ＡＤＳＰシリーズ

ＡＤＳＰ６７４－００

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」.　付属ソフトウェア

⑴.　概要

付属ソフトウェアは、ＡＤＳＰ６７４－００に標準で添付されているソフトウェアです。　このソフトウェアは、ＡＤＳＰ６７４－００ボードを、ユーザーアプリケーションから操作するための橋渡しを行うソフトウェアで、ＡＤＳＰ６７４－００ボードにユーザープログラムをダウンロードしたり、メモリの操作を行うために利用します。

⑵.　ドライバー設定プログラム

デバイスドライバーの設定を、表示・変更するためのプログラムです。
ＡＤＳＰ６７４－００ボードの設定を出荷時から変更した場合に使用します。

⑶.　ユーティリティープログラム

Ｗｉｎｄｏｗｓ９５/９８/ME/ＮＴ４．０/２０００/ＸＰ上からＡＤＳＰ６７４－００ボードをコントロールするユーティリティープログラムです。
ＡＤＳＰ６７４－００ボードにユーザープログラムのダウンロード、実行、ＲＯＭ化、ＤＳＰメモリ内容の表示を行うなどの機能を持っています。

⑷.　付属ライブラリー

ホスト上で動作するユーザープログラムから、ＡＤＳＰ６７４－００ボードをコントロールしたり、ＤＳＰメモリへアクセスするなどの各種機能を実現するためのＤＬＬです。
Ｖｉｓｕａｌ　Ｃ＋＋およびＶｉｓｕａｌ　Ｂａｓｉｃから使用することができます。

⑸.　デバイスドライバー

ＡＤＳＰ６７４－００ボードをＷｉｎｄｏｗｓ９５/９８/ME/ＮＴ４．０/２０００/ＸＰ上から使用するための仮想デバイスドライバーで、ボードを制御するために必ず必要になるものです。
設定を行うツールとして、ドライバー設定プログラムが用意されています。

(6). フォルダー構造

サポートソフトウェアはセットアップ後、下記のフォルダー構造に格納されます。

```
¥ｾｯﾄｱｯﾌﾟ ﾃﾞｨﾚｸﾄﾘ (ﾃﾞﾌｫﾙﾄ C:¥ADSP67X)
├──Fftlib
│      └── readme.txt          補足説明
├──Help
│      ├── A67Xtool.cnt        サポートソフトウェア Help Contents File
│      └── A67Xtool.hlp        サポートソフトウェア Help File
├──Lib                         ＩＳＡ版ライブラリ
│      ├── Dsp
│      │      └── A674.cmd     リンカーコマンドファイル
│      └── Host
│             ├── A67X32.lib   Ｖｉｓｕａｌ Ｃ用インポートライブラリー
│             ├── A67xdll.bas  Ｖｉｓｕａｌ Ｂａｓｉｃ用標準モジュール
│             └── A67xdll.h    Ｖｉｓｕａｌ Ｃ用インクルードファイル
├──driver                      ＵＳＢドライバー
└──Sample
       ├── Dsp                 ＤＳＰサンプルプログラム
       └── Host
              ├── Vb           Ｖｉｓｕａｌ Ｂａｓｉｃ サンプルプログラム
              └── Vc           Ｖｉｓｕａｌ Ｃ サンプルプログラム
```

９５／９８／ＭＥの場合

```
¥Windows
└── SYSTEM
       ├── A67X32.VXD          デバイスドライバ
       └── A67X32.DLL          ライブラリ（ＤＬＬ）
```

ＮＴ４．０／２０００／ＸＰの場合

```
¥Windows
└── SYSTEM32
       ├── A67X32.DLL          ライブラリ（ＤＬＬ）
       └── DRIVERS
              └── A67xnt.sys   デバイスドライバ
```

「2」.　セットアップ

(1).　ハードウェアのセットアップ

ボードの実装方法は、下記の手順で行ってください。

1).リソースの確認

９５／９８／MEの場合

1. デスクトップ上の「マイコンピュータ」のプロパティーを開き、「デバイスマネージャ」タグを選択します。
2. 「コンピュータ」をダブルクリックして「コンピュータのプロパティー」を開きます。
3. 「Ｉ/Ｏポートアドレス」を選択し 、Ｉ/Ｏポート空き領域を確認します。(ﾎﾞｰﾄﾞ初期設定 300H)
4. 「メモリ」を選択し 、メモリ空き領域を確認します。　(ﾎﾞｰﾄﾞ初期設定 E0000H)
5. 「割り込み要求（IRQ)」を選択し、５・９・１０・１１・１２および１５の何れかが空きであることを確認します。空いていない場合は、上記の割り込みリソースを使用しているボードの設定を変更して空きにするか、ボードを外してください。　(ﾎﾞｰﾄﾞ初期設定 10)
6. 3～5 の設定については「ＡＤＳＰ６７４－００ スタートアップガイド」を参照して下さい。

ＮＴ４．０の場合

1. タスクバーのスタート～プログラム～管理ツール～WindowsNT 診断ﾌﾟﾛｸﾞﾗﾑを開きます。
2. リソース」タグを選択します。
3. 「Ｉ/Ｏポート」を選択し、Ｉ/Ｏポート空き領域を確認します。(ﾎﾞｰﾄﾞ初期設定 300H)
4. 「メモリ」を選択し、メモリ空き領域を確認します。(ﾎﾞｰﾄﾞ初期設定 E0000H)
5. 「ＩＲＱ」を選択し、５・９・１０・１１・１２および１５の何れかが空きであることを確認します。空いていない場合は、上記の割り込みリソースを使用しているボードの設定を変更して空きにするか、ボードを外してください。(ﾎﾞｰﾄﾞ初期設定 10)
6. 3～5 の設定については「ＡＤＳＰ６７４－００ スタートアップガイド」を参照して下さい。

２０００／ＸＰの場合

1. デスクトップ上の「マイコンピュータ」のプロパティを開き、「ハードウェア」タグを開き、「デバイスマネージャ」ボタンを選択します。
2. メニューの表示～リソースを選択します。
3. 「入出力（Ｉ／Ｏ)」を選択し、Ｉ／Ｏポート空き領域を確認します。(ﾎﾞｰﾄﾞ初期設定 300H)
4. 「メモリ」を選択し、メモリ空き領域を確認します。(ﾎﾞｰﾄﾞ初期設定 E0000H)
5. 「割り込み要求（ＩＲＱ)」を選択し、５・９・１０・１１・１２および１５の何れかが空きであることを確認します。空いていない場合は、上記の割り込みリソースを使用しているボードの設定を変更して空きにするか、ボードを外してください。(ﾎﾞｰﾄﾞ初期設定 10)
6. 3～5 の設定については「ＡＤＳＰ６７４－００スタートアップガイド」を参照して下さい。

2)．ボードの実装

1. 1)．リソースの確認の項で確認した空きリソースに合致するように、ボードのディップ・スイッチの設定を行ってください。(設定の方法は、「ハードウェアマニュアル」を参照してください。)
2. パソコン本体に実装する場合も、拡張ラックへ実装する場合も、空いているＩＳＡスロットへ実装してください。(拡張ラックを使用している場合は、ＤＳＰボードの設定に合わせた、メモリ・Ｉ/Ｏ範囲、および、ＩＲＱの設定が必要になる場合があります。)

3)．動作確認

1. ユーティリティーを使用して動作を確認しますので、続くサポートソフトウェアのセットアップおよびデバイスドライバーの設定を行ってください。
2. ユーティリティーを実行し、ＤＳＰ接続一覧ウィンドウに表示されているＤＳＰボードが、実際にパソコンに接続されているＤＳＰボードの数と一致することを確認してください。

(2). サポートソフトウェアのセットアップ

ＡＤＳＰ６７４－００サポートソフトウェアを使用するに当たって、ＡＤＳＰ６７４－００ボードが､パソコンに正しくセットアップされている必要があります｡ＤＳＰボードのセットアップ方法は、「２」-（１）ハードウェアのセットアップの項を参照してください。

サポートソフトウェアのセットアップは、下記の手順で行います。

1). ＣＤ－ＲＯＭドライブに、サポートソフトウェアのディスクを挿入します。
2). 自動でセットアッププログラムが起動されます。
3). 自動で起動されない場合は、「スタート」→「ファイル名を指定して実行」を選択し、名前の欄に“[DRIVE]:¥setup.exe”を入力し、“ＯＫ”を押してください。[DRIVE]にはＣＤ－ＲＯＭドライブ名を入れてください。

セットアッププログラムが起動されたら、セットアッププログラムの指示に従って、セットアップを完了します。

(3). デバイスドライバーの設定

ボードの設定を出荷時設定から変更して実装した場合は、セットアップ終了後に下記の手順でデバイスドライバーの設定を変更する必要があります。

出荷時設定のままで実装されている場合は、この項は必要ありませんので、読み飛ばしていただいて結構です。

1). 「スタート」→「プログラム」→「ADSP67X 32Bit Driver」→「ADSP67X ドライバー設定」の順で選択し、ドライバー設定プログラムを起動してください。

2). 図２－１のようなダイアログが表示されますので、ボードのディップスイッチの設定と合うように設定を行い「設定」を押します。

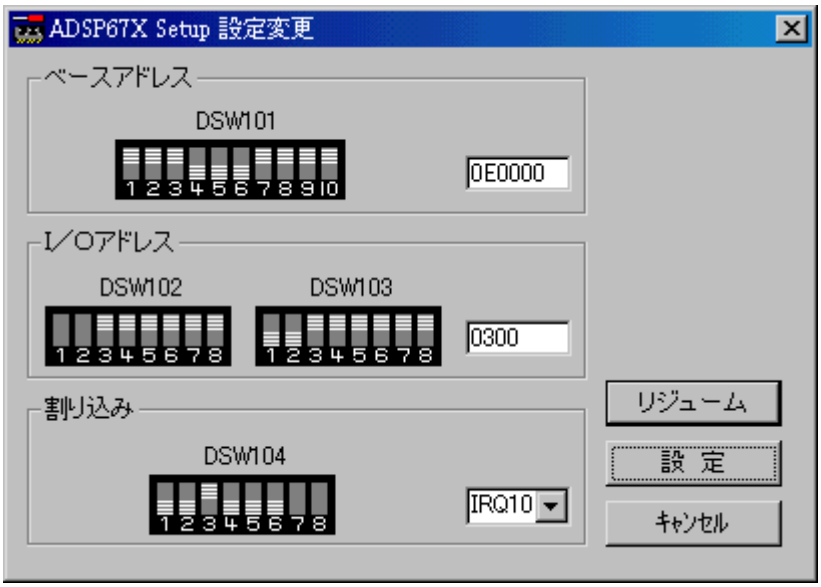


図２－１

3)． 設定を変更した場合、図２－２のようなダイアログが表示されますので、「設定」を押します。

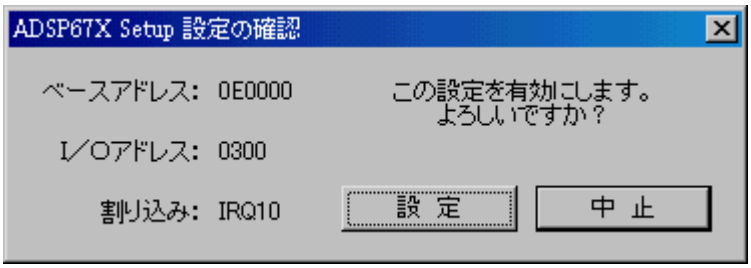


図２－２

4)． 図２－３のようなダイアログが表示されますので、「今、再起動をする。」か「後で、再起動をする。」かを選択して、「OK」を押してください。
「今、再起動をする。」を選択した場合は、直ちにＷｉｎｄｏｗｓが再起動されます。
「後で、再起動をする。」を選択した場合は、Ｗｉｎｄｏｗｓの再起動は行われませんので、マニュアルで再起動を行ってください。
注）再起動しないと、設定が反映されません。

