DASmini E500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5月６日
改訂　平成１５年7月２５日
改訂　平成１５年9月１７日
改訂　平成１６年5月２４日
改訂　平成１６年7月２１日
改訂　平成１９年4月２５日
改訂　平成１９年5月１６日
改訂　平成１９年8月２２日
改訂　平成１９年10月３１日
改訂　平成２４年4月１０日

ケイテクノス株式会社

目　次

改訂履歴

・ 平成 15 年 5 月 6 日 初版作成

・ 平成 15 年 7 月 25 日 改訂

・ 平成 15 年 9 月 17 日 inet_io_blkf（）関数説明を修正

・ 平成 16 年 5 月 24 日 サブルーチン実行基本フローの inet_io_packet()->inet_io_blkf()

の順番を inet_io_blkf()->inet_io_packet()に変更

・ 平成 16 年 7 月 21 日　frame1 の設定範囲を訂正
０～4294967295　－＞　０～ 268435455 に訂正
frame2 の設定範囲を訂正
＜プリトリガ時＞
訂正前　０ ＜ 設定値 ＜　FIFOサイズ／チャンネル数－100
訂正後　０ ＜ 設定値 ＜　（FIFOサイズ／２）／チャンネル数－100
＜ポストトリガ時＞
訂正前　０ ＜ 設定値 ＜ １６７７７７２１６
訂正後　０ ＜ 設定値 ＜ １６７７７２１６

・ 平成19年4月25日　frame1及びframe5の説明を変更

・ 平成19年5月16日　コンパイルオプションBIGによりHP-UX等のBIG_Endianマシンに対応。

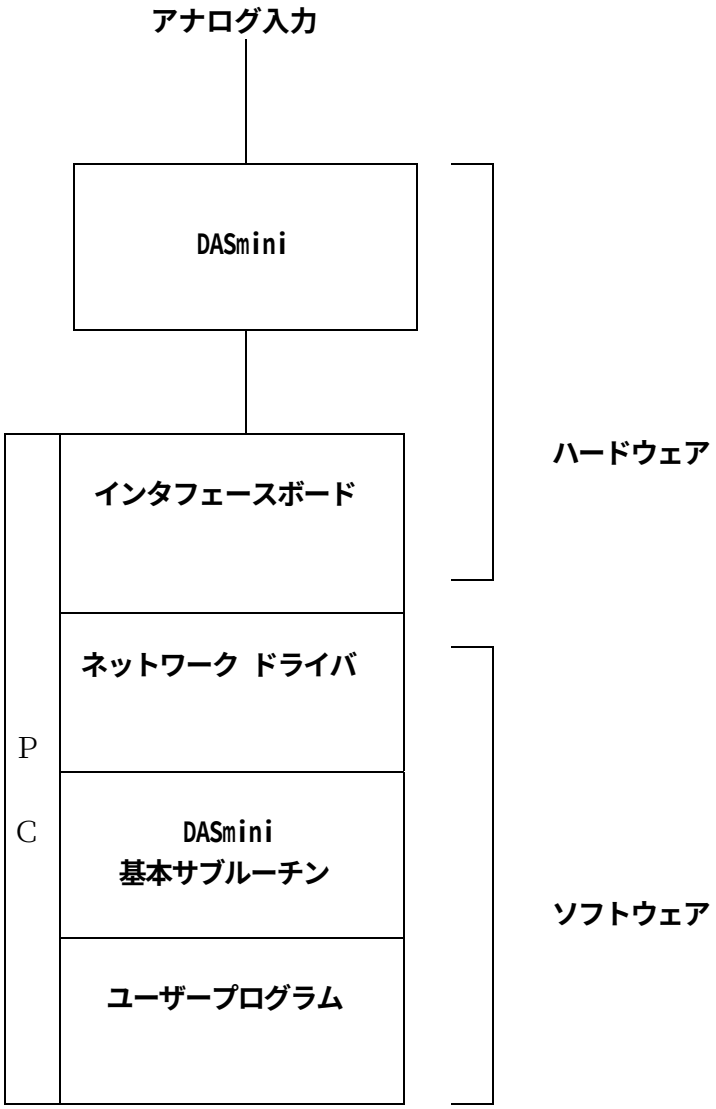
・ 平成19年8月22日　サンプルソフトdaswave_2追加による取り扱い説明、コンパイル時のワーニング修正。

・ 平成19年10月31日　ライブラリーバージョンアップ2.04－＞2.05にて、プリトリガ動作時のプレマスタ、プレスレーブ指定追加。
複数台を同期して計測する場合の標準ライブラリーinet_mio_***を追加。

・ 平成 23 年 4 月 10 日
１． 32bit Windows 用 DLL 及び 64bit Windows 用 DLL 作成
２． C#言語のサンプル追加及び C#関数説明追加
３． Visual Studio2008 対応

第1章 概　要

本サブルーチンは、DASmini E500シリーズを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、DASmini 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。
基本サブルーチンはLinux/Unix/Windows共通Cソースファイル(*.c,*.h)で提供しています。
Windows用には別に32bit/64bit版DLLとC#クラスライブラリを提供しています。



第２章 DASmini 関数の説明

２．１　単体（1 台）で計測する場合

・inet_io_open()

DASminiと接続されているデバイスをオープンします。

・inet_io_close()

DASminiと接続されているデバイスをクローズします。

・inet_io_packet()

DASminiに対して動作を指示します。

・inet_io_cond()

DASminiの状態（トリガ等）を取得します。

・inet_io_stat()

DASminiがAD動作中にエラーが発生した時、DASminiのエラーステータスを読み取ります。

・inet_io_stop()

DASmini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()

DASmini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()

DASminiのインフォメーションを得ます。

・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_io_adstart()

DASminiに対してAD（入力）動作を開始させます。

・inet_io_adread()

DASminiからのADデータをメモリ上に読み込みます。

・inet_io_adfile()

DASminiからのADデータをファイルに書き込みます。

・inet_io_pre()

プリトリガが動作を行なった時、計測チャンネル数に対しての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。

・inet_io_pre2()

プリトリガが動作を行なった時、１チャンネルに対してのトリガ以前何データが無効データか、及びトリガ位置のずれデータ数をホストに知らせます。

２．２　複数台で同期計測する場合

・inet_mio_open()

