# CAMAC 用デバイスドライバ

T.Takahashi, *Dept. Phys.,Tohoku University*

2000/7/3

要 旨

Linux Kernel 2.2.x で動く CAMAC Crate Controller CC7000、CC7700 と CCP-F、また、VME-PCI Inerface (Bit3) 616/617 のドライバ vmehb を使った Kinetic K2917/K3922 の Device Driver 及び Library を開発した。この文書では、これらの Driver 及び Library の使い方を説明する。また、行った性能テストの結果も報告する。

## 1 Driver 及び Library の作成とインストール

開発した Driver は、TOYO/CC7000 用と HOSHIN/CCP-F 用である。HOSHIN 用は、まだテストは行っていないが、TKO Controller の CCP-TKO-II での TKO もサポートしている。それぞれの在処は、

| | |
|---|---|
| CC7000 | http://lambda.phys.tohoku.ac.jp/~takahasi/unidaq/cc70.tar.gz |
| CC7700 | http://lambda.phys.tohoku.ac.jp/~takahasi/unidaq/cc77.tar.gz |
| CCP-F | http://lambda.phys.tohoku.ac.jp/~takahasi/unidaq/ccp.tar.gz |
| K2917/K3922 | http://lambda.phys.tohoku.ac.jp/~takahasi/unidaq/k2917.tar.gz |
| VME-PCI | http://lambda.phys.tohoku.ac.jp/~takahasi/unidaq/vmehb-2.2.7.tar.gz |

である。入手したアーカイブを適当なディレクトリで展開するとその中に diver と example というディレクトリができる。driver へ行き、作成とインストールを行う。example には、テストや例題のプログラムが入っている。

それぞれの device が使用する IO Port と IRQ 及びキャラクターデバイスのメジャー番号は、デフォールトでは、次の様に設定されている。必要なら、cc70_config.h(ccp_config.h) 及び Makefile を修正する。

| Device | IO Port | IRQ | Major Number |
|---|---|---|---|
| CC7000 | 0x310–0x317 | 5 | 62 |
| CC7700 | automatic | auto | 45 |
| CCP-F | 0x17E–0x17F | 3 | 45 |
| K2917 | | | 46 |
| VMEHB | automatic | auto | 62 |

なお、k2917 は、VME Address は、A16 mode で、0xFF00、VME IRQ/VECTOR は、4/0x10 としている。Bit3 の VME ボードは、システムコントローラとし、VMEIRQ4 を CINT4 につなぐように Jumper 設定が必要である。

- make
  Driver 及び Library の作成
- make nod
  Device file の作成 (root 権限が必要)

- make install
  Driver 及び Library のインストール (root 権限が必要 )

Driver は、カーネルモジュールのインストール先 (cf. /lib/modules/2.2.14/misc/ ) へインストールされる。Library (libcamlib.a) は、/usr/local/daq/lib/cc70(ccp) へ、必要な Header files は、/usr/local/daq/include/cc70(ccp) へインストールされる。インストール先を変更したい場合は、Makefile を修正する。

Driver のカーネルへの組み込みは、insmod コマンドで行う。kerneld の機能を使うと使用時に自動的に組み込むことが可能だが、その方法は、他書に譲る。

## 2 ライブラリ関数

Device Driver がサポートしているシステムコールは、open, close, ioctl のみであり、これらを直接ユーザープログラムから使うのは大変である。そこでよく使う CAMAC とのやりとりなど、ライブラリ関数化した。これらの関数は、KEK 標準の関数と互換性をできる限り保つ様に作 ってある。以下に機能別にこれらの関数の使い方を示す。

- COPEN

  ```
  int COPEN();
  return value: 0:success 1:failure
  ```

  CAMAC をオープンする。オープンできるプロセスは、デバイスに対して 1 つである。失敗の原因として考えられるのは、Device Driver がロードされていないか、他のプロセスがオープン中である。

- CCLOSE

  ```
  int CCLOSE();
  return value: 0:success others:error (errno)
  ```

  CAMAC をクローズする。通常失敗は考えられない。

- CRESET

  ```
  int CRESET();
  return value: always 0
  ```

  クレートリセット。Not implemented.