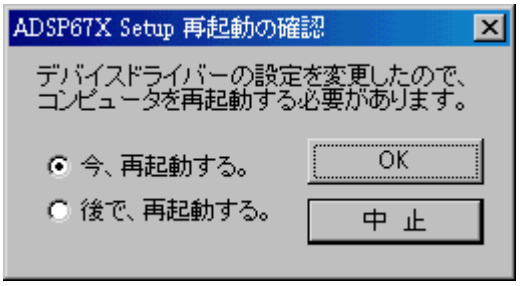


図２－３

(4). アンインストール

1). 本ソフトウェアをアンインストールするには、「スタート」→「設定」→「コントロール パネル」の順でクリックし、コントロール パネルを開きます。

2). 「アプリケーションの追加と削除」をダブルクリックします。

3). 「セットアップと削除」タグを選択し、一覧の中から「ADSP67X 32Bit Support Driver *.**.**」を選択し、「追加と削除」ボタンをクリックします。

4). 図２－４のような確認のダイアログが表示されますので、「はい」をクリックします。

図２－４

5). サポートソフトウェアがアンインストールされ、アンインストールが完了したことを伝える、ダイアログが表示されますので、「ＯＫ」をクリックして終了です。

「3」.　ユーティリティー

(1).　概要

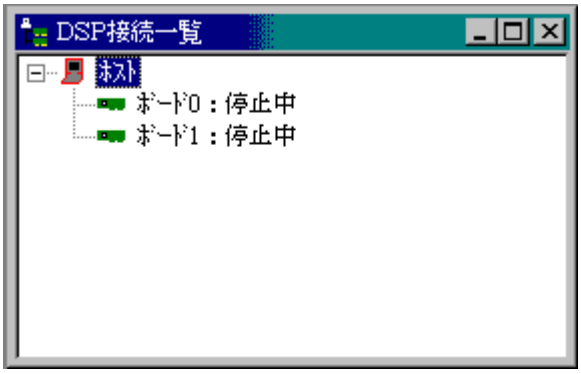
ユーティリティーは、ユーザープログラムのダウンロード、実行、ＲＯＭ化、ＤＳＰメモリ内容の表示を行うための、Ｗｉｎｄｏｗｓ上で動作するアプリケーションです。

(2).　ウィンドウ

ウィンドウは、２種類存在します。

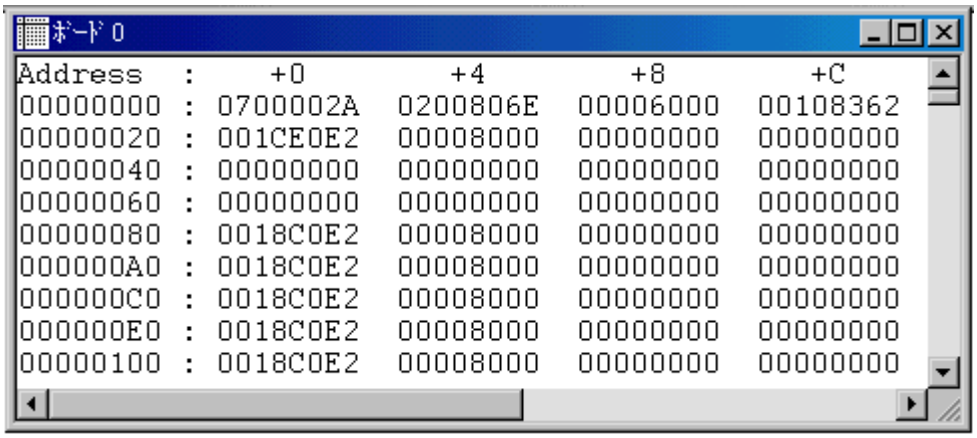
1).ＤＳＰ接続一覧ウィンドウ

ユーティリティーを起動すると最初に開くウィンドウを、「ＤＳＰ接続一覧ウィンドウ」と呼びます。
このウィンドウには、現在パソコンに接続されているＡＤＳＰ６７４－００ボードの接続状態が、ツリー形式で表示されます。
ユーティリティーのコマンドは、このウィンドウで選択したＤＳＰボードに対して実行されるようになっています。　そのため、このウィンドウでＤＳＰボードを選択していない場合は、メニューコマンドのほとんどが選択不可能な状態になっています。　コマンドを実行するには、このウィンドウでコマンドを発行するＤＳＰボードを選択する必要があります。
※ＤＳＰが表示されない場合は、デバイスドライバの設定とＤＳＰボードの設定が、食い違っている可能性がありますので、設定を確認してください。



2).メモリ内容表示ウィンドウ

ＤＳＰ接続一覧ウィンドウでＤＳＰボードを選択した後、[メモリの表示] コマンドを実行すると表示されるウィンドウを、「メモリ表示ウィンドウ」と呼びます。
このウィンドウには、ＤＳＰボードのメモリ内容を５種類の表示形式で表示することができます。
表示形式の変更は、「表示形式の設定」コマンドを選んでください。



(3). メニューの説明

メニューは、トップレベルにあるウィンドウによって、２種類のメニューが存在します。

1). ＤＳＰ接続一覧ウィンドウがトップレベルにある場合

「ファイル（F)」
- 「ライブラリーの再初期化（L)」
- 「ＤＳＰの初期化（I)」
- 「実行ファイルのロード（O)」
- 「ロードファイルの実行（E)」
- 「実行ファイルのＲＯＭ化（R)」
- 「リセントファイル」
- 「終了（X)」

「表示（V)」
- 「ツールバー（T)」

「操作（O)」
- 「メモリの表示（V)」
- 「メモリのフィル（F)」
- 「メモリの移動（M)」

「ウィンドウ（W)」
- 「重ねて表示（C)」
- 「上下に並べて表示（H)」
- 「左右に並べて表示（V)」
- 「アイコンの整列（A)」

「ヘルプ（H)」
- 「トピックの検索（H)」
- 「バージョン情報（A)」

2). メモリ内容表示ウィンドウがトップレベルにある場合

「ファイル」
- 「閉じる（C)」
- 「終了（X)」

「表示（V)」
- 「表示アドレスの設定（A)」
- 「表示形式の設定（F)」
- 「ツールバー（T)」

「ウィンドウ（W)」
- 「重ねて表示（C)」
- 「上下に並べて表示（H)」
- 「左右に並べて表示（V)」
- 「アイコンの整列（A)」

「ヘルプ（H)」
- 「トピックの検索（H)」
- 「バージョン情報（A)」

(4). ツールバーの説明

ツールバーは、ユーティリティーのコマンドをワンタッチで呼び出すためのボタンです。 ツールバーの配置は、図３－１のようになっています。

図３－１

ツールバーのボタンの配置とユーティリティーのコマンドとの対応は、左から
「ライブラリーの再初期化」
「ＤＳＰ接続一覧ウィンドウを開く」

「ＤＳＰの初期化」
「実行ファイルのロード」
「ロードファイルの実行」
「実行ファイルのＲＯＭ化」

「メモリの表示」
「表示アドレスの設定」
「表示形式の設定」

「メモリのフィル」
「メモリの移動」

「ウィンドウを重ねて表示」
「ウィンドウを上下に並べて表示」
「ウィンドウを左右に並べて表示」
「アイコンの整列」

「バージョン情報」
「ポップヒント」

となっています。

無効（コマンドが実行不可能な状態）の場合は、ボタン部分が灰色になります。
各コマンドの詳細に付いては、それぞれのコマンド説明を参照してください。

(5). 「ファイル」メニュー

1). ライブラリーの再初期化

このコマンドは、ライブラリーの再初期化を行うコマンドです。

ユーティリティーを使用中に、ＤＳＰがデッドロック状態や暴走してしまった場合は、ＤＳＰを完全に初期状態に戻すためにこの機能を実行します。

このコマンドを選択すると、接続されている全てのＤＳＰを初期状態に戻します。

2). ＤＳＰの初期化

このコマンドは、ＤＳＰの初期化を行うコマンドです。

ＤＳＰ接続一覧ウィンドウより初期化を行いたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では無効になっています。

このコマンドを選択すると、選択されたＤＳＰのみを初期化します。

3). 実行ファイルのロード

このコマンドは、ＤＳＰメモリにユーザープログラムのダウンロードを行うコマンドです。

ＤＳＰ接続一覧ウィンドウより、ユーザープログラムのダウンロードを行いたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドでロードできるファイルは、ＴＩ社製のＣコンパイラであるＴＭＳ３２０６Ｘで作成された、ＤＳＰ用の実行ファイル形式（拡張子″.out″）です。

このコマンドを選択すると、ファイルの選択ダイアログが表示されますので、ダウンロードしたいユーザープログラムを選択して「開く」を、ダウンロードしない場合は「キャンセル」を押してください。「開く」を押した場合は、選択されたユーザープログラムのダウンロードを開始します。

4). ロードファイルの実行

このコマンドは、ダウンロードされたユーザープログラムの実行を行うコマンドです。

ＤＳＰ接続一覧ウィンドウより、ユーザープログラムの実行を行いたいＤＳＰを選択し、すでにユーザープログラムがダウンロードされていることで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドを選択すると、実行ダイアログが表示されますので、実行する場合は「ＯＫ」を、実行しない場合は「キャンセル」を押してください。「ＯＫ」を押した場合はダウンロードされたユーザープログラムを実行します。

