UTILISATION DU SYSTEME CAMAC (Standard ESONE/IEEE) D.R. juin 04
==========

Le systeme CAMAC est une norme pour un système modulaire de transfert de données de mesures controlées par ordinateur. Ce système a été défini pour la 1ere fois par le comité ESONE ("European Standards On Nuclear Electronics" devenu ensuite "European Studies On Norms for Electronics") issu de l'EURATOM. La norme a été reprise plus tard par l'IEEE (Institute of Electrical and Electronics Engineers). La norme définit électriquement et mécaniquement des Modules (tiroirs) enfichées dans des Crates (baies). Les crates peuvent être reliés pour former une "Branche". Le système total peut être composé de plusieurs branches.
Aux TPAN nous utilisons un système composé d'un seul crate interfacé par un "crate-controller" Kinetics 3920 relié à une carte Kinetics 2925 sur le bus ISA des PCs.
Une commande (un cycle) CAMAC est spécifiée par une adresse et une fonction. 24 bits de données peuvent être transférés. Des bits de status (Q et X) indiquent le niveau de succès de l'opération.

L'ADDRESSAGE: Une entité s'adresse par: B,C,N,A

B: Numéro de la branche
C: Numéro du crate dans la branche
N: Position du module dans le crate (Station, Niche). N=1 à 23 (5bits)
A: Sous-adresse. A=0 à 15 (4 bits). Un module peut contenir plusieurs sous-ensembles. Par exemple plusieurs échelles de comptage.

LA FONCTION: 0 < F < 31 (5 bits) désigne le type d'opération:

LECTURE: F= 0 à 7. F=00XXX: Transfert de données CAMAC --> ordinateur.
Exemples: F=0 : lecture, F=2: lecture+effacement

ECRITURE: F= 16 à 23. F=10XXX: Transfert de données ordinateur --> CAMAC.

CONTROLE: F: 8-15 et 24-31, F=X1XXX: Fonctions de controle ou de test sans transfert de données.
Exemples: F=8:test LAM (réponse dans Q), F=9: effacement, F=27: test status.

LES DONNEES: 24 bits. (beaucoup de modules n'en utilisent que 16)

LES REPONSES Q et X:

Q indique le résultat d'une fonction de test ou le succès d'une opération.
X indique que le module connait (a reconnu) la combinaison A et F.

LES CONTROLES COMMUNS AU CRATE: envoyés à tous les modules d'un crate.

INITIALIZE (Z): Ce signal remet les modules dans leur état initial. Les registres sont remis à zéro. (allume l'inhibit)
INHIBIT (I) : Cette ligne est souvent utilisée comme VETO. Par exemple, des compteurs ne comptent plus ou des ADCs ne convertissent plus si ce signal est présent.
CLEAR (C) : Efface des registres.

LES APPELS: (LAM = Look At Me)

