DASBOX Model-5x0

16bit版

基本サブルーチン仕様書

for

LINUX
UNIX
Windows

第１版　平成１４年２月
第２版　平成１５年２月
第３版　平成１５年８月
第４版　平成１５年10月

システムデザインサービス株式会社

目　　次

改版履歴

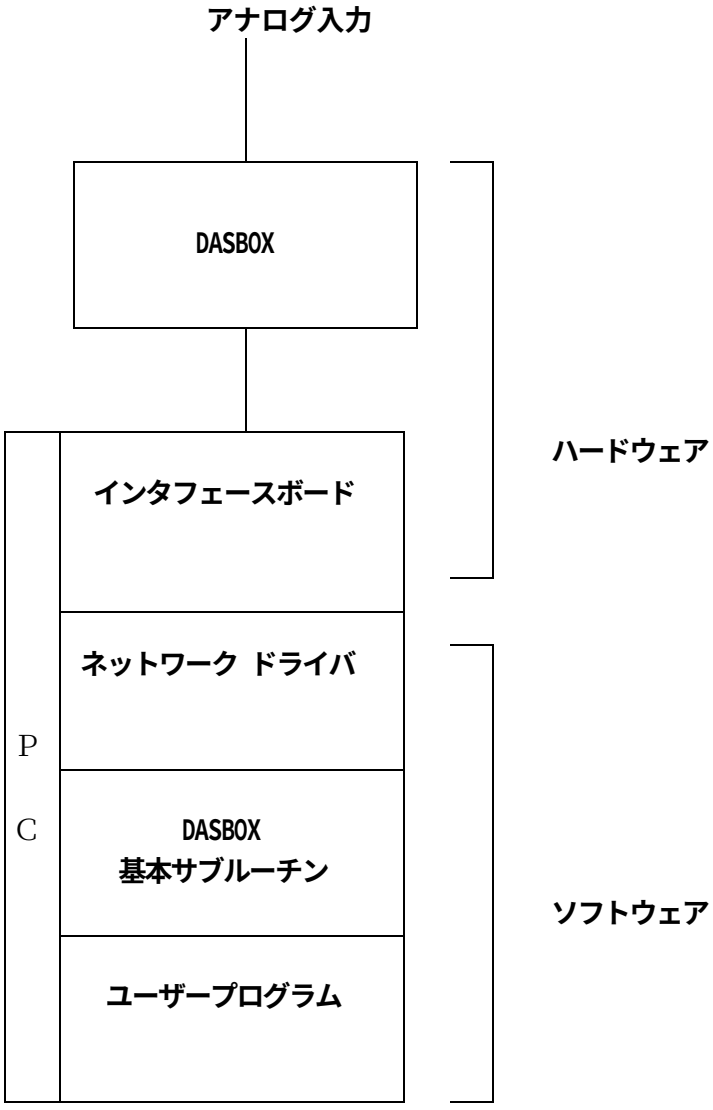
2003.10.15　3-14 inet_io_stat 関数説明修正
PC-SAD ボードを DASBOX に修正
下記の文を追加
また、計測実行中にこの関数を実行すると、計測が終了してしまいますので、計測中は使用しないでください。
ステータス値の変更

3-22 inet_io_out 関数の追加
3-23 inet_io_in 関数の追加
3-15 inet_io_stop 関数の機能訂正
3-16 inet_io_init 関数の機能訂正
3-11 inet_io_cond 関数の戻り値に AD 終了ステータスを追加