5). 実行ファイルのＲＯＭ化

このコマンドは、ユーザープログラムのＲＯＭ化を行うコマンドです。ユーザープログラムをＤＳＰボード上のＲＯＭへ書き込み、スタンドアロン動作を可能にします。

ＤＳＰ接続一覧ウィンドウより、ユーザープログラムのＲＯＭ化を行いたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドでロードできるファイルは、ＴＩ社製のＣコンパイラであるＴＭＳ３２０６Ｘで作成された、ＤＳＰ用の実行ファイル形式（拡張子″.out″）です。

このコマンドを選択すると、ファイルの選択ダイアログが表示されますので、ＲＯＭ化したいユーザープログラムを選択して「開く」を、ＲＯＭ化しない場合は「キャンセル」を押してください。「開く」を押した場合は、ＲＯＭ化を開始し、ＲＯＭ化実行ステータスダイアログが表示されます。途中で中断したい場合は「中止」ボタンを押してください。

6). リセントファイル

ＤＳＰ接続一覧ウィンドウを選択時は、「ファイル」メニューに過去にロードしたことのある、ユーザープログラムの一覧がリストアップされています。　この一覧のファイルを選択することにより、ＤＳＰ接続一覧ウィンドウで選択されているＤＳＰに、ユーザープログラムをダウンロードすることが可能です。

7). 閉じる

このコマンドは、選択されているウィンドウがメモリ表示ウィンドウの場合に有効になります。
このコマンドを選択すると、選択されているメモリ表示ウィンドウを閉じます。

8). 終了

ユーティリティープログラムを終了します。

(6). 「表示」メニュー

1). 表示アドレスの設定

このコマンドは、メモリ表示ウィンドウに表示するメモリアドレスの設定を行うコマンドです。

メモリ表示ウィンドウを選択している場合に有効なコマンドで、それ以外のウィンドウを選択している場合は、無効になっています。

このコマンドを選択すると、図３－２のようなダイアログが表示されますので、表示したいメモリ領域を選択し、その領域内の表示開始オフセットアドレスを設定して、「OK」を押してください。表示アドレスを変更したくない場合は、「キャンセル」を押してください。

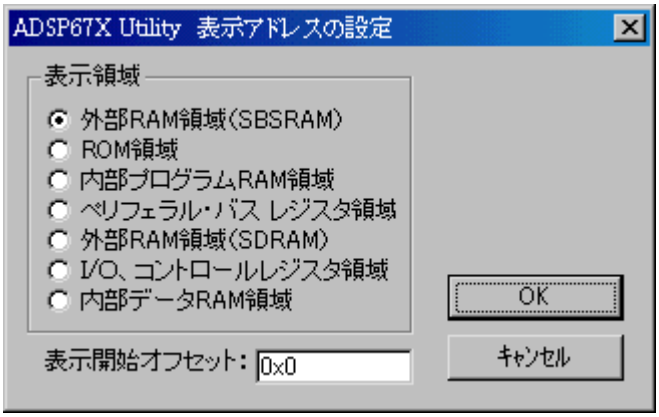


図３－２

2). 表示形式の設定

このコマンドは、メモリ表示ウィンドウに表示するメモリ内容の形式設定を行うコマンドです。

メモリ表示ウィンドウを選択している場合に有効なコマンドで、それ以外のウィンドウを選択している場合は、無効になっています。

このコマンドを選択すると、図３－３のようなダイアログが表示されますので、表示する書式を選択して、「OK」を押してください。表示形式を変更したくない場合は、「キャンセル」を押してください。

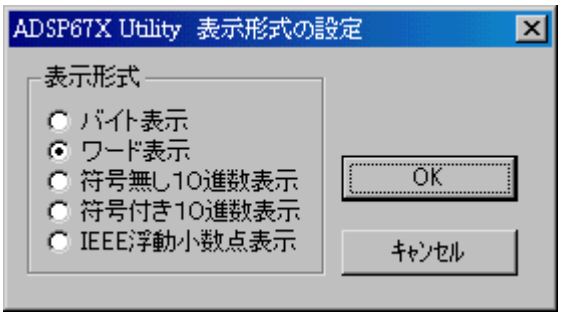


図３－３

3). ツールバー

このコマンドは、ツールバーの表示・非表示を切り替えます。

ツールバーには、ユーティリティーでよく使われるコマンドと同じコマンドを持ったツールが含まれています。ツールバーが表示されている時は、このコマンド名の横にチェックマークが表示されます。

(7). 「操作」メニュー

1). メモリの表示

このコマンドは、ＤＳＰのメモリ内容の表示を行うコマンドです。

ＤＳＰ接続一覧ウィンドウより、メモリ内容の表示を行いたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドを選択すると、選択されているＤＳＰのメモリ内容を表示するメモリ表示ウィンドウを開きます。　ウィンドウを開いた直後は、ＤＳＰのローカルメモリ領域の先頭から表示されます。表示形式はワード表示です。

2). メモリのフィル

このコマンドは、ＤＳＰの指定メモリ範囲に、任意のデータでフィルを行うコマンドです。

ＤＳＰ接続一覧ウィンドウより、メモリのフィルを行いしたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドを選択すると、図３－２のようなダイアログが表示されますので、フィル領域、先頭オフセット、サイズ、データ（フィルデータ)、データ増加幅（１ワード変化するごとにデータ増加幅で指定した値が加算されます）を設定して、「ＯＫ」ボタンを押してください。フィルを中止したい場合は、「キャンセル」を押してください。

図３－４

3)．メモリの移動

このコマンドは、ＤＳＰの特定のメモリ領域から、特定のメモリ領域へ、データの転送を行うコマンドです。

ＤＳＰ接続一覧ウィンドウより、メモリ内容の転送を行いたいＤＳＰを選択することで有効になります。　それ以外の場所や、メモリ表示ウィンドウを表示している状態では、無効になっています。

このコマンドを選択すると、図３－２のようなダイアログが表示されますので、転送元の領域および先頭オフセット、転送先の領域および先頭オフセット、転送データサイズを設定して、「ＯＫ」を押してください。転送を中止したい場合は、「キャンセル」を押してください。

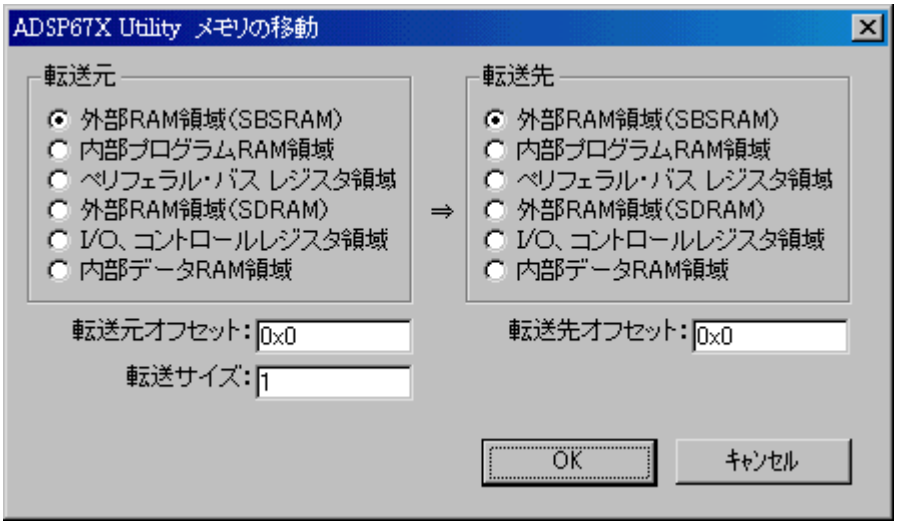


図３－５

(8). 「ウィンドウ」メニュー

1). 重ねて表示

このコマンドは、現在表示されている複数のウィンドウを、タイル状に重ねて表示します。

2). 上下に並べて表示

このコマンドは、現在表示されている複数のウィンドウを、縦に整列させて表示します。

3). 左右に並べて表示

このコマンドは、現在表示されている複数のウィンドウを、横に整列させて表示します。

4). アイコンの整列

このコマンドは、アプリケーションのメインウィンドウの最下部にアイコン化されているウィンドウを整列させます。もし、ウィンドウが開いていて、メインウィンドウの最下部にかかっている場合、このウィンドウの下になっているアイコンは見えません。

5). ウィンドウ一覧

アプリケーションが、現在開いているウィンドウのリストが「ウィンドウ」メニューの最下部に表示されます。　また、現在アクティブになっているウィンドウ名の横にチェックマークが表示されます。　このリストアップされている項目を選択することにより、そのウィンドウをアクティブにすることができます。

(9). 「ヘルプ」メニュー

1). トピックの検索

このコマンドは、ヘルプのトピックを表示します。

2). バージョン情報

このコマンドは、アプリケーションのバージョンや版権などについての情報を表示します。

(10). その他のコマンド

1). ＤＳＰ接続一覧ウィンドウを開く

このコマンドは、ＤＳＰ接続一覧ウィンドウを表示、または、トップレベルへ持ってきます。

2). ポップヒント

このコマンドは、アプリケーションのさまざまな部分についてのヘルプ情報を表示します。ツールバーの「ポップヒント」を選択すると、マウスポインタの形が矢印と疑問符を組み合わせたものに変わります。この状態で、ツールバーの他のボタン、アプリケーションのウィンドウ部分などをクリックすると、その部分に対応したヘルプ情報が表示されます。

(11). ＤＳＰメモリの予約領域

ユーティリティーを使用時、ＤＳＰとの通信等を行うために、ＤＳＰにモニタープログラムをロードし、実行させています。そのため、ＤＳＰメモリのある領域へアクセス（メモリへの書き込み）を行うと、ＤＳＰが正しく動作しないほか、思わぬ動作をする場合がありますので注意してください。

使用しているメモリ領域は、表３－１のとおりです。

ユーザープログラムをリンクする際には、モニターとの領域が重ならないように注意してください。

範　　囲	用　　途
００００００００ｈ～０００００３ＦＦｈ	リセットベクタ領域
００００４００ｈ～００００２ＦＦＦｈ	モニター・プログラム領域 ＲＯＭブート・ローダー領域
００００３０００ｈ～００００３ＦＦＦｈ	モニター・通信領域

表３－１

これらの予約領域は、弊社で定めた予約領域です。

これらの予約領域は、付属のリンカーコマンドファイル（A674.CMD）を使用する場合は、特に注意する必要はないと思われます。

モニター・プログラム領域およびモニター・データ領域は、ユーティリティーの実行時のみ必要となり、ＲＯＭブート・ローダー領域は、スタンドアロン機能の実行時のみ必要となります。

