ＡＤＳＰ３２４－１４５

マルチファンクションボード
ソフトウェア・ユーザーズ・マニュアル

中部電機株式会社

目　次

１．はじめに

本マニュアルは、弊社製ＡＤＳＰ３２４－００Ａ／ＡＤＳＰ６７４－００シリーズにて、マルチファンクションＩ／Ｏボード・ＡＤＳＰ３２４－１４５のソフトウェア開発用マニュアルです。記述内容はＡＤＳＰ６７４－００シリーズを標準として書かれています。ＡＤＳＰ３２４－００Ａにてご使用の際は以下の様に読みかえてご使用ください。

ファイル名	A145_67xxxx.x	→ A145_32xxx.x
関数名	A67X_xxxxx	→ A32X_xxxx
推奨ベースアドレス	0x03004000	→ 0x00901000

＊　一部この読み替えと異なる場合があります。各章を参照ください。

２．概要

ＡＤＳＰ３２４－１４５サポートソフトウェアは、マルチファンクションＩ／Ｏボード・ＡＤＳＰ３２４－１４５を使用するための基本機能を含んだＢＩＯＳプログラム及び、それを用いたサンプルプログラムから構成されています。

ＢＩＯＳプログラムはＣ言語で書かれているため実際の使用には速度的に問題がありますが、ＡＤＳＰ３２４－１４５を動作させる上で大きなヒントとなると思われます。

また、A67X_145init()関数実行後、指定したベースアドレスとボード枚数より以下の変数が参照可能です。プログラム作成時にご活用ください。

a145_VECT_PORT	全割込みボードフラグのアドレス
a145_Port	ボードのベーズアドレス

３．機能一覧

ＢＩＯＳプログラムには次の機能があります。

① ＡＤＳＰ３２４－１４５ボード初期化
② ウォッチドグタイマ機能
③ カウンタデータ入力・プリセット機能
④ パルス出力機能
⑤ PWM出力機能
⑥ キャプチャ機能
⑦ デジタル入出力機能

４．供給形態

ＢＩＯＳプログラムは、ソースファイル及びライブラリ形式で供給されます。A145_67bios.h(ADSP324-00A 時は A145bios.h)と A145_67bios.lib を C6X_C_DIR（ADSP324-00A 時は C_DIR）環境変数の示すフォルダへコピーしておけば容易に利用することができます。

５．供給ファイル一覧

Readme.txt	A145_67bios の簡単な説明が書かれています。
A145_67bios.c	A145_67bios のソースファイル
A145_67bios.h	A145_67bios を使用するためのヘッダファイル (ADSP324-00A 時 A145bios.h)
A145_67bios.obj	A145_67bios のオブジェクトファイル
A145_67bios.lib	A145_67bios のライブラリファイル
A145_67bios.cmd	A145_67bios を利用するためのコマンドファイル
Common_145.h	A145_67bios のヘッダファイル
Common_145def.h	ADSP324-145 ポートのデータ定義用ヘッダファイル 詳細は、“７データ定義について”を参照してください。
Smplxx.c	ADSP674-00 ｼﾘｰｽﾞ用サンプルプログラム
Smplxx.cmd	ADSP674-00 ｼﾘｰｽﾞ用サンプルコマンドファイル
Smplxx.out	ADSP674-00 ｼﾘｰｽﾞ用サンプル実行ファイル
Timer.c	ADSP674-00 ｼﾘｰｽﾞ用タイマプログラム
Samplex.c	ADSP324-00A 用サンプルプログラム
Samplex.lnk	ADSP324-00A 用サンプルリンカーファイル
Samplex.bat	ADSP324-00A 用サンプルコンパイル用バッチファイル
Samplex.out	ADSP324-00A 用サンプル実行ファイル
Timsub.c	ADSP324-00A 用タイマプログラム

６．実装方法

６.１ ＡＤＳＰ３２４－００Ａ用

６.１.１ ＣＯＦＦファイル形式のオブジェクトリンク

カレント・ディレクトリに A145_32bios.c、Timsub.c をコピーして、ユーザ・プログラムとリンクしてください。**コンパイルオプションに ”IS_324”を必ず定義してください。**

６.１.２ ライブラリリンク

環境変数 C_DIR の示すディレクトリに、A145bios.h /Common_145.h/Common_145def.h と A145_32bios.lib をコピーすることにより、リンカの -l オプションでライブラリを指定してリンクして下さい。環境変数 C_DIR の内容を、確認します。

C_DIR = C:\c30\tic\include : C:\c30\tic\lib... <Ver5.1 の場合>

通常は、DSP の C コンパイラのライブラリが格納されているパスがセットされています。

上記の場合は

A145_32bios.h を　C:\c30\tic\include

A145_32bios.lib を C:\c30\tic\lib

へコピーします。

プログラムのリンク時に A145_32bios.lib をリンクして下さい。

なお、C_DIR 環境変数の設定については、TI の C コンパイラの取り扱い説明書を参照して下さい。

６.２ ＡＤＳＰ６７４－００シリーズ用

６.２.１ ＣＯＦＦファイル形式のオブジェクトリンク

カレント・ディレクトリに A145_67bios.c、Timer.c をコピーして、ユーザ・プログラムとリンクしてください。**コンパイルオプションに ”IS_674”を必ず定義してください。**

A145_67bios.c、Timer.c をプロジェクトに追加します。

「Project」-「Add Files to Project」

A145_67bios.c、Timer.c を選択 → 開く

６.２.２ ライブラリリンク

環境変数 C6x_C_DIR の示すディレクトリに、A145_67bios.h /Common_145.h/Common_145def.h と A145_67bios.lib をコピーすることにより、リンカの -l オプションでライブラリを指定してリンクして下さい。環境変数 C6x_C_DIR の内容を、確認します。

C6x_C_DIR = C:\ti\c6000\cgtools\include : C:\ti\c6000\cgtools\lib...