第1章 概　要

本サブルーチンは、DASBOXをＰＣ上で使用するためのプログラムです。

本サブルーチンは、DASBOXを使用するための基本操作を行うプログラムから構成されておりAD入力及びそれらに付随する設定をサブルーチンコールによって行います。



第２章 DASBOX 関数の説明

・inet_io_open()
　　DASBOXと接続されているデバイスをオープンします。

・inet_io_close()
　　DASBOXと接続されているデバイスをクローズします。

・inet_io_packet()
　　DASBOXに対して動作を指示します。

・inet_io_adread()
　　DASBOXからのADデータをメモリ上に読み込みます。

・inet_io_adfile()
　　DASBOXからのADデータをファイルに書き込みます。

・inet_io_dawrite()
　　DASBOXにDAデータをメモリ上から書き込みます。

・inet_io_dafile()
　　DASBOXへのDAデータをファイルから読み込みます。

・inet_io_dawait()
　　実際のDAが終了するまで待ち状態にします。

・inet_io_pre()
　　プリトリガが動作を行なった時、トリガ以前何データが有効データかホストに知らせます。

・inet_io_pre2()
　　プリトリガが動作を行なった時、トリガ以前の有効データ数とトリガ位置のずれデータ数をホストに知らせます。

・inet_io_cond()
　　トリガ入力の状態を取得します。

・inet_io_adstart()
　　DASBOXに対してADスタートコマンドを送信します。

・inet_io_dastart()
　　DASBOXに対してDAスタートコマンドを送信します。

・inet_io_stat()
AD/DA動作中にエラーが発生した時、DASBOXのエラーステータスを読み取ります。

・inet_io_stop()
DASBOXに対してストップコマンドを送信します。

・inet_io_init()
DASBOXに対してイニシャライズコマンドを送信します。

・inet_io_info()
DASBOXの現在の設定状態を得ます。

・inet_io_error()
サブルーチン呼び出しでエラーが発生した場合、エラー内容を返します。

・inet_io_blkf()
DASBOXデータの転送サイズを送信します。

・inet_amp_set()
アンプ関連項目の設定を行います。（オプション）

・inet_amp_cutoff()
フィルタ関連項目の設定を行います。（オプション）

第３章 DASBOX 関数の使用方法

・コントロールブロック

DASBOXを制御するための環境エリアとして本関数が使用する構造体をコントロールブロックと呼びます。コントロールブロックはDASBOX一台に対し、ひとつのコントロールブロックが必要になります。あらかじめ、ユーザーは構造体をstatic変数として定義する必要があります。このコントロールブロックに対するユーザーの書き込みは一切不要です。

・DASBOXアーギュメント

DASBOXに対して動作を指示する構造体でinet_io_packet()関数で使用します。

内容は第４章で説明します。

コントロールブロック及びDASBOXアーギュメントは、pci_sad.hファイルにその他の定義と共に納められています。ユーザーは、本関数を使用する場合pci_sad.hファイルをインクルードして使用します。

3-1　inet_io_open

機能：DASBOXと接続されているデバイスをオープンします。
　　他の関数は、本関数呼び出し後動作可能になります。

形式：

```
int  inet_io_open(cblock,  dasarg,  ip_address);

CBLK  *cblock;

PCISAD_ARG  *dasarg;

char  *ip_address;
```

引数：

cblock	コントロールブロック
dasarg	DASBOXアーギュメント
ip_address	Host NameかIP ADDRESSを直接指定します。

戻り値：

0	正常終了
-1	異常終了

3-2　inet_io_close

機能：DASBOXと接続されているデバイスをクローズします。

形式：

```
int  inet_io_close(cblock);

CBLK  *cblock;
```

引数：

cblock	コントロールブロック

戻り値：

0	正常終了
-1	異常終了

3-3 inet_io_packet

機能：DASBOXに対して動作を指示します。

形式：

```
int  inet_io_packet(cblock,  dasarg);

CBLK  *cblock;

PCISAD_ARG  *dasarg;
```

引数：

cblock	コントロールブロック
dasarg	DASBOXアーギュメント

戻り値：

0	正常終了
0以外	異常終了

3-4　inet_io_adread

機能：DASBOXからのADデータをメモリ上に読み込みます。

形式：

```
int  inet_io_adread(cblock,buffer,count);

CBLK  *cblock;

short  *buffer;

int  count;
```

引数：

cblock	コントロールブロック
buffer	ADデータを格納するバッファのポインタ
count	読み込むADデータ数

戻り値：

0	正常終了
-1	異常終了

3-5　inet_io_adfile

機能：DASBOXからADデータをファイルに書き込みます。

形式：

```
int  inet_io_adfile(cblock,  buffer,
                    size,  file_name,  frame);

CBLK  *cblock;

unsined  short  *buffer;

int  size;

char  *file_name;

long  frame;
```

引数：

cblock	コントロールブロック
buffer	テンポラリバッファのポインタ
size	データの転送サイズ（word）
file_name	書き込むファイル名
frame	書き込み数（word）

戻り値：

0	正常終了
-1	異常終了

3-6　inet_io_dawrite

機能：DASBOXにDAデータをメモリ上から書き込みます。

形式：

```
int  inet_io_dawrite(cblock,buffer,count);

CBLK  *cblock;

short  *buffer;

int  count;
```

