Notice de Programmation

———

SYSAM-SP5

5, rocade de la Croix St Georges
Bussy-St-Georges
77603 MARNE LA VALLEE CEDEX 3

TEL : 01 64 76 34 34
FAX : 01 64 76 34 39
WEB : http://www.eurosmart.fr

Table des matières :

5, rocade de la Croix St Georges
Bussy-St-Georges
77603 MARNE LA VALLEE CEDEX 3

TEL : 01 64 76 34 34
FAX : 01 64 76 34 39
WEB : http://www.eurosmart.fr

I. Présentation générale :

1. Introduction à la programmation de la SYSAM-SP5 :

Cette notice décrit les différents registres de la centrale d'acquisition SYSAM-SP5. Pour accéder à ces registres, il faut utiliser la DLL SP5.dll, installée automatiquement avec le pilote de l'interface SYSAM-SP5. Cette DLL est copiée dans le répertoire System32 de Windows.

Il s'agit d'une DLL type 32 bits, donc utilisable en programmation Windows 32 bits (Windows98/2000/XP). Elle est écrite avec la convention Pascal.

En plus de la DLL, l'unité uSP5.pas en Pascal (ou uSP5.c et uSP5.h en C), est installée en même temps que les exemples de programmation présents sur le CD accompagnant cette notice. Cette unité est conçue pour être liée au projet en cours.

Des exemples de programmation, en DELPHI et en C++, sont fournis sur ce même CD-ROM.

⇒ Chargement de la librairie

L'unité uSP5 contient la classe TSP5. La création d'une instance de cette classe entraîne le chargement de la DLL SP5.dll de la centrale. Cette DLL, copiée automatiquement lors de l'installation du driver, doit être située soit dans le même répertoire que l'exécutable, soit dans le répertoire system32 de Windows.

⇒ Mise à disposition des ressources

La création de l'instance précédemment décrite provoque également, en cas de réussite, l'affectation des diverses routines de configuration et de programmation de la carte. Ces routines sont décrites dans les paragraphes suivants.

L'utilisateur doit commencer par appeler la routine *Present* qui teste la présence du système. Si cette routine renvoie *TRUE*, les autres routines sont alors utilisables.

La SYSAM-SP5 possède des paramètres de calibration mémorisés. C'est pourquoi, il convient d'appeler la routine *RecupereCalibration* après avoir détecté la carte.

⇒ Déchargement de la librairie

Si la centrale ne doit plus être utilisée, il convient de libérer la mémoire qu'elle occupe en appelant le destructeur de l'instance TSP5.

2. Description de l'unité uSP5.pas :

TYPE DES ARGUMENTS UTILISES

- ⇒ *Byte* : un octet non signé (8 bits)
- ⇒ *Word* : un mot de deux octets non signés (16 bits)
- ⇒ *ShortInt* : un octet signé (8 bits)
- ⇒ *Smallint* : un mot de deux octets signés (16 bits)
- ⇒ *Integer* : un mot de quatre octets signés (32 bits)
- ⇒ *Double* : un réel double précision, codé sur 8 octets suivant le format Exposant/Mantisse (64 bits)
- ⇒ *Boolean* : un octet représentant un état logique TRUE ou FALSE (8 bits)

Note : La description présentée ici correspond à la convention Pascal.

Définition de la classe TSP5 :

Cette classe permet de s'interfacer avec la centrale SYSAM-SP5.

```
type
TSP5 = class
protected
  HandleDLL: THandle;
  FNomDLL: string;
  fCalibrageSP5: TCalibrageSP5;
  function GetCalibrageSP5: TCalibrageSP5;
  procedure LiaisonFonction(var aPointer: Pointer; aNomFonction: string);
  function ChargementDLL(aGenereException: Boolean = True): Boolean; virtual;
  procedure LiberationDLL;
  function GetRegistre(aAdr: BYTE): BYTE;
  procedure SetRegistre(aAdr: BYTE; const Value: BYTE);
  procedure TestChargementDLL;
public
  constructor Create; virtual;
  destructor Destroy; override;
  procedure RecupereCalibration;
  function Present: Boolean;
  procedure Reset;
  procedure ViderFifo;
  procedure TailleTransfertFifo(aTaille: Integer);
  function LireFifo(aPtrFifoSP5: TPtrFifoSP5): integer;
  procedure EcrireFifo(aPtrFifoSP5: TPtrFifoSP5; aSize: Integer);
  property CalibrageSP5: TCalibrageSP5 read GetCalibrageSP5;
  property NomDLL: string read FNomDLL write FNomDLL;
  property Registre[aAdr: BYTE]: BYTE read GetRegistre write SetRegistre;
end;
```

Description des fonctions membres :

constructor Create;
Description : L'utilisateur doit commencer par créer une instance de la classe TSP5 pour pouvoir s'interfacer avec la centrale SYSAM-SP5.

destructor Destroy;
Description : L'utilisateur doit détruire l'instance créée avec le constructeur après avoir fini d'utiliser la centrale SYSAM-SP5.

function Present: Boolean;
Description : Cette fonction informe si l'interface SYSAM-SP5 a été trouvée et que la DLL a pu être chargée.

Retour : Renvoie TRUE si l'interface SYSAM-SP5 a été trouvée et que la DLL a pu être chargée.

procedure Reset;
Description : Cette procédure permet de réinitialiser toutes les valeurs des registres de la SYSAM-SP5.

procedure ViderFifo;
Description : Cette procédure vide tous les points en mémoire.

Function TailleTransfertFifo(aTaille: Integer) : Integer;
Description : Cette procédure permet de fixer la taille des paquets de points transférés.

Paramètres : *aTaille* correspond au nombre de points à transférer par paquet. Elle doit être comprise entre 1 et 256 points. A noter que plus les paquets seront gros, plus l'acquisition pourra être rapide.

Retour : Renvoie la vraie taille de paquet.

function LireFifo(aPtrFifoSP5: TPtrFifoSP5): integer;
Description : Cette fonction permet de récupérer les points stockés en mémoire.

Paramètres : *aPtrFifoSP5* est un pointeur sur un enregistrement de type *TFifoSP5* décrit ci-dessous.

Retour : Renvoie la taille du tableau *TFifoSP5* situé à l'adresse *aPtrFifoSP5*.

```
TFifoSP5 = array[0..255] of WORD;   {Tableau de 256 points}
TPtrFifoSP5 = ^TFifoSP5;            {Pointeur sur un tableau TFifoSP5}
```

procedure EcrireFifo(aPtrFifoSP5: TPtrFifoSP5; aSize: Integer);
Description : Cette procédure permet de charger la mémoire de la carte avec un tableau de points.

Paramètres : *aPtrFifoSP5* pointe sur le tableau de points à stocker dans la FIFO de la centrale SYSAM-SP5.
aSize spécifie la taille du tableau pointé par *aPtrFifoSP5* utilisé pour SA.

property CalibrageSP5: TCalibrageSP5 read GetCalibrageSP5;
Description : Cette propriété permet de lire les paramètres de calibration usine de la centrale SYSAM-SP5.

Retour : Renvoie un enregistrement de type *TCalibrageSP5* décrit ci-dessous.

```
type
  TCalibrageSP5 = record
  SerialNumber: string[12];           {Numéro de série de la centrale}
  Ampli_CAN: array[0..3] of double;   {Valeur d'amplification de chaque CAN}
  Ampli_DAC: array[0..1] of double;   {Valeur d'amplification de chaque DAC}
  Offset_CAN: array[0..3] of double;  {Valeur d'offset de chaque CAN}
  Offset_DAC: array[0..1] of double;  {Valeur d'offset de chaque DAC}
  DateEtalonnage: TDateTime;          {Date d'étalonnage de la centrale}
end;
TPrtCalibrageSP5 = ^TCalibrageSP5;  {Pointeur sur un enregistrement de type
                                     TCalibrageSP5}
```

property NomDLL: string read FNomDLL write FNomDLL;
Description : Cette propriété permet de lire ou de modifier le nom de la DLL à charger.

Entrée / Sortie : Permet de lire ou de modifier le nom de la DLL à charger en mémoire (par défaut, SP5.dll).

property Registre[aAdr: BYTE]: BYTE read GetRegistre write SetRegistre;
Description : Cette propriété permet d'accéder en lecture/écriture aux registres de la centrale SYSAM-SP5.

Paramètres : *aAdr* correspond à l'adresse du registre à lire ou à écrire.

Entrée/Sortie : Valeur du registre lue ou à écrire.

3. Programmation d'un registre :

Une fois une instance de TSP5 créée, les registres se programment grâce à la propriété **Registre**.

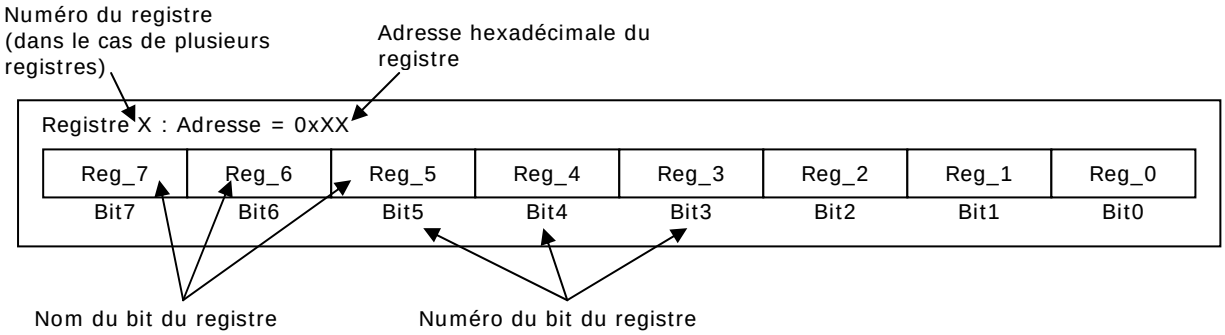
```
var InterfaceSP5:TSP5;

InterfaceSP5 := TSP5.Create;

if InterfaceSP5.Present then
begin
  InterfaceSP5.Registre[$XX] := $YY;
  //Corps du Programme
end;
```

Où $XX représente l'adresse à programmer et $YY la valeur à affecter.

