

Programmation de SYSAM sous LAZARUS avec la bibliothèque DLL

ProgrammationSYSAM_SP5.dll

Introduction







Cet article a pour objectifs

- de présenter sommairement la mise en œuvre de LAZARUS ;
- d'expliquer en détail chacune des commandes de la DLL « ProgrammationSYSAM_SP5.dll » ;
- de décrire la réalisation d'un programme exemple simple utilisant ces 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, 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 de cet article sont tributaires de la version de LAZARUS utilisée. 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 :

Nom	Date de modificati...	Type	Taille
 ProgrammationSYSAM_SP5.dll	23/08/2010 16:04	Extension de l'app...	417 Ko
 project1.lrs	24/08/2010 12:01	Fichier LRS	413 Ko
 unit1.lfm	23/08/2010 18:00	Lazarus form	2 Ko
 project1.lpi	24/08/2010 12:01	Lazarus project inf...	7 Ko
 project1.lpr	23/08/2010 09:51	Lazarus project inf...	1 Ko
 unit1.pas	23/08/2010 18:00	Object Pascal Unit	37 Ko

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

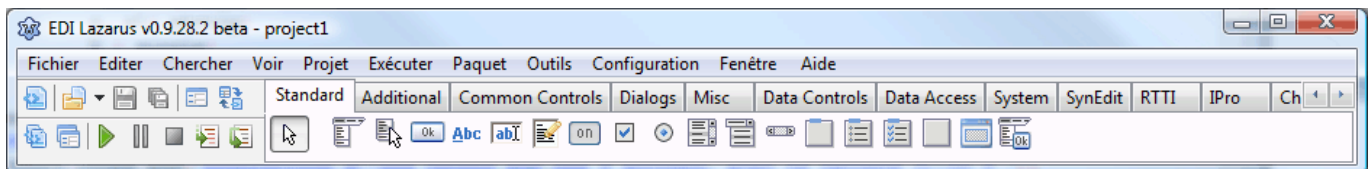
Sélectionnez la version de LAZARUS (Lazarus Windows) et téléchargez cette version. Réalisez la procédure d'installation classique (Attention : le dossier dans lequel se fera l'installation ne peut pas être dans le classique « Program files » car dans ce nom de dossier, il y a un espace... Acceptez simplement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

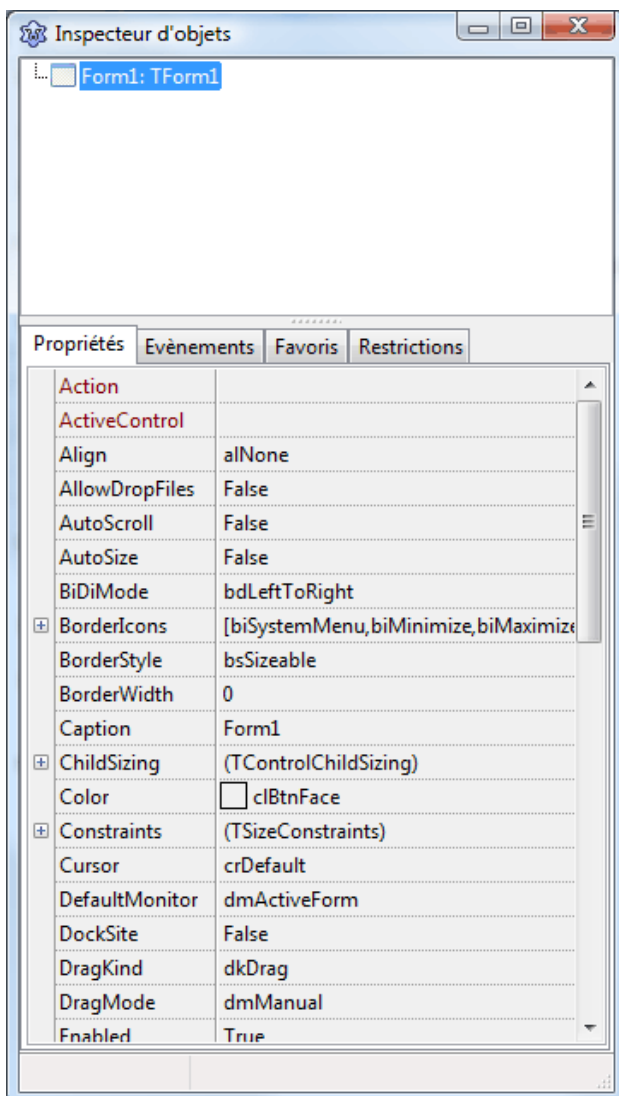
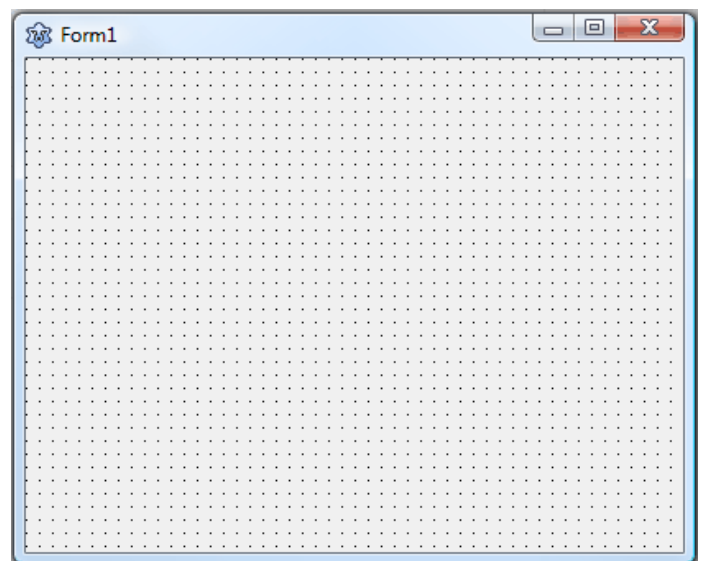
a. Le projet LAZARUS et l'EDI (environnement de développement intégré)

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.



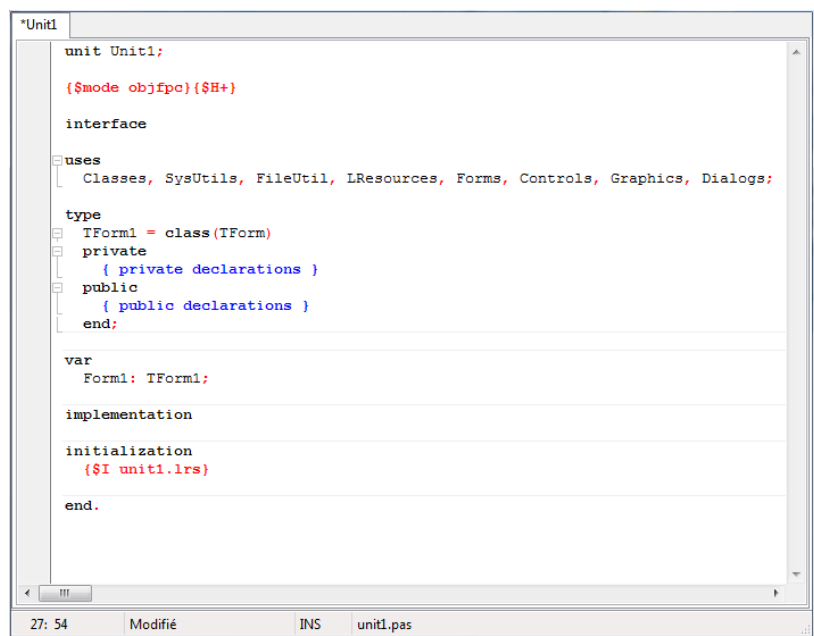
La fiche chantier, qui est une image de ce que sera la fenêtre du programme, sur laquelle on place des composants « prélevés » dans la palette des composants de la fiche principale

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.



Les trois fichiers cités ne diffèrent que par leur extension : la racine du nom est choisie lors du premier enregistrement du projet. 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.

Le 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, sans compter d'autres fichiers annexes ajoutés par Lazarus.

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'événements

→ 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 :

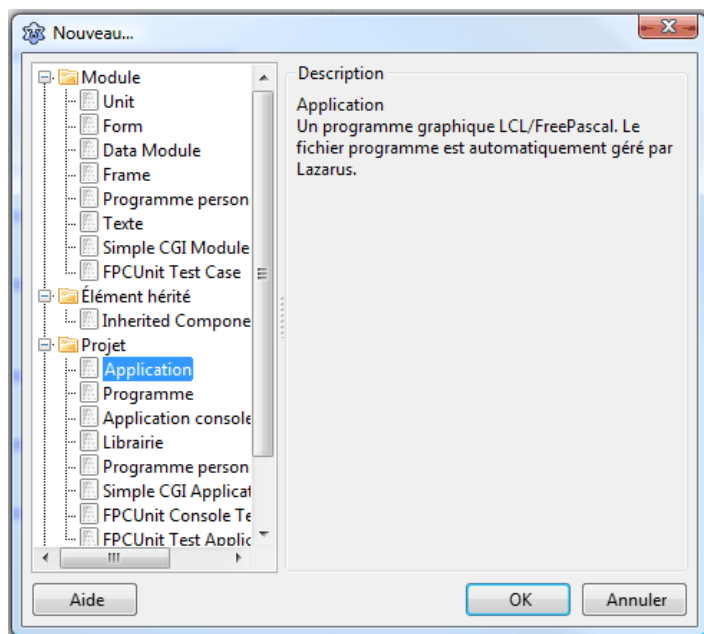
- démarrage du programme
- fin du programme, pour ne citer que les plus simples.