指定数のデバイスをオープンします。

・inet_mio_close()

指定数のデバイスをクローズします。

・inet_mio_packet()

複数台のDASminiに対して動作を指示します。

・inet_mio_cond()

マスタDASminiの状態（トリガ等）を取得します。

・inet_mio_stat()

マスタDASminiがAD動作中にエラーが発生した時、DASminiのエラーステータスを読み取ります。

・inet_mio_stop()

複数台のDASmini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_mio_init()

複数台のDASmini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_mio_info()

マスタDASminiのインフォメーションを得ます。

・inet_mio_blkf()

複数台のデータ転送のブロックサイズを設定します。

・inet_mio_adstart()

複数台のDASminiに対してAD（入力）動作を開始させます。

・inet_mio_adread()

複数台のDASminiからのADデータをメモリ上に読み込みます。
チャンネルシリアルのデータフォーマットでメモリーに読み込みます。

・inet_mio_adread2()

複数台のDASminiからのADデータをメモリ上に読み込みます。
チャンネルパラレルのデータフォーマットでメモリーに読み込みます。

・inet_mio_adfile()

複数台DASminiからのADデータをファイルに書き込みます。

・inet_mio_pre()

複数台でプリトリガが動作を行なった時、計測チャンネル数に対しての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。

・inet_mio_pre2()

複数台でプリトリガが動作を行なった時、１チャンネルに対してのトリガ以前何データが無効データか、及びトリガ位置のずれデータ数をホストに知らせます。

２．３　複数台で同期計測する場合の制限

１）ADPRE（プリトリガモード）で計測する場合は、１６チャンネル単位のチャンネル設定とし、ランダムチャンネル指定は禁止（１チャンネルから順番に指定）してください。この制限はプリトリガのトリガポイントを同期させる為に必要となります。

２）計測と途中停止及びデータを全て読み込まないで終了する場合は必ず、inet_mio_init（）関数を実行してください。

第３章 DASmini 関数の使用方法

・コントロールブロック

DASmini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DASmini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。特にユーザがコントロールブロックに設定する必要はありません。

・DASminiアーギュメント

DASminiに対して動作を指示する構造体でinet_io_packet(),inet_mio_packet()関数等で使用します。

内容は第４章で説明します。

C関数ではコントロールブロック及びDASmini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。

C#関数ではコントロールブロック及びDASminiアーギュメントは、dasbox.csファイルにその他の定義と共にDasboxクラスとして納められています。ユーザーは、本関数を使用する場合dasbox.csファイルをプロジェクトに追加して使用します。
また本関数で使用する構造体の作成はnewで作成するのではなくNewメソッドを使用してください。

```
Dasbox.CBLK  cblk  =  Dasbox.CBLK.New();
```

3.1 inet_io_open

機能：DASmini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。

C形式：

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

CBLK *cblock;
PCISAD_ARG *dasarg;
char *ip_address;
```

C#形式：

```
int Dasbox.inet_io_open(ref cblock, ref dasarg, ip_address);

Dasbox.CBLK cblock = Dasbox.CBLK.New();
Dasbox.PCISAD_ARG dasarg = Dasbox.PCISAD_ARG.New();
String ip_address;
```

引数：

cblock	コントロールブロックのポインタ
dasarg	DASmini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

3.2 inet_io_close

機能：DASminiと接続されているデバイスをクローズします。

C形式：

```
int  inet_io_close(cblock);

CBLK  *cblock;
```

C#形式：

```
int  Dasbox.inet_io_close(ref  cblock);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
```

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

3.3 inet_io_packet

機能：DASmini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DAS miniアーギュメント説明を参照願います。

C形式：

```
int  inet_io_packet(cblock,  dasarg);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
```

C#形式：

```
int  Dasbox.inet_io_packet(ref  cblock,  ref  dasarg);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.PCISAD_ARG  dasarg  =  Dasbox.CBLK.New();
```

引数：

cblock	コントロールブロックのポインタ
dasarg	DASmini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[256]の設定エラー
-7,-8	パケットコマンドエラー

3.4 inet_io_cond

機能： DASmini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動作が完了するまでの期間となります。

C形式：

```
int inet_io_cond(cblock,  data);

CBLK *cblock;
int *data;
```

C#形式：

```
int Dasbox.inet_io_cond(ref cblock, data);

Dasbox.CBLK cblock = Dasbox.CBLK.New();
int[] data = new int[1];
```

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
DASmini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	3 ~0
*	*	*	*	*	NEM	*	*	*	EN	*	TRG	*	*

＊ 印のビットは不定データとなります。

Bit11: NEM
DASminiのFifoが空でない時に“１”となります。

Bit7: EN
DASmini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

3.5 inet_io_stat

機能：DASmini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()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。

C形式：

```
int  inet_io_stat(cblock,  dasarg);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
```

C#形式：

```
int  Dasbox.inet_io_stat(ref  cblock,  ref  dasarg);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.PCISAD_ARG  dasarg  =  Dasbox.PCISAD_ARG.New();
```

引数：

cblock	コントロールブロックのポインタ
dasarg	dasmini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

3.6 inet_io_stop

機能：DASmini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、DASmini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。また、この関数を実行する時はinet_io_blkf関数で指定した

C形式：

```
int  inet_io_stop(cblock,  dasarg);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
```

C#形式：

```
int  Dasbox.inet_io_stop(ref  cblock,  ref  dasarg);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.PCISAD_ARG  dasarg  =  Dasbox.PCISAD_ARG.New();
```

引数：

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

戻り値：

0	正常終了
-1	異常終了

3.7 inet_io_init

機能：DASmini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DASmini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。

C形式：

```
int  inet_io_init(cblock);

CBLK  *cblock;
```

C#形式：

```
int  Dasbox.inet_io_init(ref  cblock);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
```

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

3.8 inet_io_info

機能：DASminiの内部情報を得ます。

C形式：

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

CBLK  *cblock;
PCISAD_INFO  *statptr;
int  size
```