従って、ユーティリティーおよびスタンドアロン機能を使用されない場合は、この領域を使用されてもかまいません。

これ以外にＴＭＳ３２０Ｃ６７０１で予約されている領域もありますので、ＴＭＣ３２０Ｃ６７０１ユーザーズマニュアルを参照してください。

「4」.　付属ライブラリー

(1).　概要

この付属ライブラリーは、Ｗｉｎｄｏｗｓ９５/９８/ME/NT４.０/２０００/ＸＰ上で動作するユーザープログラムからＤＳＰを制御するために使用するライブラリーです。　ＤＳＰの初期化、実行ファイルのロード、メモリの読み書きなどの機能をサポートしています。

対応する開発言語は、

ＭｉｃｒｏｓｏｆｔのＶｉｓｕａｌ　Ｃ++　Ｖｅｒｓｉｏｎ４.ｘ以降の製品

Ｖｉｓｕａｌ　Ｂａｓｉｃ　Ｖｅｒｓｉｏｎ４.ｘ以降の製品

となっています。

Ｖｉｓｕａｌ　Ｃ++で使用する場合は、ユーザープログラムの先頭で、プロトタイプ宣言ファイルである“A67XDLL.H”をインクルードすることと、ユーザーのプロジェクトへ、インポートライブラリーの“A67X32.LIB”を追加する必要があります。

また、Ｖｉｓｕａｌ　Ｂａｓｉｃでは、ユーザーのプロジェクトファイルに、標準モジュールの“A67XDLL.BAS”を、追加する必要があります。

具体的な手順は、下記のようになります。

1).Ｖｉｓｕａｌ　Ｃ++でライブラリーを利用する方法

ＤＳＰのライブラリーを使用するソースファイルの先頭で#include 命令を使用して、“A67XDLL.H”ファイルを取り込みます。

プロジェクトに、インポート・ライブラリーを追加するために、メニューの「プロジェクト」→「プロジェクトへ追加」→「ファイル」を選択し、“A67X32.LIB”を追加してください。　この時のファイルの種類は、ライブラリファイル（lib）を、選択している必要があります。

2).Ｖｉｓｕａｌ　Ｂａｓｉｃでライブラリーを利用する方法

メニューの「プロジェクト」→「標準モジュールの追加」を選択し、「既存のファイル」タグで“A67XDLL.BAS”を追加します。

※　開発ツールによっては、上記手順と操作が異なる可能性がありますので、それぞれの開発環境の取り扱い説明書を参照の上、各操作を行ってください。

⑵．　レジストリ

ＡＤＳＰ６７４－００付属ライブラリーでは、以下のレジストリを使用しています。

９５／９８／MEの場合

HKEY_LOCAL_MACHINE¥Software¥Chubu Electric Co.,Ltd.¥ADSP67X 32Bit Support Driver¥Settings¥IOBase
HKEY_LOCAL_MACHINE¥Software¥Chubu Electric Co.,Ltd.¥ADSP67X 32Bit Support Driver¥Settings¥IRQNumber
HKEY_LOCAL_MACHINE¥Software¥Chubu Electric Co.,Ltd.¥ADSP67X 32Bit Support Driver¥Settings¥SegmentBase
HKEY_LOCAL_MACHINE¥Software¥Chubu Electric Co.,Ltd.¥ADSP67X 32Bit Support Driver¥Settings¥Version

ＮＴ４．０／２０００／ＸＰ／Ｖｉｓｔａの場合

HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥A67xnt¥a67xnt0¥Parameters¥IOBase
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥A67xnt¥a67xnt0¥Parameters¥IRQNumber
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥A67xnt¥a67xnt0¥Parameters¥SegmentBasE
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥A67xnt¥a67xnt0¥Parameters¥Version

それぞれの意味は下記のとおりで、バイナリ（ＬＳＢファースト）４桁で記述されています。

IOBase　　：ＤＳＰのＩ/Ｏアドレスで、ＤＳＰボードの設定値と同一内容を示します。
IRQNumber　：割り込み番号で、ＤＳＰボード設定値と同一内容を示します。
SegmentBase：メモリベースアドレスで、ＤＳＰボード設定値を１/１６倍した値
　　（１６進数値では、下１桁を切り取った値）を示します。
Version　　：本ライブラリーのバージョンを示します。

(3). 関数一覧

1). C関数

ここでは、型の表記を以下のように簡略化しています。

V　：　void
C　：　char
I　：　int
L　：　long
UI　：　unsigned int
UL　：　unsigned long

1. ライブラリー制御関数

int	A67X_libenter(V);	デバイスドライバーのオープン
void	A67X_libexit(V);	デバイスドライバーのクローズ
void	A67X_getversion(*UL);	デバイスドライバーのバージョン取得(＊1)
int	A67X_getboardsetup(*I,*I,*I);	デバイスドライバーの設定値取得(＊1)
int	A67X_setboardsetup(I,I,I);	デバイスドライバーの設定値登録(＊1)
int	A67X_resume();	デバイスドライバーの強制開放(＊1)

2. ボード関連関数

void	A67X_bdsel(I);	ボード選択
void	A67X_bdinit(I);	ボード初期化
void	_A67X_bdinit(V);	
int	A67X_valid(I);	ボード実装確認

3. ボード制御関数

int	A67X_run(I);	ボード実行開始
int	_A67X_run(V);	
int	A67X_reset(I);	ボードリセット
int	_A67X_reset(V);	
int	A67X_resetstus(I);	ボードリセット状態の確認
int	_A67X_resetstus(V);	
int	A67X_hold(I);	ボード停止
int	_A67X_hold(V);	
int	A67X_holdcancel(I);	ボード停止解除
int	_A67X_holdcancel(V);	
int	A67X_holdstus(I);	ボード停止状態の確認
int	_A67X_holdstus(V);	

4. アップ／ダウンロード関数

int	A67X_getmem(I,UL,*V);	メモリからのアップロード
int	_A67X_getmem(UL,*V);	
int	A67X_putmem(I,UL,*V,I);	メモリへのダウンロード
int	_A67X_putmem(UL,*V,I);	
int	A67X_ArrayGet(I,UL,L,*V);	メモリからのアップロード
int	_A67X_ArrayGet(UL,L,*V);	
int	A67X_ArrayPut(I,UL,L,*V,I);	メモリへのダウンロード
int	_A67X_ArrayPut(UL,L,*V,I);	
int	A67X_loadc(I,*C);	ＣＯＦＦファイルのダウンロード
int	_A67X_loadc(*C);	
int	A67X_boot(I,UL);	リセットベクタの設定
int	_A67X_boot(UL);	
int	A67X_loadcs(I,*C);	ＣＯＦＦファイルのダウンロードと
int	_A67X_loadcs(*C);	リセットベクタの設定
int	A67X_savec(I,UL,UL,UL,*C);	ＣＯＦＦファイルへのアップロード
int	_A67X_savec(UL,UL,UL,*C);	

5. 特殊制御関数

void	A67X_int4(I);	ＤＳＰボードへ割り込みＩＮＴ４発生
void	_A67X_int4(V);	
void	A67X_nmi(I);	ＤＳＰボードへ割り込みNＭＩ発生
void	_A67X_nmi(V);	
int	A67X_resetirqhndl(V);	ＤＳＰからの割り込みハンドラ解除(＊1)
int	A67X_setirqhndl(HWND,UINT,WPARAM,LPARAM);	ＤＳＰからの割り込みハンドラ設定(＊1)

6.　オフライン関数

int	A67X_entryc(*C*,*UL);	ＣＯＦＦファイルからエントリ取り出し
int	A67X_entrym(*C,*C,*UL);	マップファイルからエントリ取り出し
int	A67X_symbolc(*C,*C,*UL);	ＣＯＦＦファイルからシンボル取り出し
int	A67X_symbolm(*C,*C,*UL);	マップファイルからシンボル取り出し

関数名称の先頭に‘_’がある関数とない関数の違いは、ない関数はボード番号の指定が付いており、ある関数は現在のボード（A67X_bdsel で選択したボード）を対象に実行される関数です。

ＵＳＢ接続での注意
　関数説明の最後に”＊1”が付いているものはＵＳＢ接続では使用できません。

2). ＢＡＳＩＣサブルーチンおよびファンクション

ここでは、型の表記を以下のように簡略化しています。
AP ： ByRef ? As Any
S ： ByVal ? As String
L ： ByVal ? As Long
LP ： ByRef ? As Long

1. ライブラリー制御関数

Function A67X_libenter() As Long	デバイスドライバーのオープン
Sub A67X_libexit()	デバイスドライバーのクローズ
Sub A67X_getversion(LP)	デバイスドライバーのバージョン取得(＊1)
Function A67X_getboardsetup(LP,LP,LP) As Long	デバイスドライバーの設定値取得(＊1)
Function A67X_setboardsetup(L,L,L) As Long	デバイスドライバーの設定値登録(＊1)
Function A67X_resume()	デバイスドライバーの強制開放(＊1)

2. ボード関連関数

Sub A67X_bdsel(L)	ボード選択
Sub A67X_bdinit(L)	ボード初期化
Sub SA67X_bdinit()	
Function A67X_valid(L) As Long	ボード実装確認

3. ボード制御関数

Function A67X_run(L) As Long	ボード実行開始
Function SA67X_run() As Long	
Function A67X_reset(L) As Long	ボードリセット
Function SA67X_reset() As Long	
Function A67X_resetcancel(L) As Long	ボードリセット解除
Function SA67X_resetcancel() As Long	
Function A67X_resetstus(L) As Long	ボードリセット状態の確認
Function SA67X_resetstus() As Long	
Function A67X_hold(L) As Long	ボード停止
Function SA67X_hold() As Long	
Function A67X_holdstus(L) As Long	ボード停止状態の確認
Function SA67X_holdstus() As Long	

4. アップ／ダウンロード関数

Function A67X_getmem(L,L,LP) As Long	メモリへのアップロード
Function SA67X_getmem(L,LP) As Long	
Function A67X_putmem(L,L,L,L) As Long	メモリからのダウンロード
Function SA67X_putmem(L,L,L) As Long	
Function A67X_ArrayGet(L,L,L,LP) As Long	メモリへのアップロード
Function SA67X_ArrayGet(L,L,LP) As Long	
Function A67X_ArrayPut(L,L,L,LP,L) As Long	メモリからのダウンロード
Function SA67X_ArrayPut(L,L,LP,L) As Long	
Function A67X_loadc(L,S) As Long	ＣＯＦＦファイルのダウンロード
Function SA67X_loadc(S) As Long	
Function A67X_boot(L,L) As Long	リセットベクタの設定
Function SA67X_boot(L) As Long	
Function A67X_loadcs(L,S) As Long	ＣＯＦＦファイルのダウンロードと
Function SA67X_loadcs(S) As Long	リセットベクタの設定
Function A67X_savec(L,L,L,L,S) As Long	ＣＯＦＦファイルへのアップロード
Function SA67X_savec(L,L,L,S) As Long	