4. Présentation d'un registre :



II. Entrées Analogiques :

1. Présentation :

Ce bloc permet de paramétrer l'acquisition sur les entrées analogiques :

- ➔ Choix des entrées à acquérir
- ➔ Choix du calibre de chaque entrée
- ➔ Choix du nombre de points par voie
- ➔ Configuration en mode simple ou différentiel
- ➔ Choix du temps d'échantillonnage

2. Détails des Registres :

- **EA_Active : 1 registre**

Détermine les entrées analogiques à activer ou non (le bit 0 correspondant à l'entrée EA0 et le bit 7 à l'entrée EA7). Pour activer une entrée, mettre le bit correspondant à 1.

Cette valeur est un entier 8 bits occupant donc un registre.

Registre 0 : Adresse = 0x00

EAA_7	EAA_6	EAA_5	EAA_4	EAA_3	EAA_2	EAA_1	EAA_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour activer les entrées EA1 et EA5, écrire la valeur 0x22 à l'adresse 0x00.

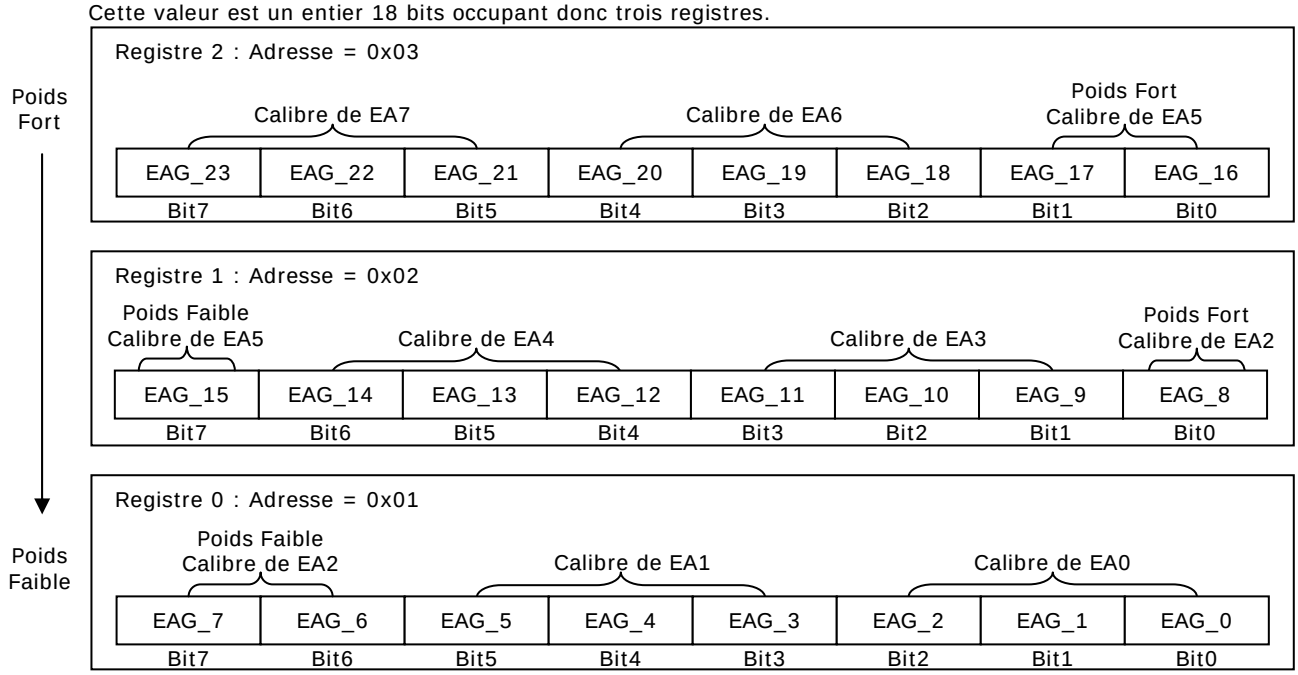
- **EA_Calibre : 3 registres**

Permet de définir le calibre propre à chaque entrée analogique.

Le calibre correspondant à chaque entrée est codé sur 3 bits tels que :

000	±10 V
001	± 2 V
010	± 0,2 V
100	± 5 V
101	± 1 V
110	± 0,1 V

Cette valeur est un entier 18 bits occupant donc trois registres.



Exemple :

Pour mettre le calibre ± 5 V sur les entrées *EA1 et EA5, et mettre le calibre* ± 10 V sur toutes les autres entrées :

000 correspond au calibre ± 10 V
100 correspond au calibre ± 5 V

Entrée 7 : 000
Entrée 6 : 000
Entrée 5 : 100
Entrée 4 : 000
Entrée 3 : 000
Entrée 2 : 000
Entrée 1 : 100
Entrée 0 : 000

Soit :
0000 0010 0000 0000 0010 0000

Donc, il convient d'écrire
la valeur 0x20 à l'adresse 0x01
la valeur 0x00 à l'adresse 0x02
la valeur 0x02 à l'adresse 0x03

- **EA_Nb_Skip : 5 registres**

Défini (en nombre de coups d'horloge) le temps d'inhibition entre chaque point valide pour qu'un autre point soit acquis.

Le temps d'inhibition est un multiple de 100 ns.

Cette valeur est un entier 40 bits occupant donc cinq registres.

Poids Fort

Registre 4 : Adresse = 0x08							
EAS_39	EAS_38	EAS_37	EAS_36	EAS_35	EAS_34	EAS_33	EAS_32
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 3 : Adresse = 0x07							
EAS_31	EAS_30	EAS_29	EAS_28	EAS_27	EAS_26	EAS_25	EAS_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x06							
EAS_23	EAS_22	EAS_21	EAS_20	EAS_19	EAS_18	EAS_17	EAS_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x05							
EAS_15	EAS_14	EAS_13	EAS_12	EAS_11	EAS_10	EAS_9	EAS_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x04							
EAS_7	EAS_6	EAS_5	EAS_4	EAS_3	EAS_2	EAS_1	EAS_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Si on veut un temps d'échantillonnage de 1 min, soit 599 999 999 x 100 ns, il faudra entrer la valeur

600 000 000 ⇔ 0x0023C345FF

Soit :
Ecrire la valeur 0x00 à l'adresse 0x08
Ecrire la valeur 0x23 à l'adresse 0x07
Ecrire la valeur 0xC3 à l'adresse 0x06
Ecrire la valeur 0x45 à l'adresse 0x05
Ecrire la valeur 0xFF à l'adresse 0x04

- **Nb_NbPoint : 3 registres**

Défini le nombre de points à acquérir par entrée.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x0B							
x	x	x	x	x	x	EAP_17	EAP_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x0A							
EAP_15	EAP_14	EAP_13	EAP_12	EAP_11	EAP_10	EAP_9	EAP_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x09							
EAP_7	EAP_6	EAP_5	EAP_4	EAP_3	EAP_2	EAP_1	EAP_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Si on veut obtenir 200 points par voie, il faut écrire :

200 ⇔ 0xC8

D'où :

Ecrire la valeur 0xC8 à l'adresse 0x09
Ecrire la valeur 0x00 à l'adresse 0x0A
Ecrire la valeur 0x00 à l'adresse 0x0B

- **Mode Différentiel : 1 registre**

Permet de déterminer les CAN en mode différentiel.

Cette valeur est codée sur 4 bits, et occupe donc 1 registre.
Pour activer ou non un CAN en mode différentiel, activer le bit correspondant.

Registre 0 : Adresse = 0x0C							
x	x	x	x	EAD_3	EAD_2	EAD_1	EAD_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour mettre les CAN 0 et 1 en mode différentiel, il faut écrire :

0011 soit 0x03 à l'adresse 0x0C.

3. Exemples :

EXEMPLE 1 :

On veut acquérir sur EA1 et EA6, tel que le temps d'échantillonnage soit de 10 ms pendant 10 s, avec un calibre de ± 1V.

1. Ecriture du registre pour les voies actives :

   ⇨ **Ecriture de 01000010 à l'adresse 0x00**

2. Ecriture du registre pour les calibres :

   000 011 000 000 000 000 011 000,
   soit 0x0C0018.

   ⇨ **Ecriture de 0x0C** (Poids Fort) **à l'adresse 0x03**
   ⇨ **Ecriture de 0x00 à l'adresse 0x02**
   ⇨ **Ecriture de 0x18** (Poids Faible) **à l'adresse 0x01**

3. Ecriture du registre pour le nombre de skip :

   On veut que le temps d'échantillonnage soit de 10 ms, un coup d'horloge durant 100 ns, il faut donc mettre 99.999 comme valeur dans Nb_Skip, soit 0x1869F.

   ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x08**
   ⇨ **Ecriture de 0x05 à l'adresse 0x07**
   ⇨ **Ecriture de 0x01 à l'adresse 0x06**
   ⇨ **Ecriture de 0x86 à l'adresse 0x05**
   ⇨ **Ecriture de 0x9F** (Poids Faible) **à l'adresse 0xA4**

4. Ecriture du registre pour le nombre de points :

   $$\text{Nombre de points} = \frac{10 \times 10^3}{10} = 1000 = 0x0003E8$$

   ⇨ **Ecriture de 0x00 à l'adresse 0x0B**
   ⇨ **Ecriture de 0x03 à l'adresse 0x0A**
   ⇨ **Ecriture de 0xE8 à l'adresse 0x09**

5. Ecriture du registre pour mode différentiel :

   Ici, on ne souhaite pas travailler en mode différentiel

   ⇨ **Ecriture de 0x00 à l'adresse 0x0C**

EXEMPLE 2 :

On veut acquérir en mode différentiel sur le CAN1 (Voie EA4 et EA5), tel que le temps d'échantillonnage soit de 25 ms pendant 5 s, avec un calibre de ± 1V.