引数：

cblock	コントロールブロック
buffer	ADデータを格納するバッファのポインタ
count	書き込むDAデータ数(word)

戻り値：

0	正常終了
-1	異常終了

3-7　inet_io_dafile

機能：DASBOXへのDAデータをファイルから読み込みます。

形式：

```
int  inet_io_dafile(cblock,  buffer,
                    size,  file_name,  frame);

CBLK  *cblock;

unsined  short  *buffer;

int  size;

char  *file_name;

long  frame;
```

引数：

cblock	コントロールブロック
buffer	テンポラリバッファのポインタ
size	データの転送サイズ（word）
file_name	読み込むファイル名
frame	読み込み数（word）

戻り値：

0	正常終了
-1	異常終了

3-8　inet_io_dawait

機能：実際にDAが終了するまで待ち状態にします。

形式：

```
int  inet_io_dawait(cblock,  buffer,  int  time);

CBLK  *cblock;

int  time;
```

引数：

CBLK	コントロールブロック
time	待ち時間（秒）

戻り値：

0	正常終了
-1	待ち時間以内にDAが終了しなかった
-2	異常終了

3-9 inet_io_pre

機能：

プリトリガAD動作完了後、トリガ以前の無効データ数を得ます。

predataは設定されたプリトリガサイズに対し、無効であったデータ数を返します。プリトリガデータが全て有効であれば0を返します。

形式：

```
int  inet_io_pre(cblock,predata);

CBLK        *cblock;
int         *predata;
```

引数：

cblock	コントロールブロックのポインタ
predata	無効データ数ポインタ

戻り値：

0	正常終了
-1	異常終了

3-10　inet_io_pre2

機能：

プリトリガAD動作完了後、トリガ以前の無効データ数とトリガ位置のずれデータ数を得ます。

predataは設定されたプリトリガサイズに対し、無効であったデータ数を返します。プリトリガデータが全て有効であれば0を返します。

形式：

```
int  inet_io_pre(cblock,predata,adjust);

CBLK        *cblock;
int         *predata;
int         *adjust;
```

引数：

cblock	コントロールブロックのポインタ
predata	無効データ数ポインタ
adjust	ずれデータ数ポインタ

戻り値：

0	正常終了
-1	異常終了

3-11　inet_io_cond

機能：

計測中にDASBOXの状態を取得します。

このステータスはいつでも読み出すことが出来るため､ステータス情報の正当性は、inet_io_adstart()関数の呼び出し後、ADもしくはDA動作が完了するまでの期間となります。

形式：

```
int inet_io_cond(cblock,  data);

CBLK        *cblock;
int         *data;
```

引数：

cblock	コントロールブロックのポインタ
data	トリガー有り無し及びAD終了のステータス トリガステータス（０ｘ４） ビット2=0　トリガー無し ビット2=1　トリガー検出。 AD終了ステータス（０ｘ８） ビット3=0　AD実行中 ビット3=1　AD終了。

戻り値：

0	正常終了
-1	異常終了

3-12　inet_io_adstart

機能：DASBOXへADスタートコマンドを送信します。

形式：

```
int  inet_io_adstart(cblock);

CBLK  *cblock;
```

引数：

cblock	コントロールブロック

戻り値：

0	正常終了
-1	異常終了

3-13　inet_io_dastart

機能：DASBOXへDAスタートコマンドを送信します。

形式：

```
int  inet_io_dastart(cblock);

CBLK  *cblock;
```

引数：

cblock	コントロールブロック

戻り値：

0	正常終了
-1	異常終了

3-14　inet_io_stat

機能：DASBOXの動作状態のステータスを取得します。
次のデータ転送関数inet_io_adread()、inet_io_dawrite()やda_wait()
などの制御関数でエラーが発生した場合、inet_io_stat()でエラーの詳細を
突き止めます。
尚、一度通知したエラーは通知したときにクリアされます。
また、計測実行中にこの関数を実行すると、計測が終了してしまいますので、
計測中は使用しないでください。

形式：

```
int  inet_io_stat(cblock,  dasarg);

CBLK  *cblock;

PCISAD_ARG  *dasarg;
```

