

DASBOX Model-Eシリーズ DASmini-E2000シリーズ

基本サブルーチン仕様書

for

LINUX／UNIX／Windows

作成	平成15年	4月
改訂	平成15年	5月 6日
改訂	平成15年	7月24日
改訂	平成16年	5月24日
改訂	平成16年	12月20日
改訂	平成17年	1月21日
改訂	平成17年	1月27日

システムデザインサービス株式会社

目 次

改訂履歴.....	3
第1章 概 要.....	5
第2章DASBOX関数の説明.....	6
2.1 AD/DA (入出力) 共通.....	6
2.2 AD (入力) 専用.....	7
2.3 DA(出力)専用.....	7
第3章DASBOX関数の使用方法.....	8
3.1 AD/DA (入出力) 共通.....	9
3.1.1 inet_io_open.....	9
3.1.2 inet_io_close.....	10
3.1.3 inet_io_packet.....	11
3.1.4 inet_io_cond.....	12
3.1.5 inet_io_stat.....	13
3.1.6 inet_io_stop.....	14
3.1.7 inet_io_init.....	15
3.1.8 inet_io_info.....	16
3.1.9 inet_io_error.....	18
3.1.10 inet_io_blkf.....	19
3.1.11 inet_amp_set.....	20
3.1.12 inet_amp_cutoff.....	21
3.1.13 inet_amp_bcutoff.....	22
アンプフィルタ設定アーギュメントの説明 (PCI-AF8 アーギュメント)	23
フィルタ設定アーギュメントの説明 (DAS-16AF-Aアーギュメント)	26
3.1.14 inet_io_info2.....	27
3.2 AD (入力) 専用.....	29
3.2.1 inet_io_adstart.....	29
3.2.2 inet_io_adread.....	30
3.2.3 inet_io_adfile.....	32
3.2.4 inet_io_pre.....	33
3.3 DA (出力) 専用.....	34
3.3.1 inet_io_dastart.....	34
3.3.2 inet_io_dawrite.....	35
3.3.3 inet_io_dafile.....	37
3.3.4 inet_io_dawait.....	38
3.3.5 inet_io_cycle_stop.....	39
3.3.6 inet_io_daclear.....	40
3.4 サブルーチン実行基本フロー	41
3.4.1 AD動作.....	41
3.4.2 DA動作.....	42
第4章 DASBOXアーギュメント説明.....	43
4-1 mode.....	44
4-2 stat.....	45
4-3 dmatime.....	45

4-4	attn (未使用)	45
4-5	gain (未使用)	45
4-6	trgslp	46
4-7	clkmode	46
4-8	clock	46
4-9	frame1	47
4-10	frame2	47
4-11	frame3	47
4-12	frame4	47
4-13	frame5	48
4-14	mutelevel	48
4-15	trglevel	48
4-16	trgsrc	48
4-17	trgch	48
4-18	channels	49
4-19	chanum 【1024】	49
4-20	master,ext_parm,dmasize,pre_mem_size	49
第5章 <i>DASBOX</i>ソフトウェア作成		50
5-1	インストール	50
5-2	キット内容	50
5-3	ユーザープログラムへの組み込み	51
5-4	サンプルソフト使用方法	52

改訂履歴

・平成 15 年 4 月 初版作成

・平成 15 年 5 月 6 日

3.1.5 inet_io_stat 関数の戻り値変更 (他機種と統一する為)		
変更前	変更後	エラーの内容
8000 (DEC)	8000 (HEX)	SDS_PCI_FIFO_OVERFLOW
8001 (DEC)	8001 (HEX)	SDS_PCI_OVER_SAMPLING
8002 (DEC)	8002 (HEX)	SDS_PCI_ADDA_ABORT
8003 (DEC)	8003 (HEX)	SDS_PCI_TIMEOUT

平成 15 年 7 月 24 日

4-3 dmatime

変更前

値	動作
0	基本サブルーチンのデフォルト値になります。

変更後

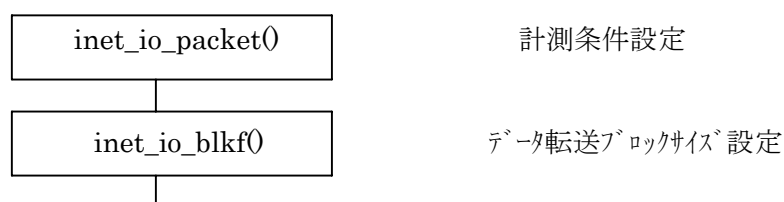
値	動作
0	基本サブルーチンのデフォルト値 (30) になります。 但し、基本サブルーチンのVer2.03からの対応となります。 Ver2.01, Ver2.02の場合は”0”がそのまま使用され、inet_io_adread、Inet_io_adfile、inet_io_dawirte、inet_io_dafire関数にて、異常終了となります。