5. 特殊制御関数

Sub	A67X_int4(L)	ＤＳＰボードへ割り込みＩＮＴ４発生
Sub	SA67X_int4()	
Sub	A67X_nmi(L)	ＤＳＰボードへ割り込みＮＭＩ発生
Sub	SA67X_nmi()	

6.　オフライン関数

Function A67X_entryc(S,LP) As Long	ＣＯＦＦファイルからエントリ取り出し
Function A67X_entrym(S,S,LP) As Long	マップファイルからエントリ取り出し
Function A67X_symbolc(S,S,LP) As Long	ＣＯＦＦファイルからシンボル取り出し
Function A67X_symbolm(S,S,LP) As Long	マップファイルからシンボル取り出し

関数名称の先頭に‘S’があるファンクションおよびサブルーチンとない物の違いは、ない物はボード番号の指定が付いており、ある物は現在のボード（A67X_bdsel で選択したボード）を対象に実行されるファンクションおよびサブルーチンです。

ＵＳＢ接続での注意

関数説明の最後に”＊1”が付いているものはＵＳＢ接続では使用できません。

(4). 関数詳細

1). A67X_ArrayGet　　(メモリへのアップロード)

記述　：ＶＣ

```
int     A67X_ArrayGet(int board,                    ← 任意ボード
                      unsigned long top,
                      long size,
                      void *buffer);
int     _A67X_ArrayGet(unsigned long top,           ← 現在ボード
                       long size,
                       void *buffer);
```

ＶＢ

```
Function A67X_ArrayGet(ByVal board As Long,         ← 任意ボード
                       ByVal top As Long,
                       ByVal size As Long,
                       ByRef buffer As Any) As Long
Function SA67X_ArrayGet(ByVal top As Long,          ← 現在ボード
                        ByVal size As Long,
                        ByRef buffer As Any) As Long
```

引数　：board　　ボード番号
top　　アップロード開始アドレス
size　　アップロードサイズ（ワード単位)
buffer　　アップロードデータ格納配列

説明　：ＤＳＰのメモリからホストへのメモリへ、データをアップロードします。
データ長は任意サイズで、単位はワード（４バイト）単位です。

戻り値　：０　　正常終了しました。
－１　　アップロードサイズが０以下。
その他のエラー。(USB のみ)
－２　　通信タイムアウト

使用例　：

ＶＣ

```
unsigned long     buffer[0x100];
if(A67X_ArrayGet(0L, 0x1000L, 0x100L, buffer)){
    printf( “読出エラー.¥n” );
}
```

ＶＢ

```
Dim buffer(&H100) As Long
If A67X_ArrayGet(0, &H1000, &H100, buffer(0)) Then
    Print “読出エラー.”
End If
```

2). A67X_ArrayPut　　（メモリからのダウンロード）

記述　：ＶＣ　int　A67X_ArrayPut(int board,　← 任意ボード
unsigned long top,
long size,
void *buffer,
int verify = -1);
int　_A67X_ArrayPut(unsigned long top,　← 現在ボード
long size,
void *buffer,
int verify = -1);

ＶＢ　Function A67X_ArrayPut(ByVal board As Long,　← 任意ボード
ByVal top As Long,
ByVal size As Long,
ByRef buffer As Any,
Optional ByVal verify As Long = -1) As Long
Function SA67X_ArrayPut(ByVal As Long,　← 現在ボード
ByVal size As Long,
ByRef buffer As Any,
Optional ByVal verify As Long = -1) As Long

引数　：board　ボード番号
top　ダウンロード開始アドレス
size　ダウンロードサイズ（ワード単位）
buffer　ダウンロードデータ格納配列
verify　ダウンロードデータの照合選択
０　：照合しない
－１（デフォルト）：ＩＳＡ接続の場合　照合する
ＵＳＢ接続の場合　照合しない
０、－１以外　：照合する

説明　：ホストのメモリからＤＳＰのメモリへ、ダウンロードします。
データ長は任意サイズで、ワード（４バイト）単位です。

戻り値　：０　正常終了しました。
－１　ダウンロードサイズが０以下。
メモリへの書き込み異常（照合不一致）。
その他のエラー。（USB のみ）
－２　通信タイムアウト。

使用例　：

ＶＣ

```
unsigned long     buffer[0x100];
int               i;
for(i = 0;i < 0x100;++i) buffer[i] = i;
if(A67X_ArrayPut(0L,0x1000L,0x100L,buffer)){
    Printf(“書込エラー.¥n”);
}
```

ＶＢ

```
Dim buffer(&H100) As Long;
For i = 0 To &HFF
    buffer(i) = i
Next i
If A67X_ArrayPut(0,&H1000,&H100,buffer(0)) Then
    Print “書込エラー.”
End If
```

3). A67X_bdinit　　(ボード初期化)

記述　：ＶＣ　void　A67X_bdinit(int board);　← 任意ボード
　　void　_A67X_bdinit(void);　← 現在ボード

　　ＶＢ　Sub　A67X_bdinit(ByVal board As Long)　← 任意ボード
　　Sub　SA67X_bdinit()　← 現在ボード

引数　：board　　ボード番号

説明　：ＤＳＰボードの外部メモリ等の初期化設定を行います。
ボードリセット関数実行後は、この関数を実行してＤＳＰボードの初期化設定を行ってください。

戻り値　：ありません。

使用例　：

ＶＣ
```
#define  BD_NO    0
A67X_bdinit(BD_NO);
```

ＶＢ
```
Const BD_NO As Long = 0
Call A67X_bdinit(BD_NO)
```

4). A67X_bdsel　　(ボード選択)

記述　　: ＶＣ　　void　　A67X_bdsel(int board);

　　ＶＢ　　Sub　　A67X_bdsel(ByVal board As Long)

引数　　: board　　ボード番号

説明　　: アクセスの対象となるボードを選択します。
　　ボード指定機能の付いていない関数を使用するのに先立ち、対象ボードを決定します。

戻り値　: ありません。

使用例　: ボード番号０から３に同じプログラムをロードする例です。(エラー処理は省略)

ＶＣ

```
int   board;
for(board = 0;board < 4;++board){
    A67X_bdsel(board);
    _A67X_bdinit();
    _A67X_loadc("USER.OUT");
}
```

ＶＢ

```
Dim board As Long;
For board = 0 to 3
    Call A67X_bdsel(board)
    Call SA67X_bdinit
    Call SA67X_loadc("USER.OUT")
Next board
```

5). A67X_boot　　(リセットベクタの設定)

記述　：ＶＣ　int　A67X_boot(int board,　← 任意ボード
　　unsigned long address);
　　int　_A67X_boot(unsigned long address);　← 現在ボード

　　ＶＢ　Function A67X_boot(ByVal board As Long,　← 任意ボード
　　ByVal address As Long) As Long
　　Function SA67X_boot(ByVal address As Long) As Long　← 現在ボード

引数　：board　ボード番号
　　address　実行開始番地（エントリアドレス）

説明　：ＤＳＰボードのリセットベクターを設定します。
　　この関数を用いてＤＳＰの任意の番地から実行させることができます。
　　ユーザープログラムをロード後、この関数にてリセットベクタープログラムをロードします。

戻り値　：０　正常終了しました。
　　０以外　異常終了しました。 原因は、次のいずれかです。
　　１）メモリへの書込異常（照合不一致）。

使用例　：ユーザープログラムのロードから実行するまでの例です。(エラー処理は省略)

ＶＣ
```
unsigned long address;
_A67X_loadc(“USER.OUT”);
A67X_entryc(“user.out”,&address);
_A67X_boot(address);
_A67X_run();
```

ＶＢ
```
Dim address As Long
Call SA67X_loadc(“USER.OUT”);
Call A67X_entryc(“USER.OUT”,address);
Call SA67X_boot(address)
Call SA67X_run;
```

6). A67X_entryc　　（ＣＯＦＦファイルからエントリ取り出し）

記述　：ＶＣ　int　A67X_entryc(const char *path,
　　unsigned long *entry);

　　ＶＢ　Function A67X_entryc(ByVal path As String,
　　ByRef entry As long) As Long

引数　：path　　ＣＯＦＦファイル名
　　entry　　実行開始番地（エントリアドレス）を格納する変数

説明　：ＣＯＦＦファイルから、実行開始番地を取得します。
　　ＣＯＦＦファイルには、オブジェクトコードのほかにプログラム実行開始番地情報も格納されています。
　　これを取り出し、A67X_boot 関数よってＤＳＰに実行開始番地を設定できます。
　　マップファイルからも同様に実行開始番地を取り出すことができます。

戻り値　：０　　正常終了しました。
　　０以外　異常終了しました。 原因は次のいずれかです。
　　１）ファイルが見つからない。
　　２）ファイルが異常である。
　　３）ファイルはＣＯＦＦファイルではない。

参考　：A32X_entrym

使用例　：

ＶＣ
```
unsigned long　　address;
if(A67X_entryc("USER.OUT",&address)){
　　printf("エントリ取得エラー.¥n");
}else{ printf("Entry address is %08lX¥n",address)
}
```

ＶＢ
```
Dim address As Long
If A67X_entryc("USER.OUT",address) Then
　　Print "エントリ取得エラー."
Else:Print "Entry address is ";Hex$(address)
End If
```

7). A67X_entrym　　（マップファイルからエントリ取り出し）

記述　：ＶＣ　int　A67X_entrym(const char *path,
　　const char *entry_name,
　　unsigned long *entry);

　　ＶＢ　Function A67X_entrym(ByVal path As String,
　　ByVal entry_name As String,
　　ByRef entry As Long) As Long

引数　：path　　ＭＡＰファイル名
　　entry_name　　エントリーシンボル名を格納する変数
　　entry　　実行開始番地（エントリアドレス）を格納する変数

説明　：ＤＳＰプログラム開発時にリンカーの出力したＭＡＰファイルから、
　　エントリーポイントのシンボル名とアドレスを取得します。
　　これを取り出し、A67X_boot 関数よってＤＳＰに実行開始番地を設定できます。
　　ＣＯＦＦファイルからも同様に実行開始番地を取り出すことができます。

戻り値　：０　　正常終了しました。
　　０以外　異常終了しました。 原因は次のいずれかです。
　　１）ファイルが見つからない。
　　２）ファイルが異常である。
　　３）ファイルはＭＡＰファイルではない。