引数：

cblock	コントロールブロック
dasarg	daboxアーギュメント

戻り値：

0	正常終了
-1	異常終了
8000（HEX）	PCI_TIMEOUT
8001（HEX）	OVER_SAMPLING

3-15　inet_io_stop

機能：DASBOXを初期状態（パワーオン）に戻します。
　　設定されたパラメータは消去されますので、再度パラメータ設定が必要です。

形式：

```
int  inet_io_stop(cblock,  dasarg);

CBLK  *cblock;

PCISAD_ARG  *dasarg;
```

引数：

cblock	コントロールブロック
dasarg	dasbox　アーギュメント

戻り値：

0	正常終了
-1	異常終了

3-16　inet_io_init

機能：DASBOXに対してインシャライズコマンドを送信し、初期状態（パワーオン）に戻します。
設定されたパラメータは消去されますので、再度パラメータ設定が必要です。

形式：

```
int  inet_io_init(cblock,  dasarg);

CBLK  *cblock;

PCISAD_ARG  *dasarg;
```

引数：

cblock	コントロールブロック
dasarg	dasboxアーギュメント

戻り値：

0	正常終了
-1	異常終了

3-17　inet_io_info

機能：DASBOXの現在の設定情報を得ます。

形式：

```
int  inet_io_info(cblock,  statptr,  size);

CBLK  *cblock;

PCISAD_INFO  *statptr;

int  size
```

引数：

cblock	コントロールブロック
statptr	情報データポインタ
size	読み込む情報量 (word))

戻り値：

0	正常終了
-1	異常終了

```
PCISAD_INFO  {
int                     fifo;
unsigned  short         fifo_wide;
unsigned  int           input_channel;
unsigned  int           output_channel;
unsigned  int           board_id;
unsigned  short         module_input;
unsigned  short         module_output;
}
```

3-17-1 fifo

DASBOXに実装されているFIFOのワードサイズが入ります。。

3-17-2 fifo_wide

DASBOXに実装されているFIFOのビット長が入ります。
　現在は固定長の１６が入っています。

3-17-3 input_channel

DASBOXのADチャネル数が入ります。

3-17-4 output_channel

DASBOXのDAチャネル数が入ります。

3-17-5 board_id

PC-SADボードのIDが入ります。

3-17-6 module_input

DASBOXのADモジュールレジスタ内容が入ります。
システムが使用します。

3-17-7 module_output

DASBOXのDAモジュールレジスタ内容が入ります。
システムが使用します。

3-18　inet_io_error

機能：サブルーチン呼び出しでエラーが発生した場合エラー内容を文字列で返します。

形式：

```
char  *inet_io_error(cblock);

CBLK  *cblock;
```

引数：

cblock	コントロールブロック

戻り値：

文字列へのポインタ

3-19　inet_io_blkf

機能：ADデータの転送数を指定します。

形式：

```
int  inet_io_blkf(cblock,  size)

CBLK  *cblock;
int  size;
```

引数：

cblock	コントロールブロック
size	転送サイズ

戻り値：

0	正常終了
-1	異常終了

3-20 inet_amp_set

機能： 指定したチャネルにアンプ関連項目の設定を行います。

形式：

```
int inet_amp_set (cblock, afarg, channel);

CBLK   *cblock;
struct af_arg   *afarg;
struct int   channel;
```

引数：

cblock コントロールブロック

afarg PCI-AF8 アーギュメント

gain[128]	ゲイン
offval[128]	オフセット電圧値
offmode[128]	オフセット電圧値のレベル
input[128]	AC/DC 切替
imode[128]	SIGNAL/GND 切替

channel 設定チャネル番号

0	=	All Channel
1 ~ 128	=	Select Channel

戻り値：

0	正常終了
-1	異常終了

補足：

本関数はアンプフィルタ機能を持った DASBOX で有効です。
PCI-AF8 アーギュメントの説明は 3-21.inet_amp_cutoff の項に書いています。

3-21 inet_amp_cutoff

機能： 全てのチャネルにフィルタ関連項目の設定を行います。

形式：

```
int inet_amp_cutoff(CBLK                *cblock,
                    struct af_arg       *afarg);

CBLK   *cblock;
struct af_arg   *afarg;
```

