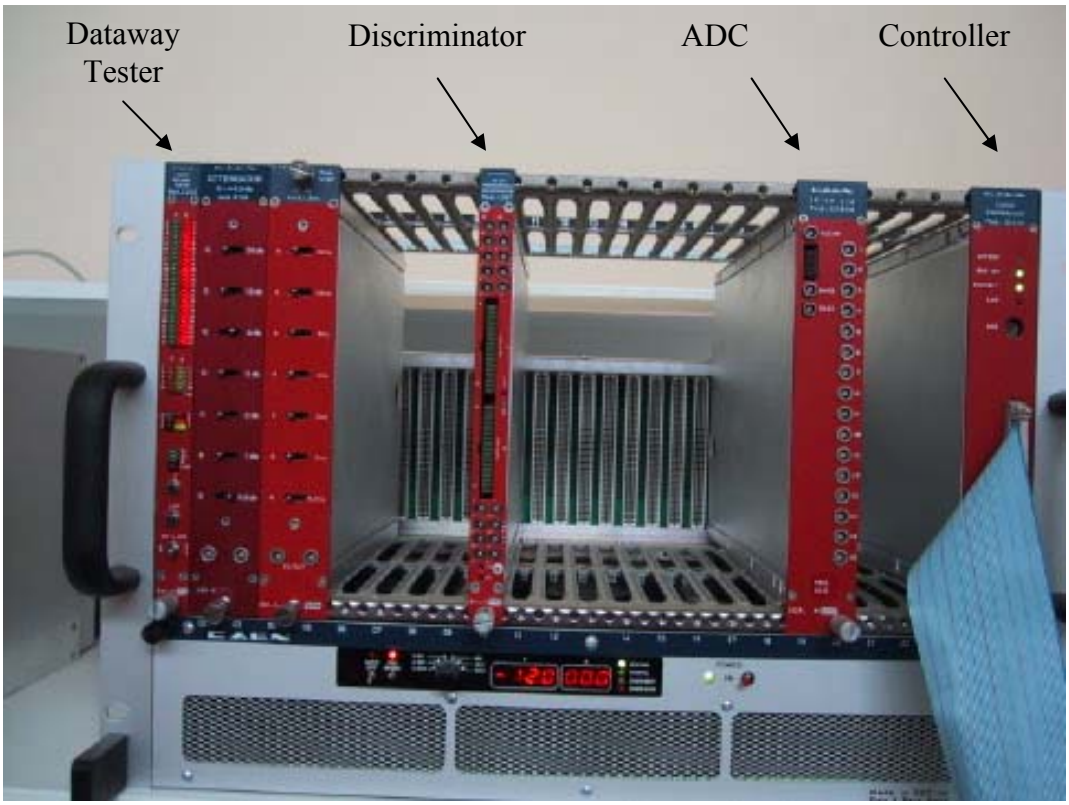
Πειραματική Ενότητα VI

ΛΗΨΗ ΔΕΔΟΜΕΝΩΝ ΜΕ CAMAC

Α) Σκοπός

Η πειραματική αυτή ενότητα αποσκοπεί στην εξοικείωση του φοιτητή με τη λειτουργία ενός σύγχρονου συστήματος λήψης δεδομένων. Με δεδομένη την προγραμματιστική υλοποίηση των βασικών εντολών-λειτουργιών του συστήματος, ο φοιτητής καλείται να υλοποιήσει, τόσο από πλευράς προγραμματισμού όσο και σε ότι αφορά την ηλεκτρονική διάταξη, μια σειρά από απλές εφαρμογές.

Β) Εισαγωγή στο CAMAC



Β1) Τι είναι το CAMAC

Το σύστημα CAMAC είναι ένα διεθνώς καθιερωμένο σύστημα επικοινωνίας μεμονωμένων ηλεκτρονικών. Η λειτουργία του έγκειται στην προσφορά ενός σχήματος που να επιτρέπει την ενδοεπικοινωνία μεμονωμένων ηλεκτρονικών μονάδων και ταυτόχρονα την επικοινωνία αυτών με έναν υπολογιστή. Με αυτόν τον τρόπο, η επέκταση ενός συστήματος μεταφοράς δεδομένων και ελέγχου, μπορεί να πραγματοποιηθεί με την προσθήκη νέων ηλεκτρονικών μονάδων και τις αντίστοιχες προσθήκες στο λογισμικό. Έτσι, το CAMAC επιτρέπει τη μεταφορά πληροφορίας από και προς τις ηλεκτρονικές μονάδες.

Οι μονάδες CAMAC τοποθετούνται στο κιβώτιο (crate) CAMAC που περιλαμβάνει 25 σταθμούς [1](stations) αριθμημένους από το 1 μέχρι το 25. Ο δεξιότερος σταθμός (25)[2]

[1] Υποδοχές τοποθέτησης μονάδων

χρησιμοποιείται για την τοποθέτηση του ελεγκτή (crate controller), ενώ οι υπόλοιποι σταθμοί (1-24) για κανονικές μονάδες CAMAC. Ο σκοπός του ελεγκτή είναι να εξασφαλίσει τη σωστή μεταφορά της πληροφορίας μεταξύ του υπολογιστή και των διαφόρων μονάδων.