C#形式：

```
int  Dasbox.inet_io_info(ref  cblock,  ref  statptr,  size);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.PCISAD_INFO  statptr  =  Dasbox.PCISAD_INFO.New();
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

DASminiに実装されているFIFOのワードサイズが入ります。。

標準は３３５５４４３２

2） fifo_wide

DASmini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

DASminiの最大ADチャネル数が入ります。

標準は １６

4） output_channel

DASminiの最大DAチャネル数が入ります。

標準は ０

5） board_id

常に ０

6） module_input

未使用（拡張用）

7） module_output

未使用（拡張用）

3.9 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。

C形式：

```
char  *inet_io_error(cblock);

CBLK  *cblock;
```

C#形式：

```
StringBuilder  Dasbox.inet_io_error(ref  cblock);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
```

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

3.10 inet_io_blkf

機能： １チャンネルに対する1回のinet_io_adread()のデータの転送数 （ブロック数）を指定します。inet_io_adstart()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()から有効となります。

C形式：

```
int  inet_io_blkf(cblock,  size)

CBLK  *cblock;
int  size;
```

C#形式：

```
int  Dasbox.inet_io_blkf(ref  cblock,  size)

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
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～DASminiﾒﾓﾘｰｻｲｽﾞ／２／計測ﾁｬﾝﾈﾙ数 標準： DASminiﾒﾓﾘｰｻｲｽﾞ = 33554432（32MW） オプション： DASminiﾒﾓﾘｰｻｲｽﾞ = 67108864（64MW） 設定制限 ： ﾄﾗﾝｼﾞｪﾝﾄ計測（AD終了後、データを取り込む）で、計測を行う場合は、size×計測ﾁｬﾝﾈﾙ数が８の倍数である必要があります。

戻り値：

0	正常終了
-1	異常終了

補足： DASmini内部には、ハードウェアの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の様な転送となります。

例１ ｻﾝﾌﾟﾙｸﾛｯｸ= １Hz、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　ｻﾝﾌﾟﾙｸﾛｯｸ= 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。

3.11 inet_io_adstart

機能：DASminiのAD（入力）動作（計測）を開始させます。

C形式：

```
int  inet_io_adstart(cblock);

CBLK  *cblock;
```

C#形式：

```
int  Dasbox.inet_io_adstart(ref  cblock);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
```

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

3.12 inet_io_adread

機能：DASminiからのADデータをメモリ上に読み込みます。

C形式：

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

CBLK *cblock;
short *buffer;
int count;
```

C#形式：

```
int Dasbox.inet_io_adread(ref cblock, buffer, count);

Dasbox.CBLK cblock = Dasbox.CBLK.New();
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、総データ量（frame1×channels）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 DASmini 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

DASmini から 2000 データ Read
inet_io_adread(,,1000)
inet_io_adread(,,1000)
DASmini から 2000 データ Read
inet_io_adread(,,1000)
inet_io_adread(,,1000)
DASmini から 96 データ Read
inet_io_adread(,,96)

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 DASmini 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。

3.13 inet_io_adfile

機能：DASminiからADデータをファイルに書き込みます。

C形式：

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

CBLK *cblock;
short *buffer;
int size;
char *file_name;
unsigned int  frame;
```

C#形式：

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

Dasbox.CBLK cblock = Dasbox.CBLK.New();
short[,] buffer;
int size;
String file_name;
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

3.14 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。DASminiはinet_io_pre2（）関数を使用してください。

C形式：

```
int  inet_io_pre(cblock,  predata);

CBLK  *cblock;
int  *predata;
```

C#形式：

```
int  Dasbox.inet_io_pre(ref  cblock,  predata);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
int[]  predata  =  new  int[1];
```

引数：

cblock	コントロールブロックのポインタ
predata	無効データ数を格納する変数のポインタ

戻り値：

0	正常終了
-1	異常終了

3.15 inet_io_pre2

機能：

プリトリガAD動作完了後、１チャンネルに対する、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。

C形式：

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

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

C#形式：

```
int  Dasbox.inet_io_pre(ref  cblock,  predata,adjust);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
int[]  predata  =  new  int[1];
int[]  adjust  =  new  int[1];
```

引数：

cblock	コントロールブロックのポインタ
predata	無効データ数を格納する変数のポインタ
adjust	ずれデータ数を格納する変数のポインタ

戻り値：

0	正常終了
-1	異常終了

DASmini のプリトリガーAD 変換の動作はパケットで指定された presize に満たない場合でも、トリガーを検出すると有効と見なします。inet_io_adread()関数では presize 前からのデータを返します。この無効のデータ数を知るための情報です。
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（１チャンネル分）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。

また、書き込みタイミングとサンプリング周波数の関係でトリガポイントがずれる場合があります。adjustはずれたデータ量（１チャンネル分）を示します。
0=ずれ無し。n=n データ分トリガー位置が前方にずれた。presize が n 分少なくなった事を示します。よって、実際のトリガポイントのデータは設定値のポイント（presize）より、adjust 分前にある事となります。

3.16 inet_mio_open

機能：複数台のデバイスをオープンします。
他のinet_mio_**()関数は、本関数呼び出し後動作可能になります。

形式：

```
int  inet_mio_open(cblock,  dasarg,  ip_address,block_channels,das_number);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
char      *ip_address;
int       block_channels;
int       das_number;
```

引数：

cblock	コントロールブロックのポインタ 台数分のコントロールブロックを確保してください。
dasarg	DASmini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。 台数分の［128］の配列を確保してください。
block_channels	DASmini-E500の１台に実装されているADチャンネル数を設定します。標準は１６、この設定が複数台の１台のチャンネル単位となります。
das_number	計測で使用するDASmini-E500の台数

実行例： 1台のチャンネル単位が16CHで、台数が4台の場合

```
 #define  BLOCK_CHANNELS    (16)
 #define  DAS_NUMBER    (4)
static  char        ip_address[][128]  =  {"192.168.0.2",
                                           "192.168.0.3",
                                           "192.168.0.4",
                                           "192.168.0.5"
 };
 CBLK                    block[DAS_NUMBER];
 PCISAD_ARG              dasarg;
Retval=inet_mio_open(&block[0],  dasarg,  &ip_address[0][0],BLOCK_CHANNELS,
                          DAS_NUMBER);
```

戻り値：

0	正常終了
-1	異常終了

3.17 inet_mio_adread2

機能：複数台のDASminiからのADデータをメモリ上に読み込みます。データは
チャンネルパラレルのデータで読み込まれます。

形式：