引数：

cblock コントロールブロック
afarg PCI-AF8 アーギュメント
cutoff カットオフ周波数

戻り値：

0 正常終了
-1 異常終了

補足：

本関数はアンプフィルタ機能を持った DASBOX で有効です。

af_arg （PCI-AF8 アーギュメント）

PCI-AF8 のアンプ・フィルタ機能の設定を行うには、inet_amp_set 関数, inet_amp_cutoff 関数を使用します。inet_amp_set 関数, inet_amp_cutoff 関数の第２パラメータ(af_arg)が PCI-AF8 のアンプ・フィルタ機能の設定内容になっています。

```
struct af_arg
{
    int     gain[128];
    int     offval[128];
    int     offmode[128];
    int     input[128];
    int     imode[128];
    int     pathen[128];
    int     cutoff;
    int     calsel;
    int     revolution;
    int     pulse;
    int     change;
    int     average;
    int     cmplevel;
    float   teibai;
};
```

説明

gain[128]

指定したチャネルに対してアンプゲイン設定を行います。

配列	設定チャネル
[0]	１チャネル
[1]	２チャネル
｜	｜
[127]	１２８チャネル

設定値	入力レベル	
0	±10V	(0.5 倍)
1	± 5V	(1 倍)
2	±2.5V	(2 倍)
3	±1.25V	(4 倍)
4	±625mV	(8 倍)
5	±312.5mV	(16 倍)
6	±156.25mV	(32 倍)
7	±78.25mV	(64 倍)

offval[128]　　オプション

オフセット設定機能は、オプションです。通常は、全チャネル、ゼロを指定して下さい。
指定したチャネルに対してオフセット電圧を指定します。
設定値は offmode の値により異なります。

配列	設定チャネル
[0]	１チャネル
[1]	２チャネル
｜	｜
[127]	１２８チャネル

設定値	指定電圧値 (offmode:Couse/Fine)
127(7f)	+FS (+5V / +0.5V)
｜	｜
0	0 (0V)
｜	｜
-128(80)	-FS (-5V / -0.5V)

offmode[128]　　オプション

オフセット設定機能は、オプションです。通常は、全チャネル、ゼロを
指定して下さい。
指定したチャネルに対してオフセット電圧値のレベルモードを設定します。

配列	設定チャネル
[0]	１チャネル
[1]	２チャネル
｜	｜
[127]	１２８チャネル

設定値	オフセットレベル
0	±5.0V　(Couse)
1	±0.5V　(Fine)

input[128]

指定したチャネルの入力モード(AC/DC)を設定します。

配列	指定チャネル
[0]	１チャネル
[1]	２チャネル
｜	｜
[127]	１２８チャネル

設定値	入力モード
0	DC.Coupling
1	AC.Coupling

imode[128]

指定したチャネルの入力モード(SIGNAL/GND)を設定します。

配列	指定チャネル
[0]	１チャネル
[1]	２チャネル
｜	｜
[127]	１２８チャネル

設定値	入力モード
0	シグナル
1	GND

pathen[128]

予約

cutoff

全チャネル共通フィルタのカットオフ周波数を設定します。

設定値	カットオフ周波数	
0	10Hz	
1	20Hz	
2	50Hz	
3	100Hz	
4	200Hz	
5	500Hz	
6	1kHz	
7	2kHz	
8	5kHz	
9	10kHz	
１０	20kHz	
１１	25kHz	
１２	-----	
１３	-----	
１４	-----	
１５	EXT.Mode	（オプション）

calsel

予約

revolution

予約

pulse

予約

change

予約

average

予約

cmplevel

予約

teibai

予約

3-22 inet_io_out

機能：オプションにて DO ポートを追加された場合有効となり、DO ポートにデータを出力します。

形式　inet_io_out（cblock,p_wirte_data,wide,select）

```
CBLK          *cblock
unsigned int   p_write_data
int            wide
int            select
```

引数

cblock	コントロールブロック
p_write_data	出力データ（long 以外は 32Bit の下位から有効） 実際の出力は負論理となります。 パワーオン時全て“ OFF”、” H” “ １ ”をセットにて外部は“ L”
wide	データ幅 ０ ＝ BYTE（８ビット） １ ＝ WORD（１６ビット） ２ ＝ LONG（３２ビット）
select	各 wide 設定により異なる

