Stream Device のすすめ

2013/02/13 T. Obina

Outline

- Stream Device とは
- なぜ Stream Device をススメるのか？

- セットアップ方法（管理者が一度やるお仕事）
- ユーザアプリケーション作成例
  - GPIB/Ethernet機器
  - Serialで、向こうがデータを垂れ流してくるとき
  - 簡単なデバッグ方法

- Stream Device が苦手なこと
- みんなで protocol file を作って共有しよう！

- Reference

Stream Deviceとは

- スイスにある PSI (Paul Scherrer Institute) にて開発
- Author : Dirk Zimoch
- Web
  - http://epics.web.psi.ch/software/streamdevice/
  - Download もここから
  - 現時点での最新バージョンは 2.6

なぜ StreamDevice をススメるのか？

- 制御したい機器が新たに出てきたとき、以前(R3.13の頃)は「Device Support」を書くしかありませんでした。
- C言語の知識が必要です。
- Waveformを取り扱うにはポインタの知識も必要です。
- VMEには適しています。

- 制御屋さんにとってはたいした障壁ではありません。
- しかし、普通の(?)人にとって見れば、「よくわからないので、制御グループにお願いします」となりがち。
- その機器を使ってなにかを測定して、データを取り込んだ「後に」彼らのやりたいことがあるのです。もちろん、そのための道具をそろえることも仕事の一部ですので、やって当然という考え方もアリ。
- 見よう見まねでコピペするところからはじまります。教育には良いですが、時間がかかります。

- 別の機器を制御したいとき、一から書いていく必要があります。
- 簡便にするため、KEKBではGDLなども開発しました。

Device Supportとは何か？

- Record と Hardware をつなぐインターフェース
- Record Support が呼ぶ
- Recordの任意のフィールドへread/writeアクセスできる
- Sync/Asyncを決めたり、割り込みをやったり、色々。

```
record(longin, “PFROP:BKTSEL:BKTNUM_RB”) {
        field(DTYP, ”HIMV-630”)
        field(INP, ”#C0 S6 @”)
        field(SCAN, ”1 second”)
}
```

```
dbd file:

device(longin,  VME_IO, devLiHimv630,   ”HIMV-630”)
device(longout, VME_IO,  devLoHimv630,   ”HIMV-630”)
```

レコード例(VME_IO, longin)

```
devHimv630.c :

struct {
        long            number;
        DEVSUPFUN       report;
        DEVSUPFUN       init;
        DEVSUPFUN       init_record;
        DEVSUPFUN       get_ioint_info;
        DEVSUPFUN       read_longin;
        DEVSUPFUN       special_linconv;
}devLiHimv630={
        6,
        NULL,
        NULL,
        init_record,
        NULL,
        read_longin,
        NULL
};
```

実際のI/Oをするルーチン

device support の中身

```
devHimv630.c :
……
static long init_record(plongin)
struct longinRecord     *plongin;
{
 …… 色々とレコード初期化時の処理……
  plongin->udf=FALSE;
  return(0);
}
……
```

UNDEFフィールドをFALSEに

```
devHimv630.c :

……
static long read_longin(plongin)
struct longinRecord     *plongin;
{
  cardN = plongin->inp.value.vmeio.card;
  switch(plongin->inp.value.vmeio.signal){
        case 0:
                plongin->val = cards[cardN].card->chan0;
                if (debug_flag >10) logMsg("read_longin chan0 %d¥n", plongin->val);
                break;
        case 6:
                s2 = (cards[cardN].card->chan1) & 0xff00;
                plongin->val = s2;
                break;
  }
  return(CONVERT);
}
```