```
int  inet_io_adread2(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：

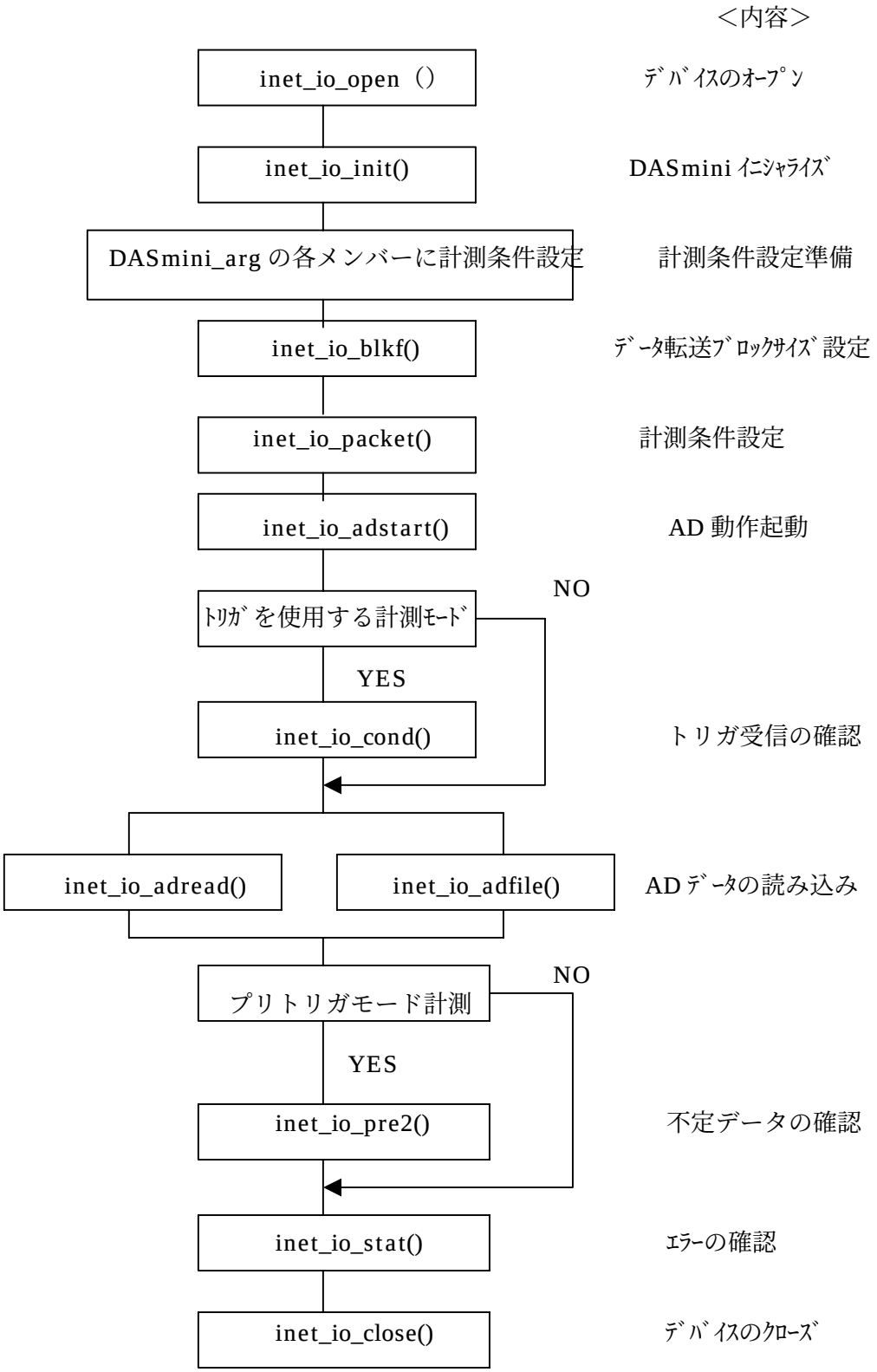
0	正常終了
0以外	異常終了

詳細：チャンネルパラレルデータのフォーマット
例　計測チャンネル数＝３、count=9 の場合
メモリーには以下の順番でデータが置かれます。
ch1-1,ch1-2,ch1-3 ,ch2-1,ch2-2,ch2-3,ch3-1,ch3-2,ch3-3

3.18 その他の inet_mio_＊＊＊

機能：複数台のDASminiを制御しますが、機能としては、inet_io_＊＊＊と
同じ動作をおこないます。

・サブルーチン実行基本フロー



第４章 DASmini アーギュメント説明

DASminiを動作させるには、inet_io_packet, inet_mio_packet,関数を使用して行います。関数の第２パラメータがDASminiの動作内容になっています。Inet_mio_packet関数の場合も、DASminiは１台と考えてアーギュメントを設定をする事ができます。但し、以下の設定はできません.。

channnels： 最大チャンネル数（block_channels×das_number）

chanum[]： ランダム指定（通しのチャンネル番号、但し、若い番号順に指定する事）

例 トータル6チャンネルの時
良い設定 1,2,17,18,19,33等
エラー設定 1,17,2,18,33,19等

```
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[256];
        unsigned  short    master;
        unsigned  short    ext_parm;
        int                dmasize;
        int                pre_mem_size;
        }PCISAD_ARG；
```

4-1　mode

DASmini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
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DASmini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_adfile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）

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）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　　０．５Hz～最大サンプリング周波数
＜外部分周＞
　設定範囲　　２～６５５３６

4-9　frame1

AD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、modeの指定がADRET及びADREPOST以外のモードの場合は“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　　０～２６８４３５４５５

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／２）／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DASmini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 ＜ １６７７７２１６

4-11 frame3

未使用
常に“１”を設定してください。

4-12 frame4

未使用
常に“１”を設定してください。

4-13 frame5

modeがADRET及びADREPOSTのときトリガ回数を指定します。“０”を設定すると、無限取り込みモードとなります。その他のモードの場合は常に“１”を設定してください。

設定範囲　０～６５５３６

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　０～１２７（DEC）
０～７F（HEX）

値		電圧	±５V	±１０V
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±５Vの時は　0.078125V
入力電圧±１０V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功となり、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

