実用EPICS入門-8
技術部専門講習

飛山真理
帯名崇

VMEバス

産業用の標準バスの一種で、モトローラ社のマイクロプロセッサ68000シリーズ用のバスVERSAbusを基本として、VERSAmodule Eurocard(VMEbus)として広く使われるようになった。CPU、メモリー、入出力装置、アナログインターフェースなどをモジュール化し、これらをVMEbusのバックプレーン、電源を有するサブラックに収納して、1つのコンピュータシステムを構築する。

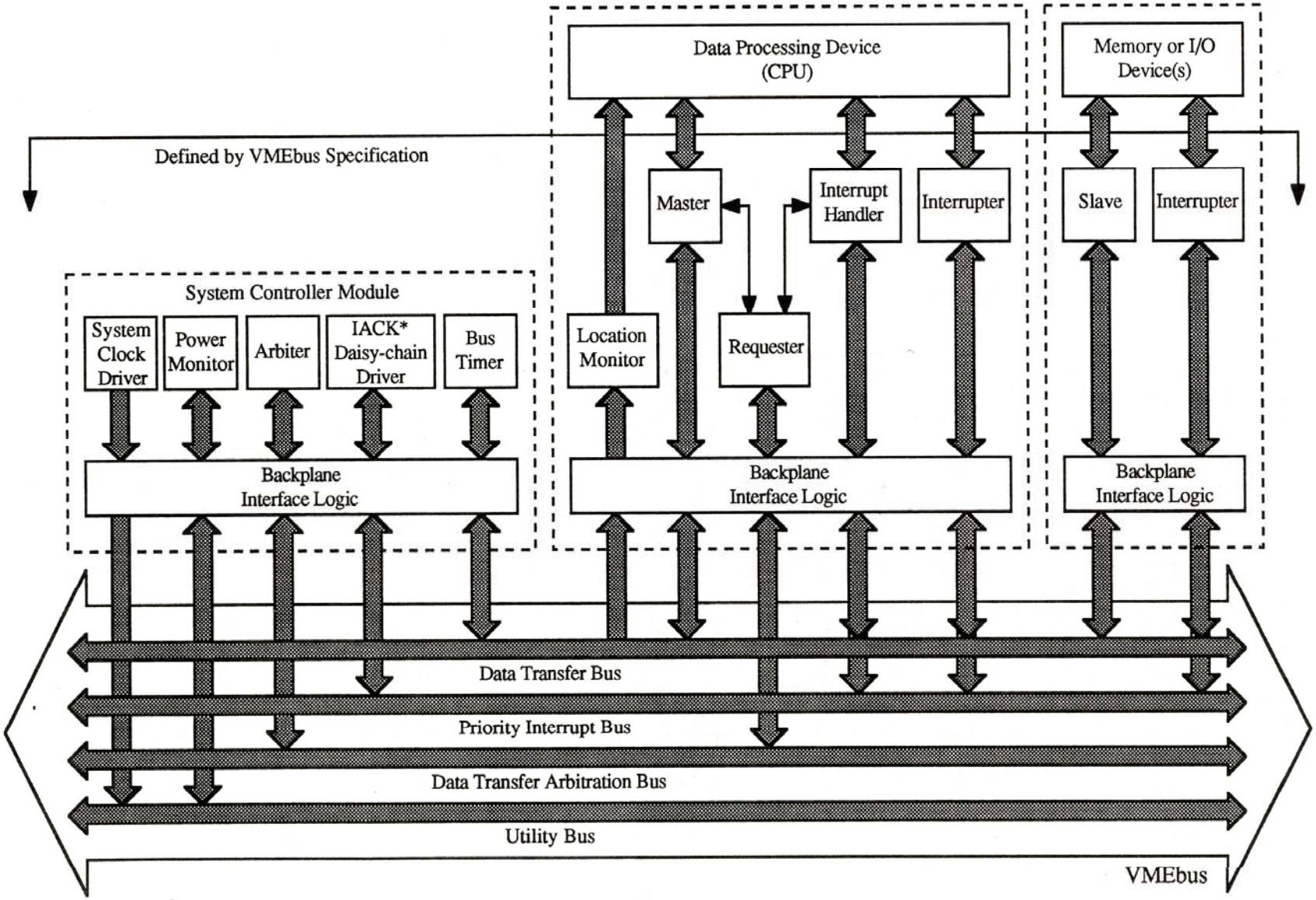
IEC821及びIEEE1014-1987で正式に規格が定まり、多くの種類のボードが開発され、産業用制御、軍事、航空機制御、鉄道道路関係、通信制御、シミュレーション、医学解析、高エネルギー・原子核物理学分野など広い分野で使われている。現在の規格はVITA (the VMEbus International Trade Association)で決められ、最新の標準規格はANSI/VITA 1-1994(VME64)である。現在でも広く使われているIEEE-1014-1987(original VMEbus)では、(1)マスター／スレーブ構造、(2)非同期バス、(3)バスに対する応答が早いデバイスにも遅いデバイスにも対応するハンドシェイクプロトコル、(4)マルチプレクスなしのバス、(5)アドレスサイズは16ビットから32ビット、(6)データバスサイズは8ビットから32ビット、(7)バスの転送速度は最高毎秒40Mバイト、(8)マルチプロセッサー可能、(9)7レベルの割り込み、(10)単一バックプレーンで21カードスロットまで可、となっていた。VME64ではOriginal VMEbusを上位互換性を持ちながら大幅に拡張し、(1)6Uサイズカード(233.35mm×160mm、P1及びP2コネクタ使用)使用時にバスのマルチプレクスにより最大64ビットのデータ幅、アドレスサイズ、(2)3Uサイズカード(100mm×160mm、P1コネクタのみ使用)に対して最大32ビットデータ幅、40ビットアドレスサイズ、(3)2倍の転送速度（最大毎秒80Mバイト)、(4)低ノイズコネクター、(5)VMEサイクルのretryに関する規定、(6)バスロックサイクル規定、(7)ボードの自動検出によるプラグ・アンドプレイ機能、(8)SERCLK及びSERDATピンの再定義、などを行っている。このほか、さらに高速化、利便化を目指して拡張を行ったVME64xやVME320などもあり、一部分野で使用されている。

アドレス、データ

- アドレス空間：A16(Short)、A24(Standard)、A32(Extended)、(A40、A64 Long)
- データサイズ：D8、D16、D32、(D64)
- AM(Address Modifier)コード
  - A16特権(0x2D)　A16非特権(0x29)
  - A24特権データ(0x3D) A24非特権データ(0x39)
  - A32特権データ(0x0D) A32非特権データ(0x09)

VMEバス参考書

- The VMEbus Handbook (W.D.Peterson)
  ISBN 1-885731-08-6
- インターフェース 1993年4月号
- インターフェース 1987年2月号
- DIGITAL BUS HANDBOOK (J. D. Giacomo)
  ISBN 0-07-016923-3
- The System Engineer's Handbook
  - A guide to building VMEbus and VXIbus systems (J.Black) ISBN 0-12-102820-8


Data Processing Device
(CPU)
Memory or I/O
Device(s)
Defined by VMEbus Specification
Master
Interrupt
Handler
Interrupter
Slave
Interrupter
System Controller Module
System
Clock
Driver
Power
Monitor
Arbiter
IACK*
Daisy-chain
Driver
Bus
Timer
Location
Monitor
Requester
Backplane
Interface Logic
Backplane
Interface Logic
Backplane
Interface Logic
Data Transfer Bus
Priority Interrupt Bus
Data Transfer Arbitration Bus
Utility Bus
VMEbus

