2010 年度 P6dark 班中間レポート

相澤俊博 井上諒

2011 年 2 月 10 日

1 はじめに

このレポートは P6dark 班の実験で、μ-PIC を用いて放射線信号を見てスペクトルをとるところまでを記したものである。

2 実験目的

本実験の目的は μ-PIC を用いて荷電粒子の 2 次元及び 3 次元イメージングを行い、それを応用してダークマター探索を行うことである。実験で使用するガス検出器 μ-PIC は、比例計数管をピクセル状に 256 × 256 ヶ配し、信号をアノードとカソードで読み取ることで 2 次元イメージングを行うことができる。さらに、検出時間から粒子の軌跡の相対的な高さ情報が得られ、3 次元イメージングも可能である。現在は、μ-PIC で波形情報を取得し解析している段階である。今後、さらに発展的に μ-PIC を活用していく予定である。

3 実験装置

本実験で用いた機材及び試料を以下に示す。

- μ-PIC　(10cm 角)
  部品名　MPGC チュウケイキバン N3035-KA195
  製造 No.　SN 060830-2
- ASD
  アノード (16ns[C])、カソード (16nsC 仕様)
- PAN16-10A 直流安定化電源
  ASD 用電源 (−) として用いる　　−3.37V
- PAN16-30A 直流安定化電源
  ASD 用電源 (+) として用いる　　+3.36V
- VME module
  - FADC(RPV-160)
  - RPV-130
- NIM module

  - ハイボル MODEL RPH-030
    DRIFT Top,GEM Bottom に使用
  - ハイボル Model N471A
    GEM Top, アノードに使用
  - OCTAL DISCRIMINATOR
  - 4ch LOGIC FAN-IN/OUT
  - 2ch GATE DELAY GENERATOR (N-TE307)
  - QUAD LINEAR FAN-IN/OUT
  - 8ch VARIABLE GAIN AMPLIFER
- FUNCTION GENERATOR (TEKTRONIX CFG253)
- オシロスコープ (TECTRONIX TDS2024)
- $Ar + C_2\ H_6$(7.74 %) ガス
- ^{133}Ba

4　ノイズ落とし

μ-PIC にガスを入れ電源を繋いだところで、アノードからの信号をオシロスコープで見ると、数十 mV のノイズが見られた。およそ 10?20mV の信号が見えると考えると、それに比べてノイズが小さくなるようにしなければならない。そのために以下のことを行った。

- μ-PIC 及び ASD を同じアルミ板の上に乗せる。
- μ-PIC、ASD、ASD 用電源、VME、NIM を蛇腹や銅板などでつなぐ。

以上のようにしてグランドをそろえると、ノイズを 5mV 程度まで下げることができた。

図 1　ノイズ落とし

5 Dynamic range

FADC(RPV-160) の dynamic range の測定のために、各 ch ごとにピーク電圧がそれぞれ-64mV、-321mV、-657mV の周期の長い矩形波を入力し、そのピークに相当する信号より平均をとった（表 1)。

表 1 の 3 点でそれぞれ fitting を行い、各 ch ごとの目盛と電圧の関係を求めた (表 2)。

$$\text{電圧}\ [mV] = a \times \text{目盛} + b \tag{1}$$

これより、FADC(RPV-160) では約 +30mV -890mV の信号を入力でき、δ=3.6mV の信号を識別できることが分かった。

表 1　-64mV、-321mV、-657mV に対する測定値

	-64mV	-321mV	-657mV
ch0	27.7333	98.1437	190.475
ch1	26.9067	97.2643	188.305
ch2	25.9610	98.4733	190.927
ch3	25.7037	96.9557	186.732
ch4	25.5273	97.9197	189.749
ch5	25.8527	99.2950	190.501
ch6	26.0353	98.3223	188.999
ch7	26.1323	99.1937	190.889

表 2

	a	b
ch0	-3.644	36.88
ch1	-3.675	35.44
ch2	-3.596	30.73
ch3	-3.685	32.72
ch4	-3.613	29.87
ch5	-3.605	31.96
ch6	-3.642	33.04
ch7	-3.602	32.33

