実用EPICS入門-6
技術部専門講習

飛山真理
帯名崇

device support

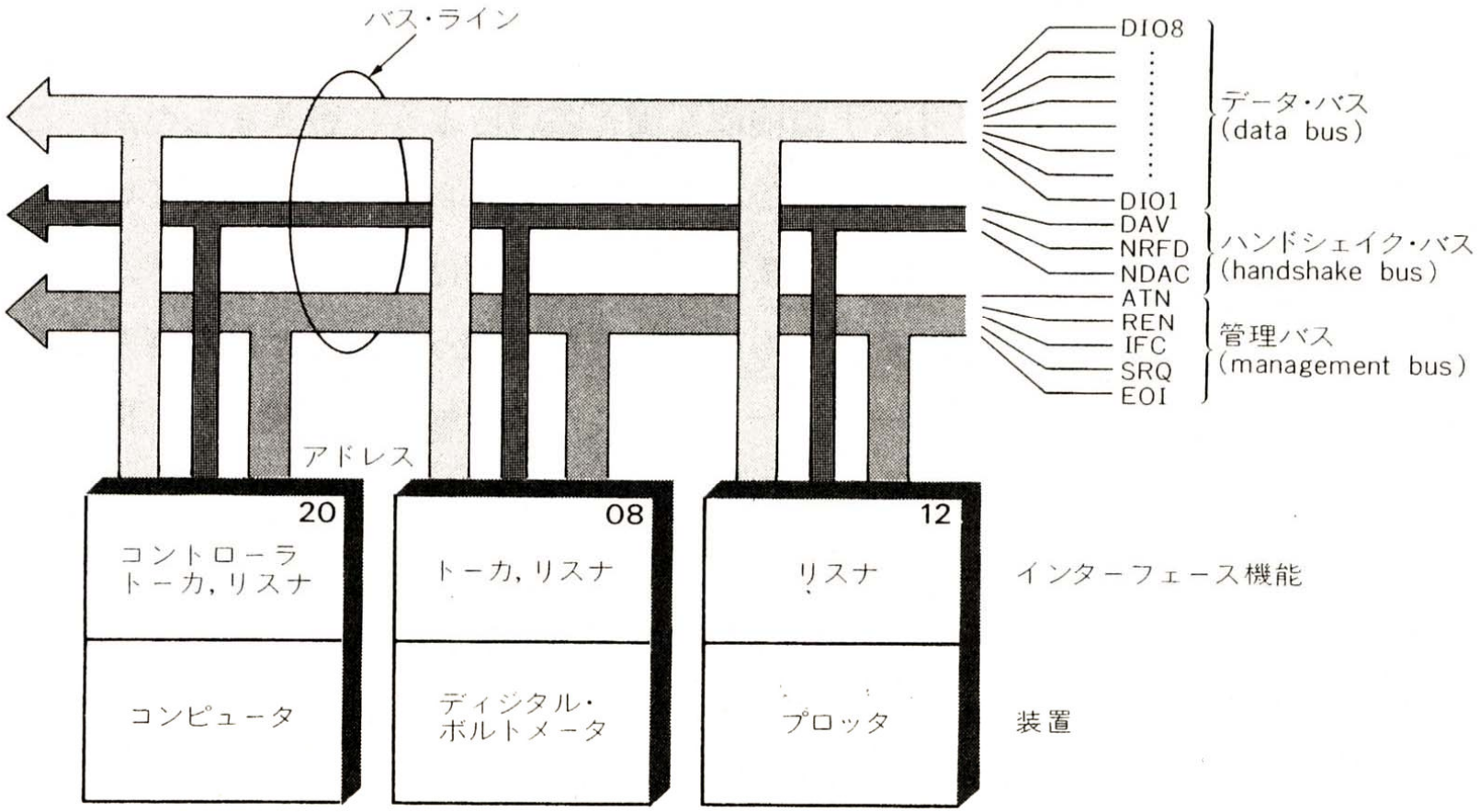
- ハードウエアとEPICSの世界をつなぐもの
  - のうち、比較的上位の部分
  - GP-IB自体とか、Ethernet自体を扱う部分は素人の手に負えないので、専門家が作っている
- よく使われているもの
  - GP-IB
  - LAN直接(message baseのものと、もろバイナリなもの)
  - VME/VXI

GP-IB

- もともとHP社が決めたHP-IBからはじまり、IEEE-488で標準化
- ほとんど神代の昔に決まった規格だが、いまだにしぶとく生き残っている(しかしさすがに衰退しつつある)
- 大手メーカーの製品でも、規格をちゃんと守っているものは意外と少ない
  - 特注品は、まず完全規格外と思って良い