〔図２〕ユーロカードの寸法マトリックス

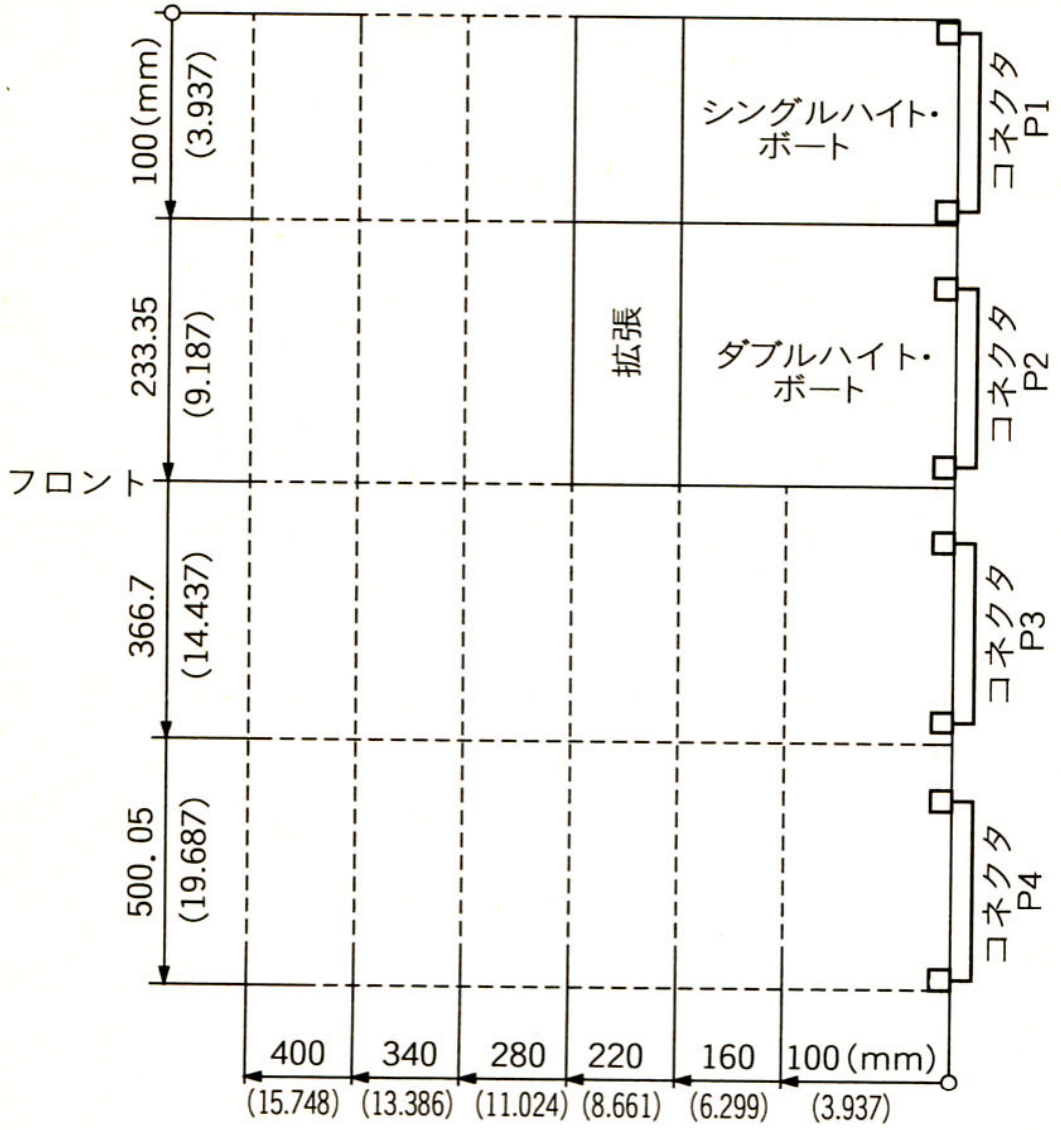


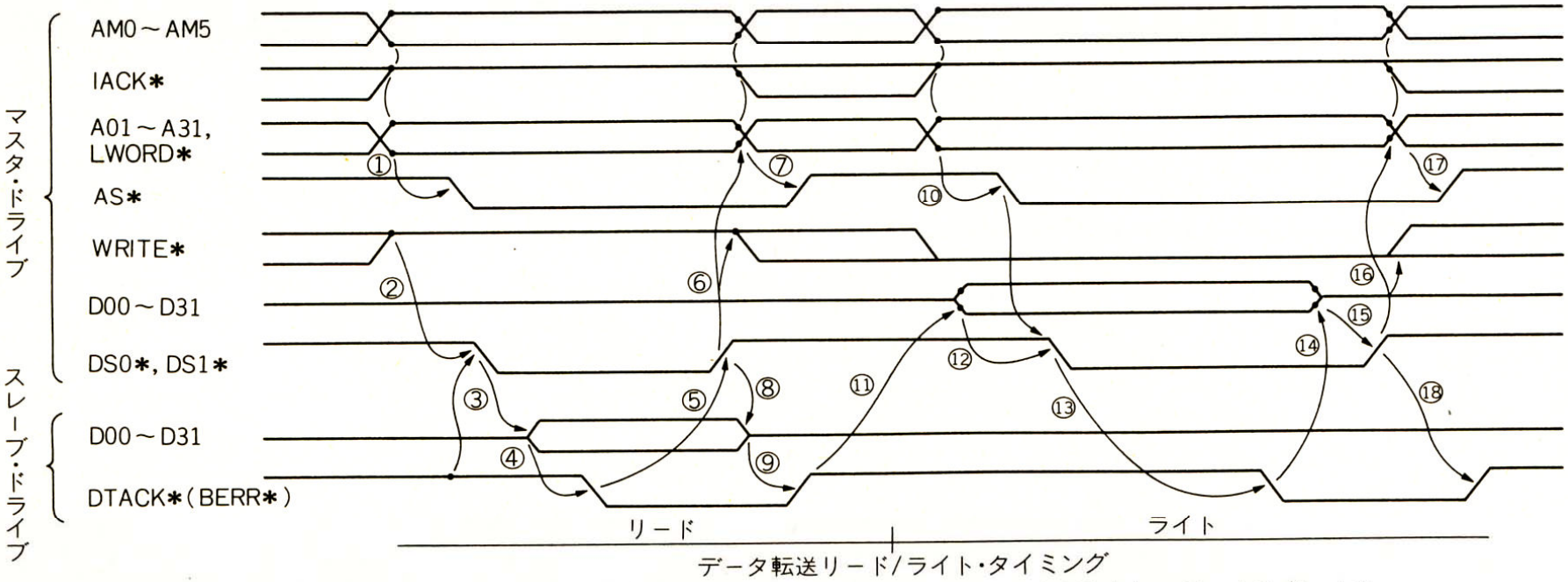
Table 1-3. VMEbus P1/J1 and P2/J2 pin assignments.

