GPIB-ENET による RAIDEN の磁場制御

日野原 伸生

2002 年 10 月 26 日

1 GPIB-ENET について

GPIB-ENET は National Instruments 社 (http://www.ni.com/) 製の GP-IB コントローラです。Ethernet を介してリモートコンピュータより GP-IB バスを持つ計測機器と通信することが出来ます。GP-IB コントローラにはこの他に ISA バスや PCI バスとして PC の拡張ボードに直接接続出来るものがありますが、ENET は Ethernet ベースの TCP/IP ネットワーク上で計測器を制御出来るので、GP-IB バスの 総ケーブル長 20m 以内というような制限をくらう必要もありません。また、複数のホストコンピュータから GPIB 機器を共有することも可能です。

現在 ENET のドライバがインストールされているホストは W 実験室にある coral.rcnp.osaka-u.ac.jp です。このコンピュータには saho を通じてログイン出来ます。

```
% ssh saho.rcnp.osaka-u.ac.jp -l username
```

として saho にログインした後に saho より

```
% ssh coral.rcnp.osaka-u.ac.jp -l username
```

としてログインします。coral のユーザ名やパスワードがわからない人は誰かに聞いて下さい。

RAIDEN の磁場制御のために 2 台の GPIB-ENET/100 を使用します。それぞれの名前 (DNS 名) は raiden および gpibhp です。raiden は RAIDEN の D1Magnet の上に置いてあり、GP-IB バスで 3 台の NMR と接続されています。この NMR はそれぞれ Q1、Q2、D2 Magnet の磁場を測定します。

gpibhp は磁石に流す電流の電源装置の下にあります。これには 3 台の電源が接続されており、Q1,Q2,D2Magnet に流す電流や電圧を指定することが出来ます。現在 ENET でコントロール出来るのは補助電源のみです。D1 Magnet には補助電源がないため、主電源を用います。主電源の電流、電圧設定を D1 Magnet にあわせたのちに残りの 3 つの磁石の電流値を ENET で調整します。主電源は加速器のコントロール下にありますので加速器のオペレータの方にコントロールをお願いする形となります。

2 GPIB-ENET ドライバのインストール

National Instruments 社がバイナリを配布しています。Web ページからドライバをダウンロードします。

ftp://ftp.ni.com/support/gpib/linux/nienet-linux-1.2.tar.gz

このドライバは Linux カーネル 2.0,2.2,2.4 で動作するとこのとです。カーネルのバージョンが分からない

場合は/usr/src/linux-**/を見ればわかります。ENET は Vine Linux2.1.5(カーネル 2.2.18) で動作したことは確認しました。

まずは拾ってきたドライバを解凍します。

```
% tar zxvf nienet-linux-1.2.tar.gz
% cd nienet/
```

細かいことは README ファイルを読んで下さい。以下は nienet-linux-1.2 ディレクトリの内容に関する説明です。

2.1 GPIB-ENET ドライバインストール

root で instgpib を実行します。この操作で ugpib.h の LIB へのコピー、/etc/gpibrc の作成、/usr/local/bin/に ibconfenet の追加が行われます。

```
% su
Password:
# ./instgpib
```

2.2 GPIB-ENET のネットワーク設定

EthernetConfig を起動します。

```
% ./EthernetConfig
```

EthernetConfig は同一サブネット内にある GPIB-ENET を探しだして一覧を表示します。下のような画面が表示されます。

```
     IP Address/      Ethernet           Serial
     Hostname         Address            Number    Model          Comment
     ----------       --------           ------    -----          -------
<0>  192.168.2.242    00:80:2F:0A:06:BA  00C6A7D0  GPIB-ENET/100
<1>  192.168.2.226    00:80:2F:0A:09:5D  00C83B45  GPIB-ENET/100
<2>  192.168.2.228    00:80:2F:0A:0B:5E  00C94F7B  GPIB-ENET/100
<3>  192.168.2.227    00:80:2F:0A:0B:68  00C95071  GPIB-ENET/100
<4>  192.168.2.229    00:80:2F:0A:0C:0D  00C9A0A9  GPIB-ENET/100
<5>  192.168.2.230    00:80:2F:0A:0C:3E  00C9EFD1  GPIB-ENET/100


No more devices to display.

You may choose to:
   View Properties of Entry <0>-<5>      <R>efresh List of Devices
```

```
   Display <H>elp for Screen              <E>xit Program
What do you want to do? [0-5,R,H,E]:
```