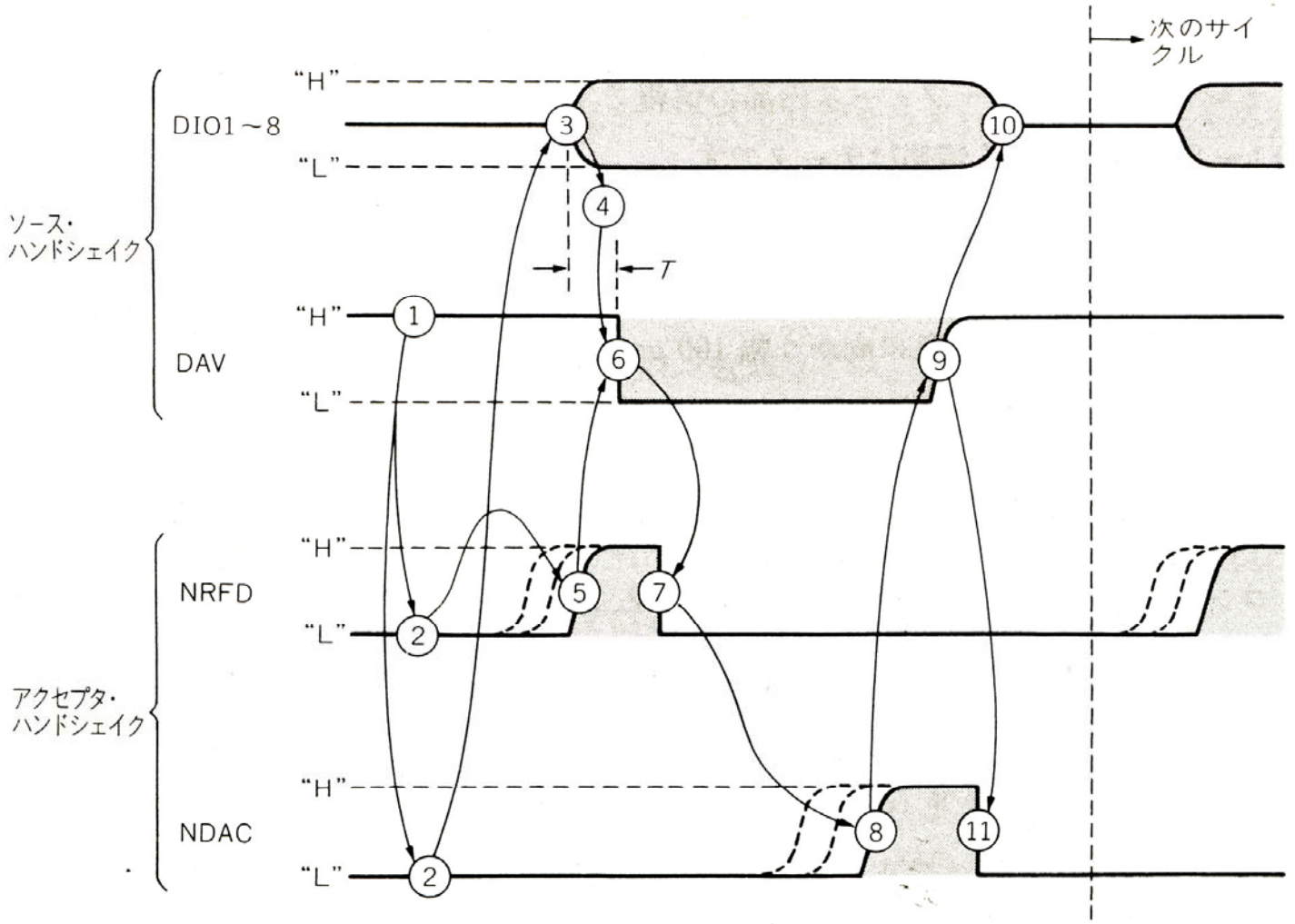
GP-IB

- データバス8本(DIO1～DIO8)
- ハンドシェイクバス3本(DAV,NRFD,NDAC)
- 管理バス5本(ATN,REN,IFC,SEQ,EOI)
  - 全てsingle-endのTTLレベルなのでノイズにやたらに弱い
  - 転送速度は規格上は1MB/s以上いくが、各種問題のため実用上は遙かに遅い
  - 参考:IEEE-448(GPIB)とその応用:岡村迪夫著,
    - CQ出版社　ISBN4-7898-3663-0


バス・ライン
DIO8
DIO1
データ・バス
(data bus)
DAV
NRFD
NDAC
ハンドシェイク・バス
(handshake bus)
ATN
REN
IFC
SRQ
EOI
管理バス
(management bus)
アドレス
20
コントローラ
トーカ, リスナ
コンピュータ
08
トーカ, リスナ
ディジタル・
ボルトメータ
12
リスナ
プロッタ
インターフェース機能
装置

3線ハンドシェイク

図 4.4　3 線ハンドシェイクの信号タイミング



ハンドシェイク例(Talkerが送るデータを複数のListnerが受け取る)

- はじめはDAV=H、NRFD,NDAC=L
- accepter側は,data受け入れ可ならNRFD(ready for data)=Hにする
  - open collectorなので、どれかのlistnerがlowなら結果としてlowになる
- source側は、NDAC=Lを確認して、データラインにデータを(あらかじめ)出力、その後NRFDがHを確認してDAV(data valid)をLにする
  - NRFDがH出なければ、DAV=Hで待ち続ける

ハンドシェイク(続き)

- accepterはDAV=Lを検出するとNRFDをLにして、バスライン読み込みを開始する。
- 読み込みが終わると、NDAC(data accepted)をHにする。
  - これもopen collectorなので、どれかが終わっていなければ結果はLのまま。
- NDACがHになると、全てのaccepterの受け入れが完了したので、sourceはDAVをHにし、バスライン上のデータ出力を終了する(HiZにする)
- accepterはDAVがHに戻ったことを確認して、NDACをLに戻し、次のデータの受け入れ準備を開始する。
- 頭に戻って、次のサイクルが開始される