P1 / J1 Pin Assignments					
Pin	Row z (††)	Row a	Row b	Row c	Row d (††)
1	MPR	D00	BBSY*	D08	VPC (§)
2	GND	D01	BCLR*	D09	GND (§)
3	MCLK	D02	ACFAIL*	D10	+V1
4	GND	D03	BG0IN*	D11	+V2
5	MSD	D04	BG0OUT*	D12	RsvU
6	GND	D05	BG1IN*	D13	-V1
7	MMD	D06	BG1OUT*	D14	-V2
8	GND	D07	BG2IN*	D15	RsvU
9	MCTL	GND	BG2OUT*	GND	GAP*
10	GND	SYSCLK	BG3IN*	SYSFAIL*	GA0*
11	RESP*	GND	BG3OUT*	BERR*	GA1*
12	GND	DS1*	BR0*	SYSRESET*	+3.3V
13	RsvBus	DS0*	BR1*	LWORD*	GA2*
14	GND	WRITE*	BR2*	AM5	+3.3V
15	RsvBus	GND	BR3*	A23	GA3*
16	GND	DTACK*	AM0	A22	+3.3V
17	RsvBus	GND	AM1	A21	GA4*
18	GND	AS*	AM2	A20	+3.3V
19	RsvBus	GND	AM3	A19	RsvBus
20	GND	IACK*	GND	A18	+3.3V
21	RsvBus	IACKIN*	SERA (†)	A17	RsvBus
22	GND	IACKOUT*	SERB (†)	A16	+3.3V
23	RsvBus	AM4	GND	A15	RsvBus
24	GND	A07	IRQ7*	A14	+3.3V
25	RsvBus	A06	IRQ6*	A13	RsvBus
26	GND	A05	IRQ5*	A12	+3.3V
27	RsvBus	A04	IRQ4*	A11	LI/I*
28	GND	A03	IRQ3*	A10	+3.3V
29	RsvBus	A02	IRQ2*	A09	LI/O*
30	GND	A01	IRQ1*	A08	+3.3V
31	RsvBus	-12 VDC	+5VSTDBY	+12 VDC	GND (§)
32	GND	+5 VDC	+ 5VDC	+5 VDC	VPC (§)

P2 / J2 Pin Assignments					
Pin	Row z (††)	Row a	Row b	Row c	Row d (††)
1	UsrDef	UsrDef	+5 VDC	UsrDef	UsrDef (§)
2	GND	UsrDef	GND	UsrDef	UsrDef (§)
3	UsrDef	UsrDef	RETRY* (†)	UsrDef	UsrDef
4	GND	UsrDef	A24	UsrDef	UsrDef
5	UsrDef	UsrDef	A25	UsrDef	UsrDef
6	GND	UsrDef	A26	UsrDef	UsrDef
7	UsrDef	UsrDef	A27	UsrDef	UsrDef
8	GND	UsrDef	A28	UsrDef	UsrDef
9	UsrDef	UsrDef	A29	UsrDef	UsrDef
10	GND	UsrDef	A30	UsrDef	UsrDef
11	UsrDef	UsrDef	A31	UsrDef	UsrDef
12	GND	UsrDef	GND	UsrDef	UsrDef
13	UsrDef	UsrDef	+5 VDC	UsrDef	UsrDef
14	GND	UsrDef	D16	UsrDef	UsrDef
15	UsrDef	UsrDef	D17	UsrDef	UsrDef
16	GND	UsrDef	D18	UsrDef	UsrDef
17	UsrDef	UsrDef	D19	UsrDef	UsrDef
18	GND	UsrDef	D20	UsrDef	UsrDef
19	UsrDef	UsrDef	D21	UsrDef	UsrDef
20	GND	UsrDef	D22	UsrDef	UsrDef
21	UsrDef	UsrDef	D23	UsrDef	UsrDef
22	GND	UsrDef	GND	UsrDef	UsrDef
23	UsrDef	UsrDef	D24	UsrDef	UsrDef
24	GND	UsrDef	D25	UsrDef	UsrDef
25	UsrDef	UsrDef	D26	UsrDef	UsrDef
26	GND	UsrDef	D27	UsrDef	UsrDef
27	UsrDef	UsrDef	D28	UsrDef	UsrDef
28	GND	UsrDef	D29	UsrDef	UsrDef
29	UsrDef	UsrDef	D30	UsrDef	UsrDef
30	GND	UsrDef	D31	UsrDef	UsrDef
31	UsrDef	UsrDef	GND	UsrDef	GND (§)
32	GND	UsrDef	+5 VDC	UsrDef	VPC (§)

Notes: (†) Pin(s) redefined under the VME64 specification.
(††) Pin(s) redefined under the VME64x specification.

〔図７〕基本的なデータ転送のタイミング



データ転送リード/ライト・タイミング

①AS*の“H”を確認し，AS*でストローブされる信号群を有効にする，その後セットアップ時間をとりAS*を“L”にする
②WRITE*のセットアップ時間をとり，AS*“L”およびDTACK*“H”を確認しDSn*を“L”にする
③DSn*“L”により，有効なアドレス情報を受けたスレーブは，データをバス上へ置く
④データをバス上へ置いたスレーブは，DTACK*をドライブする
⑤DTACK*“L”を受けたマスタはバス上のデータを受け取り，DSn*を“H”にする
⑥DSn*を“H”にしたマスタはWRITE*とAS*でストローブされる信号群を解放する
⑦WRITE*とAS*でストローブされる信号群を解放したマスタはAS*を“H”にする
⑧DSn*“H”を認識したスレーブは，データ・バスを解放する
⑨データ・バスを解放したスレーブはDTACK*を解放する
⑩AS*の“H”を確認し，AS*でストローブされる信号群を有効にする．その後，セットアップ時間をとりAS*を“L”にする
⑪DTACK*“H”を確認したマスタがデータをバス上へ置く
⑫データのセットアップ時間をとり，AS*“L”を確認しDSn*を“L”にする
⑬DSn*“L”により，有効なアドレス情報を受けたスレーブはデータを受け取り，DTACK*をドライブする
⑭DTACK*“L”を認識したマスタが，データ・バスを解放する
⑮データ・バスを解放したマスタがDSn*を“H”にする
⑯DSn*を“H”にしたマスタがWRITE*およびAS*でストローブされる信号群を解放する
⑰WRITE*およびAS*でストローブされる信号群を解放したマスタがAS*を“H”にする
⑱DSn*“H”を認識したスレーブはDTACK*を解放する

※ここでのDSn*は，転送サイズおよびロケーションにより，DS0*とDS1*のいずれか，または両方を示す

VMEバス機器用device support

- 例:TD4V(500MHz digital timing delay)

環境

- EPICS R314.9
  - R313とは色々違う。特にdevice support関係は大幅に変更になっている
- IOC:モトローラMVME5500
  - PPC750とは全然別物
- 開発環境
  - vxWorks上では開発できない。UNIXマシン上で開発し、cross compileする

VME機器

- VMEのアドレス空間に、レジスタをマップする
  - TD4Vでは

アドレス	R/W	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0	W	Delay Preset(16bit)															
0	R	Preset Readback															
4	W	N/A															Inhibit
4	R	N/A											OUTPUT	START	RF	Ext Inhibit	Enable

Output、Start、RF、Enable、Ext Inhibitは全て0で正常、Inhibitは1でInhibit

VMEアクセス例

- delayの値をセットする
  - 0x*******0アドレスにPreset値を書き込む
- delayの値を読む
  - 0x*******0アドレスの値を読む