平成 16 年 5 月 24 日

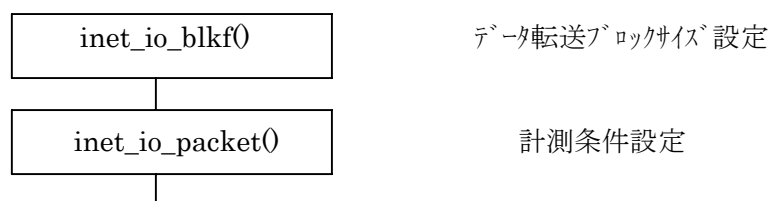
3.4 サブルーチン実行基本フロー

3.4.1 AD動作及び 3.4.2 DA動作の inet_io_packet()と Intet_io_blkf()の順番を入れ替えた。

<修正前>



<修正後>



平成16年12月20日

3.1.13 `inet_amp_bcutoff` 関数を追加。

平成17年1月21日

DASmini-E2000 シリーズサポートに伴い説明追加

平成17年1月27日

3.1.14 `inet_io_info2` 関数を追加

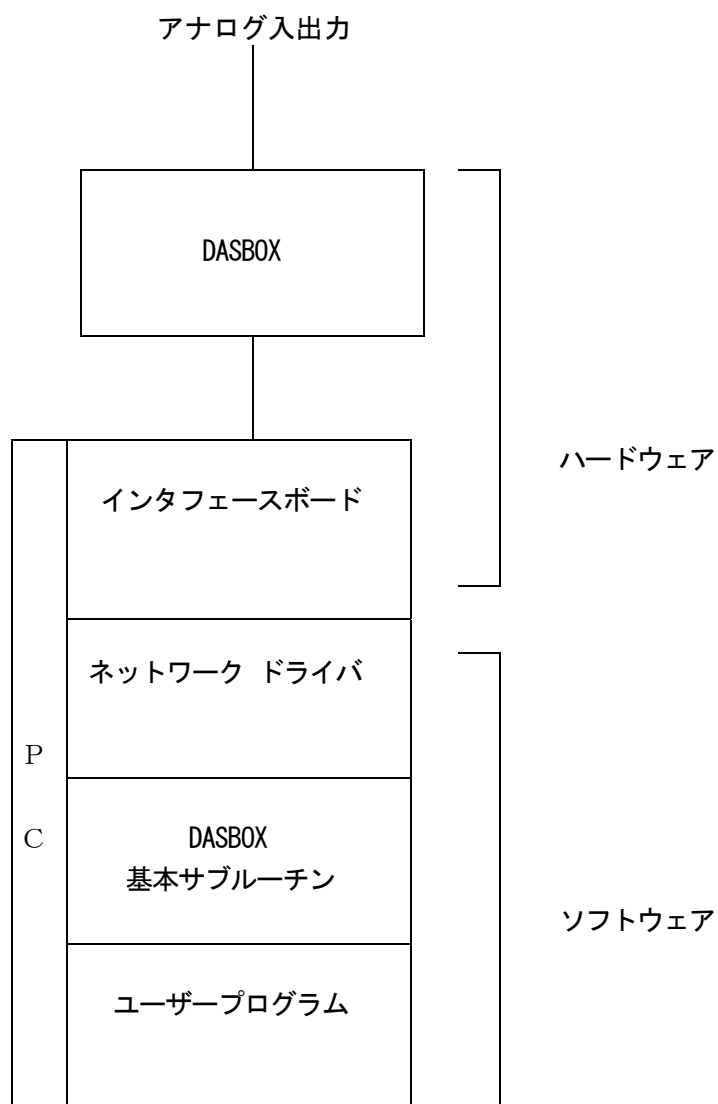
サンプルプログラムにアンプ・フィルタ設定を追加した為、説明追加
DASBOX アーギュメント説明 `unsigned short dastime;` → `dmatime`

3.3.6 `inet_io_daclear` 関数を追加

第1章 概要

本サブルーチンは、**DASBOX Model-E**シリーズ及び**DASmini-E2000**シリーズ（以降**DASBOX** と表記）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、DA 出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

2.1 AD/DA (入出力) 共通

- [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_cond\(\)](#)
DASBOXの状態 (トリガ等) を取得します。
- [inet_io_stat\(\)](#)
DASBOXが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の計測動作を中止させ、コマンド待ちの状態となります。
動作設定内容は保持されます。
- [inet_io_init\(\)](#)
DASBOXを初期状態に戻します。
動作設定内容は無効となり、新たに動作指示を行う必要があります。
- [inet_io_info\(\)](#)
DASBOXのインフォメーションを得ます。
- [inet_io_info2\(\)](#)
DASBOXのバージョン情報を得ます。
- [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\(\)](#)
データ転送のブロックサイズを設定します。
- [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)

- [inet_amp_bcutoff\(\)](#)
ブロック単位で、フィルタ関連項目の設定を行います。（オプション）

2.2 AD（入力）専用

- [inet_io_adstart\(\)](#)
DASBOXに対してAD（入力）動作を開始させます。
- [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_pre\(\)](#)
トリガが動作を行なった時、トリガ以前何データが無効データかホストに知らせます。

2.3 DA(出力)専用

- [inet_io_dastart\(\)](#)
DASBOXに対してDA（出力）動作を開始させます。
- [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、DASBOXにデータを送信します。
- [inet_io_dawait\(\)](#)
実際のDAが終了するまで待ち状態にします。
- [inet_io_cycle_stop\(\)](#)
DASBOXのDAサイクル動作をフレームの区切り目で終了させます。
- [inet_io_daclear\(\)](#)
DASBOXのDAサイクル動作をフレームの区切り目で終了させます。

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()関数等で使用します。

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 AD/DA (入出力) 共通

3.1.1 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のポインタ

dasarg DASBOXアーギュメントのポインタ

ip_address Host NameかIP ADDRESSを直接指定します。
 キャラクタのポインタ

戻り値 :

0 正常終了

-1 異常終了

3.1.2 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のポインタ
--------	-----------------

戻り値 :

0	正常終了
---	------

-1	異常終了
----	------

3.1.3 inet_io_packet

機能 : DASBOXに対して動作を指示します。設定内容に関しては、第4章のDASBOXアーギュメント説明を参照願います。

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のポインタ

dasarg DASBOXアーギュメントのポインタ

戻り値 :

0	正常終了
-1	modeの設定エラー
-2	clockの設定エラー
-4	clkmodeの設定エラー
-5	channelsの設定エラー
-6	chanum[1024]の設定エラー
-7,-8	パケットコマンドエラー

3.1.4 inet_io_cond

機能： 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(),inet_io_dastart()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 ステータスを格納する変数のポインタ