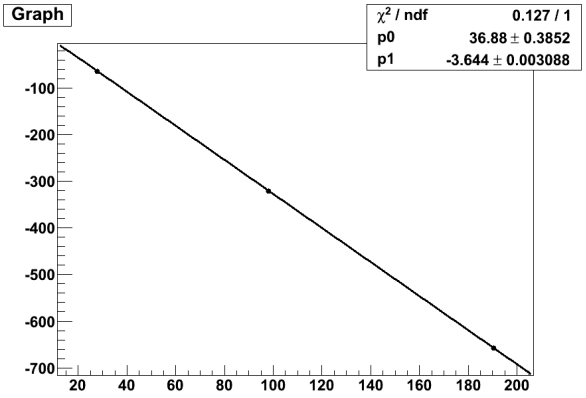
Graph
χ² / ndf 0.127 / 1
p0 36.88 ± 0.3852
p1 -3.644 ± 0.003088
-100
-200
-300
-400
-500
-600
-700
20
40
60
80
100
120
140
160
180
200

図 2　ch0

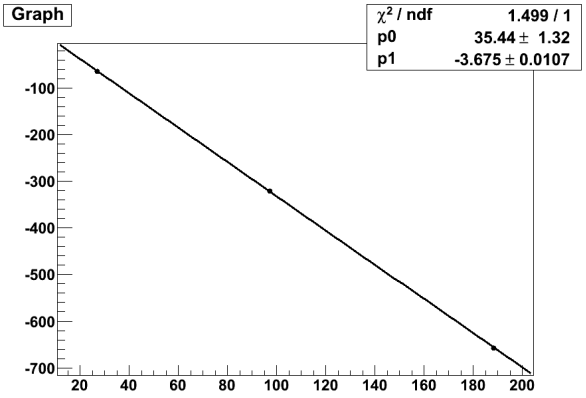
Graph
χ² / ndf 1.499 / 1
p0 35.44 ± 1.32
p1 -3.675 ± 0.0107
-100
-200
-300
-400
-500
-600
-700
20
40
60
80
100
120
140
160
180
200

図 3　ch1

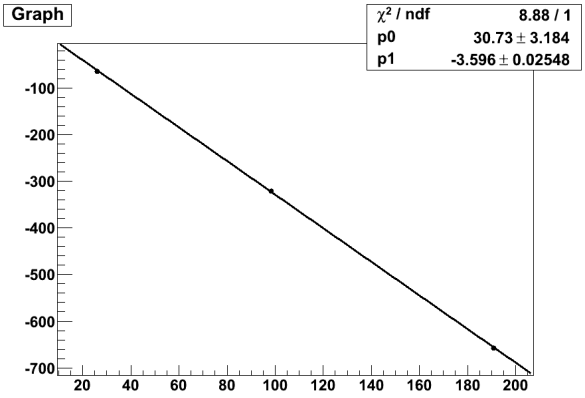
Graph
χ² / ndf 8.88 / 1
p0 30.73 ± 3.184
p1 -3.596 ± 0.02548
-100
-200
-300
-400
-500
-600
-700
20
40
60
80
100
120
140
160
180
200

図 4　ch2

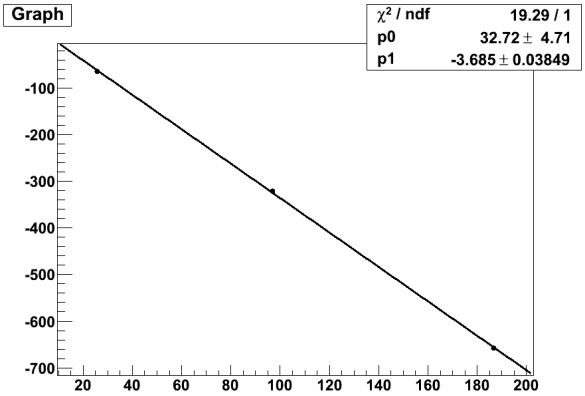
Graph
χ² / ndf 19.29 / 1
p0 32.72 ± 4.71
p1 -3.685 ± 0.03849
-100
-200
-300
-400
-500
-600
-700
20
40
60
80
100
120
140
160
180
200