通常は、DSP の C コンパイラのライブラリが格納されているパスがセットされています。

上記の場合は

A145_67bios.h を　C:\ti\c6000\cgtools\include

A145_67bios.lib を C:\ti\c6000\cgtools\lib

へコピーします。

プログラムのリンク時に A145_67bios.lib をリンクして下さい。

なお、C6x_C_DIR 環境変数の設定については、TI の C コンパイラの取り扱い説明書を参照して下さい。

7．データ定義について

ADSP324-145 ポートのデータ定義は、<Common_145def.h>の中で以下のように定義されています。データの詳細については、“ＡＤＳＰ３２４－１４５　ハードウェア・ユーザーズ・マニュアル”を参照してください。

1) ポート定義

```
typedef     struct      {
            A145tm              tm;         // ｳｫｯﾁﾄﾞｸﾞﾀｲﾏ
            A145dg              dg1;        // Digital I/O  CN12
            union body1{
                    A145pwm     data;       // ＰＷＭ
                    A145capt    capt;       // キャプチャ
            } a;
            A145dg              dg2;        // Digital I/O CN13
            union body2{
                    A145pls     pls;        // パルス出力
                    A145cnt     cnt;        // カウンタ入力
            } b;
            unsigned int        dumy145a[16];
            unsigned int        dumy145b[128];
            A145_GOT_COMM       got;        // ＧＯＴ制御用
} A145_PORT;
```

2) 共通エリア

```
struct timer_tag{
        unsigned int    dumy1[10];
        int             cntrol;             // コントロール
        unsigned int    reset;              // カウンタリセット
        unsigned int    intcnt;             // 割込みコントロール
        unsigned int    intflg;             // 割込みポートフラグ
        unsigned int    dumy2[2];
};
typedef struct timer_tag        A145tm;
```

３)デジタル入出力

```
struct digital_tag{
        unsigned int    odata[3];               // 出力データ
        unsigned int    idata[3];               // 入力データ
        unsigned int    dmy1[2];
        unsigned int    hold[2];                // ホールド要求
        unsigned int    status;                 // ﾊﾝﾄﾞﾘﾝｸﾞ信号状態
        unsigned int    dmy2;
        unsigned int    method;                 // ﾊﾝﾄﾞﾘﾝｸﾞ方式
        unsigned int    flg;                    // ﾊﾝﾄﾞﾘﾝｸﾞ極性
        unsigned int    oncoef;                 // ﾊﾝﾄﾞﾘﾝｸﾞ ON 時間
        unsigned int    delcoef;                // ﾊﾝﾄﾞﾘﾝｸﾞ遅延時間
};
typedef struct digital_tag      A145dg;
```

４) ＰＷＭ

```
struct pport_tag {
        unsigned int    freq;                   // ｷｬﾘｱ周波数
        int             u_duty;                 // Ｕ相 Ｄｕｔｙ
        int             v_duty;                 // Ｖ相 Ｄｕｔｙ
        int             w_duty;                 // Ｗ相 Ｄｕｔｙ
        unsigned int    bwrite;                 // 一括書込み
};
typedef struct pport_tag                pport;

struct pwm_tag {
        pport           port[3];
        unsigned int    dumy1;
        unsigned int    freq_3;
        int             u_duty_3;
        int             v_duty_3;
        unsigned int    bwrite_3;
        unsigned int    dtime[4];
        unsigned int    dumy3[4];
        unsigned int    outenable;
        unsigned int    reflect;
        unsigned int    dumy4[2];
} ;
typedef struct pwm_tag          A145pwm;
```

5)キャプチャ

```
struct cport_tag{
        int             period;                 // 周期
        int             uvon;                   // ＵＶ ＯＮ時間
        int             vwon;                   // ＶＷ ＯＮ時間
        int             wuon;                   // ＷＵ ＯＮ時間
    };
    typedef struct cport_tag                cport;

    struct capture_tag{
        cport           port[4];
        unsigned int    order[4];               // 計測スタート指令
        unsigned int    flg;                    // 計測完了フラグ(0:未完了、1:完了)
        unsigned int    dumy1[7];
        unsigned int    mode;                   // モード（0:３相、1:単相）
        unsigned int    dumy2[3];
    };
    typedef struct capture_tag      A145capt;
```

6)パルス出力

```
struct lport_tag{
        unsigned int    period;                 // 周期
        unsigned int    direct;                 // 回転方向(1:ＣＷ、2:ＣＣＷ)
        unsigned int    wbat;
        unsigned int    dumy0;
};
typedef struct lport_tag    lport;

struct pulse_tag{
        lport           port[4];
        unsigned int    dumy1[4];
        unsigned int    num[4];                 // １回転のパルス数
        unsigned int    dumy2[4];
        unsigned int    enable;                 // 出力許可
        unsigned int    reflect;
        unsigned int    dumy3[2];
};
typedef struct pulse_tag    A145pls;
```

７)カウンタ

```
struct counter_tag{
            unsigned int    count[4];           // カウンタ値
            unsigned int    dumy1[4];
            unsigned int    preset[4];          // プリセット
            unsigned int    reset;              // リセット許可
            unsigned int    select;             // 逓倍選択
            unsigned int    dumy2[18];
};
typedef struct counter_tag  A145cnt;
```

８．関数一覧

○Ｇｅｎｅｒａｌ関数

○ウォッチドグタイマ

○デジタル入出力

＜３２ビット用＞

＜１６ビット用＞

９．関数リファレンス（アルファベット順）

９．１ A67X_145captfread　　(キャプチャ計測完了フラグ読み出し)

【機能】

キャプチャ時、各チャネルの計測が完了したかを調べるために使用します。計測未完了時は０、完了時は１が戻ります。

【書式】

```
int         A67X_145captfread(
                                int bdno,       ボード番号
                                int ch)         チャネル番号
```

【引数】

bdno	A67X_145init で指定した範囲のボード番号０～３の範囲で指定します
ch	チャネル番号を　０～３で指定します。

【戻り値】

０	計測未完了
１	計測完了
９９	引数エラー

【参考】

A67X_145captfreset

９.２ A67X_145captfreset　(キャプチャ計測完了フラグリセット)

【機能】

キャプチャ時使用する、計測完了フラグをリセットします。

【書式】

```
int     A67X_145captfreset(
                        int bdno)          ボード番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145captfread

９.３　A67X_145captinit　　(キャプチャ初期化)

【機能】

キャプチャを使用するにあたってモード設定をします。

【書式】

```
int     A67X_145captinit(
                    int bdno,          ボード番号
                    int mode)          入力モード
```

【引数】

bdno　　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

mode　　０　３相モード

　　１　単相モード

【戻り値】

０　　モード選択完了

－１　　引数エラー

【参考】

A67X_145captstart, A67X_145captread

【使用例】