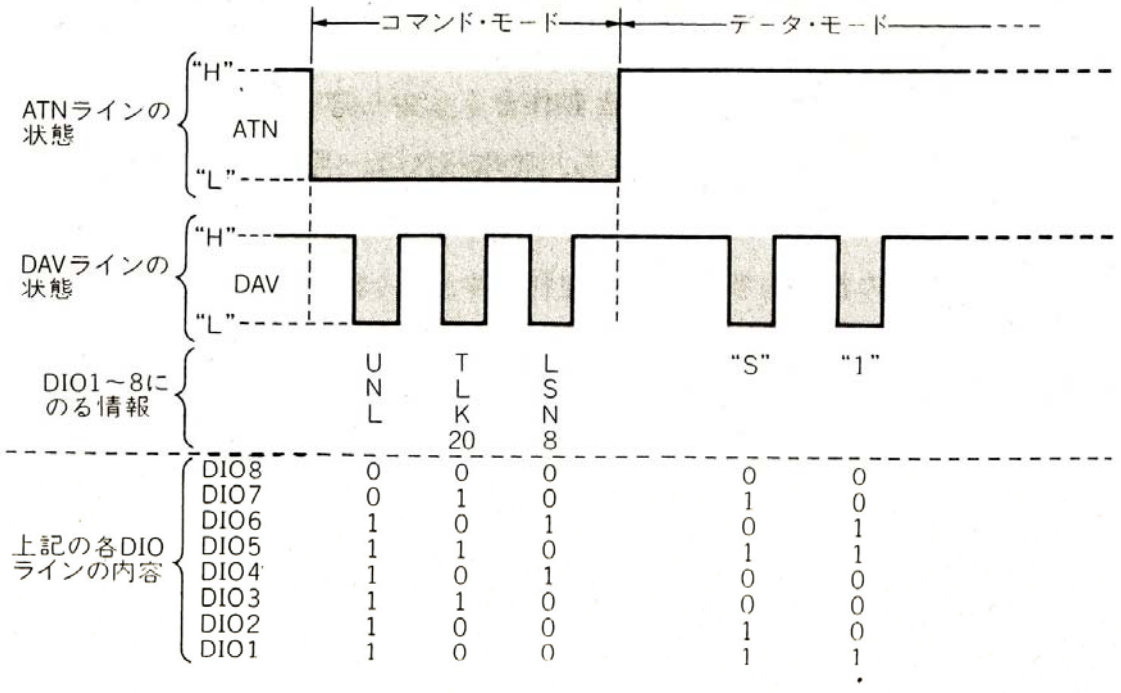
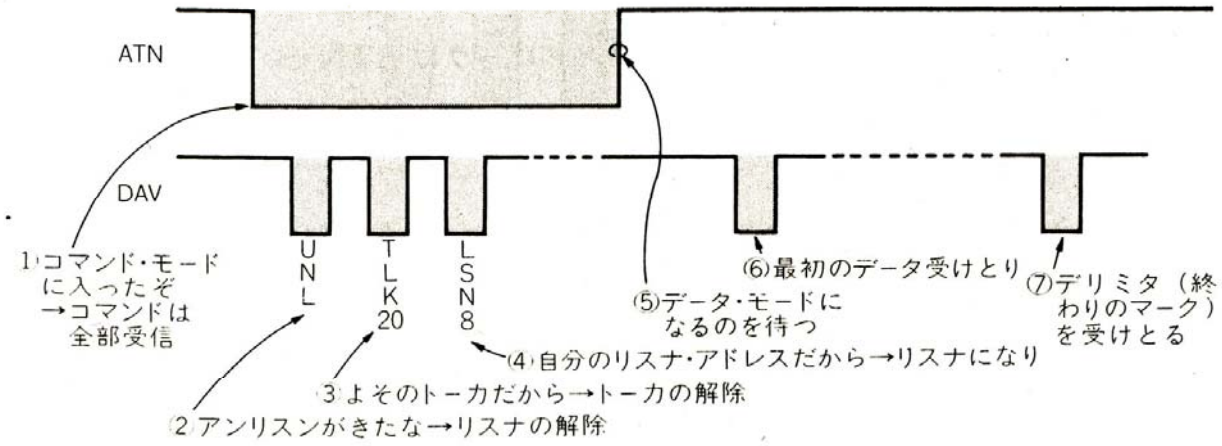


