# 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１８年　６月　７日
（kit2.11、またはkit2.11-24に対応）
改訂　平成１８年　６月１６日（kit2.12）
改訂　平成１９年　４月　２日（kit2.12bに対応）
改訂　平成１９年　５月２２日（kit2.13に対応）
改訂　平成１９年　７月　３日（kit2.14に対応）
改訂　平成１９年　８月１７日（kit2.14aに対応）
改訂　平成１９年　９月１８日（kit2.14bに対応）
改訂　平成２０年１１月１９日（kit2.15に対応）
改訂　平成２２年　９月１５日（kit2.16に対応）
改訂　平成２３年　４月　７日（kit2.17に対応）
改訂　平成２４年　３月２３日（kit2.18に対応）

ケイテクノス株式会社

# 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.....8
*第３章 DASBOX 関数の使用方法*.....*9*
3.1 ＡＤ／ＤＡ（入出力）共通.....10
3.1.1 inet_io_open.....10
3.1.2 inet_io_close.....11
3.1.3 inet_io_packet.....12
3.1.4 inet_io_cond.....14
3.1.5 inet_io_stat.....16
3.1.6 inet_io_stop.....17
3.1.7 inet_io_init.....18
3.1.8 inet_io_info.....19
3.1.9 inet_io_error.....21
3.1.10 inet_io_blkf.....22
3.1.11 inet_amp_set.....23
3.1.12 inet_amp_cutoff.....24
3.1.13 inet_amp_bcutoff.....26
3.1.14 inet_amp_afcal（オプション）.....27
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）.....28
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）.....31
3.1.15 inet_io_info2.....32
3.1.16 inet_io_clock_sync（24bit 専用）.....34
3.1.17 inet_k_amp_set.....35
風圧センサーアンプ設定アーギュメントの説明（KAMP アーギュメント）.....36
3.2 ＡＤ（入力）専用.....38
3.2.1 inet_io_adstart.....38
3.2.2 inet_io_adread.....39
3.2.3 inet_io_adfile.....41
3.2.4 inet_io_pre.....42
3.2.5 inet_io_count.....43
3.2.6 inet_io_adda_cal（24bit 専用）.....45
3.2.7 inet_io_24b16m_adread（24bit 機 16 ビットデータモード専用）.....46
3.2.8 inet_io_24b16m_adfile (24bit 機 16 ビットデータモード専用).....48
3.3 ＤＡ（出力）専用.....49
3.3.1 inet_io_dastart.....49
3.3.2 inet_io_dawrite.....50
3.3.3 inet_io_dafile.....52
3.3.4 inet_io_dawait.....53
3.3.5 inet_io_cycle_stop.....54
3.3.6 inet_io_daclear.....55
3.3.7 inet_io_mute（24bit 専用）.....56
3.3.8 inet_io_24b16m_dawrite（24bit 機 16 ビットデータモード専用）.....57

3.3.9 inet_io_24b16m_dafile（24bit 機 16 ビットデータモード専用）..... 59
3.4 サブルーチン実行基本フロー ..... 60
3.4.1 ＡＤ動作 ..... 60
3.4.2 DA 動作 ..... 62
*第４章 ＤＡＳＢＯＸアーギュメント説明 ..... 63*
4-1 mode ..... 64
4-2 stat ..... 65
4-3 dmatime ..... 65
4-4 attn（未使用） ..... 65
4-5 gain ..... 66
4-6 trgslp ..... 66
4-7 clkmode ..... 66
4-8 clock ..... 67
4-9 frame1 ..... 68
4-10 frame2 ..... 68
4-11 frame3 ..... 68
4-12 frame4 ..... 68
4-13 frame5 ..... 69
4-14 mutelevel（24 ビット機種の inet_io_packet 時のみ有効） ..... 69
4-15 trglevel ..... 69
4-16 trgsrc ..... 69
4-17 trgch ..... 70
4-18 channels ..... 70
4-19 chanum【1024】 ..... 70
4-20 master（24 ビット機種の inet_io_packet 時のみ有効） ..... 70
4-21 ext_parm（24 ビット機種の inet_io_packet 時のみ有効） ..... 71
4-22 dmasize,pre_mem_size ..... 71
*第５章 ＤＡＳＢＯＸソフトウェア作成 ..... 72*
5-1 インストール ..... 72
5-2 キット内容 ..... 72
5-3 ユーザープログラムへの組み込み ..... 75
5-4 コマンドベースのサンプルソフト使用方法 ..... 76
5-5 ダイアログベースのサンプルソフト使用方法 ..... 82

## 改訂履歴

・平成 15 年４月　初版作成

・平成 18 年 6 月 7 日　（V2.11、または V2.11-24）

１．　True64 UNIX 等６４ビットコンパイラに対応、
Long 使用の型宣言を int に変更。

２．　コンパイルオプション B24 により２４ビット分解能 DASBOX、DASminiE2000 を使用するサブルーチン・オブジェクトを生成する。

３．　コンパイルオプション COMPATI_24b5x0 により従来 DASBOX Model-5x0 基本サブルーチンとのできる限りの互換性をとる。
また、COMPATI_24b5x0 を使う場合は 32 ビットデータモードのみ使用で、常にマスタ装置（master=1）になります。

４．　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。（２４ビット DASminiE2000、DASBOX-E のみ）

５．　関数名の変更

２４ビット DASminiE2000　32 ビットデータモード時

inet_io_adread32　-→　inet_io_adread
inet_io_adfile32　-→　inet_io_adfile
inet_io_dawrite32　-→　inet_io_dawrite
inet_io_dafile32　-→　inet_io_dafile
inet_io_packet24　-→　inet_io_packet

関数追加

２４ビット DASminiE2000　１６ビットデータモード

inet_io_24b16m_adread
inet_io_24b16m_adfile
inet_io_24b16m_dawrite
inet_io_24b16m_defile

平成１８年6月１６日（kit2.12）

１．１６ビット DASBOX の read・write も BIG エンディアン OS でのデータスワップに対応。コンパイルオプション BIG 付きでコンパイルする。

２．１６ビット機、２４ビット機のキットを共通化。

平成１９年４月　２日

１．3.1.14　inet_amp_afcal　関数をオプション扱いに変更

２．アンプフィルタ設定アーギュメントの説明にて
offmode=Fine の場合、±0.1Ｖを±0.2V に訂正

３．Offset 機能標準装備（オプション扱いなし）

平成１９年５月２２日

１．DASBOX-E　AI32-24/20M ﾓｼﾞｭｰﾙ対応のため、コンパイルオプション ADC1271 を追加。
DASBOX-E シリーズにて AI32-24/20M ﾓｼﾞｭｰﾙを使用する場合はコンパイルオプション B24 と ADC1271 付きでコンパイルする。

