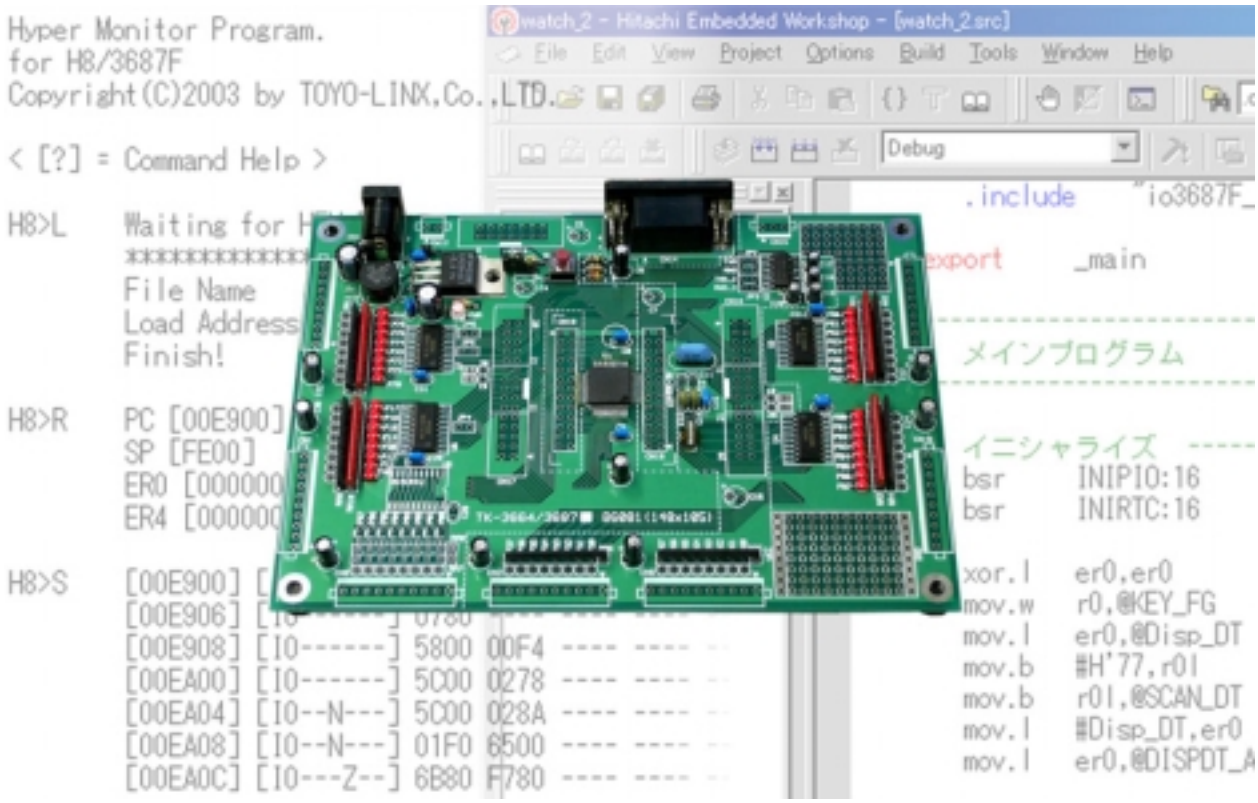
ＴＫ－３６８７

ユーザーズマニュアル

(Ver.1.40)



目 次

Toyo-linx,Co.,Ltd.

1 はじめに、、、

TK-3687 はこれからマイコンを修得しようとする方々が手軽に使えて、マイコンのハードソフトに親しみながら学習できるように作られた簡易リモートモニタ付トレーニングボードです。H8/300H Tiny シリーズのうち標準的かつ多機能な H8/3687（H8/3664 の上位コンパチブル）を採用し、扱い易さと機能の面で 16 ビットワンチップマイコンの主流となった H8 シリーズのベース基板としてご期待に沿えるハードおよびソフト構成になっております。

H8/3687 に用意されている I/O ポートはポート単位で基板周囲から 10 ピンコネクタを介して接続できます。また、インターフェースを配慮し LED による状態表示、プルアップなどの処理がなされています。また、シリアルポートは RS232C レベルで 1 ポート用意されており、9 ピン D-Sub コネクタが取り付けられ市販のモデムケーブル（ストレート）でパソコンと接続できます。

TK-3687 のプログラミングは Tiny、SLP 無償版コンパイラ、HEW（High-performance Embedded Workshop [Tiny SLP]）を使用します。HEW のコンパイラは C 言語とアセンブラの双方に対応しており、メーカによるサポートはないものの実際の製品開発でも使用可能な性能を持っています。

プログラムのデバッグのために、TK-3687 のフラッシュメモリにはあらかじめ“ハイパーH8”が書き込まれています。“ハイパーH8”は Windows95/98 等に標準で搭載されているターミナルソフト「ハイパーターミナル」を使用した簡易モニタです。プログラムを RAM にダウンロードしてデバッグする関係上あまり大きなサイズのプログラムをデバッグすることはできませんが、お手持ちのパソコンと TK-3687 のシリアルポートを RS-232C ケーブルで接続することで、簡単なモニタ環境を作ることができます。ハイパーH8 は、HEW が出力する S タイプファイルのロードやプログラムの実行、ダンプ表示、メモリリード/ライト、レジスタ操作など、簡単なアプリケーションのデバッグであれば十分な機能を備えています。トレース（=ステップ）実行時には、コンディションレジスタ、マシン語、及び逆アセンブルによるニーモニック表示を行います。汎用レジスタを表示させることも可能です。

TK-3687 はフラッシュメモリに書き込まれたプログラムを直接デバッグするためのエミュレータ（E7 等）を接続することができます。エミュレータを使用することで、C 言語でのデバッグ、ラベルの表示、リアルタイムでのトレースなど、本格的な実機デバッグが可能になります。

効果的なマイコンの学習のためには CPU ボードだけではなく、CPU ボードに接続するインターフェースも必要です。本マニュアルには、実習用回路の一例を紹介するとともに、その回路をバージョンアップしたオプションが用意されています。

2004-2005 Toyo-linx,Co.,Ltd.

2 組み立て（キットでお買い求めのお客様へ）

※ 完成品をお買い求め頂いたお客様はこの作業は不要です。次の章へお進み下さい。

■必要な工具

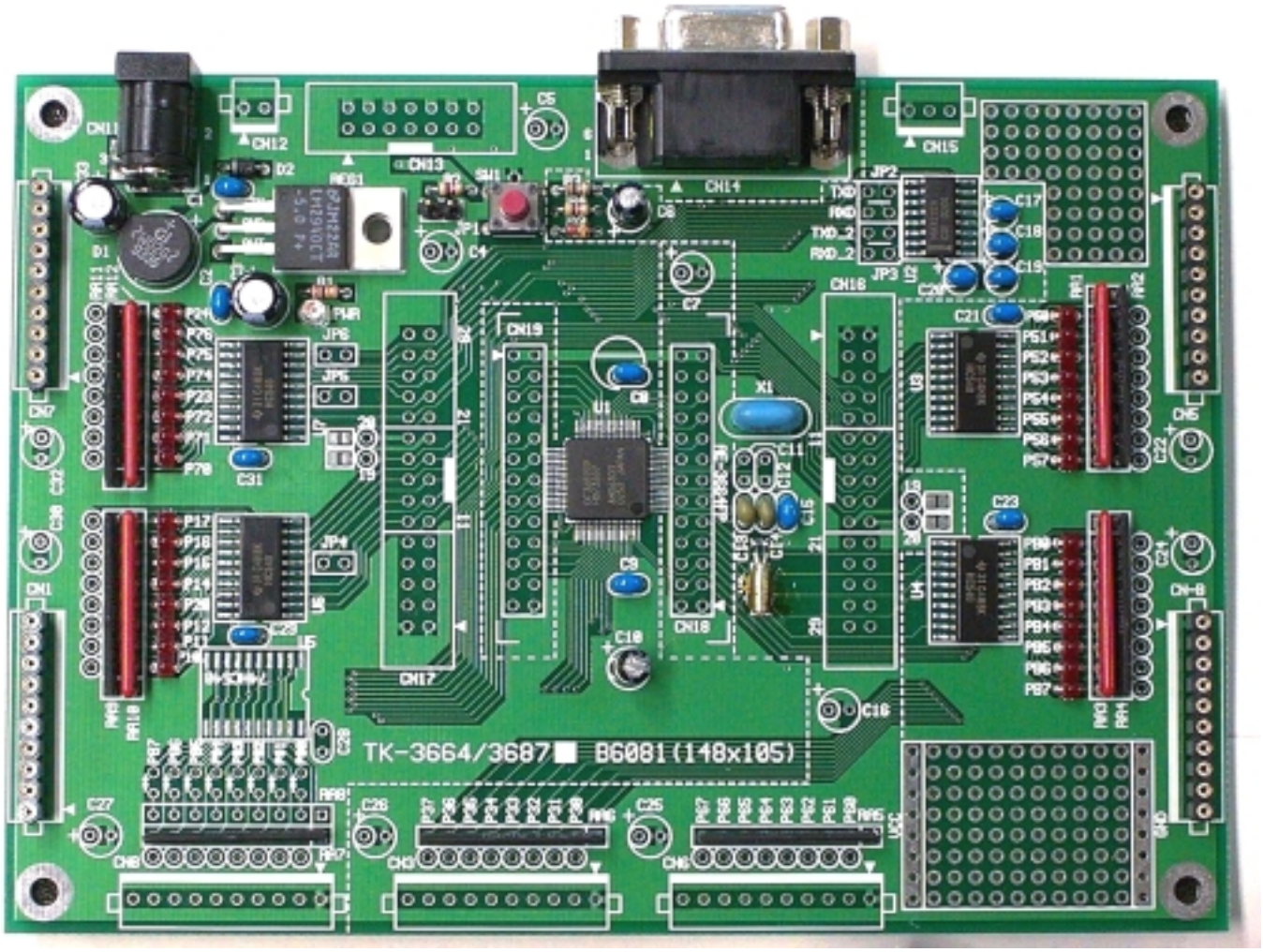
TK-3687 を組み立てるに当たって、次の工具が必要です。

半田ごて、ハンダ、ニッパー

その他、ピンセット、小手先を拭う為の濡らした布切れ等があると便利です。半田ごては大変熱くなりますので火傷には注意してください。うっかり触れてしまった際には急いで氷か水で冷やしてください。

■実装部品と配置

TK-3687 の完成写真を示します。実装を始める前に、大体の部品の種類と半田付けする位置を実際の基板を見て把握しておきましょう。



※写真は CN1,5,7,B に丸ピンを実装した状態です。丸ピンの実装は接続するインターフェースによって使い分けてください。

2004-2005 Toyo-linx,Co.,Ltd.

■部品の確認

まず部品が全てそろっているか、巻末の部品表と照らし合わせて確認して下さい。

■基板の組み立て

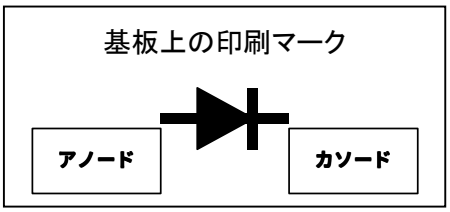
部品の確認ができたら基板に部品を半田付けしていきます。部品は全て部品面（白い印刷のある面）から挿し込み半田面（白い印刷の無い面）で半田付けしていきます。半田付けは次の順で行うと作業がし易いです。

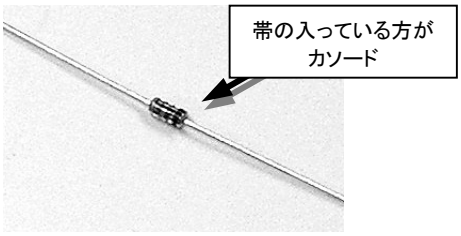
抵抗・ダイオード→LED→セラミックコンデンサ→モジュール抵抗→その他

取り付けに注意が必要な部品のみ以下に示します。挿し間違えない様、よく説明文に目を通してから半田付けして下さい。もし、間違って取り付けてしまった場合には、ハンダをしっかり取り去ってから部品を外します。

※ダイオード

部品に帯の入っている方がカソードです。逆に取り付けない様、基板に印刷されている記号に合わせて部品を挿し込み半田付けして下さい。

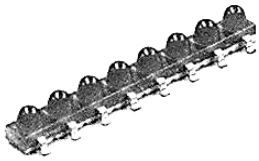




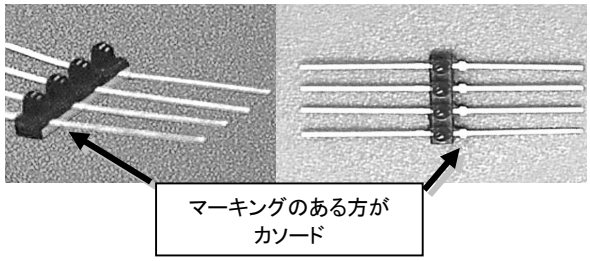
※LED

基板上で“PWR”には透明な LED を、また“Pxx”と記された8ヶ並んでいる場所には4連又は8連 LED モジュールを実装します。極性は、4連、又は8連の LED モジュールはリード（足）に出っ張りのある方、もしくは LED 本体のリードが出ている側面が白色に塗られている方がカソード（K）です。透明な LED は足の短い方がカソードです。基板に印刷されている記号に合わせて部品を挿し込み半田付けして下さい。

LED

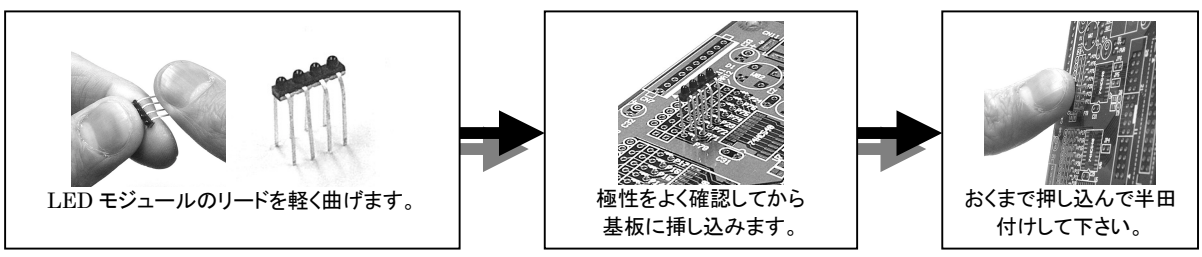


LED モジュール



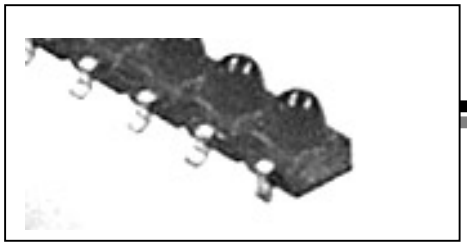
LED モジュールの実装方法を以下に示しますので説明文に従って実装して下さい。

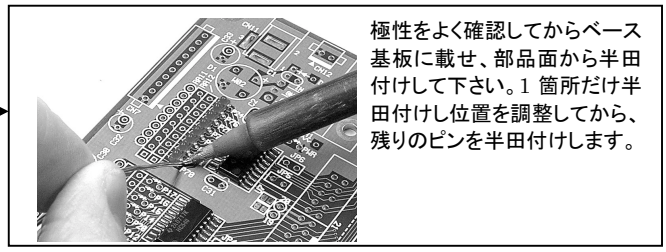
※4連 LED モジュールの実装方法



2004-2005 Toyo-linx,Co.,Ltd.