Η μεταφορά πληροφορίας, ο έλεγχος των λειτουργιών καθώς και η τροφοδοσία των μονάδων πραγματοποιείται μέσω του dataway. Αυτό αποτελείται από μια σειρά συζευγμένων (bus) και ανεξάρτητων γραμμών κατά μήκος του πίσω μέρους του κιβωτίου CAMAC.Οι γραμμές του dataway περιλαμβάνουν ψηφιακές γραμμές μεταφοράς πληροφορίας, γραμμές σήματος (stobe signal), καθώς και γραμμές ελέγχου και κατεύθυνσης.

B2) Ορισμός των εντολών

Σε μία τυπική λειτουργία του dataway, ο ελεγκτής μεταδίδει μια εντολή CAMAC, η οποία περιλαμβάνει τον αριθμό ενός σταθμού **(N)**, την υποδιεύθυνση στον σταθμό αυτό **(A)** και τον κωδικό της ενέργειας που πρέπει να εκτελεστεί **(F)**. Σε απάντηση, η υποδιεύθυνση της μονάδας θα δημιουργήσει το σήμα, λήψη έγκυρης εντολής **(X-response)** και θα ενεργήσει κατάλληλα. Αν η εντολή απαιτεί τη μεταφορά πληροφορίας θα ενεργοποιηθούν και οι κατάλληλες γραμμές διαβάσματος **(R)** ή γραψίματος **(W)** . Αξίζει να σημειωθεί ότι οι όροι διάβασμα και γράψιμο αφορούν στον ελεγκτή και όχι στη μονάδα. Για παράδειγμα, με μια εντολή διαβάσματος, ο ελεγκτής διαβάζει δεδομένα που βρίσκονται στη μονάδα.

Ειδικότερα, κατά τη διάρκεια μιας ενέργειας του dataway, ο ελεγκτής δημιουργεί μια εντολή που περιλαμβάνει σήματα σε γραμμές μεμονωμένων σταθμών για να ορίσει συγκεκριμένα μία ή περισσότερες μονάδες, σήματα σε γραμμές υποδιευθύνσεων για να ορίσει την υποενότητα της μονάδας που θα δεχθεί την εντολή και σήματα στις γραμμές ενεργειών για να προσδιορίσει τη συγκεκριμένη ενέργεια που θα πραγματοποιηθεί. Τα παραπάνω σήματα συνοδεύονται και από ένα σήμα στη γραμμή κατειλημμένου (busy), που είναι διαθέσιμο σε όλους τους σταθμούς και δηλώνει ότι βρίσκεται σε εξέλιξη μια ενέργεια του dataway.

Οποτεδήποτε δεν βρίσκεται σε εξέλιξη κάποια ενέργεια του dataway (οπότε δεν είναι ενεργοποιημένο το σήμα busy), οποιαδήποτε μονάδα μπορεί να ενεργοποιήσει ένα σήμα στην δικιά της γραμμή Look-At-Me, δηλώνοντας έτσι ότι απαιτεί την προσοχή.

Αριθμός Σταθμού (station number) (N)

Σε κάθε σταθμό (εκτός του ελεγκτή) απευθυνόμαστε με ένα σήμα στην αποκλειστική γραμμή του (N) που προέρχεται από τον αντίστοιχο ακροδέκτη του ελεγκτή σταθμού. Οι σταθμοί είναι αριθμημένοι, σε δεκαδική βάση, από το 1 μέχρι το 25 ξεκινώντας από αριστερά όπως βλέπουμε το κιβώτιο CAMAC.

Υποδιεύθυνση (sub-address) (A8, A4, A2, A1)

Απευθυνόμαστε σε διαφορετικές υποδιευθύνσεις ενός σταθμού, μέσω σημάτων στις τέσσερις γραμμές (A). Τα σήματα αυτά αποκωδικοποιούνται στη μονάδα για να επιλεχθεί μία από τις 16 δυνατές επιλογές.

Ενέργεια (function) (F16, F8, F4, F2, F1)

Η ενέργεια που θα πρέπει να πραγματοποιηθεί στην επιλεγμένη υποδιεύθυνση του επιλεγμένου σταθμού ορίζεται από σήματα στις 5 γραμμές F. Τα σήματα αποκωδικοποιούνται στη μονάδα ώστε να επιλεχθεί μία από τις 32 δυνατές ενέργειες.

---

[2] Συχνά για τον ελεγκτή απαιτείται και ο σταθμός 24, αφού οι περισσότερες τέτοιες μονάδες έχουν πλάτος δύο θέσεων

Στη γενική τους μορφή οι δυνατές ενέργειες περιγράφονται στον παρακάτω πίνακα, δεν είναι όμως απαραίτητο μια μονάδα να ανταποκρίνεται σε όλες από αυτές, ούτε να τις αντιλαμβάνεται απαραίτητα με τον τρόπο που περιγράφεται παρακάτω.

