本書は MPLAB 開発システムのインストール、使用方法を段階的に示すチュートリアルです。フロッピーディスク、CD-ROM またはハードディスクのいずれかに MPLAB のインストールファイルを入れてください。インストールファイルは www.microchip.com からダウンロードできます。

このチュートリアルではテキストに 2 種類のフォントサイズを使用しています。この文章は通常のフォントサイズです。さらに「注意」として、以下のような小さいフォントで補足的な情報を示します。

> 注意：この文章は注意です。注意はこのチュートリアルに必要不可欠な情報ではありませんが、詳しい説明を加えるものです。

MPLAB チュートリアル

MPLAB は、Microchip の PICmicro コントローラ上で動作するコードの書き込みや、テストを可能にする統合開発環境です。MPLAB はテキストエディタ、プロジェクト管理機能、組み込みシミュレータなどのほか、アプリケーションのデバッグおよびテストを行う各種ツールを備えています。また、Microchip の言語系製品、デバイスプログラマやエミュレータシステム、サードパーティツールなどの単独ユーザーインターフェースとなります。

このチュートリアルでは、MPLAB のユーザーインターフェースを簡単に紹介しています。1～2 時間で 10 ステップからなるチュートリアルをすべて終了できるでしょう。

MPLAB は Windows 3.11 用に設計され、Windows 95 および Windows 98 で動作します。Windows やその操作については、理解されていることとして詳しく説明していません。

このチュートリアルで説明するとおりに MPLAB を操作すると、以下の事項ができるようになります。

- MPLAB デスクトップを理解する
- アセンブリソースコードファイルを新規作成し、16F84 の新規プロジェクトを作成する
- 単純なエラーを発見し、修正する
- 組み込みシミュレータを稼働させる
- ブレイクポイントを設定する
- ウォッチウィンドウを作成する
- 各種デバッグウィンドウを理解する

本書は MPLAB のすべてにわたって詳細に説明をするものではありません。MPLAB をすぐに使用できるように初歩的な説明を行うものです。基本事項を理解してしまえば、Microchip のホームページ（www.microchip.com.）から、より高度なトピックのアプリケーションやチュートリアルをダウンロードして使用できるようになります。

MPLAB チュートリアル目次

ステップ 1: インストール

インストール用のソフトウェアファイルをダウンロードし、MPxxxxx.EXE **ファイルを実行します。もう** 1 **台のコンピュータにインストールする場合は、これら別個のファイルをフロッピーディスクへコピーできます。バージョンによっては、ファイル名が多少異なります。例えば、**MPLAB **の** Version 4.00 **は以下の様なファイル構成となります。**

MP40000.EXE
MP40000.W02
MP40000.W03
MP40000.W04
MP40000.W05
MP40000.W06

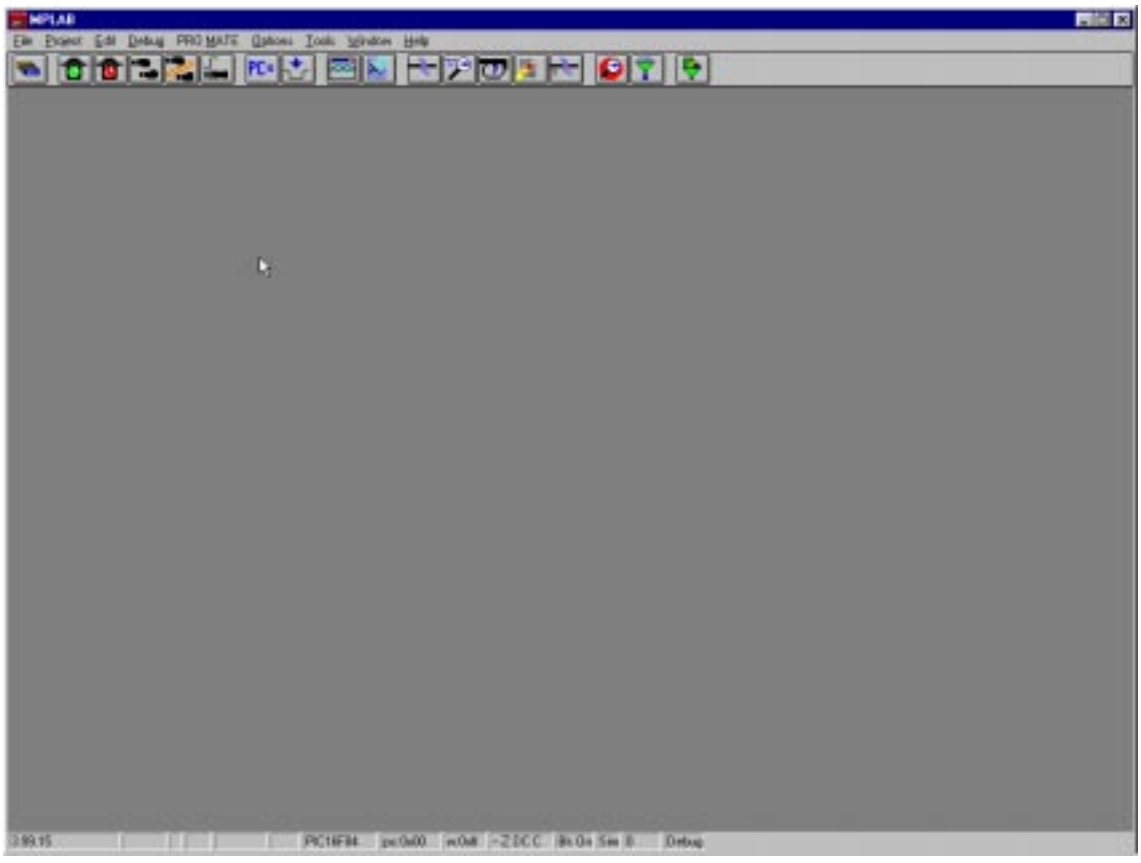
.EXE **ファイルを実行すると、システム上へ** MPLAB **のインストールが開始します。システムにインストールする** MPLAB **のファイルを選択することができます。デバイスプログラマやエミュレータをお持ちでない場合は、以下のソフトウェアツールのみをインストールするのみで構いません。**

MPLAB IDE files
MPASM/MPLINK/MPLIB files
MPLAB-SIM Simulator Support Files
Help Files

注意: PICSTART **Plus プログラマをお持ちの場合は、**「PICSTART Plus Support Files」**も選択してください。**PICMASTER, MPLAB-ICE, ICEPIC, SIMICE **などのツールがある場合は、ダイアログの関連項目のチェックボックスをクリックし、そのソフトウェアドライバをインストールしてください。**MPLAB **をあとで再インストールして、コンポーネントを追加することもできます。**

次のメニューでは、インストールする Microchip の言語コンポーネントを選択できます。通常は、すべて選択します（デフォルト）。

インストールが終了したら、MPLAB.EXE を実行するか MPLAB のアイコンをクリックし、システムを起動します。以下のような MPLAB のデスクトップが表示されます。

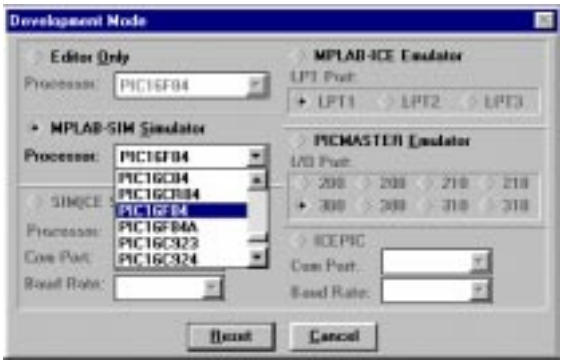


ステップ 2: 開発モードの設定

基本的な MPLAB のデスクトップは、（前ページの図のように）ほとんどの Windows のアプリケーションと同じです。一番上の行にはメニュー、ツールバー、一番下にはステータスバーがあります。ステータスバーには、現在のシステムの設定が出ています。これらの機能については、あとでより詳細に説明します。ここでは開発モードの設定方法を見てみましょう。