※8連 LED モジュールの実装方法

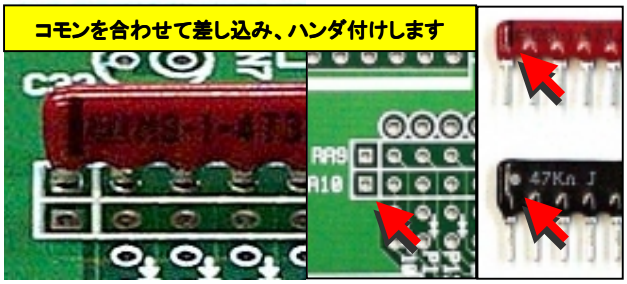




極性をよく確認してからベース基板に載せ、部品面から半田付けして下さい。1 箇所だけ半田付けし位置を調整してから、残りのピンを半田付けします。

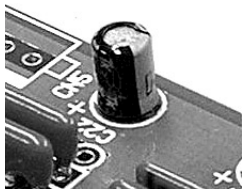
※抵抗モジュール

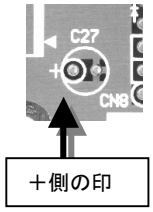
部品端の黒いライン又は白い点側と基板上の四角で囲んである穴(コモン)を合わせて挿し込み半田付けします。尚、同じ形状でも定数が違うので、部品番号と定数を確認してから半田付けして下さい。



※電解コンデンサ

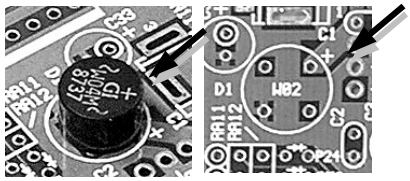
電解コンデンサはリードの長い方が“＋”、短い方が“－”です。また、マイナス側には部品本体に白のラインが入っています。基板上ではプラス側に“＋”印と挿し込む穴が円で囲ってありますので、それに合わせて実装して下さい。





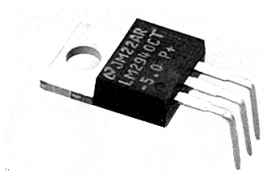
※ダイオードブリッジ

部品上面に“＋”の印刷があるので、基板上の“＋”の穴に合わせて挿し込み半田付けします。



※レギュレータ

リードを根元から約 7mm のところで曲げてから、基板に挿し込んで半田付けします。



全ての部品を実装し終えたら、最後にハンダ面の基板四隅にゴム足を貼り付けて完成です。

動作チェックは簡易モニタ“ハイパーH8”を使用して I/O チェックプログラムを実行します(ハイパーH8 については 4 章で説明していますのでそちらをご参照下さい)。6000h 番地に I/O チェックプログラムが書かれているので、コマンド“G6000”を入力し Enter キーを押して実行して下さい。ポート P1,P5,P7 の LED が 1bit づつ点灯し各ビットをスキャンして行きます。JP4,5,6 がショートされていない場合は、P20,P23,P24 の LED は常に点灯したままです。又、PB ポートは全 bit 入力のみですので、スキャンは行われません。PB ポートのチェックを行う場合は、GND と RA4 の隣のスルーをそれぞれショートし、ショートさせた bit の LED が消灯する事を確認します。この時 GND と Vcc がショートしないよう気を付けて作業を行って下さい。

もし点灯しない LED がある場合は、その LED の極性を確認して下さい。又、スキャンが全く行われない場合はサブクロック(X2)周辺の半田付けを確認して下さい。

2004-2005 Toyo-linx,Co.,Ltd.

3 メモリーマップ

TK-3687 のメモリーマップを示します。マップ中の“ユーザ RAM エリア”の範囲がユーザが自由に使用できるエリアです。HEW 等でプログラムを作成する際下記のアドレスに従ってセクションを割り振ってください（6 章参照）。
ユーザ割り込みベクタはアドレス H’E800 番地にセットします。ユーザ割り込みベクタを H’E800 番地に設定することでハイパーH8はプログラムロード時にリセットベクタを読み込みプログラムカウンタを自動的にセットします。

“E7”エミュレータを使用する場合は H’F780～H’FB7F までの領域は“E7”が使用するので、ユーザは使用することができません。ワークエリア等割り付けないように注意して下さい。

<table>
<tr><th>アドレス</th><th colspan="3">マップ</th></tr>
<tr><td>H’0000

H’DFFF</td><td colspan="2">モニタプログラム
ハイパーH8</td><td>内蔵 ROM
(56kByte)</td></tr>
<tr><td></td><td colspan="2">未使用</td><td>未使用</td></tr>
<tr><td>H’E800</td><td>ユーザ割り込みベクタ</td><td rowspan="3">ユーザ RAM エリア</td><td rowspan="3">内蔵 RAM
(2kByte)</td></tr>
<tr><td>H’E860</td><td>resetPRG</td></tr>
<tr><td>H’EA00

H’EFFF</td><td>P</td></tr>
<tr><td></td><td colspan="2">未使用</td><td>未使用</td></tr>
<tr><td>H’F700
H’F77F</td><td colspan="2">内部 I/O レジスタ</td><td>内部 I/O レジスタ</td></tr>
<tr><td>H’F780

H’FB7F</td><td>変数領域</td><td>ユーザ RAM エリア
※E7 使用時、ユーザ使用不可</td><td>内蔵 RAM
(1kByte)
フラッシュメモリ書き換え用
ワークエリア</td></tr>
<tr><td>H’FB80</td><td></td><td rowspan="2">ユーザ RAM エリア</td><td rowspan="3">内蔵 RAM
(1kByte)</td></tr>
<tr><td>H’FD80

H’FDFF</td><td>Stack</td></tr>
<tr><td>H’FE00

H’FF7F</td><td colspan="2">ハイパーH8
ワークエリア</td></tr>
<tr><td>H’FF80
H’FFFF</td><td colspan="2">内部 I/O レジスタ</td><td>内部 I/O レジスタ</td></tr>
</table>

＜TK-3687 メモリーマップ＞

2004-2005 Toyo-linx,Co.,Ltd.

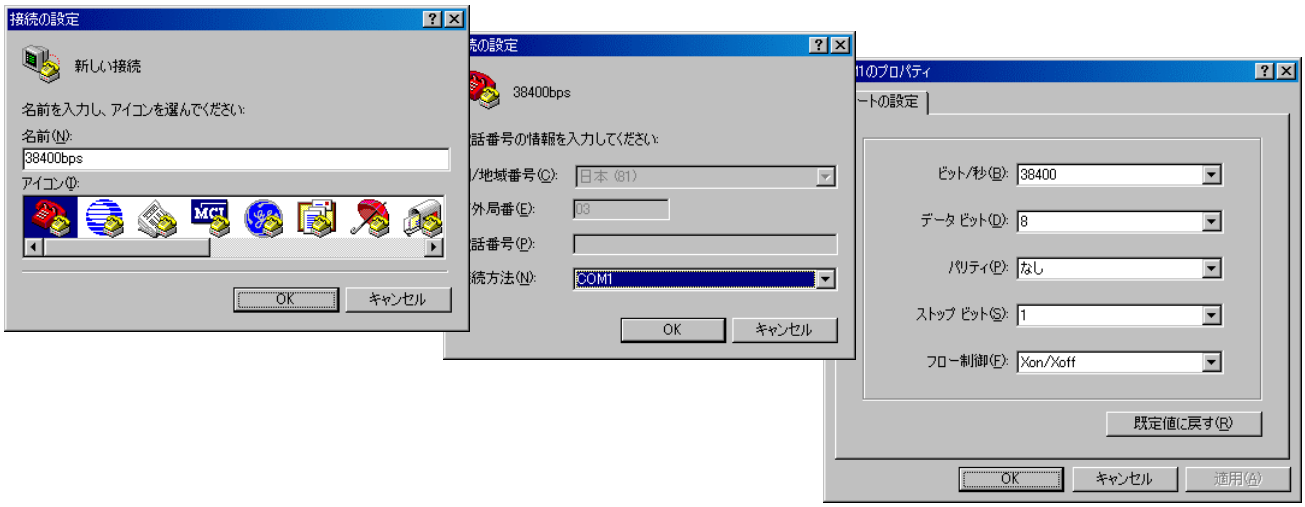
4 簡易モニタ“ハイパーH8”

■ハイパーH8 とは

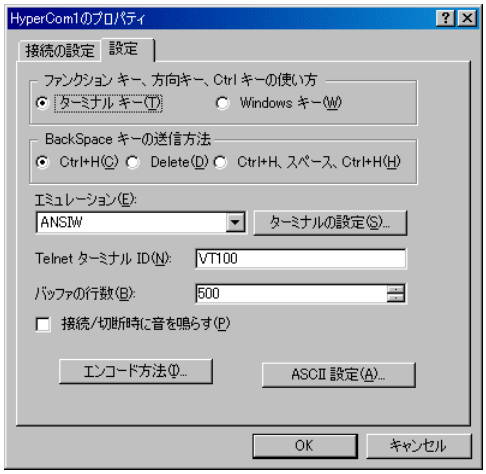
“ハイパーH8”は Windows95/98 等に標準で搭載されているターミナルソフト「ハイパーターミナル」を使用した簡易モニタです。お手持ちのパソコンと TK-3687 のシリアルポートを RS-232C ケーブルで接続することで、簡単なモニタ環境を作ることができます。高度なデバッグ機能を必要としないビギナー向けのモニタで、HEW が出力する S タイプファイルのロードやプログラムの実行、ダンプ表示、メモリリード/ライト、レジスタ操作など、簡単なアプリケーションのデバッグであれば十分な機能を備えています。

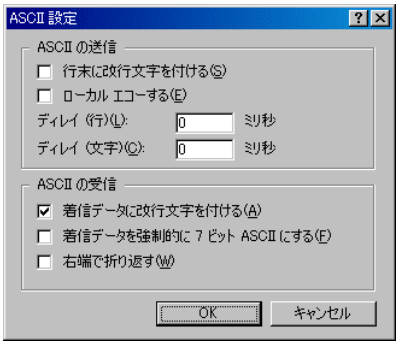
■ハイパーターミナルの設定

ハイパーH8を使用する為にハイパーターミナルの設定を行います。ハイパーターミナルを起動すると設定ダイアログが表示されるのでそれに従い設定して下さい。名前は接続速度がわかるように“38400bps”とします。接続方法は **ComN へダイレクト**(N は接続する Com ポートの番号)、ポートの設定は **38400bps**, **8bit**, **Non-P**, **Stop-1**, **Xon/Xoff** です。設定し終えたら **OK** をクリックします。



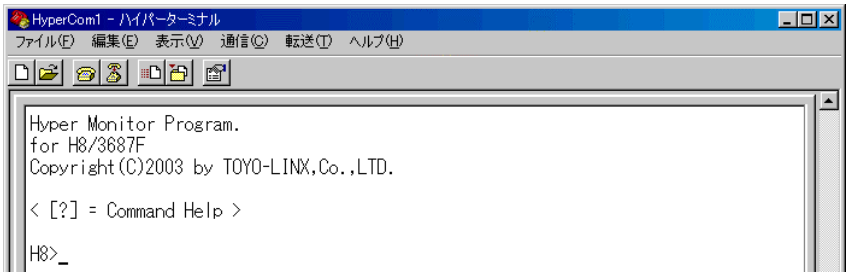
ポートの設定を終えたら次に**ファイル** > **プロパティ**からプロパティを開きます。設定タブをクリックし**エミュレーション**を ANSIW に指定して下さい。次に **ASCII 設定**ボタンをクリックし、ASCII の受信欄内の**“着信データに改行文字を付ける”のみチェック**して OK をクリックします。



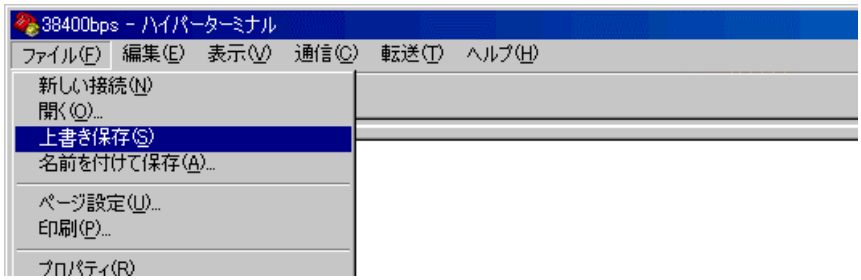


2004-2005 Toyo-linx,Co.,Ltd.