これらの GPIB-ENET のうち、RAIDEN で磁場制御に使う GPIB-ENET は以下の 2 つです。ファームウェアのバージョンが違うのですが特に問題はないようです。

```
GPIB-ENET/100 Properties
------------------------
     Serial Number:      00C6A7D0
     Ethernet Address:   00:80:2F:0A:06:BA
     Firmware Version:   B.7

     Hostname:           raiden

     ( )Obtain an IP address automatically (DHCP)
     (*)Use the following IP settings:
           IP address:   192.168.2.242
           Subnet mask:  255.255.255.0
           Gateway:      192.168.2.1
           DNS server:   172.16.1.21

     Comment (optional):
```

```
GPIB-ENET/100 Properties
------------------------
     Serial Number:      00C9A0A9
     Ethernet Address:   00:80:2F:0A:0C:0D
     Firmware Version:   B.8

     Hostname:           gpibhp

     ( )Obtain an IP address automatically (DHCP)
     (*)Use the following IP settings:
           IP address:   192.168.2.229
           Subnet mask:  255.255.255.0
           Gateway:      192.168.2.1
           DNS server:   172.16.1.21

     Comment (optional):
```

```
You may choose to:
   Change Host<N>ame                    Use <D>HCP
   Enter <C>omment                      Modify <S>tatic IP Settings
   <E>xit Screen                        Display <H>elp for Screen
```

EthernetConfig を使うとこれらのホスト名、DHCP を使うかどうか、Comment を付けるかどうか、などの設定をすることが出来ます。ここで設定出来るホスト名は DNS 名ではなくて人間が識別するためのものですので通信に使用されることはありません。出荷時はシリアルナンバーになっていました。

また、GPIB-ENET の一覧に通し番号がふられていますが、これは MAC アドレスの順番にならべているだけで、後で出てくる board index とは関係ありません。

設定を変更した場合はこれらの設定を GPIB-ENET に書き込みにいきます。その際に GPIB-ENET の再起動をする必要があります。ですので京都から GPIB-ENET の設定は変更しないようにしましょう。

2.3 各種設定

同一サブネットに複数 GPIB-ENET があるような状況も含めてそれぞれの ENET をユーザプログラムの中で識別するために、ライブラリでは board index という値を用います。どの index がどの ENET に対応するかは/etc/gpibrc を参照します。また、あるボードに GPIB バスでぶらさがっている機器は device index で指定することが出来ます。これも/etc/gpibrc を参照します。バイナリになっているので less などで読むことは出来ません。

/etc/gpibrc の設定を変更するには ibconf を用います。ibconf はドライバインストール時にコピーされる/usr/local/bin/ibconfenet と同一のようです。(未確認) ibconf を開くには root になる必要があります。

```
% su
Password:
# ./ibconf

================================================================================
National Instruments  |    Device map for board gpib0    |       LINUX
================================================================================
         -----------
         | gpib0   |     * Use cursor keys h,j,k,&l to select a device or board.
     ----| raiden  |     * Use control keys below to select desired action.
     |   |_________|     * Use ^B/^F to display maps for other boards.
     |
     |------------------
     |---0 dev1        |---0 dev12          1 dev23          2 dev34
     |---0 dev2        |---0 dev13          1 dev24          2 dev35
     |---0 dev3        |---0 dev14          1 dev25          2 dev36
     |---0 dev4           1 dev15          1 dev26          2 dev37
     |---0 dev5           1 dev16          1 dev27          2 dev38
```

```
     |---0 dev6            1 dev17           1 dev28           2 dev39
     |---0 dev7            1 dev18           2 dev29           2 dev40
     |---0 dev8            1 dev19           2 dev30           2 dev41
     |---0 dev9            1 dev20           2 dev31           2 dev42
     |---0 dev10           1 dev21           2 dev32           3 dev43
     |---0 dev11           1 dev22           2 dev33           3 dev44
================================================================================
                                        |
================================================================================
^Q: help       ^R: rename       ^T: (dis)connect       ^I: edit        ^O: exit
```