> 注意：「Development Mode」ではコードを実行するツールを設定します。このチュートリアルでは、ソフトウェアシミュレータ MPLAB-SIM を使用します。エミュレータがすでにある場合は、あとでエミュレータオペレーションの 1 つに切り替えることができます。操作はほとんど同じです。「Editor Only」モードではコードを実行することができません。「Editor Only」モードはおもに、シミュレータをインストールしていないか、エミュレータがない場合、または PICmicro のプログラミング用コードを作成するのみの場合に使用します。

「Options>Development Mode」のメニュー項目を選択すると、下の図のようなダイアログボックスが開きます。



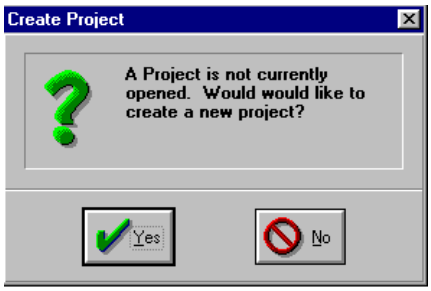
MPLAB は常にバージョンアップが行われている製品なので、画面に表示されるダイアログボックスは、ここに示したものと若干違っているかもしれません。 MPLAB-SIM シミュレータのラジオボタンをクリックし、プルダウンメニューにある、シミュレータがサポートしているプロセッサの中から 16F84 を選択します。16F84 をクリックしたあと、「Reset」ボタンをクリックします。シミュレータは初期化され、「16F84」と「Sim」が MPLAB のデスクトップの下にあるステータスバーに表示されます。これで、16F84 のシミュレータモードに設定されました。

ステップ 3:　　簡単な新規プロジェクトの作成

このシミュレータは、PICmicro へプログラム書き込みするための「hex file」と呼ばれるファイルを使用して動作します。シミュレータを動作させるには、まずソースコードファイルを作成し、そのソースコードを正しくアセンブルします。

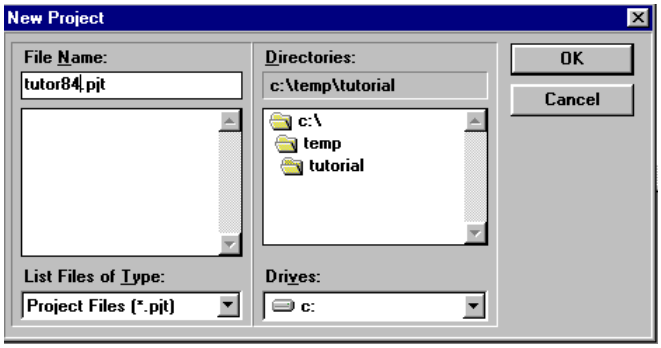
> **注意: アセンブラは、hex file を生成します(他のものも生成しますが)。このファイルにはファイル拡張子.hex が付いています。このチュートリアルでは、このファイル名は tutor84.hex とします。あとでこのファイルは、アセンブラーまたは MPLAB プロジェクトを使用しないで、直接デバイスプログラマへロードすることができます。またこのファイルは、ほとんどのサードパーティのプログラマにもロードできます。**

メニューの「File>New」を選択すると、以下のようなダイアログが表示されます。



「Yes」をクリックすると、Windows の通常のブラウズダイアログが出ます。プロジェクトを作成するディレクトリを決めて、あとで必要になる場合に備えて、その場所を覚えておきます。このチュートリアルでは `c:¥temp¥tutorial` のディレクトリを使用し、プロジェクトファイル名を `tutor84.pjt` とします。

「PJT」は、MPLAB のプロジェクトファイルに通常使用される拡張子です。プロジェクトファイル名(このチュートリアルでは「tutor84」)は、MPLAB が使用したり、作成したりするファイルのほとんどでデフォルト名として使用されます。



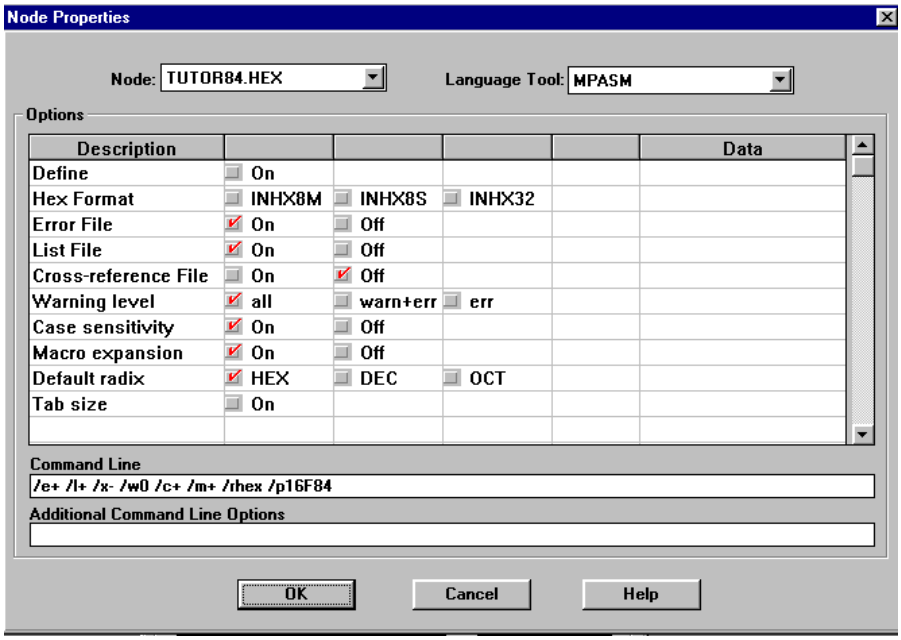
マウスを使用して「OK」をクリックすると、MPLAB プロジェクトダイアログが開きます。このダイアログは複雑に見えますが、実は非常に単純です。

> 注意: MPLAB で使用するシミュレータ、プログラマおよびエミュレータシステムは、ソースコードをアセンブルしたり、コンパイルまたはリンクして作成された hex file を使用します。いくつかの異なるツールによっても hex file は作成でき、それらのツールはそれぞれのプロジェクトの一部分となります。プロジェクトを作成することで、どのようにアプリケーションが構築されているかや、どのソフトウエアツールが.hex file の作成に使用されているかをを柔軟に設定できます。このチュートリアルでは、これらの事項については詳しく説明しませんが、必要に応じて、「Node Properties」を使用して設定を行ってください。より複雑なプロジェクトについては、MPLAB v3.40 Project Tutorial を参照してください。

「Edit Project」のダイアログは、以下のように表示されます。

Target Filename に指定したファイル名があるか確認してください。前もって設定した開発モードはすでに認識されています。デフォルトで Microchip の言語ツールを使用する設定となっています。「Project Files」のウィンドウには `tutor84.[hex]`と表示されます。このファイル名をハイライトすると、「Node Properties」のボタンが有効になります。最初に、MPLAB に hex file の作成方法を指定しなければなりません。これを行うには、「Node Properties」のボタンをクリックします。「Node Properties」のダイアログが開きます。

このダイアログには、言語ツールのデフォルト設定がすべて表示されます。この場合は、ダイアログの右上にあるように MPASM となっています。最も簡単なプロジェクトは、1 つのアセンブリソースファイルで hex file を作成するプロジェクトです。下の図は、「Node Properties」のダイアログが表示された時のデフォルト設定です。



注意：このダイアログには、多くの行と列があります。列はそれぞれ「スイッチ」と対応しており、ツールが起動された時にコマンドラインに設定される項目です。実際、これらのスイッチの設定は、下の方にある「Command Line」のウィンドウに表示されます。これは、MPLAB から MPASM を起動する時に発行される実際のコマンドラインです。ここでは、デフォルト設定を使用しますが、アプリケーションの構築に慣れてくると、変更したい設定がでてくるかもしれません。