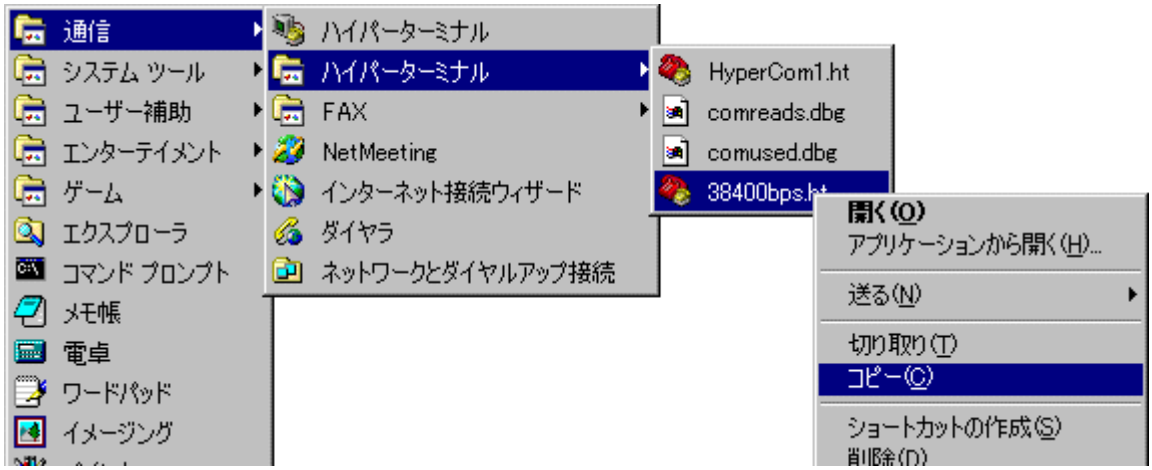
最後に CPU ボードとパソコンとを **RS-232C ストレートケーブル**で接続し電源を入れます。すると下記のような画面が表示されます。



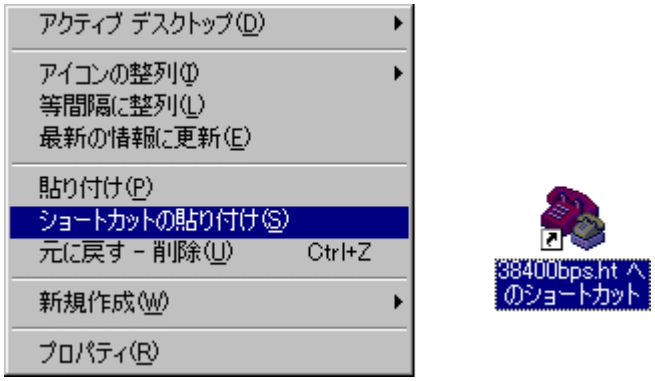
接続できたらハイパーターミナルの設定を保存しておきましょう。 **ファイル > 上書き保存** を選択し保存して下さい。



すぐに呼び出せるようデスクトップにショートカットを作成しましょう。スタートメニューから **スタート > プログラム > アクセサリ > 通信 > ハイパーターミナル > 38400bps.ht** までカーソルを進め右クリックします。プルダウンメニューの中の **コピー** を選択して下さい。



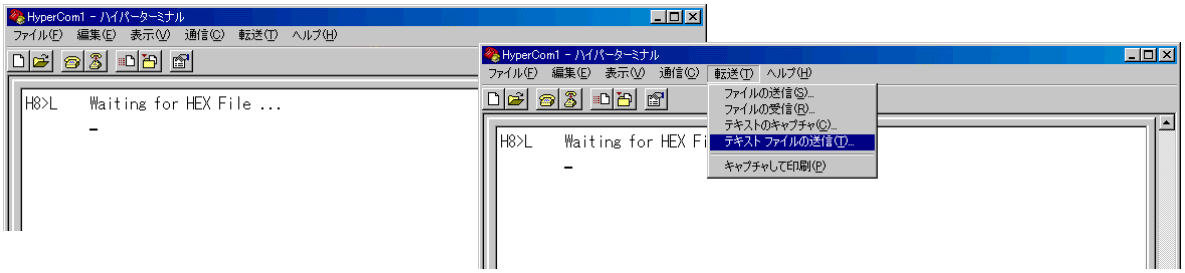
デスクトップで再度右クリックし **ショートカットの貼り付け** を選択してショートカットを作成します。



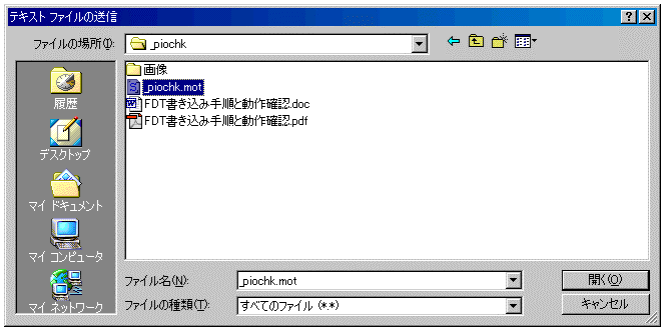
※ここで示した方法は Windows2000 の場合です。上記の方法でショートカットが作成できない場合はエクスプローラやファイルの検索を使用してデスクトップにショートカットを作成してください。

■HEX ファイルのロード

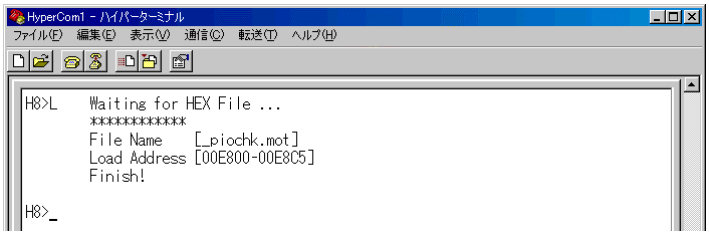
まずは作成したプログラムをロードします。ここでは付属の CD-ROM に添付されている I/O チェックプログラム“_piochk.mot”を例に話を進めていきます。コマンド“**L**”を入力し **Enter キー**押すと“Waiting for HEX File...”の表示とともに HEX ファイルのロード待ちになるので、メニューから**転送 ＞ テキストファイルの送信**を選択します。



テキストファイルの送信ウィンドウから転送する HEX ファイル“_piochk.mot”を選択します。なお、HEW で生成される HEX ファイルはモトローラ形式(.mot)ですので予めファイルの種類を全てのファイル(*.*)にして下さい。ファイルの場所は、E:¥TK-3687¥ﾏﾆｭｱﾙ¥_piochk.mot(CD-ROM ドライブが E の場合)です。

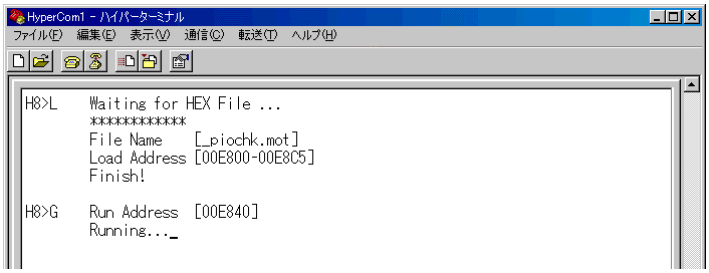


転送の経過は＊で表示されます(プログラムが極端に短い場合は表示されない場合があります)。ロードアドレスと Finish!の文字が表示されればロード完了です。



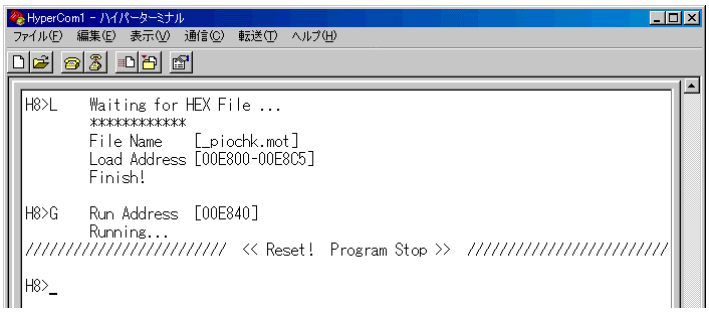
■実行とデバッグ

ロードが終了したらプログラムを実行してみましょう。コマンド“**G**”を入力し **Enter キー**を押します。ロードしたプログラムの先頭アドレスから実行が開始しされ、基板上の LED がスウィングします。

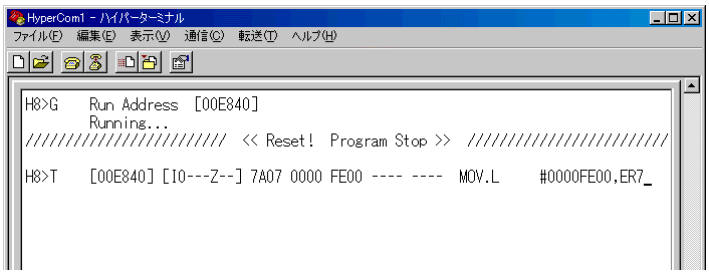


2004-2005 Toyo-linx,Co.,Ltd.

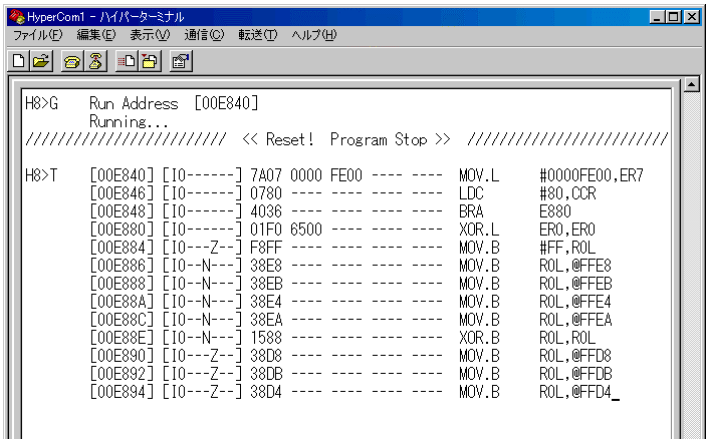
実行を終了する場合はボードの **Reset** ボタンを押して下さい。リセットしてもロードしたプログラムは保持されています。



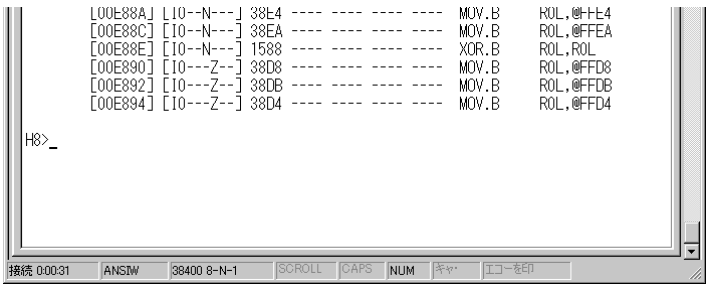
プログラムを一行毎実行して動作を確認したい時はトレース実行を行います。コマンド“**T**”を入力し **Enter キー**を押すと画面に現在のアドレス，CCR，命令（マシン語とニーモニック）が表示されます。但し、この段階では実行はされていません。



トレースを実行するには **Enter キー**を押して下さい。表示されていた行を実行して次の行を表示します。



トレースを終了するには“/”を入力します。

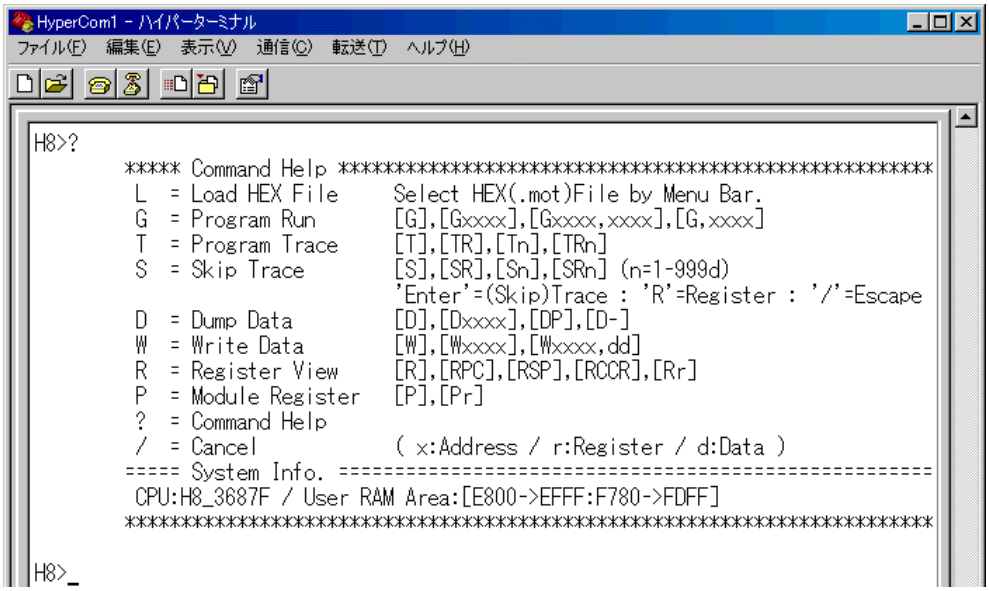


プログラムの流れや CCR の状態を確認しながらトレースを実行しデバッグを行って下さい。尚、6 章では実際の使用例を示していますのでご参照下さい。

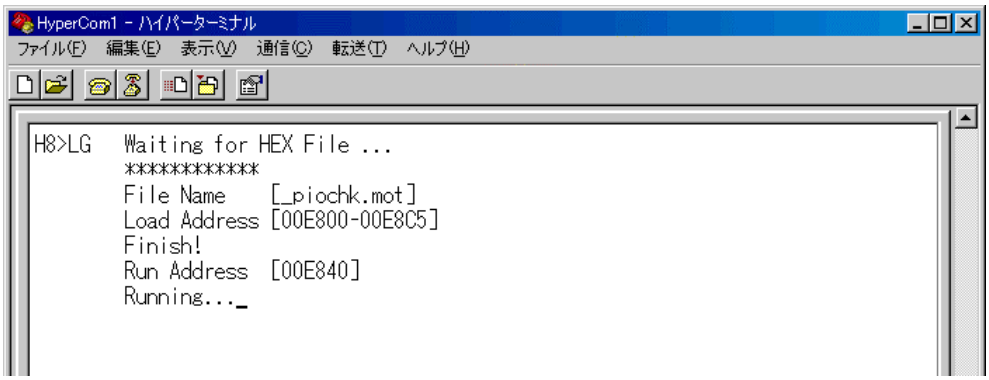
2004-2005 Toyo-linx,Co.,Ltd.