ここではボードのホスト名を変えることが出来ます。GPIB-ENET は DNS を使わず、このホスト名は/etc/hostsに IP アドレスと対応された形で書かれている必要があります。

```
# vi /etc/hosts

192.168.2.242     raiden.rcnp.osaka-u.ac.jp raiden
192.168.2.229     gpibhp.rcnp.osaka-u.ac.jp gpibhp
```

実際は gpib0 が raiden、gpib1 が gpibhp と設定してあります。そのほかボードやデバイスの細かい設定を行うことが出来ます。以上で設定は終わりです。

2.4 ibic

ibic(Interface Bus Interactive Control Program) は対話的にデバイスと通信することの出来るプログラムです。

3 GPIB プログラムの記述

4 C 言語ライブラリ

GPIB 機器と通信するために、C 言語のライブラリ ugpib.h が用意されています。このライブラリで定義されているよく使う関数です。詳しい定義については NI-488 のマニュアルを参照して下さい。

```
int ibdev(int bdIndx, int pad, int sad, int tmo, int eot, int eos)
```

デバイスを開き、初期化します。

INPUT

bdIndx　アクセスするデバイスのぶらさがっている GPIB ボードの index

pad　デバイスのプライマリ GPIB アドレス

sad　デバイスのセカンダリ GPIB アドレス

tmo　I/O タイムアウト値

eot　デバイスの EOI モード

eos　デバイスの EOS 文字列モード

OUTPUT

返り値はデバイスデスクリプタ (ud) か-1 です。

デバイスを開き、初期化します。

```
int ibsre(int ud,int v)
```

INPUT

ud　デバイスデスクリプタ

v　パラメータ

OUTPUT

ibsta の値を返します。

REN(Remote ENable) ラインをセット、クリアします。もし v がゼロでなければ ud で指定されたデバイスと REN(Remote ENable) ラインを確立します。ゼロの場合は信号をクリアします。

```
int ibclr(int ud)
```

INPUT

ud　デバイスデスクリプタ

OUTPUT

ibsta の値を返します。

デバイスにクリアーコマンドを送ります。

```
ibwrt(int ud, void *wrtbuf, long count)
```

INPUT

ud　デバイスデスクリプタ

wrtbuf　書き込むバイトを含むバッファのアドレス

count　書き込むバイトのバイト長

OUTPUT

ibsta の値を返します。

デバイスに wrtbuf で指定されたメモリの場所から count 文字数分だけの文字列型データを書きこみます。データはダブルクオートでで囲みます。

```
ibrd(int ud,rdbuf *, long count)
```

INPUT

ud　デバイスデスクリプタ

count　GPIB から読みだすデータのバイト長

OUTPUT

rdbuf　読みだしたデータを格納するためのバッファのアドレス

RETURN　返り値は ibsta の値です。

デバイスから count 文字数分だけデータを読み込み、rdbuf で特定されるバッファに格納します。

4.1　ソースのコンパイル

コンパイルは

```
% gcc test.c -o test cib.o
```

などとしてオブジェクトファイル cib.o をリンクして下さい。

5　NMR 読みだし

NMR は gpib0 である raiden に接続されています。NMR は Primary Address 5,6 につながっています。どのアドレスの NMR がどの磁石の磁場を読みに行っているのかは私はわかりません。