1. Ecriture du registre pour les voies actives :

   ⇨ **Ecriture de 00010000 à l'adresse 0x00**

2. Ecriture du registre pour les calibres :

   000 000 000 011 000 000 000 000,
   soit 0x3000.

   ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x03**
   ⇨ **Ecriture de 0x30 à l'adresse 0x02**
   ⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x01**

3. Ecriture du registre pour le nombre de skip :

   On veut que le temps d'échantillonnage soit de 25 ms, un coup d'horloge durant 100 ns, il faut donc mettre 250.000 comme valeur dans Nb_Skip, soit 0x3D090.

   ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x00**
   ⇨ **Ecriture de 0x05 à l'adresse 0x00**
   ⇨ **Ecriture de 0xF5 à l'adresse 0x03**
   ⇨ **Ecriture de 0xE1 à l'adresse 0xD0**
   ⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x90**

4. Ecriture du registre pour le nombre de points :

   $$\text{Nombre de points} = \frac{50 \times 10^3}{25} = 2000 = 0x0007D0$$

   ⇨ **Ecriture de 0x00 à l'adresse 0x0B**
   ⇨ **Ecriture de 0x07 à l'adresse 0x0A**
   ⇨ **Ecriture de 0xD0 à l'adresse 0x09**

5. Ecriture du registre pour mode différentiel :

   Ici, on ne souhaite pas travailler en mode différentiel

   ⇨ **Ecriture de 0x02 à l'adresse 0x0C**

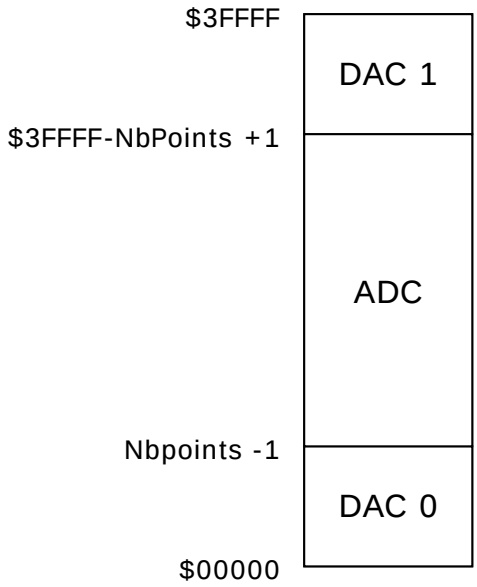
III. Sorties Analogiques :

1. Présentation :

Cette partie permet de configurer les 2 sorties grâce aux différents registres :

- ➔ DAC0_NbPoint : Défini le nombre de points à utiliser pour la première sortie
- ➔ DAC1_NbPoint : Défini le nombre de points à utiliser pour la seconde sortie
- ➔ DAC0_NbSkip : Défini le temps (en nombre de coups d'horloge) entre l'émission de deux points sur le DAC 0.
- ➔ DAC1_NbSkip : Défini le temps (en nombre de coups d'horloge) entre l'émission de deux points sur le DAC 1.
- ➔ DAC_MonoCoup : Permet de n'émettre sur la sortie SA1 ou SA2 qu'une salve à la fois.
- ➔ AccesDirectRam : configuration de l'accès à la RAM
- ➔ AccesDirectRAM_ADR_Debut : Adresse de début de remplissage de la RAM
- ➔ AccesDirectRAM_ADR_Fin : Adresse de fin de remplissage de la RAM

Gestion de la RAM :



2. Détails des Registres :

- **DAC_MonoCoup : 2 bits**

Permet de ne pas reboucler l'émission en continu. Après chaque émission dans ce mode, il est nécessaire de faire un STOP puis un START pour lancer une nouvelle salve sur la sortie.

Registre 0 : Adresse = 0x4C

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
x	x	x	x	x	x	D1MC	D0MC

- **DAC0_NbPoint (SA1) : 3 registres**

Défini le nombre de points à émettre sur la première sortie.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x0F

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
x	x	x	x	x	x	DA0P_17	DA0P_16

Registre 1 : Adresse = 0x0E

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DA0P_15	DA0P_14	DA0P_13	DA0P_12	DA0P_11	DA0P_10	DA0P_9	DA0P_8

Registre 0 : Adresse = 0x0D

Poids Faible

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DA0P_7	DA0P_6	DA0P_5	DA0P_4	DA0P_3	DA0P_2	DA0P_1	DA0P_0

Exemple :

Pour émettre 100 points sur SA1 :
Ecrire la valeur 0x00 à l'adresse 0x0F
Ecrire la valeur 0x00 à l'adresse 0x0E
Ecrire la valeur 0x64 à l'adresse 0x0D

- **DAC1_NbPoint (SA2) : 3 registres**

Défini le nombre de points à émettre sur la seconde sortie.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x12

x	x	x	x	x	x	DA1P_17	DA1P_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x11

DA1P_15	DA1P_14	DA1P_13	DA1P_12	DA1P_11	DA1P_10	DA1P_9	DA1P_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x10

DA1P_7	DA1P_6	DA1P_5	DA1P_4	DA1P_3	DA1P_2	DA1P_1	DA1P_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour émettre 10000 points sur SA2 :
Ecrire la valeur 0x00 à l'adresse 0x12
Ecrire la valeur 0x27 à l'adresse 0x11
Ecrire la valeur 0x10 à l'adresse 0x10

- **DAC0_NbSkip : 5 registres**

Défini (en nombre de coups d'horloge) le temps d'inhibition après l'émission d'un point valide pour qu'un autre point soit émis sur la première sortie.

Le temps d'inhibition est un multiple de 200 ns.

Cette valeur est un entier 40 bits occupant donc cinq registres.

Poids Fort

Registre 4 : Adresse = 0x17							
DA0S_39	DA0S_38	DA0S_37	DA0S_36	DA0S_35	DA0S_34	DA0S_33	DA0S_32
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 3 : Adresse = 0x16							
DA0S_31	DA0S_30	DA0S_29	DA0S_28	DA0S_27	DA0S_26	DA0S_25	DA0S_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x15							
DA0S_23	DA0S_22	DA0S_21	DA0S_20	DA0S_19	DA0S_18	DA0S_17	DA0S_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x14							
DA0S_15	DA0S_14	DA0S_13	DA0S_12	DA0S_11	DA0S_10	DA0S_9	DA0S_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x13							
DA0S_7	DA0S_6	DA0S_5	DA0S_4	DA0S_3	DA0S_2	DA0S_1	DA0S_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour émettre 1 point toutes les secondes, il faut donc attendre 4999999 de fois 200 ns ⇨ 0x00004C4B3F

Ecrire la valeur 0x00 à l'adresse 0x17
Ecrire la valeur 0x00 à l'adresse 0x16
Ecrire la valeur 0x4C à l'adresse 0x15
Ecrire la valeur 0x4B à l'adresse 0x14
Ecrire la valeur 0x3F à l'adresse 0x13

- **DAC1_NbSkip : 5 registres**

Défini (en nombre de coups d'horloge) le temps d'inhibition après l'émission d'un point valide pour qu'un autre point soit émis sur la seconde sortie.

Le temps d'inhibition est un multiple de 200 ns.

Cette valeur est un entier 40 bits occupant donc cinq registres.

Poids Fort

Registre 4 : Adresse = 0x1C

DA1S_39	DA1S_38	DA1S_37	DA1S_36	DA1S_35	DA1S_34	DA1S_33	DA1S_32
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 3 : Adresse = 0x1B

DA1S_31	DA1S_30	DA1S_29	DA1S_28	DA1S_27	DA1S_26	DA1S_25	DA1S_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x1A

DA1S_23	DA1S_22	DA1S_21	DA1S_20	DA1S_19	DA1S_18	DA1S_17	DA1S_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x19

DA1S_15	DA1S_14	DA1S_13	DA1S_12	DA1S_11	DA1S_10	DA1S_9	DA1S_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x18

DA1S_7	DA1S_6	DA1S_5	DA1S_4	DA1S_3	DA1S_2	DA1S_1	DA1S_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Exemple :

Pour émettre un point par milliseconde, il faut donc attendre 2499999 fois 200 ns ⇔ 0x000026259F

Ecrire la valeur 0x00 à l'adresse 0x1C
Ecrire la valeur 0x00 à l'adresse 0x1B
Ecrire la valeur 0x26 à l'adresse 0x1A
Ecrire la valeur 0x25 à l'adresse 0x19
Ecrire la valeur 0x9F à l'adresse 0x18

- **AccesDirectRAM_Actif : 1 registre**

Défini la configuration de l'accès à la RAM.

Cette valeur nécessite un bit et occupe donc un registre.