平成１９年7月3日

１．DASBOX アーギュメントの説明にて、gain 設定を有効とし、説明を追加した。

２．Kit2.14 にて下記のサンプルソフトを追加及び変更した為、説明を追加

１）dawave　VisualStudio6.0 環境でのダイヤログベースのサンプルソフト

２）samplevc VisualStudio6.0 環境でのコマンドベースのサンプルソフト

３）コマンドベースサンプルソフトソース apl90.c に DARET,DACYCL DATCYC,DARCYC、ADTEST,DATEST のモード動作を追加

平成１９年8月１７日

１．サンプルソフト daswave　VisualStudio6.0 環境でのダイヤログベースのサンプルのバージョンアップのため、説明追加

１）IP　ADDRESS 設定可能

２）ランダムチャンネル設定の最大チャンネル番号 16－＞64

３）gain 設定対応

２．サンプルソフト apl90.c のバージョンアップのため、説明追加

１）アーギュメントメンバーgain（トリガゲイン選択）のサポート

２）ifdef にて、B24 の場合はチャンネルトリガ選択禁止とした。

平成１９年9月１９日

１．4-18　channles 設定にて、奇数チャンネル設定での注意事項及び偶数チャンネル設定の推奨。

平成２０年1月１９日

１．3.2.4　トータルチャンネルに対しての個数でなく、１チャンネルに対しての個数に変更。

平成２０年１１月１９日

１．4-21　ext_parm（24 ビット機種の inet_io_packet 時のみ有効）追加
4-22　修正

平成２２年　9月１５日

3.1.17　inet_k_amp_set()関数の追加

平成２３年　4月　7日

１．　32bit Windows 用 DLL 及び C#関数説明追加

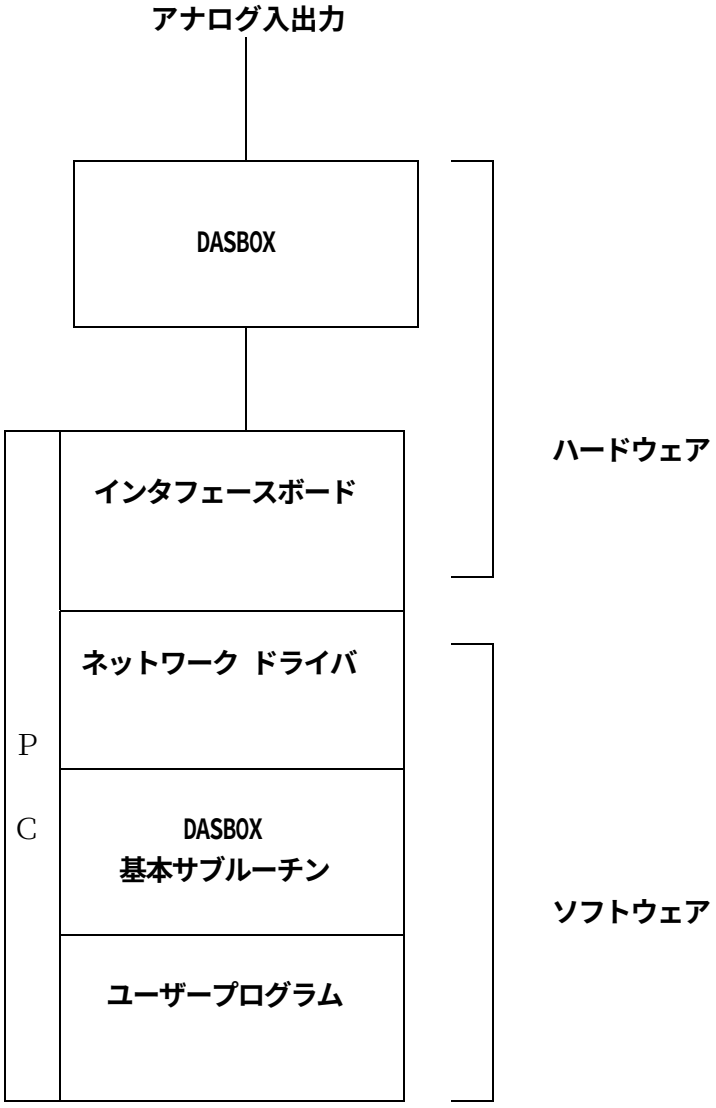
平成２４年　3月２３日

１．　64bit Windows 用 DLL 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。
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 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。
　　基本サブルーチンV2.11で２４ビットDASBOXに対応し共通となりました。

・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）

・inet_io_packet24()　　―――＞　V2.11より当関数は削除。
　　DASBOXに対して動作を指示します。（24ビットモジュール専用）

・inet_io_clock_sync()
　　DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）に対して動作を指示します。
　　複数台のDASBOXのサンプリングクロックの位相を同期させます。

・inet_k_amp_set()
　　風圧センサーアンプ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。
　　基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。但し24ビット機16ビットデータモードではinet_io_24b16m_adread()を使用します。

・inet_io_adfile()
　　DASBOXからのADデータをファイルに書き込みます。
　　基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。但し24ビット機16ビットデータモードではinet_io_24b16m_adfile()を使用します。

・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無効データかホストに知らせます。

・inet_io_count()
　　DASmini-E2000シリーズにてオプションでＤＩ機能を追加した場合に、各カウンタ（パルスカウンタ１，パルスカウンタ２，エンコーダ入力）を“０”クリアし、エンコーダ入力の機能設定をします。（オプション）

・inet_io_adda_cal()
　　DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）内のADのオフセット・キャリブレーションを行います。ADCのDCオフセットの自動校正を行う関数です。
　　必ずしも毎回の計測で行う必要はありませんが、装置の電源投入後１回は行うことを推奨します。

・inet_io_24b16m_adread()

24ビット機16ビットデータモードでDASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）からのADデータをメモリ（16ビットデータ）上に読み込みます。

・inet_io_24b16m_adfile()

24ビット機16ビットデータモードでDASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）からのADデータ（16ﾋﾞｯﾄ）をファイルに書き込みます。

### 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。
基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。但し24ビット機16ビットデータモードではinet_io_24b16m_dawrite()を使用します。

・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。
基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。
但し24ビット機16ビットデータモードではinet_io_24b16m_defile()を使用します。

・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()

動作（計測）停止状態のDASBOXのDA出力を０V出力状態にします。

・inet_io_mute()

DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）内の全てのDA出力にmuteをかけます。

・inet_io_24b16m_dawrite()

24ビット機16ビットデータモードでDASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）にDAデータ（16ビット）をメモリ上から書き込みます。

・inet_io_24b16m_dafile()

24ビット機16ビットデータモードでDASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）へのDAデータをファイルから読み込み、DASBOXにDAデータ（16ビット）を送信します。

# 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。

C関数では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。（sadtp24.hを、インクルードしても同様に機能します。）

C#関数ではコントロールブロック及びDASBOX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 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。

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 | 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。<br>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。

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 |

### 3.1.3 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。
基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。
但し、次の２つの場合で引数の数がことなります。

C形式：

① １６ビット迄のDASBOX、及び２４ビット機で３２ビットデータモードの時

```
int  inet_io_packet(cblock,  dasarg);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
```

②２４ビット機の場合は１６ビットデータモード、３２ビットデータモードとも
次の仕様で扱うことができます。

```
int  inet_io_packet(cblock,  dasarg,   bit16);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
int   bit16;
```

C#形式：

① １６ビット迄のDASBOX、及び２４ビット機で３２ビットデータモードの時

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

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

②２４ビット機の場合は１６ビットデータモード、３２ビットデータモードとも
次の仕様で扱うことができます。

```
int  Dasbox.inet_io_packet(ref  cblock,  ref  dasarg,   bit16);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.PCISAD_ARG  dasarg  =  Dasbox.CBLK.New();
int   bit16;
```

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 |
| bit16 | ２４ビット装置にて１６ビットデータモードを指定<br>0 = ３２ビットデータモード |

１　＝　１６ビットデータモード

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。

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
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ＊ | ＊ | ＊ | ＊ | ＊ | NEM | ＊ | ERR | ＊ | EN | ＊ | TRG | ＊ |

＊ 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。

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 | 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。

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 | 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。

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 |

### 3.1.8 inet_io_info

機能：DASBOX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))<br>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チャネル数が入ります。

**4） output_channel**

DASBOXの最大DAチャネル数が入ります。

**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“

| Ｓ１ | | | |
|---|---|---|---|
| 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。

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

### 3.1.10 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。

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。<br>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=１Hz、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“AF”モデルに対応しています。

C形式：

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

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

C#形式：

```
int Dasbox.inet_amp_set (ref cblock, ref afarg, channel);

Dasbox.CBLK   cblock = Dasbox.CBLK.New();
Dasbox.AF_ARG   afarg = Dasbox.AF_ARG.New();
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の項を参照願います。

### 3.1.12 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。
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“AF”モデルに対応しています。

C形式：

```
int inet_amp_cutoff(cblock, afarg);

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

C#形式：

```
int Dasbox.inet_amp_cutoff(ref cblock, ref afarg);

Dasbox.CBLK   cblock = Dasbox.CBLK.New();
Dasbox.AF_ARG   afarg = DASbox.AFARG.New();
```

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“AF”モデルに対応しています。

C 形式：

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

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

C#形式：

```
int Dasbox.inet_amp_bcutoff(ref cblock, ref barg, block);

Dasbox.CBLK  cblock = Dasbox.CBLK.New();
Dasbox.BCUTOFF_ARG barg = Dasbox.BCUTOFF_ARG.New();
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 ・・・全てのチャネル (cutoff[0]～[15])
block = 1 ・・・１チャネル～８チャネル (cutoff[0])
block = 2 ・・・９チャネル～１６チャネル (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];
};
```

### 3.1.14　inet_amp_afcal（オプション）

機能： 8チャネルのブロック単位にオフセットのキャリブレーションを行います。
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“AF”モデルに対応しています。