Un module peut demander de l'attention avec sa ligne individuelle "LAM". Cet état peut être testé (ou effacé) avec F8 (ou F10) dans un module particulier. Ces lignes individuelles sont regroupées au niveau du crate-controller. Ces LAMs peuvent être reliés au système d'interrutions de l'ordinateur.

```
======================================================================
ON TROUVE, DANS LES "DATA-SHEETS" (SPECIFICATIONS) DE CHAQUE MODULE,
LA LISTE DES FONCTIONS ET DES SOUS-ADRESSES RECONNUES PAR CE MODULE.
======================================================================

                  LE SOFTWARE POUR LE CAMAC
                  =========================
Il existe des standards ESONE/IEEE pour le software CAMAC. Ils ont été
implémentés dans divers laboratoires, pour les divers interfaces.

Nous avons implémenté pour le langage C, les fonctions standards suivantes:
(sauf mention contraire, les variables sont de type "int")

cdreg(&ext,b,c,n,a);     L'adresse d'un module, donnée par b,c,n,a,
                         est encodée dans "ext".

cfsa(f,ext,&data24,&q);  Execute la fonction "f" dans le module "ext".
                         "data24" contient la donnée (24 bits) et la
                         réponse Q est retournée dans "q".

cssa(f,ext,&data16,&q);  Comme "cfsa" mais le transfert est restreint
                         à 16 bits et "data16" est "unsigned short".

cccz(ext) ;              Execute le Z (initialiZe) dans le crate.
                         L'inhibit est allumé comme conséquence.

cccc(ext) ;              Execute le C (Clear) dans le crate.

ccci(ext,Iwant) ;        Allume ou éteint l'Inhibit suivant "Iwant".

ctci(ext,&Iact) ;        Retourne l'état actuel de l'inhibit dans "Iact".

ctstat(&k) ;             Retourne le résultat du dernier cycle effectué:
                         k=0: Q X,  k=1: noQ X,  k=2: Q noX,  k=3: noQ noX.

                  LA FONCTION kbhit()
                  ===================
Cette fonction retourne 1 si l'opérateur a tapé sur le clavier. On peut ainsi
introduire un test pour sortir d'une boucle infinie (acquisition par exemple)

                  UTILISATION SOUS LINUX ET C++
                  =============================
#include "/home/esocam/cam_kinetics.c"      // inclure les routines CAMAC
#include "/home/esocam/kbhit.c"             // inclure la fonction "kbhit".
...
g++ myprog.c -o myprog              :compilation
chm myprog                          :autoriser l'accès aux ports I/O.
./myprog                            :executer le programme

                  TEST DU CAMAC, LE PROGRAMME "camchk"
                  ====================================
Ce programme (self explaining) permet des accès CAMAC "manuels" et peut tester
le CAMAC avec un DATAWAY-DISPLAY BORER. Les transferts 24 bits sont testés ainsi
que la transmission correcte des A et F et les fonctions Z, C et Inhibit.
Utilisation: camchk
                  ... suivre les instructions...
```

```
//-------------------------------------------------------------
//   EXEMPLE1.C  Programme pour incrementer une variable et
//               l'afficher dans un dataway-display BORER.
//               Auteur: Divic RAPIN  DPNC UNI-GE   2004
//
#include <stdio.h>
#include <ctype.h>
#include "/home/esocam/cam_kinetics.c"        // routines d'accès au CAMAC
#include "/home/esocam/kbhit.c"               // fonction kbhit()

//=============================================================================
//=============================================================================
main(){
int borer, niche, q, data ;

// demander l'adresse du BORER et préparer son adresse:
printf("Donnez l'adresse (niche) du BORER :");   // demander et ...
scanf("%d", &niche) ;                            //... entrer l'adresse.
cdreg(&borer,0,0,niche,0);                       // encoder dans "borer"

printf("regardez le BORER.  Pour arreter: pressez une touche\n");

data=0 ;
while(1)  {                                      // boucle infinie ...
    data++ ;                                     // incrementer "data"
    cfsa(16,borer,&data,&q) ;                    // l'envoyer dans le CAMAC
    if(kbhit())break;         }                  // evtl. sortir de la boucle.

}  // fin de main()
```

```
//-------------------------------------------------------------
//   EXEMPLE2.C  Programme pour mesurer un taux de comptage dans une
//               échelle de comptage CAMAC (DPNC789, 2003, SIN S500, ...)
//               Auteur: Divic RAPIN  DPNC UNI-GE   2004
//
#include <stdio.h>
#include <ctype.h>
#include "/home/esocam/cam_kinetics.c"        // routines d'accès au CAMAC
#include "/home/esocam/kbhit.c"               // fonction kbhit()

//=============================================================================
//=============================================================================
main(){
int scaler, niche, a, q, nsec, ntmax,  comptage ;
float taux ;

// demander l'adresse du compteur et préparer son adresse:
//-------------------------------------------------------
printf("Donnez l'adresse du compteur (N A) :");   // demander et ...
scanf("%d%d", &niche, &a ) ;                      //... entrer l'adresse.
cdreg(&scaler,0,0,niche,a);                       // encoder dans "scaler"

// faire plusieurs mesures en demandant à l'opérateur la durée
//------------------------------------------------------------
while(1)  {
 printf("Durée de mesure (<0=exit) [sec]:") ;     // demander et ...
 scanf("%d", &ntmax) ;                            // lire la durée.
  if(ntmax<=0)break;                              // négatif pour s'arreter.

//   Effectuer une mesure
//   --------------------
  ccci(scaler, 1) ;                 // Inhibit ON  (veto empeche le comptage)
  cfsa( 2, scaler, &comptage, &q); // Effacer le scaler.(F2 = read&clear)
                                    // On pourrait aussi utiliser F9
  ccci(scaler, 0) ;                 // Inhibit OFF ( le comptage peut commencer)

  nsec=0;                           // Attente ...
  for( int i=0 ; i<ntmax ; i++ )    // ... de "ntmax"...
   {  sleep(1) ;                    // ... secondes ...
      nsec++  ;                     // ... au maximum ...
      if(kbhit())break ;  }         // ... avec interruption possible.

  ccci(scaler, 1) ;                 // Inhibit ON pour arreter le comptage
  cfsa( 0, scaler, &comptage, &q); // Lire le comptage (F0:lecture)
                                    //On pourrait aussi utiliser F2 (read/clear)

  taux = (float) comptage / nsec ;                // Calcul taux (flottant) ...
  printf("scaler:%d  temps:%d  taux:%f Hz\n",comptage,nsec,taux); //...et print

   }  // fin de la mesure (while(1))
}  // fin de main()
```