VALフィールドに値をセット

Device Support：GPIB機器の場合

- Asyn のdevGpibがある。
  - http://www.aps.anl.gov/epics/modules/soft/asyn/R4-20/devGpib.html

devTekTDS3000.c

```
#define DSET_AI    devAiTekTDS3000Gpib
#define DSET_AO    devAoTekTDS3000Gpib
......
gDset DSET_AI   = {6, {report, init_dev_sup, devGpibLib_initAi, NULL,
        devGpibLib_readAi, NULL, (DRVSUPFUN)&devSupParms,
        (DRVSUPFUN)devGpibLib_aiGpibWork, (DRVSUPFUN)devGpibLib_aiGpibSrq}};
gDset DSET_AO   = {6, {NULL, NULL, devGpibLib_initAo, NULL,
        devGpibLib_writeAo, NULL, (DRVSUPFUN)&devSupParms,
        (DRVSUPFUN)devGpibLib_aoGpibWork, NULL}};
......
static char *RangeList[] = { "1mV" ,"2mV" ,"5mV" ,"10mV" ,"20mV","50mV" ,"100mV" ,"200mV" ,"500mV" ,"1V" };
static unsigned long   RangeVal[] = { 0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 };
static struct devGpibNames Range = { 10, RangeList, RangeVal, 4 };
......
static struct gpibCmd gpibCmds[] = {
    /* CMMAND 0 get_ch1_volt */
  {&DSET_AI, GPIBREAD, IB_Q_LOW, "CH1:VOL?￥r", "%lf", 0, 32, NULL, 0, 0, NULL, NULL, -1 },

    /* CMMAND 1 set_ch1_volt */
  {&DSET_AO, GPIBWRITE, IB_Q_LOW, NULL, "CH1:VOL %lf￥r", 0, 32, NULL, 0, 0, NULL, NULL, -1 },

    /* CMMAND 2 get_ch2_volt */
  {&DSET_AI, GPIBREAD, IB_Q_LOW, "CH2:VOL?￥r", "%lf", 0, 32, NULL, 0, 0, NULL, NULL, -1 },
```

GPIB用DSET(Device Support Entry Table) のSkeltonが用意されている

やることは、このgpibCmdsを書くだけ…..のはずだが

```
STATIC int rd_wf(struct gpibDpvt *pdpvt,  int p1,  int p2,  char **p3)
{
        struct        waveformRecord  *
        pwf = (struct waveformRecord *) (pdpvt->precord);
        char                  *craw;
        short                 *clean;
        unsigned long         numElem;
        char  c[2], cmd[10];
        short leng, bufsize;
        short *sptr;

        clean = (short *) pwf->bptr;       /* data buffer of waveform record */
        craw=pdpvt->msg;                   /* assign pointer from GPIB device */

        craw++;                           /* skip data header "#" */
        strncpy(&c, craw, 1);             /* get data size */
        c[1] = '¥0';                       /* add null */
        leng = atoi(&c);

        craw++;
        strncpy(&cmd, craw, leng);
        cmd[leng] = '¥0';                 /* add null */
        bufsize = atoi(&cmd);

        craw += leng;
        numElem = bufsize/sizeof(short);

        if (numElem > pwf->nelm)
                numElem = pwf->nelm;
        pwf->nord = numElem;

        sptr  = (short *) craw;

        while( numElem-- ) {
            *clean++ = *sptr++;
        }
        return(OK);
}
```

waveform record に入れる部分は
自分で書く必要がある。

さすがに全部を手で書くのは無理

- KEKB ではGDLを開発。すこしマシになった。
- .gtファイルをもとにdevice support を生成 .listができる

```
devHP81130AGpib.gt :
Device HP81130A

EfastTable ArmMode    = ":ARM:MODE GAT",
                        ":ARM:MODE STAR";

EfastTable Senceset   = ":ARM:SENC POS",
                        ":ARM:SENC NEG";

ParamTable
{
   "CLS"    {
      rec=ao,
      command="*CLS￥r￥n",
   }
.....
   "IDN?"    {
      rec=wf,
      command="*IDN?￥r￥n",
      conv=rd_wf,
      leng=100,
   }
 "SOUR:PULS:DEL1"    {
      rec=ao,
      command=":PULS:DEL1 %lf NS￥r￥n",
   }
```

```
devHP81130AGpib.list :
/*** COMMAND LIST ***/

No.  0   AO      CLS
No.  1   AO      RESET
No.  2   AO      WAI
No.  3   WF      IDN?
No.  4   AO      ARM:LEV
No.  5   AO      ARM:LEV:TERM
No.  6   BO      ARM:MODE
No.  7   BO      ARM:SENS
No.  8   MBBO    ARM:SOUR
No.  9   MBBO    DISPLAY
```

```
record(ao,"BMAPRFY:PG:RST") {
  field(DESC, "analog output record")
  field(SCAN, "Passive")
  field(DTYP, "HP81130A")
  field(OUT, "#L0 A14 @1")
...
}
```