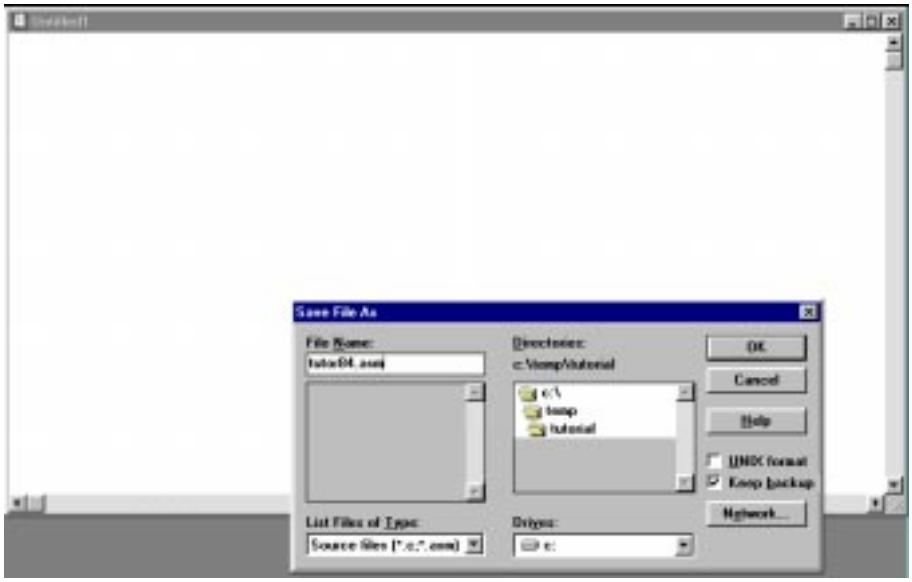
「OK」をクリックすると、デフォルト設定され、「Edit Project」のダイアログへ戻ります。「Add Node」のボタンが有効になります。

「Add Node」のボタンをクリックすると、ウィンドウの通常のブラウズダイアログが開きます。プロジェクトのディレクトリと同じディレクトリが表示されます。ファイル名 `tutor84.asm` を入力し、「OK」をクリックします。「Edit Project」のダイアログに戻り、「`tutor84.asm`」が使用されているノードとして hex file の下にインデントされます。

「OK」をクリックすると、MPLAB のデスクトップへ戻り、オープンされているが、まだファイル名が付いていないソースコードファイルが表示されます。

ステップ 4:　　簡単な新規ソースファイルの作成

作成された空のファイルウィンドウのブランクスペースを、一回クリックします。題名は「Untitled**」となっていますが、これによって、この空のウィンドウはフォーカスされます。「**File>Save As...**」のメニューオプションを使用し、この空のファイルを** `tutor84.asm` **という名前で保存します。通常のブラウズダイアログが開くと、プロジェクトの現在使用中のディレクトリになっています。ファイル名を入力して「**OK**」をクリックします。**



これで MPLAB **デスクトップと空のファイルウィンドウが表示されます。ファイルウィンドウ名は更新されています。**

> **注意：この種のプロジェクトでは、ソースファイル名とプロジェクト名は同じでなければなりません（このチュートリアルでは「**tutor84**」）。リンカを使用した複数ファイルのプロジェクトは入力ファイル名と違うファイル名を出力ファイル名にすることができます（リンカを使用するマルチファイルプロジェクトには別のチュートリアルがあります）。このチュートリアルでは、１つのソースファイルのプロジェクトなので、**MPASM **は常に出力** hex file **をソースファイルと同じファイル名で作成するので、これを変更できません。ソースファイル名を変更する場合は、必ずプロジェクト名も変更してください。**

ステップ 5: ソースコードの入力

マウスを使用してカーソルの位置を tutor84.asm **の空のファイルウィンドウの中に置きます。以下のテキストを示したとおりに入力します。セミコロンの後に続くコメント文は入力する必要はありません。**

```
   list   p=16f84
   include <p16F84.inc>

c1  equ    h'0c'      ; Set temp variable counter c1 at address 0x0c

   org    h'00'  ; Set program memory base at reset vector 0x00
reset
   goto   start      ; Go to start of the main program

   org    h'04'      ; Set program memory base to beginning of user code
start
   movlw  h'09'  ; Initialize counter to arbitrary value greater than zero
   movwf  c1    ; Store value in temp variable a defined above
loop
   incfsz  c1,F  ; Increment counter, place results in file register
   goto   loop  ; Loop until counter overflows

   goto   bug ; When counter overflows, got to start to re-initialize
   end
```

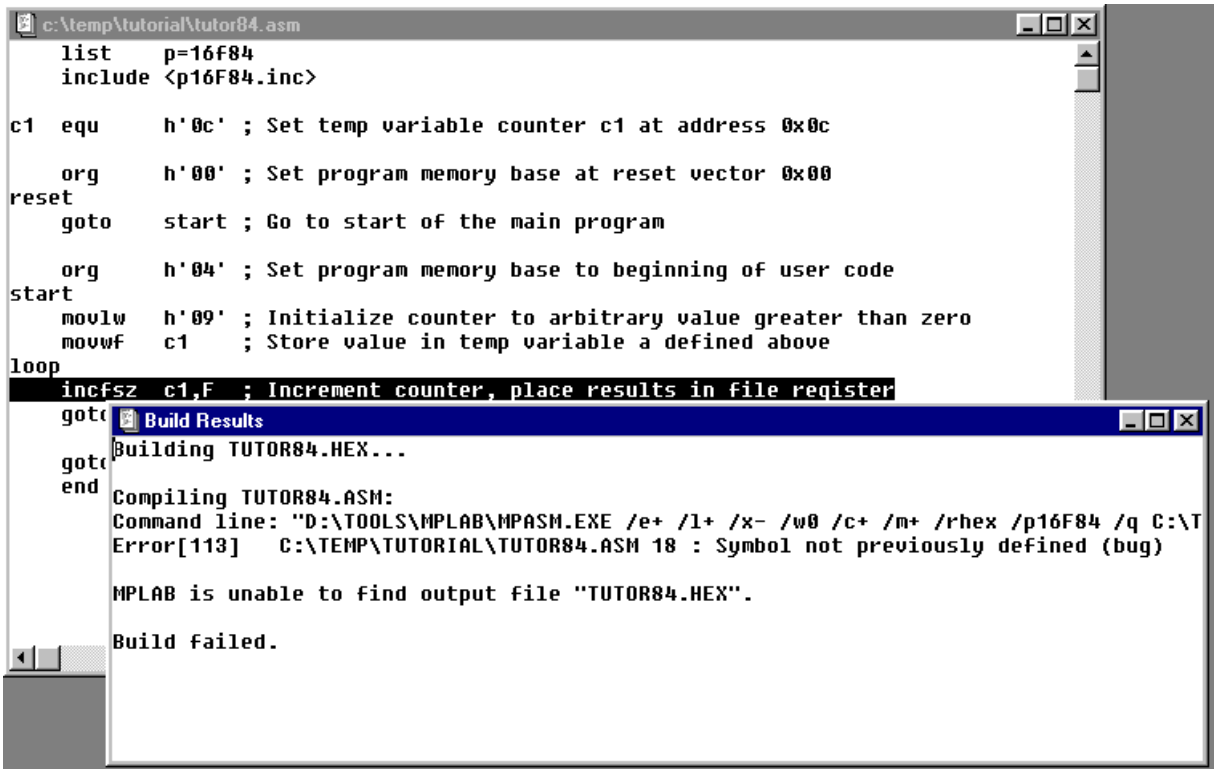
このコードは大変簡単なプログラムで、カウンタをインクリメントし、カウンターがゼロになったらあらかじめ決めた値にリセットします。

> **注意**: **すべてのラベルは最初の列から記入します。最後の行には「**end**」疑似命令を付けます。疑似命令についての詳細は、**MPASM **と** MPLINK **および** MPLIB **ユーザーガイドを参照してください。** PICmicro **のデータシートには使用例と命令についての説明が記載されています。**

「File>Save**」 メニューの機能を使用してファイルを保存します。**

ステップ 6:　　ソースファイルのアセンブル

ファイルをアセンブルする方法はいくつかあります。ここに示した方法はその一つです。「Project>Build All**」 メニュー項目を使用します。これは、前述のプロジェクトで保存したデフォルトを使用して、バックグラウンドで** MPASM **アセンブラを実行します。アセンブル処理が完了すると、「**Build Results**」ウィンドウが表示されます。**