```
#include   “A145_67bios.h”
#defineBD_MAX   4
#defineBD_BASE  0x03004000

void main()
{
        unsigned int ftime, ontime[4];
        A67X_145init(BD_MAX, BD_BASE );
        A67X_145captinit( 0, 0 );
        A67X_145captstart( 0, 0 );       // 計測ｽﾀｰﾄ
        A67X_145captread1(0,0,&ftime,ontime);
}
```

９．４ A67X_145captread1　　(キャプチャ計測データ読み出し ﾌﾗｸﾞﾁｪｯｸ有り)

９．５ A67X_145captread2　　(キャプチャ計測データ読み出し ﾌﾗｸﾞﾁｪｯｸなし)

【機能】

キャプチャ計測結果、周期時間・ＯＮ時間を読み出します。A67X_145acptread1 は読み出し前に計測完了フラグを判定します。A67X_145captread2 は計測完了フラグを判定しません。A67X_145captread2 を使用時は、前もって A67X_145captfread で計測完了を判定してください。

【書式】

```
int     A67X_145captread1(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float *ftime,      周期時間(nSec)
                    float ontime[3])   ON 時間(nSec)
```

【引数】

bdno	A67X_145init で指定した範囲のボード番号０～３の範囲で指定します
ch	チャネル番号を　０～３で指定します。
ftime	周期ﾃﾞｰﾀを格納するアドレスを指定します。
ontime	ＯＮ時間ﾃﾞｰﾀを格納するアドレスを指定します。

【戻り値】

０	正常終了
－１	引数エラー

【参考】

A67X_145captinit, A67X_145captstrat

９．６ A67X_145captstart　　(キャプチャ計測開始)

【機能】

キャプチャ計測を開始します。

【書式】

```
int     A67X_145captstart(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

ch　　チャネル番号を　０～３で指定します。

【戻り値】

０　　正常スタート

－１　引数エラー

【参考】

A67X_145captinti, A67X_145captread

９．７　A67X_145cntin　　（カウンタ読み出し）

【機能】

カウンタ値を読み出します。

【書式】

```
int     A67X_145cntin(
                    int bdno,       ボード番号
                    int ch,         チャネル番号
                    int *data)      データ格納ﾎﾟｲﾝﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

ch　　チャネル番号を　０～３で指定します。

data　読み出したカウンタデータを格納する領域のポインタを指定します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145cntinit,A67X_145cntout

９.８ A67X_145cntinit　　(カウンタ初期化)

【機能】

カウンタのリセット許可・逓倍選択を設定します。

【書式】

```
int     A67X_145cntinit(
                    int bdno,       ボード番号
                    int ch,         チャネル番号
                    int reset,      リセット許可
                    int mag)        逓倍選択
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

ch　　チャネル番号を　０～３で指定します。

reset　カウンタリセット信号の許可設定を指定します

　　０：未許可

　　１：許可

mag　　カウンタ逓倍を選択します。

　　１：×１、２：×２、４：×４

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145cntin,A67X_145cntout

【使用例】

```
#include   “A145_67bios.h”
#define    BD_MAX    4
#define    BD_BASE   0x03004000
void main()
{
     unsigned int    odata,idata,tdata;
     A67X_145init(BD_MAX, BD_BASE );
     A67X_145cntinit( 0, 0, 0, 1 );
     odata = 71;
     for( i = 0; i < 5; i++ ){
       for( j=0, odata=odata; j<5; j++, odata=odata*279 + 959 ){
          A67X_145out(BOARD_NO, 0, odata);      // Output
          A67X_145in(BOARD_NO, 0, &idata);      // Loop backin
       }
     }
}
```

９．９　A67X_145cntout　　(カウンタデータプリセット)

【機能】

カウンタへデータをプリセットします。

【書式】

```
int     A67X_145cntout(
                    int bdno,       ボード番号
                    int ch,         チャネル番号
                    int data)       プリセットデータ
```

【引数】

bdno	A67X_145init で指定した範囲のボード番号０～３の範囲で指定します
ch	チャネル番号を　０～３で指定します。
data	カウンタへプリセットするデータを指定します

【戻り値】

０	正常終了
－１	引数エラー

【参考】

A67X_145cntin,A67X_145cntin

９.１０　A67X_145init　（ボード初期化）

【機能】

ＡＤＳＰ３２４－１４５ボードの初期化をします。ボードを使用するにあったって、一番最初に必ず実行してください。

【書式】

```
int     A67X_145init(
                int bdnum,              ボード実装枚数
                unsigned int base) ボードベースアドレス
```

【引数】

bdnum　実装ボード枚数を１～４で指定します。未実装のボードを指定するとエラーとなります。

base　ボードアドレスの先頭番地を指定します。以降１枚毎に ADSP674-00 ｼﾘｰｽﾞ時は 0x800、ADSP324-00A 時は 0x200 づつアドレスが増加します。

【戻り値】

０　正常終了

－１　引数エラー又は未実装のボードを指定した。

【参考】

ベースアドレス

ADSP674-00 ｼﾘｰｽﾞ	0x03004000 の時	１枚目	0x03004000
		２枚目	0x03004800
ADSP324-00A	0x00901000 の時	１枚目	0x00901000
		２枚目	0x00901200

【使用例】

```
#include            “A145_67bios.h”
#define  BD_MAX                         4
#define  BD_BASE                        0x03004000
void main()
{
        if(A67X_145init(BD_MAX, BD_BASE ))
          printf(“Initialize Error!\n”);
}
```

９.１１　A67X_145intdefine　(割込み処理関数設定)

【機能】

ADSP324-145 からの割り込み時の実行関数を設定します。この関数を実行後はＩＮＴ７は禁止された状態です。A67X_145intenable()で割込み許可をしてください。

【書式】

```
int     A67X_145intdefine(
                    int bdno,           ボード番号
                    void (*func()))     処理関数ｱﾄﾞﾚｽ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

func　割り込み時に実行する関数アドレスを指定します

【戻り値】

０　割込み未発生

－１　引数エラー

【参考】

A67X_145intset, A67X_145intenable, A67X_145intdisable

９.１２　A67X_145intdisable　(割込み禁止)

【機能】

ＩＮＴ７の割込みを禁止します。この関数を実行すると ADSP324-145 以外の周辺Ｉ／Ｏボードからの割り込みも禁止されます。

【書式】

```
void    A67X_145intdisable(void);
```

【引数】

なし

【戻り値】

なし

【参考】

A67X_145intenable

９．１３　　A67X_145intenable　　(割込み許可)

【機能】

ＩＮＴ７の割込みを許可します。この関数を実行すると ADSP324-145 以外の周辺Ｉ／Ｏボードからの割り込みも許可されます。（ＧＩＥレジスタも許可されます）

【書式】

```
void    A67X_145intenable(void);
```

【引数】

なし

【戻り値】

なし

【参考】

A67X_145intdisable

９.１４　　A67X_145intflg　　(割込み状況チェック)

【機能】

割込みポートフラグレジスタをチェックします。割込み処理内で使用する場合速度的に間に合わない時があります。その時は直接Ｃ言語にて記述してください。またこの関数では割込みフラグをリセットしていませんので、全割込み確認後割込みポートフラグをリセットしてください。（ハードウェアマニュアル参照）

【書式】