Εντολές CAMAC

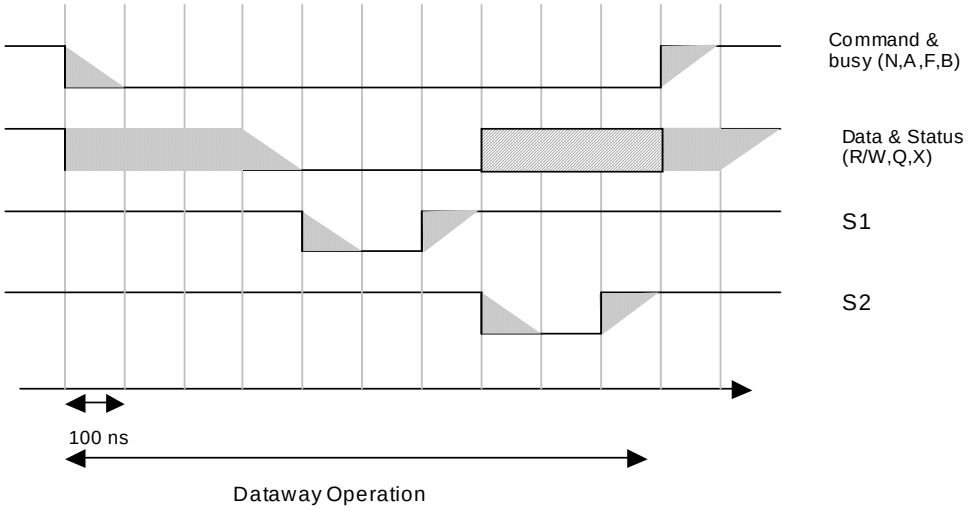
F( )	Περιγραφή εντολής	F16	F8	F4	F2	F1
0	Διάβασμα Καταχωρητή 1	0	0	0	0	0
1	Διάβασμα Καταχωρητή 2	0	0	0	0	1
2	Διάβασμα και καθάρισμα Κ 1	0	0	0	1	0
3	Διάβασμα και καθάρισμα Κ 2	0	0	0	1	1
4	Ακαθόριστο	0	0	1	0	0
5	Δεσμευμένο	0	0	1	0	1
6	Ακαθόριστο	0	0	1	1	0
7	Δεσμευμένο	0	0	1	1	1
8	Έλεγχος LAM	0	1	0	0	0
9	Καθάρισμα Καταχωρητή 1	0	1	0	0	1
10	Έλεγχος και καθάρισμα LAM	0	1	0	1	0
11	Καθάρισμα Καταχωρητή 1	0	1	0	1	1
12	Ακαθόριστο	0	1	1	0	0
13	Δεσμευμένο	0	1	1	0	1
14	Ακαθόριστο	0	1	1	1	0
15	Δεσμευμένο	0	1	1	1	1
16	Γράψιμο Καταχωρητή 1	1	0	0	0	0
17	Γράψιμο Καταχωρητή 2	1	0	0	0	1
18	Επιλεγμένο γράψιμο Κ 1	1	0	0	1	0
19	Επιλεγμένο γράψιμο Κ 2	1	0	0	1	1
20	Ακαθόριστο	1	0	1	0	0
21	Επιλεγμένο καθάρισμα Κ 1	1	0	1	0	1
22	Ακαθόριστο	1	0	1	1	0
23	Επιλεγμένο καθάρισμα Κ 2	1	0	1	1	1
24	Απενεργοποίηση	1	1	0	0	0
25	Εκτέλεση	1	1	0	0	1
26	Ενεργοποίηση	1	1	0	1	0
27	Έλεγχος κατάστασης	1	1	0	1	1
28	Ακαθόριστο	1	1	1	0	0
29	Δεσμευμένο	1	1	1	0	1
30	Ακαθόριστο	1	1	1	1	0
31	Δεσμευμένο	1	1	1	1	1

Σήματα χρονισμού (strobe signals) (S1, S2)

Δύο σήματα χρονισμού δημιουργούνται σε σειρά σε διαφορετικές γραμμές. Αυτά τα σήματα χρησιμοποιούνται για να μεταφέρουν πληροφορίες στις μονάδες ή για να οδηγούν τις μονάδες στην έναρξη κάποιας ενέργειας. Και τα δύο σήματα δημιουργούνται κατά την εκτέλεση ενός κύκλου του dataway. Το S1 χρησιμοποιείται για ενέργειες που δε μεταβάλλουν την κατάσταση των σημάτων στο dataway. Όλες οι μονάδες οι οποίες διαβάζουν ή γράφουν δεδομένα, το κάνουν σε απάντηση του σήματος S1. Το S2 χρησιμοποιείται για ενέργειες που μπορεί να μεταβάλλουν την κατάσταση των σημάτων στο dataway, για παράδειγμα το καθάρισμα ενός καταχωρητή (register).

Στο παρακάτω σχήμα, παρουσιάζεται ο χρονισμός των σημάτων για μία κατευθυνόμενη ενέργεια του dataway. Ο κύκλος του CAMAC διαρκεί τουλάχιστον 1μs για

την ενέργεια αυτή (τυπικά 1.2μs). Για μη κατευθυνόμενες ενέργειες (Z,C,I) το παρακάτω σχήμα διαφέρει ως προς το ότι χρησιμοποιείται μόνο το S2.



Χρονισμός σημάτων στο dataway για μία κατευθυνόμενη ενέργεια

B3) Μεταφορά δεδομένων

Η μεταφορά δεδομένων γίνεται μέσω παραλλήλων γραμμών. Μέχρι 24 bits μπορούν να μεταφερθούν παράλληλα μεταξύ του ελεγκτή και της επιλεγμένης μονάδας. Υπάρχουν ανεξάρτητες γραμμές για διάβασμα και γράψιμο.

