ＡＤＳＰシリーズ

ＡＤＳＰ３２４－１１

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．概要

ＡＤＳＰ３２Ｘ－１１／６１サポートソフトウェアは、ＡＤＳＰ３２Ｘ－１１／６１を使用するための基本機能を含んだＢＩＯＳプログラム（Ａ１１ＢＩＯＳ）および、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。
Ａ１１ＢＩＯＳはＣで書かれているため高速度で使用するときには問題がおきるかもしれませんが、ＡＤＳＰ３２Ｘ－１１／６１を動作させる上で大きなヒントになると思われます。

２．機能一覧

Ａ１１ＢＩＯＳには次の機能があります。

1） ＡＤＳＰ３２Ｘ－１１／６１ボードの初期化
2） カウンターのデータ入力機能
3） カウンターのデータプリセット機能
4） 絶縁出力ポートへの出力機能
5） 絶縁入力ポートからの入力機能
6） 割り込み機能

３．供給形態

Ａ１１ＢＩＯＳはソースファイルおよび、ＣＯＦＦファイル形式のオブジェクト、ライブラリー形式で供給されます。
ユーザプログラムとリンクして使用してください。
A11BIOS.H と A11BIOS.LIB を C_DIR 環境変数の示すディレクトリにコピーしておけば簡単に利用することができます。

４．供給ファイルの一覧

README.DOC	Ａ１１ＢＩＯＳの簡単な説明が書かれています
A11BIOS.H	Ａ１１ＢＩＯＳを使用するためのヘッダファイル
A11BIOS.C	Ａ１１ＢＩＯＳのソースファイル
A11BIOS.OBJ	Ａ１１ＢＩＯＳのオブジェクトファイル
A11BIOS.LIB	Ａ１１ＢＩＯＳのライブラリファイル
SMPL.BAT	サンプルソフトをコンパイルするためのバッチファイル
SAMPLE.C	Ａ１１ＢＩＯＳを用いたサンプルプログラム
SAMPLE.LNK	サンプルソフトの実行ファイルを作成するためのリンクファイル

５．関数の一覧

○初期化関数

A32X_11init	ボードの初期化およびライブラリの初期化を行います

○カウンター入出力関数

A32X_11cntin	指定ボードの指定カウンターからデータを入力します
A32X_11cntout	データを指定ボードの指定カウンターへプリセットします

○絶縁入出力関数

A32X_11out	データを指定ボードの絶縁出力ポートへ出力します
A32X_11set	指定ボードの絶縁出力ポートの指定ビットをセットします
A32X_11reset	指定ボードの絶縁出力ポートの指定ビットをリセットします
A32X_11or	指定ボードの絶縁出力ポートのデータをＯＲします
A32X_11and	指定ボードの絶縁出力ポートのデータを反転ＡＮＤします
A32X_11in	指定ボードの絶縁入力ポートのデータを取得します
A32X_11outm	指定ボードの絶縁出力ポートの状態をモニターします

○割り込み関数

A32X_11intentry	割り込み処理関数の登録／解除を行います

６．関数詳細

関数名　　ボードの初期化およびライブラリの初期化

記述　　int　　A32X_11init (max,base)；

引き数　　int　　max；　　１１ボードの実装枚数（１～４）
unsigned long　base；　　１１ボードのベースアドレス

説明　　ＡＤＳＰ３２Ｘ－１１／６１を初期化出力を全てＯＦＦにします。また、
ライブラリの諸設定をおこないます。
ボード実装枚数の指定は、１～４が指定可能です。
ボードのベースアドレスは、１枚目のボードから１０ｈステップで各ボードを
設定し、１枚目のベースアドレスを指定してください。
すべてのボードアドレスの上位８ビットは、同一にする必要があります。

戻り値　　－1 _ERROR　　パラメータ異常
0　_NOERRROR　　正常に初期化されました

使用例

```
#include        <a11bios.h>

#define         BD_BASE      0x900100       /*  ボードベース  */

void  main (void)
   {
    A32X_11init (1,BD_BASE)；                /*  初期化  */
    }
```