```
int     A67X_145intflg(
                    int bdno,       ボード番号
                    int bdflg,       ボート割り込みフラグ
                    int intno)      ポート番号
```

【引数】

bdno　　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

bdflg　　ＤＳＷ１０４で設定する割込みフラグを１～８で指定します

intno　　割込みコントロールのチェックするポートを指定します。

例）ポート０Ｈをチェックする場合

```
intno = EBL_H0;
```

“A145_67bios.h”で定義されている EBL_xx を使用してください。

【戻り値】

０　　割込み未発生

１　　割込み発生

－１　　引数エラー

【参考】

A67X_145intset

９.１５　A67X_145intflgreset　(割込みフラグリセット)

【機能】

割込みポートフラグレジスタと割込みボードフラグレジスタ（該当分）をリセットします。

【書式】

```
int     A67X_145intflgreset(
                    int bdno)       ボード番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
指定します

【戻り値】

０　正常終了
－１　引数エラー

【参考】

A67X_145intset, A67X_145intflg

９.１６　A67X_145intset　(割込みコントロール設定)

【機能】

割込みコントロールレジスタの設定を行います。

【書式】

```
int     A67X_145intset(
                    int bdno,       ボード番号
                    int intno,      ポート番号
                    int intslp)       ﾄﾘｶﾞｽﾛｰﾌﾟ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

intno　割込み許可するポート番号を指定します。
例）ポート０H，ポート１Hを許可する場合

```
intno = EBL_H0 + EBL_H1;
```

intslp　外部割込みのトリガスロープを指定します。
０：↓
１：↑

【戻り値】

０　正常終了
－１　引数エラー

【参考】

A67X_145intflg

９．１７　　A67X_145pioand　　(デジタル出力 反転 AND 32bit)

【機能】

指定ポートへ反転したデータをＡＮＤ出力します。

【書式】

```
int     A67X_145pioand(
                    int bdno,               ボード番号
                    int port,               ポート番号
                    unsigned int data) 反転 AND するﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

data　反転ＡＮＤするデータ

【戻り値】

０　　正常終了

－１引数エラー

【参考】

A67X_145pioinit1, A67X_145pioor

【使用例】

```
#include “A145_67bios.h”
#defineBD_BASE      0x03004000
#defineBD_MAX       4
void main()
{
        A67X_145init( BD_MAX, BD_BASE );
        A67X_145pioinit1( 0, 0);
        A67X_145pioand( 0, 0, 0x222 );
}
```

９.１８　A67X_145pioand16　（デジタル出力 反転 AND 16bit）

【機能】

指定ポートへ反転した 16bit データをＡＮＤ出力します。

【書式】

```
int     A67X_145pioand16(
                    int bdno,           ボード番号
                    int port,           ポート番号
                    int ud,             上位/下位
                    unsigned int data)  反転 AND するﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

ud　　１：上位、２：下位のいずれかを指定します。

data　反転ＡＮＤするデータ

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioor16

９.１９　A67X_145piobreset　（デジタル出力 ﾋﾞｯﾄﾘｾｯﾄ 32bit）

【機能】

指定ポートの指定位置ビットをＯＦＦにします。

【書式】

```
int     A67X_145piobreset(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int bit)        ビット位置
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

bit　ビット位置を０～３１で指定します。

【戻り値】

０　　正常終了

　－１引数エラー

【参考】

A67X_145pioinit1, A67X_145piobset

【使用例】

```
#include “A145_67bios.h”
#defineBD_BASE      0x03004000
#defineBD_MAX       4
void main()
{
        A67X_145init( BD_MAX, BD_BASE );
        A67X_145pioinit1( 0, 0);
        A67X_145piobreset( 0, 0, 1 );
}
```

９．２０　A67X_145piobreset16　（デジタル出力 ﾋﾞｯﾄﾘｾｯﾄ 16bit）

【機能】

指定ポートの指定位置ビットをＯＦＦします。

【書式】

```
int     A67X_145piobreset16(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int ud,         上位/下位
                    int bit)        ビット位置
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

ud　　１：上位、２：下位のいずれかを指定します。

bit　　ビット位置を０～１５で指定します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145piobset

９．２１　A67X_145piobset　（デジタル出力 ﾋﾞｯﾄｾｯﾄ 32bit）

【機能】

指定ポートの指定位置ビットをＯＮにします。

【書式】

```
int     A67X_145piobset(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int bit)        ビット位置
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

bit　ビット位置を０～３１で指定します。

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145piobreset

【使用例】

```
#include “A145_67bios.h”
#defineBD_BASE      0x03004000
#defineBD_MAX       4
void main()
{
        A67X_145init( BD_MAX, BD_BASE );
        A67X_145pioinit1( 0, 0);
        A67X_145piobset( 0, 0, 1 );
}
```

９.２２　A67X_145piobset16　（デジタル出力 ﾋﾞｯﾄｾｯﾄ 16bit）

【機能】

指定ポートの指定位置ビットをＯＮします。

【書式】

```
int     A67X_145piobset16(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int ud,         上位/下位
                    int bit)        ビット位置
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

ud　　１：上位、２：下位のいずれかを指定します。

bit　　ビット位置を０～１５で指定します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145piobreset16

９.２３　A67X_145piohold　(デジタル入力ホールド要求)

【機能】

デジタル入力時、ハンドリング方式がホールド選択されている時に外部回路へのホールド要求信号を出力します。

【書式】

```
int     A67X_145piohold(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int ud)         上位・下位
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を１又は３で指定します。

ud　　１：上位、２：下位のいずれかを指定します。
　　A67X_145pioset にて 16ﾋﾞｯﾄ設定時のみ有効です。

【戻り値】

０　　正常終了

－１　引数エラー又はハンドリング方式がホールド以外に設定

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioholdreset

９.２４　A67X_145pioholdreset　(デジタル入力ホールド要求解除)

【機能】

デジタル入力時、ハンドリング方式がホールド選択されている時に外部回路へのホールド要求信号を解除します。

【書式】

```
int     A67X_145pioholdreset(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int ud)         上位・下位
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port ポート番号を１又は３で指定します。

ud　１：上位、２：下位のいずれかを指定します。
　　A67X_145pioset にて 16ﾋﾞｯﾄ設定時のみ有効です。

【戻り値】

０　正常終了

－１　引数エラー又はハンドリング方式がホールド以外に設定

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145piohold

９.２５　A67X_145pioin　(デジタル入力 32bit)

【機能】

３２bit デジタル入力を行います。

【書式】