デバイスサポート(devTd4v.c)

```
#include <vxWorks.h>
#include <types.h>
#include <stdioLib.h>
#include <string.h>
#include <vme.h>
#include <dbDefs.h>
#include <dbAccess.h>
#include <recSup.h>
#include <devSup.h>
#include <link.h>
#include <module_types.h>
#include <longinRecord.h>
#include <longoutRecord.h>
#include <boRecord.h>
#include <biRecord.h>
#include <mbbiDirectRecord.h>
#include <epicsMutex.h>
#include (epicsExport.h>
```

```
struct DvmeTd4v {
    unsigned short PreSet;
    unsigned short Dummy0;
    unsigned short sio;
    unsigned short Dummy1;
    unsigned short Last_PreSet;
    unsigned short Dummy2;
    unsigned short Dummy3;
    unsigned short Dummy4;
};
```

registerマップに対応した構造体。1ボードで8ワード占有するので、この構造体が全部で8ワード分の大きさになるように設定する。

```
static long init();
static long init_record();
static long read_longin();
static long write_longout();
static long init_bi_record();
static long init_bo_record();
static long read_bi();
static long write_bo();
static long init_mi_record();
static long read_mbbiDirect();
static int  checkLink();
```

これから使う関数をforward宣言しておく

```
struct {
        long            number;
        DEVSUPFUN       report;
        DEVSUPFUN       init;
        DEVSUPFUN       init_record;
        DEVSUPFUN       get_ioint_info;
        DEVSUPFUN       read_longin;
        DEVSUPFUN       special_linconv;
}devLiTd4v={
        6,
        NULL,
        init,
        init_record,
        NULL,
        read_longin,
        NULL};
epicsExportAddress(dset,devLiTd4v);
```

longinレコードに対する
actionを定義する

おまじない

```
struct {
        long            number;
        DEVSUPFUN       report;
        DEVSUPFUN       init;
        DEVSUPFUN       init_record;
        DEVSUPFUN       get_ioint_info;
        DEVSUPFUN       write_longout;
        DEVSUPFUN       special_linconv;
}devLoTd4v={
        6,
        NULL,
        NULL,
        NULL,
        NULL,
        write_longout,
        NULL};
epicsExportAddress(dset,devLoTd4v);
```

```
struct {
        long            number;
        DEVSUPFUN       report;         /* used by dbior */
        DEVSUPFUN       init;           /* called 1 time before & after all records */
        DEVSUPFUN       init_record;    /* called 1 time for each record */
        DEVSUPFUN       get_ioint_info; /* used for COS processing (not used)*/
        DEVSUPFUN       write_bo;       /* output command goes here */
}devBoTd4v={
        5,
        NULL,
        NULL,
        init_bo_record,
        NULL,
        write_bo
};
epicsExportAddress(dset,devBoTd4v);
```

```
struct {
        long            number;
        DEVSUPFUN       report;         /* used by dbior */
        DEVSUPFUN       init;           /* called 1 time before & after all records */
        DEVSUPFUN       init_record;    /* called 1 time for each record */
        DEVSUPFUN       get_ioint_info; /* used for COS processing */
        DEVSUPFUN       read_bi;        /* input command goes here */
}devBiTd4v={
        5,
        NULL,
        NULL,
        init_bi_record,
        NULL,
        read_bi
};
epicsExportAddress(dset,devBiTd4v);
```

```
struct{
        long            number;
        DEVSUPFUN       report;
        DEVSUPFUN       init;
        DEVSUPFUN       init_mi_record;
        DEVSUPFUN       get_ioint_info;
        DEVSUPFUN       read_mbbiDirect;
}devMbbiTd4v={
        5,
        NULL,
        NULL,
        init_mi_record,
        NULL,
        read_mbbiDirect};
epicsExportAddress(dset,devMbbiTd4v);
```

```
struct ioCard {
        volatile struct DvmeTd4v     *card; /* address of this card */
        epicsMutexId                 lock; /* semaphore */
      };
```

このカードの構造体の定義

```
#define CONST_NUM_LINKS 20

static int          debug_flag = 1;
static unsigned long Base_IO = 0x21000000;
int                 td4v_num_links = 0;

static struct ioCard cards[CONST_NUM_LINKS];
static int          init_flag = 0;
```

```
int devTd4vConfig(ncards,a32base)
int ncards;
long a32base;
{
  td4v_num_links = ncards;
  Base_IO = a32base;
  logMsg("VmeTd4 NumLink= %d BaseIO=%x¥n",  td4v_num_links,  Base_IO,
     0,0,0);
  return init(0);
}
```

使うTd4vの数と、ベースアドレスをセットする関数。外から使うため、static宣言して
はいけない。

```
static long init(after)
int after;
{
  int                    cardNum, chanNum;
  unsigned char          probeVal[2];
  volatile  struct DvmeTd4v    *p;

  if (init_flag != 0 )  return(OK);
  init_flag = 1;
  if (td4v_num_links == 0) return (OK);          /* Do nothing if no modules configured */
  if (sysBusToLocalAdrs(VME_AM_EXT_SUP_DATA,(char *)Base_IO,(char **)&p) == ERROR)
       { logMsg("VmeTd4v: cannot find extended address space¥n",0,0,0,0,0,0);
              return(ERROR);}
for (cardNum=0; cardNum< td4v_num_links; cardNum++)
   {
     if (vxMemProbe((char*) &(p->PreSet), READ, 2, &probeVal[0])< OK)
       {
         if (debug_flag >0 )
         logMsg("No TD4V with cardNum= %d¥n probe= %x¥n",cardNum,p,0,0,0,0);
         cards[cardNum].card = NULL;
       }
```

```
 else
      {
        if (debug_flag >0)
        logMsg("Found TD4V with cardNum= %d¥n address= %x¥n",cardNum,p,0,0,0,0);
        cards[cardNum].card = p;  /* Remember address of the board */
        cards[cardNum].lock = epicsMutexMustCreate();
      epicsMutexUnlock((cards[cardNum].lock)); /* Init the board lock  &&*/
      }
    p++;
  }
 return(OK);
}
```

Td4Vが実際に存在するかvxMemProbeで調査、あればそのアドレス(ベースアドレス)をcards[cardNum].cardにセットする。

セマフォの初期設定を行う

```
static long init_record(plongin)
struct longinRecord     *plongin;
{
  return(0);
}
```