前出の通りにコードを入力した場合、少なくとも 1 **つのエラーを意図的に入力しました。プログラムの最後の「**goto**」　は「**bug**」という名の存在しないラベルを参照しています。このラベルはあらかじめ定義されていないので、アセンブラはエラーを報告します。他にもエラーがある場合があります。**

マウスを使用してエラーメッセージをダブルクリックします。 こうするとカーソルがソースコードの中のエラーのある行に移動します。「bug**」を「**start**」に変更します。「**Build Results**」ウィンドウを使用してエラーを検索して、ソースコードの中の他のバグを修正することができます。「**Project>Build All**」メニュー機能を実行して再アセンブルします。エラーがなくなるまでこの手順を何度か繰り返します。**

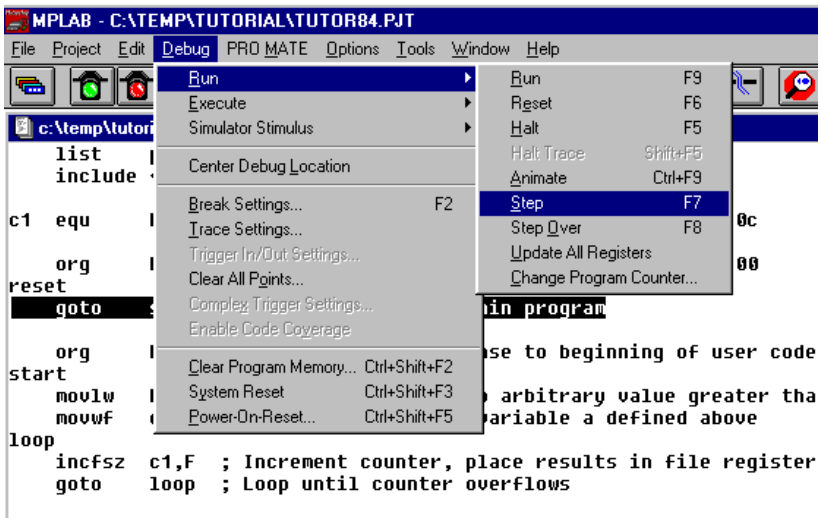
> **注意:　プロジェクトを再構築する時に、すべてのソースファイルがディスクに保存されます。**

ソースコード中のすべての問題を修正すると、「Build Results**」ウィンドウに「**Build completed successfully.**」と表示されます。これで、シミュレーターを使用して実行できるプロジェクトができました。**

ステップ 7:　　プログラムの実行

「Debug>Run>Reset」を使用してシステムを初期化します。プログラムカウンターはゼロにリセットされます。これは 16F84 のリセットベクタです。このアドレスのソースコードの行はダークバーでハイライトされます。MPLAB の一番下のステータスバーで、PC が 0x00 にセットされます。

「Debug>Run>Step」のメニュー項目を使用すると、次の命令の位置へプログラムカウンタが進みます。このダークバーはソースコードに続き、ステータスバーに表示されたプログラムカウンタは「4」へと進みます。



「Debug>Run>Step」のメニュー項目を実行すると「F7」というメニュー項目が右側に表示されます。これはキーボード上の 「ファンクションキー 7」 を表します。多くの MPLAB 機能は「ホット・キー」に割り当てられています。これらのキーはメニュー項目そのものを実行するのと同じことができます。F7 を数回押し、プログラムを通してプログラムカウンタとダークバーがプログラムにより進むのを確認してください。

「Debug>Run>Run」メニュー項目を実行するか F9 を押して、現在のプログラムカウンターの位置からプログラムを実行開始します。ステータスバーは色を変えて、プログラムが命令を実行していることを示します。ステータスバーの他のフィールドはプログラムが停止するまで更新されません。

「Debug>Run>Halt」メニュー項目を実行するか F5 を押して、プログラムを停止します。ステータスバーが元の色へ戻り、現在のプログラムカウンターと他のステータス情報が更新されます。

> 注意:　画面上部のツールバーを使用して機能を実行する方法もあります。ツールバー上の項目にカーソルを置くと、下にあるステータスバーの中に機能の名前が表示されます。ツールバーの左端のボタンは標準の「変更ツールバー」ボタンです。利用できるいくつかのツールバーをスクロールすることができます。これらは、チュートリアルの最後の「ヒントとコツ」で説明しますが、カスタマイズできます。デバックツールバーでは、緑のライトが F9 (Run) に相当し、赤のライトが F5 (Halt) となります。

ステップ 8: その他のデバック用ウィンドウのオープン

MPLAB を使用してプログラムの実行状況を確認する方法はたくさんあります。例えば、このプログラムはテンポラリーのカウンタをインクリメントしますが、それが確実に実行されているのをどのように確認できるでしょうか。一つは、ファイルレジスターウィンドウを開き、詳しく調べる方法があります。「Window>File Registers」メニュー項目を実行してください。すべてのファイルレジスタまたは 16F84 の RAM のウィンドウが表示されます。

F7 (シングルステップを実行) を数回押して、ファイルレジスタウィンドウの値が更新されているのを確認します。カウンターの変数をアドレスの 0x0C に置きます。テンポラリーのカウンタがインクリメントしたら、ファイルレジスタウィンドウに反映されます。簡単に確認できるように値が変化した時に色が変わるようになっています。しかし、非常に複雑なプログラムの場合、多くの値が変わることがあるので、1 または 2 つの変数に注目することが困難になります。この問題は、ウォッチウィンドウを使用することで解決します。

ステップ 9: ウォッチウィンドウの作成

「Window>New Watch Window」メニュー項目を実行します。「Add Watch Symbol」ダイアログが表示されます。

シンボル名のボックスに 「c1」 と入力すると、希望の記号にスクロールできるリストが表れます。希望の記号をハイライトして「Add」ボタンをクリックし、次に「Close」ボタンをクリックします。MPLAB **デスクトップ上のウォッチウィンドウにテンポラリーのカウンタ値「c1」 の現在の値が表示されます。**

F7 **を押してプログラムを数回シングルステップすると、カウンタをインクリメントするごとにウォッチウィンドウのディスプレイが更新されます。ファイルレジスタウィンドウをオープンのままにした場合、同様に更新されます。**

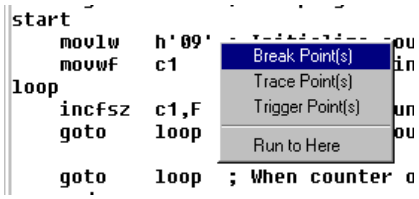
オプション ウォッチウィンドウの保存
ウォッチウィンドウのシステムボタンの下にある 「Save Watch」 項目を実行すると、ウォッチウィンドウとそのセッティングの保存ができます。システムボタンはウォッチウィンドウの左上の角にあります。このボタンをクリックすると、メニューが下へカスケード表示されます。「Save Watch」を選択すると、標準ブラウズダイアログを使用してプロジェクトの現在作業中のディレクトリが表示されます。任意の名前を入力して「OK」を押してください。

ウォッチウィンドウに名前を付けない場合、MPLAB **が名前を付けます。次にプロジェクトを開いた時に、ウォッチウィンドウも修復されるように、ウィンドウのオープンまたはクローズ状態とスクリーン上の位置がプロジェクトとともに保存されます。**

> **注意**: **ウォッチウィンドウは、作成後にもエディットできます。システムボタンを使用して 「Add Watch」を選択し、ダイアログを立ち上げて項目を追加します。「Ins」キーでも同じことができます。項目をハイライトし、**Delete **キーを押すとウィンドウからその項目が消去されます。システムメニューから「Edit Watch」を選択し、項目の表示方法（16 進、バイナリ、8 ビットではなく 16 ビットの変数としてなど）を変更することもできます。**

ステップ 10: ブレークポイントの設定

F5 (「Debug>Run>Halt」)を押してシミュレータのプロセッサが停止しているのを確認します。「movlw h'09'」と書かれた「Start」ラベルの直後の行にあるソースコードウィンドウをクリックします。 マウスの右ボタンを押すとメニューが表示されます。



「Break Point(s)」メニュー項目をクリックすると、メニューが消えてカーソルが置かれていた行の色が変わり、ブレイクポイントがその位置に設定されていたことを示します。