■その他の機能

ハイパーH8 には今まで使用したコマンド以外にもレジスタの表示・変更やメモリダンプ等、デバッグをサポートする機能が盛り込まれています。“?”を入力するとコマンドの一覧と入力形式が表示されますので、初心者でもマニュアル無しに簡単に使いこなすことができます。



更にコマンドの連結が可能で、例えば”**LG**”と入力するとプログラムをロード後、直ちに実行させることもできます。



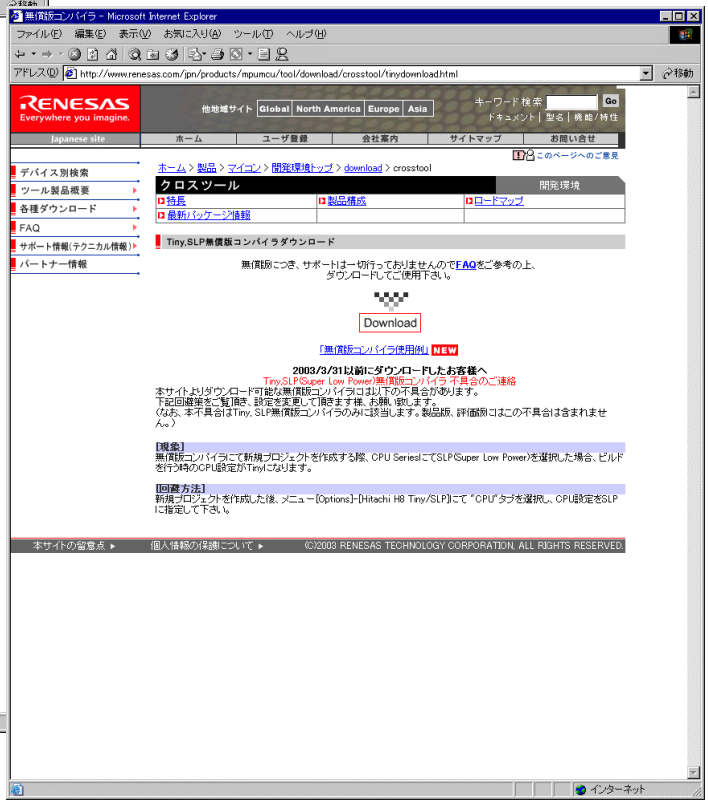
また、直前のコマンドをリピートする際には、**Enter キー**のみで OK です。

2004-2005 Toyo-linx,Co.,Ltd.

5 無償版コンパイラのインストール

TK-3687 のプログラミングで使用する無償版コンパイラ、HEW（High-performance Embedded Workshop）は株式会社ルネサステクノロジのホームページよりダウンロードします。ダウンロードサイトの URL は以下の通りです。

株式会社ルネサステクノロジ
http://www.renesas.com/jpn/



無償版コンパイラ HEW
ダウンロードサイト
http://www.renesas.com/jpn/products/mpumcu/tool/download/crosstool/tinydownload.html

※上記画面・URL は 2003 年 7 月現在のものです。

ダウンロードサイトで“Download”をクリックし必須事項を入力してダウンロードを開始します。入力したメールアドレス宛にファイルを解凍する為のパスワードが送られてきますので、送られてきたパスワードでダウンロードしたファイルを解凍しインストールして下さい。

Attention!!

ここで紹介している「Tiny/SLP 無償版コンパイラ」は今は公開されていません。ルネサステクノロジは現在，High-performance Embedded Workshop V. 4（HEW4）に対応した「無償評価版コンパイラ」を公開しています。「無償評価版コンパイラ」については本マニュアルの「付録 7」をご覧下さい。

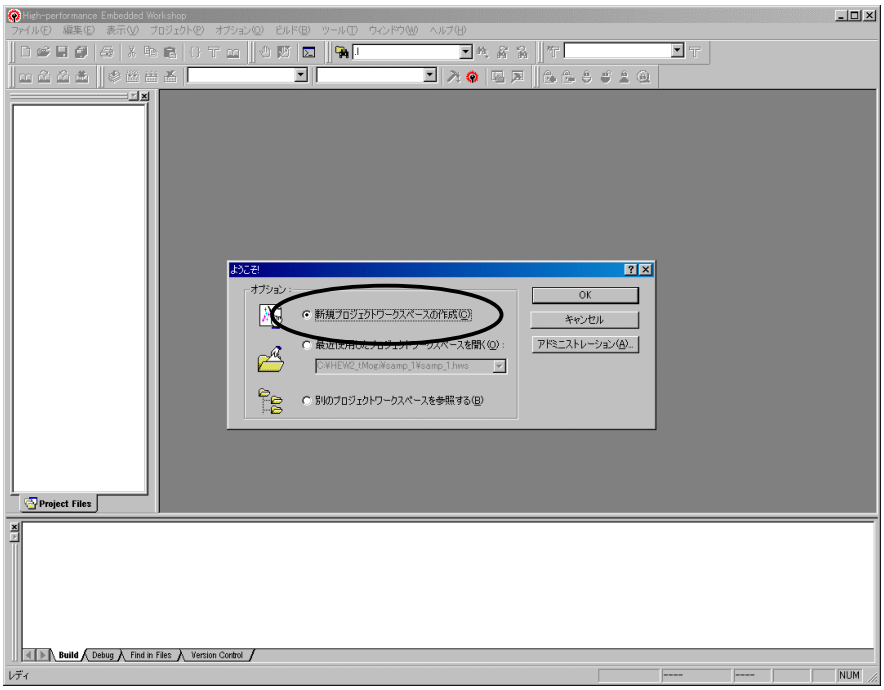
2004-2005 Toyo-linx,Co.,Ltd.

6 プログラムの作成と確認

統合化環境;HEW2 ではプログラム作成作業をプロジェクトと呼び、そのプロジェクトに関連するファイルは 1 つのワークスペース内にまとめられ管理されます。通常はワークスペース、プロジェクト、メインプログラムに共通の名前が付けられます。以下に、新規プロジェクトとして‘samp_01’を作成する手順と動作確認の手順を示しますが、作業に先立ち，HEW2 専用作業フォルダとして，HEW2_xxx（xxxには個人名を入れます）を作っておきます。ここでは，C:¥Hew2_tMogi としていますので，個人名の部分を置き換えてお読み下さい。

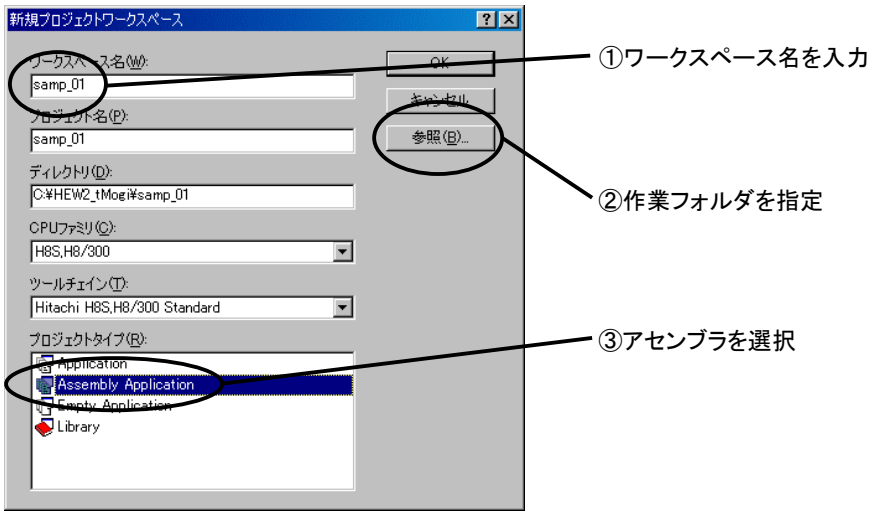
◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

1.HEW2 を開くと、右記の画面が現れるので、新規作成の場合、“新規プロジェクトワークスペースの作成”を選択します。尚、過去に作成したワークスペースであれば、“最近使用したプロジェクトワークスペースを開く”をチェックします。ここでは、新規作成にチェックを入れて OK を押します。すると以下のようなワークスペースの設定画面が開きます。



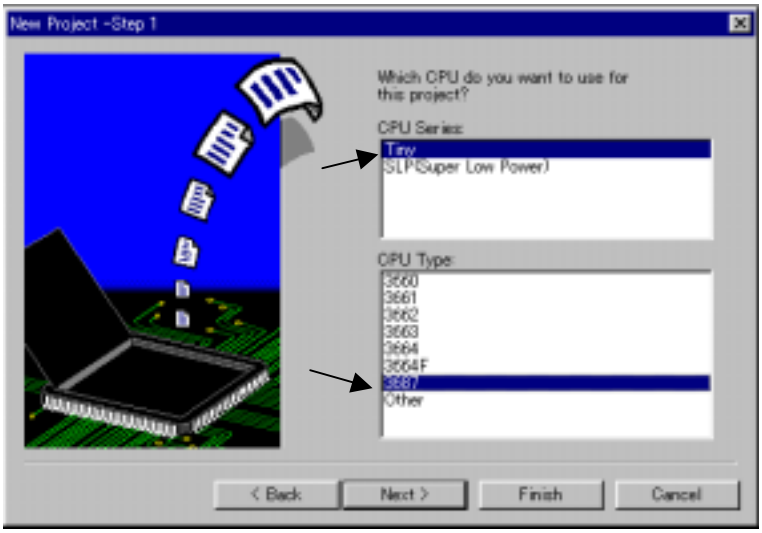
2.先ずワークスペース名を入力します。プロジェクト名と同じとしますから，最初の項目に samp_01 と入力して下さい。プロジェクト名は自動的に同名で入力されます。

ワークスペースの場所は右 3 段目の参照ボタンをクリックし、予め用意した HEW2 専用作業フォルダ（ここでは，Hew2_tMogi）を指定します。設定後、3 項目のディレクトリ欄が正しい事を確認します。次に、作成するプログラム言語を選択します。ここではアセンブラですから、Assembly Application を選択し、OK をクリックして下さい。間違えてデフォルトの Application が選択されると C のファイルが作られてしまいます。

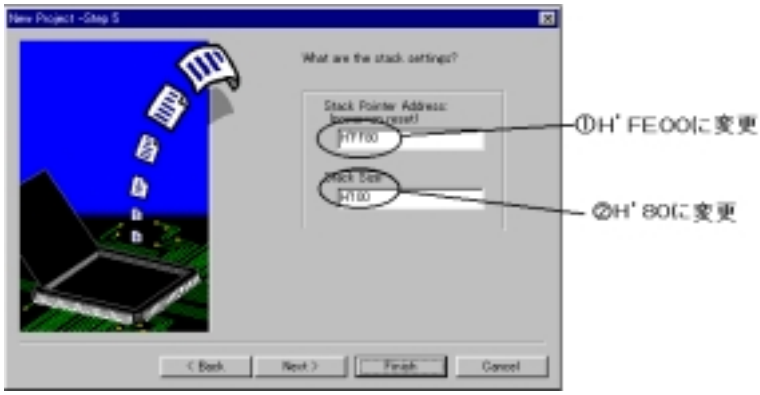


2004-2005 Toyo-linx,Co.,Ltd.