longinレコードの初期化。何もしていない。

```
static long read_longin(plongin)
struct longinRecord     *plongin;
{
  short cardN;
  if (debug_flag >10)  logMsg("read_longin called...¥n",0,0,0,0,0,0);
  cardN = plongin->inp.value.vmeio.card;
  if (checkLink(cardN) == ERROR)  return(ERROR);
              switch(plongin->inp.value.vmeio.signal){
                    case 0:
                          plongin->val = cards[cardN].card->PreSet;
                          break;
                    case 1:
                          plongin->val = cards[cardN].card->PreSet;
                          break;
                    default:
                          return(-1);
              }
        return(0);}
```

```
static long write_longout(plongout)
struct longoutRecord    *plongout;
{
  short cardN;
  long  setpreset;

  cardN = plongout->out.value.vmeio.card;
  if (checkLink(cardN) == ERROR) return(ERROR);
  epicsMutexMustLock((cards[cardN].lock));
  setpreset = plongout->val;
  if (setpreset<=0)
    cards[cardN].card->PreSet = 1;
  else if (setpreset>65535)
    cards[cardN].card->PreSet = 65535;
  else
    cards[cardN].card->PreSet = (unsigned short)(plongout->val);
epicsMutexUnlock((cards[cardN].lock));

        return(0);
}
```

```
static long init_bo_record(pbo)
struct boRecord *pbo;
{       pbo->mask = 1;
            return(0);}

static long write_bo(pbo)
struct boRecord *pbo;
{
  short cardN;
  cardN = pbo->out.value.vmeio.card;
  if (checkLink(cardN) == ERROR)
    {      logMsg("Error--- No TD4V for card %d¥n",cardN,0,0,0,0,0);
     return(ERROR);
    }
  epicsMutexMustLock((cards[cardN].lock));
  cards[cardN].card->sio = pbo->rval & pbo->mask;
  epicsMutexUnlock((cards[cardN].lock));
       return(0);
}
```

```
static long init_bi_record(pbi)
struct biRecord *pbi;
{       pbi->mask = 1;
                return(0);
}
```

```
static long read_bi(pbi)
struct biRecord *pbi;
{
  short cardN;
  cardN = pbi->inp.value.vmeio.card;
  if (checkLink(cardN) == ERROR) return(ERROR);

  switch(pbi->inp.value.vmeio.signal){
        case 0:
              pbi->rval = cards[cardN].card->sio & pbi->mask;
               break;
        case 1:
              pbi->rval = (cards[cardN].card->sio)>>2 & pbi->mask;
               break;
        case 2:
              pbi->rval = (cards[cardN].card->sio)>>3 & pbi->mask;
               break;
```

```
        case 3:
                pbi->rval = (cards[cardN].card->sio)>>1 & pbi->mask;
                 break;
          case 4:
                pbi->rval = (cards[cardN].card->sio)>>4 & pbi->mask;
                  break;
          default:
                 return(-1);
    }
    return(0);
}
```

過去のdatabaseとの互換性のためにある。各ステータスをそれぞれbiで読むと、VMEのアクセスが無駄に増える。mbbiDirectで一気に全部読んで、あとでデータベース上でbiレコードに分配する方が賢い

```
static long init_mi_record(pmbbi)
struct MbbiDirectRecord *pmbbi;
{
     return(0);
}

static long read_mbbiDirect(pmbbi)
struct mbbiDirectRecord *pmbbi;
{
  short cardN;

  cardN = pmbbi->inp.value.vmeio.card;
  if (checkLink(cardN) == ERROR)  return(ERROR);

  pmbbi->rval = cards[cardN].card->sio;
        return(0);
}
```

```
static int checkLink(cardN)
short   cardN;
{
  if (cardN >= td4v_num_links)
    return(ERROR);
  if (cards[cardN].card == NULL)
    {
      logMsg("No TD4V with this number = %d¥n",cardN,0,0,0,0,0);
      return(ERROR);
    }

  if (debug_flag >10)
    logMsg("Yes you have TD4V with card No= %d¥n",cardN,0,0,0,0,0);
  return(OK);
}
```

devTd4V.dbd

device(longin,VME_IO,devLiTd4v,"TD-4V")
device(longout,VME_IO,devLoTd4v,"TD-4V")
device(bo,VME_IO,devBoTd4v,"TD-4V")
device(bi,VME_IO,devBiTd4v,"TD-4V")
device(mbbiDirect,VME_IO,devMbbiTd4v,"TD-4V")

自分で作る

Makefile

```
PROD_IOC = fbppc
DBD += fbppc.dbd

fbppc_DBD += devTd4v.dbd
fbppc_SRCS += devTd4v.c
```

データベース

Record	DTYP	Signal	Name	機能
longout	TD-4V	0	$(U):TD4:PRESET	プリセット値
longin	TD-4V	0	$(U):TD4:PRESET_R	リードバック
bo	TD-4V	0	$(U):TD4:ENABLE	0でenable
mbbiDirect	TD-4V	0	$(U):TD4:STAT_R	ステータス
bi	Soft Channel		$(U):TD4:START_R	スタート入力
bi	Soft Channel		$(U):TD4:RF_R	クロック信号
bi	Soft Channel		$(U):TD4:ENABLE_R	enable
bi	Soft Channel		$(U):TD4:INH_R	外部inhibit
bi	Soft Channel		$(U):TD4:OUTPUT_R	出力

データベース例

```
record(longout,"$(USER):TD4:PRESET") {
        field(DESC,"long output record")
        field(SCAN,"Passive")
        field(DTYP,"TD-4V")
        field(FLNK,"$(USER):TD4:PRESET_R.VAL")
        field(OUT,"#$(CHAN) S0 @")
}
```

CHANにはC0、C1と言った形の値を入れる

```
 record(longin,"$(USER):TD4:PRESET_R") {
        field(DESC,"long input record")
        field(SCAN,"5 second")
        field(PHAS,"0")
        field(DTYP,"TD-4V")
        field(INP,"#$(CHAN) S0 @")
}
 record(bo,"$(USER):TD4:ENABLE") {
        field(DESC,"binary output record")
        field(SCAN,"Passive")
        field(DTYP,"TD-4V")
        field(OUT,"#$(CHAN) S0 @")
}
```

```
 record(mbbiDirect,"$(USER):TD4:STAT_R") {
                field(DESC,"mbbiDirect record")
                field(SCAN,"5 second")
                field(PHAS,"0")
                field(DTYP,"TD-4V")
                field(FLNK,"$(USER):TD4:FAN1.VAL")
                field(INP,"#$(CHAN) S0 @")
}
 record(fanout,"$(USER):TD4:FAN1") {
                 field(DESC,"fanout record")
                 field(LNK1,"$(USER):TD4:START_R")
                 field(LNK2,"$(USER):TD4:RF_R")
                 field(LNK3,"$(USER):TD4:ENABLE_R")
                 field(LNK4,"$(USER):TD4:INH_R")
                 field(LNK5,"$(USER):TD4:OUTPUT_R")
                 field(LNK6,"0.00000000000000e+00")
        }
```

```
 record(bi,"$(USER):TD4:START_R") {
        field(DESC,"binary input record")
        field(DTYP,"Soft Channel")
        field(INP,"$(USER):TD4:STAT_R.B4")
}
 record(bi,"$(USER):TD4:RF_R") {
        field(INP,"$(USER):TD4:STAT_R.B2")
}
 record(bi,"$(USER):TD4:ENABLE_R") {
        field(INP,"$(USER):TD4:STAT_R.B0")
}
 record(bi,"$(USER):TD4:INH_R") {
        field(INP,"$(USER):TD4:STAT_R.B1")
}
 record(bi,"$(USER):TD4:OUTPUT_R") {
        field(INP,"$(USER):TD4:STAT_R.B4")
}
```

スタートアップファイル

dbLoadRecords("db/FB_TD4X.db","USER=BM_GBPM2:CLK, CHAN=C0")
dbLoadRecords("db/FB_TD4X.db","USER=BM_GBPM2:GAT, CHAN=C1")

複数枚のTD4Vを使うときは、CHAN=C0、CHAN=C1となっていく

devTd4vConfig(2,0x21000000)

参考リンク

- EPICS R313 vxWorksに関しては

  http://ahfb1.kek.jp/~tobiyama/epics/default.html
- VME Linux IOCに関しては

  http://pfrdata.kek.jp/PF/Control/epics/magApp/index_html

VMEカードを使うときの注意

- 有名どころの製品でも、最新のVME規則に従っていないものもある(address pipeliningなど)
  - 最後はlogic analyzerでタイミング確認が必要なこと有り
- 最近のIOCはCPUとVMEバスとの間にPCIバスを使うのが普通なので、たとえvxWorksを使っていても割り込みに対する早いレスポンスは期待できない
  - 一定時間内のレスポンスは保障されている
- OSにLinuxを使うと、リアルタイム性は全く保障できない
  - リアルタイム性は本当に必要なのか？