参考　：A32X_entryc

使用例　：

ＶＣ

```
char     entry_name[20];
unsigned long     address;
if(A67X_entrym("USER.MAP",entry_name,&address)){
       printf("エントリ取得エラー.¥n");
}else{ printf("Entry is [%s] %08lX¥n",entry_name,address);
}
```

ＶＢ

```
Dim entry_name As String
Dim address As Long
entry_name = String(255,vbNullChar)
If A67X_entry("USER.MAP",entry_name,address) Then
    Print "エントリ取得エラー."
Else
    Entry_name = Left(entry_name,InStr(entry_name,vbNullChar)-1)
    Print "Entry is [";entry_name;" ] " ;Hex$(address)
End If
```

8). A67X_getboardsetup　　(デバイスドライバーの設定値取得)

記述　：ＶＣ　int　A67X_getboardsetup(int *segment,
　　int *io,
　　int *irq);

　　ＶＢ　Function A67X_getboardsetup(ByRaf segment As Long,
　　ByRaf io As Long,
　　ByRef irq As Long) As Long

引数　：segment　　セグメントアドレス
　　io　　Ｉ／Ｏアドレス
　　irq　　割り込み番号

説明　：デバイスドライバーに設定されている各設定値を取得します。

戻り値　：０　　正常終了しました。
　　０以外　異常終了しました。

参考　：A67X_setboardsetup

使用例　：

ＶＣ
```
int　　Seg,IO,IRQ;
If(A67X_getboardsetup(&Seg,&IO,&IRQ)){
    printf("設定値取得エラー.¥n");
}
```

ＶＢ
```
Dim Seg As Long,IO As Long,IRQ As Long
If A67X_getboardsetup(Seg,IO,IRQ) Then
    Print "設定値取得エラー."
End If
```

9). A67X_getmem　　(メモリからのアップロード)

記述　：ＶＣ　int　A67X_getmem(int board,　← 任意ボード
　　unsigned long address,
　　void *data);
　　int　_A67X_getmem(unsigned long address,　← 現在ボード
　　void *data);

　　ＶＢ　Function A67X_getmem(ByVal board As Long,　← 任意ボード
　　ByVal address As Long,
　　ByRef data As Any) As Long
　　Function SA67X_getmem(ByVal address As Long,　← 現在ボード
　　ByRef data As Any) As Long

引数　：board　ボード番号
　　address　メモリのアドレス
　　data　メモリ内容格納変数

説明　：ＤＳＰのメモリからホストへのメモリへ、データをアップロードします。
　　データ長は１ワードのみです。

戻り値　：０　正常終了しました。
　　－１　その他のエラー。(USB のみ)
　　－２　通信タイムアウト

参考　：A67X_putmem

使用例　：

ＶＣ
```
unsigned long　data;
_A67X_getmem(0x1000L,&data);
```

ＶＢ
```
Dim data As Long
Call _A67X_getmem(&H1000,data)
```

10). A67X_getversion　　(デバイスドライバーのバージョン取得)

記述　　：ＶＣ　　void　　A67X_getversion(unsigned long *data);

　　ＶＢ　　Sub　　A67X_getversion(ByRef data As Long)

引数　　：data　　バージョン格納変数

説明　　：デバイスドライバーのバージョンを取得します。
　　（ Ver 1.23.45 の場合 0x12345 (&H12345) となります。）

戻り値　：ありません。

参考　　：A67X_setboardsetup

使用例　：

ＶＣ
```
unsigned long     ver;
A67X_getversion(&ver);
printf("Version is %lX¥n",ver);
```

ＶＢ
```
Dim ver As Long
A67X_getversion(ver)
Print "Version is "; Hex$(ver)
```

11). A67X_hold　　(ボード停止)

記述　：ＶＣ　int　A67X_hold(int board);　← 任意ボード
　　int　_A67X_hold(void);　← 現在ボード

　　ＶＢ　Function A67X_hold(ByVal board As Long) As Long　← 任意ボード
　　Function SA67X_hold() As Long　← 現在ボード

引数　：board　　ボード番号

説明　：ＤＳＰを停止状態にします。
この関数を実行すると、ＤＳＰ自体が停止するためメモリへのアクセス等ができなくなります。したがって、通常この関数を使用する必要はないと思われます。
停止状態の解除はA67X_holdcancel,A67X_run 関数にて行います。

注意　：ＤＳＰを停止状態にすると、ＳＤＲＡＭのメモリ内容が壊れる可能性があります。

戻り値　：０　　正常終了。
０以外　異常終了。

参考　：A67X_holdcancel,A67X_holdstus

使用例　：

ＶＣ
```
if(_A67X_hold()){
      printf("ボード停止失敗.¥n");
}else{ printf("ボード停止正常終了.¥n");
}
```

ＶＢ
```
If SA32X_hold Then
    Print "ボード停止失敗."
Else:Print "ボード停止正常終了."
End If
```

12). A67X_holdcancel　　(ボード停止解除)

記述　：ＶＣ　int　A67X_holdcancel(int board);　← 任意ボード
　　int　_A67X_hold(void);　← 現在ボード

　　ＶＢ　Function A67X_holdcancel(ByVal board As Long) As Long ← 任意ボード
　　Function SA67X_hold() As Long　← 現在ボード

引数　：board　　ボード番号

説明　：ＤＳＰを停止解除状態にします。
この関数を実行すると、ＤＳＰ自体の停止状態を解除し、メモリへのアクセス等ができるようになります。

戻り値　：０　　正常終了。
０以外　異常終了。

参考　：A67X_hold,A67X_holdstus, A67X_run

使用例　：

ＶＣ
```
if(_A67X_holdcancel()){
      printf("ボード停止解除失敗.¥n");
}else{ printf("ボード停止解除正常終了.¥n");
}
```

ＶＢ
```
If SA32X_holdcancel Then
    Print "ボード停止解除失敗."
Else:Print "ボード停止解除正常終了."
End If
```

13). A67X_holdstus　　(ボード停止状態の確認)

記述　：ＶＣ　int　A67X_holdstus(int board);　← 任意ボード
　　int　_A67X_holdstus(void);　← 現在ボード

　　ＶＢ　Function A67X_holdstus(ByVal board As Long) As Long　← 任意ボード
　　Function SA67X_holdstus() As Long　← 現在ボード

引数　：board　ボード番号

説明　：ボードの停止状態を取得します。
　　ボードが停止状態か、または、停止解除状態かを返します。

戻り値　：０　停止解除状態。
　　０以外　停止状態。

使用例　：

ＶＣ
```
if(_A67X_holdstus()){
      printf("ボード停止状態.¥n");
}else{ printf("ボード停止解除状態.¥n");
}
```

ＶＢ
```
If SA32X_holdstus Then
    Print "ボード停止状態."
Else:Print "ボード停止解除状態."
End If
```

14). A67X_int4　　（ＤＳＰボードへの割り込みＩＮＴ４発生）

記述　：ＶＣ　Void　A67X_int4(int board);　← 任意ボード
　　Void　_A67X_int4(void);　← 現在ボード

　　ＶＢ　Sub　A67X_int4(ByVal board As Long)　← 任意ボード
　　Sub　SA67X_int4()　← 現在ボード

引数　：board　　ボード番号

説明　：ホストからＤＳＰに対して、割り込みＩＮＴ４を発生します。
　　ホストからＤＳＰへの同期に利用できます。 ハードウェアによる割り込みです。

戻り値　：ありません。

参照　：A67X_nmi

使用例　：

ＶＣ　_A67X_int4();

ＶＢ　Call SA67X_int4

15). A67X_libenter　　(デバイスドライバーのオープン)

記述　：ＶＣ　int　A67X_libenter(void);

ＶＢ　Function A67X_libenter() As Long

引数　：ありません。

説明　：ライブラリの初期化と、仮想デバイスドライバーをオープンします。
本ライブラリのほかの機能を使用するに先立って、この関数を必ず実行してください。
この関数を実行することによって、実装されている全てのＤＳＰがリセットされ、初期化設定されます。

戻り値　：０　正常にオープンできました。
１　デバイスドライバーがオープンできません。
(A67X32.VXD が見つからない。)
２　デバイスがすでに使用されています。
(前回 A67X_libexit を実行せずに終了した。)
３　メモリが確保できませんでした。
(ＤＳＰのメモリ空間が、Ｗｉｎｄｏｗｓで使用不可能。)
４　有効なＤＳＰが１枚も見つかりません。
(デバイスドライバーの設定がＤＰＳボードの設定と合っていない。)

参照　：A67X_libexit

使用例　：

ＶＣ
```
if(A67X_libenter()){
    printf("ライブラリー初期化エラー.¥n");
    exit(1);
}
```

ＶＢ
```
If A67X_libenter Then
    Print "ライブラリー初期化エラー."
    End
End If
```

16). A67X_libexit　　(デバイスドライバーのクローズ)

記述　　：ＶＣ　　void　　A67X_libexit(void);

　　ＶＢ　　Sub　　A67X_libexit

引数　　：ありません。

説明　　：デバイスドライバーを開放します。
ユーザープログラムを終了する直前にこの関数を呼び出し、デバイスドライバーを開放する必要があります。　この関数を実行しないでプログラムを終了した場合、次回のA67X_libenter 関数の呼び出しが失敗する場合があります。
この場合、ドライバ設定ユーティティーの「リジューム」ボタンで、デバイスドライバーを強制開放してください。

戻り値　：ありません。

参考　　：A67X_libenter

使用例　：

ＶＣ　　A67X_libexit();

ＶＢ　　Call A67X_libexit

17). A67X_loadc　　（ＣＯＦＦファイルのダウンロード）

記述　：ＶＣ　int　A67X_loadc(int board,　← 任意ボード
　　const char *path);
　　int　_A67X_loadc(const char *path);　← 現在ボード

　　ＶＢ　Function A67X_loadc(ByVal board As Long,　← 任意ボード
　　ByVal path As String) As Long
　　Function SA67X_loadc(ByVal path As String) As Long　← 現在ボード

引数　：board　ボード番号
　　path　ＣＯＦＦファイル名

説明　：ＣＯＦＦフォーマットのオブジェクトファイルをＤＳＰのメモリにダウンロードします。
　　ロード可能なファイルは、リンカーが出力したＣＯＦＦファイルフォーマットの実行形式のモジュールです。
　　ロード可能な領域は、ＤＳＰのメモリ（０番地からメモリの実装されている範囲）です。
　　すでにロードされている領域への重複したロードの検出は行っていないので、ロードするアドレスには十分注意してください。