1～16　AD入力チャンネル番号

4-18 channels

ADするチャンネル数を指定します。

範囲　1～256

注）ADPREモードで、inet_mio_packet()関数を使用する場合は、16チャンネル単位で設定してください。

4-19 chanum【256】

ランダムチャンネルの指定を行います。AD 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　１～２５６

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[255]	256 番目のチャンネル番号

注）ADPREモードで、inet_mio_packet()関数を使用する場合は、ランダム指定は行わず、１から順番に指定してください。

4-20　ext_parm,dmasize,pre_mem_size

未使用

DASmini－E シリーズでは、このメンバーは使用しません。

4-21 master

プリトリガモードの場合有効となり、トリガ受信及びトリガ出力信号のタイミングを設定します。

注）複数台を同期して使用しない場合は、常にスレーブを指定してください。また、１台で使用する場合もスレーブを指定してください。

値	動作
1	プリトリガマスタとして、動作します。複数台で同期する場合の、マスタ動作（サンプリングクロックを発生する）の時に指定。
0	プリトリガスレーブとして、動作します。複数台で同期する場合の、スレーブ動作（サンプリングクロックをマスタから入力する）の時に指定。又はプリトリガ動作でない場合。

第５章 DASmini 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:\dasmini）を作成して提供メディアから\kit*.*
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DASmini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*.*
    /doc
    /inetlib
    /single
        sample/   samplevc/   daswave_2008/   samplec#/
    /multi
        sample/   samplevc/
```

また各ディレクトリには以下のファイルがあります。