図 1.10　アドレス 8 番のインターフェースの動作



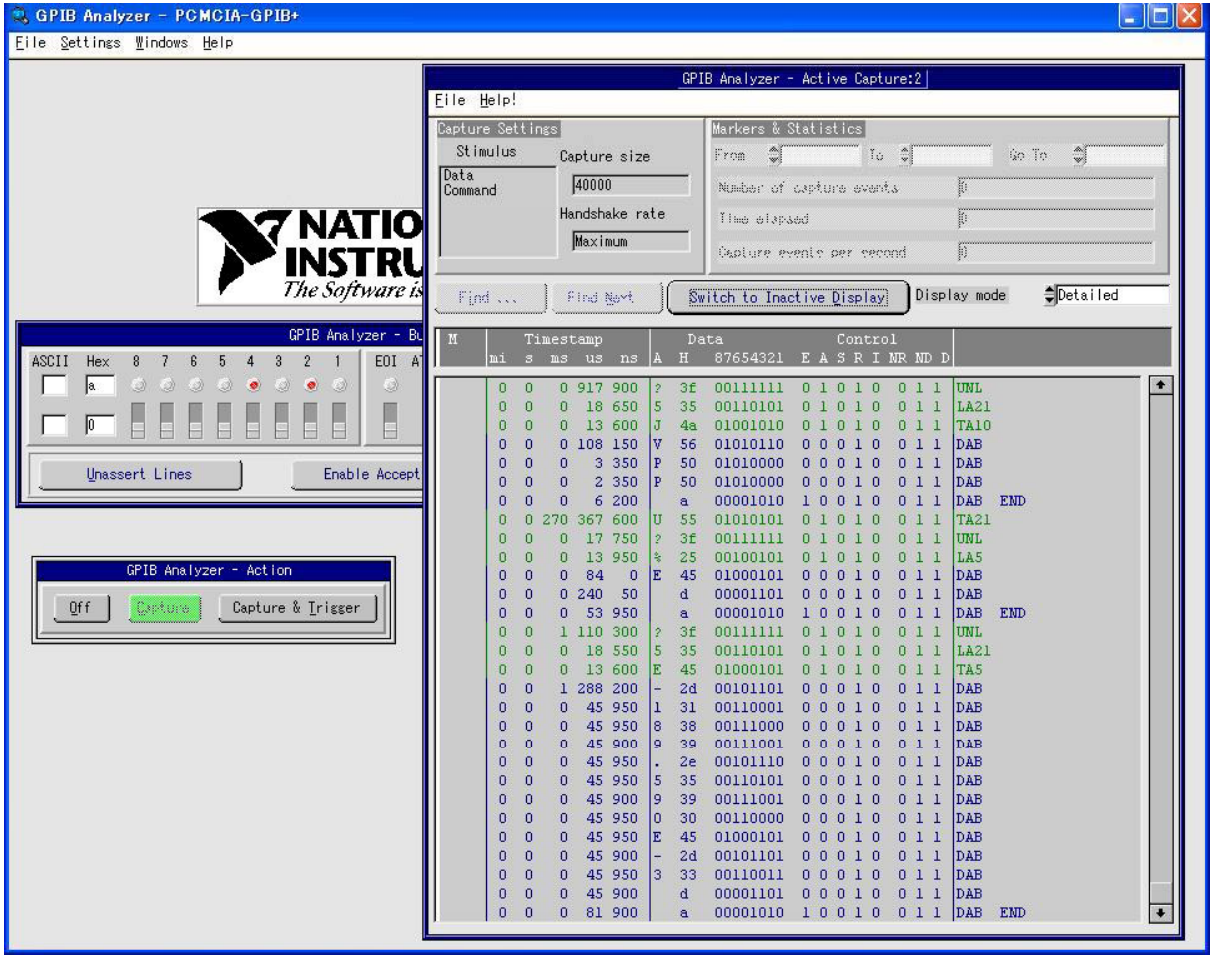
データ列の終わり

- データ終わりを定義しておく必要がある
  - EOIライン(ハードラインなので、データによらずOK)
  - CR LF+EOI(古い機器で多いタイプ)
  - LF+EOI
  - 他にも、機器側で設定できるものもある

問題のあるGP-IB機器

- デリミタでEOIが出せないもの
  - LAN-GPIBでは使用不可、NI GP-IBボードなら可
- ちゃんとハンドシェイクしないもの
  - デバイスサポート側で巨大なwaitを入れる必要有り、要高等技術

GP-IBアナライザ


GPIB Analyzer - PCMCIA-GPIB+
File Settings Windows Help
GPIB Analyzer - Active Capture:2
File Help!
Capture Settings
Stimulus
Data
Command
Capture size
40000
Handshake rate
Maximum
Markers & Statistics
From
To
Go To
Number of capture events
Time elapsed
Capture events per second
Find ...
Find Next
Switch to Inactive Display
Display mode
Detailed
M
Timestamp
Data
Control
mi s ms us ns
A H 87654321 E A S R I NR ND D
0 0 0 917 900 ? 3f 00111111 0 1 0 1 0 0 1 1 UNL
0 0 0 18 650 5 35 00110101 0 1 0 1 0 0 1 1 LA21
0 0 0 13 600 J 4a 01001010 0 1 0 1 0 0 1 1 TA10
0 0 0 108 150 V 56 01010110 0 0 0 1 0 0 1 1 DAB
0 0 0 3 350 P 50 01010000 0 0 0 1 0 0 1 1 DAB
0 0 0 2 350 P 50 01010000 0 0 0 1 0 0 1 1 DAB
0 0 0 6 200 a 00001010 1 0 0 1 0 0 1 1 DAB END
0 0 270 367 600 U 55 01010101 0 1 0 1 0 0 1 1 TA21
0 0 0 17 750 ? 3f 00111111 0 1 0 1 0 0 1 1 UNL
0 0 0 13 950 % 25 00100101 0 1 0 1 0 0 1 1 LA5
0 0 0 84 0 E 45 01000101 0 0 0 1 0 0 1 1 DAB
0 0 0 240 50 d 00001101 0 0 0 1 0 0 1 1 DAB
0 0 0 53 950 a 00001010 1 0 0 1 0 0 1 1 DAB END
0 0 1 110 300 ? 3f 00111111 0 1 0 1 0 0 1 1 UNL
0 0 0 18 550 5 35 00110101 0 1 0 1 0 0 1 1 LA21
0 0 0 13 600 E 45 01000101 0 1 0 1 0 0 1 1 TA5
0 0 1 288 200 - 2d 00101101 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 1 31 00110001 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 8 38 00111000 0 0 0 1 0 0 1 1 DAB
0 0 0 45 900 9 39 00111001 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 . 2e 00101110 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 5 35 00110101 0 0 0 1 0 0 1 1 DAB
0 0 0 45 900 9 39 00111001 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 0 30 00110000 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 E 45 01000101 0 0 0 1 0 0 1 1 DAB
0 0 0 45 900 - 2d 00101101 0 0 0 1 0 0 1 1 DAB
0 0 0 45 950 3 33 00110011 0 0 0 1 0 0 1 1 DAB
0 0 0 45 900 d 00001101 0 0 0 1 0 0 1 1 DAB
0 0 0 81 900 a 00001010 1 0 0 1 0 0 1 1 DAB END
GPIB Analyzer - Bu
ASCII Hex 8 7 6 5 4 3 2 1 EOI A
Unassert Lines
Enable Accept
GPIB Analyzer - Action
Off
Capture
Capture & Trigger

EPICSの世界での普通の使い方

- VMEにNIのGP-IBコントローラを入れ、そこからコントロール(VxWorks)
- LAN-GPIB(Aginet)をEPICS Asynドライバーを使ってLinux IOCから制御する
- いずれも、GP-IBを使う仕組み自体はすでに用意されているので、ユーザーは各機械に合わせたデバイスサポート(+データベース)を用意する。