C形式：

```
int  inet_amp_afcal(cblock,  block);

CBLK   *cblock;
int  block;
```

C#形式：

```
int  Dasbox.inet_amp_afcal(ref  cblock,  ref  block);

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

引数：

cblock　コントロールブロックのポインタ
block　設定するブロックを指定します。
例）
block = 0　・・・全てのチャネル
block = 1　・・・１チャネル～８チャネル
block = 2　・・・９チャネル～１６チャネル

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。

## 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） |

２）offval[128]

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。
通常は全チャンネル、0 を設定して下さい。

| 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 / +0.2V） |
| ｜ | ｜ |
| 0 | 0 （0V） |
| ｜ | ｜ |
| -128(80) | -FS （-5V / -0.2V） |

3）offmode[128]

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。
通常は全チャネル、1=Fine を 設定して下さい。

| 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 | ±0.2V （Fine） |

4）input[128]

指定したチャネルの入力モード(AC/DC/ICP)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 |
| 2 | ICP |

5）imode[128]

指定したチャネルの入力モード(SIGNAL/GND)を設定します。
通常は全チャンネル、0=SIGNAL に設定して下さい。

| 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（計測可能状態） |
| 1 | GND （入力信号を切り離し GND に接続） |

6）pathen[128]

予約

7）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 | Through | |
| １５ | EXT.Mode | （オプション） |

8）calsel

予約

9）revolution

予約

１０）pulse

予約

１１）change

予約

１２）average

予約

１３）cmplevel

予約

１４）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 | 指定チャネル |
|---|---|
| [0] | １～８チャネル |
| [1] | ９～１６チャネル |
| ｜ | ｜ |
| [15] | １２１～１２８チャネル |

| 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） |

### 3.1.15 inet_io_info2

機能：DASBOXのバージョン情報を得ます。

C形式：

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

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

C#形式：

```
int  Dasbox.inet_io_info2(ref  cblock,  ref  statptr,  size);

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
Dasbox.DASINFO  statptr  =  Dasbox.DASINFO.New();
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))<br>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）

### 3.1.16　inet_io_clock_sync（24bit 専用）

機能：DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）に対して動作を指示します。
　　複数台のDASBOXのサンプリングクロックの位相を同期させます。

C形式：

```
int  inet_io_clock_sync(cblock);

CBLK  *cblock;
```

C#形式：

```
int  Dasbox.inet_io_clock_sync(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 |

詳細：２台以上のDASBOXでMASTER、SLAVEでサンプリングクロックの位相を同期させるときに必要なコマンドです。
MASTER側DASBOXのみ本コマンドを実行します。
コマンドを実行するときは、MASTER装置とSLAVE装置を専用の同期ケーブルで接続し、このケーブルの信号によりSLAVE装置はハード的に同期合わせが行われます。
１台でDASBOXを使用するときは、当コマンドは必要ありません。

### 3.1.17 inet_k_amp_set

機能： 指定したチャネルに風圧センサー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風圧センサーアンプモジュール、
に対応しています。

C形式：

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

CBLK   *cblock;
KAF_ARG *afarg;
int    channel;
```

C#形式：

```
int Dasbox.inet_k_amp_set（cblock, afarg, channel）;

Dasbox.CBLK   cblock = Dasbox.CBLK.New();
Dasbox.KAF_ARG afarg = Dasbox.KAF_ARG.New();
int    channel;
```

引数：

cblock　コントロールブロックのポインタ
afarg　KAMP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 |
| pathen[128] | フィルタ切替 |

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：
本関数は風圧センサーアンプモジュールを内蔵したDASBOXで有効です。
KAMPアーギュメントはpci_sad.hに構造体として定義されています。

## 風圧センサーアンプ設定アーギュメントの説明（KAMP アーギュメント）

１） 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 | 倍率 |
|---|---|
| 0 | ２００倍 |
| 1 | ５００倍 |
| 2 | １０００倍 |
| 3 | ２０００倍 |

２）offval[128]
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。
通常は全チャンネル、0 を設定して下さい。
オフセット電圧は入力電圧に対する値ですので、アジャスト値は
下記の式で算出してください。

0V 入力時の AD 変換値（電圧値）/ アンプ倍率 ＝ 入力オフセット電圧

アジャスト値は-入力オフセット電圧となります。

| 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 | |
|---|---|---|
| 2048 | +FS | +20.830mV |
| 2047 | +FS-1LSB | +20.819mV |
| ｜ | ｜ | ｜ |
| 1 | +1LSB | +0.010mV |
| 0 | 0 | 0.000mV |
| -1 | -1LSB | -0.010mV |
| ｜ | ｜ | ｜ |
| -2047 | -FS+1LSB | -20.819mV |

１タップの設定電圧は 20.830mV/2048 0.01017mV

3）pathen[128]

指定したチャネルに対してフィルタの使用/未使用を設定します。

| 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 | 内容 |
|---|---|
| 0 | 使用 |
| 1 | 未使用（パス） |

＊）フィルタはローパスフィルタで
４次のバタワース特性にてカットオフ周波数は５ｋHz 固定となります。

## 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。

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 |

### 3.2.2 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。
基本サブルーチンV2.11より２４ビット機に対応し共通となりました。

C形式：

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

CBLK  *cblock;
short  *buffer;  （２４ビット機の３２ビットデータモードでは  int  *buffer
                となります。）
int  count;
```

C#形式：

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

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
short[,]  buffer;  （２４ビット機の３２ビットデータモードでは  int[,]  buff
                erとなります。）
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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| count | 読み込むADデータ数<br>基本的には、データ転送の効率を上げるために、下記の<br>設定にしてください。<br>inet_io_blkf()で設定したsize×計測チャンネル数<br>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<br>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<br>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)
DASBOX から 2000 データ Read
inet_io_adread(,,1000)
inet_io_adread(,,1000)
DASBOX 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 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。
　　基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。

C形式：

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

CBLK  *cblock;
short  *buffer;   （２４ビット機の３２ビットデータモードでは　int　*buffer
                    となります。）
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;   （２４ビット機の３２ビットデータモードでは　int[,]  buffer
                    となります。）
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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| 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、1チャンネルに対して
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。

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 | 無効データ数ポインタ |

戻り値：

| | |
|---|---|
| 0 | 正常終了 |
| -1 | 異常終了 |

### 3.2.5 inet_io_count

機能： オプションにてＤＩ機能を追加した場合に、各カウンタ（パルスカウンタ１，パルスカウンタ２，エンコーダ入力）を“０”クリアし、エンコーダ入力の機能設定をします。

C形式：

```
int inet_io_count(cblock, modedata);

CBLK *cblock;
int modedata
```

C＃形式：

```
int Dasbox.inet_io_count(ref cblock, modedata);

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

引数：

| | |
|---|---|
| cblock | コントロールブロックのポインタ |
| modedata | エンコーダ入力モード設定データ |

| D31―――D9 | D7 D6 D5 D4 | D3 D2 D1 D0 |
|---|---|---|
| 未使用 | ENC2 | ENC1 |

ENC1：

・エンコーダ１の逓倍モード設定 （標準）

| Ｄ１ | Ｄ０ | 逓倍値 | |
|---|---|---|---|
| ０ | ０ | １逓倍 | （パワーオン時） |
| ０ | １ | ２逓倍 | |
| １ | ０ | ４逓 | |

Ｄ２： 未使用

Ｄ３：Ｚ相有効／無効

| | | |
|---|---|---|
| ０ | 有効 | （パワーオン時） |
| １ | 無効 | |

ENC2:

・エンコーダ２の逓倍モード設定 （特注増設した場合）

| Ｄ５ | Ｄ４ | 逓倍値 | |
|---|---|---|---|
| ０ | ０ | １逓倍 | （パワーオン時） |
| ０ | １ | ２逓倍 | |
| １ | ０ | ４逓倍 | |

Ｄ６： 未使用

Ｄ７：Ｚ相有効／無効

| | | |
|---|---|---|
| ０ | 有効 | （パワーオン時） |

1　　無効

設定データ例：

modedata=2　　ENC1：　４逓倍にてＺ相有効

modedata=10　　ENC1：　４逓倍にてＺ相無効

### 3.2.6 inet_io_adda_cal（24bit 専用）

機能：DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）内のADのオフセット・キャリブレーションを行います。ADCのDCオフセットの自動校正を行う関数です。
必ずしも毎回の計測で行う必要はありませんが、装置の電源投入後１回は行うことを推奨します。又、この関数でDASBOXアーギュメントメンバーのclock設定値を使用しますので、計測時の値を入れておく必要があります。
但し、DASBOX-EシリーズのAI32-24/20Mﾓｼﾞｭｰﾙの場合はこの機能はありません。