途中にコマンドを挿入すると、番号がずれて破綻

……これを解決するのが Stream Device なの DEATH か？

- 基本方針:DeviceSupportを書くのではなく、あくまでも「Stream Device」として扱う…..つまりデバイスサポートはすでに書いてある、ということ。
- そのデバイスに対して、ある「protocol」で入出力する
- ユーザが書くのはprotocol file と database file のみ
- protocol file はASCIIで、ioc起動時に解釈
  - ソースの再コンパイルは不要
- コマンド名は自分で定義。device support のように、順番を気にしたり、間を飛ばすなどの小細工は不要

- 管理者はStreamDeviceをインストールする作業が必要

セットアップ（管理者がやっておくこと）

- 前述のサイトからダウンロードして、インストール
  - Asyn必須。対応バージョンに注意。
  - 標準的には $(EPICS_BASE)/../modules/soft/stream/2-6 など
  - Template 登録（必須ではないが、やっておくと楽）

```
$ makeBaseApp.pl -l
Valid application types are:
        support
        ioc
        example
        caClient
        caServer
        cerlModules
Valid iocBoot types are:
        example
        ioc
        cerlModules
```

Configure/RELEASEファイルの中：

```
#SNCSEQ=$(EPICS_BASE)/../modules/soft/seq/2.1.7
#ASYN=$(EPICS_BASE)/../modules/soft/asyn/4-19
#STREAM=$(EPICS_BASE)/../modules/soft/stream/2-6
#NETDEV=$(EPICS_BASE)/../modules/soft/netDev/1.0.2
```

としておいて、コメントを外せばすぐに使えるように

アプリケーション作成例

- Ethernet Socket I/O : Keithley 2701
- 通常の epics application 作成手順 (makeBaseApp.pl)で 一連のディレクトリ構造を作る。
  - この例では application name として keith2701 とする

```
Keith2701/
    |-- configure
    |       |-- RELEASE
    |
    |-- keith2701App
    |       |
    |       |-- Db
    |       |    |-- Makefile
    |       |    |-- test.db
    |       |    |-- keith2701.proto
    |       |
    |       |-- src
    |            |-- Makefile
    |            |-- K2701.dbd
    |
    |-- iocBoot
           |-- iockeith2701
                |-- st.cmd
```

内容を変更するファイルは左の通り

アプリケーション作成例：続き

- configure/RELEASE

```
#SNCSEQ=$(EPICS_BASE)/../modules/soft/seq/2.1.7
ASYN=$(EPICS_BASE)/../modules/soft/asyn/4-19
STREAM=$(EPICS_BASE)/../modules/soft/stream/2-6
#NETDEV=$(EPICS_BASE)/../modules/soft/netDev/1.0.2
```

- src ディレクトリ

```
% Makefile

keith2701_DBD += base.dbd
keith2701_DBD += asyn.dbd
keith2701_DBD += stream.dbd
keith2701_DBD += K2701.dbd

keith2701_LIBS += asyn
keith2701_LIBS += stream
```

```
% K2701.dbd  新規作成

registrar(drvAsynIPPortRegisterCommands)
```