F6 を押して「Debug>Run>Reset」メニュー項目を実行し、システムをリセットします。次に、F9 を押してシステムを実行開始します。プログラムが実行開始され、ブレークポイントの直後で命令が停止します。「c1」は、ウォッチウィンドウやファイルレジスタウィンドウに表示されますが、MPLAB 起動後のリセットステータスのゼロとなります。１ステップ進むとロードを実行し、c1 は 0x09 になります。F9 を数回押すと実行中にステータスバーの色が変わり、プロセッサが停止すると再び色が変わります。

まとめ

このチュートリアルでは以下の方法を説明しました。

- 新規プロジェクトの設定
- ソースファイルの作成とプロジェクトへの入力
- コードのアセンブル
- シミュレータを使用したコードの実行
- ブレイクポイントの設定とコードのシングルステップ
- コードの変数の確認

ここで紹介したトピックを慣れましたら、次章でさらに MPLAB の他の機能についての説明をご覧ください。

ヒントとコツ

ブレークポイント –「Windows>Program Memory」ウィンドウ、ソースファイルウィンドウ(今回は tutor84.asm)、または「Windows>Absolute Listing」ウィンドウでブレイクポイントを設定します。

ソースファイル - 「Window>Project Window」を使用してソースファイルのリストを表示します。ファイル名をダブルクリックし、エディタにファイルを表示します。

MPASM エラー - MPASM にエラーが発生した場合、エラーウィンドウの中のエラーをダブルクリックして、ソースコードのエラーへ進みます。複数のエラーが発生した時は常に最初のエラーを選択してください。（１つのエラーが次のエラーの原因となることが多いので、最初のエラーを修正するとすべてが修正されることがあります。）

コンフィグレーションビットおよびプロセッサモードー ソースファイルの中のコンフィグレーションビットはシミュレータ（またはエミュレータ）のプロセッサーモードを設定しません。「Options>Processor Setup>Hardware」を使用して設定します。MPASM または MPLAB-C17 ソースファイルにこれらのビットを設定できますが、MPLAB は自動的にモードを変更しません。例えば、デバイスにプログラムを書き込む時、Watch Dog Timer Enable コンフィグレーションビットは設定可能なので、Watch Dog Timer がオンになるように設定できます。MPLAB では、シミュレータまたはエミュレータで WDT をイネーブルするには、更に「Options>Processor Setup>Hardware」ダイアログで設定する必要があります。これによりソースコードを変えずに WDT をオンまたはオフにしてデバックすることができます。

オプション - 次の事柄を行うには「Options>Environment Setup」へ進みます。

- MPLAB 機能と特別な ASCII 文字を European キーにマッピングする。
- スクリーンフォント、またはフォントサイズを変更する。
- ツールバーを画面の横、または下に配置する。
- ツールバーを変更する。
- 表示するラベルの文字数を変更する。

マップファイル - 「Project>Edit Project」ダイアログへ進み、MPASM のノードプロパティを変更して `tutor84.map.` という MAP ファイルを作成します。プロジェクトを構築したら `tutor84.map` を見て、ビルド情報を確認してください。

グレイアウトされたメニュー – メニューの表示がグレイアウトされている場合は、「Editor Only」モードになっていないことを確認してください。すべてが正しく設定されているなら、MPLAB を再起動してください。

チュートリアルで説明していない MPLAB 機能

MPLAB にはこの簡単なチュートリアルで説明している以外にも多くの機能があります。すべての機能について踏み込んだ説明をして、極端に長く複雑なチュートリアルにしてしまうより、ここでは MPLAB についての基本的な知識を身に付けましょう。MPLAB の基本的なもの以外のツールについてもいくつか説明しています。

他のデータウィンドウおよびダイアログ

チュートリアルではプログラムメモリウィンドウとウォッチウィンドウについて説明しました。その他のウィンドウについては説明しませんでした。「Window>Stack」および「Window>Special Function Registers」を選択すれば、MPLAB は他のメモリ領域も表示できます。。

ブレークポイント設定のための「Debug>Break Settings…」ダイアログなど、他のダイアログもあります。

これらの機能に関してはオンラインヘルプおよび MPLAB ユーザーズガイドで説明しています。

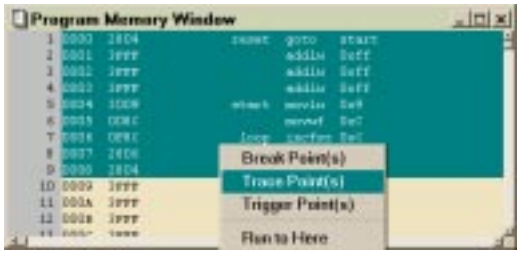
トレース

トレースウィンドウではプログラム実行の「スナップショット」を撮ります。トレースバッファを備えたエミュレータでは、作成したプログラムが実際の速度でどのように実行しているかを表示できます。

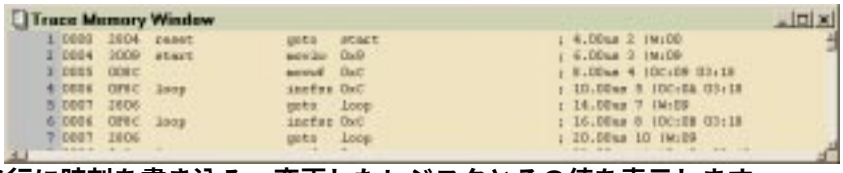
> 注意：モーターの制御システムなど、停止させることができないアプリケーションもあります。バグの中にはアプリケーションが稼働している間だけ現われるものもあります。(シングルステップでコードを実行してもバグが発生しない場合) トレースバッファなら別のツールでそのようなアプリケーションをテストできます。ハードウェアのトレースバッファに収集した情報に関しては、エミュレータのユーザーズガイドで確認してください。

シミュレータでは、トレースバッファは長いプログラム実行記録の収集をするのに役立ちます。記録の収集により、あとで、戻って解析をすることができます。その場合、シミュレータはエミュレータのトレースとは多少異なる情報を提示します。

シミュレータのトレースバッファを使用するには、まずトレースするコードを選択する必要があります。トレースする命令を選ぶプログラム・メモリ・ウィンドウ全体をクリック･アンド･ドラッグする場合は、マウスの右ボタンをクリックします。するとダイアログが表示され、「Trace Point(s)」を選択することができます。



プログラムをリセットし、実行します。数秒実行した後、停止させます。「Window>Trace」を選択し、収集されたトレースを確認します。



シミュレータは各行に時刻を書き込み、変更したレジスタとその値を表示します。.

スティミュラス

スティミュラスはシミュレータへの入力信号を生成します。ピンを High または Low に設定したり、値を直接レジスタに入力することができます。スティミュラスには以下の 4 モードがあります。

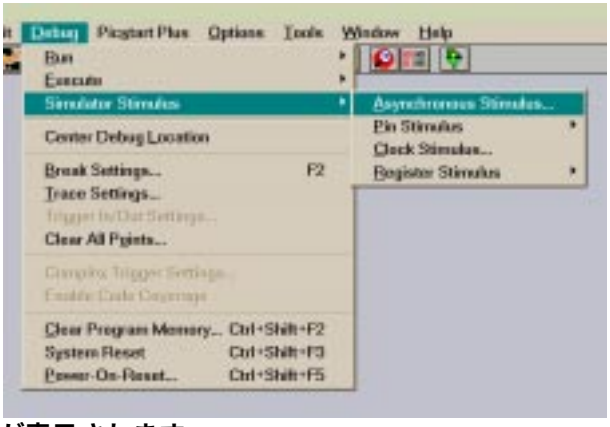
- 非同期スティミュラス – 入力ピンの信号をダイアログのボタンを使用して制御します。
- スティミュラス・ピン・ファイル – テキストファイルの内容をピンの入力信号として入力します。
- スティミュラス・レジスタ・ファイル – テキストファイルの内容を使用して、レジスタに直接 8 ビットの値を入力します。
- クロックスティミュラス – プログラム可能なクロックを入力します。