Γραμμές γραψίματος (Write lines) (W1-W24)

Κατά τη διάρκεια μιας ενέργειας γραψίματος, ο ελεγκτής δημιουργεί σήματα στις γραμμές W. Τα σήματα αυτά φτάνουν σε σταθερή κατάσταση πριν από το S1 και παραμένουν σε αυτή μέχρι το τέλος της ενέργειας, εκτός αν μεταβληθούν από το S2.

Γραμμές διαβάσματος (Read lines) (R1-R24)

Κατά τη διάρκεια μιας ενέργειας διαβάσματος, η επιλεγμένη μονάδα δημιουργεί σήματα στις γραμμές R. Τα σήματα αυτά φτάνουν σε σταθερή κατάσταση πριν από το S1 και παραμένουν σε αυτή μέχρι το τέλος της ενέργειας, εκτός αν μεταβληθούν από το S2.

B4) Πληροφορίες κατάστασης και εντολές ελέγχου

Οι πληροφορίες κατάστασης μεταφέρονται με σήματα στις γραμμές Look-at-Me (LAM), busy (B) και response (Q).

Look-at-Me (LAM)

Αυτή, όπως και η γραμμή N, είναι μια ανεξάρτητη γραμμή μεταξύ κάθε σταθμού και ενός ακροδέκτη του ελεγκτή. Όταν δεν υπάρχει κάποια ενέργεια σε εξέλιξη στο dataway (το B δεν είναι ενεργοποιημένο), οποιαδήποτε μονάδα μπορεί να ενεργοποιήσει σήμα στη δική της γραμμή LAM, για να δείξει ότι απαιτεί προσοχή.

Busy (B)

Το σήμα αυτό χρησιμοποιείται για τον έλεγχο της πρόσβασης διαδικασιών που ανταγωνίζονται στη χρήση του dataway. Δημιουργείται κατά την εκτέλεση μιας ενέργειας στο dataway και ειδικότερα οποτεδήποτε είναι ενεργοποιημένη κάποια γραμμή N είναι ενεργοποιημένο και το B.

Response (Q)

Η γραμμή αυτή χρησιμοποιείται κατά τη διάρκεια μιας ενέργειας στο dataway, για τη μεταφορά σήματος που δηλώνει την κατάσταση ενός επιλεγμένου χαρακτηριστικού της μονάδας που ενεργοποιείται.

Τα κοινά σήματα ελέγχου μεταβιβάζονται σε όλες τις μονάδες, χωρίς να χρειάζονται διεύθυνση κατά την εντολή. Για να αποφεύγονται ενέργειες «κατά λάθος», τα σήματα Initialize (Z) και Clear (C) πρέπει να έχουν ταυτόχρονη ενεργοποίηση του S2.

Initialize (Z)

Το σήμα αυτό έχει απόλυτη προτεραιότητα απέναντι σε όλα τα υπόλοιπα. Θέτει όλες τις μονάδες στη βασική τους κατάσταση, αρχικοποιώντας ταυτόχρονα όλα τα σήματα και τους καταχωρητές.

Inhibit (I)

Η παρουσία αυτού του σήματος αποτρέπει οποιαδήποτε δραστηριότητα του συστήματος (όπως για παράδειγμα η λήψη δεδομένων).

Clear (C)

Το σήμα αυτό καθαρίζει όλους τους καταχωρητές

Γ) Διαθέσιμα Όργανα

Κατά την εκτέλεση της εργαστηριακής αυτής άσκησης είναι διαθέσιμα ένα CAMAC crate εφοδιασμένο με τις μονάδες που απαιτούνται για την υλοποίηση του κυρίου κορμού της άσκησης λήψης δεδομένων, ένα NIM crate εφοδιασμένο με μονάδες οι οποίες θα επιτρέψουν την εξαγωγή κατάλληλων παλμών που θα χρησιμεύσουν ως είσοδοι στο σύστημα λήψης δεδομένων και ένας παλμογράφος.

I) Το CAMAC crate (βλ. παρακάτω εικόνα) είναι εφοδιασμένο με:

1. Crate Controller (MOD C111A)
2. Dataway Tester (MOD C222)
3. 32 channel charge integrating ADC (MOD C205)
4. 16 channel Programmable Discriminator (MOD C207)
5. Fast Coincidence Unit
6. Fan-in fan-out Unit

II) Το NIM crate είναι εφοδιασμένο με:

1. Pulser
2. Ψηφιακό μορφοποιητή
3. Ενισχυτή- μορφοποιητή
4. Coincidence Unit
5. Delay Unit

Δ) Πειραματική διαδικασία

Δ1) Εξοικείωση

Στο πρώτο στάδιο της πειραματικής διαδικασίας εξοικειωνόμαστε με το προγραμματιστικό περιβάλλον που θα χρησιμοποιήσουμε (γλώσσα προγραμματισμού C και/ή LabView) και μελετάμε την προγραμματιστική υλοποίηση των βασικών εντολών-λειτουργιών του συστήματος Η προγραμματιστική υλοποίηση όλων των εντολών παρατίθεται στο παράρτημα Π1. Στη συνέχεια με ένα απλό interactive πρόγραμμα και τη βοήθεια που προσφέρει η οπτική αναπαράσταση των λειτουργιών, στο Dataway Tester (το οποίο παρατίθεται στη συνέχεια), διαπιστώνουμε πως δίνονται οι βασικές εντολές, στο σύστημα CAMAC και τη συγκεκριμένη υλοποίησή τους στο σύστημα λήψης δεδομένων που θα χρησιμοποιήσουμε στη συνέχεια. Το παρακάτω πρόγραμμα έχει μια πολύ απλή δομή, που