Socket I/O をするための、ポートドライバ

Db ディレクトリ

```
% Makefile

DB += test.db
```

```
% test.db

record(stringin, "obina:idn")
{
    field(DESC, "get IDN")
    field(DTYP, "stream")
    field(INP,  "@keith2701.proto getidn PS1")
}
```

```
% keith2701.proto

Terminator = CR LF;
#
getIDN {
    out "*IDN?";
    in "%39c";
}
```

startup script

```
% st.cmd

#!../../bin/linux-x86_64/keith2701

## Load record instances
dbLoadRecords("db/test.db")
epicsEnvSet("STREAM_PROTOCOL_PATH", ".:../../keith2701App/Db")
drvAsynIPPortConfigure ("PS1", "172.28.8.244:1394")
```

IP, port指定

```
% epics> dbpf obina:idn.PROC 1
DBR_UCHAR:          1         0x1
PS1 obina:idn: 18 bytes surplus input "10912,A09  /A02  <0a>"
PS1 obina:idn: after 39 bytes "...S INC.,MODEL 2701,11 "

epics> dbpr obina:idn
ASG:                DESC: get IDN       DISA: 0             DISP: 0
DISV: 1             NAME: obina:idn     SEVR: INVALID       STAT: READ
SVAL:               TPRO: 0
VAL: KEITHLEY INSTRUMENTS INC.,MODEL 2701,11
```

エラーは出るが、とりあえず値は取れている

先ほどのエラーへの対処

- EPICSのstringinは39byteまで。

対処法1:protocolファイルで39文字以上を無視する方法

```
% keith2701.proto

getIDN {
    out "*IDN?";
    in "%39c";
    ExtraInput = Ignore;
}
```

エラーハンドラ記述

対処法2:char の waveform でデータを受け取る方法

```
% test.db

record(waveform, "obina:idn_wf")
{
    field(DESC, "get IDN with waveform")
    field(DTYP, "stream")
    field(INP,  "@keith2701.proto getIDN PS1")
    field(FTVL, "CHAR")
    field(NELM, "100")
}
```

waveformで取った場合

```
$ caget obina:idn_wf
obina:idn_wf 100 75 'K' 69 'E' 73 'I' 84 'T' 72 'H' 76 'L' 69 'E' 89 'Y'
32 ' ' 73 'I' 78 'N' 83 'S' 84 'T' 82 'R' 85 'U' 77 'M' 69 'E' 78 'N' 84 'T'
83 'S' 32 ' ' 73 'I' 78 'N' 67 'C' 46 '.' 44 ',' 77 'M' 79 'O' 68 'D' 69 'E'
76 'L' 32 ' ' 50 '2' 55 '7' 48 '0' 49 '1' 44 ',' 49 '1' 49 '1' 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$ caget -d0 obina:idn_wf

[pfoper@pfrproc1 ~]$ python
Python 2.5.2 (r252:60911, Sep  7 2008, 16:48:40)
[GCC 3.4.6 20060404 (Red Hat 3.4.6-10)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import ca
>>> ca.Get('obina:idn_wf')
[75, 69, 73, 84, 72, 76, 69, 89, 32, 73, 78, 83, 84, 82, 85, 77, 69, 78, 84,
83, 32, 73, 78, 67, 46, 44, 77, 79, 68, 69, 76, 32, 50, 55, 48, 49, 44, 49, 49,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0]
>>>
```

その他のコマンド・ハンドラー

- 本家のDocument参照

```
out string;
in string;
wait milliseconds;
```

```
WriteTimeout
ReplyTimeout
Terminator
InTerminator / OutTerminator
Separator
ExtraInput
```

```
@mismatch
@writetimeout
@replytimeout
@readtimeout
@init
```

例（本家のDocumentより）

ai/ao float conversion

```
getFrequency {
    out "FREQ?";
    in "%f";
}

setFrequency {
    out "FREQ %f";
    @init { getFrequency; }
}
```

init使用例

```
setPosition {
    out "POS %f";
    @init { out "POS?"; in "POS %f"; }
}
```

例２：bi/bo

```
getSwitch {
    out ”SW?”;
    in ”SW %{OFF|ON}”;
}

setSwitch {
    out ”SW %{OFF|ON}”;
    @init { getSwitch; }
}
```