C形式：

```
int  inet_io_adda_cal(cblock,  dasarg);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
```

C#形式：

```
int  Dasbox.inet_io_adda_cal(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 | DASBOXアーギュメントのポインタ<br>アーギュメントメンバーのclock設定を参照しますので、<br>実行する前にclockの設定が必要です。 |

戻り値：

| | |
|---|---|
| 0 | 正常終了 |
| -1 | 異常終了 |

### 3.2.7 inet_io_24b16m_adread（24bit機 16 ビットデータモード専用）

機能：DASBOXからのADデータ（16bitデータ）をメモリ上に読み込みます。
24Bit機種で使用しますが、24bit機種の16bitモード専用関数で32ビットデータモードを使用する場合は、inet_io_adread（）を使用してください。

C形式：

```
int  inet_io_24b16m_adread(cblock,  buffer,  count);

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

C#形式：

```
int  Dasbox.inet_io_24b16m_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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| count | 読み込むADデータ（１６ビットデータ）数<br>基本的には、データ転送の効率を上げるために、下記の<br>設定にしてください。<br>inet_io_blkf()で設定したsize×計測チャンネル数<br>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<br>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<br>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_24b16m_adread(,,1000)
inet_io_24b16m_adread(,,1000)
DASBOX から 2000 データ Read
inet_io_24b16m_adread(,,1000)
inet_io_24b16m_adread(,,1000)
DASBOX から 96 データ Read
inet_io_24b16m_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 DASBOX からの Read は 2000 データ Read して inet_io_24b16m_adread(,,96)で 96 データ buffer に渡します。

### 3.2.8 inet_io_24b16m_adfile (24bit 機 16 ビットデータモード専用)

機能：DASBOXからADデータ（16bitデータ）をファイルに書き込みます。
24Bit機種16ビットデータモードで使用しますが、24bit機種の32bitデータモードで使用する場合は、inet_io_adfile（）を使用してください。

C形式：

```
int inet_io_24b16m_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_24b16m_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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| size | テンポラリバッファのサイズ（16bitデータ） |
| 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（16bitデータ） |

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。

C形式：

```
int  inet_io_dastart(cblock);

CBLK  *cblock;
```

C#形式：

```
int  inet_io_dastart(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 |

### 3.3.2 inet_io_dawrite

機能：DASBOXにDAデータをメモリ上から書き込みます。
基本サブルーチンV2.11より24ビットDASBOXに対応し共通となりました。

C形式：

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

CBLK  *cblock;
short  *buffer;   （２４ビット機の３２ビットデータモードの場合、int   *buffer となります。）
int  count;
```

C#形式：

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

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
short[,]  buffer;   （２４ビット機の３２ビットデータモードの場合、int[,]   buffer となります。）
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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| count | 書き込むDAデータ数<br>基本的には、データ転送の効率を上げるために、下記の<br>設定にしてください。<br>inet_io_blkf()で設定したsize×計測チャンネル数<br>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<br>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<br>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_dawrtie(,,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。
　　基本サブルーチンV2.11より２４ビットDASBOXに対応し共通となりました。

C形式：

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

CBLK  *cblock;
short  *buffer;   （２４ビット機の３２ビットデータモードの場合、int   *buffer
                  となります。）
int  size;
char  *file_name;
unsined  int  frame;
```

C#形式：

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

Dasbox.CBLK  cblock  =  Dasbox.CBLK.New();
short[,]  buffer;   （２４ビット機の３２ビットデータモードの場合、int[,]   buffer
                  となります。）
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<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| 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。

C形式：

```
int  inet_io_dawait(cblock,  time);

CBLK  *cblock;
int  time;
```

C#形式：

```
int  Dasbox.inet_io_dawait(ref  cblock,  time);

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

C形式：

```
int  inet_io_cycle_stop(cblock);

CBLK  *cblock;
```

C#形式：

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

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

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。

C形式：

```
int  inet_io_daclear(cblock);

CBLK  *cblock;
```

C#形式：

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

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

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 |

### 3.3.7 inet_io_mute（24bit 専用）

機能：DASBOX（24ﾋﾞｯﾄﾓｼﾞｭｰﾙ専用）内の全てのDA出力にmuteをかけます。

C形式：

```
int  inet_io_adda_cal(cblock,  dasarg,  mute_on);

CBLK  *cblock;
PCISAD_ARG  *dasarg;
int        mute_on;
```

C#形式：

```
int  Dasbox.inet_io_adda_cal(ref  cblock,  ref  dasarg,  mute_on);

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

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<br>アーギュメントメンバーのclock設定を参照しますので、<br>実行する前にclockの設定が必要です。 |
| mute_on | muteのON/OFFを切り替えます.<br>０　＝　muteをOFFにします。<br>１　＝　muteをONにします。 |

戻り値：

| | |
|---|---|
| 0 | 正常終了 |
| -1 | 異常終了 |

### 3.3.8 inet_io_24b16m_dawrite（24bit機16ビットデータモード専用）

機能：DASBOXにDAデータ(16bitデータ)をメモリ上から書き込みます。
24bit機種の16ビットデータモードで使用しますが、24bit機種の32bitデータモードで使用する場合は、inet_io_dawrite（）を使用してください。

C形式：

```
int  inet_io_24b16m_dawrite(cblock,  buffer,  count);

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

C#形式：

```
int  Dasbox.inet_io_24b16m_dawrite(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 |
| buffer | DAデータ(16bit)を格納するバッファのポインタ<br>Cではバッファを2次元配列とした場合、配列要素は<br>[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は<br>[フレーム数][チャネル数]となります。 |
| count | 書き込むDAデータ（16ビットデータ）数<br>基本的には、データ転送の効率を上げるために、下記の<br>設定にしてください。<br>inet_io_blkf()で設定したsize×計測チャンネル数<br>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<br>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<br>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_24b16m_dawrite(,,1000)
inet_io_24b16m_dawrtie(,,1000)
DASBOX に 2000 データ Write
inet_io_24b16m_dawrite(,,1000)
inet_io_24b16m_dawrite(,,1000)
DASBOX に 2000 データ Write
inet_io_24b16m_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。

### 3.3.9 inet_io_24b16m_dafile（24bit 機 16 ビットデータモード専用）

機能：DASBOXへのDAデータ(16bitデータ)をファイルから読み込み、DASBOXに転送します。24bit機種の16ビットデータモードで使用しますが、24bit機種の32ビットデータモードで使用する場合は、inet_io_dafile（）を使用してください。

C形式：

```
int inet_io_24b16m_dafile(cblock, buffer, size, file_name, frame);