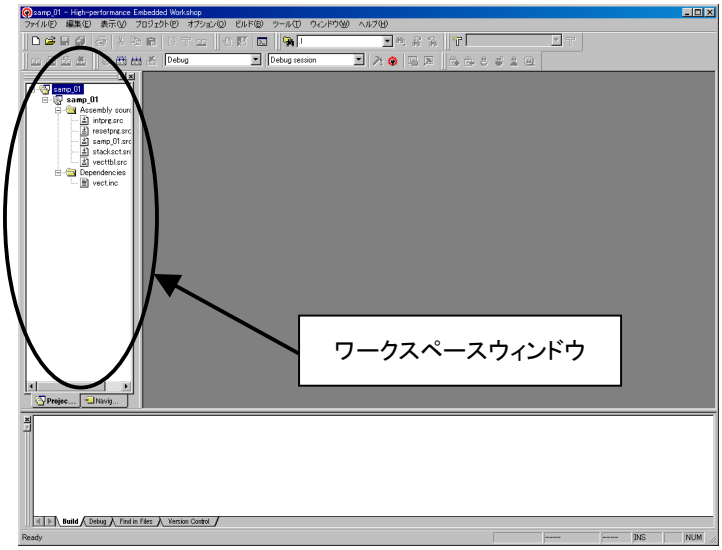
3.使用するCPUのシリーズ(Tiny)とCPUタイプ(3687)を設定し、Nextを続けて3回クリックします。



4.スタックを H'FE00 番地に、スタックサイズを H'80 に設定します。この設定はデフォルトのままでも良いのですが，判り易い変数領域として，H'FD00～H'FD7F を使いたいので，スタックの領域サイズを小さくしておきます。尚，H'FE00 以降は 3 章で示されている通り搭載モニタ;ハイパーH8が変数領域として占有するので，ユーザは使用できません。

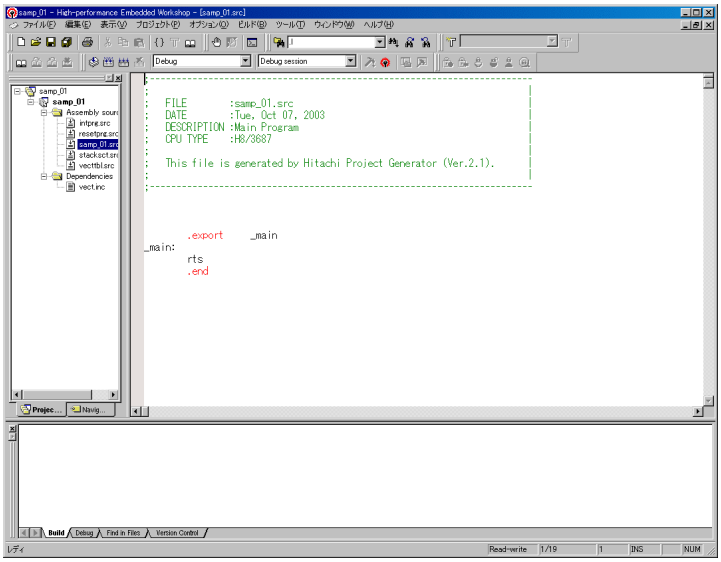


5.更に Next で 3 回送って行き、最後に Finish ＞Project Summary＞OK で、ワークスペースが完成します。統合化環境;HEW2 はプロジェクトに必要なファイルを自動生成し、それらのファイルは左端のワークスペースウィンドウに一覧表示されます。

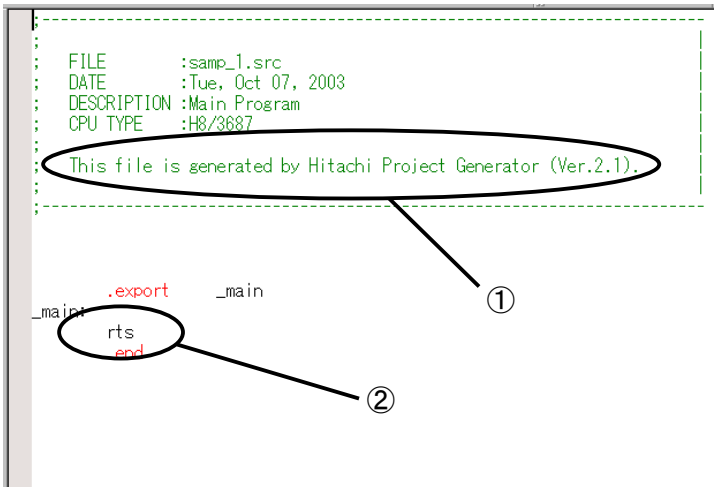


2004-2005 Toyo-linx,Co.,Ltd.

6.HEW2 が作成したメインプログラム；samp_01.src の編集を行なうには、左端のワークスペースウィンドウ内の samp_01.src をクリックすれば自動的に HEW2 のエディタが起動され，samp_01 のソースファイルの編集モードに入ります。右のように画面には HEW2 が自動生成した samp_01 のソースプログラムが表示されます。



7.次に自動生成された samp_01.src の追加・修正を行なうわけですが，右リストの①，②を下記の様に変更します。なお，変数名やラベル名で使われる文字の大文字と小文字は別の文字として判断されますので，注意が必要です。但し，命令語やレジスタ名は区別されません。



① This file is programed by xxx／xxxxx （xxx／xxxxxには所属／制作者名を記入）

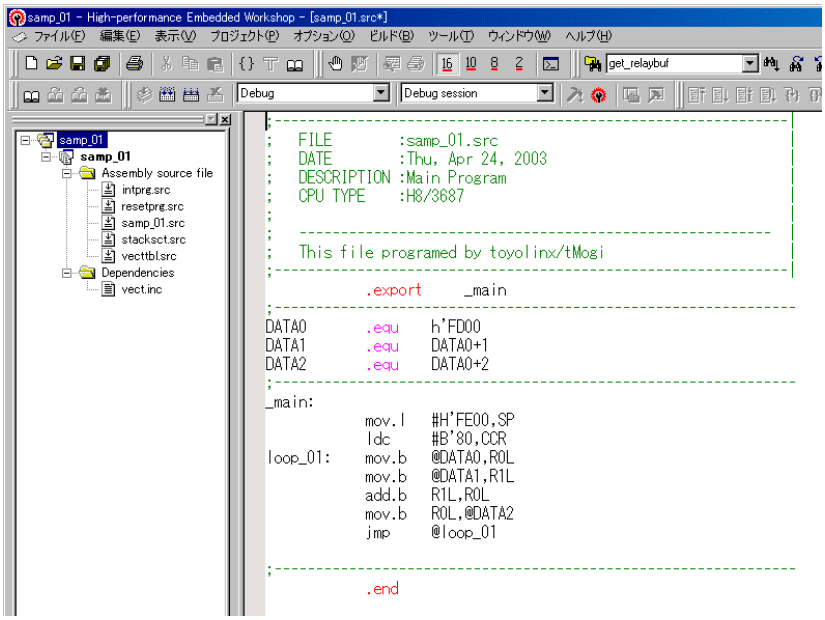
② プログラムはメモリ（FD00,FD01 番地）のデータをレジスタに転送し，それらを加算し，結果をメモリ（FD02 番地）に転送するというものですが，ここでは命令についての説明は後回しにして，アセンブルの手順を覚えることに専念することにします。先ずは下記のリストを打ち込んでみましょう。先ほどの大文字・小文字の注意を思い出し，ラベルは小文字に統一します。但し，レジスタ名は‘1’と‘l’が紛らわしいので，大文字を使うことにします。

```
            mov.l    #H'FE00, SP
            ldc      #H'80,CCR
loop_01:    mov.b    @h'FD00,R0L
            mov.b    @h'FD01,R1L
            add.b    R1L,R0L
            mov.b    R0L,@h'FD02
            jmp      @loop_01
```

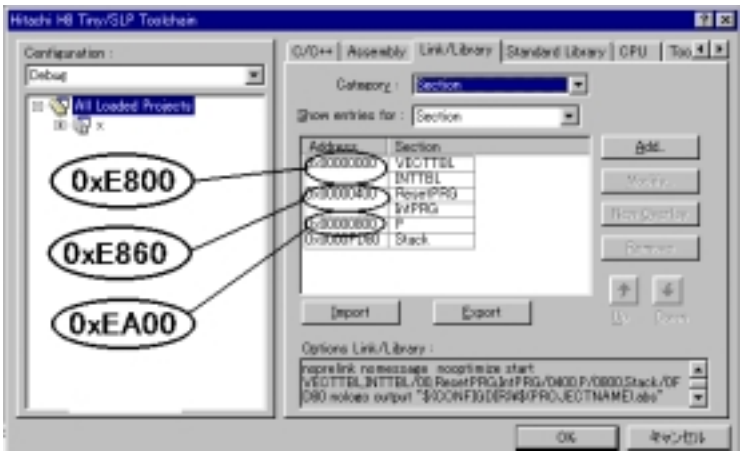

2004-2005 Toyo-linx,Co.,Ltd.

8.変更後の画面を、上書きで保存します。なお，ここでプログラミング上のことで 1 つコメントしたい事があります。

今入力したリストと右画面を見比べて下さい。表現が少し違いますね。それは入力したリストではメモリアドレスを直接書いていますが，右画面ではメモリ番地を.EQU で定義したアドレス名を使っています。アセンブル後の命令には全く差がありません。多少入力ステップが増えるかもしれませんが，判別のし易さや間違いを減らすという意味で，今後アドレスを含め固有の変数は.EQU で定義することにします。



9.次にリンカに渡す各セクションのアドレス設定を行ないます。メニューバーのオプション＞H8 Tiny/SLP…＞Link/Liblary で Category のドロップダウンメニューを開き、Section をクリックすると、右のような各 Section の先頭番地を設定する画面になります。そこで、VECTTBL、ResetPRG、P の各アドレス欄にカーソルを当て、Modify ボタンで、表中の値を図中のように書き換えます（3 章メモリマップ参照）。確認後、OK で閉じますが，その前に今後作成するプロジェクトの事を考慮し，このセクション設定を Export ボタンを使って，HEW2 専用作業フォルダに保存しておきます。ファイル名は section(.his)とします。次の新規プロジェクトではこのファイルを Import すれば，簡単に済ませることができます。



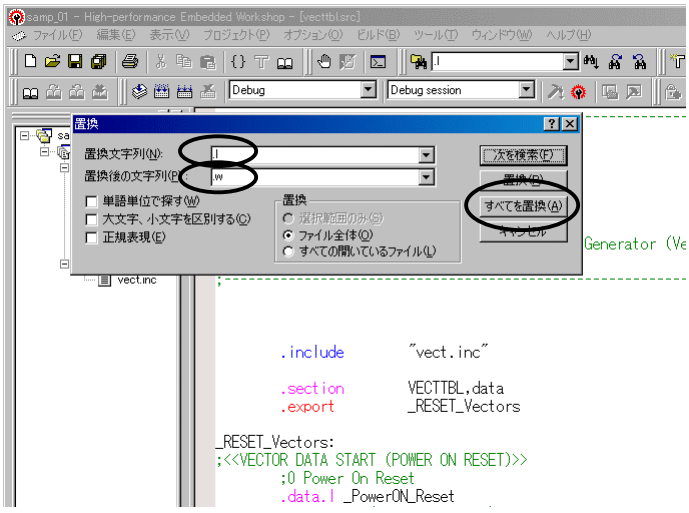
10.アセンブルに入る前にもひとつ重要な変更があります。これは，HEW2 のバグと思われるですが，ワークスペースウィンドウの中の vecttbl.src をクリックし，リスト中の.data.l を.data.w に置換します。置換の手順は以下の通りです。

編集＞置換＞置換文字列に‘.l’

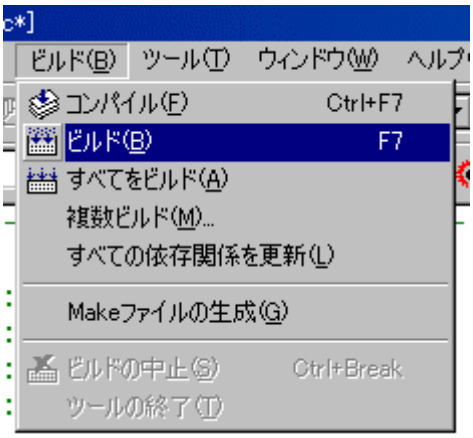
＞置換後の文字列に‘.w’＞すべてを置換

以上の変更で，アセンブルの準備が全て整った事になります。

（尚、この置換作業を省略する手順を付録-4 Topics-2 に収録しています。そちらも合わせてご参照下さい）



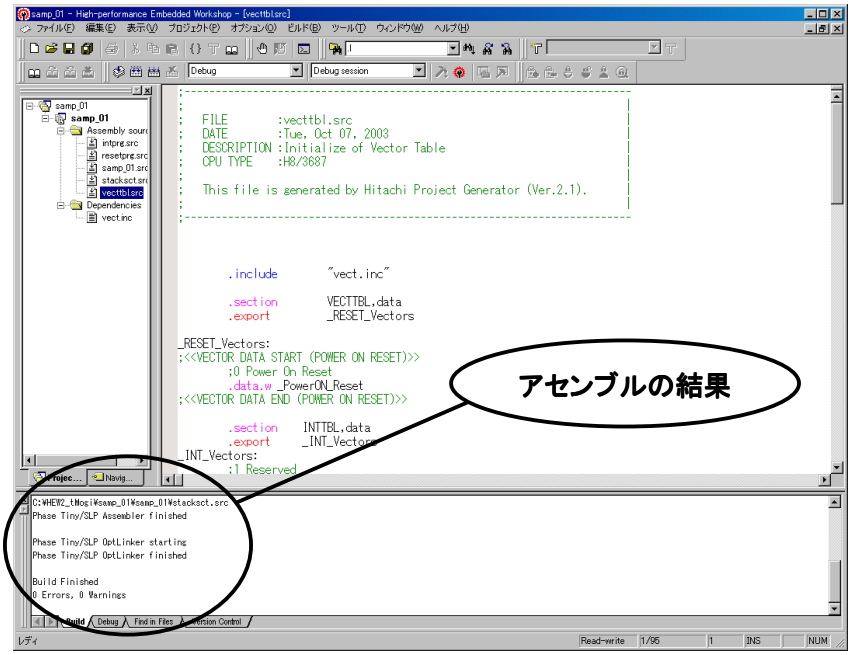
11.さて，次はいよいよアセンブルです（ここが第 1 の関門で，プログラムが通るかどうかが問題です）。メニュバーのビルド＞ビルドでアセンブルを開始します。尚，この操作はツールバーのアイコンにも用意されています。何れかの操作をしてみて下さい。