- CSETBR

  ```
  int CSETBR( bno );
    int bno; Branch Number
  return value: always 0
  ```

  Brach 番号のセット。Not implemented.

- CSETCR

```
int CSETCR( cno );
  int cno;  Crate Number
return value: always 0
```

Crate の選択。CC7000 では、0 から 3 までの最大 4 台、CCP-F では、0 から 7 までの最大 8 台のクレートを接続できる。オープン時には、0 に設定されている。

- CGENZ

```
int CGENZ();
return value: 0:success others:failure (errno)
```

Z を発行。CCP-F は Z により I(Inhibit) がかかるので注意。

- CGENC

```
int CGENC();
return value: 0:success others:failure (errno)
```

C を発行。

- CSETI

```
int CSETI();
return value: 0:success others:failure (errno)
```

I をセットする。

- CREMI

```
int CREMI();
return value: 0:success others:failure (errno)
```

I をリセットする。

- NAF

```
NAF(N,A,F);
```

CAMAC N,A,F から NAF コードを生成する。 (マクロ定義 ) 以下の関数で用いる。

- CAMACL

```
int CAMACL(naf,&data,&q,&x);
  int naf; NAF コード
  int data; Read/Write Data (Input/Output)
  int q,x; Q,X Response (Output)
return value: 0:success others:failure (errno)
```

CAMAC 24Bit のシングルアクション。CAMAC との別名もある。

- CAMACW

```
int CAMACW(naf,&data,&q,&x);
  int naf; naf コード
  short data; Read/Write Data (Input/Output)
  int q,x; Q,X Response (Output)
return value: 0:success others:failure (errno)
```

CAMAC 16Bit のシングルアクション。data が short であることに注意。

- CENLAM

```
int CENLAM(lampat);
  int lampat; LAM Pattern
return value: 0:success others:failure (errno)
```

LAM 割り込みを許可する。LAM Pattern にあう N からの割り込みが、CWTLAM で OK となる。

- CDSLAM

```
int CDSLAM();
return value: 0:success others:failure (errno)
```

LAM 割り込みを許可しない。 (注 ) 実際には、CPU は割り込み処理を行っています。User Program に割り込みが通知されないだけです。

- CWTLAM

```
int CWTLAM(timeout);
  int timeout; Timeout までの時間 (sec.)
return value: 0:LAM 割り込み others(errno): Timeout や、他の割り込み発生
```

LAM 割り込みを timeout 秒待つ。タイムアウトや他の割り込みの発生でもリターンする。

これらの Library を使うプログラムを書くには、ソースファイルの先頭で、camlib.h をインクルードして、libcamlib.a をリンクする。Makefile の例を以下にあげておく。

```
#
# Example of Makefile for Use of CAMAC Library
#
CLIB    = -L/usr/local/daq/lib/ccp -lcamlib
CINC    = -L/usr/local/daq/include/ccp
CFALGS  = -O

camtest.o: camtest.c
        $(CC) $(CFLAGS) $(CINC) -c $<
camtest: camtest.o
        $(CC) $(CFLAGS) -o $@ $(CLIB)
```