```
int     A67X_145pioin(
                int bdno,               ボード番号
                int port,               ポート番号
                unsigned int *data)     ﾃﾞｰﾀ格納ﾎﾟｲﾝﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

port　ポート番号を１又は３で指定します。

data　データを格納するポインタを指定します。

【戻り値】

０　正常終了

－１　引数エラー又はアクセス設定エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioout

９.２６　A67X_145pioin16　（デジタル入力 16bit）

【機能】

１６bit デジタル入力を行います。

【書式】

```
int     A67X_145pioin16(
                    int bdno,           ボード番号
                    int port,           ポート番号
                    int ud,             上位/下位
                    unsigned int *data) ﾃﾞｰﾀ格納ﾎﾟｲﾝﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を１又は３で指定します。

ud　　１：上位、２：下位のいずれかを指定します。

data　データを格納するポインタを指定します。

【戻り値】

０　　正常終了

－１　引数エラー又はアクセス設定エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioout16

９.２７　A67X_145pioinit1　(デジタル入出力初期化<0/1ﾎﾟｰﾄ>)
９.２８　A67X_145pioinit2　(デジタル入出力初期化<2/3ﾎﾟｰﾄ>)

【機能】

デジタル入力時、ハンドリング方式がホールド選択されている時に外部回路へのホールド要求信号を解除します。

【書式】

```
int     A67X_145pioinit1(
                    int bdno,          ボード番号
                    int flg,           ﾊﾝﾄﾞﾘﾝｸﾞ信号極性
                    int oncoef,        ﾊﾝﾄﾞﾘﾝｸﾞ信号 ON 時間
                    int delcoef)       ﾊﾝﾄﾞﾘﾝｸﾞ信号遅延時間
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

flg　　ハンドリング極性を 0：負論理、1：正論理で指定します。

oncoef ハンドリング信号のＯＮ時間を下記計算式より係数(1～255)に
　　変換して指定します。

delcoef ハンドリング信号の遅延時間を下記計算式より係数(1～255)に
　　変換して指定します。

　　ＴＴＬ時…ＯＮ時間　又　遅延時間(nSec)　/40nSec
　　絶縁時　…ＯＮ時間　又　遅延時間(μSec)　/100μSec
　　ハンドリング未使用時は０を指定してください。

【戻り値】

０　　正常終了

－１　引数エラー

９.２９　A67X_145pioor　（デジタル出力 OR 32bit）

【機能】

指定ポートへ反転したデータをＯＲします。

【書式】

```
int     A67X_145pioor(
                    int bdno,           ボード番号
                    int port,           ポート番号
                    unsigned int data)  ＯＲするﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

data　ＯＲするデータ

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioand

【使用例】

```
#include “A145_67bios.h”
#defineBD_BASE      0x03004000
#defineBD_MAX       4
void main()
{
        A67X_145init( BD_MAX, BD_BASE );
        A67X_145pioinit1( 0, 0);
        A67X_145pioset( 0, 0, 0, 0);
        A67X_145pioor( 0, 0, 0x222 );
}
```

９.３０　A67X_145pioor16　（デジタル出力 OR 16bit）

【機能】

指定ポートへ反転した 16bit データをＯＲします。

【書式】

```
int     A67X_145pioor16(
                    int bdno,               ボード番号
                    int port,               ポート番号
                    int ud,                 上位/下位
                    unsigned int data)      ＯＲするﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

port　ポート番号を０又は２で指定します。

ud　１：上位、２：下位のいずれかを指定します。

data　ＯＲするデータ

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioand16

９.３１　A67X_145pioout　（デジタル出力 32bit）

【機能】

３２bit デジタル出力を行います。

【書式】

```
int     A67X_145pioout(
                int bdno,               ボード番号
                int port,               ポート番号
                unsigned int data)      出力ﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

data　出力するデータを指定します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioin

９.３２　A67X_145pioout16　（デジタル出力 16bit）

【機能】

１６bit デジタル出力を行います。

【書式】

```
int     A67X_145pioout16(
                    int bdno,           ボード番号
                    int port,           ポート番号
                    int ud,             上位/下位
                    unsigned int data)  出力ﾃﾞｰﾀ
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

port　ポート番号を０又は２で指定します。

ud　　１：上位、２：下位のいずれかを指定します。

data　出力するデータを指定します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioset, A67X_145pioin16

９.３３　A67X_145pioset　（デジタル入出力設定）

【機能】

デジタル入出力のハンドリング方式とアクセス方法（32bit/16bit）を設定します。

【書式】

```
int     A67X_145pioset(
                    int bdno,       ボード番号
                    int port,       ポート番号
                    int bit,        32bit/16bit 上位・下位
                    int method)     ﾊﾝﾄﾞﾘﾝｸﾞ方式
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

port　ポート番号を０～３で指定します。

bit　０：３２bit,１：１６bit 上位,２：１６bit 下位のいずれかを指定します。

method　ハンドリング方式を指定します。
０：なし、１：STB/ACK、２：ﾗｲﾄｽﾄﾛｰﾌﾞ、３：ﾎｰﾙﾄﾞ

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pioinit1, A67X_145pioin, A67X_145pioout

９.３４　　A67X_145plsinit　　(パルス出力初期化)

【機能】

パルス出力の初期設定とパルス出力時の１回転のパルス数・設定データ反映タイミング設定を４チャネル全て行います。パルス出力するにあったって最初に必ず実行してください。

【書式】

```
int     A67X_145plsinit(
                    int bdno,          ボード番号
                    int num[4])        １回転のパルス数
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します

num[4]　１回転のパルス数は、Ｚ相の出力をＡ・Ｂ相パルスの何パルスに１回出力するかを 2～50000000 の範囲で設定します
使用しないチャネルは 0 を設定してください。

【戻り値】

０　　正常終了

－１　　引数エラー

【参考】

A67X_145plsstart, A67X_145plsstop, A67X_145plsset

【使用例】

```
#include   “A145_67bios.h”
#defineBD_MAX   4
#defineBD_BASE  0x03004000

void main()
{
        int num[4]={2000,0,0,0};
        A67X_145init(BD_MAX, BD_BASE );
        A67X_145plsinit( 0, num );
        A67X_145plsset( 0, 0, 1.0, 0 );  // 1Hz,正転
        A67X_145plsstart( 0, 0 );        // ﾊﾟﾙｽｽﾀｰﾄ
            ・
            ・
            ・
        A67X_145plsstop( 0, 0 );         // ﾊﾟﾙｽｽﾄｯﾌﾟ
}
```