```
Doc/ ・・・  マニュアルディレクトリ
  DASmini_lib*.*.pdf      ・・・基本サブルーチンマニュアル
  DASmini-E500_**.pdf     ・・・ハードウェアマニュアル
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
  C#DLL/・・・C#ファイル
        inetlib.dll     ・・・32bit Windows用DLLファイル
        inetlib64.dll   ・・・64bit Windows用DLLファイル
        Dasbox.cs       ・・・32bit/64bit C#定義ファイル
```

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
Makefile_little ・・・Little_Endianアーキテクチャ(LINUXを含む)でのサンプルメイクファイル
Makefile_big ・・・Big_Endianアーキテクチャでのサンプルメイクファイル
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(シングル動作)
apl90m.c ・・・サンプルソフトソースファイル（複数台で同期動作）
das ・・・サンプルソフト実行ファイル
samplevc/ ・・・Windows Visual Studio6.0 or 2008 コンパイル環境
(コマンドベースのサンプルプログラム)
sample.dsw ・・・Visual Studio6.0 プロジェクトファイル
sample.sln ・・・Visual Studio2008 ソリューションファイル
apl90.c ・・・・サンプルソフトソースファイル（シングル動作）
apl90m.c ・・・サンプルソフトソースファイル（複数台で同期動作）
das.exe ・・・・Windowsサンプルソフト実行ファイル

samplec#/ ・・・(C# Visual Studio 2010 SP1 コンパイル環境)
(コマンドベースのサンプルプログラム 32bit/64bit対応)
sample_ad.sln・・（Visual Studio 2010 ソリューションファイル）
Dasbox.cs ・・(32bit/64bit共通 C#定義ファイル)
Program.cs ・・(サンプルソースファイル)

daswave_2008/ ・・・Windows Visual Studio6.0 or 2008 コンパイル環境
（ウィンドウベースのサンプルプログラム）
daswave.exe ・・・サンプルプログラム実行ファイル
default.cnd ・・・初期設定ファイル
daswave ・・・計測サンプルソフトウェアのソース
testmini.dsw・・・Visual Studio6.0 プロジェクトファイル
testmini.sln ・・・Visual Studio2008 プロジェクトファイル

5-3 ユーザープログラムへの組み込み

DASmini-E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BigエンディアンOS環境の時はBIGコンパイルオプションをつけます。サンプルのMakefile_bigを参照してください。
（HP及びSUNのワークステーション等）
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。
Windows(C#)の場合はCSファイル”Dasbox.cs”をプロジェクトに追加します。
Windows(C#)の32bitアプリを作成する場合はDLLファイルC#DLL/inetlib.dllをユーザープログラムと同じディレクトリに置いてください。Windows(C#)の64bitアプリを作成する場合はDLLファイルC#DLL/inetlib64.dllをユーザープログラムと同じディレクトリに置いてください

＜コンパイルオプションの指定＞

WINNT ： WindowsNT、及びWindows2000、WindowsXPで使用する場合必要です。
サンプルのmk.cmdを参照して下さい。
wsock32.libとuser32.libをリンクして下さい。

BIG ： OS環境がBIG_Endianシステムの場合必要です。
サンプルのMakefile_bigを参照して下さい。

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DASminiに対する動作の指定です。

INIT	DASminiにイニシャライズコマンドを送ります。
STOP	DASminiにストップコマンドを送ります。
STATus	DASminiのステータスを表示します。
INFO	DASminiの内部情報を表示します。
COND	DASmini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。
ADTEst	固定アーギュメントの設定で一連の動作を実行します。

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。

HostnameはDASmini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 必ず出力されます。

[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,μSEC で入力したい場合 3

Input(Hz/KHz/μsec)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、トリガレベルを聞いてきますので、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	±５V	±１０V
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±５Vの時は　0.078125V
入力電圧±１０V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。

⑥　必ず出力されます。

Input　frame　size

１チャンネルに対するデータ量を指定します。

⑦ 必ず出力されます。