Registre 0 : Adresse = 0x45

x	x	x	x	x	x	x	ADRA_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

ADRA_0 : (AccesDirectRam_Actif) : Mettre ce bit à 1 pour activer le remplissage de la RAM

- **AccesDirectRAM_ADR_Debut : 3 registres**

Défini l'adresse de début de remplissage de la RAM.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x48

x	x	x	x	x	x	ADRD_17	ADRD_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x47

ADRD_15	ADRD_14	ADRD_13	ADRD_12	ADRD_11	ADRD_10	ADRD_9	ADRD_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x46

ADRD_7	ADRD_6	ADRD_5	ADRD_4	ADRD_3	ADRD_2	ADRD_1	ADRD_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- **AccesDirectRAM_ADR_Fin : 3 registres**

Défini l'adresse de fin de remplissage de la RAM.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x4B

x	x	x	x	x	x	ADRF_17	ADRF_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x4A

ADRF_15	ADRF_14	ADRF_13	ADRF_12	ADRF_11	ADRF_10	ADRF_9	ADRF_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

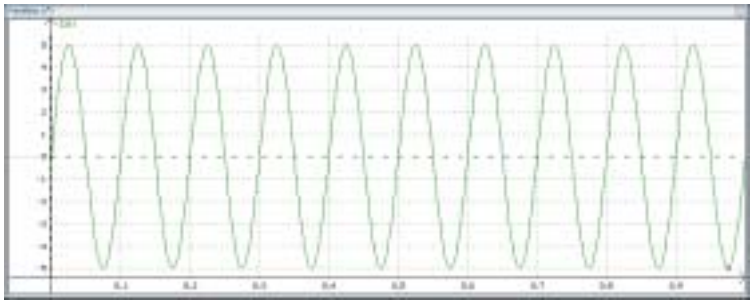
Registre 0 : Adresse = 0x49

ADRF_7	ADRF_6	ADRF_5	ADRF_4	ADRF_3	ADRF_2	ADRF_1	ADRF_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

3. Exemples :

EXEMPLE :

On veut émettre sur SA1 une tension sinusoïdale d'amplitude 5V, ayant une période de 100ms, sur 10 périodes, totalisant 1000 points.



1. Ecriture du registre pour le nombre de points :

   1000 points :
   - ⇨ **Ecriture de 0x00 à l'adresse 0x0F**
   - ⇨ **Ecriture de 0x03 à l'adresse 0x0E**
   - ⇨ **Ecriture de 0xE8 à l'adresse 0x0D**

2. Ecriture du registre pour le nombre de skip :

   Le temps d'échantillonnage est :

   $\frac{0.1 \times 10}{1000} = 0.001s = 1ms.$

   On veut que le temps d'échantillonnage soit de 1ms, un coup d'horloge durant 200ns, il faut donc mettre 4999 comme valeur dans Nb_Skip, soit 0x00001387.

   - ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x17**
   - ⇨ **Ecriture de 0x00 à l'adresse 0x16**
   - ⇨ **Ecriture de 0x00 à l'adresse 0x15**
   - ⇨ **Ecriture de 0x13 à l'adresse 0x14**
   - ⇨ **Ecriture de 0x87** (Poids Faible) **à l'adresse 0x13**

3. Ecriture de l'adresse de début de remplissage de la RAM pour le DAC0 :

   - ⇨ **Ecriture de 0x00 à l'adresse 0x48**
   - ⇨ **Ecriture de 0x00 à l'adresse 0x47**
   - ⇨ **Ecriture de 0x00 à l'adresse 0x46**

4. Ecriture de l'adresse de fin de remplissage de la RAM pour le DAC0 :

   On veut émettre 1000 points, d'où, une adresse de fin de 1000-1, soit 999 => 0x03E7
   - ⇨ **Ecriture de 0x00 à l'adresse 0x4B**
   - ⇨ **Ecriture de 0x03 à l'adresse 0x4A**
   - ⇨ **Ecriture de 0xE7 à l'adresse 0x49**

5. Activation du remplissage de la RAM :

   - ⇨ **Ecriture de 0x01 à l'adresse 0x45**

6. Remplissage de la FIFO :

```
Var SP5 :TSP5 ;
      lFifo: TFifo;
      lSize,i: Integer ;
begin
  lSize := 1000 ;
  for i := 0 to lSize -1 do
     lFifo[i] := (5*Sin((0+0.001*i)*40*pi/2));
  RemplirFifo(@lFifo,1000) ;
end ;
```

7. Désactivation du remplissage de la RAM :

⇨ **Ecriture de 0x00 à l'adresse 0x45**

IV. Triger / PréTriger :

1. Présentation :

Ce bloc permet de configurer le triger et le prétriger :

- ➔ Type de synchronisation (externe, sur seuil de l'une des voies actives)
- ➔ Type de front à utiliser pour le déclenchement (montant ou descendant)
- ➔ Numéro de la voie en cas d'acquisition sur seuil
- ➔ Valeur du seuil de déclenchement
- ➔ Taille de l'hystérésis dans le cas d'un déclenchement par seuil
- ➔ Nombre de points à garder avant le déclenchement
- ➔ Possibilité de synchroniser même si le nombre de points en prétrig n'est pas atteint.
- ➔ Stockage de la donnée lue avant trig
- ➔ Stockage de la donnée lue après trig
- ➔ Nombre de coup d'horloge entre le dernier point acquis et le trig externe

2. Détails des Registres :

- **Paramétrage du Triger-Pretriger : 1 registre**

Registre 0 : Adresse = 0x1D

x	TNM_2	TNM_1	TNM_0	PTS_0	TFM_0	TS_0	TSX_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Trig_SynchroExt (Bit 0) : Mettre ce bit à 1 pour activer le déclenchement sur synchro externe.

Trig_Seuil (Bit 1) : Mettre ce bit à 1 pour activer le déclenchement sur seuil (résultat d'une des voies acquises)

Trig_FrontMontant (Bit 2) : Mettre ce bit à 1 pour effectuer le déclenchement sur front montant.

PreTrig_Souple (Bit 3) : Mettre ce bit à 1 pour permettre à la centrale de triger même si le nombre de points en pretrig n'a pas été atteint.

Trig_NumeroVoie (Bit 4 -> 6) : Numéro de la voie sur lequel s'effectue le déclenchement (000 : Voie 0 ; 110 : Voie 6)

Exemple :

Pour réaliser un déclenchement sur EA3 sur front montant à la valeur avec un pretrig partiel possible :

0110110

Ecrire la valeur 0x36 à l'adresse 0x1D

- **Trig_Seuil_Valeur : 2 registres**

Défini le seuil de déclenchement (en valeur entière).

Cette valeur est un entier 12 bits occupant donc deux registres.

Poids Fort

Registre 1 : Adresse = 0x1F							
x	x	x	x	TSV_11	TSV_10	TSV_9	TSV_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x1E							
TSV_7	TSV_6	TSV_5	TSV_4	TSV_3	TSV_2	TSV_1	TSV_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour un déclenchement sur seuil correspondant à la valeur entière 1976 :
Ecrire la valeur 0x07 à l'adresse 0x1F
Ecrire la valeur 0xB8 à l'adresse 0x1E

- **Pretrig_Nb_Point : 3 registres**

Défini le nombre de points gardé avant le déclenchement.

Cette valeur est un entier 18 bits occupant donc trois registres.

Poids Fort

Registre 2 : Adresse = 0x22							
x	x	x	x	x	x	PTN_17	PTN_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x21							
PTN_15	PTN_14	PTN_13	PTN_12	PTN_11	PTN_10	PTN_9	PTN_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x20							
PTN_7	PTN_6	PTN_5	PTN_4	PTN_3	PTN_2	PTN_1	PTN_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour obtenir 500 points avant le déclenchement :

Ecrire la valeur 0x00 à l'adresse 0x22
Ecrire la valeur 0x01 à l'adresse 0x21
Ecrire la valeur 0xF4 à l'adresse 0x20

- **Trig_Seuil_Hysteresis : 1 registre**

Défini la valeur de l'hystérésis du triger en valeur entière.

Cette valeur est un entier 8 bits occupant donc un registre.

Registre 0 : Adresse = 0x23

TSH_7	TSH_6	TSH_5	TSH_4	TSH_3	TSH_2	TSH_1	TSH_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Exemple :

Pour avoir une valeur d'hystérésis de 100 :

Ecrire la valeur 0x64 à l'adresse 0x23

- **S_DATA_PreTrig : 2 registres**

Permet de lire la donnée précédant le trig, utilisée avec *S_DATA_PostTrig*, elle permet de re-synchroniser la courbe acquise.

Poids Fort ↓ Poids Faible

Registre 1 : Adresse = 0x6C

SDPreT_11	SDPreT_10	SDPreT_9	SDPreT_8	SDPreT_7	SDPreT_6	SDPreT_5	SDPreT_4
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x6B

SDPreT_3	SDPreT_2	SDPreT_1	SDPreT_0	x	x	x	x
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- **S_DATA_PostTrig : 2 registres**

Permet de lire la donnée suivant le trig, utilisée avec *S_DATA_PreTrig*, elle permet de re-synchroniser la courbe acquise.

Poids Fort ↓ Poids Faible

Registre 1 : Adresse = 0x6E

SDPosT_11	SDPosT_10	SDPosT_9	SDPosT_8	SDPosT_7	SDPosT_6	SDPosT_5	SDPosT_4
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

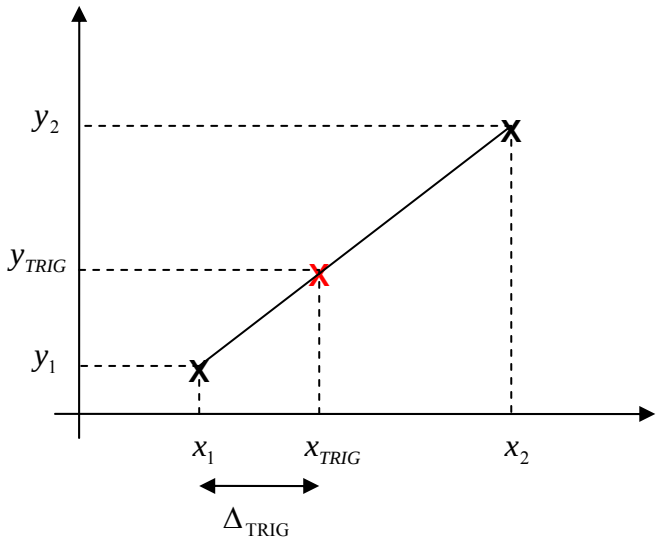
Registre 0 : Adresse = 0x6D

SDPosT_3	SDPosT_2	SDPosT_1	SDPosT_0	x	x	x	x
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Utilisation de *S_DATA_PreTrig* et de *S_DATA_PostTrig* :

- Dans le cas d'une utilisation en mode analogique (avec déclenchement sur seuil) :

  Ces deux registres permettent d'extrapoler la valeur de x_{TRIG} .