DASBOXの状態を返します。各ビット毎に意味を持ちます。

31~16	15	14	13	12	11	10	9	8	7	6	5	4~0
*	*	*	*	*	N E M	*	E R R	*	E N	*	T R G	*

* 印のビットは不定データとなります。

Bit11: NEM

DASBOXのFifoが空でない時に“1”となります。

Bit9: ERR

DASBOXが計測中にサンプリングエラー又はFiFoオーバーフローエラーとなった場合に“1”となります。

inet_io_adstart(),inet_io_dastart(),inet_io_init(),inet_io_stop()で、“0”クリアされます。

Bit7: EN

DASBOXが計測中に“1”となり、計測終了すると“0”となります。

Bit5: TRG

計測スタート時に“0”クリアされ、トリガを使用するモードの時に、トリガを受信すると“1”となります。

リトリガモードの場合は、1回目のトリガで“1”となりますので、2回目以降のトリガのステータスとしては使用できません。

2回目以降のトリガの判断は、NEMビットを使用し、1フレーム分のデータを読み込んでも、NEMビットが“1”の場合は次のトリガを受信して、データが格納されたと判断します。

戻り値：

0 正常終了

-1 異常終了

3.1.5 inet_io_stat

機能 : DASBOXに動作エラーが発生した時のステータスを取得します。
データ転送関数inet_io_adread()、inet_io_dawrite()やinet_io_dawait()
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。

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のポインタ

dasarg dasboxアーギュメントのポインタ

戻り値 :

0	正常終了
-1	異常終了
8000 (HEX)	SDS_PCI_FIFO_OVERFLOW
8001 (HEX)	SDS_PCI_OVER_SAMPLING
8002 (HEX)	SDS_PCI_ADDA_ABORT
8003 (HEX)	SDS_PCI_TIMEOUT

3.1.6 inet_io_stop

機能 : DASBOXの計測を強制終了させます。設定されたパラメータは保持されますので、inet_io_packet()にて再度パラメータを設定する必要はありません。但し、DASBOX内のメモリー内に取り込まれているデータは消去されます。

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
----	------

3.1.7 inet_io_init

機能 : DASBOXを初期状態に戻し、設定されているパラメータは消去されます。
計測中は、強制終了しDASBOX内のメモリー内に取り込まれているデータは消去されます。

形式 :

```
int inet_io_init(cblock);
```

```
CBLK *cblock;
```

引数 :

cblock コントロールブロックのポインタ

戻り値 :

0 正常終了

-1 異常終了

3.1.8 inet_io_info

機能：DASBOXの内部情報を得ます。

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のポインタ

statptr 情報データのポインタ

size 読み込む情報量 (word)
通常はboard_idまでが有効ですので、9を設定してください。

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;  
}
```

1) fifo

DASBOXに実装されているFIFOのワードサイズが入ります。。

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) **input_channel**

DASBOXの最大ADチャンネル数が入ります。

4) **output_channel**

DASBOXの最大DAチャンネル数が入ります。

5) **board_id**

DASBOX-Eシリーズの場合、ID (DASBOXのモジュール設定)が入ります。
詳細はDASBOX-Eシリーズ ハードウェアマニュアルの第3章のS1設定
を参照願います。

“ON”で、” 1 “

S 1			
4	3	2	1
D3	D2	D1	D0

DASmini-E2000シリーズの場合は不定データとなります。

6) **module_input**

未使用 (拡張用)

7) **module_output**

未使用 (拡張用)

3.1.9 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のポインタ
```

戻り値：

```
文字列へのポインタ
```

3.1.10 inet_io_blkf

機能： 基本的には、1チャンネルに対する1回のinet_io_adread()及びinet_io_dawrite()のデータの転送数（ブロック数）を指定します。inet_io_adstart()及びinet_io_dastart()の前に1度だけ指定してください。データ転送中に変更した場合は、保持されますが、次のinet_io_adstart()及びinet_io_dawrite()から有効となります。

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のポインタ
size	1チャンネルに対しての、1回のブロック（データ転送）サイズを指定します。単位はワードです。 設定範囲： 1～2147483648（2G）

戻り値：

0 正常終了

-1 異常終了

補足： DASBOX内部には、ハードウェアのFiFoメモリー及びファームウェアの中間バッファがあります。転送効率を上げるため、ファームウェアはホストからのデータ転送要求がなくても、データの先読みを行います。Sizeの設定はこの時の、データの先読みデータ量の指定も兼ねております。よってAD動作の場合、size×計測チャンネル数指定とinet_io_adread()の読み込み数（count）が一致しない場合例1のような転送となります。

例1 サンプルクロック=1 Hz、size=2、計測チャンネル数=10、count=10
チャンネルに対して2データずつ読み込みを行うため、
inet_io_adread（）を連続に行った場合に、戻り時間が約2秒、1秒以下、約2秒、1秒以下のくり返しとなる。

例2 サンプルクロック=1 Hz、size=1、計測チャンネル数=10、count=10
チャンネルに対して1データずつ読み込みを行うため、
inet_io_adread（）の戻り時間が約1秒のくり返しとなる。

3.1.11 inet_amp_set

機能： 指定したチャンネルにアンプ関連項目の設定を行います。
DASBOX-E シリーズに対応したアンプフィルタモジュール、
DASmini-E2000 の “A F” モデルに対応しています。

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のポインタ  
afarg PCI-AF8 アーギュメントのポインタ  
下記のアーギュメントメンバーが有効となります。  
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 設定チャンネル番号
```

```
0 = All Channel  
1 ~ 128 = Select Channel
```

戻り値：

```
0 正常終了  
-1 異常終了
```

補足：

本関数はアンプモジュールを内蔵した DASBOX で有効です。
PCI-AF8 アーギュメントの説明は 3.1.12 inet_amp_cutoff の項に
を参照願います。

3.1.12 inet_amp_cutoff

機能： 全てのチャンネルにフィルタ関連項目の設定を行います。
DASBOX-E シリーズに対応したアンプフィルタモジュール、
DASmini-E2000 の “A F” モデルに対応しています。

形式：