Normal Randum 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チャンネルずる手動に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。

5-5　ダイアログベースのサンプルソフト使用方法

5-5-1)概要

本ソフトはMicrosoftVisualC++で作られた計測サンプルソフトです。本サンプルソフトでは基本サブルーチンを呼び出し、DASmini及びDASBOXに対して制御を行い、データ計測、ファイル化、モニター、DA出力などの操作ができます。

5-5-2)ソフト仕様

１）動作環境

WindowsNT4.0、Windows2000、WindowsXP,Windows7

DASmini-E2000及びDASmini-E500シリーズ対応

２）多チャンネル波形モニター　（最大16CH）

３）多チャンネル計測　（最大16CH）

４）多チャンネルDA出力　（最大16CH）

５）ファイルフォーマット

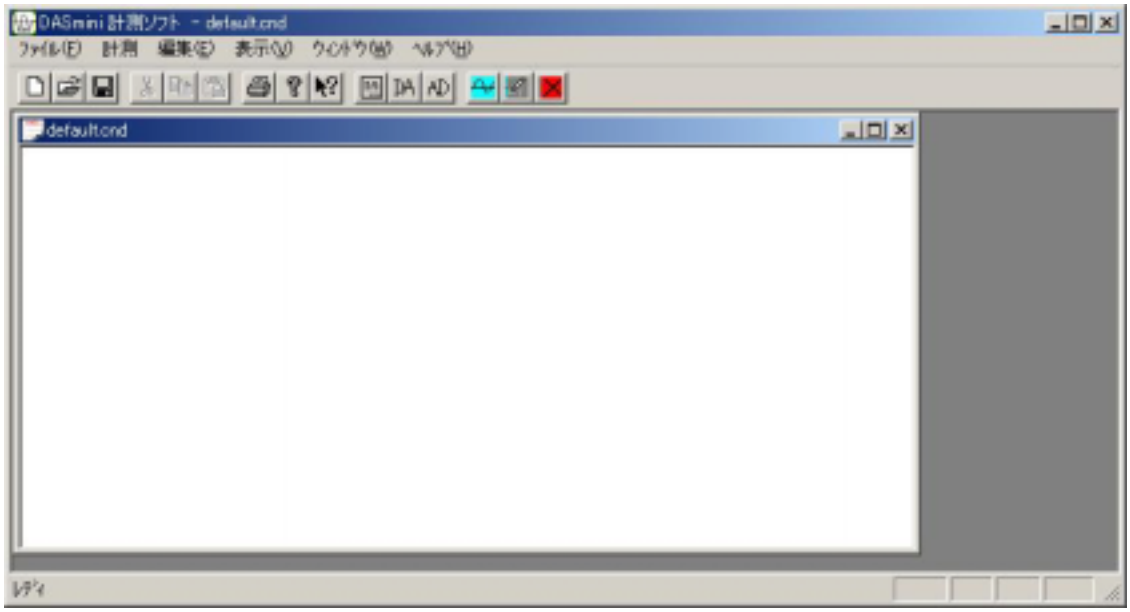
NORMAL,DATA_ONLY,DATA_HEADER

６）NORMALフォーマットファイルをCSV形式のテキストファイルに変換可能

5-5-３)操作説明

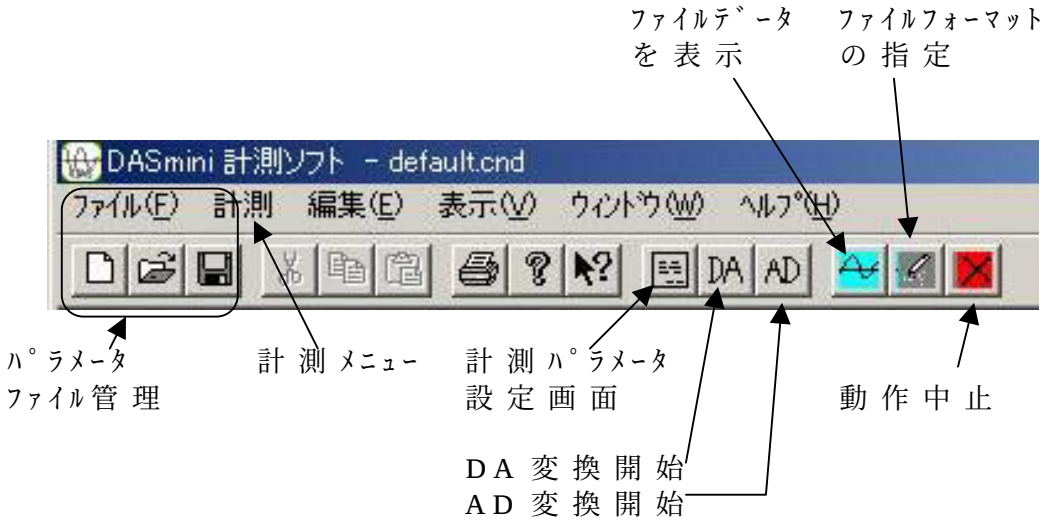
１）ソフトの立ち上げ

daswave.exeを実行すると、メイン画面が表示され、default.cndというパラメータファイルが自動的に読み込まれます。.

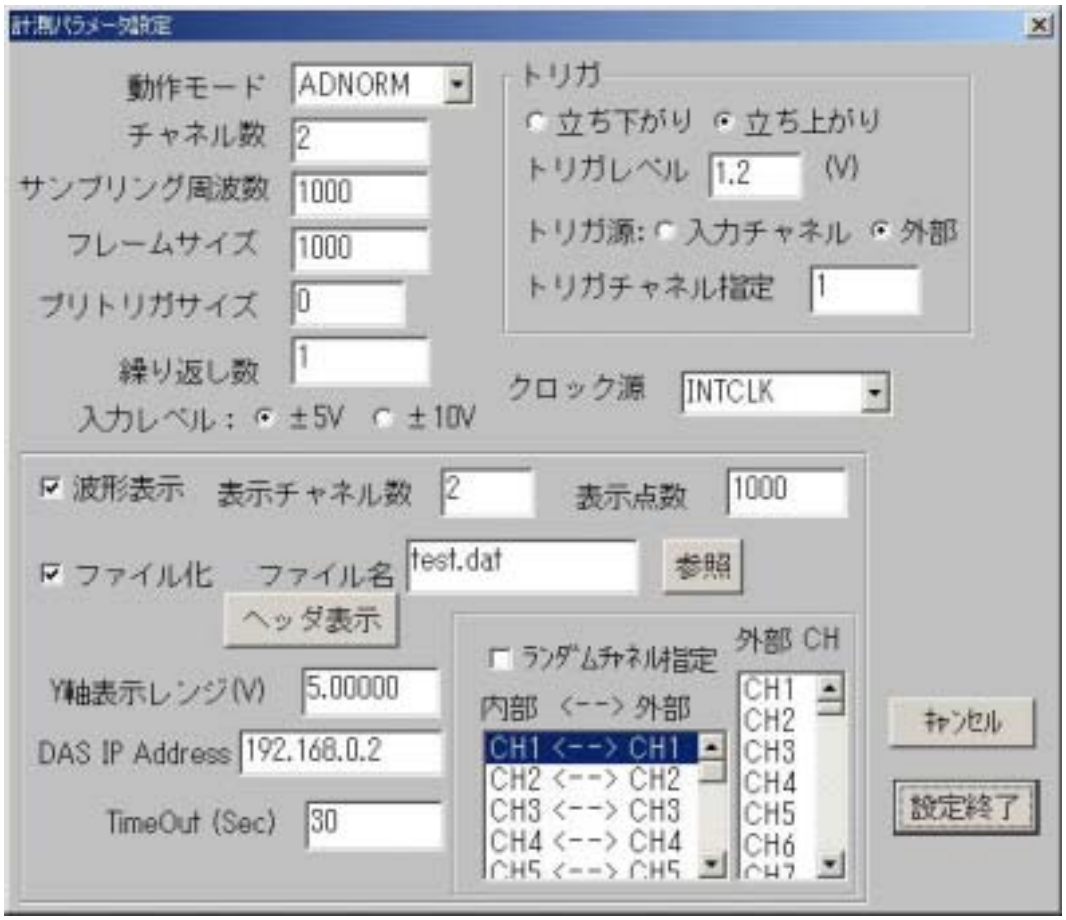


daswaveのメイン画面

２） メニューバー及びツールバーの説明



３） 計測パラメータの設定
下記の画面でAD変換とDA変換のパラメータを指定します。設定した結果は現在アクティブ状態になっているウィンドウと対応するパラメータとして有効になります。又パラメータはファイルとして保存する事が出来ます。この場合、コマンドメニューバーの「ファイル(F)」をクリックして「上書き保存」又は「名前を付けて保存」を選択して保存します。default.cndの名称で保存すると、立ち上げ時のデフォルトパラメータとなります。



3-1)パラメータ説明（詳細はﾊｰﾄﾞｳｪｱﾏﾆｭｱﾙを参照）

a) 動作モード

a-1)ADNORM（ADﾉﾝﾄﾘｶﾞｽﾀｰﾄﾓｰﾄﾞ）

このモードは、ホストコンピュータからのADスタートコマンドにより、AD動作を開始します。計測の開始を外部と同期する必要がない場合に使用します。ADの取り込みデータ数は、フレームサイズ×チャネル数になります。

a-2)ADTRG（ADﾉｰﾏﾙﾄﾘｶﾞｽﾀｰﾄﾓｰﾄﾞ）

このモードは、ホストコンピュータからのADスタートコマンドにより、外部からのトリガ信号待ちの状態（Trigger LED 緑点灯）になります。その後、

トリガ信号を検出すると（Trigger LED 消灯）、ＡＤ動作を開始します。計測の開始を外部と同期を取る必要がある場合に使用します。
ＡＤの取り込みデータ数は、フレームサイズ×チャネル数になります。

a-3)ＡＤＰＲＥ（ＡＤ ﾌﾟﾘﾄﾘｶﾞｽﾀｰﾄﾓｰﾄﾞ）
このモードは、ホストコンピュータからのＡＤスタートコマンドによりＡＤ動作を開始しますが、その後、トリガ信号を検出するとトリガ以前の設定された時点からのＡＤデータをホストコンピュータに転送します。ある外部事象（トリガ信号）が発生する以前の状態を必要とする計測に使用します。トリガ信号以前のデータ量はプリトリガサイズで設定します。
ＡＤの取り込みデータ数は、フレームサイズ×チャネル数になります。
但し、フレームサイズはプリトリガサイズを含むサイズを指定し、プリトリガサイズには以下の制限があります。
プリトリガサイズ×チャネル数≦ＤＡＳメモリ容量（標準4MW）－１００

a-4)ＡＤＲＥＴ（ＡＤ ﾘﾄﾘｶﾞｽﾀｰﾄﾓｰﾄﾞ）
このモードは、ノーマルトリガスタートと同様にＡＤ動作を開始しますが、１フレーム（フレームサイズ分）計測が終了すると、再度トリガ信号待の状態になりＡＤ動作を繰り返し行います。この繰り返しは繰り返し数で指定した回数実行します。
ＡＤの取り込みデータ数は、フレームサイズ×チャネル数×繰り返し数
になります。

b) チャネル数
AD、DA 変換のチャネル数を指定します。

c) サンプリング周波数
AD、DA 変換のサンプル周波数（Hz）を指定します。

d) フレームサイズ
計測する１フレーム（１回分の計測を１フレームと呼び、リトリガモードはこのフレームを指定した回数だけ繰り返します。）のサンプリング点数（１チャネル当たりのデータ点数）を指定します。

e) プリトリガサイズ
AD ﾌﾟﾘﾄﾘｶﾞｽﾀｰﾄﾓｰﾄﾞの時、プリトリガサイズ（トリガ以前のデータ数）を１チャネル当たりのデータ数で指定します。

f) 繰り返し数
「ADRET」、の時に、繰り返し数を指定します。

g)トリガの指定
計測パラメータ設定画面の右上にトリガ設定のセクションがあります。AD,DA のトリガを使用するモードの時に設定します。トリガの条件は、「立ち上がり」と「立ち下がり」の選択、トリガレベルの電圧指定（入出力レベル指定により設定範囲が異なります。±5V 又は±10V)、トリガソースの「外部入力端子」と「入力チャネル（オプション)」の選択が指定できます。

h) クロックソース
AD,DA 変換のクロックソースを指定します。
「INTCLK」： 内部クロックを使用します。（サンプリング周波数指定）
「EXTCLK」： 外部クロック（CLK IN 端子）を使用します。入力するクロックの立ち上がりで変換が開始されます。
「PRECLK」： 外部クロックを分周して使用します。分周値はサンプリング周波数設定の項目で設定します。
例： クロックソース=[PRECLK]、CLKIN 入力クロック＝1000Hz、サンプリング周波数指定＝10
変換クロック＝1000/10＝100Hz

i) □ 波形表示
モニター表示するかどうかを指定します。チェックで表示

j) 表示チャネル数
モニター表示するチャネル数を指定します。

k) 表示点数
モニター表示するデータの点数を指定します。
計測チャネル数×表示点数が取り込みメモリー領域(32768Word)サイズを超える場合は、自動的に表示点数が最大値に変更されます。

l) □ ファイル化
ファイルに書き込むかどうかを指定します。チェックで書き込み