(x_1 , y_1) : Point acquis juste avant le trig. y_1 est la valeur stockée dans *S_DATA_PreTrig.*

(x_2 , y_2) : Point acquis juste après le trig. y_2 est la valeur stockée dans *S_DATA_PostTrig.*

y_{TRIG} est la valeur de seuil de déclenchement.

x_2 - x_1 = 1 Te

La droite ainsi tracée a pour équation : y = ax + b

a = $\frac{y_2 - y_1}{x_2 - x_1}$ avec $x_2 - x_1$ = 1 Te

On prend x_1 = 0 , et on obtient :

b = y_1

$$\Delta_{TRIG} = \frac{y_{TRIG} - b}{a} = \frac{y_{TRIG} - y_1}{\frac{y_2 - y_1}{Te}}$$

5, rocade de la Croix St Georges
Bussy-St-Georges
77603 MARNE LA VALLEE CEDEX 3

TEL : 01 64 76 34 34
FAX : 01 64 76 34 39
WEB : http://www.eurosmart.fr

- Dans le cas d'une utilisation en mode numérique (avec déclenchement sur Synchro Externe) :

  S_ Delta_TrigExterne mémorise le nombre de coups d'horloge (cadencée à 80 MHz) entre le point précédent le trig et le trig. De là, il est simple d'en déduire le Δ_{TRIG} .

  $\Delta_{TRIG} = \frac{S\_\ Delta\_TrigExterne}{80e^6}$ (en seconde).

- **S_ Delta_TrigExterne : 4 registres**

Retourne le nombre de coup d'horloge (80 MHz) entre le dernier point acquis avant le trig et le trig

Poids Fort

Registre 3 : Adresse = 0x72

SDTExt_31	SDTExt_30	SDTExt_29	SDTExt_28	SDTExt_27	SDTExt_26	SDTExt_25	SDTExt_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x71

SDTExt_23	SDTExt_22	SDTExt_21	SDTExt_20	SDTExt_19	SDTExt_18	SDTExt_17	SDTExt_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x70

SDTExt_15	SDTExt_14	SDTExt_13	SDTExt_12	SDTExt_11	SDTExt_10	SDTExt_9	SDTExt_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

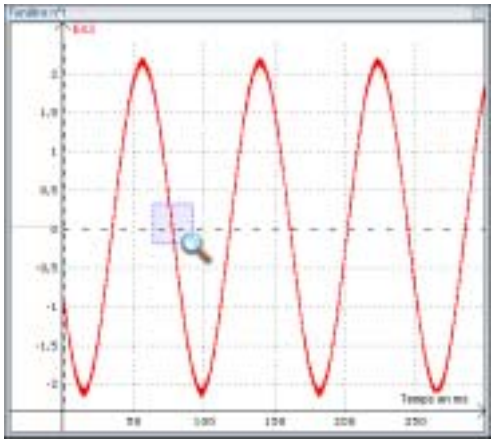
Registre 2 : Adresse = 0x6F

SDTExt_7	SDTExt_6	SDTExt_5	SDTExt_4	SDTExt_3	SDTExt_2	SDTExt_1	SDTExt_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

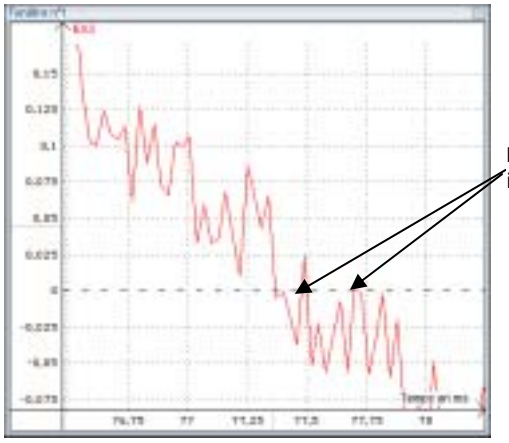
3.Exemples :

EXEMPLE 1 :

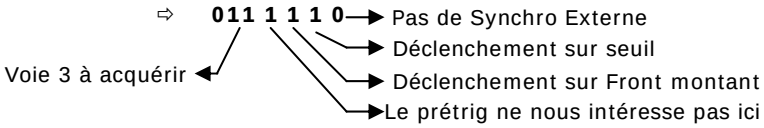
On veut effectuer l'acquisition sur EA3 d'un signal sinusoïdal bruité de fréquence = 110 Hz et de Valeur efficace = 1.5V. On veut synchroniser ce signal sur front montant à 0 V.



Le zoom ci-dessous montre que le bruit présent sur la courbe va nécessiter une hystérésis.



1. Ecriture du registre de paramétrage :



⇨ **Ecriture de 0x3E à l'adresse 0x1D**

2. Ecriture du registre pour la valeur de seuil de déclenchement :

⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x1F**
⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x1E**

3. Ecriture des registres de Prétrig :

Ici, le prétrig ne nous intéresse pas :

⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x22**
⇨ **Ecriture de 0x00 à l'adresse 0x21**
⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x20**

4. Ecriture du registre pour la taille de l'hystérésis :

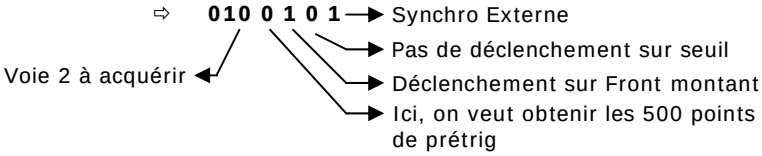
L'acquisition est ici réalisée sans capteur avec un calibre de +/- 10V. Si on veut une hystérésis de 1V : 4096/20 ≈ 205.

⇨ **Ecriture de 0xCD à l'adresse 0x23**

EXEMPLE 2 :

On veut réaliser une acquisition sur EA2 avec synchronisation externe avec un prétrig de 500 points.

➔ Ecriture du registre de paramétrage :



⇨ **Ecriture de 0x25 à l'adresse 0x1D**

➔ Ecriture du registre pour la valeur de seuil de déclenchement (valeur ignorée dans ce mode) :

⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x1F**
⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x1E**

➔ Ecriture des registres de Prétrig :
Ici, on veut un prétrig de 500 points :

⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x22**
⇨ **Ecriture de 0x01 à l'adresse 0x21**
⇨ **Ecriture de 0xF4** (Poids Faible) **à l'adresse 0x20**

➔ Ecriture du registre pour la taille de l'hystérésis :

L'hystérésis n'est pas applicable dans le cas d'une synchronisation externe.

⇨ **Ecriture de 0x00 à l'adresse 0x23**

V. Mise en route des modules :

1. Présentation :

Ce bloc permet de mettre en fonctionnement les différents modules de la centrale :

- ➔ Lancement de l'acquisition
- ➔ Fonctionnement du DAC 0
- ➔ Fonctionnement du DAC 1
- ➔ Fonctionnement du Timer
- ➔ Lancement du mode lecture directe
- ➔ Latch des entrées pour le mode lecture directe
- ➔ Remise à zéro du Timer
- ➔ Acquisition en mode permanent

2. Détails du Registre :

Registre 0 : Adresse = 0x24

MMMP_0	MMZT_0	MML_0	MMC_0	MMT_0	MMD1_0	MMD0_0	MMS_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Start (Bit 0) :	Lance l'acquisition sur front montant de ce bit.
Dac0_Actif (Bit 1) :	DAC 0 (SA1) en fonctionnement
Dac1_Actif (Bit 2) :	DAC 1 (SA2) en fonctionnement
Run_Timer (Bit 3) :	Timer en fonctionnement
LectureDirecte (Bit 4) :	Permet d'effectuer une lecture directe des valeurs lues, sans passer par la FIFO.
Latch_Lecture_Directe (Bit 5) :	Permet de latcher les entrées pour le mode lecture directe.
RAZ_Timer (Bit 6) :	Remise à zéro du Timer
ModePermanent (Bit 7) :	Acquisition en mode permanent.

Exemple :

Pour mettre en fonctionnement le DAC0, le DAC1, le Timer, les leds et lancer l'acquisition :

Ecrire la valeur 0x4F à l'adresse 0x24

VI. Flag :

1. Présentation :

Ce bloc permet d'obtenir certaines informations sur la centrale :

- ➔ Dépassement du Timer
- ➔ Fin du Timer
- ➔ Acquisition possible à 10 MHz

2. Détails du Registre :

Registre 0 : Adresse = 0x25

x	x	x	x	x	FA10_0	FRT_0	FOF_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Overflow (Bit 0) : Indique un dépassement du Timer.

Resultat_Timer_OK (Bit 1) : Indique la fin du Timer

Acquisition10MHz (Bit 2) : Indique que l'acquisition peut s'effectuer à 10 MHz.

Exemple :

Lecture de la valeur 0x04

Le Timer n'est ni en dépassement, ni terminé. L'acquisition à 10 MHz est possible.