La quatrième 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 : que peut-on faire faire à SYSAM SP5 avec ProgrammationSYSAM_SP5.dll ?

Nous appellerons « programme utilisateur » le programme utilisant les fonctions et procédures de la bibliothèque ProgrammationSYSAM_SP5.dll pour le contrôle de SYSAM : il s'agit du programme que nous allons créer.

A. Exposé sommaire de la conversion analogique – numérique

Les entrées de SYSAM comportent des convertisseurs analogiques numériques permettant de convertir une tension mesurée (soit directement sur une entrée de SYSAM connectée à un circuit, soit à la sortie d'un capteur) en une valeur numérique entière (abrégée en VNE dans la suite de ce texte) comprise entre 0 et 4095. **La transmission des mesures de SYSAM SP5 vers l'ordinateur et le programme utilisateur se fait exclusivement par des VNE.**

La correspondance entre la valeur numérique et la tension est affine :

$$U = a \cdot VNE + b.$$

Les logiciels pilotant SYSAM doivent faire la conversion des valeurs reçues lors de la transmission pour disposer des valeurs de tensions mesurées lors des acquisitions.

Les coefficients a et b dépendent du calibre et de l'entrée utilisés et des réglages précis de SYSAM SP5 en usine. Par exemple, si le calibre est +/-1V sur une entrée donnée, on a :

$$\begin{aligned} -1 \text{ V} &<-> VNE = 0 \\ +1 \text{ V} &<-> VNE = 4095 \\ U &<-> VNE1 \end{aligned}$$

On en tire que

$$U = 2 / 4095 \times VNE1 - 1$$

et donc

$$a = 2/4095 = 4,88 \cdot 10^{-4} \quad \text{et} \quad b = -1$$

En réalité, les valeurs de a et de b sont légèrement différentes, et la ProgrammationSYSAM_SP5.dll utilise les valeurs récupérées dans les données de calibrage de SYSAM (mémorisées à part sur SYSAM au moment du contrôle en usine).

En ce qui concerne les sorties de SYSAM (SA1 et SA2), le processus est symétrique : le programme envoie à SYSAM des VNE judicieusement calculées, qui sont stockées dans sa RAM ; puis SYSAM les convertit en tensions qu'elle fournit à ses entrées, au rythme imposé par le paramétrage temporel des sorties.

B. Les actions possibles

Les instructions de ProgrammationSYSAM_SP5.dll permettent de faire faire à la centrale SYSAM SP5 les actions suivantes.

1. Mesure unique sur une de ses entrées analogique. Nous dirons qu'il s'agit d'une **mesure ponctuelle**.

2. Mesure unique presque simultanée sur l'ensemble de ses entrées analogiques actives. Nous dirons qu'il s'agit d'une **acquisition ponctuelle**. De manière générale, nous appellerons « **point** » l'ensemble des valeurs obtenues par une mesure simultanée sur les entrées actives (une valeur par entrée active).

3. Réalisation d'un nombre fini de mesures répétées à intervalle de temps régulier sur l'ensemble des voies actives. Il s'agira d'une **acquisition répétée monocoup**. On obtiendra donc un ensemble limité de points (qui peut être important). L'acquisition cesse dès que le nombre de points donné aura été acquis.

4. Réalisation d'un nombre infini de mesures répétées à intervalle de temps régulier sur l'ensemble des voies actives. Il s'agira d'une **acquisition répétée permanente**. L'acquisition ne cesse que par une action de l'utilisateur du programme pilote.

5. Envoi d'un signal sur chacune des sorties : il s'agit d'une **émission**. On transmet à SYSAM une succession de VNE, qui deviennent par l'intermédiaire du Convertisseur Numérique Analogique (CNA) un signal de tension.

6. Détection des capteurs branchés sur SYSAM, et obtention de leurs numéros d'identifications.

7. Lecture ou paramétrage des états logiques des ports du boîtier BOLOGIC.

Pour chacune des actions, on peut effectuer des opérations spécifiques et préliminaires de paramétrage : par exemple pour les acquisitions, activer des entrées au choix, régler un calibre, choisir un type de déclenchement, une tension de seuil, un nombre de points à acquérir ainsi que la durée d'acquisition...

Les instructions de ProgrammationSYSAM_SP5.dll permettent de réaliser ces opérations de manière beaucoup plus simple qu'en utilisant les techniques standard données dans la documentation d'Eurosmart, et rendent la programmation de SYSAM accessible aux lycéens et aux étudiants (et aux professeurs !) même novices en programmation.

Pour finir, on notera que les instructions de ProgrammationSYSAM_SP5.dll ne permettent pas d'exploiter de manière exhaustive les possibilités de SYSAM SP5. ProgrammationSYSAM_SP5.dll a été construite avec l'expérience acquise par la programmation d'OSCILLO5, sans pour autant inclure toute cette expérience. Si un lecteur, après avoir exploité entièrement « ProgrammationSYSAM_SP5.dll » souhaite pousser plus avant la programmation de SYSAM, il lui faut contacter l'auteur : contact@logisciences.fr ou bien repartir de la documentation d'Eurosmart.

Troisième partie : le détail du jeu de commandes

A. La bibliothèque « ProgrammationSYSAM_SP5.dll ».

Les commandes ou instructions nécessaires au pilotage de SYSAM SP5 sont disponibles dans la bibliothèque « ProgrammationSYSAM_SP5.dll ». Il vous faut télécharger cette DLL et la placer (lien ici)

- soit dans le dossier de tout programme qui va l'utiliser, en l'occurrence ici dans le dossier du projet,
- soit dans le dossier C:\Windows\System32

B. Accès aux fonctions.

Pour accéder aux fonctions et procédures, il est nécessaire de déclarer dans le code les fonctions utilisées de la DLL avec le mot réservé « external ». L'emplacement précis sera indiqué dans la troisième partie de ce document. Le fichier « Declarations_Pascal.txt » fournit les déclarations qu'il vous suffira simplement de copier dans l'éditeur de code du projet LAZARUS.

C. Les structures à déclarer.

Pour une bonne exploitation de certaines fonctions, il faut disposer de structure que l'on devra passer comme paramètres ou variables à ces fonctions : il s'agit des tableaux de résultats des mesures effectuées par SYSAM. Ces structures sont à déclarer juste avant les procédures et fonctions de la DLL dans le programme utilisateur.

Type TTabMesures = Array[0..7] of word;

Type du tableau destiné à recueillir les valeurs des mesures effectuées par l'appel à la procédure SP5_Acquisition (acquisition ponctuelle sur les voies actives) 8 valeurs numérotées de 0 à 7, constituées d'entier positifs compris entre 0 et 65535 (16 bits). Dans la pratique, les valeurs sont comprises entre 0 et 4095 (12 bits seulement, limite de la résolution de SYSAM).

Type TTamponSP5 = array[0..256 - 1] of word;

Type du tableau destinés à recueillir les valeurs du tampon d'acquisition pour une acquisition répétée.

Type Tvpb = array[0..7] of byte; (* valeurs 0 ou 1 *)

Type du tableau utilisé pour la préparation et la récupération des valeurs des bits du port logique B de SYSAM (vpb signifie : Valeurs du Port B)

D. Les constantes à déclarer.

Ces constantes dont le but est purement pratique, ne sont pas disponibles dans la ProgrammationSYSAM_SP5.dll. Elles doivent être déclarées dans le code du programme utilisateur. 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. Leur utilité est d'obtenir un code plus sympathique et plus lisible : passer à une procédure la constante EA3 au lieu de la valeur 3 rend ce code plus clair. Il y aurait d'autres façons de faire, mais

cela obligerait à entrer dans le détail du langage pascal, ce qui n'est pas l'objet de cet article.