```
int inet_amp_cutoff(cblock,afarg);
```

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

引数：

```
cblock   コントロールブロックのポインタ  
afarg    PCI-AF8 アーギュメントのポインタ  
         下記のアーギュメントのメンバーが有効となります。  
         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;  
};
```

3.1.13 inet_amp_bcutoff

機能： 8チャンネルのブロック単位にフィルタ関連項目の設定を行います。
DASBOX-Eシリーズに対応したアンプフィルタモジュール、
DASmini-E2000の“A F”モデルに対応しています。

形式：

```
int inet_amp_bcutoff(cblock,barg,block);
```

```
CBLK *cblock;  
struct bcutoff_arg *barg;  
int block;
```

引数：

cblock	コントロールブロックのポインタ
barg	bcutoff_arg アーギュメントのポインタ 下記のアーギュメントのメンバーが有効となります。 Cutoff カットオフ周波数
block	設定するブロックを指定します。 例) block = 0 ...全てのチャンネル (cutoff[0]~[15]) block = 1 ...1チャンネル~8チャンネル (cutoff[0]) block = 2 ...9チャンネル~16チャンネル (cutoff[1])

戻り値：

0	正常終了
-1	異常終了

補足：

本関数はフィルタ機能を持った DASBOX で有効です。

bcutoff_arg (DAS-16AF-A アーギュメント)

DAS-16AF-A のフィルタ機能の設定を行うには、inet_amp_bcutoff 関数を使用します。inet_amp_bcutoff 関数の第2パラメータ(barg)が DAS-16AF-A のフィルタ機能の設定内容になっています。

```
struct bcutoff_arg  
{  
    int    cutoff[16];  
};
```

アンプフィルタ設定アーギュメントの説明 (PCI-AF8 アーギュメント)

1) gain[128]

指定したチャンネルに対してアンプゲイン設定を行います。

配列	設定チャンネル
[0]	1チャンネル
[1]	2チャンネル
[127]	128チャンネル

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)

2) 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、全チャンネル、ゼロを指定して下さい。

指定したチャン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	設定チャンネル
[0]	1チャンネル
[1]	2チャンネル
[127]	128チャンネル

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)

3) 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、全チャンネル、ゼロを指定して下さい。
指定したチャン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	設定チャンネル
[0]	1チャンネル
[1]	2チャンネル
[127]	1 2 8チャンネル

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

4) input[128]

指定したチャンネルの入力モード(AC/DC)を設定します。

配列	指定チャンネル
[0]	1チャンネル
[1]	2チャンネル
[127]	1 2 8チャンネル

設定値	入力モード
0	DC.Coupling
1	AC.Coupling

5) imode[128]

指定したチャンネルの入力モード(SIGNAL/GND)を設定します。

配列	指定チャンネル
[0]	1チャンネル
[1]	2チャンネル
[127]	1 2 8チャンネル

設定値	入力モード
0	シグナル
1	GND

6) pathen[128]

予約

7) cutoff

全チャン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	
1 0	20kHz	
1 1	25kHz	
1 2	-----	
1 3	-----	
1 4	Through	
1 5	EXT.Mode	(オプション)

8) calsel

予約

9) revolution

予約

1 0) pulse

予約

1 1) change

予約

1 2) average

予約

1 3) cmplevel

予約

1 4) teibai

予約

フィルタ設定アーギュメントの説明 (DAS-16AF-A アーギュメント)

1) cutoff[16]

8チャンネル単位にカットオフ周波数を設定します。

配列	指定チャンネル
[0]	1～8チャンネル
[1]	9～16チャンネル
[15]	121～128チャンネル

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	Through
15	EXT.Mode (オプション)

3.1.14 inet_io_info2

機能 : DASBOXのバージョン情報を得ます。

形式 :

```
int inet_io_info2(cblock, statptr, size);
```

```
CBLK *cblock;
```

```
DASINFO *statptr;
```

```
int size
```

引数 :

cblock コントロールブロックのポインタ

statptr バージョンデータのポインタ

size 読み込む情報量 (34word)
通常はsizeof(DASINFO)/2で指定してください。

戻り値 :

0 正常終了

-1 異常終了

```
DASINFO {  
int    versionA;  
int    versionH;  
int    versionF;  
int    versionD;  
int    versionL;  
int    fatal;  
int    runLevel;  
int    status;  
int    type;  
int    hfifo;  
int    sfifo;  
int    wide;  
int    adChannel;  
int    daChannel;  
int    boardNum;  
int    adNum;  
int    daNum;
```

}

1) **versionA**

DASBOXの内部FPGAバージョンの値が16進数 (HEX) で入ります。

2) **versionH**

DASBOXのハードウェアバージョンの値が16進数 (HEX) で入ります。

3) **versionF**

DASBOXのファームウェアバージョンの値が16進数 (HEX) で入ります。

4) **versionD**

DASBOXのファームウェアバージョンの値が16進数 (HEX) で入ります。

5) **versionL**

DASBOXの内部ライブラリバージョンの値が16進数 (HEX) で入ります。

6) **fatal~daNum**

未使用 (拡張用)

3.2 AD（入力）専用

3.2.1 inet_io_adstart

機能：DASBOXのAD（入力）動作（計測）を開始させます。

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
----	------

3.2.2 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のポインタ

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（16ビットデータ）数
基本的には、データ転送の効率を上げるために、下記の
設定にしてください。

inet_io_blkf()で設定したsize×計測チャンネル数
ただし、データ転送の効率の問題ですので、必ず一致させる
必要はありません。途中及び最後等に上記と異なるサイズで
転送を行っても問題ありません。

戻り値：

0 正常終了

0以外 異常終了

詳細：この関数の第3パラメータ count は一回に buffer に読み込むデータ数を設定します。

設定する値として0以外の値で blkf に関係なく自由な値が設定できます。ただしこの関数を一回以上実行した時の count の合計数は frame1 で設定した総データ量と一致させて下さい。

例えば総データ量が 4096 の場合、count=1000 で4回実行して最後に count=96 実行して終了します。

実際には DASBOX とのデータ転送は blkf に合わせて内部でキューイングして行っています。

例えば総データ量が 4096(2ch,frame1=2048)で blkf=1000 の場合

DASBOX から 2000 データ Read