９．３５　A67X_145plspause　(パルス出力一時停止)

【機能】

パルス出力を一時停止します。制御内カウンタはリセットされません。

【書式】

```
int     A67X_145plspause(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します

ch　　チャネル番号を０～３で指定してください。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145plsset, A67X_145plsstop,
A67X_145plsinit, A67X_145plsrestart

９.３６　A67X_145plsrestart　（パルス出力再スタート）

【機能】

パルス出力を開始します。出力開始時は、制御内カウンタはリセットされません。

【書式】

```
int     A67X_145plsrestart(
                    int bdno,       ボード番号
                    int ch,         チャネル番号
                    int direct)     回転方向
```

【引数】

bdno	67X_145init で指定した範囲のボード番号０～３の範囲で指定します
ch	チャネル番号を０～３で指定してください。
direct	回転方向を ０：正転（ＣＷ） １：逆転（ＣＣＷ）で指定してください。

【戻り値】

０	正常終了
－１	引数エラー

【参考】

A67X_145plsset, A67X_145plsstop,
A67X_145plsinit, A67X_145plspause

９.３７　A67X_145plsset　(パルス出力設定)

【機能】

チャネル毎に、パルス出力時の周期・回転方向を設定します。

【書式】

```
int     A67X_145plsset(
                    int bdno,       ボード番号
                    int ch,         チャネル番号
                    float period,   周期
                    int direct)     回転方向
```

【引数】

bdno	A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。
ch	チャネル番号を０～３で指定してください。
period	出力するパルス周期（Ｈｚ）を 0.01～6MHz の範囲で指定してください。
direct	回転方向を ０：正転（ＣＷ） １：逆転（ＣＣＷ）で指定してください。

【戻り値】

０	正常終了
－１	引数エラー

【参考】

A67X_145plsstart, A67X_145plsstop, A67X_145plsinit

９.３８　A67X_145plsstart　(パルス出力スタート)

【機能】

パルス出力を開始します。出力開始時は、制御内カウンタはリセット状態です。

【書式】

```
int     A67X_145plsstart(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します。

ch　　チャネル番号を０～３で指定してください。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145plsset, A67X_145plsstop, A67X_145plsinit

９.３９　A67X_145plsstop　(パルス出力停止)

【機能】

制御プロセス内のカウンタがリセットされ、パルス出力を停止します。

【書式】

```
int     A67X_145plsstop(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で
　　指定します。

ch　　チャネル番号を０～３で指定してください。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145plsset, A67X_145plsstop, A67X_145plsinit

９.４０　A67X_145pwmduty　（PWMDuty データ設定 単相用）

【機能】

単相時のＤｕｔｙ比を設定します。

【書式】

```
int     A67X_145pwmset(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float duty[2],     Duty 比
                    int direct[2])     方向
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定します。

duty[2]　Duty 比を 0.0～１00.0 の範囲 U・V 相の順で指定します。

direct[2] 出力方向を０，１で指定します。

１・・・N側出力

０・・・P側出力

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pwminit, A67X_145pwmduty3

９.４１　A67X_145pwmduty3　（PWMDuty データ設定 ３相用）

【機能】

３相時のＤｕｔｙ比を設定します。

【書式】

```
int     A67X_145pwmset(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float duty[3],     Duty 比
                    int direct[3])     方向
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

duty[3]Duty 比を－100.0～１00.0 の範囲 U・V・W 相の順で指定します。

direct[3] 出力方向を０，１で指定します。

１・・・N側出力

０・・・P側出力

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pwminit, A67X_145pwmduty

９.４２　A67X_145pwminit　（PWM初期値設定 単相用）
９.４３　A67X_145pwminit3　（PWM初期値設定 ３相用）

【機能】

ＰWM初期値設定は、単相用 A67X_145pwminit・３相用 A67X_145pwminit3 にて、各チャネルのデッドタイムと設定データ反映タイミングを設定します。この関数を実行することにより、モード（単相・３相）が設定されます。

【書式】

```
int     A67X_145pwminit(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float dtime,       デッドタイム(mSec)
                    int reflect,       反映タイミング
                    int mode )
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

dtime　デットタイムを０～２０００μSec で指定してください。

reflect　設定データ反映タイミングを０：周期に同期、１：書込み時で指定してください。

mode　出力モードを０，１で指定します。
０・・・フルブリッジ
１・・・ハーフブリッジ

【戻り値】

０　正常終了
－１　引数エラー

【参考】

A67X_145pwmset，A67X_145pwmset3

【使用例】

```
#include   “A145_67bios.h”
#defineBD_MAX   4
#defineBD_BASE  0x03004000

void main()
{
        A67X_145init(BD_MAX, BD_BASE );
        A67X_145pwminit( 0, 0, 0, 0, 0 );
        A67X_145pwmset( 0, 0, 24, 0.5 ); //24Hz,Duty50%       A67X_145pwmstart(   0,
0 );        // PWM ｽﾀｰﾄ
             ・
             ・
             ・
        A67X_145pwmstop( 0, 0 );                    // PWM ｽﾄｯﾌﾟ
}
```

９.４４　A67X_145pwmset　（PWMデータ設定 単相用）

【機能】

単相時のキャリア周波数・Ｄｕｔｙ比を設定します。

【書式】

```
int     A67X_145pwmset(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float freq,        キャリア周波数
                    float duty[2],     Duty 比
                    int direct[2] )    方向
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

freq　キャリア周波数を１２０KHz～２４Hz で指定してください。

duty[2]Duty 比を 0.0～100.0 の範囲 U・V 相の順で指定してください。

direct[2] 出力方向を０，１で指定します。

１・・・N側出力

０・・・P側出力

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pwminit, A67X_145pwmset3

９.４５　A67X_145pwmset3　（PWMデータ設定 ３相用）

【機能】

３相時のキャリア周波数・Ｄｕｔｙ比を設定します。

【書式】

```
int     A67X_145pwmset(
                    int bdno,          ボード番号
                    int ch,            チャネル番号
                    float freq,        キャリア周波数
                    float duty[3],     Duty 比
                    int direct[3] )    方向
```

【引数】

bdno　　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

freq　　キャリア周波数を１２０KHz～２４Hz で指定してください。

duty[3]Duty 比を 0.0～１00.0 の範囲 U・V・W 相の順で指定してください。

direct[3] 出力方向を０，１で指定します。

１・・・N側出力

０・・・P側出力

【戻り値】

０　　正常終了

－１　　引数エラー

【参考】

A67X_145pwminit, A67X_145pwmset

９.４６　A67X_145pwmstart　（PWM出力開始）

【機能】

指定したチャネルのＰＷＭ出力を開始します。

【書式】