Constantes numériques :

Numéros des entrées

```
EA0 = 0 ;  
EA1 = 1 ;  
EA2 = 2 ;  
EA3 = 3 ;  
EA4 = 4 ;  
EA5 = 5 ;  
EA6 = 6 ;  
EA7 = 7 ;
```

Numéros des sorties :

```
SA1 = 1 ;  
SA2 = 2 ;
```

Index des calibres disponibles sur les entrées analogiques:

```
cal_10V = 0;  
cal_05V = 1;  
cal_01V = 2 ;  
cal_002V = 3 ;(*calibre 0,2 V*)
```

Mode des voies :

```
mvDiff      = true;  
mvNormal    = false;
```

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..3] of string  
= ('-10/+10', '-5/+5', '-1/+1', '-0,2/+0,2');
```

Valeurs des calibres :

```
calibresEA : array[0..3] of double  
= (10.0000, 5.0000, 1.000, 0.2000 );
```

Ces trois tableaux sont pratiques pour les boucles de code...

E. Le détail des fonctions

Afin de distinguer les fonctions et procédures de « ProgrammationSYSAM_SP5.dll » des procédures et fonctions standard de LAZARUS (pascal), dans le code du programme utilisateur, tous les noms commencent par « SP5_ »

1. PROCEDURES GENERALES

procedure SP5_Reinitialisation;

Remet l'interface SYSAM SP5 dans son état standard, à savoir:
- arrête les émissions et les acquisitions

- remet les calibres des entrées à +/-10V (index 0)
- désactive les voies, etc.

procedure SP5_FinUtilisation;

Libère la mémoire occupée par les structures présentes dans la DLL.
A appeler juste avant la fin du programme utilisateur

function SP5_Presente: Boolean;

Renvoie true si le pilote de SYSAM a été installé et si la centrale est connectée et utilisable

2. TOUT POUR L'ACQUISITION

2.A. Paramètres d'acquisition

procedure SP5_ActiveVoieA (numVoie : byte);

Active la voie n° numVoie (de EA0 à EA7)

procedure SP5_DesactiveVoieA (numVoie : byte);

Désactive la voie n° numVoie (de EA0 à EA7)

procedure SP5_CalibreVoieA (numVoie, calibre : byte);

Sélectionne le calibre d'une voie

Paramètres :

- numVoie : numéro de la voie: de EA0 à EA7
- calibre : 0 pour +/-10V, 1 pour +/-5V, 2 pour +/-1V et 3 pour +/-0,2V

procedure SP5_ModeDiff (canal1 : integer; mode1 : boolean);

Permet de placer un canal en mode différentiel

Un canal de SYSAM est constitué de deux entrées, par exemple EA0 et EA4 forment le canal 0. Si pour acquérir/mesurer un signal on utilise les deux bornes d'un même canal en mode différentiel, on évite d'utiliser la masse de SYSAM et on mesure la ddp VEA0 - VEA4.

Pour que ce soit possible, on active le mode différentiel

Paramètres:

- canal1 : numéro du canal à mettre en mode différentiel (de 0 (pour le canal 0) à 3 (pour le canal 3))
- mode1 : true pour activer le mode différentiel, false pour le désactiver.

function SP5_nbVoiesActives: integer

Permet de connaître à tout moment le nombre de voies activées. La valeur renvoyée peut faciliter l'exploitation des acquisitions.

2.B. Acquisition d'un point sur une voie

function SP5_AcquisitionVoieUnique(voie1: integer; var mesure1: word): integer;

Cette procédure provoque la mesure d'un point sur la seule voie1 et renvoie la valeur dans le paramètre mesure1 sous forme de valeur entière comprise entre 0 et 4095 (12 bits)

2.C. Acquisition d'un point en simultanée sur toutes les voies actives

procedure SP5_Acquisition(var M1: word);

Cette procédure provoque la mesure d'un point sur toutes les voies activées et renvoie les valeurs obtenues dans un tableau de type TTabMesures (exemple TabMesures), dont le premier élément (tabMesures[0]) est passé comme paramètre mesure1 de la procédure.

2.D.a. Préparation des acquisitions de points multiples

procedure SP5_DureesEtPoints(dureeTotale1:double; nbPts1:int64);

Permet de choisir la durée totale de l'acquisition et le nombre de points à acquérir. Cependant, le choix réalisé peut être légèrement modifié par cette procédure de manière à tenir compte du fait que le temps d'échantillonnage (temps séparant la mesure de deux points successifs) doit être un multiple

- de 100 ns si aucune des voies EA4, EA5, EA6, EA7 n'est activée en mode non différentiel
- de 2 µs si une des voies EA4, EA5, EA6, EA7 est activée en mode non différentiel

La modification éventuelle est toujours réalisée dans le sens d'une légère augmentation et n'a généralement lieu que pour les acquisitions très rapides.

Pour connaître les valeurs effectives de la durée totale et du nombre de points, utiliser les fonctions ci-dessous au §2.D.b.

procedure SP5_ModeDeclenchement(modeDecl1:integer);

Cette procédure permet de sélectionner le mode de déclenchement :

Valeur tsAucune = 0 de modeDecl1 : l'acquisition commence au moment où la procédure **SP5_DeclencheAcqMonoCoup** ou **SP5_DeclencheAcqPermanente** est appelée

Valeur tsSeuil = 1 de modeDecl1 : synchro sur seuil

Les procédures **SP5_DeclencheAcqMonoCoup** ou **SP5_DeclencheAcqPermanente** déclenchent l'acquisition, mais SYSAM SP5 ne commence à écrire les valeurs acquises dans sa RAM que lorsque le signal passe par une valeur de seuil donnée (réglable avec SP5_SeuilDeclenchement) dans un sens donné (réglable avec une SP5_SensDeclSeuil)

Valeur tsSynchroExt = 2 de modeDecl1 : synchro externe

Les procédures **SP5_DeclencheAcqMonoCoup** ou **SP5_DeclencheAcqPermanente** déclenchent l'acquisition, mais SYSAM SP5 ne commence à écrire les valeurs acquises dans sa RAM que lorsque le signal injecté sur l'entrée spéciale SYNCCHRO EXT. de SYSAM SP5 passe par une valeur proche de 1,6V dans le sens réglable par **SP5_SensDeclEchelon** (ci-dessous).

procedure SP5_VoieSynchro(voie1:integer)

Cette procédure permet de choisir l'entrée du signal de référence de la synchronisation sur seuil. L'entrée choisie doit avoir été activée AVANT ce choix...

procedure SP5_Pretrig(idxPretrig1: integer);

Cette procédure sert à la gestion de la pré-acquisition. Elle règle la durée de la pré-acquisition en % de la durée totale. Lorsqu'il y a pré-acquisition, SYSAM mémorise dans sa RAM les valeurs acquises dès le déclenchement par **SP5_DeclencheAcqMonoCoup** ou **SP5_DeclencheAcqPermanente**, tout en attendant au minimum idxPretrig1 % de la durée totale avant de commencer à tester les conditions de déclenchement.

Lorsque ces conditions sont réunies (soit t0 cet instant), elle efface les valeurs précédentes à l'instant t1 = t0 - idxPretrig / 100 * dureeTotale et commence à transmettre les valeurs acquises à partir de t1, qui sont récupérées par le programme utilisateur à l'aide de la fonction **SP5_LireTamponAcq** décrite ci-dessous.

procedure SP5_SensDeclSeuil(sensDecl1:integer);

Cette procédure sert à fixer le sens de passage par la valeur de seuil qui va provoquer le déclenchement de l'acquisition (en mode de synchronisation sur seuil uniquement)

Valeur sensDecl1 = sdMontant = + 1 : sens montant

Valeur sensDecl1 = sdDescendant = - 1 : sens descendant.

procedure SP5_SensDeclEchelon(sensDecl1:integer);

Cette procédure sert à fixer le sens de passage par 1,6 V du signal injecté sur l'entrée SYNCHRO EXT qui va provoquer le déclenchement de l'acquisition (en mode de synchronisation SYNCHRO EXT.)

Valeur sensDecl1 = sdMontant = + 1 : sens montant

Valeur sensDecl1 = sdDescendant = - 1 : sens descendant.

procedure SP5_SeuilDeclenchement(UseuilDecl1 :single);

Cette procédure sert à fixer la valeur de seuil de déclenchement de l'acquisition (en mode de synchronisation sur seuil.)

2.D.b. Paramètres effectifs d'acquisition tenant compte des performances de SYSAM

function SP5_DtAcq: double;

Cette fonction renvoie l'intervalle de temps (Dt = delta t) entre deux points successifs. Cet intervalle est déterminé par la procédure SP5_DureesEtPoints décrite ci-dessus, comme le rapport dureeTotale/ nbPts et est forcément (pour des raisons techniques) un multiple du temps d'échantillonnage de base de SYSAM, à savoir :

- de 100 ns si aucune des voies EA4, EA5, EA6, EA7 n'est activée en mode non différentiel

- de 2 µs si une des voies EA4, EA5, EA6, EA7 est activée en mode non différentiel.

function SP5_DureeTotaleAcq: double;

Cette fonction renvoie la durée totale effective de l'acquisition, qui peut parfois différer légèrement par excès de la durée totale souhaitée et passée en paramètre de la procédure **SP5_DureesEtPoints**.

function SP5_NbPtsAcq: integer;

Cette fonction renvoie le nombre de points effectif de l'acquisition, qui peut parfois différer légèrement par excès du nombre de points souhaité et passé en paramètre de la procédure **SP5_DureesEtPoints**.

2.D.c. Gestion de la récupération des valeurs acquises

procedure SP5_NbPtsDansTamponAcq(nbPts1: Integer);

Fixe le nombre de points dans le tampon d'acquisition, ainsi que la taille de ce tampon d'acquisition. Le contenu du tampon ne peut pas dépasser 256 valeurs (à partager entre les voies actives). Le nombre de valeurs dans le tampon est normalement égal au produit du nombre de points dans le tampon multiplié par le nombre de voies actives (une valeur par voie active et par point).

Cette procédure ne doit être appelée qu'après l'activation/désactivation des voies et avant l'acquisition.

Le nombre de points passé en paramètre n'est pas forcément celui qui est retenu. Pour obtenir la valeur retenue, utiliser la fonction suivante. Il est indispensable d'utiliser cette valeur dans l'exploitation des acquisitions.

function SP5_NbPtsTampon: integer;