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

C#形式：

```
int Dasbox.inet_io_24b16m_dafile(ref cblock, buffer, size, file_name, frame);

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

引数：

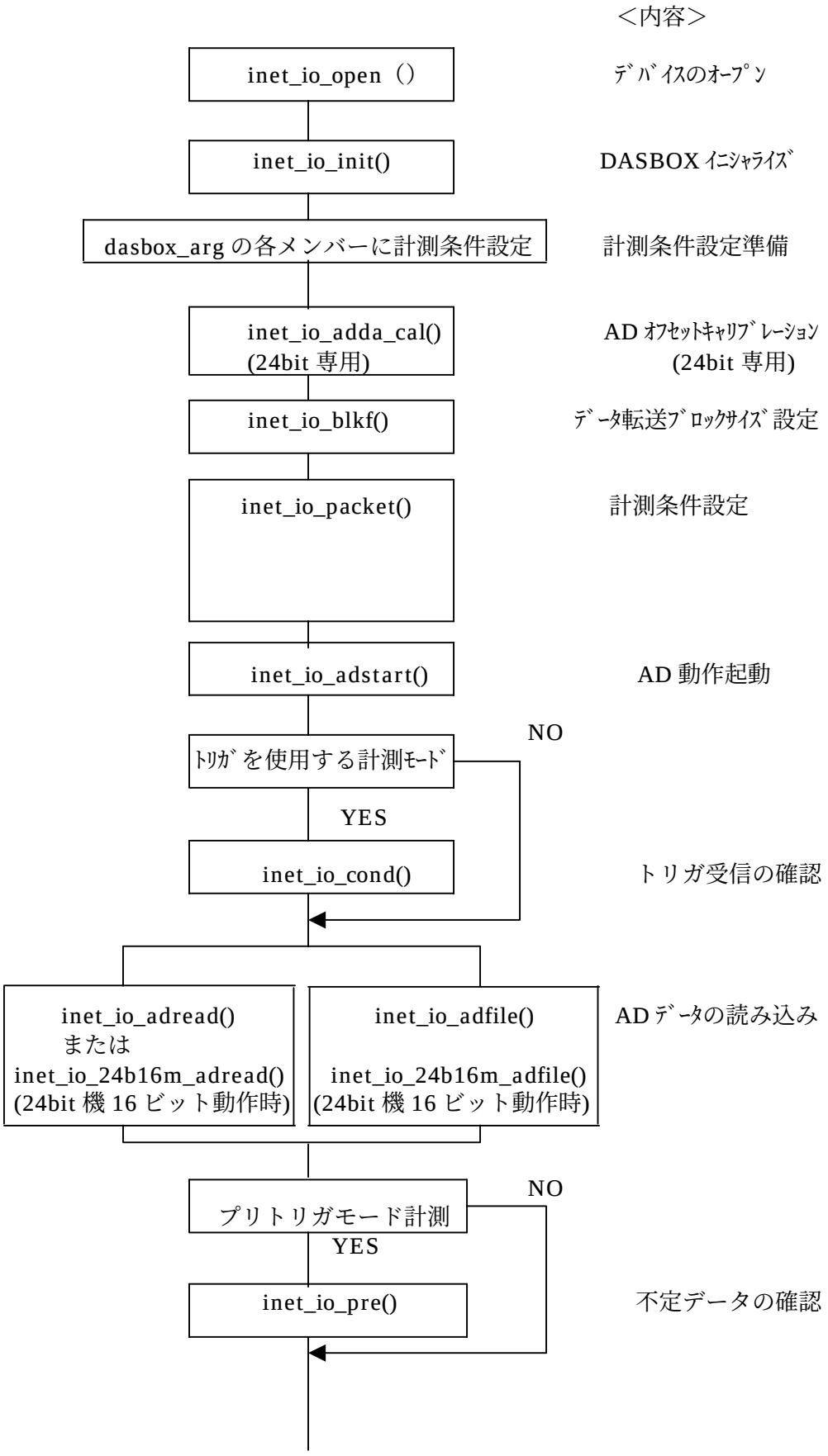
| | |
|---|---|
| cblock | コントロールブロックのポインタ |
| buffer | テンポラリバッファのポインタ<br>Cではバッファを2次元配列とした場合、配列要素は[チャネル数][フレーム数]となります。<br>C#ではバッファは2次元配列で生成します。配列要素は[フレーム数][チャネル数]となります。 |
| size | テンポラリバッファのサイズ(16bitデータ） |
| file_name | 読み込むファイル名（キャラクタのポインタ） |
| frame | トータル読み込み数（16bitデータ） |

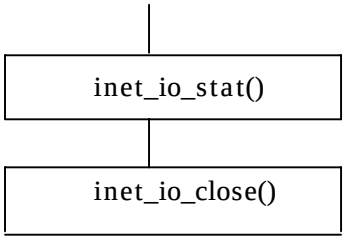
戻り値：

| | |
|---|---|
| 0 | 正常終了 |
| -1 | 異常終了 |

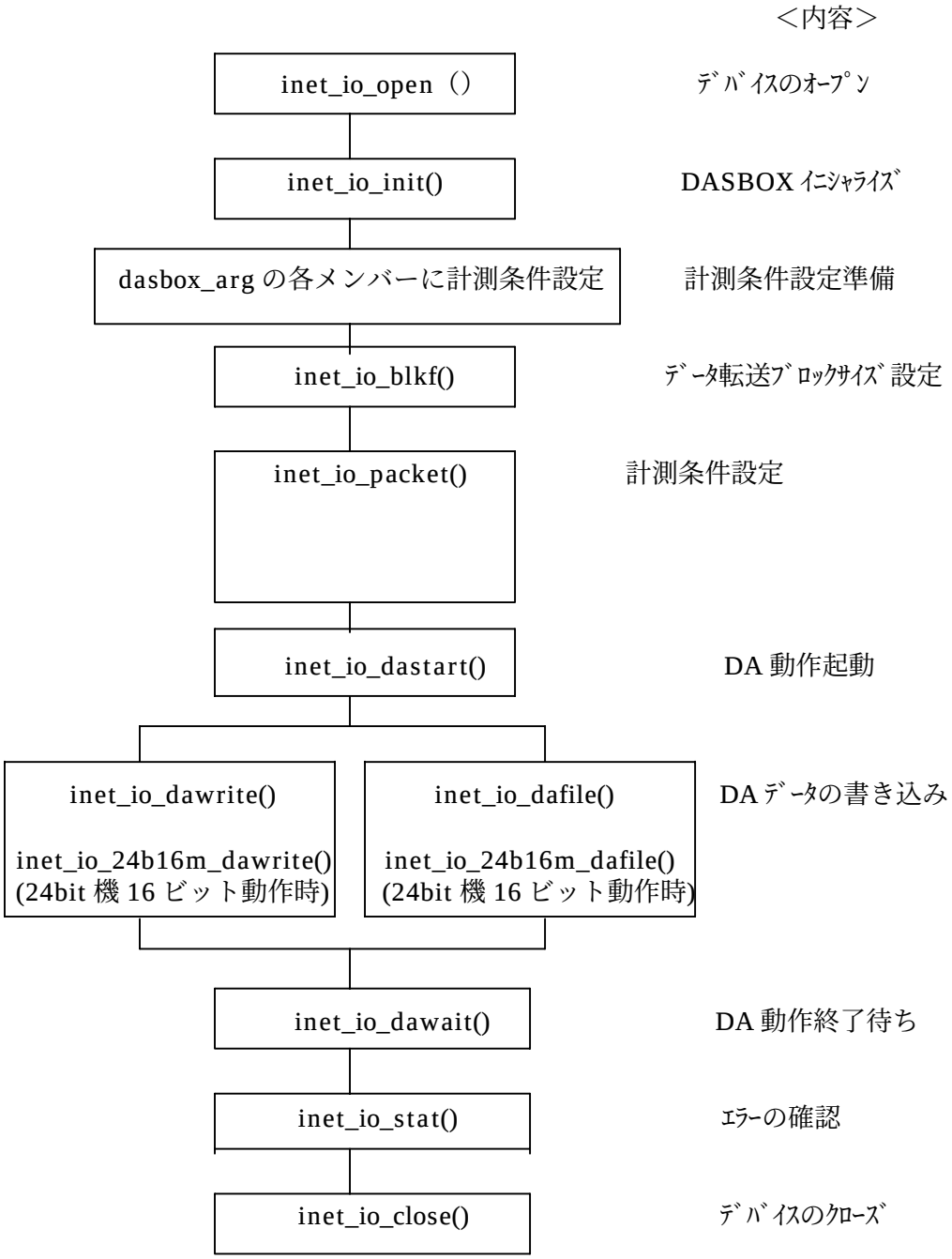
## 3.4 サブルーチン実行基本フロー

### 3.4.1 ＡＤ動作




inet_io_stat()
エラーの確認
inet_io_close()
デバイスのクローズ

### 3.4.2 DA 動作


＜内容＞
inet_io_open（）
ﾃﾞﾊﾞｲｽのｵｰﾌﾟﾝ
inet_io_init()
DASBOX ｲﾆｼｬﾗｲｽﾞ
dasbox_arg の各メンバーに計測条件設定
計測条件設定準備
inet_io_blkf()
ﾃﾞｰﾀ転送ﾌﾞﾛｯｸｻｲｽﾞ設定
inet_io_packet()
計測条件設定
inet_io_dastart()
DA 動作起動
inet_io_dawrite()
inet_io_24b16m_dawrite()
(24bit 機 16 ビット動作時)
inet_io_dafile()
inet_io_24b16m_dafile()
(24bit 機 16 ビット動作時)
DA ﾃﾞｰﾀの書き込み
inet_io_dawait()
DA 動作終了待ち
inet_io_stat()
ｴﾗｰの確認
inet_io_close()
ﾃﾞﾊﾞｲｽのｸﾛｰｽﾞ

# 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関数及びinet_io_packet24関数を使用して行います。
inet_io_packet関数及びnet_io_packet24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。<br>但し、基本サブルーチンのVer2.03からの対応となります。<br>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_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

TRG　IN 端子入力の電圧レベルを指定します。

| Define名 | 値 | TRGINレベル |
|---|---|---|
| なし | 0 | ±５V入力 |
| GAINS | 0x200 | ±１０V入力 |

＊）24 ビット機の 16 ビットデータモード時で
inet_io_packet（）関数にて、引数 bit16 を“１”に指定した場合は、
チャンネル共通のゲインを下記の設定で行います。TRGIN 端子の電圧レベルは±５V 固定となります。

| 値 | 倍率 | 入力電圧レンジ（標準） |
|---|---|---|
| 0 | 1 | ±5V |
| 1 | 2 | ±2.5V |
| 2 | 4 | ±1.25V |
| 3 | 8 | ±0.625V |
| 4 | １６ | ±0.3125V |

### 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 |

注１） ２４ビット機の場合は、値０も内部クロックとなります。

### 4-8 clock

・24ビット機以外のinet_io_packet()関数の場合

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　　０．０３Hz～最大サンプリング周波数
＜外部分周＞
　設定範囲　　２～６５５３６

・24ビット機のinet_io_packet()関数の場合

内部クロックを指定した場合有効で、周波数をHz単位で指定します。
また、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。AD/DAはオーバーサンプリング方式のため、内部クロックベース（25.6MHz,24.576MHz,22.5792MHz、26.2144MHz，20.48MHz）を分周して、サンプリングクロックの256倍（又は128倍）のクロックを作成します。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。

AD動作時
＜内部クロック＞
・ DASmini-E2000　Model-24**-AD及びDASBOX-E　AI32-24/100Mﾓｼﾞｭｰﾙの場合は
　　設定範囲　　400Hz～最大サンプリング周波数（102.4kHz）
・DASBOX-E　AI32-24/20Mﾓｼﾞｭｰﾙの場合は
　　設定範囲　　100Hz～最大サンプリング周波数（20kHz）

DA動作時
＜内部クロック＞
　設定範囲　　8ｋHz～最大サンプリング周波数（204.8kHz）

### 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、ADRET,ADRPOST,DARET,DACYC,DATCYC,DARCYCモード以外で“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　　０～４２９４９６７２９５

### 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 ＜ １６７７７７２１６

### 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

DACYC,DATCYC,DARCYC 等の 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。“０”を設定すると無限に繰り返します。
その他のモードは常に“１”を設定してください。DARCYCモードの場合のトリガの回数もframe4で設定します。
　設定範囲　　０～６５５３６

### 4-13 frame5

modeがADRET、ADREPOST、DARETのときトリガ回数を指定します。
“０”を設定すると無限に繰り返します。
設定範囲　０～６５５３６

### 4-14 mutelevel（24 ビット機種の inet_io_packet 時のみ有効）

24 ビット機種以外の inet_io_packet（）関数の場合は、“０”を設定してください。

24 ビット機種の inet_io_packet（）関数の場合で、DASmini-E2000 及び DASBOX-E シリーズの AI32-24/100M ﾓｼﾞｭｰﾙの場合は AD 入力段の HPF の ON/OFF 設定を行います。AI32-24/20M ﾓｼﾞｭｰﾙはこの機能はありません。

| 値 | |
|---|---|
| 0 | HPF が OFF で DC からの計測を行う。 |
| 1 | HPF が ON で DC 成分をカットする。 |

### 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 |
|---|---|
| 1～1024 | 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。

注）外部クロックモードで動作させ、計測中に指定サイズに達しない状態で外部クロックが停止した場合、強制終了にて動作を停止する必要があり、奇数チャンネル計測の場合のみ、不具合が発生する可能性がありますので、偶数チャンネルでの計測を推奨いたします。

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（24 ビット機種の inet_io_packet 時のみ有効）

24 ビット機種以外の inet_io_packet（）関数の場合は、 不定でも問題なし
24 ビット機種 inet_io_packet（）関数の場合は、
下記の設定を行います。

| 値 | Master/Slave |
|---|---|
| 0 | スレーブモードとなり、マスターからクロックを供給します。 |
| 1 | マスターモードとなり、スレーブにクロックを供給します。<br>１台で使用する場合は、マスター設定としてください。 |

注）コンパイルオプション COMPATI_24b5x0 が使われている場合は、master=1 に自動的に設定されています。

### 4-21　ext_parm（24 ビット機種の inet_io_packet 時のみ有効）

DASmini-E2000 の 24 ビット機種の場合有効です。24 ビット機種の DA/AD の同期をとる場合に DA（マスター側）の同期クロックを 1:1 で出力するか 1/2 分周して出力するかの指定を行います。
詳しくは、24 ビット機種の AD/DA 同期説明.pdf を参照してください。
DASBOX－E シリーズでは、このメンバーは使用しません。

| 値 | 同期クロック出力 |
|---|---|
| 0 | 1:1（標準） |
| 1 | 1/2 |

### 4-22　dmasize,pre_mem_size

未使用
DASBOX－E シリーズ及び DASmini-E2000 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/floopy/kit.tar
```