```
inet_io_adread(,1000)  
inet_io_adread(,1000)
```

```
inet_io_adread(,1000)      DASBOX から 2000 データ Read
inet_io_adread(,1000)
inet_io_adread(,96)       DASBOX から 96 データ Read
```

という手順でデータ転送を行います。
尚 frame1=0 の場合は最後の DASBOX からの Read は 2000 データ
Read して inet_io_adread(,96) で 96 データ buffer に渡します。

3.2.3 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;
```

```
short *buffer;
```

```
int size;
```

```
char *file_name;
```

```
unsigned int frame;
```

引数 :

cblock	コントロールブロックのポインタ
buffer	テンポラリバッファのポインタ
size	テンポラリバッファのサイズ (word)
file_name	書き込むファイル名 (キャラクタのポインタ)
frame	トータル書き込み数 (word)

戻り値 :

0	正常終了
-1	異常終了

3.2.4 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、トータルチャンネルにて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。プリトリガモード計測は、プリトリガサイズに満たないでトリガが入った場合、不足分のデータを無効データとします。また、データ転送は無効データも送られてきます。

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

3.3 DA（出力）専用

3.3.1 inet_io_dastart

機能：DASBOXのDA（出力）動作（計測）を開始させます。但し、DAの場合はDASBOXアーギュメントのframe3（オートスタートカウンタ）分のデータが終了した時点で、出力を開始します。

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
----	------

3.3.2 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 DA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（16ビットデータ）数
基本的には、データ転送の効率を上げるために、下記の
設定にしてください。

inet_io_blkf()で設定したsize×計測チャンネル数
ただし、データ転送の効率の問題ですので、必ず一致させる
必要はありません。途中及び最後等に上記と異なるサイズで
転送を行っても問題ありません。

戻り値：

0 正常終了

-1 異常終了

詳細：この関数の第3パラメータ count は一回に buffer から書き込むデータ数を設定します。

設定する値として0以外の値で blkf に関係なく自由な値が設定できます。ただしこの関数を一回以上実行した時の count の合計数は frame1 で設定した総データ量と一致させて下さい。

例えば総データ量が 4096 の場合、count=1000 で4回実行して最後にcount=96実行して終了します。

実際には DASBOX とのデータ転送は blkf に合わせて内部でキューイングして行っています。

例えば総データ量が 4096(2ch,frame1=2048)で blkf=1000 の場合

```
inet_io_dawrite(,1000)
```

```
inet_io_dawrite(,1000)
```

DASBOX に 2000 データ Write

```
inet_io_dawrite(,1000)
```

inet_io_dawrite(,1000)

DASBOX に 2000 データ Write

inet_io_dawrite(,96)

DASBOX に 96 データ Write

という手順でデータ転送を行います。
尚最後に 96 データ転送するためには総データ量がわからないと判断できないため frame1=0 の場合は最後の 96 データは転送しません。
frame1=0 の場合は blkf の倍数になるようにして下さい。

3.3.3 inet_io_dafile

機能：DASBOXへのDAデータをファイルから読み込み、DASBOXに転送します。

形式：

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

```
CBLK *cblock;
```

```
short *buffer;
```

```
int size;
```

```
char *file_name;
```

```
unsigned int frame;
```

引数：

cblock コントロールブロックのポインタ

buffer テンポラリバッファのポインタ

size テンポラリバッファのサイズ (word)

file_name 読み込むファイル名 (キャラクタのポインタ)

frame トータル読み込み数 (word)

戻り値：

0 正常終了

-1 異常終了

3.3.4 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,time);
```

```
CBLK *cblock;
```

```
int time;
```

引数：

CBLK コントロールブロック

time タイムアウト値 (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

3.3.5 inet_io_cycle_stop

機能 : DAサイクルモード動作以外の場合は使用しないでください。

DASBOX内部のストップフラグを立てて、次のフレームで繰り返し動作を終了させます。サイクル動作は、フレームの区切り目で正常に終了します。

形式 :

```
int inet_io_cycle_stop(cblock);
```

```
CBLK *cblock;
```

引数 :

```
CBLK          コントロールブロックのポインタ
```

戻り値 :

```
0              正常終了
```

```
-1             異常終了
```


3.3.6 inet_io_daclear

機能：動作（計測）停止状態のDASBOXのDA出力を0V出力状態にします。

形式：