VII. Timer :

1. Présentation :

Ce bloc permet d'avoir :

- Un chronomètre
- Un compteur

2. Détails des Registres :

- **Configuration : 1 registre**

Registre 0 : Adresse = 0x26

SelIO_4	SelIO_3	SelIO_2	SelIO_1	SelIO_0	FallingT_0	RisingT_0	MCPT_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0 : M_CPT (Mode_CPT)
Permet de définir si on utilise le Timer en chronomètre ou en compteur.
0 → Mode Chronomètre
1 → Mode Compteur

Bit1 : Rising_T (Rising_Timer)
Mode Chronomètre :
Sélectionne comment est démarré le chronomètre.
0 → ↘ (Départ sur Front Descendant)
1 → ↗ (Départ sur Front Montant)
Mode Compteur :
Sélectionne si les fronts montants (↗) sont comptés.
0 → Non
1 → Oui

Bit2 : Falling_T (Falling_Timer)
Mode Chronomètre :
Sélectionne comment est arrêté le chronomètre.
0 → ↗ (Arrêt sur Front Montant)
1 → ↘ (Arrêt sur Front Descendant)
Mode Compteur :
Sélectionne si les fronts descendants (↘) sont comptés.
0 → Non
1 → Oui

Bit 7->3 : SelIO (Select_IO_Timer)
Détermine l'entrée du timer (PortB, PortC ou Entrée Chrono)

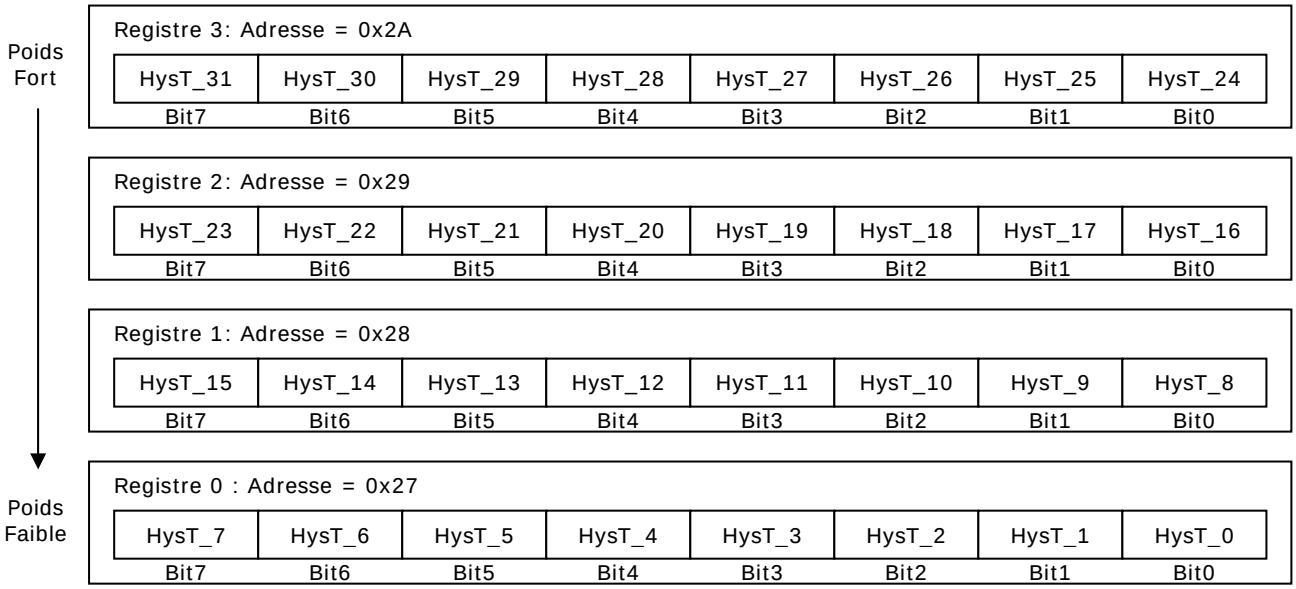
Valeur	Entrée Timer
00000	Bit 0 du port B
00001	Bit 1 du port B
00010	Bit 2 du port B
00011	Bit 3 du port B
00100	Bit 4 du port B
00101	Bit 5 du port B
00110	Bit 6 du port B
00111	Bit 7 du port B
11111	Entrée Chrono

Valeur	Entrée Timer
01000	Bit 0 du port C
01001	Bit 1 du port C
01010	Bit 2 du port C
01011	Bit 3 du port C
01100	Bit 4 du port C
01101	Bit 5 du port C
01110	Bit 6 du port C
01111	Bit 7 du port C

- **Hysteresis_Timer : 4 registres**

Défini (en nombre de coups d'horloge) le temps d'inhibition après un front valide pour qu'un autre front soit valide.

Cette valeur est un entier 32 bits occupant donc quatre registres.



- **TFenetre_CPT : 6 registres**

Uniquement utilisé avec le compteur.
Nombre de coups d'horloge pendant lequel on compte les évènements.

Cette valeur est un entier 48 bits occupant donc six registres.

Poids Fort

Registre 0 : Adresse = 0x30

TFEN_47	TFEN_46	TFEN_45	TFEN_44	TFEN_43	TFEN_42	TFEN_41	TFEN_40
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x2F

TFEN_39	TFEN_38	TFEN_37	TFEN_36	TFEN_35	TFEN_34	TFEN_33	TFEN_32
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x2E

TFEN_31	TFEN_30	TFEN_29	TFEN_28	TFEN_27	TFEN_26	TFEN_25	TFEN_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x2D

TFEN_23	TFEN_22	TFEN_21	TFEN_20	TFEN_19	TFEN_18	TFEN_17	TFEN_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x2C

TFEN_15	TFEN_14	TFEN_13	TFEN_12	TFEN_11	TFEN_10	TFEN_9	TFEN_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x2B

TFEN_7	TFEN_6	TFEN_5	TFEN_4	TFEN_3	TFEN_2	TFEN_1	TFEN_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- **DATA_Timer : 6 registres**

Dans le cas de l'utilisation en chronomètre, correspond au nombre de coups d'horloge entre les deux conditions.
Dans le cas de l'utilisation en compteur, correspond au nombre d'évènements dans la fenêtre programmée.

Cette valeur est un entier 48 bits occupant donc six registres.

Poids Fort

Registre 0 : Adresse = 0x52							
DatT_47	DatT_46	DatT_45	DatT_44	DatT_43	DatT_42	DatT_41	DatT_40
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x51							
DatT_39	DatT_38	DatT_37	DatT_36	DatT_35	DatT_34	DatT_33	DatT_32
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x50							
DatT_31	DatT_30	DatT_29	DatT_28	DatT_27	DatT_26	DatT_25	DatT_24
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x4F							
DatT_23	DatT_22	DatT_21	DatT_20	DatT_19	DatT_18	DatT_17	DatT_16
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x4E							
DatT_15	DatT_14	DatT_13	DatT_12	DatT_11	DatT_10	DatT_9	DatT_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

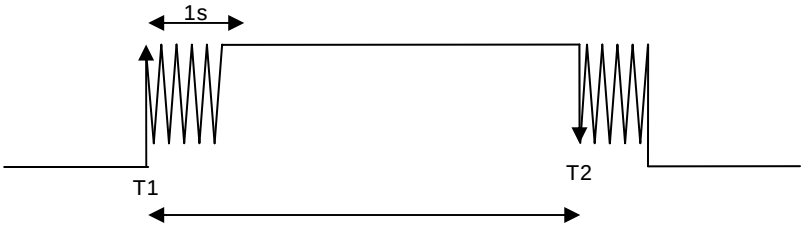
Poids Faible

Registre 0 : Adresse = 0x4D							
DatT_7	DatT_6	DatT_5	DatT_4	DatT_3	DatT_2	DatT_1	DatT_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

3. Exemples :

EXEMPLE 1 :

On veut mesurer le temps séparant T1 de T2.



1. Ecriture du registre de configuration :

   ⇨ **Ecriture de 0x000E à l'adresse 0x26**
   Mode Chronomètre
   Déclenchement sur Front Montant
   Arrêt sur Front descendant

2. Ecriture des registres de Hysteresis_Timer :

   On veut que le temps d'inhibition soit de 1s, un coup d'horloge durant 10ns, il faut donc mettre 100.000.000 comme valeur dans Hysteresis_Timer, soit 0x05F5E100.

   ⇨ **Ecriture de 0x05** (Poids Fort) **à l'adresse 0x2A**
   ⇨ **Ecriture de 0xF5 à l'adresse 0x29**
   ⇨ **Ecriture de 0xE1 à l'adresse 0x28**
   ⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x27**

3. Lancement du chronomètre et Remise à zéro du Timer : Ecriture du registre de commande (cf. Mise en route des modules)

   ⇨ **Ecriture de 0x48 sur registre de commande à l'adresse 0x24**

4. Attente de la fin de la mesure (cf. Mise en route des modules) : Lecture en boucle du registre de commande (**0x25**) et vérification du bit 1.
   Deux possibilités :
   ⇨ **Bit1 = 0 :** La mesure est terminée
   ⇨ **Bit1 = 1 :** La mesure est en cours

5. Lecture du Résultat : Lecture des 3 Registres DATA_Timer :

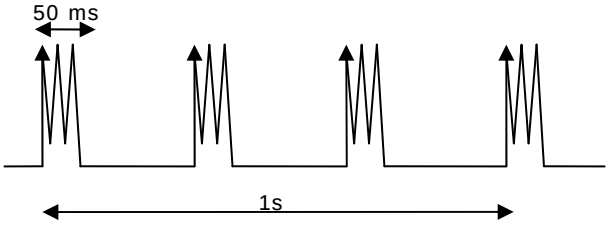
   Exemple : On lit les valeurs :

   - ⇨ **Lecture de 0x00** (Poids Fort) **à l'adresse 0x30**
   - ⇨ **Lecture de 0x00 à l'adresse 0x2F**
   - ⇨ **Lecture de 0x02 à l'adresse 0x2E**
   - ⇨ **Lecture de 0x98 à l'adresse 0x2D**
   - ⇨ **Lecture de 0x1E à l'adresse 0x2C**
   - ⇨ **Lecture de 0x4B** (Poids Faible) **à l'adresse 0x2B**

   Soit 43523659 coups d'horloge, soit une durée de 4.3523659s.