今回の実習では

- LAN-GPIBを使って、GP-IB機器を制御するデバイスサポートを作ります
  - DMM(Advantest R6551)
  - Function generator(Agilent 33220A)
- 例として、Advantest R6551デジタルマルチメータを制御するデバイスサポートを作ってみます

implementする機能

- 初期化(DCVに設定)
- DCVレンジ設定
- サンプルrate設定
- ホールド
- 値の読み込み

GP-IBコマンド

- 初期化 DCVに設定 :　F1
- DCVレンジ
  - AUTO、300mV、3000mV、30V、300V、1000V

    F1,R0　F1,R3　F1,R4　F1,R5 F1,R6 F1,R7
- Sample rate
  - Fast, Middle,Slow

    RP1　PR2　PR3
- Hold

  M0 M1
- Read

    E

準備1

- cd ~/epicsApp/configure
- RELEASEファイルを編集して
  ASYN=$(EPICS_BASE)/../modules/soft/asyn/4-8
- cd ~/epicsApp
- make distclean

準備2

- cdでホームディレクトリ(/home/sluser)に移動
- mkdir tmp
- cd tmp
- /opt/epics/R3.14.9/modules/soft/asyn/4-8/bin/¥
  linux-x86/makeSupport.pl –t devGpib R6551
- これで、~/tmpディレクトリに

Makefile R6551Sup configure documentation

のfile/directoryができているはず。

準備3

- cd ~/epicsApp/trainApp/src

必要なファイルだけコピーします

- cp ~/tmp/R6551Sup/devR6551.c .
- cp ~/tmp/R6551Sup/devR6551.dbd .

準備4

- Makefileを編集

  train_DBD += drvAsynIPPort.dbd

  train_DBD += drvVxi11.dbd

  train_DBD += devR6551.dbd

  train_SRCS += devR6551.c

  train_LIBS += asyn

devR6551.cを編集

```
/*
 * R6551 device support
 */

#include <epicsStdio.h>
#include <devCommonGpib.h>

/*****************************************************************************
 *
 * The following define statements are used to declare the names to be used
 * for the dset tables.
 *
 * A DSET_AI entry must be declared here and referenced in an application
 * database description file even if the device provides no AI records.
 *
 *****************************************************************************/
```

```
#define DSET_AI     devAiR6551
#define DSET_AO     devAoR6551
#define DSET_BI     devBiR6551
#define DSET_BO     devBoR6551
#define DSET_EV     devEvR6551
#define DSET_LI     devLiR6551
#define DSET_LO     devLoR6551
#define DSET_MBBI   devMbbiR6551
#define DSET_MBBID  devMbbidR6551
#define DSET_MBBO   devMbboR6551
#define DSET_MBBOD  devMbbodR6551
#define DSET_SI     devSiR6551
#define DSET_SO     devSoR6551
#define DSET_WF     devWfR6551

#include <devGpib.h> /* must be included after DSET defines */

#define TIMEOUT     1.0    /* I/O must complete within this time */
#define TIMEWINDOW  2.0    /* Wait this long after device timeout */
```

```
/*****************************************************************
 * Strings used by the init routines to fill in theznam,onam,...
 * fields in BI and BO record types.
 *****************************************************************/

static char  *freeHoldList[] = { "Free","Hold" };
static struct devGpibNames   freeHold = {
    2,freeHoldList,0,1 };
```

- BI、BOレコード用の名前リスト、ZNAM、ONAMフィールドを定義
- devGpibNames構造体の1個目は要素数、2個目は名前リスト、3個目はmbbo、mbbiレコード時のnobtフィールド

```
/*****************************************************************************
 * Structures used by the init routines to fill in the onst,twst,... and the
 * onvl,twvl,... fields in MBBI and MBBO record types.
 *
 * Note that the intExtSsBm and intExtSsBmStop structures use the same
 * intExtSsBmStopList and intExtSsBmStopVal lists but have a different number
 * of elements in them that they use... The intExtSsBm structure only represents
 * 4 elements,while the intExtSsBmStop structure represents 5.
 *****************************************************************************/
static char *DCVRangeList[] = {
  "AUTO","300mV","3000mV","30V","300V","1000V"};
static unsigned long DCVRangeVal[] = {0,1,2,3,4,5};
static struct devGpibNames DCVRange = {6, DCVRangeList,
     DCVRangeVal,3};
static char *SampleRateList[] = {
  "Fast","Middle","Slow"};
static unsigned long SampleRateVal[] = {0,1,2};
static struct devGpibNames SampleRate = {3, SampleRateList,
     SampleRateVal,2};
```

- mbbi、mbboレコード用の名前リスト
  - onvlから必要な長さまで
- devGpibNames構造体の意味は同じ
  - 多く定義していても、長さを制限すること可
  - nobtフィールドの設定は、要素数で決まる
    - 2個なら1ビット、3個なら2ビット、7個までは3ビットとか

```
/*****************************************************************************
 * String arrays for EFAST operations. The last entry must be 0.
 *
 * On input operations,only as many bytes as are found in the string array
 * elements are compared.  Additional bytes are ignored.
 * The first matching string  will be used as a match.
 *
 * For the input operations,the strings are compared literally!  This
 * can cause problems if the instrument is returning things like ¥r and ¥n
 * characters.  When defining input strings so you include them as well.
 *****************************************************************************/
static char *freeHoldSet[] = {"M0¥r¥n","M1¥r¥n",0};
static char *DCVRangeSet[] =
    {"F1,R0¥r¥n","F1,R3¥r¥n","F1,R4¥r¥n","F1,R5¥r¥n","F1,
    R6¥r¥n","F1,R7¥r¥n",0};
static char
    *SampleRateSet[]={"PR1¥r¥n","PR2¥r¥n","PR3¥r¥n",0};
```