図 5　ch3

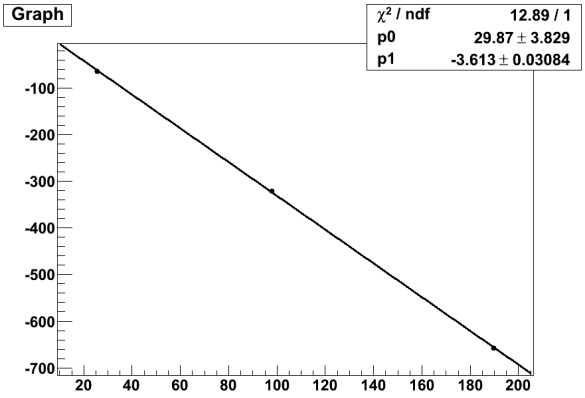

図 6　ch4

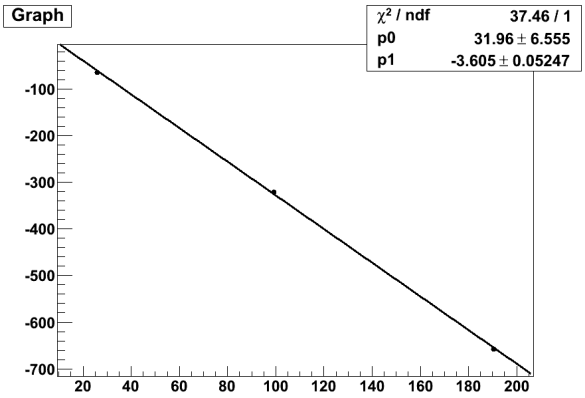

図 7　ch5

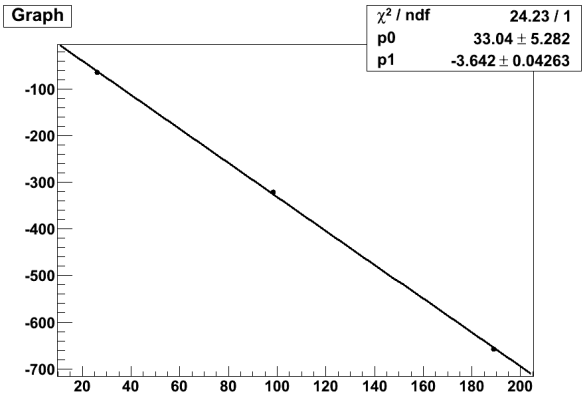

図 8　ch6

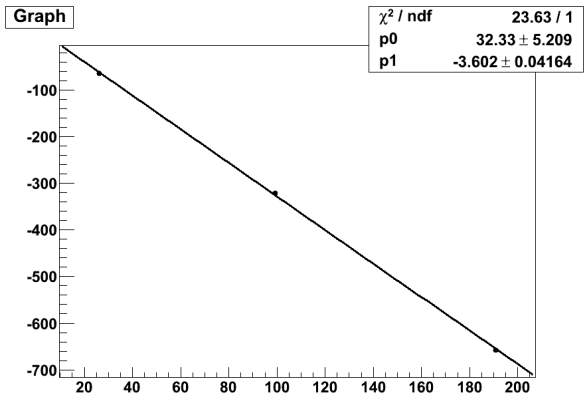

図 9　ch7

6　^{133}Ba のスペクトル取得

現在は線源 ^{133}Ba の信号を読み出し、スペクトル取得を行った段階である。
現在の設定は

- Drift Top -3195V
- GEM Top -526V
- GEM Bottom -499V
- アノード　460V
- phillips777 約 40 倍

で、ΔGEM を切っているのは、ΔGEM を 300V にしても信号が変化せず、GEM が全く機能していないことが判明した為である。また、phillips777 を用いているのはその為である。

一時、機能しない GEM を補完しようと、アノードを 480V くらいに設定してみたところ、放電により電流が不安定になってしまった為、アノードの R214 をカットしたら電流が安定した。

電荷の計算方法は、Discriminator の threshold を超えた信号が入ってきたとき約 6 μ sec の delay をかけ、STOP させる。そこから 6 μ sec 前の約 30 μ sec のデータが読みだされる。最初の 24 μ sec の平均をとることで Base Line を決定する。ここから、ノイズを除外する方法を考える。

baseline よりも +50mV 以上ある波形はノイズとみなして charge を 0 として計算し、除外する。baseline から-50mV のところにプログラムで Threshold をかける。さらに、欲しいエリアの部分は連続してその Threshold を超えたものであるとして、3 点以上連続して超えたもののみをプログラムで抽出し、それ以外の波形は charge を 0 として計算し、除外する。