```
int inet_io_daclear(cblock);
```

```
CBLK *cblock;
```

引数：

CBLK コントロールブロックのポインタ

戻り値：

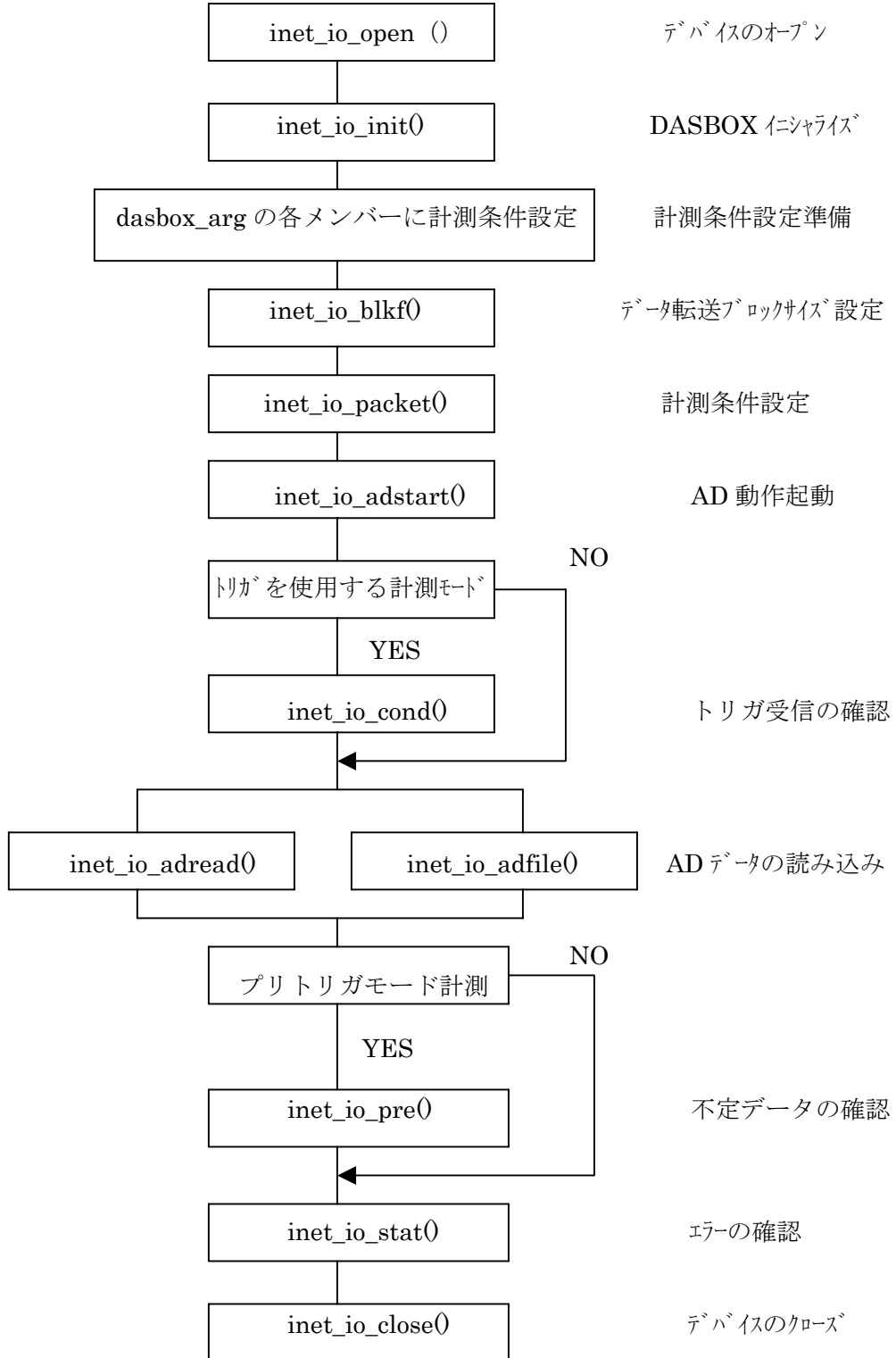
0 正常終了

-1 異常終了

3.4 サブルーチン実行基本フロー

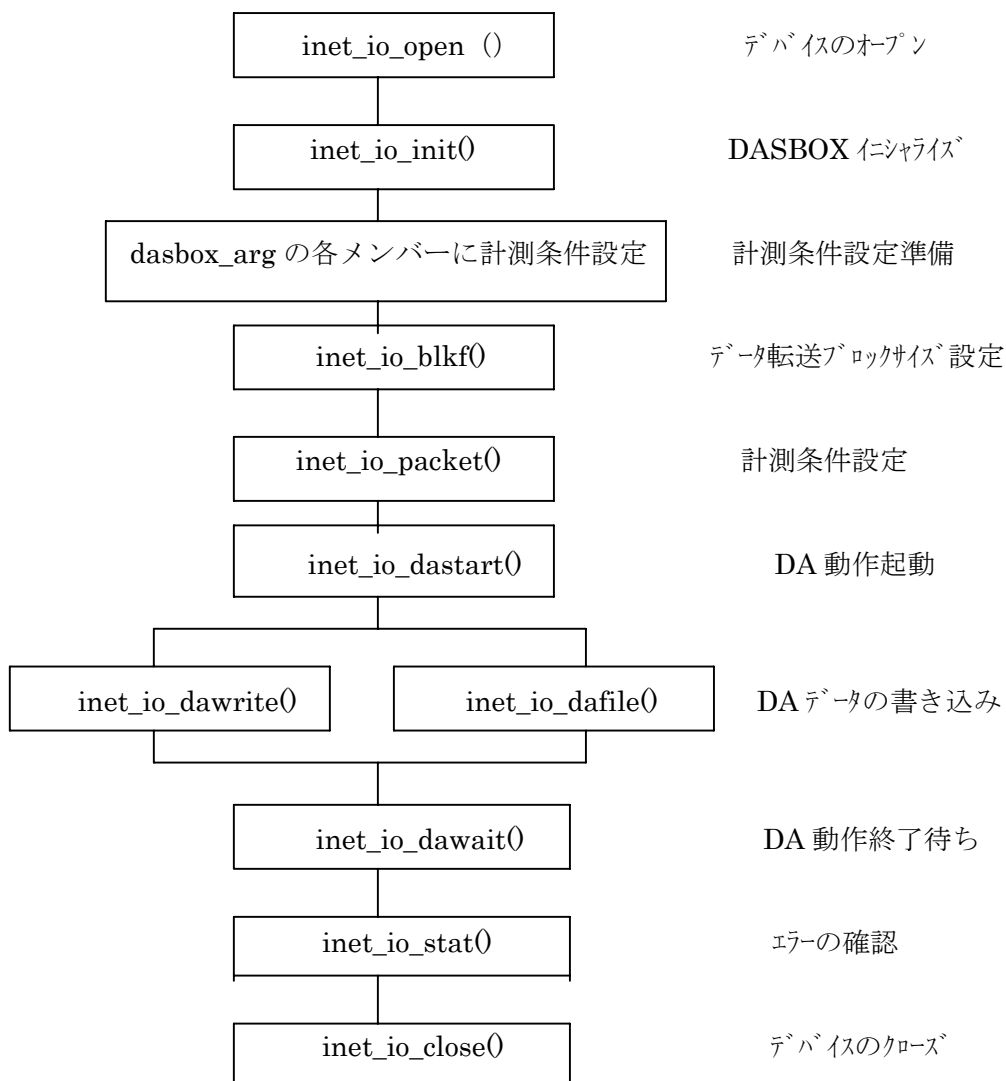
3.4.1 AD動作

<内容>



3.4.2 DA 動作

<内容>



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 PCISAD16_arg
{
    unsigned short mode;
    unsigned short stat;
    unsigned short dastime;
    unsigned short attn;
    unsigned short gain;
    unsigned short trgslp;
    unsigned short clkmode;
    int fifo;
    float clock;
    int frame1;
    int frame2;
    int frame3;
    int frame4;
    int frame5;
    unsigned int mutelevel;
    unsigned short trglevel;
    unsigned short trgsrc;
    unsigned short trgch;
    unsigned short channels;
    unsigned short chanum[1024];
    unsigned short master;
    unsigned short ext_parm;
    int dmasize;
    int pre_mem_size;
}PCISAD_ARG ;
```