関数名　指定ボードの指定カウンターデータの入力

記述　int　A32X_11cntin(bd,ch,data)；

引き数

int	bd；	ボード番号（０～３）
int	ch；	カウンターチャンネル番号（０～３）
unsigned long	*data；	入力データ格納ポインタ

説明　指定ボードの指定カウンターチャンネルからカウンターデータ入力します。

戻り値

-1 _ERROR	パラメータ異常
0 _NOERRROR	正常に実行されました

使用例

```
#include        <a11bios.h>

#define         BD_BASE      0x900100       /*  ボードベース  */

void  main (void)
   {
    unsigned  long  data ;

    A32X_06init (1,BD_BASE) ;                /*  初期化  */
    A32X_06in (0,0,&data) ;                  /*  データ入力  */
    }
```

関数名　指定ボードの指定カウンターのプリセット

記述　int　A32X_11cntout (bd,ch,data,) ;

引き数

int	bd;	ボード番号（０～３）
int	ch;	カウンターチャンネル番号（０～３）
unsigned long	data;	出力データ

説明　指定ボードの指定カウンターにデータをプリセットします。

戻り値

-1 _ERROR	パラメータ異常
0 _NOERRROR	正常に実行されました

使用例

```
#include        <a11bios.h>

#define         BD_BASE      0x900100       /*  ボードベース  */

void  main (void)
   {
    A32X_11init (1,BD_BASE) ;                /*  初期化  */
    A32X_11cntout (0,0,0x12345678L) ;        /*  12345678 をプリセット */
    }
```

関数名　指定ボードへの絶縁出力

記述　int　A32X_11out (bd,data) ;

引き数

int	bd;	ボード番号（０～３）
unsigned	data;	出力データ

説明　指定ボードの絶縁出力ポートにデータを出力します。

戻り値

-1	_ERROR	パラメータ異常
0	_NOERRROR	正常に実行されました

使用例

```
#include        <a11bios.h>

#define         BD_BASE     0x900100        /*  ボードベース  */

void  main (void)
   {
    A32X_11init (1,BD_BASE) ;               /*  初期化  */
    A32X_11out (0,0x1234) ;                 /*  1234 を出力 */
    }
```

関数名　指定ボードの絶縁出力ビットセット

記述　int　A32X_11set (bd,bit) ;

引き数
```
int          bd;     ボード番号（０～３）
int          bit;    ＯＮするビット位置（０～１５）
```

説明　指定ボードの絶縁出力ポートの指定位置ビットをＯＮにします。

戻り値
```
-1 _ERROR      パラメータ異常
0  _NOERRROR   正常に実行されました
```

使用例

```
#include        <a11bios.h>

#define         BD_BASE     0x900100        /*  ボードベース  */

void  main (void)
   {
    A32X_11init (1,BD_BASE) ;               /*  初期化  */
    A32X_11set (0,0) ;                      /*  ビット０をＯＮ */
    }
```

関数名　指定ボードの絶縁出力のビットリセット

記述　int　A32X_11reset (bd,bit) ;

引き数　int　bd;　ボード番号（０～３）
int　bit;　ビット位置（０～１５）

説明　指定ボードの絶縁出力ポートの指定位置ビットをＯＦＦにします。

戻り値　-1 _ERROR　パラメータ異常
0 _NOERRROR　正常に実行されました

使用例

```
#include <a11bios.h>

#define BD_BASE 0x900100 /* ボードベース */

void main (void)
  {
   A32X_11init (1,BD_BASE) ;  /* 初期化 */
   A32X_11reset (0,0) ;  /* ビット０をＯＦＦ */
   }
```

関数名　　指定データの絶縁出力のＯＲ

記述　　int　　A32X_11or (bd,data) ;

引き数　int　　bd;　　ボード番号（０～３）
　　unsigned int　　data;　　ＯＲするデータ

説明　　指定ボードの絶縁出力ポートにデータをＯＲします。

戻り値　-1 _ERROR　　パラメータ異常
　　0 _NOERRROR　　正常に実行されました

使用例

```
#include　　<a11bios.h>

#define　　BD_BASE　　0x900100　　/*　ボードベース　*/

void　main (void)
　{
　 A32X_11init (1,BD_BASE) ;　　/*　初期化　*/
　 A32X_11or (0,0x5555) ;　　/*　5555 をＯＲ */
　 }
```