Renvoie le nombre de points dans le tampon effectivement retenu après l'appel de la procédure précédente. Cette façon de procéder permet de sécuriser le processus et d'éviter de dépasser les limites du tableau de récupération du tampon. Le nombre de points doit être inférieur ou égal à 256 div nbVoiesActives.

function SP5_LireTamponAcq(var M1: word): integer;

Une fois l'acquisition lancée, SYSAM effectue les mesures au rythme régulier imposé et les stocke dans sa propre RAM, de manière indépendante du programme utilisateur. Il s'agit de récupérer ces valeurs, ce qu'on fait par paquets de points dont le nombre a été fixé par la procédure **SP5_NbPtsDansTamponAcq(nbPts1: Integer)**.

Appelons n le nombre de points dans le tampon.

L'appel de la fonction **SP5_LireTamponAcq** provoque la lecture des valeurs des n points dans la RAM et leur placement dans un tableau de type TTamponSP5 (nommé par exemple TamponSP5) dont le premier élément (TamponSP5[0]) doit être passé en argument M1 de cette fonction.

La valeur renvoyée par la fonction reste nulle tant que SYSAM n'a pas acquis le nombre de points n nécessaire.

Lorsque le nombre suffisant n de points a été acquis, la fonction retourne une valeur égale au nombre de valeurs présentes dans le tableau (donc n). Le programme peut alors accéder à ces valeurs et les traiter (remplir un tableau, tracer une courbe...)

Ensuite, on procède de même pour récupérer les n points suivants, et ainsi de suite jusqu'à ce qu'on ait récupéré toutes les valeurs de la RAM, acquises par SYSAM pendant l'acquisition en cours. La fonction **SP5_LireTamponAcq** est donc généralement présente dans une boucle de code.

2.E. Déclenchement de l'acquisition répétée.

procedure SP5_DeclencheAcqMonoCoup;

Prépare physiquement l'interface en tenant compte de tous les choix faits précédemment, puis lance l'acquisition d'une trame (durée finie choisie ci-dessus).

procedure SP5_DeclencheAcqPermanente;

Prépare physiquement l'interface en tenant compte de tous les choix faits précédemment, puis lance une acquisition sans fin (durée infinie).

procedure SP5_ArreteAcq;

Arrête les acquisitions multiples (de durée finie « monocoup » ou permanente)

2.F. Fonctions de conversion pour les entrées

function SP5_Ux(voie1: Integer; vNum1:integer):double;

Renvoie la tension entrée sur la voie1, en fonction du calibre et de la valeur numérique. Cette fonction est utilisée pour la conversion de la valeur numérique entière résultant d'une mesure (entre 0 et 4095) en la valeur de la tension mesurée.

Paramètres :

Voie1 = entrée analogique concernée (EA0 à EA7) . Le calibre courant de cette voie est utilisé pour le calcul, de même que les données de calibrage de SYSAM SP5.

Mesures1 = valeur numérique (entre 0 et 4095) = résultat d'une mesure antérieure

RESULTAT : tension mesurée, tenant compte des paramètres de calibrage de la voie de SYSAM SP5 utilisée.

function SP5_ValUx(voie1: integer; U1: single):integer

Fonction réciproque de la précédente.

Paramètres :

Voie1 = entrée analogique concernée (EA0 à EA7). Le calibre courant de cette voie est utilisé pour le calcul

U1 = tension à convertir en valeur numérique

RESULTAT : valeur numérique (entre 0 et 4095) correspondant à la valeur de tension *)

3. TOUT POUR L'EMISSION

procedure SP5_DeclencheSA(SA: integer);

Démarrage de l'émission.

Valeurs du paramètre: SA

SA = 1 pour SA1

SA = 2 pour SA2

SA = 3 pour SA1 et SA2 en même temps.

procedure SP5_ArreteSA(SA: integer);

Arrêt de l'émission.

Valeurs du paramètre: SA

SA = SA1 = 1 pour SA1

SA = SA2 = 2 pour SA2

SA = SA1 + SA2 pour SA1 et SA2 en même temps

procedure SP5_DeclencheSAetAcq(SA: integer);

Déclenche simultanément l'émission et l'acquisition

Valeurs du paramètre: SA

SA = SA1 = 1 pour SA1

SA = SA2 = 2 pour SA2

SA = SA1 + SA2 pour SA1 et SA2 en même temps

procedure SP5_ArreteSAetAcq;

Arrête toutes les émissions et l'acquisition en une seule opération.

function SP5_ValUxSA(SA:integer; U1: single):integer;

Renvoie la valeur numérique correspondant à la tension U1, en tenant compte des paramètres d'étalonnage de SYSAM

Paramètres :

SA: sortie concernée (valeurs: SA1 = 1 ou SA2 = 2)

U1: tension en V

Résultat: valeur numérique comprise entre 0 et 4095 correspondant à la tension U1

Fonction utilisée pour obtenir la valeur à passer en paramètre de la procédure ***SP5_EmissionValContinue*** ci-dessous.

procedure SP5_EmissionUContinue(SAi: integer; U1: single);

Provoque l'émission sur la sortie SAi (SAi = SA1 = 1 ou SAi = SA2 = 2) d'une tension continue de valeur U1.

procedure SP5_EmissionValContinue(SAi: integer; vNum: integer);

Provoque l'émission sur la sortie SAi (SAi = SA1 = 1 ou SAi = SA2 = 2) d'une tension correspondant à la valeur entière vNum comprise entre 0 (-10V) et 4095 (+10V).

Emission de signaux créés de toutes pièces

Pour créer un signal de toutes pièces, il faut utiliser les procédures suivantes, dans l'ordre, avant de déclencher l'émission.

Le principe est de réserver une portion de la RAM de SYSAM devant recevoir un nombre donné de VNE, et de transmettre ces VNE. Ces VNE sont converties par le CNA de SYSAM en signal de tension appliqué sur la sortie (SA1 ou SA2) concernée.

Etape 1 : paramétrage de l'émission.

procedure SP5_ParametreEmission (SAi: integer; deltat : single; nbVal : int64);

On paramètre

- la sortie utilisée,
- l'intervalle de temps entre deux valeurs (temps d'échantillonnage, qui sera arrondi au multiple de te)
- le nombre de valeurs (indirectement la mémoire de SYSAM réservée aux valeurs du signal). Dans le contexte de l'outil « ProgrammationSYSAM_SP5.dll », ce nombre est limité à 100000 par défaut.

procedure SP5_MonocoupSA(SA: integer; monocoupSA0:boolean);

Positionne la sortie SA sur monocoup si monocoupSA0 vaut true. Une émission monocoup s'arrête dès que la série de valeurs transmises à la RAM de SYSAM a été émise (une seule fois) sur la sortie concernée.

En mode permanent, l'émission est cyclique: l'émission de la série de valeurs reprend à son début sans interruption.

Pour émettre un signal périodique, il faut que monocoupSA0 soit sur false, ce qui est le cas par défaut (au démarrage et après l'appel de SP5_Reinitialisation).

Etape 2 : préparation du tableau des valeurs.

procedure SP5_PlaceValeur(SAi : integer; position : integer; valeur : single);

Cette procédure permet de placer la valeur « valeur » à la position « position » dans le tableau des valeurs de sortie de la sortie SA1. « Valeur » est compris entre -10V et + 10V (résolution 12 bits). On répète cette procédure jusqu'à ce que le nombre nbVal de cases soit correctement rempli (position doit varier de 0 à nbVal - 1).

Etape 3 : transmission des valeurs à la RAM de SYSAM.

procedure SP5_TransmetValeurs(SAi: integer) ;

Provoque la transmission des valeurs numériques à la RAM de SYSAM. Ne doit être appelée que si les nbVal ont été préparées par la procédure SP5_PlaceValeur, et avant le déclenchement de l'émission.

Etape 4 : utiliser ***SP5_DeclencheSA*** (ci-dessus) pour déclencher l'émission.

4. PORT LOGIQUE B.

L'accès physique au port logique B de SYSAM nécessite le boîtier BOLOGIC.

Signification des notations:

plb : port logique B

vpb : valeurs du port logique.

procedure SP5_EmetPLB0_3(B1: Byte);

Emet sur le port logique B, bits 0 à 3, les 4 premières valeurs (0 ou 1) du tableau vpb, dont seule la première case est passée par valeur dans B1 lors de l'appel de cette procédure

Exemple d'appel : SP5_EmetPLB0_3(vpb[0])

Autrement dit: fixe l'état des bits 0 à 3 du port logique aux 4 premières valeurs du tableau vpb.

procedure SP5_EmetPLB4_7(B1: Byte);

Emet sur le port logique B, bits 4 à 7, les 4 dernières valeurs (0 ou 1) du tableau vpb, dont seule la première case vpb[0] est passée par valeur dans B1 lors de l'appel de cette procédure.

procedure SP5_LitPLB0_3(var B1: Byte);

Lit les bits 0 à 3 du port logique B et les place dans le tableau vpb aux quatre premières places.

Le paramètre B1 sera la première case de vpb (vpb[0])

Exemple d'appel : SP5_LitPLB0_3(vpb[0])

procedure SP5_LitPLB4_7(var B1: Byte);

Lit les bits 4 à 7 du port logique B et les place dans le tableau vpb aux quatre dernières places.

Le paramètre B1 sera la première case de vpb (vpb[0]).

Exemple d'appel : SP5_LitPLB4_7(vpb[0])

procedure SP5_EmetPLB(numBit, B1: byte);

Emet (fixe) individuellement le bit de valeur val1 (0 ou 1) sur port logique B

function SP5_LitPLB(numBit: byte): byte;

Renvoie l'état du bit numBit (0 à 7) de valeur 0 ou 1 sur port logique B.

5. CAPTEURS

function SP5_Capteur(canal: byte): byte;

Renvoie le numéro du capteur sur le canal « canal ».

Quatrième 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'ils contiennent...)

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). Enregistrez immédiatement le projet dans un nouveau dossier.

Après le mot réservé « type » apparaît une déclaration de type TForm1, qui est le type de la fiche-fenêtre originale que nous créons en ajoutant des composants.

1. Déclarations .

Ouvrez le fichier « Declarations_Pascal.txt » (dans le zip contenant la DLL), copiez l'intégralité de ce fichier dans la partie interface, juste avant le mot réservé « Type » de la fiche TForm1 (voir ci-contre). On remarquera la présence des types décrits plus haut ainsi que celle des constantes.

Rien ne vous empêche à ce stade de modifier le nom des constantes ; par contre, pour les structures destinées à recevoir les résultats des mesures (à savoir