EXEMPLE 2 :

On veut compter le nombre de fronts montants en 1 seconde.



1. Ecriture du registre de configuration :

   - ⇨ **Ecriture de 0x0007 à l'adresse 0x26**
     Mode Compteur
     Remise à zéro du Timer
     Comptage des Fronts Montants

2. Ecriture des registres de Hysteresis_Timer :

   On veut que le temps d'inhibition soit de 50ms, un coup d'horloge durant 10ns, il faut donc mettre 5.000.000 comme valeur dans Hysteresis_Timer, soit 0x004C4B40.

   - ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x2A**
   - ⇨ **Ecriture de 0x4C à l'adresse 0x29**
   - ⇨ **Ecriture de 0x4B à l'adresse 0x28**
   - ⇨ **Ecriture de 0x40** (Poids Faible) **à l'adresse 0x27**

3. Ecriture des registres de TFenetre_CPT :

   On veut que le temps de comptage soit de 1s, un coup d'horloge durant 10ns, il faut donc mettre 100.000.000 comme valeur dans TFenetre_CPT, soit 0x000005F5E100.

   - ⇨ **Ecriture de 0x00** (Poids Fort) **à l'adresse 0x30**
   - ⇨ **Ecriture de 0x00 à l'adresse 0x2F**
   - ⇨ **Ecriture de 0x05 à l'adresse 0x2E**
   - ⇨ **Ecriture de 0xF5 à l'adresse 0x2D**
   - ⇨ **Ecriture de 0xE1 à l'adresse 0x2C**
   - ⇨ **Ecriture de 0x00** (Poids Faible) **à l'adresse 0x2B**

4. Mise à zéro et lancement du compteur : Ecriture du registre de commande (cf. Mise en route des modules)

   - ⇨ **Ecriture de 0x48 sur registre de commande à l'adresse 0x24**

5. Attente de la fin de la mesure (cf. Mise en route des modules) : Lecture en boucle du registre de commande (**0x25**) et vérification du bit 0.

   Deux possibilités :
   - ⇨ **Bit1 = 0 :** La mesure est terminée
   - ⇨ **Bit1 = 1 :** La mesure est en cours

6. Lecture du résultat : Lecture des 3 Registres DATA_Timer :

   Exemple : On lit les valeurs :

   - ⇨ **Lecture de 0x00** (Poids Fort) **à l'adresse 0x52**
   - ⇨ **Lecture de 0x00 à l'adresse 0x51**
   - ⇨ **Lecture de 0x00 à l'adresse 0x50**
   - ⇨ **Lecture de 0x00 à l'adresse 0x4F**
   - ⇨ **Lecture de 0x00 à l'adresse 0x4E**
   - ⇨ **Lecture de 0x04** (Poids Faible) **à l'adresse 0x4D**

   Soit 4 fronts montants observés pendant une seconde.

VIII. Logique :

1. Présentation :

Ce bloc permet d'avoir accès en lecture-écriture aux ports logiques de la centrale.

2. Détails des Registres :

- **Port_B_L_OE / PortB_H_OE : 1 registre**

  Défini la configuration du port B (entrée ou sortie).

  Cette valeur nécessite deux bits et occupe donc un registre.

Registre 0 : Adresse = 0x3D

x	x	x	x	x	x	PBOEH_0	PBOEL_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

PBOEL_0 : Mettre ce bit à 0 pour configurer les bits 0 à 3 du Port B en Entrée.
Mettre ce bit à 1 pour configurer les bits 0 à 3 du Port B en Sortie.

PBOEH_0 : Mettre ce bit à 0 pour configurer les bits 4 à 7 du Port B en Entrée.
Mettre ce bit à 1 pour configurer les bits 4 à 7 du Port B en Sortie.

Exemple :

Pour configurer le port B en entrée sur les bits 0 à 3 et en sortie sur les bits 4 à 7 :

⇨ ***Ecriture de 0x02 à l'adresse 0x3D***

- **Port_C_OE : 1 registre**

  Défini bit à bit, les bits du port C à utiliser en entrée ou en sortie.

  Cette valeur est un entier 8 bits occupant donc un registre.

Registre 0 : Adresse = 0x3E

PCOE_7	PCOE_6	PCOE_5	PCOE_4	PCOE_3	PCOE_2	PCOE_1	PCOE_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

PBOE_X : Mettre ce bit à 0 pour configurer le bit correspondant du port C en Entrée.
Mettre ce bit à 0 pour configurer le bit correspondant du port C en Sortie.

- **Port_B_OUT : 1 registre**

  Défini la valeur du port B en entrée/sortie.

  Cette valeur est un entier 8 bits occupant donc un registre.

Registre 0 : Adresse = 0x3F

PBO_7	PBO_6	PBO_5	PBO_4	PBO_3	PBO_2	PBO_1	PBO_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- **Port_C_OUT : 1 registre**

  Défini la valeur du port B en entrée/sortie.

  Cette valeur est un entier 8 bits occupant donc un registre.

Registre 0 : Adresse = 0x40

PCO_7	PCO_6	PCO_5	PCO_4	PCO_3	PCO_2	PCO_1	PCO_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

3. Exemples :

EXEMPLE 1 :

On veut lire la valeur présente en entrée sur le port B :

1. Configuration du Port en entrée:

   ⇨ **Ecriture de 0x00 à l'adresse 0x3D**
   Bits 0 à 3 du port B en entrée
   Bits 4 à 7 du port B en entrée

2. Lecture de la valeur présente à l'adresse 0x3F

EXEMPLE 2 :

On veut écrire la valeur 0x54 sur le port C :

1. Configuration des bits 2,4 et 6 du port C en sortie
   ⇨ **Ecriture de 0x24 à l'adresse 0x3D**

2. Ecriture de la valeur sur le port :
   ⇨ **Ecriture de 0x54 à l'adresse 0x40**

IX. Capteur :

1. Présentation :

Ce bloc permet d'obtenir l'identifiant des capteurs connectés à la centrale.

2. Détails des Registres :

- **IDFCapteurX : 1 registre**

  Donne la valeur de l'identifiant du capteur présent sur le canal X de la centrale.

  Cette valeur est un entier 8 bits occupant donc un registre.

Poids Fort

Registre 3 : Adresse = 0x44

IDF3_7	IDF3_6	IDF3_5	IDF3_4	IDF3_3	IDF3_2	IDF3_1	IDF3_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 2 : Adresse = 0x43

IDF2_7	IDF2_6	IDF2_5	IDF2_4	IDF2_3	IDF2_2	IDF2_1	IDF2_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 1 : Adresse = 0x42

IDF1_7	IDF1_6	IDF1_5	IDF1_4	IDF1_3	IDF1_2	IDF1_1	IDF1_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Poids Faible

Registre 0 : Adresse = 0x41

IDF0_7	IDF0_6	IDF0_5	IDF0_4	IDF0_3	IDF0_2	IDF0_1	IDF0_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

3. Exemple :

On veut lire la valeur de l'identifiant du capteur présent sur la voie 2 :

⇨ **Lecture de la valeur à l'adresse 0x43**

X. Accès direct :

1. Présentation :

Ce bloc permet de lire ou d'écrire un point sur une entrée ou une sortie

2. Détails des Registres :

- **Data_Voie_N : 2 registres**

  Permet de lire la valeur présente sur la voie N.

Poids Fort ↓ Poids Faible

Registre 1 : Adresse = 0x54+(2 x N)

DVN_11	DVN_10	DVN_9	DVN_8	DVN_7	DVN_6	DVN_5	DVN_4
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x53+(2 x N)

DVN_3	DVN_2	DVN_1	DVN_0	x	x	x	x
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- **DACN_Data_Direct : 2 registres**

  Permet d'écrire une valeur sur le DAC(N) (SAN+1).

  N représente le numéro du DAC sur lequel on souhaite écrire un point.

Poids Fort ↓ Poids Faible

Registre 1 : Adresse = 0x65+(2 x N)

x	x	x	x	DNDD_11	DNDD_10	DNDD_9	DNDD_8
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Registre 0 : Adresse = 0x64+(2 x N)

DNDD_7	DNDD_6	DNDD_5	DNDD_4	DNDD_3	DNDD_2	DNDD_1	DNDD_0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

ATTENTION :
Il est nécessaire d'écrire le poids faible, puis le poids fort.

3. Exemples :

EXEMPLE 1 :

On veut lire la valeur présente sur la voie EA4 :

⇨ **Initialisation de la carte en mode lecture directe :**

Ecriture de la valeur 1 au bit 4 du registre 36

⇨ **Latch des entrées :**

Ecriture de la valeur 1 au bit 5 du registre 36

⇨ **Lecture de la valeur à l'adresse 0x5B**
⇨ **Lecture de la valeur à l'adresse 0x5C**

⇨ **Arrêt du Latch des entrées :**

Ecriture de la valeur 0 au bit 5 du registre 36

⇨ **Arrêt de la carte en mode lecture directe :**

Ecriture de la valeur 0 au bit 4 du registre 36

EXEMPLE 2 :

On veut écrire a valeur 0xC2F sur SA1 (DAC0) :

⇨ **Initialisation de la carte en mode lecture directe :**

Ecriture de la valeur 1 au bit 4 du registre 36

⇨ **Latch des entrées :**

Ecriture de la valeur 1 au bit 5 du registre 36

⇨ **Ecriture de la valeur 0x2F à l'adresse 0x53**
Ecriture de la valeur 0x0C à l'adresse 0x54

⇨ **Arrêt du Latch des entrées :**

Ecriture de la valeur 0 au bit 5 du registre 36

⇨ **Arrêt de la carte en mode lecture directe :**

Ecriture de la valeur 0 au bit 4 du registre 36

XI. Numéro de Version du logiciel :

1. Présentation :

Ce bloc permet de lire connaître le numéro de version du micro logiciel présent dans le FPGA de Sysam-SP5.