関数名　指定データの絶縁出力の反転ＡＮＤ

記述　int　A32X_11and (bd,data) ;

引き数

int		bd;	ボード番号（０～３）
unsigned	int	data;	反転ＡＮＤするデータ

説明　指定ボードの絶縁出力ポートに反転したデータをＡＮＤします。

戻り値

-1	_ERROR	パラメータ異常
0	_NOERRROR	正常に実行されました

使用例

```
#include        <a11bios.h>

#define         BD_BASE     0x900100        /*  ボードベース  */

void  main (void)
   {
    A32X_11init (1,BD_BASE) ;               /*  初期化  */
    A32X_11and (0,0x5555) ;                 /*  AAAA をＡＮＤ */
    }
```

関数名　　指定ボードの絶縁入力データの入力

記述　　int　　A32X_11in (bd,data) ;

引き数　　int　　bd;　　ボード番号（０～３）
　　unsigned long　　*data;　　入力データ格納ポインタ

説明　　指定ボードの絶縁入力ポートからデータを入力します。

戻り値　　-1 _ERROR　　パラメータ異常
　　0 _NOERRROR　　正常に実行されました

使用例

```
#include　　<a11bios.h>

#define　　BD_BASE　　0x900100　　/*　ボードベース　*/

void　main (void)
　{
　 A32X_11init (1,BD_BASE) ;　　/*　初期化　*/
　 A32X_11in (0,&data) ;　　/*　データ入力　*/
　 }
```

関数名　割り込み関数

記述　int　A32X_11intentry (bd,intf,intslp, (*func) ( ));

引き数

int	bd;	ボード番号（０～３）
int	intf;	割り込み無／有（０、１）
int	intslp;	割り込み立ち下がり／上がり（０、１）
void	(*func)( );	割り込み関数

説明　指定ボードへの割り込み処理関数の設定を行います。
ボードへの割り込みが発生するとここで設定された関数が自動的に実行されます。
設定する割り込み処理関数へは、引き数は渡せません。
尚、この関数はｃ＿＿ｉｎｔｎｎではなく通常の関数名としてください。

戻り値

-1	_ERROR	パラメータ異常
0	_NOERRROR	正常に実行されました

使用例

```
#include        <a011bios.h>

#define         BD_BASE      0x900100       /*  ボードベース  */

void  intr_00 ( )
   {
    printf(“割り込み関数です。プログラムを記述してください。¥n”);
    return;
    }

void  main (void)
   {
    A32X_11init (1,BD_BASE) ;                  /*  初期化  */
    A32X_11intentry (0,1,1,intr_00) ;          /*  割り込み有効  */
    }
```

７．ボード制御ソフトを書く上での注意

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。

１）割り込み使用時の注意

１－１）割り込みプログラム

ＡＤＳＰ３２Ｘ－１１／６１ボードは割り込みの発生を１００ｎＳｅｃの
ワンショットで生成しています。また、ＴＭＳ３２０Ｃ３１の割り込みはレベル割り込みになっています。
このことにより、ＡＤＳＰ３２Ｘ－１１／６１ボードで割り込みを使用すると、１回の割り込み条件で連続して２回の割り込みが発生してしまいます。
これを回避するためには、割り込み処理プログラムに下記のプログラムを挿入して回避してください。
ボードの割り込みは、ＩＮＴ３です。

```
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

;   通常の割り込みプログラム

int_exit:       pop     xx              ;   レジスタの復帰
                pop     r0
                pop     dp
                pop     st
                reti
```

１－２）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きたか知ることが出来るようになります。

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。
このことからも分かるとおり、割り込みを使用するボードすべ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。

ベクター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。

下記にベクタを用いたプログラム例をしめします。

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    ar0
                push    xx              ;必要に応じてレジスタを退避
                ldi     if,r0           ;ＩＦレジスターのコピー
                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     DB2_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 は割り込み許可をする前に、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。

・本マニュアルの内容は製品の改良のため予告無しに
変更されることがありますので、ご了承下さい。

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

ADSP324-11

ｿﾌﾄｳｪｱ・ﾕｰｻﾞｰｽﾞ・ﾏﾆｭｱﾙ

1999.10　　第３版発行