ここで、電荷量 Q は、

$$Q = Area \times 1ch \text{ 当たりの電圧} \times 1ch \text{ 当たりの時間/オシロの抵抗値} \tag{2}$$

で求めた。

このように求めたそれぞれの ch の電荷を足しあわせ、ヒストグラムを描くと図 1 のように

なった。

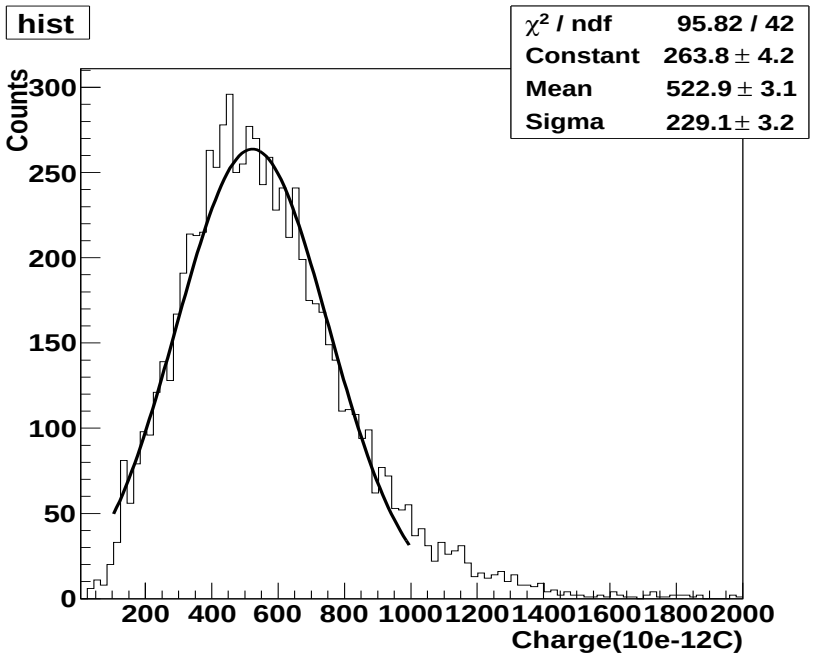

図 10　^{133}Ba のエネルギースペクトル

7 プログラムソース

RPV-160 の操作に使用するプログラム

- MakeFile

```
# target
TARGET = main

# suffix
ObjSuf          = .o
SrcSuf          = .cxx
ExeSuf          =

# set suffixs
TARGETOBJ = $(TARGET)$(ObjSuf)
TARGETSRC = $(TARGET)$(SrcSuf)
TARGETEXE = $(TARGET)$(ExeSuf)

# for linux
CXX     = g++
CXXFLAGS= -O2 -Wall -DSBS
LD      = g++
LDFLAGS = -O2 -Wall

LIBS      = $(SYSLIBS)
GLIBS     = $(SYSLIBS)

VME_OBJS = rpv160.o rpv130.o

all: $(TARGETEXE)

$(TARGETEXE): $(TARGETOBJ) $(VME_OBJS)
$(LD) $(LDFLAGS) $(LIBS) -o $(TARGETEXE) $(TARGETOBJ) $(VME_OBJS)
#mv $(TARGETEXE) main

#$(TARGETOBJ): $(TARGETSRC)
# $(CXX) $(CXXFLAGS) -c -o $(TARGETOBJ) $(TARGETSRC) -I.
```

```
.cxx.o :
$(CXX) $(CXXFLAGS) -c -o $@ $< -I.

.SUFFIXES : .o .cxx

clean:
rm -f *.o
rm -f *~
rm -f main
```

- main.cxx - main 関数

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include <iostream>
#include <fstream>



#include "rpv160.h"
#include "rpv130.h"

#define BASE 0x80000000
#define INT_BASE 0x4000

using namespace std;
```

```
RPV_130    *rpv;

void initialize(){
  rpv = new RPV_130();
  rpv->Open(INT_BASE);
  rpv->Reset();
  rpv->Level(0x00);
}

int main(int argc,char *argv[]){

  int        dev;
  size_t     map;
  unsigned int addr;
  int num;

  if(argc != 2){
    cout << "ERROR : ./main [filename]" << endl;
    exit(1);
  }
  char ofname[32];
  sprintf(ofname,"%s.dat",argv[1]);
  ofstream fout(ofname);

  addr = init_fadc( &dev, BASE, &map );

  set_fadc(addr);
  clear_fadc(addr);

  int flag;


  for( int k = 0; k < 10000; k++){
    start_fadc(addr);
    flag = read_status(addr);
    cout << "start : 0x" << hex << flag << dec << endl;

    while(1){
      sleep(1);
      flag = read_status(addr);
```

```
      cout << "0x" << hex << flag << dec << endl;
      if(flag == 0x0)
break;
    }
    cout << "after  0x" << hex << read_add_count(addr) << endl;

    num = data_num_fadc(addr);
    int ch[8][TAKE_CLOCK];

    for( int i = 0; i < 8; i ++ ){
      read_fadc(addr, num, i, &ch[i][0] );
    }

    for( int i = 0; i < TAKE_CLOCK; i ++ ){
      fout << i << " ";
      for( int j = 0; j < 8; j ++ ){
fout << ch[j][i] << " ";
      }
      fout << endl;
    }