Windowsの場合

適当なディレクトリ（例えばD:\COMEX）を作成して提供メディアから\COMEX
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*.**/
    doc/  inetlib/  sample/  sample24/  sample24_ADC1271/
  samplevc/  daswave/  /samplec#/  sample24c#/
```

また各ディレクトリには以下のファイルがあります。

```
doc/      ・・・  ドキュメントディレクトリ

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
    sadtp24.h           ・・・ユーザープログラムインクルードファイル
    16C#lib/・・・16bit機種専用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
    24C#lib/・・・24bit機種専用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/ ・・・１６ビット装置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 |
| BSDmakefile | ・・・FreeBSD5.4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 |
| das.exe | ・・・Windows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)
(12ビット、16ビット機種用サンプルディレクトリ)

| | |
|---|---|
| sample.dsw | ・・・Visual Studio6.0 プロジェクトファイル |
| sample.sln | ・・・Visual Studio2008 プロジェクトファイル |
| apl90.c | ・・・サンプルソフトソースファイル |
| das.exe | ・・・Windowsサンプルソフト実行ファイル |

samplec#/ ・・・１６ビット装置 C# 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_ad/ | ・・・AD サンプルプロジェクトディレクトリ |
| sample_da/ | ・・・DAサンプルプロジェクトディレクトリ |

daswave_3/ ・・・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）
(12ビット、16ビット機種用サンプルディレクトリ)

| | |
|---|---|
| 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 |

sample24/ ・・・２４ビット装置サンプルソフトディレクトリ
DASmini-E2000の24ﾋﾞｯﾄ機種及びDASBOX-EシリーズのAI32-24/100Mﾓｼﾞｭｰﾙの場合

| | |
|---|---|
| Makefile | ・・・UNIX(LINUIX)用サンプルソフトメイクファイル |
| BSDmakefile | ・・・FreeBSD5.4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 |
| mk24.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 |
| das.exe | ・・・Windowsサンプルソフト実行ファイル |

sample24vc/ ・・・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)
(24ビット機種用サンプルディレクトリ)

sample.dsw ・・・Visual Studio6.0 プロジェクトファイル
sample.sln ・・・Visual Studio2008 プロジェクトファイル
apl90.c ・・・サンプルソフトソースファイル
das.exe ・・・Windowsサンプルソフト実行ファイル

sample24c#/ ・・・２４ビット装置 C# 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_ad/ ・・・AD サンプルプロジェクトディレクトリ
sample_da/ ・・・DAサンプルプロジェクトディレクトリ

sample24_ADC1271/ ・・・２４ビット装置サンプルソフトディレクトリ
DASBOX-EシリーズのAI32-24/20Mﾓｼﾞｭｰﾙ（ADC1271を使用）の場合

Makefile ・・・UNIX(LINUIX)用サンプルソフトメイクファイル
BSDmakefile ・・・FreeBSD5.4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
mk24.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
das.exe ・・・Windowsサンプルソフト実行ファイル

sample24_ad-da_sync/ ・・・２４ビット装置サンプルソフトディレクトリ
24ビットのAD機種とDA機種を同期させて動作させる場合のサンプル
詳細は24ビットAD／DA同期説明.pdfを参照の事。