12.デバッグウィンドウにアセンブル結果が表示され、プログラムの表記上の間違いの有り無しが表示されます。

エラーがある場合はソースファイルを修正しなくてはなりません。その際、エラー項目にカーソルを当てクリックすると、エラー行に飛んで行きます。（この辺りの機能が統合化環境の有り難いところです）

エラーがなければ、搭載モニタ；ハイパーH8 を使い、マイコンボードへファイルをダウンロード、トレース実行し、実際の動作をマイコン上で確認する作業に入ります（手順 13 以降）。ここでは，HEW2 は閉じずに最小化しておきましょう。もし，以降で動作確認して，NG（No Good の意）の場合は再度 HEW2 へ戻り，プログラム修正する必要があるからです。



もし，ここで暫く作業を中断する場合はファイル保存し，右上の終了ボタンをクリックし、はい（Y）をクリックします。次回の作業再開は samp_01 のワークスペースファイル（samp_01.hws）をクリックすれば，HEW2 が立ち上がり，終了時の画面が復帰します。

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

これからはプログラムの確認作業に入ります。ハイパーH8 を使用する方はこのまま読み進めて下さい。エミュレータ“E7”をお使いの方は P.39 の‘付録-3 エミュレータを使ったダウンロードと実行方法’を参照願います。

13.先ず，4 章のハイパーH8 で作成したハイパーターミナルのショートカット，‘38400bps’をクリックします。次に，TK-3687 の電源を入れます。もし，既に電源が入っている場合には基板上辺のリセット SW を押して下さい。

14.L（ロード）コマンドで，samp_01.mot をマイコンへダウンロードします；　　H8＞L

この操作の詳しい説明は 4 章で行なっていますが，手順だけならば次のようになります。

L 入力＞メニュバーの転送＞テキストファイルの送信＞全てのファイルを選択＞C:¥…¥samp_01¥samp_01¥Debug 内の samp_01.mot を開く。

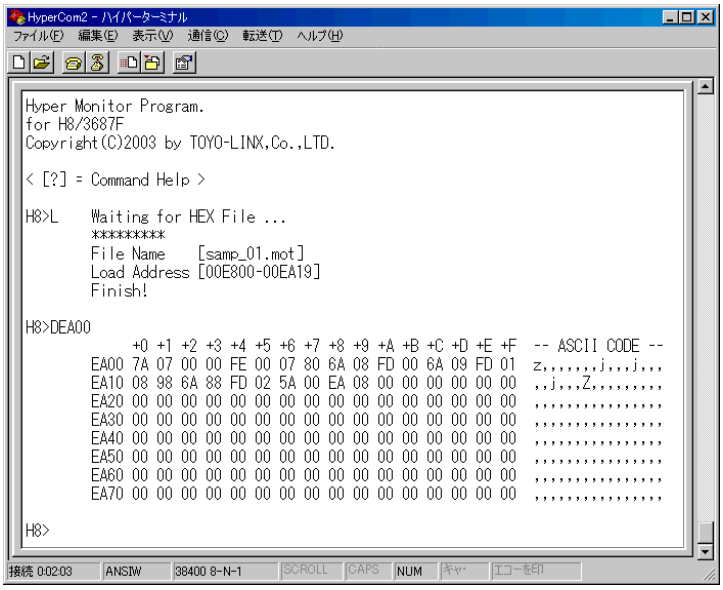
2004-2005 Toyo-linx,Co.,Ltd.

15.. 先ずは D(ダンプ)コマンドでダウンロードされたプログラムを確認します;

H8＞DEA00

★EA00～EA10 に 00 以外の 16 進数(これがアセンブラが生成した samp_01 のマシン語命令)が書き込まれていればダウンロード成功！

尚、ダウンロードプログラムの確認には DP コマンドもあります。



16.同じく D コマンドで，FD00 からの変数エリアの初期値が 00 である事を確認します;

H8＞DFD00

17.トレース実行する前にプログラムの先頭番地をプログラムカウンタ;PCにセットしなくてはなりません。何故かと云うと，PCはプログラムの実行アドレスが格納される重要なレジスタで，実行は現在のPCから開始されるからです。従って，プログラムを実行する際には必ずPCをチェックしたり，予め開始番地をセットする必要があるわけです。使うコマンドはRPC(レジスタPCセット)コマンドとRコマンドで，PCの設定と確認を行ないます;

H8 ＞RPCEA00

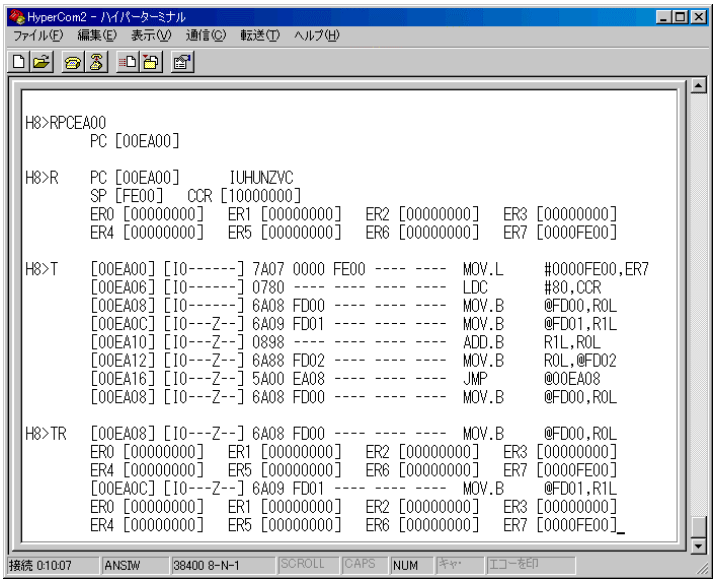
18.T 又は TR(トレース&レジスタ表示)コマンドで，プログラムのトレース実行を行ないます;

H8＞T,Enter,Enter,…

or

H8＞TR,Enter,Enter,…

★先ずは，jmp 命令が書かれている EA0E 番地で，ループの頭;EA00 番地に戻ることを確認します。



この段階では，メモリが全て 0 にクリアされているので，一見レジスタ(R0,R1)の値は 0 から変化していない様に見えますが，転送命令により 0 に書き換えられているわけです。次に，メモリに何かデータを書いて，同様な操作を行なってみましょう。

その前に注意事項が 1 つあります。T,TR と入力するとアドレスや命令が表示され，カーソルが行末で点滅してますが，この段階では実行はされていない事に注意して下さい。レジスタの値も実行前の値です。表示されている命令とレジスタは Enter を入力したのち，実行され変化します。トレース実行を中断する際には‘／’を入力します。尚、再度トレースなどプログラムを実行する際には必ず R コマンドや RPC コマンドで PC のチェック・再設定を行います。

2004-2005 Toyo-linx,Co.,Ltd.

19.W(ライト)コマンドで，変数エリアにデータを書きます；

H8＞WFD00　　FD00 番地に，56 を書く；　　5,6,Enter

FD01 番地に，78 を書く；　　7,8,Enter

FD02 番地に，00 を書く；　　0,0,Enter

20.D コマンドで FD00，FD01 番地のデータが変更されたか確認します；　H8＞DFD00

21.プログラムカウンタの設定（手順 17 と同様）；　H8＞RPCEA00

22.TR コマンドで，jmp 命令のところまでトレースとレジスタ表示を行ないます；

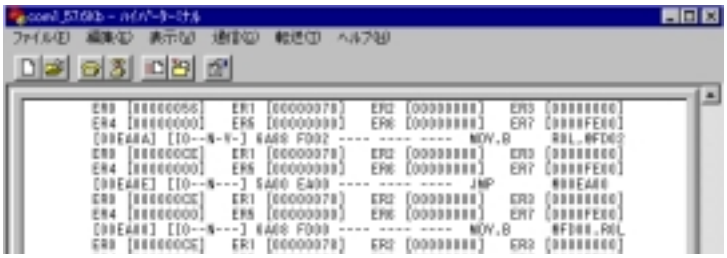
H8＞TR

★命令の実行と共に，R0L，R1L のレジスタの値が 56，78 に変化する事を確認します。

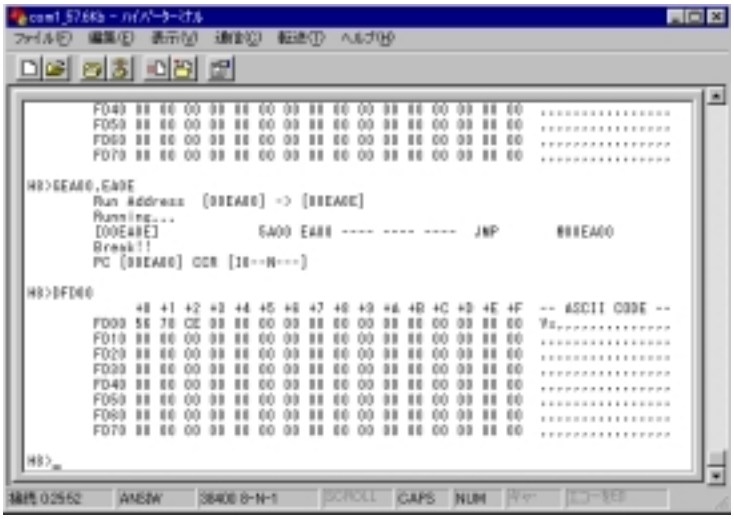
加算命令が実行された時点で，R0L が CE になることも確認します。

23.D コマンドで変数エリアをダンプし，FD02 番地の結果が CE に変化した事を確認します；

H8＞DFD00



24.以上は TR コマンドでプログラムを実行しましたが，G コマンドで実行させる事もできます。その場合はリセット SW で実行を中断させます。或いはブレーク付き G コマンドで実行も可能です。Samp_01 の場合，jmp 命令が書かれている EA0E 番地でブレークを掛けてみます（右下図）；



H8＞GEA00，EA0E

リセット又はブレーク後は，先程と同じように D コマンドで FD00 番地からをダンプして結果を確認します。

ブレーク付き G コマンドは長いプログラムの一部をトレースしたい場合に有効です。トレースする直前でブレークを張り，その後トレース実行に切り替えるわけです。その場合は予めブレークを張るアドレスを調べておく必要があります。

尚，H8 シリーズ共通して云えることですが，ブレークを張る場所には若干制限があります。詳しくは付属の CD を参照して下さい。

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

2004-2005 Toyo-linx,Co.,Ltd.

25.本章の最後に

以上が HEW2 におけるプログラム作成から動作確認までの手順です。ここでは，samp_01 を例にし説明をして来ましたが，これで一通りプログラムの作成手順が飲み込めたのではないでしょうか。この後は，各自で簡単なプログラムにトライして見て下さい。例えば，samp_01 の add の部分を他の算術命令（sub など）に置き換え，プロジェクト名を ‘samp_01a’ に代え同様な作業を試みては如何でしょうか。

大まかな命令の意味やフラッグの変化，条件ブランチなどが理解できれば，第 1 ステップ合格です。また，実習・応用プログラムを次章以降に載せてありますので，それらを参考にすれば，入力/出力ポートの制御やより高度なプログラミング・テクニックを身に付けることが可能です。

文中では，第 2 の関門は出てきませんでしたが，第 1 の関門をクリアしダウンロード・実行して，旨く動作しないこともあります。それが第 2 の関門です。プログラムの内容にも依りますが，第 2 の関門をクリアすることが極めて大変な辛い仕事になります。そこで机上デバッグやモニタを駆使してバグを探すわけですが，一番重要なことは ‘何か変だ‘とか’ 何かちょっと違う‘といった些細な疑問や変化を決して見逃さない事です。それを心がければ，バグを見つけることはそう大変なことではありません。

2004-2005 Toyo-linx,Co.,Ltd.