  }

  rpv->Close();
  end_fadc( dev, map, addr);
  return 0;
}
```

- rpv160.h - 定数の定義

```
#define TAKE_CLOCK 3000
#define MAX_CLOCK 8192
```

- rpv160.cxx - 操作用関数

```
#include "rpv160.h"
#include <iostream>
```

```
using namespace std;

int data_num_fadc(unsigned int addr){
  unsigned short  *ptr;
  int             flag, num;
  while(1){
    //printf("waiting\n");
    //usleep(100);
    ptr = (unsigned short *)(addr + 0x30004);
    cout << "Status : 0x" << hex << *ptr << endl;
    flag = *ptr & 0x20;
    if(flag == 0)
      break;
  }
  ptr = (unsigned short *)(addr + 0x30000);
  num = *ptr - 1;
  num = num * 2;

  return num;
};

int read_add_count(unsigned int addr){
  unsigned short *ptr;
  ptr = (unsigned short *)(addr + 0x30000);
  return *ptr;
};


int read_status(unsigned int addr){
  unsigned short *ptr;
  ptr = (unsigned short *)(addr + 0x30004);
  return *ptr;
};




void read_fadc(unsigned int addr, int num, int ch, int *ch_data){
  unsigned short  *ptr;
  int data[MAX_CLOCK];
```

```
  //int FADC_depth=3000;
  //int MAX_num=8192;
  num = num/2;
  //if(num>FADC_depth){
  //cout << "CH["<< ch <<"]" << endl;
  for(int i=0; i<4096; i++){
    ptr = (unsigned short *)(addr + i*2 + ch*0x2000);
    data[2*i] = (*ptr >> 8) & 0xff;
    data[2*i+1] = *ptr & 0xff;
  }

  num = num*2;
  if(num>TAKE_CLOCK){
    for( int i = num-TAKE_CLOCK; i < num; i ++ ){
      *ch_data = data[i];
      ch_data++;
    }
  }else{
    for( int i = MAX_CLOCK-TAKE_CLOCK+num; i < MAX_CLOCK; i ++){
      *ch_data = data[i];
      ch_data++;
    }
    for( int i = 0 ; i < num; i ++){
      *ch_data = data[i];
      ch_data++;
    }
  }
};


void start_fadc(unsigned int addr){
  unsigned int  *ptr;

  ptr = (unsigned int *)(addr + 0x30008);
  *ptr = 0x0001;
};

void stop_fadc(unsigned int addr){
  unsigned int  *ptr;
```

```
  ptr = (unsigned int *)(addr + 0x30008);
  *ptr = 0x0002;
};

void clear_fadc(unsigned int addr){
  unsigned int  *ptr;

  ptr = (unsigned int *)(addr + 0x30008);
  *ptr = 0x0004;
};

void set_fadc(unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x20000);
  *ptr = 0x0003;
  ptr = (unsigned short *)(addr + 0x20002);
  *ptr = 0x0012;
  ptr = (unsigned short *)(addr + 0x20004);
  *ptr = 0x5678;
  ptr = (unsigned short *)(addr + 0x20006);
  *ptr = 0x0003;

  for(int i=0; i<8; i++){
    ptr = (unsigned short *)(addr + 0x21002 + i*0x200);
    *ptr = 0x0080;
  }
};

unsigned int init_fadc(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  printf("    fadc base address    : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv32d16", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d16 ");
    exit(1);
  }
```

```
  pagesize = getpagesize();
  *mapsize = 0x40000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : fadc failed. ");
    exit(1);
  }
  addr += offset;

  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf("  fadc address on linux  : 0x%x\n", (unsigned int) addr);

  return (unsigned int)addr;
};

int end_fadc(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  printf(" memory : unmapped.\n");
  return 0;

};
```

- chage.cxx - 電荷量を計算するプログラム

```
/* 測定生データから電荷量を求める
   ソートの条件は、maen から WIDTH 以下のデータ
   + 隣りのデータも（連続であるか否か）        */
/* 出力は電荷単位（クーロン値）*/
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#define TAKE_CLOCK 3000 //全体の CLOCK
#define TAKE_CH 4
#define PEAK_LENGTH 600 //PEAk までの DELAY 分