EFASTリスト

- MBBO、MBBI、BO、BIで実際機器に送られる、あるいは機器から帰ってくるコマンドのリスト
- デリミタがある場合、ここに書いておく(EOIは自動で付くので、書く必要なし)

```
/*****************************************************************************
 * Array of structures that define all GPIB messages
 * supported for this type of instrument.
 *****************************************************************************/
static struct gpibCmd gpibCmds[] = {
    /* Param 0 -- init DVM */
    {&DSET_BO, GPIBCMD, IB_Q_LOW, "F1¥r¥n", NULL, 0, 200, NULL, 0, 0, NULL,
      NULL, NULL},
    /* Param 1 - DCV Range */
    {&DSET_MBBO, GPIBEFASTO, IB_Q_LOW,NULL, NULL, 0, 80, NULL, 0, 0,
      DCVRangeSet, &DCVRange, NULL},
    /* Param 2 - Sample Rate */
    {&DSET_MBBO, GPIBEFASTO,
      IB_Q_LOW,NULL,NULL,0,80,NULL,0,0,SampleRateSet, &SampleRate,NULL},
    /* Param 3 - Hold on off */
    {&DSET_BO, GPIBEFASTO, IB_Q_LOW,NULL,NULL,0,80,NULL,0,0,freeHoldSet,
      &freeHold, NULL},
    /* Param 4 - Read */
    {&DSET_AI, GPIBREAD,IB_Q_LOW,"E¥r¥n","%lf¥r¥n",
      0,511,NULL,0,0,NULL,NULL,NULL}
};
```

GP-IBコマンドなどを書くところ

- gpibCmd構造体に書いた順番にデータベースで使うパラメータが割り振られる(自分で管理すること)
- {&DSET_BO, GPIBCMD, IB_Q_LOW, "F1¥r¥n", NULL, 0, 200, NULL, 0, 0, NULL, NULL, NULL},

```
typedef struct gpibCmd {
    gDset *dset;        /* used to indicate record type supported */
    int type;           /* enum - GPIBREAD...GPIBSRQHANDLER */
    short pri;          /* request priority IB_Q_LOW, IB_G_MEDIUM, or IB_Q_HIGH */
    char *cmd;          /* CONSTANT STRING to send to instrument */
    char *format;       /* string used to generate or interpret msg */
    int rspLen;         /* room for response error message */
    int msgLen;         /* room for return data message length */
     /*convert is optional custom routine for conversions */
    int (*convert) (gpibDpvt *pgpibDpvt,int P1, int P2, char **P3);
    int P1;             /* P1 plays a dual role: */
                         /* For EFAST it is set internally to the  */
                        /* number of entries in the EFAST table */
                        /* For convert it is passed to convert() */
    int P2;             /* user defined parameter passed to convert() */
    char **P3;          /* P3 plays a dual role: */
                        /* For EFAST it holds the address of the EFAST table */
                        /* For convert it is passed to convert() */
    devGpibNames *pdevGpibNames; /* pointer to name strings */
    char * eos;         /* input end-of-string */
} gpibCmd;
```

gDset

- record type
- DSET_AI DSET_AO DSET_BI DSET_BO DSET_EV DSET_LI DSET_LO DSET_MBBI DSET_MBBO DSET_MBBID DSET_MBBOD DSET_SI DSET_SO DSET_WF

type

GPIBREAD

GPIBWRITE

GPIBCMD

GPIBREADW

GPIBEFASTO

GPIBEFASTI

などなど、devSupportGpib.hに定義されている

pri

IB_Q_LOW

IB_Q_MEDIUM

IB_Q_HIGH

cmd

- constant string to send to instrument
- デリミタがある場合、それも含めて書く

format

- string used to generate or interpret msg
- Cのprintf,scanf構文とよく似ている

rspLen、msgLen

- rspLen : room for respond2Writes response message
- msgLen : room for return data message

convert

- formatで解釈できないresponseを返す場合、ここに変換functionへのpointerを渡す
  - waveformの場合
  - 特殊なバイナリーデータを返す場合
- convertの必要がないときはNULL

P1

- For EFAST it is set internally to the number of entries in the EFAST table
- For convert it is passed to convert()
- 普通は0でOK

P2

- user defined parameter passed to convert()
- 普通は0

P3

- For EFAST it holds the address of the EFAST table
- For Convert it is passed to convert()
- EFASTテーブル(gpibコマンドを列挙したもの)のアドレス

devGpibNames

- pointer to name strings
- mbbi、mbboなどdevGpibNamesを定義したとき、それへのポインタを書く

eos

- input end-of-string
- なのだが、通常NULL

データベース

```
record(bo,"$(USER):DVM:INIT"){
    field(DTYP,"R6551")
    field(SCAN,"Passive")
    field(OUT,"#L0 A5 @0")
    }
```

@の後ろにgpibCmd構造体で定義した順の番号

```
record(mbbo,"$(USER):DVM:RANGE"){
        field(DTYP,"R6551")
        field(SCAN,"Passive")
        field(OUT,"#L0 A5 @1")
}
record(mbbo,"$(USER):DVM:SRATE"){
        field(DTYP,"R6551")
        field(SCAN,"Passive")
        field(OUT,"#L0 A5 @2")
}
```

```
record(bo,"$(USER):DVM:HOLD"){
    field(DTYP,"R6551")
    field(SCAN,"Passive")
    field(OUT,"#L0 A5 @3")
}
record(ai,"$(USER):DVM:GET"){
    field(DESC,"analog input record")
    field(SCAN,"1 second")
    field(DTYP,"R6551")
    field(FLNK,"0.0")
    field(INP,"#L0 A5 @4")
    field(PREC,"5")
}
```