εν γένει θα πρέπει να ακολουθηθεί σε κάθε εφαρμογή που θα αναπτύξουμε στη συνέχεια. Σε πρώτη φάση αρχικοποιούμε τον ελεγκτή και τις μονάδες. Στη συνέχεια ορίζουμε το N, F, A για την εντολή που θα ακολουθήσει (το συγκεκριμένο πρόγραμμα ζητά τα παραπάνω εκπεφρασμένα από το χρήστη) και στη συνέχεια επιλέγουμε την κατάλληλη συνάρτηση που θα υλοποιήσει την εντολή που θέλουμε. Στην προκειμένη περίπτωση οι εντολές (`cam_nfaqx_write24, cam_nfaqx_read24`) υποθέτουν δεδομένα των 24 bit (βλ. παράρτημα **Π1.2**). Αξίζει να αναφέρουμε εδώ ότι το αρχείο **cam_tu16.h**, που αναγράφεται στο παράρτημα **Π1.2**, περιλαμβάνει των κώδικα για την ανάπτυξη όλων των εντολών που μπορεί να χρειαστεί κανείς στην ανάπτυξη ενός προγράμματος λήψης δεδομένων. Ο κώδικας αυτός αναφέρεται συχνά ως **drivers** του συστήματος και μπορεί να εξαχθεί από τον πίνακα με την ερμηνεία των θέσεων στη μνήμη που παρατίθεται στο παράρτημα **Π1.1**.

```
#include <stdio.h>
#include "cam_tu16.h"
#include <dos.h>
#include <conio.h>
int main()
{
   int   n1,f1,a1;
   int   q,x;
   long int   data;
   char  inbuf [130];
#define dip_setting   0x380   /*
PC16 base address       */
cam_controller_ini(dip_setting);
   n1=1;
   clrscr();
// demonstration of clear, inhibit
and inizialise
   cam_c();
   cam_i();
   cam_z();
   cam_ci();
// enable CC16 for LAM and LAM/IRQ
cam_irq();
//   demonstration of NAF read and
write operations
 while (n1 != 0)
   {
printf("\n CAMAC station N=\t");
      gets(inbuf);
      sscanf(inbuf, "\t%d", &n1);
printf("\n CAMAC function F= \t");
      gets(inbuf);
      sscanf(inbuf, "\t%d", &f1);
printf("\n CAMAC subaddress A= \t");
     gets(inbuf);
     sscanf(inbuf, "\t%d", &a1);
     printf("\n");
     if (f1>8)
     {
printf(" data to write : \t");
     gets(inbuf);
     sscanf(inbuf, "\t%li", &data);

cam_nfaqx_write24(n1,f1,a1,data,&q,&
x);
printf("\t\t\t  Q= %i   X= %i",q,x);
     }
     else
     {
     data =
cam_nfaqx_read24(n1,f1,a1,&q,&x);
printf(" data= %li",data);
printf("\t Q= %i    X= %i \n",q,x);
     }
   }
}
```

Δ2) Εφαρμογές Προγραμματισμού με το Dataway Tester.

Στο στάδιο αυτό της πειραματικής διαδικασίας, ζητείται από κάθε φοιτητή να φτιάξει ένα πρόγραμμα που να υλοποιεί μια απλή σειρά εντολών για το Dataway Tester που να έχει ένα συγκεκριμένο οπτικό αποτέλεσμα. Για παράδειγμα, το πρόγραμμά του να έχει ως αποτέλεσμα το σταδιακό άναμμα όλων των κόκκινων led του Dataway Tester, ή το περιοδικό αναβόσβημα των περιττών ή των άρτιων ψηφίων κ.ο.κ. Στο στάδιο αυτό, κάνουμε χρήση των έτοιμων προγραμματιστικών στοιχείων (υπορουτίνες C και / ή στοιχείων LabView) που μελετήσαμε νωρίτερα. και αντιστοιχούν στις βασικές εντολές του συστήματος λήψης δεδομένων. Η περίληψη των λειτουργιών CAMAC που μπορεί να υλοποιήσει το Dataway Tester αναφέρονται στο παράρτημα **Π2.1** , ενώ για περισσότερες λεπτομέρειες μπορούμε να αναφερθούμε στο εγχειρίδιο της μονάδας.

Δ3) Προγραμματισμός Διευκρινιστή

Στο στάδιο αυτό θα επιχειρήσουμε τη βαθμονόμηση κάποιων από τα κανάλια του προγραμματιζόμενου διευκρινιστή. Για το σκοπό αυτό απαιτείται αρχικά να διαμορφώσουμε κατάλληλη συνδεσμολογία στο NIM crate, ώστε να μπορούμε να δίνουμε στην είσοδό του, τους επιθυμητούς παλμούς. Κάποια από τα **χαρακτηριστικά του διευκρινιστή,** που χρειάζεται να γνωρίζουμε για να προσδιορίσουμε τους παλμούς που θα χρησιμοποιήσουμε, είναι τα ακόλουθα:

Κανάλια Εισόδου	
Πολικότητα	Αρνητική
Εμπέδηση	50 Ω
Όριο Τάσης	5 V
Ρύθμιση Κατωφλίου	-6 mV μέχρι –510 mV ανά 2 mV

Αφού με τη βοήθεια του Pulser και του ενισχυτή που βρίσκονται στο NIM crate δημιουργήσουμε τους κατάλληλους παλμούς για την είσοδο του διευκρινιστή, πρέπει, στη συνέχεια, να φτιάξουμε ένα πρόγραμμα το οποίο να ελέγχει και να καταγράφει την τιμή κατωφλίου για την οποία ένα συγκεκριμένο κανάλι του διευκρινιστή ανταποκρίνεται στο δεδομένο παλμό εισόδου που δέχεται. Αυτό θα το πραγματοποιήσουμε με γνώμονα ότι όταν ένα κανάλι του διευκρινιστή δέχεται σήμα πάνω από το κατώφλι, ο διευκρινιστής στέλνει LAM. Η περίληψη των λειτουργιών CAMAC που μπορεί να υλοποιήσει ο διευκρινιστής αναφέρονται στο παράρτημα **Π2.2**, ενώ για περισσότερες λεπτομέρειες μπορούμε να αναφερθούμε στο εγχειρίδιο της μονάδας. Επαναλαμβάνουμε την εκτέλεση του προγράμματος για 10 παλμούς εισόδου διαφορετικού ύψους και μελετάμε την γραμμικότητα της απόκρισης και συγκρίνοντας με την τιμές που προβλέπει ο κατασκευαστής.

Δ4) Βαθμονόμηση ADC

Στο στάδιο αυτό θα επιχειρήσουμε τη βαθμονόμηση κάποιων από το κανάλια του ADC. Για να το επιτύχουμε αυτό απαιτείται αρχικά να διαμορφώσουμε κατάλληλη συνδεσμολογία στο NIM crate, ώστε να μπορούμε να δίνουμε σαν είσοδο στο ADC παλμούς επιθυμητής χρονικής διάρκειας και ύψους. (Δεδομένου ότι το ADC που θα χρησιμοποιήσουμε «ψηφιοποιεί» το φορτίο, θα πρέπει να γνωρίζουμε κάθε φορά το φορτίο με το οποίο το τροφοδοτούμε, από τη μέτρηση του παλμού εισόδου στον παλμογράφο). Η συγκεκριμένη μονάδα περιλαμβάνει δύο ADC. Το ADC1 και το ADC2. Κάποια από τα **χαρακτηριστικά του ADC,** που χρειάζεται να γνωρίζουμε για να προσδιορίσουμε τους παλμούς που θα χρησιμοποιήσουμε, είναι τα ακόλουθα:

Κανάλια Εισόδου	
Πολικότητα	Αρνητική
Εμπέδηση	50 Ω
Όριο Τάσης	1.5 V
Φορτίο μεγίστου κλίμακας	900 pC
Gate[3]	
Ύψος παλμού	Std NIM level (-0.8 V)
Χρονική διάρκεια	100 ns – 5 μs
Χρόνος Μετατροπής	1.6 ms
Συντελεστής μετατροπής	4 counts/pC
Κατώφλι	30 counts + 7 counts/100ns

[3] Χρονικό διάστημα κατά το οποίο το ADC είναι ενεργό, μετατρέποντας το αναλογικό σήμα σε ψηφιακό

Στη συνέχεια πρέπει να φτιάξουμε ένα πρόγραμμα το οποίο θα διαβάζει τη μνήμη του ADC κάθε φορά που αυτό θα «ενεργοποιείται», θα αποθηκεύει τις τιμές για να αναλυθούν στη συνέχεια και θα επαναφέρει το ADC στην αρχική του κατάσταση ώστε να ξαναδεχθεί δεδομένα. Η περίληψη των λειτουργιών CAMAC που μπορεί να υλοποιήσει το συγκεκριμένο ADC αναφέρονται στο παράρτημα **Π2.3**, ενώ για περισσότερες λεπτομέρειες μπορούμε να αναφερθούμε στο εγχειρίδιο της μονάδας.

Δ4.1 Υπολογισμός των κατωφλίων

Χωρίς σήμα εισόδου, λαμβάνουμε 10 κύκλους μετρήσεων χρησιμοποιώντας διαφορετικό χρονικό διάστημα μετατροπής (gate). Εξετάζουμε τη συμπεριφορά των αποτελεσμάτων σε ότι αφορά τη γραμμικότητα. Επαναλαμβάνουμε για διαφορετικό κανάλι του ADC.

Δ4.2 Υπολογισμός συντελεστών μετατροπής

Θέτοντας τη gate 1μs, λαμβάνουμε 10 κύκλους μετρήσεων χρησιμοποιώντας διαφορετικό ύψος παλμού εισόδου. Ο παλμός εισόδου θα πρέπει να περιλαμβάνεται ολόκληρος στο διάστημα της gate και να έπεται της ανόδου αυτής κατά 65 ns τουλάχιστον. Εξετάζουμε τη συμπεριφορά των αποτελεσμάτων σε ότι αφορά τη γραμμικότητα για τα δύο ενσωματωμένα ADC που περιλαμβάνει η κάρτα.