#define WIDTH 50 //条件幅

using namespace std;

double chenge(double charge, int ch);

int main(int argc, char *argv[]){

  if(argc =! 2){
    cout << "入力ファイルを記入してください\n";
    return 1;
  }

  char ifname[32];
  sprintf(ifname, "%s.dat", argv[1]);
  ifstream fin(ifname);

  char ofname[32];
  sprintf(ofname,"charge_%s.dat", argv[1]);
  ofstream fout(ofname);
  cout << ofname << endl;

  double data_num;

  int num, t, ch;
  double charge[TAKE_CH];
  double mean[TAKE_CH];
  int data[TAKE_CH][TAKE_CLOCK];
  int clock;
  int hit=0;
  double hit_1=0;

  // count data_num ////////////////////////////////
  ifstream fin_num(ifname);
```

```
  int num3, num2, ch0, ch1, ch2, ch3, cl;
  num3 = 0;
  num2 = 0;
  while(fin_num >> cl >> ch0 >> ch1 >> ch2 >> ch3){
    num3++;
    if(cl==2999){
      num2++;
    }
  }
  data_num = num2;
  cout << "DATA_NUM  : " << data_num << endl;
  fin_num.close();
  ////////////////////////////////////////////////////

  for(num=0;num<data_num;num++){
    //初期化
    for(ch=0;ch<TAKE_CH;ch++){
      charge[ch] = 0;
      mean[ch] = 0;
    }
    //まずは mean を求める
    for(t=0;t<TAKE_CLOCK-PEAK_LENGTH;t++){
      fin >> clock;//一個目は CLOCK 番号なのでスルー
      for(ch=0;ch<TAKE_CH;ch++){
fin >> data[ch][t];
mean[ch] += (double)data[ch][t];//とりあえず全部足しあわせる
      }
    }
    for(ch=0;ch<TAKE_CH;ch++){
      mean[ch] = mean[ch]/(TAKE_CLOCK-PEAK_LENGTH);//ここで平均にする
    }

    //次に charge を求める
    //とりあえず配列に収容
    for(t=TAKE_CLOCK-PEAK_LENGTH;t<TAKE_CLOCK;t++){
      fin >> clock;
      for(ch=0;ch<TAKE_CH;ch++){
fin >> data[ch][t];
      }
    }
```

```
    //次に条件をチェックし求める
    for(t=TAKE_CLOCK-PEAK_LENGTH;t<TAKE_CLOCK-500;t++){
      //上に大きすぎるのはノイズ混じり
      if( data[0][t] < mean[0]-50 ){
charge[0] = 0;
charge[1] = 0;
charge[2] = 0;
charge[3] = 0;
break;
}
      for(ch=0;ch<TAKE_CH;ch++){
//反応したとみなす条件
if(data[ch][t] > mean[ch]+ WIDTH){
  if((data[ch][t-1] > mean[ch] + WIDTH&&data[ch][t+1] > mean[ch]+ WIDTH)||(data[ch][t-
    charge[ch] = charge[ch] - mean[ch] + data[ch][t];
  }
}
      }
    }

    //hit 数を数える
    for(ch=0;ch<TAKE_CH;ch++){
     if(charge[ch]!=0){hit++;}
    }

    if(hit==1){

      fout << num << " ";
      for(ch=0;ch<TAKE_CH;ch++){
fout << chenge(charge[ch],ch) << " ";
      }
      hit_1++;
      fout << endl;
    }else{
      fout << num << " ";
      for(ch=0;ch<TAKE_CH;ch++){
fout << chenge(charge[ch],ch) << " ";
      }
      fout << endl;
    }
```

```
    hit=0;
  }

  cout << "1hit_data : " << hit_1 << endl;
  double parcent =(double)(hit_1/data_num);
  parcent = parcent*100;
  cout << "parcent   : " << parcent << endl;

  fin.close();
  fout.close();
  return 0;
}

double chenge(double charge, int ch){
  double a[8]; //1ch 当たりの電圧 (mV)
  a[0]=3.644;
  a[1]=3.675;
  a[2]=3.596;
  a[3]=3.685;
  a[4]=0.0;
  a[5]=0.0;
  a[6]=0.0;
  a[7]=0.0;
  double t = 10.0; //1ch 当たりの時間 (nsec)
  double regis = 50; //オシロの抵抗値 (Ω)

  charge = charge * a[ch] * t / regis;

  return charge; // 検出された電荷量 (?)(*10e-12 C)
}
```