```
int     A67X_145pwmstart(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pwmstop

９.４７　A67X_145pwmstop　（PWM出力停止）

【機能】

指定したチャネルのＰＷＭ出力を停止します。

【書式】

```
int     A67X_145pwmstop(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145pwmstart

９.４８　A67X_145pwmwrite　（PWMデータ同時書込み指示）

【機能】

指定したチャネルのPWMデータを書き込みます。

【書式】

```
int     A67X_145pwmwrite(
                    int bdno,       ボード番号
                    int ch)         チャネル番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

ch　　チャネル番号を単相時は０～３・３相時は０～２で指定してください。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145pwminit, A67X_145pwminit3

９.４９　A67X_145timerreset　(ウォッチドグタイマカウンタリセット)

【機能】

ウォッチドグタイマのカウンタをリセットします。エラー出力はリセットされません。

【書式】

```
int     A67X_145timerreset(
                    int bdno)          ボード番号
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

【戻り値】

０　正常終了

－１　引数エラー

【参考】

A67X_145timerset

９.５０　　A67X_145timerset　　（ウォッチドグタイマ周期セット）

【機能】

ウォッチドグタイマの監視周期をセットします。

【書式】

```
int     A67X_145timerreset(
                    int bdno,          ボード番号
                    int time )         周期時間(μSec)
```

【引数】

bdno　A67X_145init で指定した範囲のボード番号０～３の範囲で指定します。

time　監視周期時間を 0～1000000000μSec の範囲でμSec 単位で設定します。０を設定すると動作を停止し、エラー出力もリセットされます。０以外を設定すると自動的にウォッチドグタイマが動作開始します。

【戻り値】

０　　正常終了

－１　引数エラー

【参考】

A67X_145timerreset

【使用例】

```
#include       “A145_67bios.h”
#defineBD_MAX  4
#defineBD_BASE 0x03004000

void main()
{
       A67X_145init(BD_MAX, BD_BASE );
       A67X_145timerset( 0, 1000 );
}
```

１０．　サンプルプログラムについて

はじめに、A67X_145init()にてボード枚数を指定します。環境に応じて変更してご利用ください。

１０.１　　ＡＤＳＰ３２４－００A用

サンプルプログラムは、プログラム名と同一のバッチファイルを利用してコンパイル/リンクが可能です。又弊社製コンソールアダプタ（ADSP324-331）にて動作が可能です。コンソールアダプタをお持ちでないかたはコーディングサンプルとして活用ください。

１０.１.１ Sample1

ウォッチタイマ・タイマ割込み・カウンタ入力・PIO の使用サンプルです。ウォッチでグタイマはセーフモードとエラーモードの２回実行されます。ウォッチドグエラー信号(CN4-11 ピン)を監視しながら実行してください。カウンタは４逓倍で１Sec に１回入力し１０回表示します。又 PIO は３２/１６bit、ハンドリング有無などのサンプルです。ガイドに従って選択して実行してください。

１０.１.２ Sample2

Ｐwm・キャプチャ・パルスの使用サンプルです。単相/３相モードなどガイドに従って実行してください。

１０.２　　ＡＤＳＰ６７４－００シリーズ用

サンプルプログラムをコンパイル時は、a145_67bios.lib,a145_67bios.h 等の SearchPath 等の Project Build Opion（Compile ｵﾌﾟｼｮﾝ・Link ｵﾌﾟｼｮﾝ）を設定し、ビルドしてください。プログラムの実行は、XDS510 又は弊社製コンソールアダプタ(ADSP674-331)にて可能です。弊社製コンソールアダプタにて使用時は、インストール後各サンプルのコマンドファイル(～.cmd)で指定している標準ライブラリを rts6701.lib から rtsc674.lib へ切り替えてリビルドしてください。デフォルトはＸＤＳでの動作モードになっています。

１０.２.１ SmplGen（ウォッチドグ・カウンタ・パルス）

ウォッチタイマ・タイマ割込み・カウンタ入力の使用サンプルです。ウォッチでグタイマはセーフモードとエラーモードの２回実行されます。ウォッチドグエラー信号（CN4-11 ピン）を監視しながら実行してください。カウンタは４逓倍で１Sec に１回入力し１０回表示します。パルス出力は、10Ｈｚ、2000 パルス/回転、正転の出力です。

１０.２.２ SmplPIO1（PIO サンプル ―ハンドリングなし―）

ＰＩＯの各種関数をハンドリングなしで利用するサンプルです。又実行時にＰＩＯ入出力の環境をガイドメッセージに従って入力します。(入出力用各ボードＮＯ・ポートＮＯ、３２/１６bit 選択)

注）絶縁ＰＩＯ時はコンパイル時“IS_ISO”を Define して使用してください。

１０.２.３ SmplPIO2（PIO サンプル ―ハンドリング使用―）

ＰＩＯの各種関数をホールド・ＳＴＢ／ＡＣＫ方式を利用したサンプルで実行時は割込みを使用するため DSW104-1 を ON にしてください。

又実行時にＰＩＯ入出力の環境をガイドメッセージに従って入力します。

(入出力用各ボードＮＯ・ポートＮＯ、３２/１６bit 選択)

１０.２.４ SmplPWM（PWM サンプル）

PWM 出力のサンプルです。最初に単相又は３相を選択し開始してください。途中 DUTY 比の変更入力があります。方向指定で出力側 P・N をコントロールできます。

１０.２.５ SmplCapt（キャプチャサンプル）

キャプチャ入力のサンプルです。最初に単相又は３相を選択し開始してください。又 ADSP324-145PWM ボードを使用してキャプチャする場合は、“Is PWM of ADSP324-145 used ? (0:NO, 1:Yes)”で“１”を選択してください。１０ｋＨｚ、Duty５０％の PWM が出力されます。

１１．　ボード制御ソフトを書く上での注意

ＡＤＳＰボード制御ソフトをユーザーサイドで独自に作る場合においての注意点を説明します。

１１.１　　ＡＤＳＰ３２４－００A用

１１.１.１ ベクタの使用

割り込みを複数のボードで使用するうえで、ベクタ番号は重要な役割を持ちます。ベクタ番号の設定は、ＤＳＷ１０４で行うことができ、ボード間で重複しないように設定します。

例）

１枚目のボード　DSW104-1 をＯＮ、他はＯＦＦ
２枚目のボード　DSW104-2 をＯＮ、他はＯＦＦ

このように設定しておくことにより、どのボードから割り込み要求がきたか知ることができるようになります。

知る方法は、ベースアドレスの下位１６ビットがすべて１のアドレス（nnffffH）番地を読むことによって行います。（ ）内の nn はボードのベースアドレスの上位８ビットの設定です。このことからもわかるとおり、割り込みを使用する全てのベースアドレス上位８ビットは同一の設定である必要があります。

ベクタボードは割り込みが発生していない場合、下位８ビット全てが１です。割り込みが発生した場合、割り込み要求をだしているボードのＤＳＷ１０４のON位置のビットが０になります。

以下にベクタを用いたプログラム例を示します。

例）

ボードは２枚実装されているものとし、１枚目はベクタ番号１（DSW104-1をＯＮ)、２枚目はベクタ番号２（DSW104-2をＯＮ）とします。