また、Driver 付属の camtest のソースを例としてあげておく。

```
/* camtest.c */
#include <stdio.h>
#include <sys/types.h>
#include <camlib.h>
#define CHARLEN 144
int main( int argc, char **argv )
{
  int status, nc;
  char str[CHARLEN];
  int n=0, a=0, f=0;
  char c;
  int nr, ar, fr, dr, cr;
  int flag=1;
  int data=0, q, x;
  int crate=0;

  if(status=CAMOPN()){
    printf("CAMAC Open Failure\n"); exit(-1);
  }
  printf("Crate" ==> [%d]",crate);
  fgets(str,CHARLEN,stdin);
  if(sscanf(str,"%d",&cr)==1) crate=cr;
  CSETCR(crate);
  CREMI();
  while(1){
    printf("N A F or C, Z, I(set I), R(reset I), Q(uit) [%d %d %d] ==>",
           n,a,f);
    fgets(str,CHARLEN,stdin);
    if( (nc=sscanf(str,"%c",&c))==1 &&
       ( c=='c' || c=='C' ) ){  /* Clear */
      flag=0; CGENC();
    }
    else if( nc==1 && ( c=='z' || c=='Z' ) ){ /* Initialize */
      flag=0; CGENZ();
    }
    else if( nc==1 && ( c=='i' || c=='I' ) ){ /* Set Inhibit */
      flag=0; CSETI();
    }
    else if( nc==1 && ( c=='r' || c=='R' ) ){ /* Reset Inhibit */
      flag=0; CREMI();
    }
    else if( nc==1 && ( c=='q' || c=='Q' ) ){ /* Quit */
      flag=0; break;
```

```
    }
    else if( (nc=sscanf(str,"%d %d %d",&nr,&ar,&fr))==3 ){
      n=nr; a=ar; f=fr; flag=1;
    }
    else if( nc==2 ){
      n=nr; a=ar; flag=1;
    }
    else if( nc==1 ){
      n=nr; flag=1;
    }
    else{
      flag=1;
    }

    if(flag){
      if(f>=16 && f<24){
        printf("Write Data [%d] ==> ",data);
        fgets(str,CHARLEN,stdin);
        if(sscanf(str,%d",&dr)==1) data=dr;
      }
      status=CAMAC(NAF(n,a,f),&data,&q,&x);
      printf("ST=%d Q=%d X=%d N=%d A=%d F=%d : Data=%x(%d)\n",
             status,q,x,n,a,f,data);
      printf("-------------------------------------------\n");
    }
  }
  CSETI();
  CCLOSE();
  return 0;
}
```

# 3 List Action

上で述べたCAMAC関数で大量のデータを読む時、その度に、Device Driverに要求を出している。(これをシステムコールという。)システムコールには、なにがしかのオーバーヘッドがあり、時間がかかる。また、一般には、速さの求められる処理は、読み出しであり、F0あるいはF2のみを使うことが多い。1つのFunctionしか使わないとすれば、ハードウェアによっては、F設定のためのインストラクションコードが省けるため、その分の時間短縮が、期待される。そのため、List Actionをサポートした。

List Actionを行うには、あらかじめ、Readout Listを用意しておき、このリストによるデータの読み出しを1回のシステムコールで処理する。ただし、上で述べたようにF設定のインストラクションを省くため、CC7000では、Fの異なるCAMAC Actionは1つのリストでは、実行できない。また、読み出されたデータのフォーマットの都合上、サポートされるのは、事実上、F0または、F2である。

さらに、モジュールによっては、Variable Data Sizeのものがあるので、Q-Responseのあるうちは、同じアドレスからデータを読む(Q-Stop Read)、Q-Responseのないデータは、取り除く(Q-Suppression)

機能を組み込んだ。データとアドレスの対応付けのため、データ中に N(5) と A(4),Q(1),X(1) を加えて、32Bit Data とした。そのため、CAMAC Data は、16Bit である。

List Action 関係の関数ををあげる。

- CAMRegisterList

```
int CAMRegisterList( id, size, bufsize, action_list );
  int id;           List Number
  int size;         Action List Size
  int bufsize;      この List を実行して読んだデータを格納するのに必要な
                   Buffer のサイズ。
  int *action_list; Action List
return value: 0:success others:failure(errno)
```

Action List を番号 id で登録する。登録できる List 数は、0 から 9 の 10 個。

- CAMUnregisterList

```
int CAMUnregiterList( id );
  int id: List Number
return value: 0:success others:failure(errno)
```