## 5-3 ユーザープログラムへの組み込み

◎24ビット機種以外の場合

DASBOX-Eシリーズ(DASminiE2000)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"をコンパイル･リンクしてください。
UNIXのBigエンディアンOS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。
Windows(C)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。mk.cmdは32bit,64bit共通です。
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16C#lib/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16C#lib/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。

◎24ビット機種の場合B24、COMPATI_24b5x0、BIG,ADC1271のコンパイルオプション（またはdefine）があります。

B24 ： 24ビット機種の場合必ず必要なものです。
COMPATI_24b5x0 ： 従来機種DASBOX Model-5x0との互換性を重視した対応がなされていますが、全てを保証するものではありません。またこの場合32ビットデータモードのみの機能となり、マスタ（master=1）の動作に自動設定されます。
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。
ADC1271： 24ﾋﾞｯﾄ機種で、DASBOX-EシリーズのAI32-24/20Mﾓｼﾞｭｰﾙを使用する場合に必要です。
WINNT ： WindowsNT、及びWindows2000、WindowsXP,Windows7で使用する場合必要です。
サンプルのmk24.cmdを参照して下さい。
２４ビット機以外のときと同様にwsock32.libとuser32.libをリンクして下さい。

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24C#lib/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24C#lib/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。

## 5-4 コマンドベースの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。 |
| INF２ | 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。 |
| ADRETrg | 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。 |
| DAREtrg | DAリトリガスタートモードを起動します。 |
| DACyc | DAサイクルスタートモードを起動します。 |
| DATCyc | DAトリガサイクルスタートモードを起動します。 |
| DARCyc | DAリトリガサイクル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。 |
| CALiblation | オフセットのキャリブレーションをブロック単位で行います。 |
| COUnt | カウンタの“０”クリア及びエンコーダのモードを設定します。 |
| ADCal24 | 24ﾋﾞｯﾄADのキャリブレーションを実行します。 |
| MUTe24 | 24ﾋﾞｯﾄDAのミュート動作を実行します。 |
| SYNc24 | 24ﾋﾞｯﾄAD及びDAのクロック同期動作を実行します。 |
| ADTEst | DASBOXアージュメントに固定値を書き込み、AD動作を実行します。 |
| DATEst | DASBOXアージュメントに固定値を書き込み、DA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、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,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又はチャンネルトリガ機能がない機種は必ず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。

⑤　トリガモードを指定した場合出力されます。

[1]　trigger gain ±5V
[2]　trigger gain ±10V

Select　no.

トリガの入力レベルの指定をします。±5V を指定する場合 1、 ±10V を指定する場合 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

⑥-1　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。

⑥-2　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。

⑥-3　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。

⑥-4　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。

⑥-５　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。

⑥-６　DACYC,DATCYC モードを指定した場合出力されます。

Input cycle counter

DA にて同じデータを繰り返す場合の回数指定をします。

⑥-７　DARET,DARCYCL モードを指定した場合出力されます。

Input re-trg counter

トリガの回数指定をします。

⑦　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。

⑧ AD 及び DA 共に出力されます。

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。

## 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

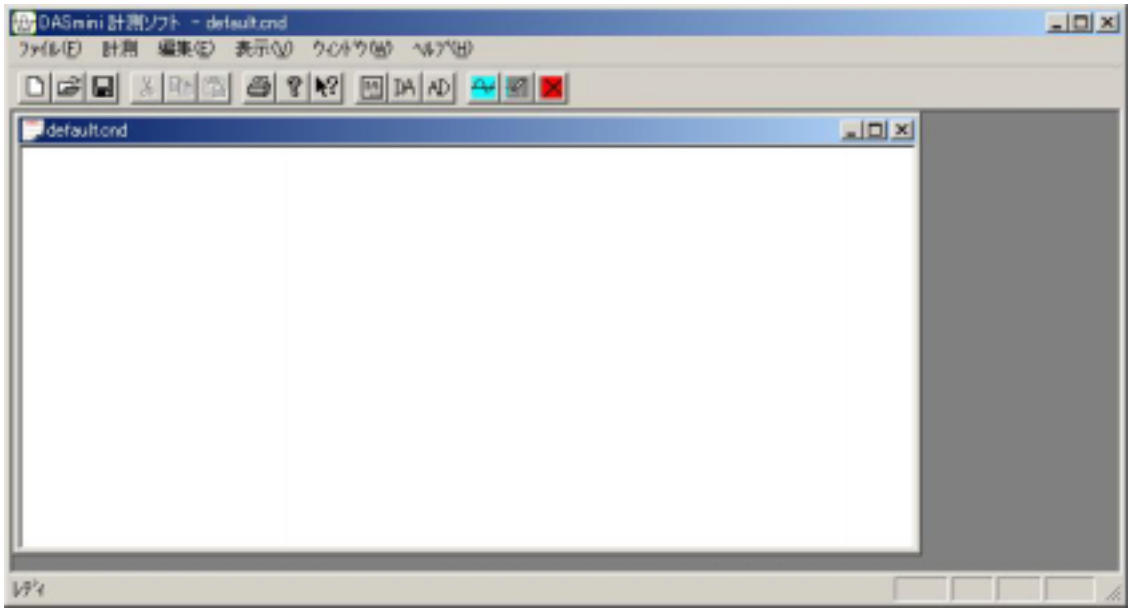
本ソフトはMicrosoftVisualC++で作られた計測サンプルソフトです。本サンプルソフトでは基本サブルーチンを呼び出し、DASmini及びDASBOXに対して制御を行い、データ計測、ファイル化、モニター、DA出力などの操作ができます。

5-5-2)ソフト仕様

１）動作環境
WindowsNT4.0、Windows2000、WindowsXP,Windows7
DASmini-E2000及びDASBOX-Eシリーズ
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

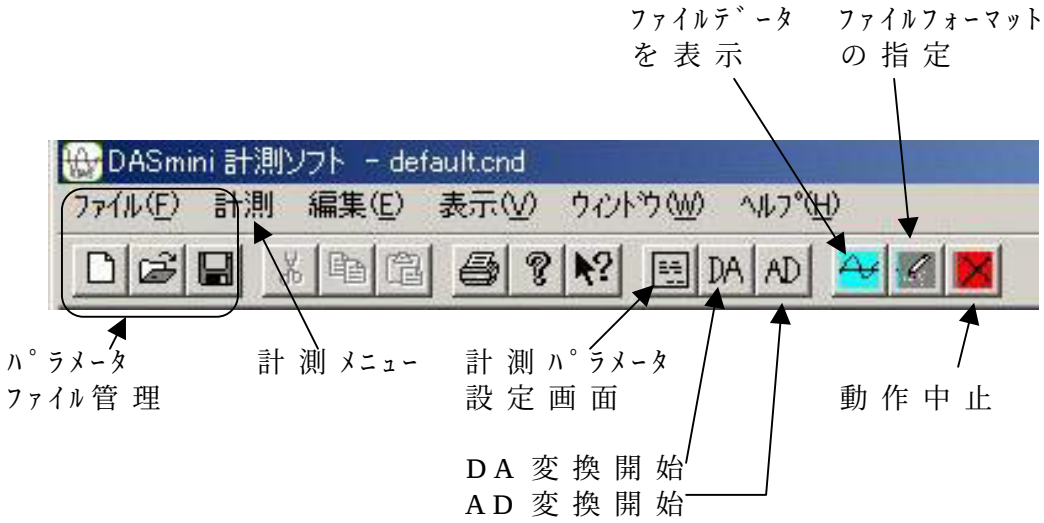
5-5-３)操作説明

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。.