磁場の値を読むためには D0 というコマンドをデバイスに送ります。

以下が磁場の値を読む read_nmr.c プログラムです。DO ではカンマで区切られた 3 つの値を返します。これらの値がそれぞれ何を示しているのかは NMR のマニュアルを参照して下さい。

```
/*************** read_nmr.c ******************/

#include <stdio.h>
#include "ugpib.h"
#include <string.h>

int dev=0;

int main(){
  char res[1024];
  char cmd[20];
char s;
  int PrimaryAddress;

  printf("Enter the Address.\n");
  scanf("%d",&PrimaryAddress);

  dev=ibdev(0,PrimaryAddress ,0,T10s,1,0);

  /*usleep(1000L);  this delay is needed, but why ... */

  ibsre(dev,1);  /* set the remote enable (REN) line  */
```

GPIB Address	model	Magnet
1	6652A	Q1
2	6654A	Q2
3	6654A	D2

表 1: 電源の型番と制御出来る磁石

```
  ibclr(dev);    /* clear a specific device (dev)     */

  /*usleep(1000L);  this delay is needed, but why ... */

 for (;;){
  sprintf(cmd,"D0");
  ibwrt(dev,cmd,strlen(cmd));
  ibrd(dev,res,40);res[ibcnt]=0;
  printf("Bytes: %d\n",ibcnt);
  printf("%s\n",res);
  sleep(1);
  }
  exit(0);
}
```

6 電流設定

電流設定には Agilent Technologies の 6652A/6654A SYSTEM SUPPLY を用います。電圧の限界値を決めた上での電流値のセットおよび電流の限界値を決めた上での電圧値のセット、電流の反転などを行うことが出来ます。P4 部屋にマニュアルをコピーしておきましたので詳しくはそれを見て下さい。

6652A はセット出来る電圧、電流は 0-20V/0-25A 6654A はセット出来る電圧、電流は 0-60V/0-9A となっています。

つまり、上流にある Q1 磁石は focal plane 上での広がりを決定しますので電流が多く流せるようになっています。

これらの電源を用いて Q1、Q2、D2 Magnet の磁場を補助的に変更することが出来ます。

電源の種類とセット出来る磁石、デバイスの GPIB Primary Address との関係は Table6 を参照してください。

6.1 フロントパネルでの設定

設定は GPIB を介さずとも電源のフロントパネルで設定出来ます。こちらの方が GP-IB を介する操作よりも直感的ですので 1 度やってみて下さい。電源の ON/OFF 以外は全て GPIB でもコントロール出来ます。

電源を入れた初期段階では電圧値、電流値、は表のように設定してあります。これは電源の暴走をふせぐ為です。そして Disable モードになっているので実際には電圧は出力されません。Disable モードを解除するにはフロントパネルの Output ON/OFF ボタンを押します (GPIB でも操作出来ます) すると CV モードに移行します。

この電源は以下のような動作をします。

初期状態では電圧が 0、電流は小さい値に設定されていますので電圧が 0 となっています。(CV モード)

片方の値を変更すると、電源、電圧ともにセットされた値を超えない組み合せの方のどちらかの値になろうとします。例をだしますと、電圧を 5V にしますと、電流が小さいので 5V になることは出来ず、電流値が実現されます。次に電流値を 20A のような非常に大きい値にしますとこの電流値は 5V では実現できないので 5V を保ち、電流値はそれに対応した電流値となります。

セットした電流値を実現している状態を CC(Constant-Current) モード、セットした電圧値を実現している状態を CV(Constant-Voltage) モードと呼びます。

電流の設定は Current 5.0 Enter と押せば 5A に設定出来ます。電流は Voltage 5.0 Enter と押せば 5V に設定出来ます。

OV はかけられる電圧の上限を示します。デフォルトでは電源の規格程度になっています。

6.2 GPIB による電源操作

6.3 コマンド

電源を設定する各種コマンドです。これらのコマンドを ibwrt で書き込んだあと、値を返すものであれば ibrd を使って読みこみます。サンプルプログラムやマニュアルを参照して下さい。

```
OUTP ON
OUTP OFF
OUTP?
```