2. Détails du registre :

Registre 0 : Adresse = 0x76

NVPF2	NVPF1	NVPF0	NVPf4	NVPf3	NVPf2	NVPf1	NVPf0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Les 5 bits de poids faibles représentent la « mineure version » de la version du logiciel.
Les 3 bits de poids forts représentent la « majeure version » de la version du logiciel.

3. Exemple :

La valeur « 00100000 » correspond donc à la version 1.0 du logiciel.

XII. Tableau d'adressage des registres :

Entrée Analogique				
Nom	Fonction	Taille	Adresse	R/W
EA_Active	Bit(n) : Voie n active	8 bits	0	R/W
EA_Calibre	Bit(n*3+2 downto n*3) : Calibre de la voie n	24 bits	3->1	R/W
EA_NbSkip	Indique le nombre d'échantillon ignoré (0 :Te=100 ns, 1 :Te=200ns,…)	40 bits	8->4	R/W
EA_NbPoint	Indique le nombre de points (par voie) voulus	18 bits	11(bits 1 -> 0) -> 9	R/W
ModeDifferentiel	Indique les can en mode différentiel	4 bits	12(bits 3->0)	R/W

Sortie Analogique				
Nom	Fonction	Taille	Adresse	R/W
DAC0_MonoCoup	Permet de ne pas reboucler l'émission sur DAC0	1 bit	76(bit 0)	R/W
DAC1_MonoCoup	Permet de ne pas reboucler l'émission sur DAC1	1 bit	76(bit 1)	R/W
DAC0_NbPoint	Indique le nombre de points pour le DAC 0	18 bits	15(bits 1 -> 0) -> 13	R/W
DAC1_NbPoint	Indique le nombre de points pour le DAC 1	18 bits	18(bits 1 -> 0) -> 16	R/W
DAC0_NbSkip	Indique le temps entre deux points émis pour le DAC 0 (0:Te=200 ns, 1: Te=400 ns …)	40 bits	23 -> 19	R/W
DAC1_NbSkip	Indique le temps entre deux points émis pour le DAC 1 (0:Te=200 ns, 1: Te=400 ns …)	40 bits	28 -> 24	R/W

RAM				
Nom	Fonction	Taille	Adresse	R/W
AccesDirectRAM_Actif	1: active le remplissage de la RAM	1 bit	69(bit 0)	R/W
AccesDirectRAM_ADR_Debut	Adresse de début de remplissage	18 bits	72(bits 1 -> 0) -> 70	R/W
AccesDirectRAM_ADR_Fin	Adresse de fin de remplissage	18 bits	75(bits 1 -> 0) -> 73	R/W

Triger / PreTriger				
Nom	Fonction	Taille	Adresse	R/W
Trig_SynchroExt	1 : Active le déclenchement par Synchro Externe	1 bit	29(bit 0)	R/W
Trig_Seuil	1 : Active le déclenchement par Seuil (résultat d'une des voies acquises)	1 bit	29(bit 1)	R/W
Trig_FrontMontant	1 : Le déclenchement s'effectue sur front montant	1 bit	29(bit 2)	R/W
PreTrig_Souple	1 : permet de triger même si le nombre de point en pretrig n'est pas atteint	1 bit	29(bit 3)	R/W
Trig_NumeroVoie	indique le numéro de voie servant au trig par seuil (000: Voie 0; 111: Voie 7)	3 bits	29(bits 6 -> 4)	R/W
Trig_Seuil_Valeur	Indique le seuil de déclanchement	12 bits	31(bits 3 -> 0) -> 30	R/W
PreTrig_Nb_Point	Indique le nombre de point gardé avant le déclenchement	18 bits	34(bits 1 -> 0) -> 32	R/W
Trig_Seuil_Hysteresis	Indique la taille de l'hystérésis pour le déclenchement par seuil	8 bits	35	R/W
S_DATA_PreTrig	Donnée (en 12 bits) précédente au Trig	12 bits	108 -> 107(bits 7 -> 4)	R
S_DATA_PostTrig	Donnée (en 12 bits) suivante au Trig	12 bits	110 -> 109(bits 7 -> 4)	R
S_Delta_TrigExterne	Nombre de battements à 80 MHz entre le dernier point acquis et le Trig	32 bits	114 -> 111	R

Mise en route des modules				
Nom	Fonction	Taille	Adresse	R/W
Start	Lance l'acquisition sur front montant de ce bit	1 bit	36(bit 0)	R/W
DAC0_Actif	1 : DAC 0 en fonctionnement	1 bit	36(bit 1)	R/W
DAC1_Actif	1 : DAC 1 en fonctionnement	1 bit	36(bit 2)	R/W
Run_Timer	1 : Timer en fonctionnement	1 bit	36(bit 3)	R/W
LectureDirecte	1 : Permet d'effectuer une lecture directe des valeurs lues, sans passage par la FIFO.	1 bit	36(bit 4)	R/W
Latch_Lecture_Directe	1 : Permet de latcher les entrées en accès direct	1 bit	36(bit 5)	R/W
RAZ_Timer	1 : RAZ du Timer	1 bit	36(bit 6)	R/W
ModePermanent	1 : Acquisition en mode permanent	1 bit	36(bit 7)	R/W

Flag				
Nom	Fonction	Taille	Adresse	R/W
OverFlow	1 : indique un dépassement du Timer	1 bit	37(bit 0)	R
Resultat_Timer_OK	1 : indique la fin du Timer	1 bit	37(bit 1)	R
Acquisition10MHZ	1 : indique que l'acquisition peut s'effectuer à 10 Mhz	1 bit	37(bit 2)	R

Timer				
Nom	Fonction	Taille	Adresse	R/W
Mode_Compteur	1 : Mode compteur (CPT) 0 : Mode Chronomètre (CHR)	1 bit	38(bit 0)	R/W
Rising_Timer	(CPT) 1 : compte les rising // (CHR) 1 : START sur Rising ; 0 : START sur Falling	1 bit	38(bit 1)	R/W
Falling_Timer	(CPT) 1 : compte les falling // (CHR) 1 : STOP sur falling ; 0 : STOP sur Rising	1 bit	38(bit 2)	R/W
Select_IO_Timer	Défini l'entrée du Timer (PortB ou PortC ou SynchroExt ou Timer_input)	5 bits	38(bits 7 -> 3)	R/W
Hysteresis_Timer	Défini le temps d'inhibition après un front valide pour qu'un autre front soit valide	32 bits	42 -> 39	R/W
Fenetre_Comptage	(CPT uniquement) Nombre de coup d'horloge pendant lequel on compte	48 bits	48 -> 43	R/W
Data_Timer	Résultat du Timer	48 bits	82 -> 77	R

Logique				
Nom	Fonction	Taille	Adresse	R/W
Port_B_L_OE	1: Bit 0 a 3 du port B en sortie	1 bit	61(bit 0)	R/W
Port_B_H_OE	1: Bit 4 a 7 du port B en sortie	1 bit	61(bit 1)	R/W
Port_C_OE	Bit n a 1 => Bit n du port C en sortie	8 bits	62	R/W
Port_B	Valeur du port B	8 bits	63	R/W
Port_C	Valeur du port C	8 bits	64	R/W

Capteur				
Nom	Fonction	Taille	Adresse	R
IDFCapteur0	Identifiant du capteur sur Canal 0	8 bits	65	R
IDFCapteur1	Identifiant du capteur sur Canal 1	8 bits	66	R
IDFCapteur2	Identifiant du capteur sur Canal 2	8 bits	67	R
IDFCapteur3	Identifiant du capteur sur Canal 3	8 bits	68	R

Acces Direct				
Nom	Fonction	Taille	Adresse	R/W
Data_Voie_0	Valeur lue sur la voie 0	12 bits	84 -> 83(bits 7 -> 4)	R
Data_Voie_1	Valeur lue sur la voie 1	12 bits	86 -> 85(bits 7 -> 4)	R
Data_Voie_2	Valeur lue sur la voie 2	12 bits	88 -> 87(bits 7 -> 4)	R
Data_Voie_3	Valeur lue sur la voie 3	12 bits	90 -> 89(bits 7 -> 4)	R
Data_Voie_4	Valeur lue sur la voie 4	12 bits	92 -> 91(bits 7 -> 4)	R
Data_Voie_5	Valeur lue sur la voie 5	12 bits	94 -> 93(bits 7 -> 4)	R
Data_Voie_6	Valeur lue sur la voie 6	12 bits	96 -> 95(bits 7 -> 4)	R
Data_Voie_7	Valeur lue sur la voie 7	12 bits	98 -> 97(bits 7 -> 4)	R
DAC0_Data_Direct	Valeur à émettre sur le DAC 0	12 bits	101(bits 3 -> 0) -> 100	R/W
DAC1_Data_Direct	Valeur à émettre sur le DAC 1	12 bits	103(bits 3 -> 0) -> 102	R/W

Registres réservés				
Nom	Fonction	Taille	Adresse	R/W
Ces registres sont réservés au fonctionnement interne de la centrale SYSAM-SP5.		2 bits	37(bits 3 -> 4)	NA
		8 bits	99	NA
		8 bits	104	NA
		8 bits	105	NA
		8 bits	106	NA
		8 bits	119	NA
Version_FPGA	Numéro de version du logiciel : Les 3 bits de poids fort désignent la « majeure version » et les 5 bits de poids faibles, la « mineure version ».	8 bits	118	R