m) ファイル名
ファイルを操作する時のデータファイル名を指定します。

n) ヘッダ表示
ファイル名に指定されているデータファイルのヘッダ情報を表示します。
表示項目：DataFileType,DataSize、DataChannels,SampleClock

o) DAS IP Address
DASBOX の IP ADDRESS を指定します。通常は“ 192.168.0.2 ”を設定

p) Y 軸表示レンジ
モニター表示するとき Y 軸の表示レンジを電圧値で指定します。DASBOX の電圧レンジは±５V 又は±10V としています。

q) TimeOut
ホストと DASBOX の間でデータ転送を行う時、TimeOut 時間を秒単位で指定します。

r) ランダムチャネル指定
ランダムチャネルの指定を有効にした時は、DASBOX のチャネル入力端子の番号と入出力順番の対応関係を指定できます。
指定の手順
- パラメータ設定画面の右下にある 内部＜――＞外部 対応関係リストから指定したい項目を選びます。外部は入出力端子、内部はメモリ及びファイルデータのチャネル番号です。
- 外部 CH のリストからチャネル番号を選ぶと、内部＜――＞外部対応関係リストの外部が変えられます。

s) キャンセル
「キャンセル」ボタンを押すと、パラメータ設定画面の設定を全てキャンセルして設定前のパラメータに戻ります。

t) 設定終了
「設定終了」ボタンを押すと、現在の設定値が有効になり、パラメータ設定画面が終了します。

u) 入出力レベル
DASBOX の外部トリガ入力レンジ選定及び DASBOX の入出力レベルを設定します。但し、トリガレンジはプログラマブルですが、各チャンネルの入出力レベルは、出荷時に固定となっております。(標準±5V,オプション±10V)

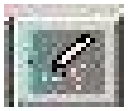
４） アイコンの説明

a)パラメータ設定
計測パラメータ設定画面が表示され、計測のパラメータ設定を行います。設定の詳細は３）計測パラメータ設定の項目を参照してください。

b)DA スタート
計測パラメータで DA 変換を行います。パラメータの動作モードが DA モードではない場合は、自動的に DANORM モードに変更します。(パラメータ設定画面には反映されません。)

c)AD スタート
計測パラメータで AD 変換を行います。パラメータの動作モードが AD モードではない場合は、自動的に ADNORM モードに変更します。(パラメータ設定画面には反映されません。)

d)ファイル表示
指定したファイルを読み込み、表示します。表示はパラメータ設定画面で設定した「表示点数」毎に 1 フレーム毎に表示します。



e)データフォーマット指定
計測した結果をファイル化する際のファーマットを指定します。
・NORMAL　　先頭にヘッダが付いた、データファイル
・DATA_ONLY　　データだけのファイル
・DATA_HEADER　テキストのヘッダファイル及びデータだけのファイル

f) 実行中止
実行中の動作を中止します。(計測を途中で中断したい場合等に用います。)

5） 計測メニュー

計測メニューは下記のプルダウンメニューを持ち各種の動作を実行します。

- パラメータ設定
- startAD
- startDA
- ファイルフォーマット
- データ表示
- 中止
- RESET
- ファイル変換（bin->text）

・パラメータ設定： パラメータ設定のアイコンと同じ動作

・startAD： AD スタートアイコンと同じ動作

・startDA： DA スタートアイコンと同じ動作

・ファイルフォーマット： データフォーマット指定アイコンと同じ動作

・データ表示： ファイル表示アイコンと同じ動作

・中止： 実行中止アイコンと同じ動作

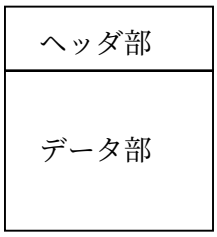
・RESET： DASBOX のイニシャライズ（inet_io_init()）を実行します。

・ファイル変換（bin->text）： NORMAL フォーマットのファイルのみ有効とし、計測パラメータ設定画面のファイル名で指定されている Binary 形式のファイルを読み込み、*****.txt の名称で CSV 形式のテキストファイルを作成します。
*****はファイル名で指定した拡張子を外した名称とします。

６）データフォーマットの説明

ａ）NORMAL

NORMAL フォーマットは本ソフト専用なデータフォーマットであり、計測条件と計測した結果を全て Binary 形式で保存されています。



1024Byte

データサイズ×チャネル数×２Byte

・ ヘッダ部（1024Byte）
計測のパラメータ、バージョン情報などが入っています。

Float	ver；	バージョン情報
Int	format;	ファイルフォーマットの指定 0=NORMAL
Int	reserved;	
Int	reserved;	
Int	flag;	ファイルの有効性　０＝無効、１＝有効
Int	channels；	チャネル数
Int	frame_size；	フレームサイズ
Float	clock;	サンプリング周波数（Hz）
Int	counter	繰り返し数
Char	dumy[988]	ダミィーデータ

・ データ部
データ型：　１６Bit 符号付き　short
レンジ：　－32768～+32767　――＞　–5V～+4.999847412V
データの並び：　[ch1-0],[ch2-0],[ch3-0]------[chN-0],
[ch1-1],[ch2-1],[ch3-1]------[chN-1],
|
[ch1-n],[ch2-n],[ch3-n]------[chN-n]

b）DATA_ONLY

DATA_ONLY フォーマットは NORMAL フォーマットのデータ部だけが Binary 形式で保存されています。

ｃ）DATA_HEADER

DATA_HEADER フォーマットは計測条件ファイル(***.hed)と NORMAL フォーマットのデータ部(***.dat)が別のファイルに保存されています。
計測条件ファイルはテキストフォーマットでファイル化されています。
計測条件ファイルの内容

Ver	*.****	バージョン情報
Clock	**.*	サンプリング周波数
Size	***	フレームサイズ
Channels	**	チャネル数
Counter	**	繰り返し数