パネルの OUTPUT ON/OFF に対応します。電源の出力を可否を設定します。disable(OFF) の状態は出力される電源電圧が 0 で、電流は電源の型番に依存する最小電流となります。OUTP?は電源の出力状態を 0 または 1 で返します。

```
OUTP:REL ON
OUTP:REL OFF
\begin{verbatim}
```

オプションのリレーコネクタがついていれば使えます。RAIDEN のはついてます。
ON にすると、リレーコンタクトがある状態になります。OFF にすると
リレーコンタクトがない状態になります。リレーコネクタを使えば電圧の向きを
反転させることが出来ます。補助電源は主電源の調整として使いますので
電圧の向きを反転させる必要があります。

```
\begin{verbatim}
```

```
OUTP:REL:POL NORM
OUTP:REL:POL REV
```

NORMAL に設定するとリレーの出力と電源の出力を同じにします。REVERSE に設定するとリレーの出力の符号を電源の出力する符号と逆向きにします。もし、リレーコマンドを送ったとき OUTP が ON になっていれば、リレーが符号を変えている間、電源の出力電圧は 0 に設定されます。

```
CURR 1.0
CURR?
```

上は電流値 1.0A にを設定します。下は現在設定されている電流値を読みます。

```
VOLT 1.0
VOLT?
```

上は電圧を 1.0V に設定します。下は現在設定されている電圧値を読みます。

```
MEAS:CURR?
MEAS:VOLT?
```

現在出力されている電流、電圧値を読みます。

6.3.1 サンプルプログラム

過去のものの流用も含め､サンプルプログラムとして以下のようなものを coral の /p42002/hinohara/supply/ に作成、動作テストしました。全てのプログラムは PrimaryAddress を聞いてきます。

enable.c OUTPUT を ON にします。disable.c OUTPUT を OFF にします。currread.c 電流値を読みこみます。voltread.c 電圧値を読みこみます。polarity.c 電圧の負号反転を設定します。currset.c 電流値をセットします。voltset.c 電圧値をセットします。status.c 電源の設定値などの状態を表示します。recall.c 出荷状態にもどします。(必ず最後に実行してください。)