```
Type TTabMesures = Array[0..7] of word;
```

```
Type TTamponSP5 = array[0..256 - 1] of word;
```

```
Type Tvpb = array[0..7] of byte;),
```

il est impératif de ne rien changer à ce qui se trouve à droite du signe « = ».

Examinons les déclarations copiées

Prenons l'exemple de la procédure

```
procedure SP5_DureesEtPoints(dureeTotale1:double; nbPts1:int64); stdCall;  
external 'ProgrammationSYSAM_SP5.dll' ;
```

Chacune des déclarations commence par le mot réservé « procedure » ou le mot réservé « function ». Suit le nom de la procédure, puis éventuellement le ou les paramètres avec



```
Unit1  
unit Unit1;  
{$mode objfpc}{$H+}  
  
interface  
uses  
  Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs;  
(* copiez ici le contenu du fichier Declarations_Pascal.txt *)  
  
type  
  TForm1 = class(TForm)  
  private  
    { private declarations }  
  public  
    { public declarations }  
  end;  
  
var  
  Form1: TForm1;  
  
implementation  
  
initialization  
  {$I unit1.lrs}  
  
end.
```

leur type, et s'il s'agit d'une fonction, le type du résultat. Suit le mot réservé stdCall, dont nous n'étudierons pas la signification.

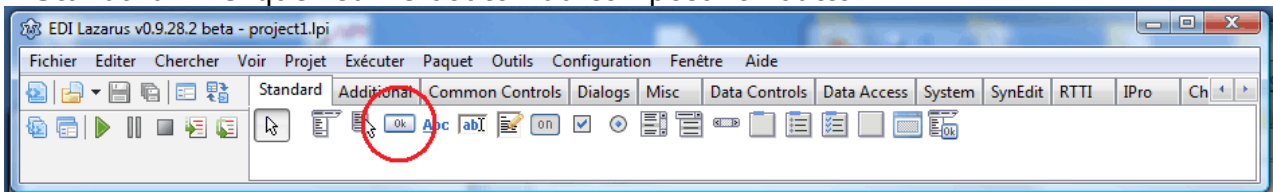
Le mot réservé « external » suivi de « ProgrammationSYSAM_SP5.dll » indique que la procédure se trouve dans cette dll. Cette façon de faire se nomme « chargement statique » de la DLL. Le programme utilisateur vérifie au démarrage (automatiquement) la présence de la DLL et « charge » les procédures ainsi déclarées. Si la DLL n'est pas présente, le programme ne peut pas fonctionner.

On peut ne déclarer que les procédures dont on a besoin.

2. Première acquisition (ou : réalisation d'un voltmètre).

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-chantier 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 (IO).

• Renommez le bouton : recherchez dans l'onglet Propriétés de l'IO la rubrique « Name », et changez « Button1 » en « BAcquisition ». Cette propriété doit respecter la syntaxe du pascal, à savoir

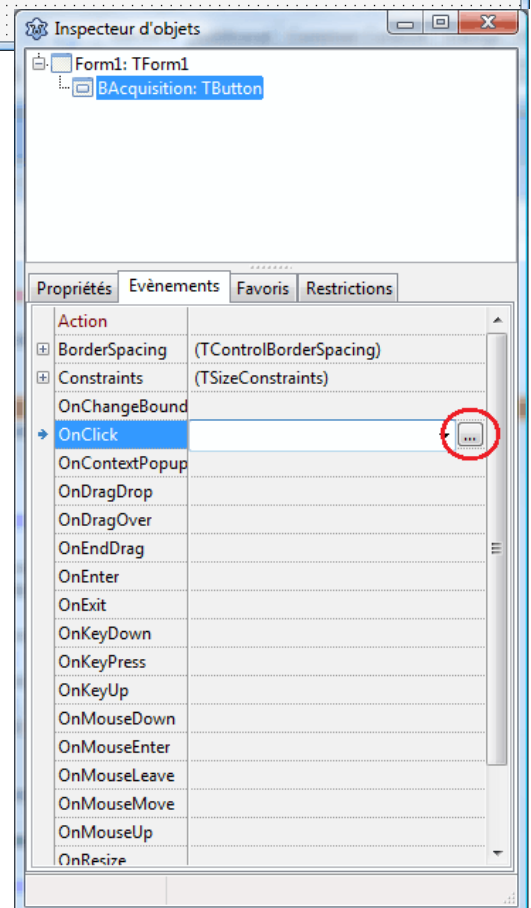
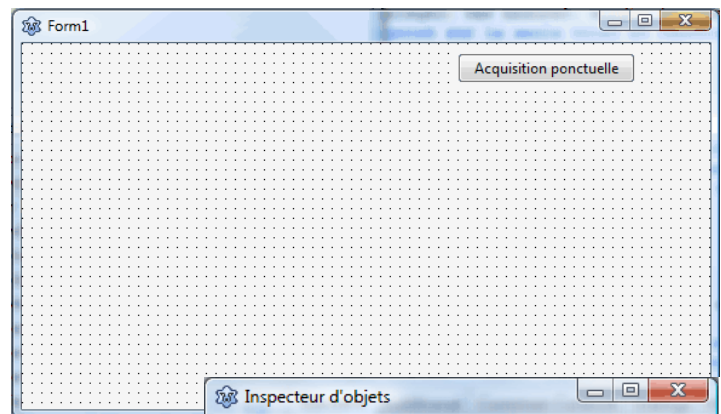
- le premier caractère est une lettre (pas un chiffre)
- ni espace ni caractères accentués ou spéciaux. Le caractère _ peut être utilisé.

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

• Allez dans l'onglet « Evènements » de l'inspecteur (F11) et double-cliquez à droite de la rubrique «OnClick» sur les trois points, 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 ;
```

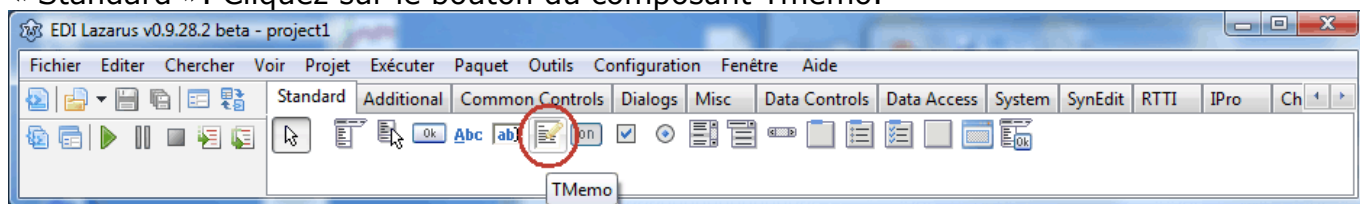


- Ajoutez entre le « begin » et le « end » les instructions permettant de réaliser une mesure unique sur l'entrée analogique EA0 avec le calibre 5V (colonne de gauche du tableau suivant).