VME

- 相手のアドレスが直接自由にさわれるので、理屈さえ分かってしまえば(たとえかなり複雑な機能のものでも)デバイスサポート開発は簡単
- VMEプロトコルは(最近の複雑なもの—PCIなどと比べて)極めて単純明快なので、デバッグも容易。カスタムボード製作も(理屈が分かっていれば)簡単
- まだそうすぐには絶滅しないが、洋々たる将来があるわけでもない
  - VXIはすでに絶滅危惧種(red)

ASYNドライバー

- Ethernet直接につながるデバイスで、メッセージベース通信しかしないなら、ASYNドライバーを使ったデバイスサポートを比較的簡単に書くことができます。
  - 例：Agilent 33220AのGP-IB用デバイスサポートをそのままASYNで使う
  - 例：RS-232CデバイスをXPORTを使ってEthernetから直接コントロール

例:ビーム位相検波器

- RFに対するビームの位相を測定する機械
- メッセージベースコマンドをEthernetから送る
  - 例:CH1アッテネータ設定

    SAT1:-17<cr><lf>
  - 例:CH1アッテネータ読み込み

    RAT1<cr><lf>[応答]AT1:-18<cr><lf>

GP-IBと同じようにsourceを作る

- 設定コマンドに対して答えを返すので
  - init_ai関数のrespond2Writeの値を-1から適当な値(秒単位)に変更する
  - DSETの該当レコードで、readback bufferの値を入れておく(コマンドの次の項目、値は適当)
- 測定値にheaderのみならず、訳の分からないでバック値も付いてくるので変換関数が必要
- その他にも色々変換関数が必要

変換関数例

```
static int convAi(gpibDpvt *pdpvt, int P1, int P2, char **P3)
{
  struct aiRecord *pai = ((struct aiRecord *)(pdpvt->precord));

  char *craw;
  char wa1[3000];

  craw = pdpvt->msg;
  craw = strtok(craw,":");
  strcpy(wa1,strtok(NULL,","));

   pai->val = atof(wa1);
  pai->udf = FALSE;
  return(asynSuccess);
}
```

データベース・スタートアップ

- GP-IBと同じようにデータベースを作る
- makeする

- スタートアップファイル

  drvAsynIPPortConfigure("L0","172.19.XXX:30704",0,0,0)

使った印象

- なんか微妙に変
  - データベースにすぐに反応しないことがある
  - 使い方が悪いのかもしれない
  - 何かの弾みに、コマンドと返事がずれることがある
  - ということで、信頼性が必要なアプリケーションにはあまりオススメできません。

netDev

- Ethenetで直接通信するデバイスをepicsから使うため、KEKの小田切淳一さんが開発したツール
- BPMC(三菱電機)、DARWIN(横川)、FA-M3 PLC(横川)、三菱PLC(三菱電機)、EMB-LAN100(KEK)、KV-LE20A(キーエンス)、CS1-PLC(オムロン)、KE3000(チノー)用のデバイスサポートがある

KE3000ネットロガー

KEシリーズは、高速：多点のデータ収録に適したネットワーク対応のロガーです。各機能はユニット化されており、必要なユニットを組み合わせることで用途に応じたシステムを構築出来ます。
イーサネットモデルは、インターネットやイントラネット環境を利用し、遠隔データ収録、データの一元管理などに対応出来ます。

インプリメントする機能

- 全チャンネルの値を一気に読み込む
  - 1スロット12チャンネル、最大60チャンネル。このデータを(とりあえず)waveformレコードに入れる
  - 何スロット読むかは外から与えられるようにする必要がある
- その他の設定などのコマンドは扱わない
  - webから簡単に設定できる
  - そうちょくちょく変えるものではない

KE3000に送るコマンド

- 例：ch1から12個分のデータを読むRTUモードのコマンド

意味	binary	備考
スレーブアドレス	0x01	常に0x01
ファンクションコード	0x04	0x04はアナログデータ要求
開始リファレンス番号(H)	0x00	ch1の開始リファレンス番号は30101だが、コマンドではこれから30001を引くので100になる
開始リファレンス番号(L)	0x64	
データ要求個数(H)	0x00	12個=0x0c
データ要求個数(L)	0x0c	
CRC(H)	0xb1	データチェックコード(CRCコード)、上記コマンドによって変わるので常に計算する必要有り
CRC(L)	0xd0	

KE3000から返してくるデータ

意味	binary	備考
スレーブアドレス	0x01	固定
ファンクションコード	0x04	アナログデータ要求
データ数	0x18	12×2(データ＋データ情報があるので)
Ch1データ(H)	0x01	0x011d=285
Ch1データ(L)	0x1d	
Ch1データ情報(H)	0x00	小数点、ステータス。この場合0x0001なので1/10で値が28.5と分かる
Ch1データ情報(L)	0x01	
ch2以降のデータ		
CRC(H)	0xb3	データチェックコード
CRC(L)	0xdf	

データ情報のフォーマット

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
A/D	I/O	U/F	N/A	EV4	EV3	EV2	EV1	ERR	Burn	OF	UF	DP3	DP2	DP1	DP0

- A/D:アナログなら0、デジタルなら1
- I/O:出力なら0、入力なら1
- U/F:バイポーラなら0、ユニポーラなら1
- EV1～EV4:各event発生で1
- ERR:入出力エラー状態
- BURN:測温抵抗体測定時断線で1
- OF:オーバーフローエラーで1
- UF:アンダーフローエラーで1
- DP3～DP0:小数点位置、0でx1、1で1/10、2で1/100、3で1/1000、4で1/10000

sourceファイル

- netDevディレクトリにsub directoryを作る
  - 例 :
    /local/R314.8.2/extensions/src/netDev/current

Makefile O.Common bpmc darwin jk_yew mel mw100 yew
O.linux-x86 common emb key misc omron

mkdir chinoL

chinoLディレクトリ

- ChinoL.dbd
  - waveformレコードしか使わない

  device(waveform,INST_IO,devWfChinoLogL,"ChinoLogL")
- EPICSデータの読み書き加工をする本体

  devChinoLogL.c
- waveform関連コード

  devWaveformChinoLogL.c

devChinoLogL.c

```
#include	<dbFldTypes.h>
#include	"drvNetMpf.h"
#include	"devNetDev.h"
#ifndef EPICS_REVISION
#include <epicsVersion.h>
#endif
#if EPICS_REVISION == 14 && EPICS_MODIFICATION >= 2
#include <epicsExport.h>
#endif

#define CHINOL_CMND_LENGTH 8
#define CHINOL_UDP_PORT    11111
#define CHINOL_GET_PROTO chino_get_protocol()
#define CHINOL_MAX_NDATA 5120
#define CHINOL_DATA_OFFSET 3
```

```
LOCAL int finsUseSamePortNumber = MPF_SAMEPORT;
LOCAL unsigned short calCRC(unsigned char cd[], int nmax);
LOCAL int TwoRawToVal(unsigned char bu[],int offset, float *res );
LOCAL long chino_parse_link(
                    struct link *,
                    struct sockaddr_in *,
                    int *,
                    void *
                    );
LOCAL int chino_get_protocol(void);
LOCAL void *chino_calloc(int, int, int, int, int);
```