6.3.2 プログラム例：currread.c

```
#include <stdio.h>
#include "ugpib.h"
#include <string.h>

/************* currread.c ***************/

void GpibError3(char *msg);        /* Error function declaration            */
int Device3 = 0;                   /* Device unit descriptor*/
int BoardIndex3 = 1;             /* Interface Index(GPIB0=0,GPIB1=1,etc.)  */
```

```
int main() {
int i;
  int   PrimaryAddress = 3 ;     /* Primary address of the device           */
  int   SecondaryAddress = 0;    /* Secondary address of the device         */
  char  Buffer[1024];            /* Read buffer                             */
  char  cmd[20];
  char  command[20];
  char  cm[20];
  char  exe[20];

  /*****************************************************************************
   * Initialization - Done only once at the beginning of your application.
   *****************************************************************************/
  //  printf("Enter command\n");
  //fgets(command,sizeof(command),stdin);
  //  command[strlen(command)-1] = '\0';
  //  strcpy(exe, command);
  //  strcat(exe, "?");
for( ; ;){
for(i = 0;i<1025;i++){
        Buffer[i] = 'a';
}
  Device3 = ibdev(               /* Create a unit descriptor handle         */
                BoardIndex3,     /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
                PrimaryAddress,  /* Device primary address                  */
                SecondaryAddress,/* Device secondary address                */
                T10s,            /* Timeout setting (T10s = 10 seconds)     */
                1,               /* Assert EOI line at end of write         */
 if (ibsta & ERR) {              /* Check for GPIB Error                    */
    GpibError3("ibdev Error");
  }

  ibclr(Device3);                /* Clear the device        */
  if (ibsta & ERR) {
    GpibError3("ibclr Error");
  }
  /*****************************************************************************
   * Main Application Body - Write the majority of your GPIB code here.
   *****************************************************************************/
```

```
  sprintf(cmd,"SYST:LANG TMSL\n");
  ibwrt(Device3,cmd,strlen(cmd));
/*  printf("%s",cmd);*/
  if (ibsta & ERR) {
    GpibError3("ibwrt2 Error");
  }

   sprintf(cm,"MEAS:CURR?",command);
   ibwrt(Device3, cm, strlen(cm));
   // sprintf(cm,"OUTP:REL:POL NORM",command);
 // ibwrt(Device3, cm, strlen(cm));  /* Send the command   */
  if (ibsta & ERR) {
    GpibError3("ibwrt Error");
  }


 /* printf("%s\n",cm);*/

   ibrd(Device3, Buffer, 100);    /* Read up to 100 bytes from the device    */
   if (ibsta & ERR) {
     GpibError3("ibrd Error");
     }

  Buffer[ibcntl] = '\0';         /* Null terminate the ASCII string         */

  printf("%s\n", Buffer);        /* Print the device identification         */

   ibonl(Device3, 0);              /* Take the device offline*/
 if (ibsta & ERR) {
     GpibError3("ibonl Error");
    }
 ibonl(BoardIndex3, 0);          /* Take the interface offline              */
  if (ibsta & ERR) {
    GpibError3("ibonl Error");
  }
sleep(1);

/*****************************************************************************
 *                      Function GPIBERROR
 * This function will notify you that a NI-488 function failed by
```

```
 * printing an error message.  The status variable IBSTA will also be
 * printed in hexadecimal along with the mnemonic meaning of the bit
 * position. The status variable IBERR will be printed in decimal
 * along with the mnemonic meaning of the decimal value.  The status
 * variable IBCNTL will be printed in decimal.
 *
 * The NI-488 function IBONL is called to disable the hardware and
 * software.
 *
 * The EXIT function will terminate this program.
 ************************************************************************/
void GpibError3(char *msg) {

  printf ("%s\n", msg);

  printf ("ibsta = &H%x  <", ibsta);
  if (ibsta & ERR )  printf (" ERR");
  if (ibsta & TIMO)  printf (" TIMO");
  if (ibsta & END )  printf (" END");
  if (ibsta & SRQI)  printf (" SRQI");
 if (ibsta & RQS )  printf (" RQS");
  if (ibsta & CMPL)  printf (" CMPL");
    if (ibsta & LOK )  printf (" LOK");
    if (ibsta & REM )  printf (" REM");
    if (ibsta & CIC )  printf (" CIC");
    if (ibsta & ATN )  printf (" ATN");
    if (ibsta & TACS)  printf (" TACS");
    if (ibsta & LACS)  printf (" LACS");
    if (ibsta & DTAS)  printf (" DTAS");
    if (ibsta & DCAS)  printf (" DCAS");
    printf (" >\n");

    printf ("iberr = %d", iberr);
    if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
    if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
    if (iberr == ENOL) printf (" ENOL <No Listener>\n");
    if (iberr == EADR) printf (" EADR <Address error>\n");
    if (iberr == EARG) printf (" EARG <Invalid argument>\n");
    if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
    if (iberr == EABO) printf (" EABO <Operation aborted>\n");
```

```
    if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
    if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
    if (iberr == ECAP) printf (" ECAP <No capability>\n");
 if (iberr == EFSO) printf (" EFSO <File system error>\n");
    if (iberr == EBUS) printf (" EBUS <Command error>\n");
    if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
    if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
    if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

    printf ("ibcntl = %ld\n", ibcntl);
    printf ("\n");

    /* Call ibonl to take the device and interface offline */
    ibonl (Device3,0);
    ibonl (BoardIndex3,0);

    exit(1);
}
```

6.4 注意!：電源の初期化

セットしますと実際に電流が流れますので終わる前に必ず recall.c を全ての電源に対して実行してください。

なお、これは location 0 に save された状態をロードしますので location 0 の状態は変更しないようにして下さい。電源が立ち上がった時も location 0 をロードします。