7 実習プログラム

この章ではH8の命令とプログラミングに慣れることを目標に，実習プログラムを作ります。各プログラムにはそれぞれ幾つかの課題がありますので，リストのプログラムが完成したら，課題に挑戦してみて下さい。これで，プログラミングの基礎力が付きます。ここでしっかり基礎を作れば，独自にプログラムを作ることもできるはずですし，次章の応用プログラムにも入って行けます。なお,次章の応用プログラムではハードの追加が要りますが，本章では一切必要ありません。

実習に入る前に一般的なプログラムの構造を表に示します。サブルーチン領域が無かったり，どの実習プログラムでも当てはまるわけではありませんが，概ね以下のようなブロックに分かれていることを把握して，リストと照らし合わせてみて下さい。

領　域	内　　容
タイトル欄	HEW2が自動作成するタイトル;プログラム名，作成日，CPU名等 プログラマの所属，名前 プログラムの説明やデータの割り振り 修正履歴など
参照宣言	外部から参照されるラベルや変数を宣言する
アドレス，変数定義	I／Oアドレスや変数の定義をする
初期化プログラム	I／Oの入出力などの設定を行なう 変数の初期値の設定を行なう （ここがプログラムの開始位置となるので，必ず，_main:を書き入れる）
メインプログラム	ループさせ，ループの中で色々な処理を行なう （このループを通常メインループと呼ぶ）
サブルーチン領域	メインプログラムで呼ばれるサブルーチンはこの場所にまとめる
プログラムの終了	. END

実習プログラムは3つあり，プログラムリストを以降に示します。先ずはリストの通りに入力し実行して見ましょう。但し，タイトルやコメント（;以降）は適当に省略して構いません。

2004-2005 Toyo-linx,Co.,Ltd.

```
;-------------------------------------------------------------|
;   FILE        :samp_02.src                                  |
;   DATE        :Thu, Jul 31, 2003                            |
;   DESCRIPTION :Main Program                                 |
;   CPU TYPE    :H8/3687                                      |
;   ----------------------------------------------------      |
;   This file is programed by toyolinx/tMogi                  |
;-------------------------------------------------------------|
;   ●samp_01の復習                                            |
;   ●モニタでメモリのデータ書き換えをする代わりに，転送命令で  |
;     メモリに書き込む。                                      |
;       56--->DATA0,78--->DATA1,34--->DATA2                   |
;   ●ADD,SUB命令を組み合わせる。                              |
;       56+78   --->DATA4                                     |
;       56+78-34--->DATA5                                     |
;   ●演算結果をポート５へLED表示する。                        |
;                                                             |
;-------------------------------------------------------------|
;   ●課題に挑戦！                                            |
;   課題１．ADD,SUB実行後のゼロ,キャリーフラッグの変化を追う。 |
;   課題２．ADD，SUB命令をAND,OR命令に置き換える。            |
;   課題３．ADD，SUB命令をXOR,NOT命令に置き換える。           |
;   課題４．                                                  |
;                                                             |
;-------------------------------------------------------------|
            .export     _main
;-------------------------------------------------------------
DATA0       .equ        h'fd00
DATA1       .equ        DATA0+1
DATA2       .equ        DATA0+2
DATA3       .equ        DATA0+3
DATA4       .equ        DATA0+4
DATA5       .equ        DATA0+5
PDR5        .equ        h'ffd8
PCR5        .equ        PDR5+h'10

;-------------------------------------------------------------
_main:      mov.b       #h'ff,r3l       ;PORT5を出力に設定
            mov.b       r3l,@PCR5

            mov.b       #h'56,r4l       ;56をDATA0にセット
            mov.b       r4l,@DATA0
            mov.b       #h'78,r4l       ;78をDATA1にセット
            mov.b       r4l,@DATA1
            mov.b       #h'34,r4l       ;34をDATA2にセット
            mov.b       r4l,@DATA2

loop_01:    mov.b       @DATA0,r0l
            mov.b       @DATA1,r1l
            mov.b       @DATA2,r2l
            add.b       r1l,r0l         ;r1l+r0l=>r0l
            mov.b       r0l,@DATA4      ;r0lをDATA4に格納
            mov.b       r0l,@PDR5       ; 〃 PDR5に出力
```


2004-2005 Toyo-linx,Co.,Ltd.

```
            sub.b       r2l,r0l         ;r0l-r2l=>r0l
            mov.b       r0l,@DATA5      ;r0lをDATA5に格納
            mov.b       r0l,@PDR5       ; 〃 PDR5に出力
            jmp         @loop_01
;-------------------------------------------------------------
            .end
```

7－2.“samp_03”;　　AND命令，OR命令で1つの LED を点滅制御する

```
;-------------------------------------------------------------|
;   FILE        :samp_03.src                                  |
;   DATE        :Thu, Jun 19, 2003                            |
;   DESCRIPTION :Main Program                                 |
;   CPU TYPE    :H8/3687                                      |
;   This file is programed by toyolinx/tMogi.                 |
;-----------------------------------------------------------  |
;   ●ポート５内の１つのＬＥＤを１秒間隔で点滅させる。        |
;   ●１秒のウェートはサブルーチンとして作成する。            |
;                                                             |
;-----------------------------------------------------------  |
;   ●以下の演習課題に挑戦!                                   |
;                                                             |
;   課題１．LEDのON/OFF時間を変更するには。                   |
;   課題２．出力するLEDをP51～P57の何れかに変更する。         |
;   課題３．ポート５以外のポート１又は７に変更する。          |
;   課題４．表示のパターンを変更する(例えば，55など)          |
;   課題５．                                                  |
;                                                             |
;-------------------------------------------------------------|
        .export     _main
;------------ 変数  & I/O_Address_Definition ------------------
PCR5        .equ    h'ffe8
PDR5        .equ    h'ffd8
wait1000    .equ    h'140000

;------------------Initialize_Routine -----------------------
_main:
        mov.b   #b'00000001,R0L     ;ポート5,ﾋﾞｯﾄ0を出力にする設定値
        mov.b   R0L,@PCR5           ;ポートコントロールレジスタに書く
        xor.b   R0L,R0L             ;R0Lを0

;---------------------- Main_Loop -----------------------------
loop_01:
        or.b    #b'00000001,R0L     ;ポート5,ﾋﾞｯﾄ0を1に
        mov.b   R0L,@PDR5           ;LEDは点灯する

        jsr     @wait1s

        and.b   #b'11111110,R0L     ;ポート5,ﾋﾞｯﾄ0を0に
        mov.b   R0L,@PDR5           ;LEDは消灯する
```


2004-2005 Toyo-linx,Co.,Ltd.

```
        jsr     @wait1s

        bra     loop_01             ;loop_01へ無条件ブランチする


;---------------------- Sub-Routine -------------------------
wait1s:                             ;１秒待つウェート
        mov.l   #wait1000,ER6
wait1s_01:
        dec.l   #1,ER6              ;ER6を-1
        bne     wait1s_01           ;0でなければ，wait1s_01ブランチ
        rts
;------------------------------------------------------------
        .end
```

7－3.“samp_04”; ローテーション命令で LED を循環させる（電飾イメージの作成）

```
;------------------------------------------------------------|
;   FILE        :samp_04.src                                 |
;   DATE        :Thu, Jul 31, 2003                           |
;   DESCRIPTION :Main Program                                |
;   CPU TYPE    :H8/3687                                     |
;   This file is programed by toyolinx/tMogi                 |
;------------------------------------------------------------|
;   ●回転命令(ローテート)の演習                              |
;       PDR1,5を使って，電飾をイメージさせる。               |
;                                                            |
;       ①h'03--->R0L,h'00--->R0H                            |
;       ②R0Lのキャリー付き左回転                            |
;       ③R0Hのキャリー付き左回転                            |
;       ④R0L--->PDR1 ,  R0H--->PDR5                         |
;                                                            |
;                                                            |
;------------------------------------------------------------|
;   ●課題に挑戦！                                           |
;   課題１．パターンの変更                                   |
;   課題２．逆回転                                           |
;   課題３．全消灯，全点灯の間欠動作                         |
;   課題４．PDR7を追加                                       |
;   課題５．                                                 |
;                                                            |
;============================================================|
            .export     _main

;---------------------- Definition -------------------------
PDR1        .equ        h'ffd4
PCR1        .equ        PDR1+h'10
PDR5        .equ        h'ffd8
PCR5        .equ        PDR5+h'10
```

```
wait200     .equ        h'30000

;---------------------- Initialize --------------------------
_main:
            mov.b       #h'ff,R0L           ;R0Lを出力に設定
            mov.b       R0L,@PCR1
            mov.b       R0L,@PCR5

            mov.b       #b'00000011,R0L     ;ﾊﾞｲﾅﾘでR0Lにセット
            mov.b       #b'00000000,R0H     ;ﾊﾞｲﾅﾘでR0Hにセット
            andc        #b'11111110,CCR     ;ｷｬﾘｰを0にセット

;---------------------- Mian_Loop ---------------------------
loop_01:    rotxl.b     R0L                 ;R0L左回転
            rotxl.b     R0H                 ;R0H左回転

            mov.b       R0L,@PDR1           ;PDR1に出力
            mov.b       R0H,@PDR5           ;PDR5に出力

            jsr         @wait

            jmp         @loop_01
;---------------------- Sub_Routine --------------------------
wait:       mov.l       #wait200,ER6        ;0.2秒待つループ
wait_01:    dec.l       #1,ER6              ;ER6を－1
            bne         wait_01             ;0でなければ，loop_02分岐
            rts
;-------------------------------------------------------------
            .end
```

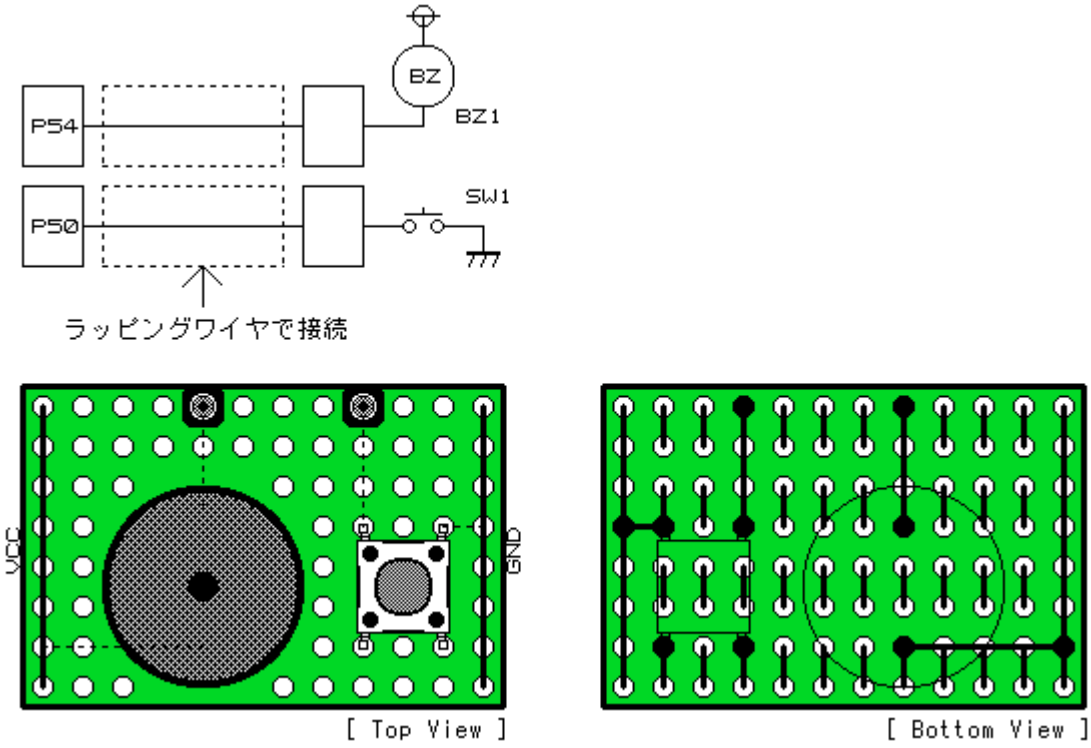

2004-2005 Toyo-linx,Co.,Ltd.

8-1-1 スイッチとブザー ＜ワンショット動作＞

まずはスイッチを押した時のみブザーを鳴らすワンショット動作をプログラムしてみましょう。ワンショット動作ですので押し続けてもブザーが鳴るのは押した瞬間のみです。

■ハードの組み立て

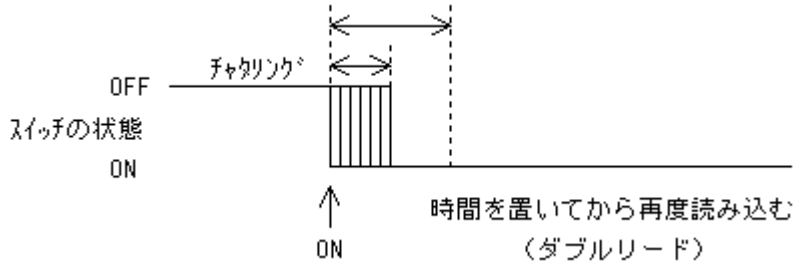
まずはブザーとスイッチを TK-3687 上のユニバーサルエリアに実装します。必要な工具はハンダごて、ハンダ、ニッパ、ワイヤストリッパです。下図 8-1-1 の回路図と実装図に従って実装して下さい。また、P5 のテストスルー（RA2 隣の 8 個のスルー）にも丸ピンソケットをハンダ付けします。ハンダ付けが終了したらラッピングワイヤでポートと接続して下さい。接続先はスイッチが P50、ブザーが P54 です。



＜図 8-1-1 回路図と実装図＞

■プログラム

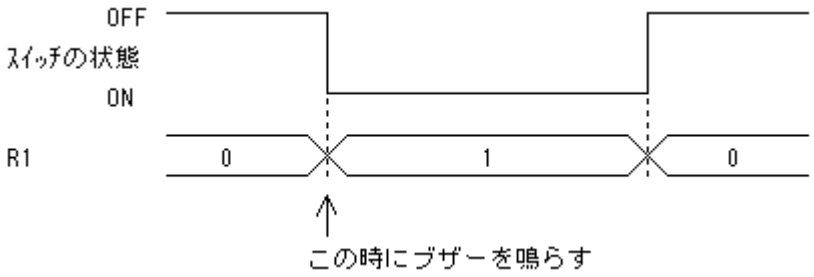
ハードが組みあがったら次にソフトを考えます。スイッチをポートで読む時にまず考えなくてはならないのはチャタリングの除去です。チャタリングとは機械式スイッチを押した時に接点がバウンドして ON/OFF が繰り返される現象です。単純にポートを読むだけですとバウンドの ON/OFF を読み取ってしまいます。チャタリングを回避する為に一定時間置いてから再度読む 2 度読み（ダブルリード）を行います。尚、スイッチは押されると GND に接続されるのでソフト読み込むとスイッチ on で Low となります。



＜図 8-1-2 チャタリングとダブルリード＞

2004-2005 Toyo-linx,Co.,Ltd.