Instructions	Actions correspondantes
Voie := EA0 ;	Choix de la voie EA0
SP5_calibreVoieA (voie, cal_05V) ;	Réglage du calibre de l'entrée EA0 sur +/-5V (notez l'usage de la constante prédéfinie cal_05V)
SP5_activeVoieA (voie) ;	Activation de l'entrée EA0 (la seule qui fera une mesure pour le moment).
SP5_AcquisitionVoieUnique(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 ponctuelle »), la VNE (valeur numérique 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, auquel on rajoute des lignes de texte au fur et à mesure des besoins.

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



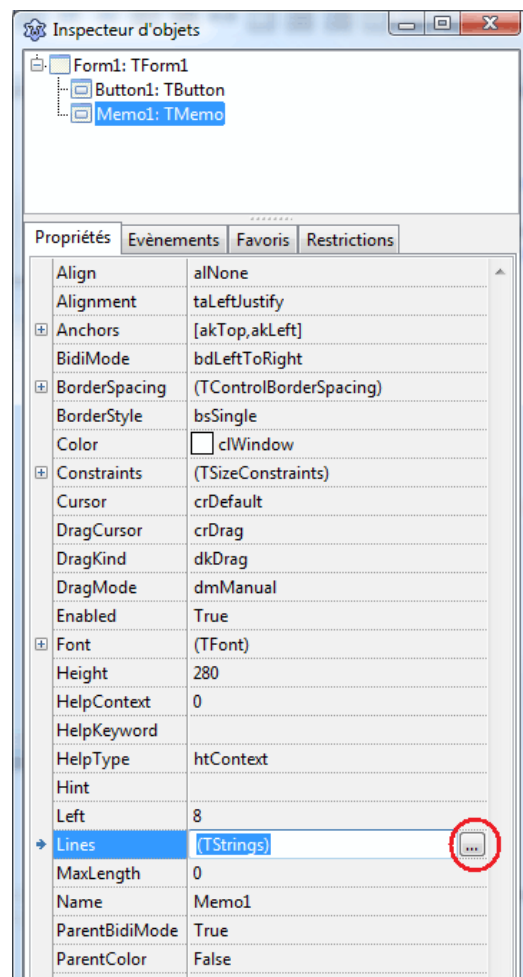
- Cliquez ensuite sur la fiche chantier, là où vous voulez déposer ce composant. Agrandissez ou réduisez la taille du memo en saisissant un de ses coins avec la souris, de manière à ce qu'il ne recouvre pas le bouton. Déplacer éventuellement le bouton. A l'aide de l'inspecteur d'objets, vous pouvez également 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 à virgule 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 := SP5_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.

- 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.

Changez le réglage du générateur ; recommencer l'acquisition...

```

"Unit1
type
  { TForm1 }
  TForm1 = class(TForm)
    BAacquisition: TButton;
    Memo1: TMemo;
    procedure BAacquisitionClick(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;

implementation
  { TForm1 }
  procedure TForm1.BAacquisitionClick(Sender: TObject);
  Var
    mesure1 : word ;
    voie : byte ;
    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*)
  begin
    Voie := EA0 ;
    SP5_calibreVoieA (voie, cal_05V ) ;
    SP5_activeVoieA (voie ) ;
    SP5_AcquisitionVoieUnique(voie, mesure1) ;

    memo1.lines.add('Mesure sur l'entrée '+ nomsEA[voie]);
    s := intToStr(mesure1);
    s := ' valeur entière: v = ' + s;
    memo1.lines.add(s);

    u := SP5_ux(voie, mesure1);
    s := floatToStrF( u, ffgeneral, 4 , 2);
    s := ' tension :      u = ' + s + ' V';

    memo1.lines.add(s);

  end;

initialization
  {$I unit1.lrs}
end.

```

3. Une autre acquisition (ou : réalisation d'un voltmètre multiple)

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 huit entrées.

Cette seconde acquisition sera déclenchée par un autre bouton que vous placerez à votre convenance sur la fiche-chantier. Assurez-vous que le bouton est bien dans l'IO. Renommez ce bouton (BACquisitionPoint), changez son « caption » en « Acquisition d'un point » ; cliquez dans l'inspecteur d'objets à droite de la rubrique onClick de l'onglet « Evénements ». Vous obtenez dans le code l'enveloppe du gestionnaire de l'événement de clic sur ce bouton, du style:

procedure TForm1.BACquisitionPointClick(sender : Tobject).

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

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

- 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');
```

{Calibre +/- 10V sur les 8 entrées}

```
for voie := EA0 to EA7 do SP5_calibreVoieA(voie, cal_10V);
```

{activation de toutes les voies :}

```
for voie := EA0 to EA7 do SP5_activeVoieA(voie);
```

{réalisation de l'acquisition :}

```
SP5_acquisition(TM[0]);
```

{lors de l'exécution du programme et après clic sur le bouton, à ce stade, le tableau "TM" contient une VNE chaque case}

{affichage des résultats dans le memo1 : }

```
for voie := EA0 to EA7 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 := SP5_ux(voie, TM[voie]);
```

{conversion de la valeurs de la tension en chaîne de caractères :}

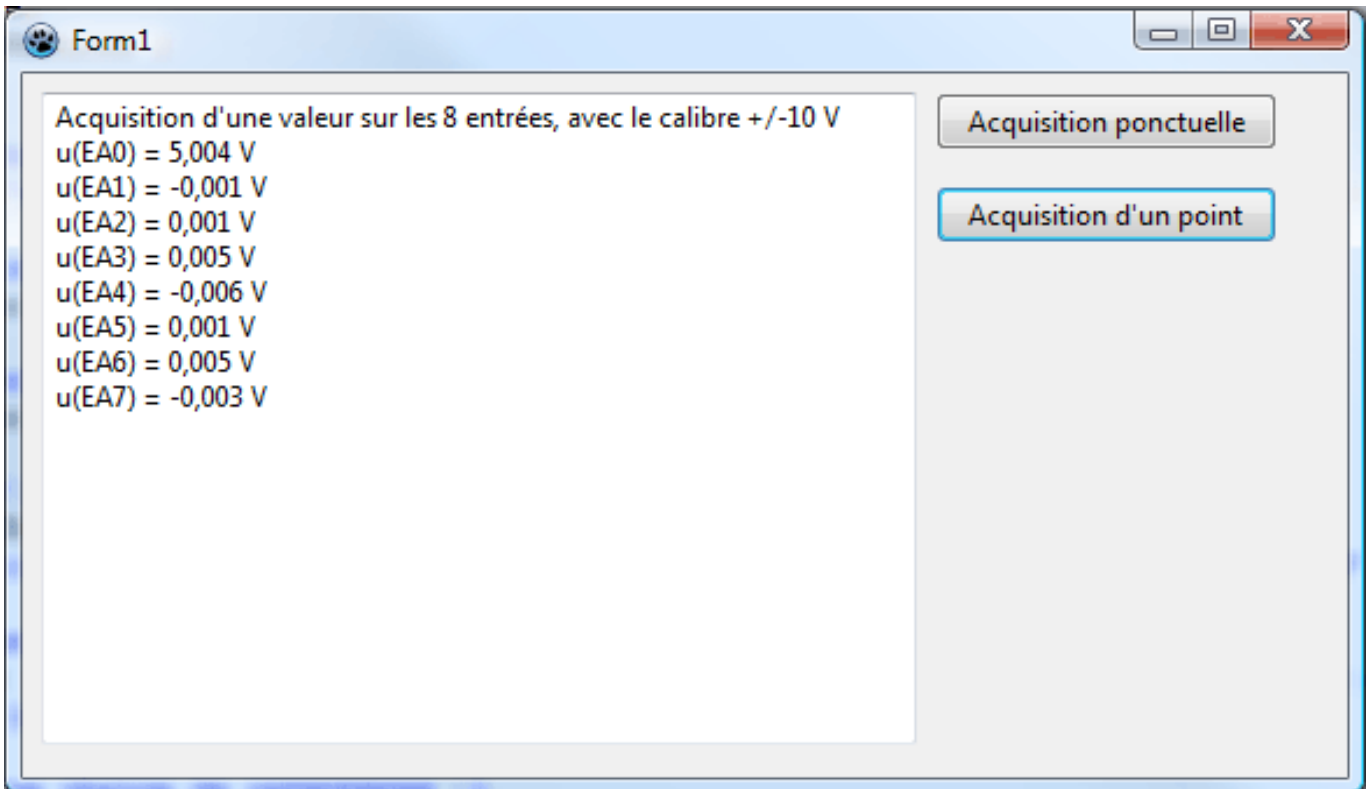
```
s1 := floatToStrF(u , ffgeneral, 4, 2);  
s := s + s1 + ' V';
```

{ajout de la chaîne dans le memo1 :}