データ構造体

```
typedef struct
{
  int      unit;
  int      type;
  int      chan;
  int      bit;
  int      width;
  int      slvA;
  int      cmd;
  int      start;
  int      ndata;
  int      data_trans;
  char     *lopt;
} CHINOL_LOG;
```

データベースのINPフィールドから拾う情報(アドレス、コマンド、データ数など)を格納する場所

通信プロトコル

- TCPかUDPを決める。TCPに決め打ち

```
int chinoLogLUseTcp = 1;

LOCAL int chino_get_protocol(void)
{
  if (chinoLogLUseTcp)
    {
      return MPF_TCP;
    }

  return MPF_UDP;
}
```

```
LOCAL void *chino_calloc(
                    int unit,
                    int type,
                    int chan,
                    int bit,
                    int width
                    )
{
  CHINOL_LOG *d;
  d = (CHINOL_LOG *) calloc(1, sizeof(CHINOL_LOG));
  if (!d)
    { errlogPrintf("devChinoLogL: calloc failed¥n");
      return NULL;
    }
  d->unit   = unit;
  d->type   = type;
  d->chan   = chan;
  d->bit    = bit;
  d->width  = width;
  return d;
}
#include       "devWaveformChinoLogL.c"
```

INPフィールドから情報を拾うfunction

- init_waveform_record(devWaveformChinoLogL.cにある)がfunction chino_parse_linkを使って、データベースのINPフィールドに書いてある情報を読み出しセットする。
- INPフィールドは

  field(INP,"@$(ip)(11111):0x01#0x04:100&60")

  のような形になっていてる。

- @IPアドレス(ポート番号):スレーブアドレス#ファンクションコード:開始リファレンス番号&データの個数
- 区切りは、parseLinkPlcCommonで決められている
  - hostname[{routing_path}][:unit]#[type:]addr[&option]
- これ以上多種のデータを渡したいときは、自分でparseLinkPlcCommonを拡張する必要がある

chino_parse_link

- chino_parse_linkはparseLinkPlcCommonをつかってINPフィールドをpeer_address、unit、address、optionに分ける
  - 全てcharなので、sscanf変換で数字に変える
  - 例:

```
if (sscanf(unit,"%x",&d->slvA) !=1) {
  errlogPrintf("devChinoLog :cannot get slave
  address¥n"); }
```

```
LOCAL long chino_parse_link(struct link *plink,struct sockaddr_in *peer_addr,int *option,
                            void *device)
{
  char *protocol = NULL;
  char *unit  = NULL;
  char *type  = NULL;
  char *addr  = NULL;
  char *route = NULL;
  char *lopt  = NULL;
  CHINOL_LOG *d = (CHINOL_LOG *) device;

  if (parseLinkPlcCommon(plink,peer_addr,&protocol,&route,&unit,&type,&addr,&lopt))
    {
      errlogPrintf("devChinoLogL: illeagal input specification¥n");
      return ERROR;
    }
```

```
if (sscanf(unit,"%x",&d->slvA) !=1)
    {errlogPrintf("devChinoLogL :cannot get slave address¥n");}
  if (sscanf(type,"%x",&d->cmd) !=1)
    {errlogPrintf("devChinoLogL :cannot get command¥n");}
   if (sscanf(addr,"%d",&d->start) != 1)
    {errlogPrintf("devChinoLogL : cannot get start address¥n");}
   if (sscanf(lopt,"%d",&d->data_trans) !=1)
    {
      errlogPrintf("devChinoLogL : cannot get transfer length¥n");
    }
return OK;
}
```

送信データを作るfunction

```
LOCAL long chino_config_command(
                        uint8_t  *buf,       /* driver buf addr     */
                        int      *len,       /* driver buf size     */
                        void     *bptr,      /* record buf addr     */
                        int      ftvl,       /* record field type   */
                        int      ndata,      /* n to be transferred */
                        int      *option,    /* direction etc.      */
                        CHINOL_LOG *d,
                        int      sid
                        )
{
  int nwrite;
  int n;
  int resCRC;
```

```
n = ndata;
  nwrite = isWrite(*option) ? (d->width)*n : 0;
  if (*len < CHINOL_CMND_LENGTH + nwrite)
    {
      errlogPrintf("devChinoLogL: buffer is running short¥n");
      return ERROR;
    }
  buf[ 0] = d->slvA;                    /* slave address */
  buf[ 1] = d->cmd;                     /* command     */
  buf[ 2] = d->start>>8;                /* start address (H)*/
  buf[ 3] = d->start;                   /* start address (L)*/
  buf[ 4] = 0x00;                       /* number (H) */
  buf[ 5] = (d->data_trans)*2;          /* number (L) 0x0c */
  resCRC = calCRC(buf,6);
  buf[ 6] = resCRC>>8;                  /* crc (H) */
  buf[ 7] = resCRC     ;                /* crc (L) */
  *len = CHINOL_CMND_LENGTH;
return 0;
}
```

動作

- bufにコマンドを並べるだけ
- 実際の動作は、read_waveform内のnetDevReadWriteXxが行う

CRC

- バッファーからCRCコードを生成する

```
LOCAL unsigned short calCRC(unsigned char cd[], int nmax)
{
  int i,j;
  unsigned short iCRC;
  unsigned short iCY,iP;
  unsigned char iC1,iC2;

  iCRC = 0xffff;

  for (i=0;i<nmax;i++)
    {
      iCRC = iCRC ^ cd[i];
      for (j=1; j<=8; j++)
        {
          iCY = iCRC & 0x1;
          if ((iCRC & 0x8000) == 0x8000)
            {
              iP = 0x4000;
              iCRC = iCRC & 0x7fff;
            }
          else
            {
              iP = 0x0;
            }
          iCRC = iCRC>>1;
          iCRC = iCRC | iP;
          if (iCY == 1){
            iCRC = iCRC ^ 0xa001;
          }
        }
    }
  if ((iCRC & 0x8000) == 0x8000)
    {
      iP = 0x80;
      iCRC = iCRC & 0x7fff;
    }
  else
    {
      iP = 0;
    }
  iC1 = iCRC & 0xff;
  iC2 = ((iCRC & 0x7f00) >>8) | iP;

  return iC1*256 + iC2;

}
```

KE3000が返すデータ

意味	binary	備考
スレーブアドレス	0x01	固定
ファンクションコード	0x04	アナログデータ要求
データ数	0x18	12×2(データ＋データ情報があるので)
Ch1データ(H)	0x01	0x011d=285
Ch1データ(L)	0x1d	
Ch1データ情報(H)	0x00	小数点、ステータス。この場合0x0001なので1/10で値が28.5と分かる
Ch1データ情報(L)	0x01	
ch2以降のデータ		
CRC(H)	0xb3	データチェックコード
CRC(L)	0xdf	