対応するDBはbi/boにして
ZNAM, ONAMを記述する

```
record(bi, ”$(user):getswitch”)
{
    field(DTYP, ”stream”)
    field(INP,  ”@xxxx.proto getSwitch PS1” )
    field(ZNAM, ”OFF”)
    field(ONAM, ”ON”)
}
```

protocol file 内での変数定義

```
f = ”FREQ”;     # sets f to ”FREQ” (including the quotes)
f1 = $f ” %f”;  # sets f1 to ”FREQ %f”

getFrequency {
    out $f ”?”; # same as: out ”FREQ?”;
    in $f1;     # same as: in ”FREQ %f”;
}

setFrequency {
    out $f1;    # same as: out ”FREQ %f”;
}
```

例 3 : mbbi/mbbo

```
getGAMO {
  out ":FREQ:ARM:STOP:SOUR?";
  in "%{IMM|EXT|TIM|DIG}";
}
setGAMO {
  out ":FREQ:ARM:STOP:SOUR %{IMM|EXT|TIM|DIG}";
  @init { getGAMO; }
}
```

```
record(mbbi, "$(user):getgamo")
{
    field(DESC, "mbbi record")
    field(SCAN, "Passive")
    field(DTYP, "stream")
    field(INP,  "@AG53181.proto getGAMO PS1 7")
    field(ZRST, "IMM")
    field(ONST, "EXT")
    field(TWST, "TIM")
    field(THST, "DIG")
}
```

例４：TDS3000オシロスコープ

- KEKB 吉井さん作成(中)

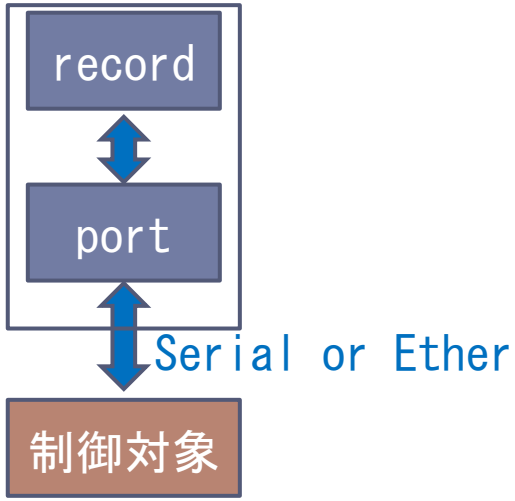
```
set_ENC {
    out "DAT:ENC %{ASCI|RIB|RPB|SRI|SRP}";
}
set_SOU {
    out "DAT:SOU %{CH1|CH2|CH3|CH4|MATH}";
}
get_DATA {
    Separator = ",";
    out "CURV?";
    in  "%d￥n";
}
```

```
record(waveform, "$(user):BPT:DATA")
{
    field(DTYP, "stream")
    field(SCAN, "1 second")
    field(INP,  "@$(proto) get_DATA $(Line) $(Addr)")
    field(NELM, "101")
    field(FTVL, "LONG")
}
```

Serial Portをつかって制御するとき

```
% st.cmd

drvAsynSerialPortConfigure ("PS1","/dev/ttyS1")
asynSetOption ("PS1", 0, "baud", "9600")
asynSetOption ("PS1", 0, "bits", "8")
asynSetOption ("PS1", 0, "parity", "none")
asynSetOption ("PS1", 0, "stop", "1")
asynSetOption ("PS1", 0, "clocal", "Y")
asynSetOption ("PS1", 0, "crtscts", "N)
```



dbdの中で
registrar(drvAsynSerialPortRegisterCommands)
を追加しておくこと

Db中ではここで定義した PS1 を使うのみ:
　レコードからみるとPS1ポートを読み書きをしているだけで、その先の物理層が
　シリアルだろうがEtherだろうが、関知しない