Παράρτημα Π1.1: Χάρτης διευθύνσεων εισόδου-εξόδου για το συγκεκριμένο σύστημα

Διεύθυνση	Όνομα	Συνάρτηση	Λειτουργία
BADR	Base Address	Διεύθυνση Crate	write
BADR+$02	NFA Port	N(0:4), F(5:9), A(10:15)	write
BADR+$04	HB_W	Write high byte	write
BADR+$06	LW_W	Write low word	write
BADR+$08	HB_R	Read high byte	read
BADR+$0A	LW_R	Read low word	read
BADR+$0C	BR_R	Block read	read
BADR+$0E	S_mode_W	Set CAMAC I(bit 10), NO I(bit 11), Lam-enable (bit 12), LAM disable (bit 13)	write
BADR+$10	S_mode_R	Read CAMAC I, Lam-enable	read
BADR+$12	R_S	Read Q (bit 0), X(bit 1)	read
BADR+$14	Crate_C	Release C cycle	write
BADR+$16	Crate_Z	Release Z cycle	write
BADR+$18	Crate_on	Set broadcast on	write
BADR+$1A	Crate_off	Set broadcast off	write

Οι προσθετέοι στις θέσεις μνήμης είναι σε δεκαεξαδικό σύστημα και αυξάνουν ανά δύο bytes.

Παράρτημα Π 1.2: Προγραμματιστική υλοποίηση εντολών CAMAC σε C++

```
// cam_tu16.h
//                CC16 / PC16-TURBO routines for C++
//  ------------------------------------------------------------
//   Unit contains :
//     cam_adr           - Set I/O Address (BADR)
//     getstatus         - Get CC16 Status
//     cam_cratecheck    - Test crate available
//     set_crate         - Set geographical crate Address
//     cam_controller_ini - Controller Initialisation
//     cam_0             - CAMAC No Inhibit
//     cam_z             - CAMAC Initialise (Z)
//     cam_c             - CAMAC Clear (C)
//     cam_i             - CAMAC Inhibit (I)
//     cam_ci            - CAMAC Clear + Inhibit (I)
//     cam_irq           - enable CAMAC LAM-Interrupt
//     cam_noirq         - disable CAMAC LAM-Interrupt
//     cam_q             - get CAMAC Q-response
//     cam_x             - get CAMAC X-response
//     cam_nfa           - set N,F,A start CAMAC cycle
//     cam_nfa_read      - set N,F,A and read 2 byte
//     cam_nfaqx_read    - set N,F,A and read 2 byte with Q and X
//     cam_nfaqx_read24 - set N,F,A and read 3 byte with Q and X
//     cam_nfa_write     - set N,F,A and write 2 byte
//     cam_nfaqx_write   - set N,F,A and write 2 byte with Q and X
//     cam_nfaqx_write24- set N,F,A and write 2 byte with Q and X
//     cam_lam           - get LAM source
//     cam_lamf          - any LAM request
//
//     badr := basic I/O address
//     copyright:    A. Ruben, W-Ie-Ne-R, Plein & Baus GmbH  21.06.95
//                   revision
// ****************************************************************
#include <dos.h>

int badr00; int badr02; int badr04; int badr06; int badr08; int badr0a;
int badr0c; int badr0e; int badr10; int badr12; int badr14; int badr16;
int badr18; int badr1a; int badr1c; int badr1e;

void cam_adr(int);
int  cam_cratecheck (void);
void set_crate (int);
void cam_controller_ini (int);
int  getstatus(void);
void cam_z(void);
void cam_c(void);
void cam_i(void);
void cam_ci(void);
void cam_0(void);
int  cam_q(void);
int  cam_x(void);
void cam_irq(void);
void cam_noirq(void);
void cam_nfa(int, int, int);
void cam_nfa_write(int, int, int, int);
void cam_nfaqx_write(int, int, int, int, int, int);
void cam_nfaqx_write24(int, int, int, long int, int, int);
unsigned int  cam_nfa_read(int, int, int);
unsigned int  cam_nfaqx_read(int, int, int, int, int);
long int cam_nfaqx_read24(int, int, int, int, int);
```

```
//***********************************

void cam_adr(int bad)
{
  badr00=bad;
  badr02=bad+2;
  badr04=bad+4;
  badr06=bad+6;
  badr08=bad+8;
  badr0a=bad+10;
  badr0c=bad+12;
  badr0e=bad+14;
  badr10=bad+16;
  badr12=bad+18;
  badr14=bad+20;
  badr16=bad+22;
  badr18=bad+24;
  badr1a=bad+26;
  badr1c=bad+28;
  badr1e=bad+30;
};
//***********************************

int  cam_cratecheck (void)
{
  int j;
  int cnr;
  cnr=0;
  outport(badr00,cnr);
  j=inport(badr10);
  j=(j & 4)-4;
  if (j<0)
  {
   j=1;
  }
  return(j);
};
//***********************************

void   set_crate (int crate)
{
  outport(badr00,(crate & 16));
};
//***********************************

void cam_controller_ini (int bad)
{
  cam_adr(bad);
  set_crate(0);
  cam_z();
};
//***********************************

int getstatus(void)
{
  char i;
  char j;
  j=0;
  j=inport(badr10);
  return(j & 7);
};
```

```
//***********************************

void cam_z(void)
{
  outport(badr16,0);
};
//***********************************

void cam_c(void)
{
  outport(badr14,0);
};
//***********************************

void cam_i(void)
{
  outport(badr0e,10);
};
//***********************************

void cam_ci(void)
{
  outport(badr14,0);
  outport(badr0e,10);
};
//***********************************

void cam_0(void)
{
  outport(badr0e,11);
};
//***********************************

int cam_q(void)
{
  char j;
  j=inport(badr12);
  return ( j & 1);
};
//***********************************

int cam_x(void)
{
  char j;
  j=inport(badr12);
  return ( j & 2) > 1;
};
//***********************************

void cam_irq(void)
{
  outport(badr0e,12);
};
//***********************************

void cam_noirq(void)
{
  outport(badr0e,13);
};

//***********************************
```

```
void cam_nfa(int n, int f, int a)
{ int nfa;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr02,nfa);
};
//**********************************

void cam_nfa_write(int n, int f, int
a, int data)
{ int nfa;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | n;
  outport(badr02,nfa);
  outport(badr06,data);
};
//**********************************

void cam_nfaqx_write(int n, int f,
int a, int data,int *q, int *x)
{ int nfa;
  char j;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | n;
  outport(badr02,nfa);
  outport(badr06,data);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
};
//**********************************

void cam_nfaqx_write24(int n, int f,
int a, long int data,int *q, int *x)
{ int nfa;
  char j;
  int ih;
  int il;
  ih= (int) ((data >> 16) & 255);
  il= (int) data;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | n;
  outport(badr02,nfa);
  outport(badr04,ih);
  outport(badr06,il);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
};
//**********************************

unsigned int cam_nfa_read(int n, int
f, int a)
{ int nfa;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr02,nfa);
  return ((unsigned int)
inport(badr0a));
};
//**********************************

unsigned int cam_nfaqx_read(int n,
int f, int a, int *q, int *x)
{ int nfa;
  char j;
  nfa= a << 5;

  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr02,nfa);
  nfa=(unsigned int) inport(badr0a);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
  return (nfa);
};
//**********************************

long int cam_nfaqx_read24(int n, int
f, int a, int *q, int *x)
{ int nfa;
  char j;
  long int lh;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr02,nfa);
  lh=inport(badr08) & 255;
  lh=lh <<16;
  lh += (unsigned int)
inport(badr0a);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
  return (lh);
};
//**********************************

int cam_lam(int station)
{
  cam_nfa(station,8,0);
  return (cam_q());
};
//**********************************

int cam_lamf(void)
{
  unsigned int j;
  j=inport(badr12);
  j= j & 4;
  return(j > 2);
};
```