非同期スティミュラスダイアログ

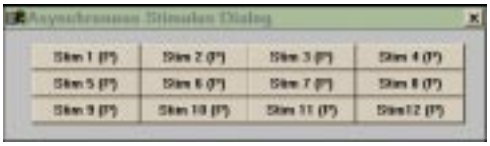
このスティミュラス機能では、入力ピンに印加されている High レベルと Low レベルをダイアログのボタンを使用して入力します。シミュレータでプログラム実行中に、このダイアログのボタンを押して、ピンのレベルを変更できます。

例として、16F84 の PORTB の I/O ピンのレベルをトグルさせるように設定します。

「Debug>Simulator Stimulus>Asynchronous Stimulus…」を選択します。



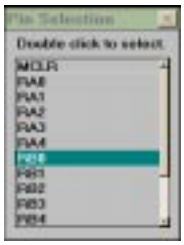
次のダイアログが表示されます。



「Stim1(P)」ボタン上にカーソルを置き、マウスの右ボタンをクリックします。ダイアログが表示されます。スクロールダウンして、「Toggle」をクリックします。

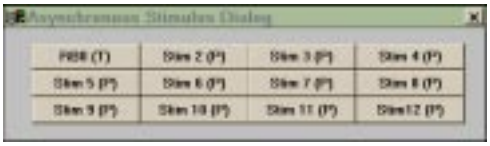
再びカーソルを「Stim1(T)」（P が「Toggle」を意味する T になっている）ボタンの上に置き、マウスの右ボタンをクリックし、「Assign Pin…」を選択します。

16F84 のピンのリストのダイアログが表示されます。



カーソルを「RB0」に合わせて、ダブルクリックします。

非同期スティミュラスダイアログが以下のように表示されます。



ボタンは「RB0(T)」になっていることを確認してください。

「Debug>Run>Animate」を選択して、「連続シングルステップ」モードでプロセッサを走らせます。実行を停止する場合はステータスバーを使用すると迅速に行えます。

非同期スティミュラスダイアログの「RB0(T)」ボタンを押します。「RB0(T)」ボタンを繰り返しクリックするごとに PORTB の 0 番ピンへの入力信号がハイ、ロー交互に変化しているのが「Special Function Register」ウィンドウの PORTB の値が変化することで確認できます。

ピンスティミュラスファイル

ピンスティミュラスファイルは CYCLE **とピン入力レベル**(1 **または** 0)**の列から構成され、**CYCLE **列がストップウォッチの「**Cycle**」の値と一致した時に** 1 **または** 0 **がピンに入力されます。**

「File>New File **」を選び、以下のテキストを入力します。「**;**」や「**!**」のコメントの区切り文字以降は入力する必要はありません。**

```
CYCLE   RB1    RB0
20      0      0
41      1      0   ; apply high to port b bit 1
52      0      1   ; apply high to port b bit 0, set bit 1 low
55      1      1
60      0      0
65      1      0   ; toggle bit 1, then...
76      0      1   ! ...toggle bit 0.
```

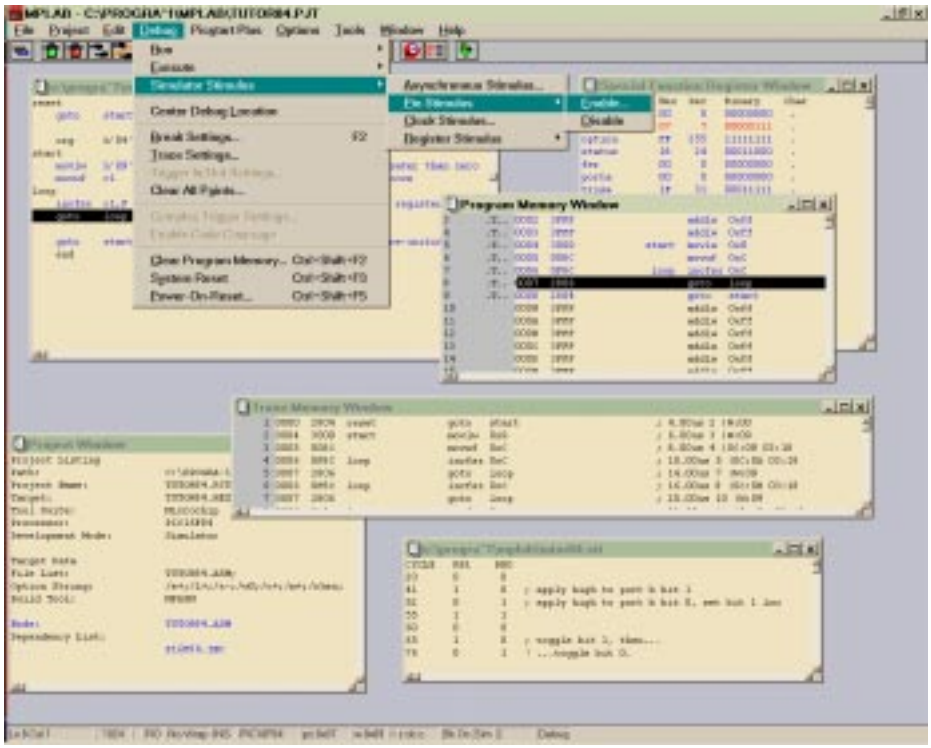
「File>Save As...**」を選択して** tutor84.sti **という名で保存します。**

> **注意**: **以前のバージョンの** MPSIM **と互換性をとるために、一行目は必ず「**CYCLE**」または「**STEP**」というワードで始めます。この** CYCLE **列で同じ行の他の列の値が入力される** CYCLE (MPLAB **のストップウォッチウィンドウ**) **を指定します。**

ファイルの 1 **行目の「**CYCLE**」というワードの後には、**High **および** Low **を疑似入力される** PICmicro **のピン名を記述します。この例では、**PORTB **の** 2 **つのピンである** RB1 **と** RB0 **に疑似入力されます。**

このファイルでは、RB1 (PORTB **ビット** 1) **に適用する値は第** 2 **列に、**RB0 (PORTB **ビット** 0)**の値は第** 3 **列にあります。これらの名前は、シミュレーションする** PICmicro **のピン名と一致していなければなりません。**

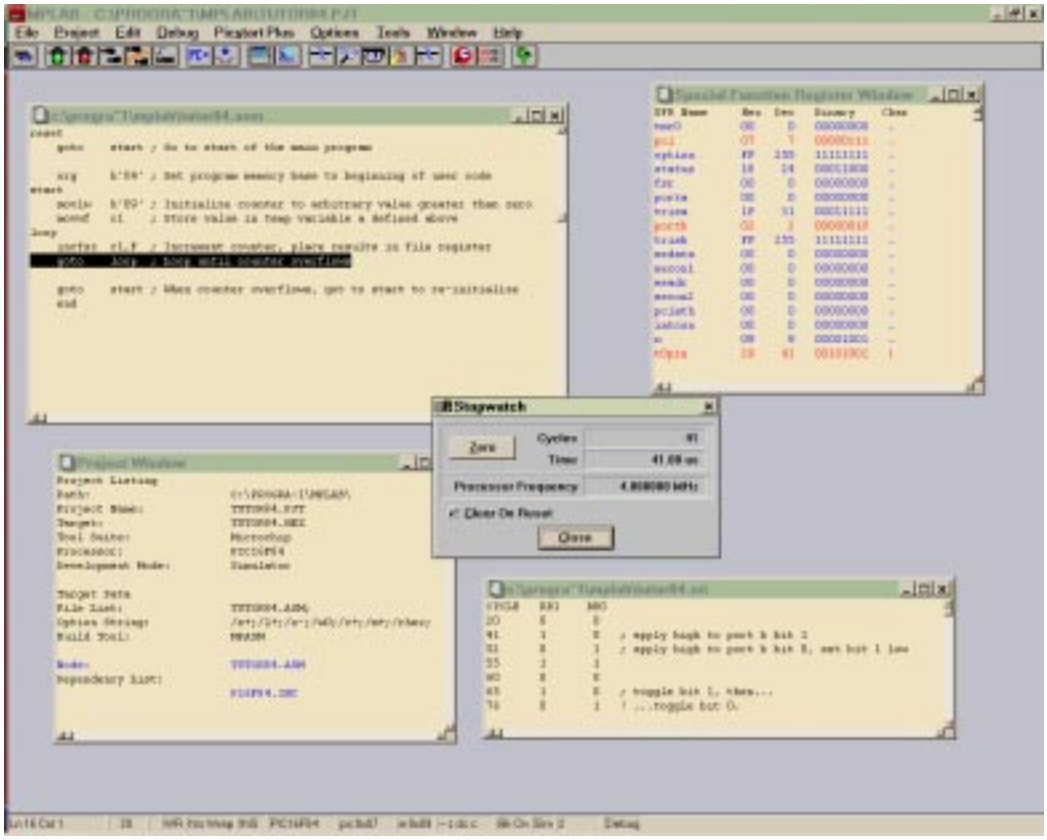
> **注意: 非同期スティミュラスのピン割当てプルダウンに、サポートしているすべてのピンのリストがあります。(疑似入力ボタンを右クリックしてください。)**



