1);
        memo1.Lines.add('Vérifiez que les LED correspondantes sont allumées');
end ;
```

b. Lecture de l'état des ports logiques B4 et B7 :

Ajoutez un bouton, renommez-le, choisissez un « caption » significatif, créez l'enveloppe de son gestionnaire d'événement de clic, déclarez les variables suivantes :

```
var
s1: string;
ValBin: integer;
```

Puis insérez le code suivant:

```
begin
        memo1.lines.add(' ');
        memo1.lines.add('Lecture de l''état des ports logiques B4 et B7');
        valBin := interfaceSP5. litPLB(4);
        s1 := '   B4 : ' + intToStr(valBin);
        memo1.lines.add(s1);
        valBin := interfaceSP5. litPLB(7);
        s1 := '   B7 : ' + intToStr(valBin);
        memo1.lines.add(s1);
end;
```

Imposez avec un générateur ou un fil un état logique à B4 et B7, puis testez.

9. Pour aller plus loin…

a. Autres propriétés des composants

Pour améliorer l'aspect de votre programme, vous pouvez, à l'aide de l'inspecteur d'objets, changer les couleurs des composants, la police de caractères utilisée, la taille de la fiche et des composants. Peut-être également remplacer le caption (libellé) par défaut de la fiche-chantier (« Form1 ») par quelque chose de plus parlant.
Il suffit de cliquer sur le composant dans la fiche-chantier pour le faire « entrer » dans l'inspecteur d'objet, puis d'éditer les propriétés.

b. Acquisitions répétées

Si on souhaite faire des acquisitions répétées régulièrement à un rythme pas trop rapide (fréquence inférieure au demi-Hertz), on utilise un composant timer et son événement onTimer (voir palette de composants « system »).

On place simplement l'ensemble du code d'acquisition ou d'émission entre le « begin » et le « end » du gestionnaire de cet événement. La propriété « interval » correspond au temps séparant deux événements OnTimer1 successifs. Il faut alors prévoir un bouton pour démarrer les acquisitions et un autre pour les arrêter, dont les gestionnaires d'événement de clic mettent la propriété « enabled » du timer1 respectivement sur true pour démarrer les acquisitions, et sur false pour les arrêter:

timer1.enabled :=true ; timer1.enabled :=false ;

Pour les acquisitions plus rapides commes celles de LATIS PRO ou Mesures électriques ou OSCILLO5, c'est beaucoup plus compliqué : cela nécessite de solides compétences en programmation ainsi qu'une étude détaillée de la documentation d'EUROSMART.

c. Ajout de composants de réglage de la tension de sortie en émission.

Ou : comment fabriquer un générateur continu réglable ?

Utilisez un composant TtrackBar (palette « Common controls », nom par défaut TrackBar1) et son événement onChange, qui a lieu à chaque déplacement de son curseur par l'utilisateur.

Mettez sa propriété max à 100 et sa propriété min à –100 ;

Utilisez la correspondance

position = 100 <---> vNum = 4095
position = -100 <---> vNum = 0

ce qui se traduit par l'équation :
vNum := 2047 * (1+ position/100) ;
puis l'instruction d'émission de la valeur numérique (voir §7).

Ajoutez également un composant d'édition de type Tedit (nom par défaut Edit1) , pour l'affichage de la tension obtenue par le réglage du curseur .

Ce qui donne le code (avec une déclaration de variables) suivant pour le gestionnaire :

```
var vNum : integer;
   U1   : single;

begin
{ calcul de la valeur numérique à émettre en fonction de la position du curseur}
  vNum := round(2047 * (1 + TrackBar1.position/100)) ;
{ calcul de la tension émise }
  U1      := interfaceSP5.UxSA(SA1, vNum);
{ affichage de la tension émise }
  edit1.text := FloatToStrF(U1, ffGeneral, 3 , 2);
{ actualisation de l'affichage }
  edit1.update;
{ émission }
  interfaceSP5. EmissionValContinue( SA1 , vNum );
end;
```

Conclusion

L'auteur de cet article se tient à la disposition du lecteur pour toute question concernant l'utilisation des instructions présentées dans cet article, et même éventuellement pour l'ajout d'instructions standard en pascal.

Pour les passionnés, les éléments de cet article peuvent suffire pour créer de petites applications d'EXAO où la rapidité d'acquisition ne joue pas un rôle primordial. On peut même imaginer d'inclure dans une application simple le dessin progressif d'un graphe…

Il n'y a aucune instruction d'utilisation de capteurs EUROSMART dans cet article, car le lycée où travaille l'auteur n'est pas encore véritablement équipé en capteurs de cette marque. Mais rien n'empêche d'utiliser des capteurs classiques à sortie analogique…

Peut-on imaginer que les ressources développées dans cet article trouvent une utilité dans d'autres filières des lycées (STI,…) ?

Pour les questions et les difficultés rencontrées dans la mise en pratique, des suggestions de développements nouveaux, une seule adresse : contact@logisciences.fr , site de l'auteur.

Pour finir, l'auteur tient à signaler que cet article n'aurait pas été rédigé sans la demande des collègues du lycée A. Heinrich de Haguenau (67), enseignant l'option MPI en seconde, Madame Nathalie Kuntz et Monsieur Pascal Peter.

Dernière mise à jour : 9 décembre 2009

Jean-Marie THOMAS
Professeur de Sciences Physiques
Lycée Jean-Monnet - Strasbourg.