戻り値　：０　正常にロードが完了しました。
　　０以外　ロード異常です。 原因は次のいずれかです。
　　１）ファイルが見つからない。
　　２）ファイルはＣＯＦＦファイルではない。
　　３）ファイルが異常。
　　４）メモリ書き込み異常。

参考　：A67X_loadm, A67X_savec

使用例　：

ＶＣ
```
if(_A67X_loadc("USER.OUT")){
        printf("ロードエラー.¥n");
}else{ printf("ロード正常終了.¥n");
}
```

ＶＢ
```
If SA67X_loadc("USER.OUT") Then
     Print "ロードエラー."
Else:Print "ロード正常終了."
End If
```

18). A67X_loadcs　　（ＣＯＦＦファイルとリセットベクタのダウンロード）

記述　：ＶＣ　int　A67X_loadcs(int board,　← 任意ボード
const char *path);
int　_A67X_loadcs(const char *path);　← 現在ボード

ＶＢ　Function A67X_loadcs(ByVal board As Long,　← 任意ボード
ByVal path As String) As Long
Function SA67X_loadcs(ByVal path As String) As Long　← 現在ボード

引数　：board　ボード番号
path　ＣＯＦＦファイル名

説明　：ＣＯＦＦフォーマットのオブジェクトファイルをＤＳＰのメモリにダウンロード（A67X_loadc と同じ）し、リセットベクターを設定（A67X_boot と同じ）します。
ロード可能な領域は、ＤＳＰメモリ（０番地からメモリの実装されている範囲）です。
すでにロードされている領域への重複したロードの検出は行っていないので、ロードするアドレスには十分注意してください。

戻り値　：０　正常にロードが完了しました。
１　ＣＯＦＦファイルのロード異常です。 原因はA67X_loadc と同じです。
２　エントリーポイントの取得異常です。原因はA67X_entryc と同じです。
３　リセットベクタ領域にユーザープログラムがロードされています。
４　メモリへの書込異常（照合不一致）。

参考　：A67X_loadc、A67X_boot

使用例　：

ＶＣ
```
if(_A67X_loadcs("USER.OUT")){
        printf("ロードエラー.¥n");
}else{ printf("ロード正常終了.¥n");
}
```

ＶＢ
```
If SA67X_loadcs("USER.OUT") Then
     Print "ロードエラー."
Else:Print "ロード正常終了."
End If
```

19). A67X_nmi　　（ＤＳＰボードへの割り込みNMＩ発生）

記述　：ＶＣ　void　A67X_nmi(int board);　← 任意ボード
　　void　_A67X_nmi(void);　← 現在ボード

　　ＶＢ　Sub　A67X_nmi(ByVal board As Long)　← 任意ボード
　　Sub　SA67X_nmi()　← 現在ボード

引数　：board　　ボード番号

説明　：ホストからＤＳＰに対して、割り込みNMＩを発生します。
　　ホストからＤＳＰへの同期に利用できます。 ハードウェアによる割り込みです。

戻り値　：ありません。

参照　：A67X_nmi

使用例　：

ＶＣ

```
_A67X_nmi();
```

ＶＢ

```
Call SA67X_nmi
```

20). A67X_putmem　　(メモリへのダウンロード)

記述　：ＶＣ　int　A67X_putmem(int board,　← 任意ボード
　　unsigned long address,
　　void *data,
　　int verify = -1);
　　int　_A67X_putmem(unsigned long address,　← 現在ボード
　　void *data,
　　int verify = -1);

　　ＶＢ　Function A67X_putmem(ByVal board As Long,　← 任意ボード
　　ByVal address As Long,
　　ByRef data As Any,
　　Optional ByVal verify = -1) As Long
　　Function SA67X_putmem(ByVal address As Long,　← 現在ボード
　　ByRef data As Any,
　　Optional ByVal verify = -1) As Long

引数　：board　ボード番号
　　address　ダウンロード・アドレス
　　data　ダウンロード・データ
　　verify　ダウンロードデータの照合選択
　　０　：照合しない
　　－１（デフォルト）：ＩＳＡ接続の場合　照合する
　　ＵＳＢ接続の場合　照合しない
　　０、－１以外　：照合する

説明　：ホストのメモリからＤＳＰ上のメモリへ、データをダウンロードします。
　　データ長は１ワードです。

戻り値　：０　正常終了しました。
　　－１　メモリへの書込異常（照合不一致)。
　　その他のエラー。(USB のみ)
　　－２　通信タイムアウト。

参考　：A67X_getmem

使用例　：

ＶＣ
```
if(_A67X_putmem(0x1000,0x12345678)){
    Printf(“書込エラー.¥n”);
}
```

ＶＢ
```
If SA67X_putmem(&H1000,&H12345678) Then
    Print “書込エラー.”
End If
```

21). A67X_reset　　(ボードリセット)

記述　：ＶＣ　int　A67X_reset(int board);　← 任意ボード
　　int　_A67X_reset(void);　← 現在ボード

　　ＶＢ　Function A67X_reset(ByVal board As Long) As Long　← 任意ボード
　　Function SA67X_reset() As Long　← 現在ボード

引数　：board　　ボード番号

説明　：ＤＳＰをリセットします。
電源投入後、ＤＳＰの状態は不定となっていますので、この関数を実行してＤＳＰの初期化を行ってください。(通常は、A67X_libenter 関数にて行っています。)
この関数を実行すると、ＤＳＰは初期化されますのでボード初期化関数 (A67X_bdinit) にて初期化設定を行う必要があります。。

戻り値　：０　　正常終了。
０以外　異常終了。

使用例　：

ＶＣ
```
if(_A67X_reset()){
      printf("ボードリセット失敗.¥n");
}else{ printf("ボードリセット正常終了.¥n");
}
```

ＶＢ
```
If SA32X_reset Then
    Print "ボードリセット失敗."
Else:Print "ボードリセット正常終了."
End If
```

22). A67X_resetstus　　(ボードリセット状態の確認)

記述　：ＶＣ　int　A67X_resetstus(int board);　← 任意ボード
　　int　_A67X_resetstus(void);　← 現在ボード

　　ＶＢ　Function A67X_resetstus(ByVal board As Long) As Long　← 任意ボード
　　Function SA67X_resetstus() As Long　← 現在ボード

引数　：board　ボード番号

説明　：ボードのリセット状態を取得します。
ボードがリセット状態か、または、リセット解除状態かを返します。
A67X_libenter、A67X_reset によって、正しく制御された後にこの関数を実行した場合、
リセット解除状態が返されます。(電源投入時のみリセット状態となる。)
この関数の用途は、未知のボード状態を取得する必要がある場合にのみです。

戻り値　：０　　リセット状態。
０以外　リセット解除状態。

使用例　：

ＶＣ
```
if(_A67X_resetstus()){
      printf("ボードリセット解除状態.¥n");
}else{ printf("ボードリセット状態.¥n");
}
```

ＶＢ
```
If SA32X_resetstus Then
    Print "ボードリセット解除状態."
Else:Print "ボードリセット状態."
End If
```

23). A67X_resetirqhndl　　（ＤＳＰからの割り込みハンドラ解除）

記述　　：ＶＣ　　int　　A67X_resetirqhndl(void);

　　ＶＢ　　なし

引数　　：なし

説明　　：A67X_setirqhndl 関数で設定した、割り込み処理ハンドラを開放します。

戻り値　：０　　正常終了
　　０以外　開放を失敗しました。

使用例　：

ＶＣ
```
if(A67X_resetirqhndl()){
        printf("割り込みハンドラ解除エラー.¥n");
}else{ printf("割り込みハンドラ解除正常終了.¥n");
}
```

24). A67X_resume　　(デバイスドライバーの強制開放)

記述　　: ＶＣ　　int　　A67X_resume(void);

　　ＶＢ　　Function A67X_resume() As Long

引数　　: board　　ボード番号

説明　　: ユーザープログラムの異常終了などで、ロックされてしまったデバイスドライバーを
　　強制的に開放します。

戻り値　: ０　　正常終了。
　　０以外　異常終了。

使用例　:

ＶＣ
```
if(_A67X_resume()){
        printf("デバイスの強制開放失敗.¥n");
}
```

ＶＢ
```
If SA32X_resume Then
     Print "デバイスの強制開放失敗."
End If
```

25). A67X_run　　(ボード実行開始)

記述　：ＶＣ　int　A67X_run(int board);　← 任意ボード
　　int　_A67X_run(void);　← 現在ボード

　　ＶＢ　Function A67X_run(ByVal board As Long) As Long　← 任意ボード
　　Function SA67X_run() As Long　← 現在ボード

引数　：board　　ボード番号

説明　：ＤＳＰボードを実行状態にします。

戻り値　：０　　実行開始しました。
　　０以外　実行開始できませんでした。 原因は、次のいずれかです。
　　１）停止状態の解除ができなかった。

使用例　：

ＶＣ
```
if(_A67X_run()){
      printf("プログラム実行エラー.¥n");
}else{ printf("プログラム実行正常終了.¥n");
}
```

ＶＢ
```
If SA67X_run Then
     Print "プログラム実行エラー."
Else: Print "プログラム実行正常終了."
End If
```

26). A67X_savec　　（ＣＯＦＦファイルへのアップロード）

記述　：ＶＣ　int　A67X_savec(int board,　← 任意ボード
unsigned long address,
unsigned long size,
unsigned long entry,
const char *path);
int　_A67X_savec(unsigned long address,　← 現在ボード
unsigned long size,
unsigned long entry,
const char *path);

ＶＢ　Function A67X_savec(ByVal board As Long,　← 任意ボード
ByVal address As Long,
ByVal size As Long,
ByVal entry As Long,
ByVal path As String) As Long
Function SA67X_savec(ByVal address As Long,　← 現在ボード
ByVal size As Long,
ByVal entry As Long,
ByVal path As String) As Long

引数　：board　ボード番号
address　アップロード・アドレス
size　アップロード・サイズ（バイト単位）
entry　実行開始番地（エントリアドレス）
path　ファイル名

説明　：ＤＳＰのメモリの内容をＣＯＦＦファイルに格納します。
プログラムだけでなく、データ領域もセーブできます。
格納したＣＯＦＦファイルは、ダウンロード関数でロードできます。

戻り値　：０　正常終了しました。
０以外　異常終了しました。 原因は、次のいずれかです。
１）ファイル名が不正で、ファイルがオープンできない。
２）同名ファイルがあり、かつリードオンリーで、オープンできない。
３）ディスクの空き領域が、不足している。

参考　：A67X_loadc

使用例　：

ＶＣ
```
if(_A67X_savec(0x1000L,0x100L,0x1000L,"USER.OUT")){
        printf("ファイル作成エラー.¥n");
}else{ printf("ファイル作成正常終了.¥n");
}
```