「;」または「!」文字の後 1 スペース以上空ければ行にコメントを記述することができます。

それぞれの命令が実行された時の経過時間が、CYCLE 値とクロック周波数により計算され、ストップウォッチウィンドウに表示されます。ストップウォッチを 0 にリセットすれば、ピンスティミュラスファイルもリセットされたことになります。

「Window>Stopwatch」を選択してストップウォッチウィンドウを開きます。さらに「Window>Special Function Registers」を選択します。その中で「portb」を重点的に見たい時、ウォッチウィンドウに portb を追加することもできます。



リセットして、41 サイクルになるまでシングルステップを実行します。これで、スティミュラスファイルの二行目に記述したように、「portb」の値が変化します。

レジスタ･スティミュラス･ファイル

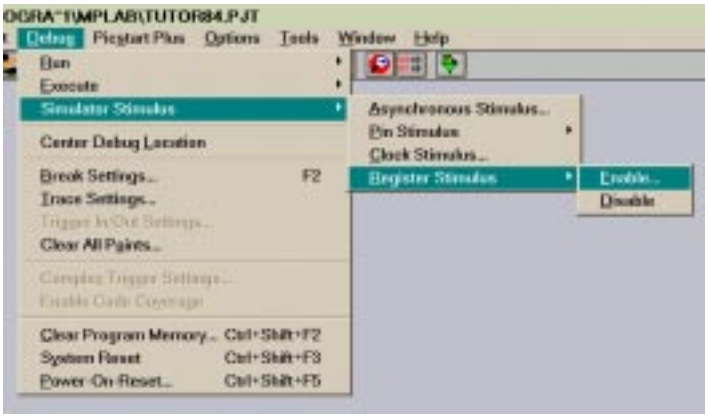
レジスタのスティミュラスファイルは、プログラムメモリのアドレスがレジスタ・スティミュラス・ダイアログで設定したロケーションを実行したときにレジスタに入力される 1 列からなる値で構成されます。これは A/D 変換のシミュレーションに役立ちます。

「File>New File**」を選択して新しいファイルを開き、次の数字のリストを入力します。**

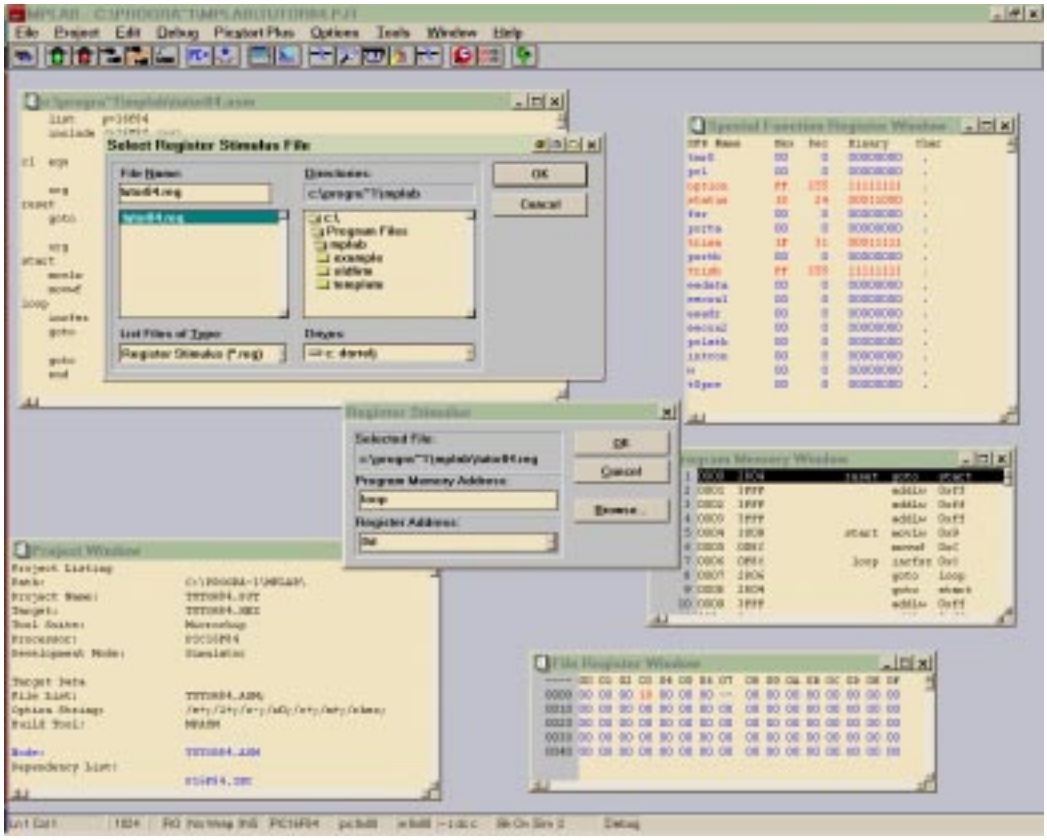
```
10
2E
38
41
50
7A
99
A0
FD
```

「File>Save As…**」を選択し** `tutor84.reg` **という名で保存します。**

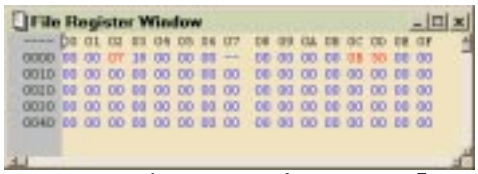
これらの値は順番にレジスタに入力されます。「Simulator Stimulus> Register Stimulus>Enable…**」を選択します。**



デモンストレーションのために、値が入力されるプログラムの場所を 「loop」に設定して、デモンストレーションのために、アドレス 0x0d のファイルレジスタに値を入力してみましょう。該当する欄に「ループ」と「0d」を設定したら、「Browse」を押してファイルダイアログを表示します。次にレジスタのスティミュラスファイルである tutor84.reg を選択します。



「Window>File Registers」でウィンドウを開き、値が入力されたことを確認します。



リセットした後、プロセッサをシングルステップします。「loop」を実行するたびに 0x0D のファイルレジスタの値が変わります。tutor84.reg の中の値が選択したファイルレジスタ (上記で 0x50 の値となっている 0x0D) に順番に入力されるはずです。0x10、0x2E などの値が「loop」を実行するたびに「Debug>Simulator Stimulus>Register Stimulus」ダイアログで選択したレジスタに入力されます。

最後の値 (tutor84.reg 内の 0xFD) が入力された後は、再び最初の値 (0x10) が使用されます。MPLAB-SIM を実行中、リストは循環します。

クロックスティミュラス

クロックスティミュラスは、プロセッサのクロックサイクルで指定したデューティサイクルでピン上に規則的な波形を入力します。

上図の「Debug>Simulator Stimulus>Clock Stimulus…**」ダイアログで、いろいろな疑似入力クロックを入力することができます。上記の設定でプログラムをシングルステップまたは実行した場合、**RB3 **は** 4 **クロックサイクル** High **になり、その後** 6 **クロックサイクル** Low **になります。**RB4 **は** 8 **クロックサイクル** High **になり、その後** 8 **クロックサイクル** Low **になります。**MPLAB **を終了したり、このダイアログで設定を削除するまで、この入力は繰り返されます。**