4-1 mode

DASBOXに対する動作を指定します。

Define名	値 (DEC)	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
DARET	6	DAリトリガモード
DACYCL	7	DAサイクルモード
DATCYC	8	DAトリガサイクルモード
DARCYC	9	DAリトリガサイクルモード
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 (HEX)	ステータス
0	正常
8000	SDS_PCI_FIFO_OVERFLOW
8001	SDS_PCI_OVER_SAMPLING
8002	SDS_PCI_ADDA_ABORT
8003	SDS_PCI_TIMEOUT

4-3 dmatime

inet_io_adread,inet_io_dawrite,inet_io_adfile,inet_io_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（30）になります。 但し、基本サブルーチンのVer2.03からの対応となります。 Ver2.01,Ver2.02の場合は”0”がそのまま使用され、inet_io_adread、inet_io_adfile、inet_io_dawrite、inet_io_dafile関数にて、異常終了となります。
1～n	秒単位になります。

4-4 attn（未使用）

常に0を設定してください。（不定でも問題なし）

4-5 gain（未使用）

常に0を設定してください。（不定でも問題なし）

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。

Define名	値	動作
INTCLK	1	内部クロックを使用
EXTCLK	2	外部からクロックを入力する
EXTDIV	3	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有効で、内部クロックを指定した場合は周波数をHz単位で指定し、外部分周を指定した場合は分周値を設定します。また、内部クロックを指定した場合は、実際にサンプリング周波数として設定された値が戻されます。サンプリングクロックは内部クロックベース（8.000MHz,8.192MHz,6.144MHz、5.6448MHz）を分周して作成します。分周の余りがでる場合は近似値が自動的に設定されますので、その設定した周波数値を返します。

<内部クロック>

設定範囲 0. 0 3 Hz～最大サンプリング周波数

<外部分周>

設定範囲 2～6 5 5 3 6

4-9 frame1

AD又はDAする1チャンネルに対するデータ量を指定します。単位はワードです。但し、“0”を設定すると、無限取り込みモードとなります。また、プリトリガモードの場合はframe2の値も含んで設定してください。

設定範囲 0～4 2 9 4 9 6 7 2 9 5

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。
<プリトリガの時>

プリトリガ動作の時は1チャンネルに対する、トリガ以前のデータ量を指定します。単位はワードです。

設定範囲 0 < 設定値 < FIFOサイズ/チャンネル数-100

注) FIFOサイズ全てはプリトリガサイズに指定できません。

トリガ受信時のDASBOX内部セットアップ時間が必要ですので、チャンネルあたり、100データ使用できないと考えてください。

1チャンネルに対するトリガ以後の計測データ数 = frame1 - frame2
の関係になります。

<ポストトリガの時>

ポストトリガ動作の時は遅延サンプルクロック数を指定します。
遅延時間はサンプルクロックの指定により、変ります。

“1”の場合1サンプル遅延

設定範囲 0 < 設定値 < 1 6 7 7 7 7 2 1 6

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1チャンネルに対するデータ量を指定します。単位はワードです。

設定範囲 0 < 設定値 = < FIFOサイズ/チャンネル数

4-12 frame4

DA サイクル動作を指定した場合有効で繰り返す回数を指定します。
その他のモードは常に“1”を設定してください。

設定範囲 1～6 5 5 3 6

4-13 frame5

modeがADRETRG及びADREPOSTのときトリガ回数を指定します。

設定範囲 1 ~ 6 5 5 3 6

4-14 mutelevel

未使用。

4-15 trglevel

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。

設定範囲 0 ~ 1 2 7 (DEC)

0 ~ 7 F (HEX)

値	電圧	± 5 V	± 1 0 V
HEX DEC			
0x7F 127	+FS(63/64)v	+4.92V	+9.84V
0x7E 126	+FS(62/64)v	+4.84V	+9.68V
0x41 65	+FS(1/64)v	+0.08V	+0.16V
0x40 64	0v	0V	0V
0x3F 63	-FS(1/64)v	-0.08V	-0.16V
0x01 1	-FS(63/64)v	-4.92V	-9.84V
0x00 0	-FS(64/64)v	-5V	-10V