ＶＢ
```
If SA67X_savec(&H1000,&H100,&H1000,"USER.OUT") Then
     Print "ファイル作成エラー."
Else:Print "ファイル作成正常終了."
End If
```

27). A67X_setboardsetup　　(デバイスドライバーの設定値登録)

記述　：ＶＣ　int　A67X_setboardsetup(int segment,
　　int io,
　　int irq);

　　ＶＢ　Function A67X_setboardsetup(ByVal segment As Long,
　　ByVal io As Long,
　　ByVal irq As Long)

引数　：segment　　セグメントアドレス
　　io　　Ｉ／Ｏアドレス
　　irq　　割り込み番号

説明　：各設定値をデバイスドライバーに登録します。
　　各設定値は、ＤＳＰボードの設定と一致させてください。

戻り値　：０　　正常終了しました。
　　０以外　異常終了しました。

参考　：A67X_getboardsetup

使用例　：

ＶＣ
```
If(A67X_setboardsetup(0xE000,0x300,10)){
        printf("設定エラー.¥n");
}else{ printf("設定正常終了.¥n");
}
```

ＶＢ
```
If A67X_setboardsetup(&HE000&,&H300,10) Then
     Print "設定エラー."
Else:Print "設定正常終了."
End If
```

28). A67X_setirqhndl　　（ＤＳＰからの割り込みハンドラ設定）

記述　：ＶＣ　int　A67X_setirqhndl(HWND hWnd,
　　UINT message,
　　WPARAM wParam,
　　LPARAM lParam);

　　ＶＢ　なし

引数　：hWnd　ウィンドウハンドル
　　message　メッセージ・コード
　　wParam　パラメータ１
　　lParam　パラメータ２

説明　：ＤＳＰからの割り込み処理ハンドラを設定します。
　　ＤＳＰから割り込みが発生した場合、hWnd で指定したウィンドウに message で指定したメッセージが送信されます。

戻り値　：０　正常終了しました。
　　０以外　設定を失敗しました。

使用例　：

ＶＣ
```
#define　　WM_DspInt　(WM_USER + 1)
if(A67X_setirqhndl(m_hWnd,WM_DspInt,0,0)){
　　printf("割り込みハンドラ設定エラー.¥n");
}else{ printf("割り込みハンドラ設定正常終了.¥n");
```

29). A67X_symbolc　　（ＣＯＦＦファイルからシンボル取り出し）

記述　：ＶＣ　int　A67X_symbolc(const char *path,
　　const char *symbol_name,
　　unsigned long *address);

　　ＶＢ　Function A67X_symbolc(ByVal path As String,
　　ByVal symbol_name As String,
　　ByRef address As Long) As Long

引数　：path　　ＣＯＦＦファイル名
　　symbol_name　　シンボル名
　　address　　アドレス値を格納する変数

説明　：ＣＯＦＦファイルから、変数などのアドレス情報を取得します。
　　ＣＯＦＦファイルを取り込み、変数名を指標にして、アドレス情報を取得します。　プログラム開発時などで、アドレスが流動的な場合に有用です。
　　取り出し可能なシンボルは、グローバル宣言のある変数・関数に限ります。
　　Ｃ言語の場合は大域変数で、クラスが static でないもの、および、static でない関数です。　Ｃ言語のシンボルは、ソースで定義した名称の先頭に_が付いたシンボルになります。

戻り値　：０　　正常に検出しました。
　　０以外　異常終了しました。 原因は次のいずれかです。
　　１）ファイルが見つからない。
　　２）ファイルが異常である。
　　３）ファイルがＣＯＦＦファイルではない。

参考　：A67X_symbolm, A67X_entryc, A67X_entrym

使用例　：

ＶＣ
```
unsigned long     adrs;
if(A67X_symbolc("USER.OUT","_value",&adrs)){
      printf("シンボル取得エラー.¥n");
}else{ printf("symbol address %08lX.¥n",adrs);
}
```

ＶＢ
```
Dim adrs As Long
If A67X_symbolc("USER.OUT","_value",adrs) Then
    Print "シンボル取得エラー."
Else:Print "Symbol address ";Hex$(adrs)
End If
```

30). A67X_symbolm　　(マップファイルからシンボル取り出し)

記述　：ＶＣ　int　A67X_symbolm(const char *path,
　　const char *symbol_name,
　　unsigned long *address);

　　ＶＢ　Function A67X_symbolm(ByVal path As String,
　　ByVal symbol_name As String,
　　ByRef address As Long) As Long

引数　：path　　ＭＡＰファイル名
　　symbol_name　　シンボル名
　　address　　アドレス値を格納する変数

説明　：ＭＡＰファイルから、変数などのアドレス情報を取得します。
　　ＤＳＰプログラムの開発時に、リンカーが出力したＭＡＰファイルを取り込み、変数名を指標にしてアドレス情報を取得します。　プログラム開発時などで、アドレスが流動的な場合に有用です。
　　取り出しの可能なシンボルは、グローバル宣言のある変数・関数に限ります。
　　Ｃ言語の場合は、大域変数でクラスが static でないもの、および、static でない関数です。　Ｃ言語のシンボルは、ソースで定義した名称の先頭に_が付いたシンボルになります。

戻り値　：０　　正常に検出しました。
　　０以外　異常終了しました。 原因は次のいずれかです。
　　１）ファイルが見つからない。
　　２）ファイルが異常である。
　　３）ファイルがマップファイルではない。

参考　：A67X_symbolc、A67X_entryc、A67X_entrym

使用例　：

ＶＣ
```
unsigned long     adrs;
if(A67X_symbolm("USER.MAP","_value",&adrs)){
      printf("シンボル取得エラー.¥n");
}else{ printf("Symbol address %08lX.¥n",adrs);
}
```

ＶＢ
```
Dim adrs As Long
If A67X_symbolm("USER.MAP","_value",adrs) Then
     Print "シンボル取得エラー."
Else:Print "Symbol address ";Hex$(adrs)
End If
```

31). A67X_valid　　(ボード実装確認)

記述　　：ＶＣ　　int　　A67X_valid(int board);

　　ＶＢ　　Function A67X_valid(ByVal board As Long) As Long

引数　　：board　　ボード番号

説明　　：指定されたボード番号のボードが実装されているかどうかを取得します。
ボードが実装されているかどうかは、ライブラリの初期化関数（A67X_libenter）が確認をし、変数に保存されています。　この関数は、その変数の値を返します。

戻り値　：０　　ボードは実装されていません。
０以外　ボードが実装されています。

使用例　：実装されているボードを、すべてリセット、初期化設定します。(エラー処理は省略)

ＶＣ

```
#define  BD_MAX   16
int      bn;
for(bn = 0;bn < BD_MAX;++bn){
    if(A67X_valid(bn)){
        A67X_bdsel(bn);
        _A67X_reset();
        _A67X_bdinit();
    }
}
```

ＶＢ

```
Const BD_MAX As Long = 16
Dim bn As Long
For bn = 0 To BD_MAX - 1
    If A67X_valid(bn) Then
        Call A67X_bdsel(bn)
        Call _A67X_reset()
        Call _A67X_bdinit()
    End If
Next bn
```

(5). ライブラリー使用上の注意

ライブラリーをより効率的かつ的確に使用していただくためには、注意すべき点がいくつかあります。下記に記しますのでプログラミングの際の、ご参考にしてください。

1). このライブラリーには、ライブラリーの初期化関数(A67X_libenter)と開放関数(A67X_libexit)があります。 初期化関数は、ほかのどの関数にも先立って実行する必要があります。 これを行わないとライブラリが正しく動作しないほか、思わぬ動作をする場合がありますので注意してください。 また、開放関数はプログラムを終了する直前で呼び出す必要があります。 これを行わないと、次回ライブラリーを使用するときに、デバイスドライバーが開放されていないため初期化異常が発生します。 Ｃ言語などでは、atexit 関数などプログラムが終了する直前、ＢＡＳＩＣでは、メインフォームの Unload イベントを使用してこの関数を呼び出すようにしてください。

2). ＤＳＰボードは、ホストから見た場合、マルチボード構成になります。 従って、ライブラリーもそれに対応しています。ホストからＤＰＳボードをアクセスする場合の、基本的な手順は、

①アクセスするボードを選択する

②ボードをアクセスする

といった手順になります。

ところが、ＤＳＰボードを１台しか使用しない場合には、対象ボードの選択は、初めに１回のみ行えばよく、毎回行う必要はありません。

そこで、ライブラリーの使いやすさとオーバーヘッドの低減を考慮し、ＤＳＰボードをアクセスする関数には、

Ａタイプ：対象ボードを選択してからアクセスする

Ｂタイプ：対象ボードを選択せず（現在選択されているボードに対して）アクセスする

の２種類のタイプがあります。

また、対象ボードを選択するだけの関数も用意されています。

接続するＤＳＰボードの台数、アクセスするボードを切り替えるタイミングなど、ユーザープログラムの構造に応じて、適切なタイプを選択して使用してください。

○ ＤＳＰボードが１台の場合―――

初期化時に対象ボード番号を選択し、以後上記Ｂタイプの関数を使用する。

○ ＤＳＰボードが複数の場合で、全ボードを平均的にアクセスする場合―――

初期化後、上記Ａタイプの関数を使用する。

○ ＤＳＰボードが複数で、特定のボードにアクセスが集中する場合―――

ボードごとの手続きを関数化し、関数の始めで対象ボードを選択し、関数内ではＢタイプの関数を使用する。

なお、ＡタイプとＢタイプの関数名称はほぼ同名で、Ａタイプの関数の先頭に

Ｃ言語では“_”

ＢＡＳＩＣでは、“S”

を付加した名称がＢタイプの関数名称となります。

(6). 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。

「5」.　デバイスドライバー

このデバイスドライバー（A67X32.VXD）は、Ｗｉｎｄｏｗｓ９５/９８/ME/NT４．０/２０００/XP上のアプリケーションから、ＤＳＰのハードウェアをアクセスするための仮想デバイスドライバーです。通常このデバイスドライバーを、ユーザーが直接操作することはなく、付属ライブラリー（ダイナミックリンク ライブラリーのA67X32.DLL）を経由して、このデバイスドライバーを使用することになります。このデバイスドライバーは、Ｗｉｎｄｏｗｓのシステム・フォルダー内に格納されている必要があります。セットアッププログラムを使用してインストールを行った場合は、自動的にインストールされます。

・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

ADSP674-00

ｿﾌﾄｳｪｱ･ﾕｰｻﾞｰｽﾞ･ﾏﾆｭｱﾙ

2000. 6　　第１版発行

2007. 9　　第６版発行