2byte×2個のでーたから1個のデータを作るfunction

- ユニポーラ、バイポーラの区別
- アナログ、デジタルの区別
- 小数点の処理
- エラー処理
  - bufferのoffset+2にA/D,I/O,U/F
  - bufferのofffset+3に小数点、burnout情報
  - データはoffset(上位),offset+1(下位)

```
LOCAL int TwoRawToVal(unsigned char bu[],int offset, float
*res )
{

  float f1;

  if ((bu[offset+2] & 0x20) == 0x20){
    f1 = bu[offset]*256 + bu[offset+1];
      }
  else if ((bu[offset] & 0x80) == 0x80){
    f1 = (bu[offset]-255)*256 +(bu[offset+1]-255);
  }
  else{
    f1 = bu[offset]*256+bu[offset+1];
  }

  switch (bu[offset+3] & 0x0f)
    {
    case 0:
      *res = f1;
      break;
    case 1:
      *res = f1*0.1;
      break;
```

```
case 2:
      *res = f1*0.01;
      break;
    case 3:
      *res = f1*0.001;
      break;
    case 4:
      *res = f1*0.0001;
      break;
    default:
      return -1;
    }

  return (bu[offset+3] & 0xf0);
}

    default:
      return -1;
    }

  return (bu[offset+3] & 0xf0);
}
```

読み取ったデータを処理する function

```
LOCAL long chino_parse_response(
                        uint8_t *buf,    /* driver buf addr     */
                        int     *len,    /* driver buf size     */
                        void    *bptr,   /* record buf addr     */
                        int      ftvl,   /* record field type   */
                        int      ndata,  /* n to be transferred */
                        int     *option, /* direction etc.      */
                        CHINOL_LOG *d,
                        int      sid
                        )
{
  int i;
  int ret;
  float temp[1000],*rawVal,*ptemp;
  rawVal = bptr;
```

```
if (isRead(*option))
   {
     for (i=0;i<(d->data_trans);i++)
       {
         ret = TwoRawToVal(buf,CHINOL_DATA_OFFSET+i*4,&temp[i]);
         if (ret != 0){
           switch (ret & 0xf0){
           case 0x10:
             temp[i]=-10.0;
             break;
           case 0x20:
             temp[i]=10.0;
             break;
           case 0x30:
             temp[i]= -1000.0;
             break;
           default:
             temp[i] = -1000.0;}
         }
       }
   }
```

エラー処理(overflow, underflow, burnout)

```
ndata = d->data_trans;
  ptemp = temp;
  i= ndata;
  while (i--)
    {
      *rawVal++ = (float) *ptemp ++;
    }
  ret = 0;

  return ret;
}
```

devWaveformChinoLogL.c

```
#include        <waveformRecord.h>

/*****************************************************************
 * Waveform (command/response IO)
 *****************************************************************/
LOCAL long init_waveform_record(struct waveformRecord
        *);
LOCAL long read_waveform(struct waveformRecord *);
LOCAL long config_waveform_command(struct dbCommon
        *, int *, uint8_t *, int *, void *, int
);
LOCAL long parse_waveform_response(struct dbCommon *,
        int *, uint8_t *, int *, void *, int
);

INTEGERDSET devWfChinoLogL = {
  5,
  NULL,
  netDevInit,
  init_waveform_record,
  NULL,
  read_waveform
};
```

```
#if EPICS_REVISION == 14 && EPICS_MODIFICATION >= 2
epicsExportAddress(dset, devWfChinoLogL);
#endif

LOCAL long init_waveform_record(struct waveformRecord *pwf)
{
  CHINOL_LOG *d;
  d = chino_calloc(0, 0, 0, 0, 2);
  return netDevInitXxRecord(
                        (struct dbCommon *) pwf,
                        &pwf->inp,
                        MPF_READ | CHINOL_GET_PROTO |
DEFAULT_TIMEOUT,
                        d,
                        chino_parse_link,
                        config_waveform_command,
                        parse_waveform_response
                        );
}

LOCAL long read_waveform(struct waveformRecord *pwf)
{
  TRANSACTION *t = (TRANSACTION *) pwf->dpvt;
  CHINOL_LOG *d = (CHINOL_LOG *) t->device;


  return netDevReadWriteXx((struct dbCommon *) pwf);
}
```

```
LOCAL long config_waveform_command(
                        struct dbCommon *pxx,
                        int *option,
                        uint8_t *buf,
                        int *len,
                        void *device,
                        int transaction_id
                        )
{
  struct waveformRecord *pwaveform = (struct waveformRecord *)pxx;
  CHINOL_LOG *d = (CHINOL_LOG *) device;
  return chino_config_command(
                        buf,
                        len,
                        pwaveform->bptr,
                        pwaveform->ftvl,
                        pwaveform->nelm,
                        option,
                        d,
                        transaction_id
                        );
}
LOCAL long parse_waveform_response(
                        struct dbCommon *pxx,
                        int *option,
                        uint8_t *buf,
                        int *len,
                        void *device,
                        int transaction_id
                        )
{
```

```
struct waveformRecord *pwaveform = (struct waveformRecord *)pxx;
  CHINOL_LOG *d = (CHINOL_LOG *) device;
  long ret;

  ret = chino_parse_response(
                        buf,
                        len,
                        pwaveform->bptr,
                        pwaveform->ftvl,
                        pwaveform->nelm,
                        option,
                        d,
                        transaction_id
                        );

  switch (ret)
    {
    case 0:
      pwaveform->nord = pwaveform->nelm;
    default:
      ;
    }

  return ret;
}
```

基本的には、他の例にある同種ファイルの名前を変えるだけ。

Makefile

- /local/R314.8.2/extensions/src/netDev/currentにあるMakefileに追加

```
SRC_DIRS += $(NETDEV)/chinoL
DBD += chinoL.dbd
netDev_SRCS += devChinoLogL.c
```

データベース

```
record(waveform,"$(USER):$(PLACE):SLOTA") {
      field(DESC,"waveform record")
      field(SCAN,"5 second")
      field(DTYP,"ChinoLogL")
      field(PRIO,"LOW")
      field(FLNK,"$(USER):$(PLACE):SLOTA:FAN0.VAL")
      field(PREC,"10")
      field(INP,"@$(ip)(11111):0x01#0x04:100&60")
      field(EGU,"V")
      field(HOPR,"10.0")
      field(LOPR,"-10.0")
      field(NELM,"60")
      field(FTVL,"FLOAT")
}
```

スタートアップ

```
dbLoadRecords("db/FB_CHINOLC1.db","USER=FB_MOVE, PLACE=LC1, ip=172.19.46.36")
dbLoadRecords("db/FB_CHINOLC2.db","USER=FB_MOVE, PLACE=LC2, ip=172.19.46.42")
dbLoadRecords("db/FB_CHINOLC3.db","USER=FB_MOVE, PLACE=LC3, ip=172.19.46.43")
dbLoadRecords("db/FB_CHINOLC4.db","USER=FB_MOVE, PLACE=LC4, ip=172.19.46.44")
dbLoadRecords("db/FB_CHINOLC5.db","USER=FB_MOVE, PLACE=LC5, ip=172.19.46.45")
dbLoadRecords("db/FB_CHINOLC6.db","USER=FB_MOVE, PLACE=LC6, ip=172.19.46.46")
dbLoadRecords("db/FB_CHINOLC7.db","USER=FB_MOVE, PLACE=LC7, ip=172.19.46.47")
dbLoadRecords("db/FB_CHINOLC8.db","USER=FB_MOVE, PLACE=LC8, ip=172.19.46.48")
```

netDevで開発したいときは

- かなりのCに関する知識が必要です。
- ドライバー群の中に一部、メーカーで作った部分があり、ライセンス関係が不明確な部分があります。
- ASYNドライバーより経験上遙かに信頼性があると思われますので、実用的にはこちらの方がオススメです(が、結構大変)。
- インストール等に関しては、コントロール専門家にお聞きください。

次回は

- 美しい表示をめざして
  - Python/Tkをつかって美しいコントロールパネルを作る