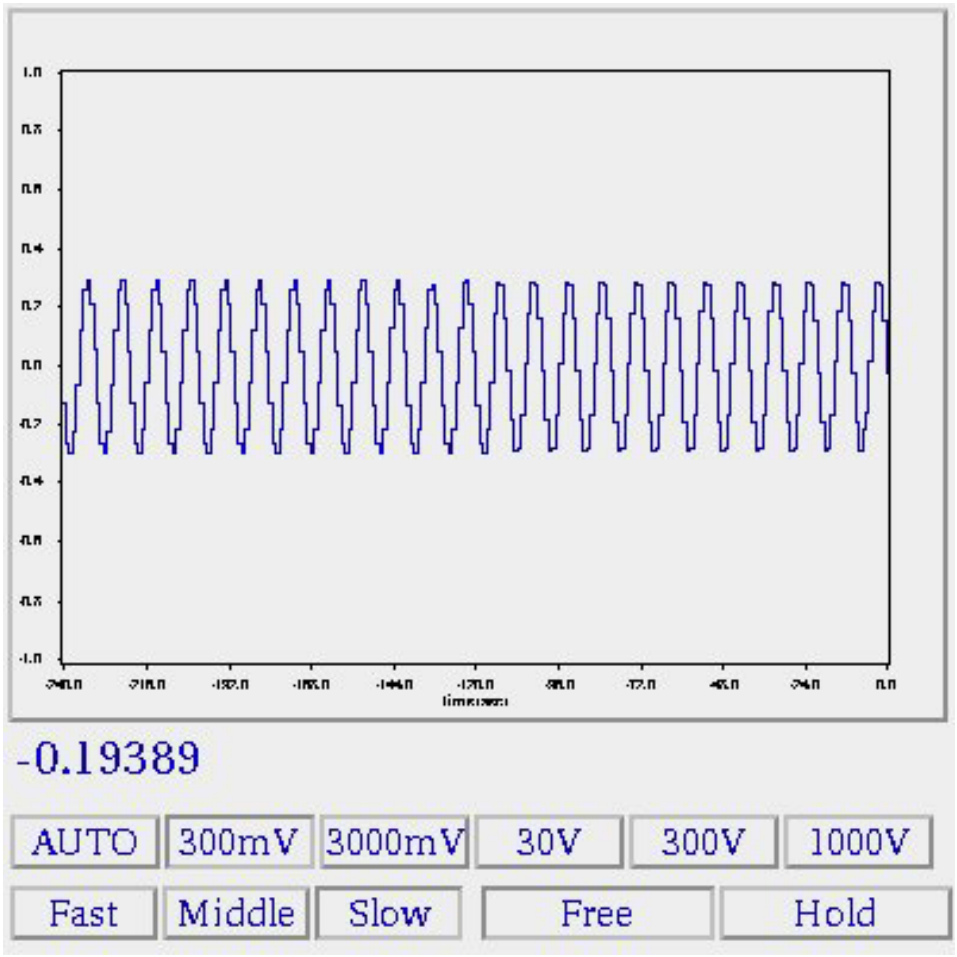
st.cmdを変更

cd ${TOP}/iocBoot/${IOC}

iocInit() の前に

＃E5810Reboot("192.168.0.2",0)

vxi11Configure("L0","192.168.0.2",0,0.0,"gpib0",0,0)


-0.19389
AUTO
300mV
3000mV
30V
300V
1000V
Fast
Middle
Slow
Free
Hold

実習

- R6551デバイスサポートを各自作って動かしてみる
- Agilent 33220A Function Generatorデバイスサポート(およびデータベース、表示ソフト)を作る(次回までの宿題)
  - GP-IBコマンドは各自webで調べること
  - Sin、Square、Rampの各functionの設定、読み返し
  - frequency、voltageを可変に(設定、読み返し)

Agilent 33220Aのコマンド

- APPLコマンドで、一気に色々な設定ができるが、EPICSデータベース構造と相性は悪いので、普通はLow-Level Commandを使う
- 例

  FUNC SIN

  FREQ 5000

  VOLT 3.0

  VOLT:OFFS -2.5

コマンド例

- ファンクション

  FUNC {SIN|SQU|RAMP|PULS|NOIS|DC|USER}

  FUNC?

- 周波数

  FREQ 周波数(Hz)

  FREQ?

- 振幅設定

  VOLT:UNIT {VPP|VRMS:DBM}

  VOLT:UNIT?

コマンド例(つづき)