Παράρτημα Π 2.1: Εντολές CAMAC του Dataway Tester (CAEN C222)

F(0) N [A(0)] Reads the Write Data Register.

F(l) N [A(0)] Reads the Status Register.

F(2) N [A(0)] Reads and clears the Write Data Register; clears LAM ("RI' display not affected).

F(8) N [A(0)] Tests if L-AM is set (Q response ff LAM is set).

F(9) N [A(0)] Clears the Write Data Register ("RI' display not affected).

F(10) N [A(0)] Clears LAM.

F(l 6) N IA(0)] The Data **(24** bits, on Wl - W24) overwrite the Write Data Register.

F(17) N [A(0)l The Data (1 **4** bits, on Wl - Wl 2, Wl **4 -** Wl **5)** overwrite the Status Register.

F(24) N [A(0)] Disables LAM.

F(24) N A(l) Disables MONITOR mode.

F(25) N [A(0)] Sets LAM.

F(26) N IA(0)] Enables LAM.

F(26) N A(l) Enables MONITOR mode.

F(27) N [A(0)] Tests d LAM is enabled (O = 1 if LAM is enabled).

X response for each valid function.

Q response for each valid function, unless otherwise specified

Παράρτημα Π 2.2 : Εντολές CAMAC του Διευκρινιστή (CAEN C207)

F(0) N A(0-15) Reads the Discriminator Thresholds on Rl..R8.

F(l) N Reads the Pattern of Inhibit on Rl.. Rl 6.

F(2) N Tests the module activity and clears the Q line.
Q response if at least one channel is over threshold.

F(16) N A(0-15) Writes the Discriminator Thresholds on Wl..W8.

F(17) N Writes the Pattern of Inhibit on Wl..Wl 6.

F(25) N Common Test.

I Vetoes the channels via CAMAC.

F(9) N, C, Z Resets the module

Παράρτημα Π 2.3 : Εντολές CAMAC του ADC (CAEN C205)

F(2) N A(0) Reads the n-th memory location and moves to the (n+l)-th location. Q-response for each reading until the 64th included. Q response is FALSE and LAM is cleared at the 65th reading (see table below).

PERFORMED FUNCTION	**CHANNEL**	**READS WORD COMING FROM**	**Q RESPONSE**	**LAM**
1 St	1	ADC1	TRUE	ON
2nd	1	ADC2	TRUE	ON
3rd	2	ADC1	TRUE	ON
4th	2	ADC2	TRUE	ON
…	…	…	…	…
63rd	32	ADC1	TRUE	ON
64th	32	ADC2	TRUE	ON
65th			FALSE	OFF

F(8) N A(0) Tests the LAM presence. Q response if LAM is true.

F(9) N A(0) Resets the module (LAM and BUSY signals are deactivated but the input charge persists). It does not give a 0 response.

F(10) N A(0) Tests and clears LAM. Q response if LAM is true.

Z,C Same as F(9) N A(O).