ＡＤＳＰシリーズ

ＡＤＳＰ３２４－０６

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－０６／５６サポートソフトウェアは、ＡＤＳＰ３２Ｘ－０６／５６を使用するための基本機能を含んだＢＩＯＳプログラム（Ａ０６ＢＩＯＳ）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。Ａ０６ＢＩＯＳはC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、ＡＤＳＰ３２Ｘ－０６／５６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

Ａ０６ＢＩＯＳには次の機能があります。

①ＡＤＳＰ３２Ｘ－０６／５６ボードの初期化
②ボードへの出力機能
③ボードからの入力機能
④他ＰＩＯボードとのパラレル同期通信機能

３．供給形態

Ａ０６ＢＩＯＳ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、ライブラリ形式で供給されています。ユーザプログラムとリンクして使用して下さい。
A06BIOS.H と A06BIOS.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　Ａ０６ＢＩＯＳの簡単な説明が書かれています。
A06BIOS.H　　Ａ０６ＢＩＯＳを使用するためのヘッダファイル
A06BIOS.C　　Ａ０６ＢＩＯＳのソースファイル
A06BIOS.OBJ　Ａ０６ＢＩＯＳのオブジェクトファイル
A06BIOS.LIB　Ａ０６ＢＩＯＳ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　　Ａ０６ＢＩＯＳ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_06init　　ボードの初期化および、ライブラリーの初期化をおこなう

○出力関数
A32X_06out　　データを指定ボードへ出力します
A32X_06set　　指定ボード出力の指定ビットをセットします
A32X_06reset　　指定ボード出力の指定ビットをリセットします
A32X_06or　　指定ボード出力にデータをＯＲします
A32X_06and　指定ボード出力のデータを反転ＡＮＤします
A32X_06outm　指定ボード出力の状態をモニターします

○入力関数
A32X_06strd　同期入力モード時のストローブ受理状態を得ます
A32X_06in　　入力データを取得します

６．関数詳細

関数名　　ボードの初期化およびライブラリーの初期化

記述　　int　A32X_06init（max,mode,base）;

引き数

int	max；	０６ボードの実装枚数
int	mode；	入力モード（0＝非同期、1＝同期）
unsigned long	base；	０６ボードのベースアドレス

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

説明　　ＡＤＳＰ３２Ｘ－０６／５６を初期化出力を全てＯＦＦにします。また、ライブラリーの諸設定をおこないます。
ボード実装枚数の指定は、１～４が指定設定可能です。
ボードのベースアドレスは、１枚目のボードから４ｈ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。

使用例

```
#include　　<a06bios.h>

#define　　BD_BASE　0x900080　　／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);　　／＊初期化＊／
}
```

関数名　　指定ボードへの出力

記述　　int　A32X_06out (bd,data) ;

引き数　　int　　bd ;　　ボード番号
　　unsigned　long　　data ;　　出力データ

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

説明　　指定ボードにデータを出力します。

使用例

```
#include　　<a06bios.h>

#define　　BD_BASE　0x900080　　／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);　　／＊初期化＊／
A32X_06out (0,0x123456678L) ;　　／＊12345678 を出力＊／
}
```

関数名　　指定ボードのビットセット

記述　　int　A32X_06set (bd,bit) ;

引き数　　int　　bd ;　　ボード番号
　　int　　bit ;　　ＯＮするビット位置（０～３１）

戻り値

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

説明　　指定ボードの出力を指定位置をＯＮにします。

使用例

```
#include　　<a06bios.h>

#define　　BD_BASE　0x900080　　／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);　　／＊初期化＊／
A32X_06set (0,0) ;　　／＊ビット０をＯＮ＊／
}
```

関数名　　指定ボードのビットリセット

記述　　int　A32X_06reset (bd,bit)；

引き数

int	bd；	ボード番号
int	bit；	ビット位置（０～３１）
unsigned long	size；	取り込みサイズ
float	＊data；	Ａ／Ｄ変換結果格納ポインタ

戻り値

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

説明　　指定ボードの出力を指定位置をＯＦＦにします。

使用例

```
#include        <a06bios.h>

#define         BD_BASE  0x900080       ／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);              ／＊初期化＊／
A32X_06reset (0,0) ;                        ／＊ビット０をＯＦＦ＊／
}
```

関数名　　指定データのＯＲ

記述　　int　A32X_06or (bd,data) ;

引き数　　int　　bd ;　　ボード番号
unsigned　long　data ;　　ＯＲするデータ

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

説明　　指定ボードの出力にデータをＯＲします。

使用例

```
#include　　<a06bios.h>

#define　　BD_BASE　0x900080　　／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);　　／＊初期化＊／
A32X_06or (0,0x55555555L) ;　　／＊55555555 をＯＲ＊／
}
```

関数名　　指定データの反転ＡＮＤ

記述　　int　A32X_06and (bd,data) ;

引き数　　int　　bd ;　　ボード番号
　　unsigned　long　　data ;　　反転ＡＮＤするデータ

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

説明　　指定ボードの出力に反転したデータをＡＮＤします。

使用例

```
#include　　<a06bios.h>

#define　　BD_BASE　0x900080　　／＊ボードベース＊／

void main(void)
{
A32X_06init(1,_ASYNC,BD_BASE);　　／＊初期化＊／
A32X_06or (0,0x55555555L) ;　　／＊AAAAAAAAをAND＊／
}
```

関数名　　ＰＩＯストローブの状態確認

記述　　int　A32X_06strb (bd,strb) ;

引き数　　int　　bd ;　　ボード番号
　　int　　＊strb ;　　ストローブ状態格納ポインタ
　　（_FALSE で未受信、_TRUE で受信）

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

説明　　指定ボードのストローブ受信状態の確認を行います。
　　初期化時のモードによって動作が異なります。
　　非同期モードの時は、常に_TRUE が返されます。
　　同期モードの時は、その時の状態が反映されます。

参照　　A32X_06in

使用例

```
#include        <a06bios.h>

#define         BD_BASE  0x900080       ／＊ボードベース＊／

void main(void)
{
int             strb;
unsigned  int   data;

A32X_06init(1,_SYNC,BD_BASE);           ／＊初期化＊／
do {                                    ／＊ストローブ受信待ち＊／
    A32X_06strb (0,&strb) ;
    } while (!strb) ;
A32X_06in (0,&data) ;                   ／＊データ入力＊／
}
```

関数名　　データの入力

記述　　int　A32X_06in (bd,data) ;

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

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

説明　　指定ボードから入力します。
初期化時のモードによって動作が異なります。
非同期モードの時は、関数が実行されるたびに入力を行います。
同期モードの時は、ストローブ受信時のみに入力値が更新されます。

参照　　A32X_06strb

使用例

```
#include        <a06bios.h>

#define         BD_BASE  0x900080       ／＊ボードベース＊／

void main(void)
{
int             strb;
unsigned  int   data;

A32X_06init(1,_SYNC,BD_BASE);           ／＊初期化＊／
do {                                    ／＊ストローブ受信待ち＊／
    A32X_06strb (0,&strb) ;
    } while (!strb) ;
A32X_06in (0,&data) ;                   ／＊データ入力＊／
}
```

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

ＡＤＳＰ３２４－０６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、ＡＤＳＰ３２４－０６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。

下記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。

・本マニュアルの内容は製品の改良のため予告無しに
変更される事がありますので、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

ADSP324-06

ｿﾌﾄｳｪｱﾏﾆｭｱﾙ

2002.12　　第４版発行