- 振幅

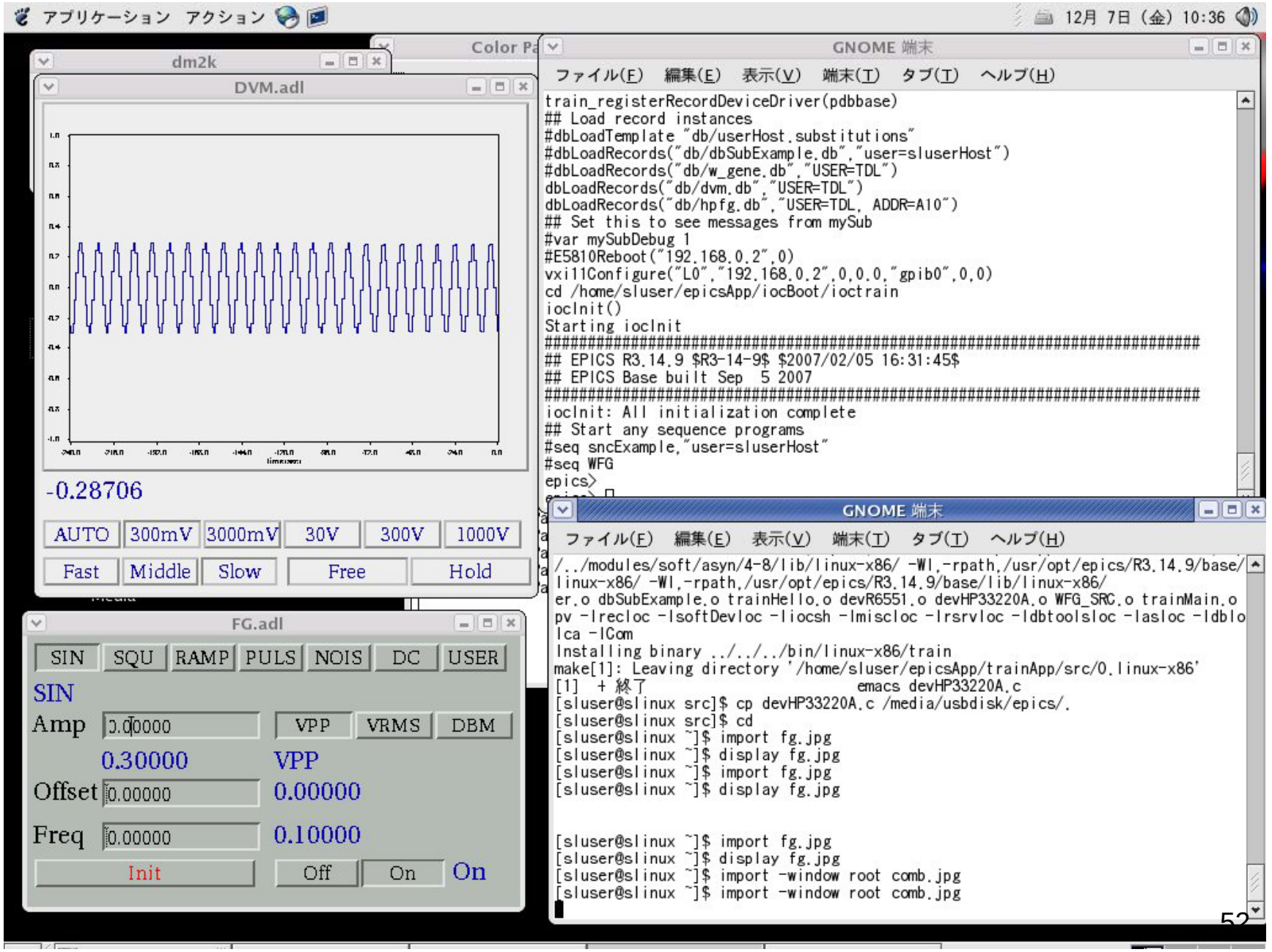
  VOLT 電圧

  VOLT?

- 出力

  OUTP {ON|OFF}

  OUTP?


アプリケーション アクション
12月 7日 (金) 10:36
dm2k
DVM.adl
-0.28706
AUTO
300mV
3000mV
30V
300V
1000V
Fast
Middle
Slow
Free
Hold
FG.adl
SIN
SQU
RAMP
PULS
NOIS
DC
USER
SIN
Amp
VPP
VRMS
DBM
0.30000
VPP
Offset
0.00000
0.00000
Freq
0.10000
0.10000
Init
Off
On
On
GNOME 端末
ファイル(F) 編集(E) 表示(V) 端末(T) タブ(T) ヘルプ(H)
train_registerRecordDeviceDriver(pdbbase)
Load record instances
#dbLoadTemplate "db/userHost.substitutions"
#dbLoadRecords("db/dbSubExample.db","user=sluserHost")
#dbLoadRecords("db/w_gene.db","USER=TDL")
dbLoadRecords("db/dvm.db","USER=TDL")
dbLoadRecords("db/hpfg.db","USER=TDL, ADDR=A10")
Set this to see messages from mySub
#var mySubDebug 1
#E5810Reboot("192.168.0.2",0)
vxi11Configure("L0","192.168.0.2",0,0.0,"gpib0",0,0)
cd /home/sluser/epicsApp/iocBoot/ioctrain
iocInit()
Starting iocInit
##
EPICS R3.14.9 $R3-14-9$ $2007/02/05 16:31:45$
EPICS Base built Sep 5 2007
##
iocInit: All initialization complete
Start any sequence programs
#seq sncExample,"user=sluserHost"
#seq WFG
epics>
GNOME 端末
ファイル(F) 編集(E) 表示(V) 端末(T) タブ(T) ヘルプ(H)
/../modules/soft/asyn/4-8/lib/linux-x86/ -Wl,-rpath,/usr/opt/epics/R3.14.9/base/lib/linux-x86/ -Wl,-rpath,/usr/opt/epics/R3.14.9/base/lib/linux-x86/
er.o dbSubExample.o trainHello.o devR6551.o devHP33220A.o WFG_SRC.o trainMain.o
pv -lrecloc -lsoftDevloc -liocsh -lmiscloc -lrsrvloc -ldbtoolsloc -lasloc -ldblo
lca -lCom
Installing binary ../../../bin/linux-x86/train
make[1]: Leaving directory '/home/sluser/epicsApp/trainApp/src/O.linux-x86'
[1] + 終了   emacs devHP33220A.c
[sluser@slinux src]$ cp devHP33220A.c /media/usbdisk/epics/.
[sluser@slinux src]$ cd
[sluser@slinux ~]$ import fg.jpg
[sluser@slinux ~]$ display fg.jpg
[sluser@slinux ~]$ import fg.jpg
[sluser@slinux ~]$ display fg.jpg
[sluser@slinux ~]$ import fg.jpg
[sluser@slinux ~]$ display fg.jpg
[sluser@slinux ~]$ import -window root comb.jpg
[sluser@slinux ~]$ import -window root comb.jpg