st.cmd内で、 drvAsynSerialPortConfigure or drvAsynIPPortConfigure 選択

Serial で相手がデータを垂れ流して来る時

- 例:放射線モニタ
  - http://pfconrg07.kek.jp:8082/trac/pfcont/wiki/Epics/stream_radmon

- db記述するときに SCANフィールドを I/O Intrに
  - データを受け取るたびにSCANされる
  - Terminator記述
  - データが来るまで無限に待つ

```
% radmon.db

record(ai, "$(user):getRadVal") {
  field(DESC, "test")
  field(DTYP, "stream")
  field(INP,  "@medic.proto getRadVal PS1")
  field(SCAN, "I/O Intr")
}
```

```
% medic.proto

InTerminator = " ";
ReadTimeout = 100;

setTestMode {
  out "T";
}
getRadVal {
  in "%f";
  @mismatch {;}
}
```

簡単なデバッグ方法

- AsynにあるTraceMaskをセット

```
epics> asynSetTraceMask(”PS1”, 0, 0x8)
epics> asynSetTraceIOMask(”PS1”, 0, 0x4)

/*asynTrace is implemented by asynManager*/
/*All asynTrace methods can be called from any thread*/
/* traceMask definitions*/
#define ASYN_TRACE_ERROR     0x0001
#define ASYN_TRACEIO_DEVICE  0x0002
#define ASYN_TRACEIO_FILTER  0x0004
#define ASYN_TRACEIO_DRIVER  0x0008
#define ASYN_TRACE_FLOW      0x0010

/* traceIO mask definitions*/
#define ASYN_TRACEIO_NODATA 0x0000
#define ASYN_TRACEIO_ASCII  0x0001
#define ASYN_TRACEIO_ESCAPE 0x0002
#define ASYN_TRACEIO_HEX    0x0004
```

GPIB / VXI-11 機器の場合

- dbdファイルに　registrar(vxi11RegisterCommands)
- 機器のアドレス指定はdbファイル中で

```
record(stringin, "obina:idn")
{
    field(DESC, "get IDN")
    field(DTYP, "stream")
    field(INP,  "@AG53181.proto getIDN PS1 22")   <---- GPIB機器のアドレスが22の場合
}
```

- LAN/GPIBならば、st.cmd内で

```
vxi11Configure ("PS1","172.28.xxx.yyy",1,1000,"gpib0")
```

Stream Device が苦手なこと

- binary ベースの入出力は苦手です
  - バイナリでも、単純な入出力はOK

- オシロのwaveformをバイナリ転送するとき
- Modbus/TCPなど、バイナリが前提のプロトコル
  - それ用のドライバを書けば良いはずだが…人柱募集
  - チェックサムが必要だったりすると、かなり面倒
- PLCなどで、バイナリ入出力をするとき
  - これはNetDevにおまかせ、ということで。

みんなで protocol file を共有しよう！

- 自分で使うコマンドだけ書く場合が多い
  - それで十分ですので。
  - 全ての機能を書くのはたいへん。
  - でも誰かが書いていると嬉しい
- 人が書いたものをコピーして、それをもとに追加・変更
  - だいたいこのパターンがほとんど
  - 「サンプルをくれ！」と言われる

- みんなで共有しましょう
- 所外の人にむけても
  - epicsUsersJP のサイト

Reference

- 本家: http://epics.web.psi.ch/software/streamdevice/
- Asyn: http://www.aps.anl.gov/epics/modules/soft/asyn/
- How To Do Serial _ Stream Device 版
  - http://www.aps.anl.gov/epics/modules/soft/asyn/HowToDoSerial_StreamDevice.html

- PFcont http://pfconrg07.kek.jp:8082/trac/pfcont/wiki/Epics
- EpicsUsersJP : http://cerldev.kek.jp/trac/EpicsUsersJP/
- Arduino (山本さん):
  - http://www-acc.kek.jp/EPICS_Gr/products/Arduino/devArduinoDebug.db

- Mailing List : epics-users@ml.post.kek.jp