*) 1LSB=2FS/128 入力電圧± 5 Vの時は 0.078125V

入力電圧± 1 0 Vの時は 0.15625V

4-16 trgsrc

トリガを使用するモードを指定した場合有効となり、トリガのソースを指定します。

値	動作
0	AD入力チャンネルトリガ
1	外部トリガ端子

4-17 trgch

trgsrcにて、AD入力チャンネルトリガを指定した場合有効となり、トリガとするチャンネル番号を指定します。

値	動作
1~1024	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 1～1024

4-19 chanum 【1024】

ランダムチャンネルの指定を行います。AD 又は DA するチャンネル番号を指定します。指定した順番でデータ転送が行われます。

範囲 1～1024

chanum[0]	1 番目のチャンネル番号
chanum[1]	2 番目のチャンネル番号
⋮	⋮
chanum[1023]	1024 番目のチャンネル番号

4-20 master,ext_parm,dmasize,pre_mem_size

未使用

DASBOX-E シリーズでは、このメンバーは使用しません。

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/floppy/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(LINUX)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-ModelEシリーズ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dasです。

```
構文    das 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。
INFO	DASBOXの内部情報を表示します。
INF 2	メンテナンスにて使用します。
COND	DASBOXの動作状態を表示します。
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。
ADRET	ADリ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。
ADPOst	ADポストトリガスタートモードを起動します。
ADREPost	ADリポスト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。

FILset	フィルタの設定をブロック単位で行います。
AMPset	アンプの設定を行います。
OFILset	フィルタの設定を全チャンネル共通で行います。

小文字の部分は省略可能です。また、入力は大文字、小文字いずれも対応します。

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。

パラメータファイルの作成

指定したパラメータがない場合パラメータファイルを作成する為に入力を促してきます。

次に各入力について述べます。

- ① AD 及び DA 共に出力されます。

[1] Use internal clock
[2] Use external clock
[3] Use prescale external clock

Select no.

計測するクロックソースを内部,外部又はプリスケール (分周) 外部クロックにするか指定します。

内部の場合 1,外部の場合 2,プリスケール (分周) 外部クロックの場合 3 を入力します。

- ②-1 内部クロックを指定した場合出力されます。

[1] Hz unit
[2] KHz unit
[3] uSEC unit

Select no.

計測するクロックの周波数をどの単位で入力するか指定します。Hz で入力したい場合 1,KHz で入力したい場合 2, μ SEC で入力したい場合 3

Input(Hz/KHz/ μ sec)rate

と出力されたら計測したい値を入力してリターンキーを押します。

- ②-2 プリスケール外部クロックを指定した場合、出力されます。

Input prescale counter

プリスケール値 (分周値) を入力します。

- ③ トリガを使用するモードを指定した場合出力されます。

トリガソースを指定します。

[1] Use external trigger
[2] Use channel trigger

Select no.

TRIG IN 端子を使用する場合は 1、入力チャンネルを使用する場合は 2 を入力します。但し、AD モジュールがない場合は必ず 1 を設定してください。

- ③-1 ③にて 2 (チャンネルトリガ) を指定した場合に出力されます。

Select channel no. [1-1024]

トリガに使用する AD チャンネル番号を指定します。(指定 AD チャンネルは他のチャンネルと同様に計測にも使用できます)

- ④ トリガモードを指定した場合出力されます。

[1]Use positive edge trigger
[2]Use negative edge trigger

Select no.

トリガの立ち上がり,立ち下がりの指定をします。立ち上がりでトリガする場合 1、立ち下がりでトリガする場合 2 を設定します。

次に、トリガレベルを聞きますので、0~127の値を設定します。
(下記の表を参照)

Set trigger level.[0-127]

設定値	電圧	± 5 V	± 1 0 V
DEC			
127	+FS(63/64)v	+4.92V	+9.84V
126	+FS(62/64)v	+4.84V	+9.68V
65	+FS(1/64)v	+0.08V	+0.16V
64	0v	0V	0V
63	-FS(1/64)v	-0.08V	-0.16V
1	-FS(63/64)v	-4.92V	-9.84V
0	-FS(64/64)v	-5V	-10V

*) 1LSB=2FS/128 入力電圧± 5 Vの時は 0.078125V
入力電圧± 1 0 Vの時は 0.15625V

- ⑤-1 AD リポストトリガモードを指定した場合出力されます。

Input posttrg size

トリガから、遅延するクロック数を設定します。サンプリングクロック設定に関連します。

Input re-posttrg count

くり返し回数を設定します。

- ⑤-2 AD ポストトリガモードを指定した場合出力されます。

Input posttrg size

トリガから、遅延するクロック数を設定します。サンプリングクロック設定に関連します。

⑤-3 AD プリトリガモードを指定した場合出力されます。

Input pretrg size

トリガ以前にほしい1チャンネルに対するデータ量を指定します。

⑤-4 AD リトリガモードを指定した場合出力されます。

Input re-trg count

くり返し回数を設定します。

⑤-5 DA モードを指定した場合出力されます。

Input auto start size

DA を起動する前に転送する1チャンネルに対するデータ量を指定します。

⑥ AD 及び DA 共に出力されます。

Input frame size

AD 及び DA する1チャンネルに対するデータ量を指定します。

⑦ AD 及び DA 共に出力されます。

Normal Random channel NO. ? Yes=0/NO=1

ランダムチャンネル設定を標準（1チャンネルから順番）で自動設定するか、1チャンネルずつ手動で入力するかを選択します。標準の場合は0、手動設定の場合は1

手動設定を選択した場合は、チャンネル数分、1番目から順に設定チャンネル番号を聞いてきます。

Input *th channel number

以上の入力を終了すると、計測を開始します。