条件付きブレーク(Conditional Break)

この複雑なブレークポイントのダイアログは特別な複雑な状態のブレークポイントの設定に役立ちます。MPLAB ユーザーズガイドを参照してください。

エディタ機能

MPLAB のエディタにはソースコードの記述、編集に便利な機能がたくさんあります。詳細は MPLAB ユーザーズガイドを参照してください。

ストップウォッチ

ストップウォッチによりコードの実行時間を測定できます。シングルステップでは正確に測定できません。ストップウォッチは、PICmicro のクロック周波数をベースにして時間を計算します。「Options>Processor Setup>Clock Frequency…」ダイアログでクロック周波数を設定します

複数ファイルプロジェクト

リンカを使用すると、複数ファイルのプロジェクトを作成できます。MPLAB v3.40 Project Tutorial を参照してください。

MPASM

MPLINK、MPLIB および MPASM のユーザーズガイドではアセンブラ、リンカ、ライブラリアンの操作について詳細が述べられており、マイクロチップのホームページからダウンロードすることができます。

PICMASTER, ICEPIC, MPLAB-ICE

エミュレータの操作方法のユーザーズガイドはそれぞれ別冊になっています。マイクロチップのホームページからダウンロードすることができます。

PICSTART Plus, PRO MATE

マイクロチップのデバイスプログラマです。MPLAB PRO MATE ユーザーズガイドおよび PICSTART Plus ユーザーズガイドを参照してください。どちらのマニュアルもマイクロチップのホームページからダウンロードすることができます。

MPLAB-C17

17cxxx コンパイラは MPLINK および MPLAB のプロジェクトマネージャ上で使用できます。MPLAB-C17 ユーザーズガイド、および MPLAB v3.40 プロジェクトチュートリアルを参照してください。

サードパーティのツール

HiTech 社 の PIC C、CCS 社の CCS コンパイラ、および Micro Engineering 社 の PIC BASIC を MPLAB 上で使用できます。MPLAB 上で使用方法についてはそれぞれの説明書を参照してください。

問題が発生した場合

www.microchip.com の PICmicro および開発システム会議にアクセスして、初心者から専門家まで対応しているフォーラムで質問してください。Microchip のアプリケーションエンジニア、PICmicro に精通したユーザー、および開発システムエンジニアがこの会議に出席していますので、問題解決、問題報告の最短距離です。また、これまで出た問題を見るだけでも何かの助けになるでしょう。(英語です)

Microchip のホームページでは本、記事、オンライン討論グループなど、その他にも役に立つサイトへのリンクがあります。

日本語での技術的な質問は代理店またはマイクロチップテクノロジー Int'l Inc.日本支社までお問い合わせください。マイクロチップテクノロジーへの技術的な質問は FAX にて FAX:045-471-6122 までお問い合わせください。(技術的な質問用の記入用紙も用意していますので、TEL:045-471-6166 までお問い合わせください)

用語集

アセンブラリスト*(Absolute Listing)* – **プロジェクト構築の際に生成されるファイル。これは実際に生成されたコードとソース命令を表示します。**MPASM **のマクロを使用する場合、またはコンパイラを使用している場合に役立ちます。特にデバッグ時に有効です。なぜなら、このウィンドウの行を追ってコンパイラが生成したコードを確認し、高級言語では見つけにくい微妙な干渉などを突き止めることができるからです。**

アセンブラ – **アセンブラのニーモニック命令をマイクロコントローラで実行するためにマシンコードに変換するプログラム。ニーモニックとマシンコードは一対一で対応しています。**

ブレークポイント – **デバッグを支援するためにプログラムに設定する停止ポイント。プログラム実行中、このプレークポイントで実行を中断し、内部レジスタとユーザー変数の確認ができます。**

プロジェクト構築*(Build Project)* – **１つのプロジェクト内のコンポーネントすべてをアセンブル、またはコンパイルすること。**

コンパイラ – **高級言語の文をマイクロコントローラで実行するためにマシンコードに変換するプログラム。高級言語の１つの文からは、通常多数のマシンコード命令が生成されます。**

開発モード*(Development Mode)* – **ツールがある場合、どのツールでコードを実行するかを設定します。このチュートリアルではソフトウェアのシミュレータである** MPLAB-SIM **を使用しました。**MPLAB-ICE **のようなエミュレータをインストールしている場合には、エミュレータを使用するモードに切り替えることができます。「**Editor Only**」モードではコードを実行することはできません。このモードは主に、**PICmicro **デバイスにプログラム書き込みのみを行う時に使用します。**

デバイスプログラマ– **ファイルからマシンコードをリードして、プログラム可能なマイクロコントローラまたはプログラム可能なメモリチップへプログラムを書き込むハードウェア機器。**

エミュレータ – **アプリケーション上で実際のマイクロコントローラの代わりに使用するハードウェア機器。エミュレータにより、アプリケーション開発の最終段階で行うマシンコードのダウンロード、実行、テストが迅速に行えます。**

ファイルレジスタ – **変数を記憶するために使用する** PICmicro **内の内蔵** RAM **の領域。**

停止*(Halt)* – **ブレークポイントの結果生じるデバッグ機能。対象となるマイクロコントローラを任意のプログラムメモリアドレスで停止させ、レジスタや変数を調べます。**

HEX ***ファイル*** – ASCII **形式のマシンコード。**HEX **ファイルは、マイクロコントローラの命令またはデータを指定するレコードから構成され、プログラム可能なメモリデバイスに格納されます。**

*IDE – Integrated Development Environment****(統合開発環境)の略。*****コード開発のために複数の機能や特性を使用するアプリケーション。別のプログラムを起動しなくてもいろいろなツールを使用できます。**

リンカ – **オブジェクトコードを実行可能なマシンコードに変換するためのプログラム。リンカは、コードをメモリのどこに書き込むか、変数用に** RAM **をどのように使用するかを決定します。**

ノード – Edit Project **ダイアログの** Project Files **ウィンドウの一覧の中にあるファイルの一つ。出**

力であるプロジェクトノードが 1 つと、1 つまたは複数の入力ノードがあります。このチュートリアルで使用している入力ノードは 1 つの MPASM ソースファイルのみです。プロジェクトタイプによっては、複数のアセンブリファイル、C ソースファイル、コンパイル済のオブジェクト、ライブラリ、および、リンカ･スクリプト･ファイルなどたくさんの入力ノードを使用することもあります。

オブジェクトコード – アセンブラまたはコンパイラが生成する中間コード。このコードにはプログラムメモリ、または、内蔵 RAM 変数用のアドレスはありません。しかし、リンカがこのコードをプログラムメモリに配置する際に決定するアドレスのマーカーを含んでいます。

プログラムメモリ – アプリケーション実行用の命令セットを格納しているマイクロコントローラ内のメモリ。

プロジェクト –アプリケーションの作成に使用する複数のファイル群。ファイルを実行可能なマシンコードに変換する際に必要なアセンブラ、コンパイラ、リンカの指定も含みます。これらのファイルはアセンブラのソースファイル、コンパイラのソースファイル、ライブラリ、コンパイル済のオブジェクトファイル、およびリンカスクリプトと呼ばれる疑似命令ファイルなどです。

シミュレータ – パソコン上でマイクロコントローラの疑似動作を行うソフトウェアプログラム。

ソースコード – アセンブラまたはコンパイラが処理するテキストファイル。処理後、中間オブジェクトファイルやマイクロコントローラで実行するためのマシンコードが生成されます。

特殊機能レジスタ – コントローラ、または、周辺機能の内部動作を制御するためにの PICmicro の内部レジスタ。割込み制御レジスタ、タイマ、および 、I/O レジスタなどがあります。

トレース – マイクロコントローラの命令の実行を表示するウィンドウ。エミュレータには、ターゲットプロセッサ動作中にリアルタイムに情報を収集するハードウェア･トレース･アナライザがあります。このデータはトレースウィンドウにアップロードされ、MPLAB で見ることができます。また、シミュレータでもトレースは行えます。

ウォッチウィンドウ – 選択した変数の内容を表示します。ASCII、hex 、バイナリ、浮動小数点などのフォーマットで値が表示できます。