```
memo1.lines.add(s);  
end; { fin du bloc }
```

Voici le code obtenu (sans les commentaires):

Ainsi qu'une capture d'écran après un clic sur « Acquisition d'un point » et une tension non nulle sur EA0, on obtient :



Les mesures réalisées avec cette technique ne sont pas parfaitement simultanées. La technique ne convient donc pas aux mesures répétées trop rapidement. Pour cela, d'autres fonctions sont utilisées ; c'est l'objet du paragraphe suivant.

4. Une acquisition répétée (ou : réalisation d'un enregistreur)

Nous allons maintenant réaliser une acquisition répétée monocoup sur les entrées EA0 et EA3.

Donc, dans l'ordre, nous commandons :

- l'activation des voies EA0 et EA3 et la désactivation de toutes les autres ;
- le calibre 5 V sur ces deux entrées utilisées
- 501 mesures réparties sur une durée de 100 ms.

Ajoutez un nouveau bouton sur la fiche, changez sa propriété name en « BAcqPointsMultiples », son caption en « Acquisition multiple monocoup » et créez l'enveloppe de son gestionnaire d'événement OnClick. Dans ce gestionnaire, placez le code minimaliste suivant :

Avant le mot réservé begin (déclaration des variables locales):

```
var kV : (* compteur pour les voies *)  
integer;
```

(* paramètres pour fixer les idées *)

```
dureeTotale : single;  
nbPts : integer;
```

Entre le « begin » et le « end » du gestionnaire :

```
SP5_reinitialisation;
```

```
memo1.lines.add(' ');  
memo1.lines.add( 'Acquisition répétée monocoup');
```

(* choix des paramètres, à faire varier pour les tests *)

```
dureeTotale := 100e-3; (* 100 ms *)  
nbPts      := 501;
```

(* activation des voies EA0 et EA3 et réglage des calibres 10V *)

```
SP5_ActiveVoieA( EA0);  
SP5_CalibreVoieA( EA0, cal_10V);
```

```
SP5_ActiveVoieA( EA3);  
SP5_CalibreVoieA( EA3, cal_10V);
```

(* réglage de la durée totale et du nombre de points à recueillir *)

```
SP5_DureesEtPoints( dureeTotale, nbPts);
```

(* déclenchement *)

```
SP5_declencheAcqMonocoup;
```

Si vous compilez et lancez le programme avec ce code, l'acquisition se fera en 100 ms. Mais aucune valeur ne sera envoyée au programme utilisateur. Comme précédemment, il faut rajouter du code pour récupérer les valeurs mesurées et les exploiter : afficher à l'écran, tracer un graphique...

Complétons le code (le plus simple étant d'effacer totalement le code déjà écrit et de le remplacer par ce qui suit : (copier-coller !)

Avant le mot réservé « begin » (déclaration des variables locales):

```
var kV,      (* compteur pour les voies *)  
    kP,      (* compteur des points dans un tour *)  
    numPt,   (* numéro de point en cours de traitement *)  
    tp       (* valeur de remplissage de tampon *)  
    : integer;  
TamponSP5: (* tableau recevant les valeurs acquises lors d'un tour *)  
    TTamponSP5;
```

(* paramètres pour fixer les idées *)

```
dureeTotale : single;  
nbPts,      (* nombre de points à "acquérir" *)  
nbPT       (* nombre de points du tampon de récupération des valeurs *)  
    : integer;
```

(* auxiliaires *)

```
vNum: Word;  
u1,t1: single;  
AcquisitionTerminee: boolean;  
s1 (* chaîne servant d'outil de mise en forme pour l'affichage des résultats *)  
    : string;
```

Entre le « begin » et le « end » du gestionnaire :

(* effacement du memo *)
memo1.lines.add(' ');

(* on place un titre dans le memo *)

memo1.lines.add('Acquisition monocoup');

(* choix des paramètres, à faire varier pour les tests *)

dureeTotale := 100e-3; (* 100 ms *)
nbPts := 500;
nbPT := 50;

(* nombre de points souhaité dans le tampon. Minimum : 1 ; maximum : 256 / nombre de voies actives. Avec ce réglage (* nbPT := 50 *), et sachant que nous activerons 2 entrées, le tampon contiendra 100 valeurs, dans l'ordre temporel :
vneEA0(0) ; vneEA3(0) ; vneEA0(Dt) ; vneEA3(Dt) ; vneEA0(2*Dt) ; vneEA3(2*Dt)
... vneEA0((50-1) * Dt) ; vneEA3((50-1)*Dt)

Le contenu du tampon correspond à un temps d'acquisition égal à :
$$\text{dureeTotale} / \text{nbPts} * \text{nbPtsDansTampon}$$

Le traitement des valeurs d'un tampon se fait lors d'un « tour » ou boucle. La valeur nbTP conditionne ainsi le temps réel: pour avoir un vrai temps réel, il faut que le nombre de points du tampon puisse être traité par le programme pendant la durée correspondante, de manière à être traité avant la fin de l'acquisition du tour suivant *)

(* remise à l'état standard de SYSAM SP5 (facultatif) *)
SP5_Reinitialisation;

(* activation des voies EA0 et EA3 et réglage des calibres 10V *)
SP5_ActiveVoieA(EA0);
SP5_CalibreVoieA(EA0, cal_10V);

SP5_ActiveVoieA(EA3);
SP5_CalibreVoieA(EA3, cal_10V);

(* réglage de la durée totale et du nombre de points à recueillir *)
SP5_DureesEtPoints(dureeTotale, nbPts);

(* réglage du nombre de points dans le tampon *)
SP5_NbPtsDansTamponAcq(nbPT);

(* préparation diverses *)
acquisitionTerminee := false;

(* mise à zéro du compteur de points *)
numPt := 0;

(* déclenchement *)
SP5_declencheAcqMonocoup;

(* BOUCLE DE RECUPERATION DES VALEURS en temps presque réel
(une boucle = 1 tour = récupération et traitement de nbPtsTampon points.

« LireTamponAcq » renvoie une valeur différente de zéro à chaque fois qu'il s'est écoulé le temps correspondant au remplissage du tampon et que SYSAM a acquis un nombre de points contenu dans le tampon *)

```
repeat  
  tp := SP5_LireTamponAcq(TamponSP5[0]);  
  if tp > 0 then  
    begin
```

(* à l'instant où $tp > 0$, on sait que le tampon a été rempli de nbPT mesures renvoyées au programme utilisateur dans le tableau TamponSP5 *)

(* TRAITEMENT DES VALEURS récupérées en temps quasi-réel.
Comme exemple, un simple ajout des valeurs récupérées dans le memo1, avec une mise en forme sommaire. Ce traitement n'est pas très rapide : l'ajout de lignes dans un mémo est lent *)

(* mise à zéro du compteur des points du tour *)

```
  kP := 0;
```

(* boucle sur les points du tampon *)

```
  while (kP < SP5_nbPtsTampon) (* test de non fin de tampon *)  
    and (numPt < SP5_nbPtsAcq ) (* test de non fin d'acquisition *)  
  do  
    begin
```

(* préparation de la chaîne de caractères s1 correspondant au point numPt *)

```
      s1 := intToStr( numPt );
```

(* instant correspondant *)

```
      t1 := numPt * SP5_DtAcq;
```

```
      s1 := s1 + ' - t = ' + floatToStrF(t1, ffgeneral, 5,2);
```

```
      memo1.lines.add( s1);
```

(* valeur numérique acquise *)

```
      vNum := TamponSP5[ kP * SP5_nbVoiesActives];
```

```
      s1 := ' - EA0 --> ' + IntToStr( vNum );
```

(* valeur de la tension mesurée *)

```
      u1 := SP5_Ux( EA0 , vNum);
```

(* ajout de la valeur de tension dans la chaîne *)

```
      s1 := s1 + ' ---- u = ' + floatToStrF(u1, ffgeneral, 5,2) + ' V';
```

```
      memo1.lines.add( s1);
```

(* valeur numérique acquise *)

```
      vNum := TamponSP5[ kP * SP5_nbVoiesActives + 1];
```

```
      s1 := ' - EA3 --> ' + IntToStr( vNum);
```

(* valeur de la tension mesurée *)

```
      u1 := SP5_Ux( EA3 , vNum);
```

(* ajout de la valeur de tension dans la chaîne *)

```
      s1 := s1 + ' ---- u = ' + floatToStrF(u1, ffgeneral, 5,2) + ' V';
```

```
      memo1.lines.add( s1);
```

(* compteur de points global : on passe au numéro suivant*)

```
      inc(numPt);
```

(* compteur de points du tour : on passe au numéro suivant *)

