Tcl/Tkによる CAMACの制御

谷口億宇

平成 14 年 10 月 21 日

概 要

これまでは CAMAC の制御を nfa プログラムを用いて行なってきた．しかし，毎回 nfa を用いるのは不便であるし，何より収集したデータを視覚的にとらえにくい．そこで，Tcl/Tk に nfa などを組み込み，CAMAC を使いやすくすることを試みた．

目 次

1 はじめに

今回作ったシステムは，

- switch register 制御
- scaler 制御
- ratemeter

の 3 つである．

まず nfa プログラムを説明し，それぞれの仕組みと使い方解説する．また，switch register の swing program を作ったので，それを最後に説明する．それぞれの source は最後にのせてある．

2 nfa program の概要

nfa program は CAMAC 制御の program である．引数を 3 つとり，

```
./nfa N F A
```

というふうに使う．N は制御する module の刺さってる場所，F は module に出す命令，A は module のどこ (subadress) に命令を出すかをあらわす．例えば，

```
./nfa 22 16 0
```

とすると，22 番に刺さってる module の subadress=0 の部分に 16(書き込み) の命令を出すという意味になる．よく使う命令は，F = 0, 9, 16 である．特に F = 0, 16 は頻繁に使う．F = 0 はデータの読み込みを，F = 16 はデータの書き込みを，F = 9 はデータの clear をあらわす．他の F の番号と命令との関係は [1] の P.345 参照.

3 switch register 制御 (source···§7.2)

ここで作ったシステムは，window から switch register と信号を送受信するものである．まず，

```
./menu.tcl
```

によって，menu(source···§7.1) を読み出す．そこで switch register をクリックすると，switch register 制御の window が開く．

switch register 制御の window には入力枠と，write, read, clear, exit ボタンがある．数字を入力して write をクリックすると，switch register にその数が入力される．また，その入力は clear をクリックすることで clear される．

また，switch register の値を読み出すのは read をクリックすれば良い．

switch register を終えるには exit をクリックする．

4 scaler 制御 (source···§7.3)

scaler 制御の window を出すには menu で scaler をクリックする．すると，数入力枠，read,clear,exit ボタンが出る．

ここで，枠にどのサブアドレスに命令を出すかを入力し，read をクリックすると scaler の値が読み出される．また，scaler の値を clear するときは clear をクリックする．終わる時には exit をクリックする．

5 ratemeter(source···§7.4)

ratemeter を出すには menu で ratemeter をクリックする．計測開始時に clear をクリックし，rate が知りたくなったら適時 rate をクリックする．それまでの rate の時間変化を知りたければ，graph をクリックすればそれまでの rate の時間変化が graph 表示される．

ここで使った回路は fig.1 のようなものである．function

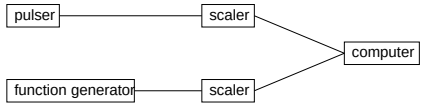

図 1: rate meter の回路

generator からの信号と pulser からの信号を computer で取り込み，pulser からの信号を基準に function generator からの信号の rate を計測している．

6 swing program(source···7.5)

switch register swing program は [2] によって用意されている．それに少し手を加えたのが test_ led2.c である．test_ led.c ではただ単に switch register のランプが上下に移動するだけだったのだが，test_ led2.c ではどう上下するかを決めることができる．

```
./test2_led.c
```

とすると，up と down を聞いてくるので，数を入力するとそれにしたがってランプが上下する．

7 source

7.1 menu

```
#!/bin/sh
# the next line restarts using wish \
exec wish "$0" "$@"
```

```
wm title . menu

button .switch -text "switch register" -command {
    exec ./switch_register.tcl &
}

button .scaler -text scaler -command {
    exec ./scaler.tcl &
}

button .rate -text ratemeter -command {
    exec ./rate.tcl &
}

button .exit -text exit -command exit

pack .switch .scaler .rate .exit
```

7.2 switch register 制御

```
#!/bin/sh
# the next line restarts using wish \
exec wish "$0" "$@"

wm title . "switch register"

button .read -text read -command {
    set value [exec ./nfa2 22 0 1]
    view $value
}
proc view {value} {
    .label configure -text $value
}

label .label2

button .exit -text exit -command exit

entry .value -textvariable v

button .write -text write -command {
    exec ./nfa 22 16 0 $v
}

button .clear -text clear -command {
    set v ""
    .label configure -text clear!
    exec ./nfa 22 16 0 0
}

label .label

pack .label .value .write .read .clear .exit
```

7.3 scaler 制御

```
#!/bin/sh
# the next line restarts using wish \
exec wish "$0" "$@"

wm title . "scaler"

button .read -text read -command {
    set value [exec ./nfa2 10 0 $adress]
    view $value
}
proc view {value} {
    .label configure -text $value
}

label .label

button .exit -text exit -command exit

button .clear -text clear -command {
    .label configure -text clear!
    exec ./nfa 10 9 $adress
}

entry .adress -textvariable adress

pack .label .adress  .read .clear .exit
```

7.4 ratemeter

```
#!/bin/sh
# the next line restarts using wish \
exec wish "$0" "$@"

wm title . "ratemeter"
set time 0.
label .label

button .rate -text rate -command {
```

```
    set pulse [ exec ./nfa2 10 0 1]
    set function [ exec ./nfa2 10 0 0 ]
    set rate [ expr 1000. * $function / $pulse ]
    set dt [ expr $pulse / 1000. ]
    set time [ expr $time + $dt ]
    .label configure -text $rate
    exec ./write $time $rate
    exec ./nfa 10 9 0
    exec ./nfa 10 9 1
}


button .exit -text exit -command exit

button .clear -text clear -command {
    .label configure -text clear!
    set time 0.
    exec rm rate.dat
    exec ./nfa 10 9 0
    exec ./nfa 10 9 1
}

button .graph -text graph -command {
    exec gnuplot "load_gnuplot"
}

pack .label .rate .clear .graph .exit
```

7.5 switch register swing program

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>

#include "camac.h"

#define WAIT 10000L

main(int argc, char *argv[])
{
  int fd, i, j,n,m;
  DCCycle dcc;
  if (argc < 2) {
    i = 22;
  } else {
    sscanf(argv[1], "%d", &i);
  }

  dcc.n  =  i;
  dcc.f  = 16;
  dcc.a  =  0;
  dcc.data = 3;
  printf("up\n");
  scanf("%d",&n);
  printf("down\n");
  scanf("%d",&m);

  for (i = 0; i < 15; i++) {
    for (j = 0; j < n; j++) {
      ioctl(fd, DC_CYCLE, &dcc);
      usleep(WAIT);
      dcc.data <<= 1;
    }
    for (j = 0; j < m; j++) {
      ioctl(fd, DC_CYCLE, &dcc);
      usleep(WAIT);
      dcc.data >>= 1;
    }
  }
  close(fd);
}
```

参考文献

[1] W.R.Leo;*Technique for Nuclear and Particle Physics Experiments-Second Revised Edition*-(Springer-verlag)

[2] 岡村弘之;CC/7000 CAMAC クレートコントローラのためのデバイスドライバ