番号 id の List を削除する。

- CAMBlockRead

```
int CAMBlockRead( id, maxbufsize, buffer );
  int id;         List Number
  int maxbufsize; buffer size
  int *buffer;    buffer pointer
return value: written size in long word
```

番号 id の List を実行して、buffer に書く。

Action List は、1 word (4 Bytes) に付き、1 つの CAMAC Action を記述する。NAF コードと QS_READ との論理和で書く。

```
#define ADC   2  /* ADC */
#define TDC   3  /* TDC */
#define DBI   4  /* Data Bus Interface */
static int clist1[]={
  NAF(ADC,0,0),          /* Read ADC A(0) */
  NAF(ADC,1,0),          /* Read ADC A(1) */
  NAF(TDC,0,0),          /* Read TDC A(0) */
  NAF(TDC,1,0),          /* Read TDC A(1) */
  NAF(DBI,0,0) | QS_READ /* Read DBI A(0) with Q-Stop Mode */
};
static int listsize1=sizeof(clist1)/sizeof(int);
```

表 1: 測定結果

| | 1A 1B | 2A 2B | 3A 3B | 4A 4B |
|---|---|---|---|---|
| T0 – T1 [$\mu$sec] | 90 – 100 | 100 – 110 | 86 | 132 |
| T1 – T2 [$\mu$sec] | 164 103 – 107 | 285 – 350 250 – 300 | 122 100 | 620 460 |

```
int buffer[50];
int nread;

CAMRegisterList(1,listsize1,100,clist1);
nread=CAMBlockRead(1,50,buffer);
CAMUnregisterList(1);
```

出力フォーマットは、上で述べたように、4Bytes/Data で、

| Bit | | | |
|---|---|---|---|
| 0 | - | 15 | Read Data(16) |
| 16 | - | 19 | A(4) |
| 20 | - | 24 | N(5) |
| | | 30 | X(1) |
| | | 31 | Q(1) |

である。

# 4 性能

開発した Device Driver 及び Hardware の性能を kstar(G6-200) 上の UNIDAQ でテストした。プラスチックシンチレータの信号を ADC(LeCroy 2249A) に A(0) に入れ、LAM を発生させ、トリガーとする。まず、Output Register に出力 (T1) し、その後 ADC 12channel 分のデータを読んで、クリアー (F(9)) する。その後 Output Register に再び出力 (T2) する。ADC の Gate に入れた時間 (T0) と T1,T2 との時間をオシロスコープで測定した。ADC 16channel 分のデータを 16Bits Single Function (A) 及び、List Action (B) で読んだ場合を CC7000 (1), CCP-F (2) で測定した。ちなみに 2249A の Conversion Time は 60$\mu$sec である。cc77(3A,3B), k2917(4A,4B)

```
CWTLAM(10);
CAMACW(NAF(OUTREG,0,16),&outpat1,&q,&x); /* Write to Output Register (T1) */

for(a=0;a<12;a++)
  CAMACW(NAF(ADC,a,0),&data++,&q,&x);
     or
CAMBlockRead(1,12,buffer);        /* Read ADC A(0)-A(11) */

CAMACW(NAF(ADC,0,9),&data,&q,&x);  /* Clear ADC */
CAMACW(NAF(OUTREG,0,16),&outpat2,&q,&x); /* Write to Output Register (T2) */
```

割り込み応答時間は、約 20$\mu$sec である。Single Action では、CCP-F は CC7000 の 2 倍弱時間がかかっている。List Action にした場合、CCP-F では、それほど速くはなっていないが、これは、CCP-F の場合節約できるのは。システムコールのオーバーヘッドのみであるためで、CC7000 の場合は、F 発行のインストラクションを省けるおかげで、劇的に速くなっている。ラフな計算で、Single Action に要する時間は、CC7000 で、12$\mu$sec、CCP-F では、20$\mu$sec 強である。