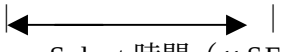
wide	select	設定
0	0	LL 設定（D0～D7）
0	1	LH 設定（D8～D15）
0	2	HL 設定 (D16～D23)
0	3	HH 設定 (D24～D31)
1	0	L 設定（D0～D15）
1	1	H 設定（D16～D31）
2	0	ノーマル（D0～D31）
2	0以外	31 ビット目パルス

wide=2、select が”0”以外のときの説明

32Bit 目（D31）をコントロールビットとして、以下の動作をファームウェアにて行う。

DATA（D0～D30）=> 設定データ出力　　設定データ出力

D31　=>　D31　　D31 の反転出力



Select 時間（μSEC 単位）

但し、DASBOX の内部ファームの都合上、２μSEC 加算された時間となります。 例 １－＞３μSEC、１０－＞１２μSEC

＊ 注）wide=2、select が”0”以外の場合は、select 時間終了後に関数から戻ります。

戻り値

0	正常終了
-1	異常終了

3-23 inet_io_in

機能：オプションにて DI ポートを追加された場合有効となり、DI ポートからデータを入力します。

形式　inet_io_in（cblock,p_read_data）

CBLK	*cblock
unsigned int	*p_read_data

引数

cblock	コントロールブロック
p_read_data	入力データを格納する変数のポインタ（32Bit） 負論理入力となります。 外部が“H”で”０“

戻り値

０	正常終了
-1	異常終了

第４章 ＤＡＳＢＯＸアーギュメント説明

DASBOXを動作させるには、inet_io_packet関数を使用して行います。
inet_io_packet関数の第２パラメータがDASBOXの動作内容になっています。

```
struct  inet_io_arg
        {
        unsigned  short   mode;
        unsigned  short   stat;
        unsigned  short   dastime;
        unsigned  short   attn;
        unsigned  short   gain;
        unsigned  short   trgslp;
        unsigned  short   clkmode;
        long              fifo;
        float             clock;
        unsigned  long    frame1;
        unsigned  long    frame2;
        unsigned  long    frame3;
        unsigned  long    frame4;
        unsigned  long    frame5;
        unsigned  long    mutelevel;
        unsigned  short   trglevel;
        unsigned  short   trgsrc;
        unsigned  short   trgch;
        unsigned  short   channels;
        unsigned  short   chanum[256];
        }
```

4-1 mode

DASBOXに対する動作を指定します。

defile名	値	動作
ADNORM	0	ADノントリガスタートモード
ADTRG	1	ADノーマルトリガスタートモード
ADPRE	2	ADプリトリガスタートモード
ADRET	3	ADノーマルリトリガスタートモード
DANORM	4	DAノントリガスタートモード
DATRG	5	DAノーマルトリガモード
ADPOST	10	ADポストトリガスタートモード
ADREPOST	11	ADポストリトリガスタートモード

4-2 stat

ステータスコマンドを実行させた時にDASBOXから送られたステータスが
格納されるメンバーです。

値	ステータス
0	正常
0x800	バッファエラー

4-3 dmatime

inet_adread,inet_dawrite,inet_adfile,inet_dafile時の転送タイムアウト
時間を設定します。
inet_io_blkfで設定する転送数に合わせてください。
例：blkf=1000000, Sampling Clock=10000Hzの場合
blkf / Sampling Clock = 100秒以上の値を設定。

値	動作
0	基本サブルーチンのデフォルト値になります。
1～n	秒単位になります。

4-4 attn（未使用）

DASBOXにATTN出力の指示を与えます。

4-5 gain（未使用）

DASBOXの入力GAINの指示を与えます。

4-6 trgslp

各トリガモードを選択した場合有効になりトリガのスロープ指示を行います。

値	動作
0以外	立ち上がりでトリガ
0	立ち下がりでトリガ

4-7 clkmode

サンプリングクロックのソースを指定します。

defile名	値	動作
INT8000	1	内部クロックを8MHzを使用
INT8192	2	内部クロックを8.192MHzを使用
INT6144	3	内部クロックを6.144MHzを使用
INT5644	4	内部クロックを5.644MHzを使用
EXTCLK	5	外部からクロックを入力する
EXTDIV	6	外部からクロックを分周する

4-8 clock

内部クロック及び外部分周を指定した場合、周波数をHz単位で指定します。

4-9 frame1

AD又はDAするデータ量を指定します。単位はワードです。

4-10 frame2

プリトリガ動作もしくはポストトリガ動作を指定した場合有効です。
プリトリガ動作の時はトリガ以前のデータ量を指定します。
ポストトリガ動作の時はトリガ以降の遅延データ量を指定します。
単位はワードです。