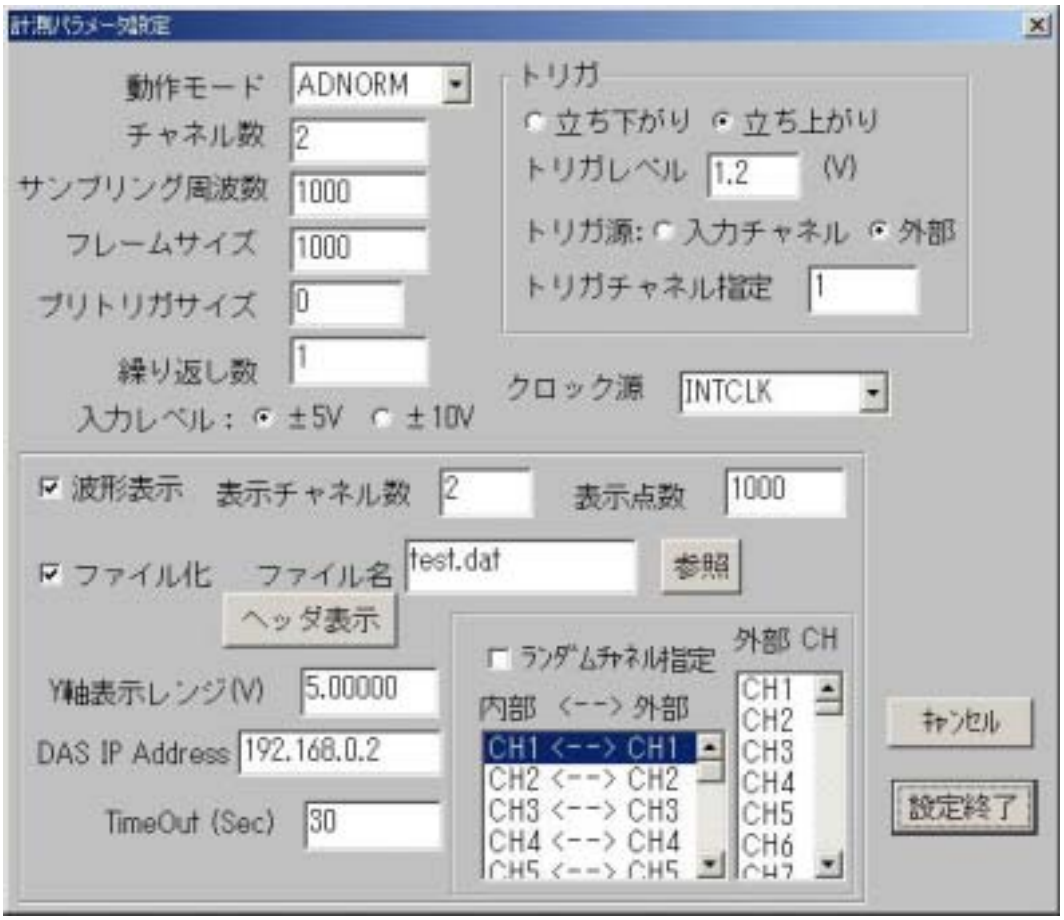



daswaveのメイン画面

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。

a-5)ＤＡＮＯＲＭ（ＤＡ 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ＤＡスタートコマンドにより、ホストコンピュータからのデータ転送を可能にします。ＤＡの出力開始は、ＤＡスタートサイズ(本ソフトウェアではフレームサイズが標準で設定されます。）の値により決定されます。ホストコンピュータからのＤＡデータの転送量がＤＡスタートサイズ×チャネル数に達した時からＤＡの出力を開始します。
ＤＡ出力するデータ数(ホスト転送数)は、フレームサイズ×チャネル数 になります。

a-6)ＤＡＴＲＧ（ＤＡ 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ＤＡスタートコマンドにより、ホストコンピュータからのデータ転送を可能にします。ＤＡの出力開始は、ＤＡスタートサイズ（本ソフトウェアではフレームサイズが標準で設定されます。）の値だけ、データをホストコンピュータより取り込み、外部からのトリガ信号待の状態（Trigger LED 緑点灯）になり、その後、トリガ信号
を検出すると（Trigger LED 消灯)、ＤＡ出力を開始します。ＤＡの出力開始を外部と同期を取る必要がある場合に使用します。
ＤＡ出力するデータ数（ホスト転送数）は、フレームサイズ×チャネル数になります。

a-6)ＤＡＲＥＴ（ＤＡ 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スタートしますが、１フ

レーム（フレームサイズ分）動作が終了すると、再度トリガ信号待の状態になりＤＡ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。ＤＡスタートサイズは初回のトリガ信号待ちの時有効で、２回目以降は意味を持ちません。ＤＡ出力するデータ数（ホスト転送数）は、フレームサイズ×チャネル数×繰り返し数になります。

a-7)ＤＡＣＹＣＬ（ＤＡ ﾉﾝﾄﾘｶﾞｽﾀｰﾄｻｲｸﾙﾓｰﾄﾞ）

このモードは、転送されたＤＡデータ（フレームサイズ×チャネル数）を繰り返し数で設定された回数分、繰り返し出力します。途中で、出力を停止したい場合は、動作停止を使用します。但し、停止する位置は各フレームの区切り目となります。このモードはあるパターンデータを繰り返し出力したい場合に有効です。

ＤＡ出力するデータ数（ホスト転送数）は、フレームサイズ×チャネル数で、このデータを繰り返し数分繰り返します。

但し、フレームサイズには以下の制限があります。

フレームサイズ×チャネル数　≦　DAS メモリ（標準 4MW）容量

a-8)ＤＡＴＣＹＣＬ（ＤＡ ﾉｰﾏﾙﾄﾘｶﾞｽﾀｰﾄｻｲｸﾙﾓｰﾄﾞ）

このモードは a-7)のモード動作をトリガ受信後実行します。

トリガ信号は初回のみ有効で、２回目以降は意味を持ちません。

ＤＡ出力するデータ数（ホスト転送数）は、フレームサイズ×チャネル数で、このデータを繰り返し数分繰り返します。

a-9)ＤＡＲＣＹＣＬ（ＤＡ ﾘﾄﾘｶﾞｽﾀｰﾄｻｲｸﾙﾓｰﾄﾞ）

このモードは a-6)のモード動作を同じデータで繰り返し数分実行します。

ＤＡ出力するデータ数（ホスト転送数）は、フレームサイズ×チャネル数で、このデータをトリガ毎に出力します。

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」、「DACYCL」、「DATCYCL」、「DARCYCL」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。

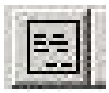
t) 設定終了

「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。


a)DA スタート

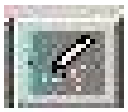
計測パラメータで DA 変換を行います。パラメータの動作モードが DA モードではない場合は、自動的に DANORM モードに変更します。（パラメータ設定画面には反映されません。）


b)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。）

c)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。



d)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

e) 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

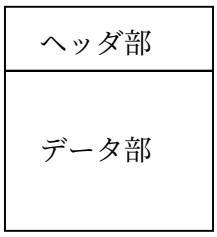
・RESET： DASBOX のイニシャライズ（inet_io_init()）を実行します。

・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<br>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 |

７）　サンプルデータファイル

実際に AD で取り込んだデータファイルです。

test2-5000.dat　：計測サンプルソフトDA出力用ファイル（ADで作成したデータファイル）

AD計測条件

計測モード＝ADTRG、トリガレベル=1.2V、トリガスロープ＝立ち上がり

トリガ入力＝AD1CH信号

チャンネル数＝２、サンプリング周波数=1000Hz、フレームサイズ=5000

AD1CH入力：　周波数＝１Hz、電圧レベル＝±４V、波形＝Sin波

AD2CH入力：　周波数＝１Hz、電圧レベル＝０－４V、波形＝方形波（duty=50%)

test8-10000.dat　：計測サンプルソフトDA出力用ファイル（ADで作成したデータファイル）

AD計測条件

計測モード＝ADTRG、トリガレベル=1.2V、トリガスロープ＝立ち上がり

トリガ入力＝AD1CH信号

チャンネル数＝８、サンプリング周波数=10000Hz、フレームサイズ=10000

AD1CH入力：　周波数＝１Hz、電圧レベル＝±４V、波形＝Sin波

AD2CH入力：　周波数＝１Hz、電圧レベル＝０－４V、波形＝方形波（duty=50%)

rtest2-3000.dat　：計測サンプルソフトDA出力用ファイル（ADで作成したデータファイル）

AD計測条件

計測モード＝ADRET、トリガレベル=1.2V、トリガスロープ＝立ち上がり

トリガ入力＝AD1CH信号

チャンネル数＝2、サンプリング周波数=1000Hz、フレームサイズ=1000

リトリガ回数＝３

AD1CH入力：1回目　周波数＝１Hz、電圧レベル＝±４V、波形＝正弦波

2回目　周波数＝１Hz、電圧レベル＝±４V、波形＝三角波

3回目　周波数＝１Hz、電圧レベル＝±４V、波形＝方形波

AD2CH入力：　周波数＝１Hz、電圧レベル＝０－４V、波形＝方形波（duty=50%）