```
BD1_VECT        .set    0001h           ;ボード１のベクタビット
BD2_VECT        .set    0002h           ;ボード２のベクタビット
                .bss    bd_base,1
VECT_MASK:      .word   0000ffffh
    int_entry:  push    st              ;必要レジスタの退避
                push    dp
                push    r0
                push    xx              ;必要に応じてレジスタを退避
                ldi     if,r0           ;ＩＦレジスタのコピー
                and     0008h,r0        ;ＩＮＴ３位置のマスク
                bnz     int_exit                ;多重割り込みならば回避
                ldp     @bd_base,1
                ldi     @bd_base,ar0    ;ボードのベースアドレス
                ldp     @VECT_MASK      ;下位１６ビットのセット
                or      @VECT_MASK
                ldi     *ar0,r0                 ;ベクタ番号の取得
                and     BD1_VECT,r0     ;ボード１のベクタ番号
                callz   bd1_prog        ;ボード１処理をコール
                and     BD2_VECT,r0     ;ボード２のベクタ番号
                callz   bd2_prog        ;ボード２処理をコール
                sti     r0,*ar0                 ;全てのボードの割り込み解除
    int_exit:   pop     xx              ;レジスタの復帰
                pop     r0
                pop     dp
                pop     st
                reti
```

bd_base は割り込みを許可する前に設定されているものとします。

bd1_prog と bd2_prog は各ボードの割り込み処理プログラムであり、全てのレジスタを破壊しないものとします。

１１.２　　ＡＤＳＰ６７４－００シリーズ用

１１.２.１ ベクタの使用

割り込みを複数のボードで使用する上で、ベクタ番号は重要な役割を持ちます。ベクタ番号の設定は、DSW104 で行うことができボード間で重複しないように設定します。

例)
1 枚目のボード　DSW104-1 を ON、他は OFF
2 枚目のボード　DSW104-2 を ON、他は OFF

このように設定しておくことにより、どのボードから割り込み要求が来たかを知ることができるようになります。

方法は、ベースアドレスの下位 20 ビットが 3fffc のアドレス(nnn3fffc)番地を読むことによって行います。()内の nnn は、ボードのベースアドレスの上位 12 ビットの設定です。この事からわかる様に、割り込みを使用するボード全てのベースアドレスの上位 12 ビットは同一の設定である必要があります。

ベクトポートは割り込みが発生していない場合、下位 8 ビット全てが 1 です。割り込みが発生した場合、割り込み要求を出しているボードの DSW104 の ON 位置のビットが 0 になります。ベクトポートのアドレスはこのライブラリを使用時は A67X_145init()関数実行後“a145_VECT_PORT”で参照できます。

以下に、ベクタを用いたプログラムを示します。
例）ボードが 2 枚実装されているものとし、1 枚目はベクタ番号 1(DSW104-1 を ON)、2 枚目はベクタ番号 2(DSW104-2 を ON)とします。

```
#define BD_MAX   2                  // 実装ボード枚数
#define BD_BASE  0x03004000         // ボードのベースアドレス
void main()
{
    A67X_145init( (int)BD_MAX, (unsigned int)BD_BASE );
    A67X_145intdefine(0, c_int90 );        // １枚目の割込み
    A67X_145intdefine(1, c_int90 );        // ２枚目の割込み
            ・
            ・
}
```

```
//=============================================
//          割り込み処理
//=============================================
 interrupt void c_int90(void)
  {
    asm("          nop     2              ");
    asm("          mvc     IFR, b3        "); //２重割込みﾁｪｯｸ
    asm("          mvk     0080h, b4      ");
    asm("          and     b4, b3, b0     ");
    asm("   [b0]   b       ext_int        ");
    asm("          nop     5              ");
    tmp = 1;
    if( !(tmp & (*a145_VECT_PORT)) ){       // １枚目の割込み発生
          tmp = (EBL_H1 >> 2);
          tmp |= ( 0x000001ff & tmp );
          if( tmp & Port[bd]->tm.intflg ) { // 割込みﾎﾟｰﾄ確認
                ・                          // 割込み時の処理記述
                ・
          }
    }
    tmp = 2;
    if( !(tmp & (*a145_VECT_PORT)) ){       // ２枚目の割込み発生
          tmp = (EBL_H1 >> 2);
          tmp |= ( 0x000001ff & tmp );
          if( tmp & Port[bd]->tm.intflg ) { // 割込みﾎﾟｰﾄ確認
                ・                          // 割込み時の処理記述
                ・
          }
    }

    Port[bd]->tm.intflg = 0;                // 割込みフラグリセット
    asm("ext_int:                           ");
}
```

太字部分は A67X_145intflg() にて実行できます。使用する際は処理時間を考慮してくださ

太字部分は A67X_145intflgreset() にて実行できます。使用する際はｚ処理時

１１.２.２　ユーザソフトをアセンブラで記述する場合

拡張バスのメモリにデータを書き込む場合は “ STB ”“ STH ”命令は使用しないで下さい。以下に説明を示します。

STB 命令では 8 bit 単位で、STH 命令では 16 bit 単位でメモリの読み書きを行います。しかし、拡張ボードにデータを書き込む場合は 32 bit(1 word)単位で実効する必要があります。

以上の様に、それぞれ扱うデータサイズが異なるため STB・STH 命令を使用した場合は不完全なデータになります。

本マニュアルの内容は製品の改良のため予告なしに
変更されることがありますので、ご了承ください。

ADSP324-145　ﾏﾙﾁﾌｧﾝｸｼｮﾝﾎﾞｰﾄﾞ
ｿﾌﾄｳｪｱ･ﾕｰｻﾞｰｽﾞ･ﾏﾆｭｱﾙ

中部電機株式会社
〒440-0004　　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2003．7　　第１版発行
2003．8　　第２版発行