4-11 frame3

DA動作を指定した場合有効で実際に起動する前に転送するデータ量を指定します。単位はワードです。

4-12 frame4

常に０を指定して下さい。

4-13 frame5

modeがADRETRG及びADREPOSTのときトリガ回数を指定します。
それ以外のmode時は１を設定して下さい。

4-14 mutelevel

未使用。

4-15 trglevel

DASBOXがトリガレベルを機能を持っている時、トリガモードを指定した場合有効でトリガがかかる電圧を指定します。

値		電圧
0x80 128		+FS(6/64)又は+FS(100/100)v
0x40 64	又は100	0v
0		-FS(64/64)v(1LSB=2FS/128)又は-FS(100/100)v

4-16 trgsrc

値	動作
0	AD入力チャンネルトリガ（未使用）
1	外部トリガ端子

4-17 trgch

値	動作
1～16	AD入力チャンネル番号

4-18 channels

AD又はDAするチャンネル数を指定します。

範囲　1～256

4-19 chanum【256】

AD又はDAするチャンネル番号を指定します。

第５章 ＤＡＳＢＯＸソフトウェア作成

5-1 インストール

提供メディアからキットを適当なディレクトリにコピーしてください。

UNIX (LINUX)の場合

提供メディアからキットを適当なディレクトリにコピーしてください。

FloppyDiskの場合のコピー方法

```
mount /mnt/floppy
tar xvf /mnt/floopy/kit.tar
```

Windowsの場合

適当なディレクトリ（例えばD:\das5x0）を作成して提供メディアから\das5x0
以下のディレクトリを作成したディレクトリにコピーしてください。

5-2 キット内容

キットにはDASBOXを動作させるための基本サブルーチン及びサンプル
ソフトが入っています。
ディレクトリ構成は以下のようになっています。

```
kit/
    inetlib/   sample/
```

また各ディレクトリには以下のファイルがあります。

```
inetlib/ ・・・基本サブルーチンディレクトリ
  inet_sad_lib.c   ・・・基本サブルーチンソースファイル
  pci_sad.h        ・・・基本サブルーチンインクルードファイル
  pci_sad_reg.h    ・・・基本サブルーチンインクルードファイル
  sadtp.h          ・・・ユーザープログラムインクルードファイル

sample/ ・・・サンプルソフトディレクトリ
  Makefile         ・・・UNIX(LINUIX)用サンプルソフトメイクファイル
  Mk.cmd           ・・・Windows用サンプルソフトメイクファイル
  apl90.c          ・・・サンプルソフトソースファイル
  sad              ・・・サンプルソフト実行ファイル
```

5-3 ユーザープログラムへの組み込み

DASBOX-Model5x0のアプリケーションを作成する場合はインクルードファイル
"pci_sad.h"をユーザープログラムへ追加してください。
そしてリンク時に基本サブルーチン"inet_sad_lib.c"をリンクしてください。
UNIX(LINUX)の場合はコンパイルオプションは特にありません。サンプルのMakefile
を参照してください。
Windowsの場合はWINNTをdefineしてwsock32.libとuser32.libをリンクして
ください。サンプルのmk.cmdを参照してください。

5-4　サンプルソフト使用方法

基本サブルーチンを使用したサンプルプログラムの使用方法を説明します。
ファイル名はsadです。

構文　　sad　mode[,parameter_file][,data_file]　[hostname]

modeはDASBOXに対する動作の指定です。

INIT	DASBOXにイニシャライズコマンドを送ります。
STOP	DASBOXにストップコマンドを送ります。
STATus	DASBOXのステータスを表示します。
ADNorm	ADノントリガスタートモードを起動します。
ADTRg	ADトリガスタートモードを起動します。
ADPre	ADプリトリガスタートモードを起動します。
DANorm	DAノントリガスタートモードを起動します。
DATRg	DAトリガスタートモードを起動します。

小文字の部分は省略可能です。

parameter_fileはサンプリング周波数やデータ量などの情報を含んだファイルです。ファイルがすでに存在している場合はそのファイルを読み込み、存在しない場合はパラメータ入力を促しファイルを作成します。
parameter_fileを指定しない場合は"def.par"というファイルが作成されます。

data_fileはADする場合ADデータを格納するファイルになり、DAをする場合転送するDAデータのファイルになります。
data_fileを指定しなかった場合はADの場合バッファに読み込まれるだけです。
DAの場合12bitのインクリメントパターンを出力します。

HostnameはDASBOXのホスト名もしくはIPアドレスを指定します。