```
      inc(kP);
```

end; (* while *)

(* test de fin d'acquisition. On se base sur le nombre de points souhaité (nbPts) car

1. le tampon peut ne pas être rempli à nbPtsTampon Points au dernier tour, si le nombre total de points n'est pas un multiple entier de nbPtsTampon

2. le nombre de points renvoyé par SP5_nbPtsAcq et sur lequel est paramétré SYSAM peut être supérieur au nombre souhaité pour des raisons techniques *)

if numPt >= nbPts - 1 then acquisitionTerminee := true;

end; (* if tp > 0 then *)

until acquisitionTerminee;

SP5_arreteAcq;

(* désactivation des voies *)

for kV := 0 to 7 do SP5_DesactiveVoieA(kV);

Lors du lancement du programme, un clic sur « Acquisition multiple monocoup » provoque donc ici le défilement des VNE acquises et converties en tensions dans le mémo. Il serait évidemment plus intéressant de faire dessiner une courbe au programme...

Branchez un GBF sur les entrées EA0 et EA3, faire les réglages et lancer une acquisition. Testez en faisant varier les paramètres (dureeTotale, nbPts, nbPT). Ne poussez pas trop sur le nombre de points et la durée totale, car dans l'état actuel du code, on peut difficilement interrompre une acquisition trop longue.

5. Emission d'une tension continue.

Ajoutez un nouveau bouton sur la fiche, changez sa propriété « name » en « BEmissionUContinue », son « caption » en « Emission tension continue » ; créez l'enveloppe de son gestionnaire d'événement OnClick, puis placez dans ce gestionnaire les instructions :

memo1.lines.add('Emission d'une tension continue de 2,00 V ');

SP5_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

memo1.lines.add('Emission d'une tension continue de 2,00 V ');

{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 :}
SP5_EmissionValContinue(SA1 , vNum1);

end ;

Il ne reste plus qu'à tester...

6. Emission d'une tension sinusoïdale.

Principe : pour la sortie utilisée :

A. ON PRÉPARE LES PARAMÈTRES D'ÉMISSION :

- a. Intervalle de temps entre deux valeurs successives : *deltat*. Cet intervalle est forcé-ment un multiple du temps d'échantillonnage de SYSAM en sortie $T_e = 200$ ns.
- b. Le nombre de valeurs à « émettre », *nbVal* , limité à environ 100000 (environ 2/5èmes de la RAM de SYSAM)

Procédure utilisée :

procedure SP5_ParametreEmission (SAi: integer; deltat : single; nbVal : int64);
SAi = SA1 ou SA2.

B. ON PREPARE UN TABLEAU (caché dans « ProgrammationSYSAM_SP5.dll ») **DE VALEURS** de tensions à émettre. Ce tableau est dimensionné par l'instruction précédente. Toutes les va-leurs doivent être comprises entre -10 V et + 10 V. (calibre unique)

Procédure utilisée :

procedure SP5_PlaceValeur(SAi : integer; position : integer; valeur : single);

C. ON ENVOIE LES VALEURS DU TABLEAU A LA RAM DE SYSAM

procedure SP5_TransmetValeurs(SAi: integer);

Il faut avoir fait au moins une fois les trois opérations précédentes, dans l'ordre avant d'utiliser la suivante.

D. ON DECLENCHE L'ÉMISSION

procedure SP5_DeclencheSA(SAi: integer);

SAi = SA1 pour faire émettre la sortie analogique SA1

SAi = SA2 pour faire émettre la sortie analogique SA2

SAi = SA1 + SA2 pour commencer l'émission simultanée sur les deux entrées.

N.B. : pour l'arrêt de l'émission, utiliser la procédure **SP5_ArreteSA(SAi: integer);**

Si on veut changer le signal émis, il faut préalablement arrêter l'émission, et au minimum changer les valeurs du tableau.

On peut arrêter et re-émettre à loisir sans forcément re-préparer les valeurs ou les autres paramètres d'émission.

Lorsqu'on émet un signal périodique, il faut veiller à ce que la durée totale de l'émission soit un multiple entier de la période du signal.

Exemple de code pour l'émission d'un sinus

Ajoutez un nouveau bouton sur la fiche, changez sa propriété name en « BEmissionSinus », son caption en « Emission d'une tension sinusoïdale » ; créez l'enveloppe de son gestionnaire d'événement OnClick, puis placez dans ce gestionnaire les instructions :

Avant le mot réservé « begin » (déclaration des variables locales):

```
var
  k: integer;
  t,u : single ;
  (* caractéristiques de la tension à émettre *)

  per, ampl, phase: single;
  sortieUtilisee: integer;
```

Entre le « begin » et le « end » du gestionnaire :

```
  if memo1.lines.count > 2000 then memo1.lines.clear;

  (* paramètres du signal *)
  per := 2.00e-3 ;
  ampl := 9 ;
  phase := pi/9;
  memo1.lines.add(' ');
  memo1.lines.add( 'Emission d"un signal sinusoïdal');
  memo1.lines.add( 'Fréquence: '+ floatToStrF(1/per, ffgeneral, 4,2)+ ' Hz');
  memo1.lines.add( 'Amplitude: '+ floatToStrF(ampl, ffgeneral, 4,2)+' V');
  (* choix de la sortie à utiliser *)
  sortieUtilisee := SA1 ;
  (* 1. On fixe les paramètres d'émission : 100 valeurs à transmettre, correspondant à
  exactement une période du signal *)
  SP5_ParametreEmission( sortieUtilisee, per/100, 100);
  (* l'intervalle de temps deltat entre deux valeurs émises est le 100eme de la période,
  arrondi au multiple de 200ns le plus proche *)
  (* 2. Remplissage du tableau *)
  for k := 0 to 100 - 1 do
    begin
      t := k * per/100;
      u := ampl * sin( 2 * pi * t / per + phase);
      SP5_PlaceValeur(sortieUtilisee, k, u);
    end; (* for k := 0 .... *)
  (* 3. transmission des valeurs à SYSAM *)
  SP5_TransmetValeurs (SortieUtilisee);
  (* 4. Déclenchement de l'émission *)
  SP5_DeclencheSA( SortieUtilisee);
```

Pour arrêter l'émission, placer un autre Tbutton sur la fiche chantier et créer son gestionnaire de clic. Placer dans le gestionnaire l'instruction destinée à arrêter l'émission :

```
SP5_ArreteSA(SA1);
```

Exercice : émettre une tension en créneaux, en dents de scie...
A Tester avec un oscilloscope...

7. 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');  
    SP5_emetPLB(0,1);  
    SP5_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 := SP5_litPLB(4);  
    s1 := ' B4 : ' + intToStr(valBin);  
    memo1.lines.add(s1);  
    valBin := SP5_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.

8. Pour aller plus loin...

a. Les autres fonctions et procédures de SYSAM

A vous de tester et d'essayer les fonctions supplémentaires disponibles dans la ProgrammationSYSAM_SP5.dll... : synchronisation de l'acquisition...

b. Autres propriétés des composants

Pour améliorer l'aspect visuel 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. On peut aussi les fixer dans l'événement onCreate de la fiche principale (fiche chantier)

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 <---> U = +10 V
position = -100 <---> U = -10 V

ce qui se traduit par l'équation :

U = position/10 ;

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 U1 : single;

begin

{ calcul de la valeur de tension à émettre en fonction de la position du curseur}

U1 := Trackbar1.position/10 ;

{ affichage de la tension émise }

edit1.text := FloatToStrF(U1, ffGeneral, 3 , 2);

{ actualisation de l'affichage }

edit1.update;

{ émission }

SP5_EmissionUContinue(SA1 , U1);

end;

d. Utiliser les événements de composants pour régler les paramètres d'acquisition et d'émission.

En prolongement et généralisation de l'esprit du paragraphe précédent, pour créer une application utilisable par un public (non programmeur) scolaire ou autre, le réglage des paramètres d'acquisition ou d'émission doit être réalisable pendant l'utilisation du programme. Il faut donc dans un projet EAXO placer les composants adéquats et utiliser les événements associés, en clair faire de la véritable programmation événementielle.

e. Travailler avec un capteur

Pour utiliser un capteur, il faut commencer par déterminer son numéro d'identification, ensuite faire la mesure (acquisition) comme s'il n'y avait pas de capteur, puis convertir la valeur de tension en valeur de grandeur physique en utilisant les indications de conversion données sur la notice du capteur par Eurosmart.

Exemple du capteur de température : la notice donne
dans la rubrique « Fonction de transfert » : $\text{Température}(\text{°C}) = \text{Tension}(\text{V}) \times 25$
dans la rubrique « Etendue de mesure » : de -25°C à 125°C

On en déduit que les tensions extrêmes sont
 -1 V (pour -25°C) et $+5\text{V}$ pour ($+125\text{°C}$)
Il est judicieux d'utiliser le calibre $-/+5\text{V}$ de SYSAM pour une acquisition.

Il suffit d'ajouter dans le code (après la mesure de la VNE et le calcul de la tension U1 correspondante) l'instruction $\text{teta} = 25 * \text{U1}$, (après avoir déclaré la variable teta comme single) puis de faire afficher cette température dans le memo1 ou ailleurs pour disposer d'un thermomètre...

On perçoit ici la multiplicité des réalisations possibles...

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 suffisent pour créer des applications d'EXAO assez performantes et utilisables en classe avec des élèves, si on maîtrise LAZARUS et le langage pascal par ailleurs. On peut imaginer d'inclure dans une application simple le dessin progressif d'un graphe...

Si votre projet finalisé doit être distribué (installé sur plusieurs machines dans un but d'utilisation collective), il faut veiller à fournir aux utilisateurs la DLL « ProgrammationSYSAM_SP5.dll » et à ce qu'elle se trouve dans le dossier du programme exécutable que vous avez créé, ou dans le dossier « C:\Windows\System32 » des ordinateurs des utilisateurs. L'absence de la DLL plante le programme dès le démarrage, du fait du chargement statique de la DLL.

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.

30 août 2010

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