次にワンショット動作を考えます。スイッチがONになった瞬間のみを検出する為にスイッチの状態をレジスタで保持しておきます。ここでは汎用レジスタ R1 をスイッチの状態レジスタとして使用する事にします。R1=0 ならスイッチは押されていない、R1=1 ならスイッチが押されている、と条件付けます。スイッチを押した瞬間を検出したいので R1 が 0 から 1 へ変化した時のみブザーを鳴らします。

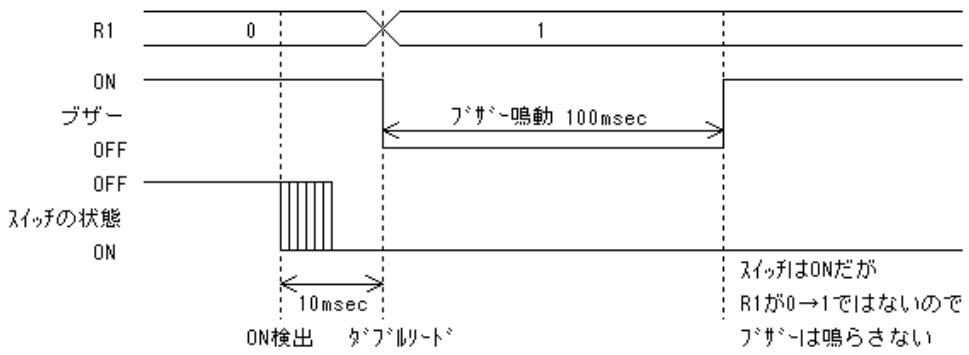


＜図 8-1-3 スイッチの状態とレジスタ R1＞

以上２点を考慮して作成したプログラムのリストとタイミングチャートを掲載します。ダブルリードの時間は 10msec としました。また、ブザーの鳴動時間は 100msec です。尚、ブザーは Low 出力で鳴動します。

```
PDR5  .equ h'FFD8               ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 5
PCR5  .equ h'FFE8               ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 5

      .export   _main

_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  --------------
      mov.b     #H'F0,r0l       ;PIO ｲﾆｼｬﾗｲｽﾞ
      mov.b     r0l,@PCR5       ; P50-53:in / P54-57:out

      bset      #4,@PDR5        ;ﾌﾞｻﾞｰ off
      xor.w     r1,r1           ;ｽｲｯﾁの状態 ｲﾆｼｬﾗｲｽﾞ

;-----  ﾒｲﾝﾙｰﾌﾟ  ---------------
_main_00:                       ;*** ｽｲｯﾁ 1st ﾘｰﾄﾞ ***
      btst      #0,@PDR5        ; ｽｲｯﾁ ﾘｰﾄﾞ
      beq       _main_02        ;  0=on / 1=off
      bra       _main_20

_main_02:                       ;*** ﾜﾝｼｮｯﾄ動作 ***
      mov.w     r1,e1           ; 今迄のｽｲｯﾁの状態をﾁｪｯｸ
      beq       _main_04        ;  1=on / 0=off
      bra       _main_00

_main_04:                       ;*** ｽｲｯﾁ 2nd ﾘｰﾄﾞ ***
      bsr       Wait_10m        ; ﾁｬﾀﾘﾝｸﾞ除去
      btst      #0,@PDR5        ; ｽｲｯﾁ ﾘｰﾄﾞ
      beq       _main_10        ;  0=on / 1=off
      bra       _main_02
```

```
_main_10:                       ;*** ｽｲｯﾁが押された ***
      bclr      #4,@PDR5        ; ﾌﾞｻﾞｰ on
      bsr       Wait_100m       ; 鳴動時間
      bset      #4,@PDR5        ; ﾌﾞｻﾞｰ off
      bset      #0,r1l          ; ｽｲｯﾁの状態ｾｯﾄ 1=on
      bra       _main_00

_main_20:                       ;*** ｽｲｯﾁは押されてない ***
      xor.w     r1,r1           ; ｽｲｯﾁの状態ｾｯﾄ 0=off
      bra       _main_00

;-----  10msec ｳｪｲﾄ  -----------
Wait_10m:
      mov.l     #H'8235,er6
Wait10m_00:
      dec.l     #1,er6
      bne       Wait10m_00
      rts

;-----  100msec ｳｪｲﾄ  ----------
Wait_100m:
      mov.l     #H'51612,er6
Wait100m_00:
      dec.l     #1,er6
      bne       Wait100m_00
      rts

      .end
```



＜図 8-1-4 タイミングチャート＞

2004-2005 Toyo-linx,Co.,Ltd.

8-1-2 スイッチとブザー ＜鈴虫の声を作ってみよう＞

前回のプログラムを応用して鈴虫の声を作ってみましょう。今までは“ピッ”という単音でしたが、Bz への ON/OFF を細かく連続して繰り返していくと“リリリ・・・”という鈴虫のような音色に変わります。不思議ですね。この音を出す為に Bz へ一定間隔の ON/OFF 制御（トグル出力）を追加します。尚、今回はワンショット動作は取り除き、Sw を押している間 Bz を鳴らすようにします。

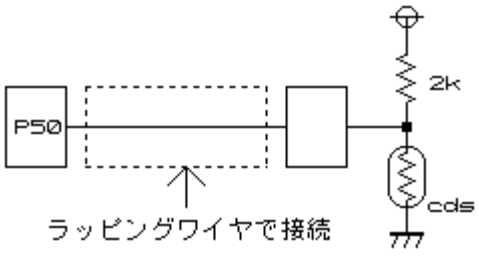
■プログラム

プログラムは前回とほぼ同じ流れです。違うのは押し続けを検出した時に Bz を鳴動させるようにしたのと、Bz への出力がトグル出力になった 2 点です。Bz の間欠間隔は Bz_CNST でカウント値を定義しています。間隔は 15msec 程度としました。これは色々試してみた結果です。音色を変えたい時は Bz_CNST の値を変えて下さい。

```
PDR5            .equ h'FFD8     ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 5
PCR5            .equ h'FFE8     ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 5

CHATR_CNST      .equ D'33330    ;ﾁｬﾀﾘﾝｸﾞ除去時間(10msec)
Bz_CNST         .equ D'49996    ;Bz 間欠時間(15msec)

          .export         _main
_main:
          mov.b     #H'F0,r0l  ;P50-53:入力
          mov.b     r0l,@PCR5  ;P54-57:出力

          bset      #4,@PDR5   ;Bz off
          xor.w     r1,r1      ;ﾌﾗｸﾞ ｸﾘｱ

loop_00:
          btst      #0,@PDR5   ;1st 入力ﾁｪｯｸ
          beq       loop_10    ;入力あれば押し続けのﾁｪｯｸへ

loop_02:                       ;入力無し
          bset      #4,@PDR5   ;Bz off
          xor.w     r1,r1      ;ﾌﾗｸﾞ ｸﾘｱ
          bra       loop_00

loop_10:
          mov.w     r1,e1      ;押し続けﾁｪｯｸ
          bne       loop_20    ;  押し続けなら Bz 鳴動

          bsr       WAITCHATR  ;ﾁｬﾀﾘﾝｸﾞ除去
          btst      #0,@PDR5   ;2nd 入力ﾁｪｯｸ
          beq       loop_20    ;入力があれば Bz 鳴動
          bra       loop_00

loop_20:                       ;Bz 鳴動
          bnot      #4,@PDR5   ;Bz ﾄｸﾞﾙ出力
          bsr       WAITBz     ;Bz 間欠時間
          mov.w     #1,r1      ;ﾌﾗｸﾞ ｾｯﾄ
          bra       loop_00

;-----  ﾁｬﾀﾘﾝｸﾞ除去の為の wait  ---------------------------
WAITCHATR:                     ;6
          mov.l     #CHATR_CNST,er6 ;6
          bra       WAIT_00

;-----  Bz 間欠動作時間の為の wait  -------------------------
WAITBz:                        ;6
          mov.l#Bz_CNST,er6    ;6

;-----  wait loop  ----------------------------------------
WAIT_00:
          dec.l     #1,er6     ;2
          bne       WAIT_00    ;4
          rts                  ;8

;=====================================================
          .end
```

■更なる改良点

Sw を光センサ（cds）に代えて、暗くなると Bz が鳴るようにしてみましょう。Sw を外して、右の回路図のように cds と抵抗をつなげます。cds は明るくなると抵抗値が下がり暗くなると上がります。従ってこの回路では暗くなるとマイコンに対しては HIGH 入力となるので、Sw の時とは逆に LOW で検出する必要があります。また、光はゆっくりと変化するのでチャタリング除去時間は大きめにした方が良いでしょう。修正箇所は次頁リストの三箇所になります。尚、コメントも併せて修正してあります。



＜図 8-1-5 cds の接続＞

2004-2005 Toyo-linx,Co.,Ltd.

```
PDR5            .equ h'FFD8     ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 5
PCR5            .equ h'FFE8     ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 5

CHATR_CNST      .equ D'333330   ;ﾁｬﾀﾘﾝｸﾞ除去時間(100msec)
Bz_CNST         .equ D'49996    ;Bz 間欠時間(15msec)

          .export       _main
_main:
          mov.b     #H'F0,r0l  ;P50-53:入力
          mov.b     r0l,@PCR5  ;P54-57:出力

          bset      #4,@PDR5   ;Bz off
          xor.w     r1,r1      ;ﾌﾗｸﾞ ｸﾘｱ

loop_00:
          btst      #0,@PDR5   ;1st 明るさﾁｪｯｸ
          bne       loop_10    ;  暗ければ鳴り続けのﾁｪｯｸへ

loop_02:                       ;明るい
          bset      #4,@PDR5   ;Bz off
          xor.w     r1,r1      ;ﾌﾗｸﾞ ｸﾘｱ
          bra       loop_00

loop_10:
          mov.w     r1,e1      ;鳴り続けﾁｪｯｸ
          bne       loop_20    ;  暗いままなら Bz 鳴動

          bsr       WAITCHATR  ;ﾁｬﾀﾘﾝｸﾞ除去
          btst      #0,@PDR5   ;2nd 明るさﾁｪｯｸ
          bne       loop_20    ;暗ければ Bz 鳴動
          bra       loop_00

loop_20:                       ;Bz 鳴動
          bnot      #4,@PDR5   ;Bz ﾄｸﾞﾙ出力
          bsr       WAITBz     ;Bz 間欠時間
          mov.w     #1,r1      ;ﾌﾗｸﾞ ｾｯﾄ
          bra       loop_00

;-----  ﾁｬﾀﾘﾝｸﾞ除去の為の wait  ----------------------------
WAITCHATR:                          ;6
          mov.l     #CHATR_CNST,er6 ;6
          bra       WAIT_00

;-----  Bz 間欠動作時間の為の wait  -------------------------
WAITBz:                             ;6
          mov.l#Bz_CNST,er6         ;6

;-----  wait loop  ----------------------------------------
WAIT_00:
          dec.l     #1,er6          ;2
          bne       WAIT_00         ;4
          rts                       ;8

;=========================================================
          .end
```

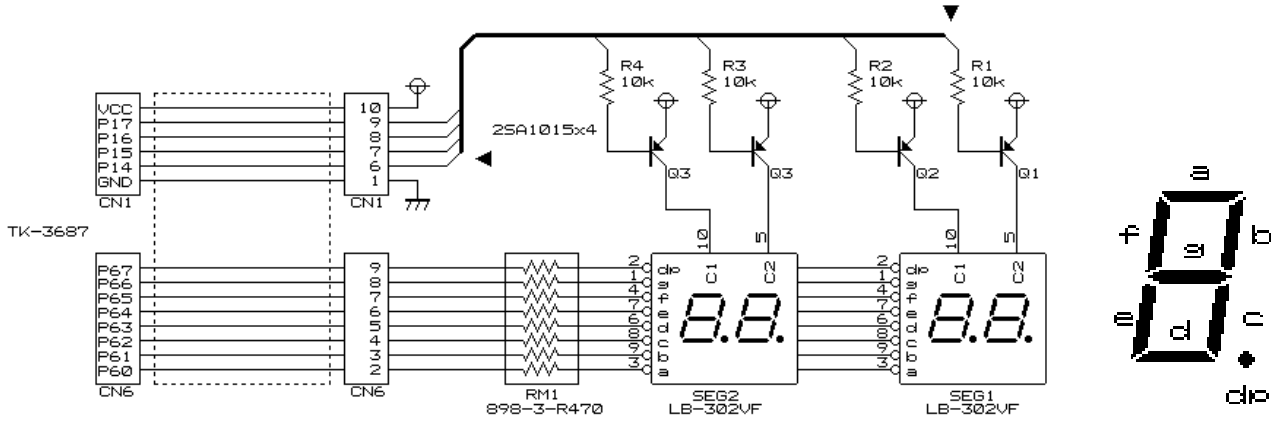
このプログラムでは暗くなると連続して“リリリ・・・”と鳴り続けるようになっています。しかし本物の鈴虫はずっと鳴きつづけている訳ではありません。例えばソフトで 1～2 秒間のお休みタイムを入れるなどすれば、より鈴虫の音色らしくなるでしょう。

2004-2005 Toyo-linx,Co.,Ltd.

8-2 ダイナミック表示

ここではダイナミック表示の応用プログラムを紹介します。部品は付属していませんので回路図を見て揃えてください。尚、オプションとして部品一式・ドキュメントをセットにした“**7セグメントLED表示&キー入力キット**:**B6086**”を用意しています。

図 8-2-1 はダイナミック表示の回路図とセグメントの割付けです。7 セグメント LED の各セグメントとアルファベットとの割付けは図のようになっています。この回路ではアノードコモンの LED を使っているのでコモン端子が“H”、a～g 及び dp の端子が“L”になった時対応するセグメントが点灯するようになっています。例えば“1”という文字を点灯させる時は b、c を“L”、コモン端子を“H”にします。



＜図 8-2-1 表示部回路図＞

P14～17 の信号を“L”にするとトランジスタが ON の状態になりコモン端子は“H”になります。その時 P60～67 の方は点灯させたいビットを“L” にするとLEDに電流が流れて点灯します。まずは、セグメントの下 1 桁に“4”を表示させるプログラムを考えてみましょう。＜Dscan_1＞

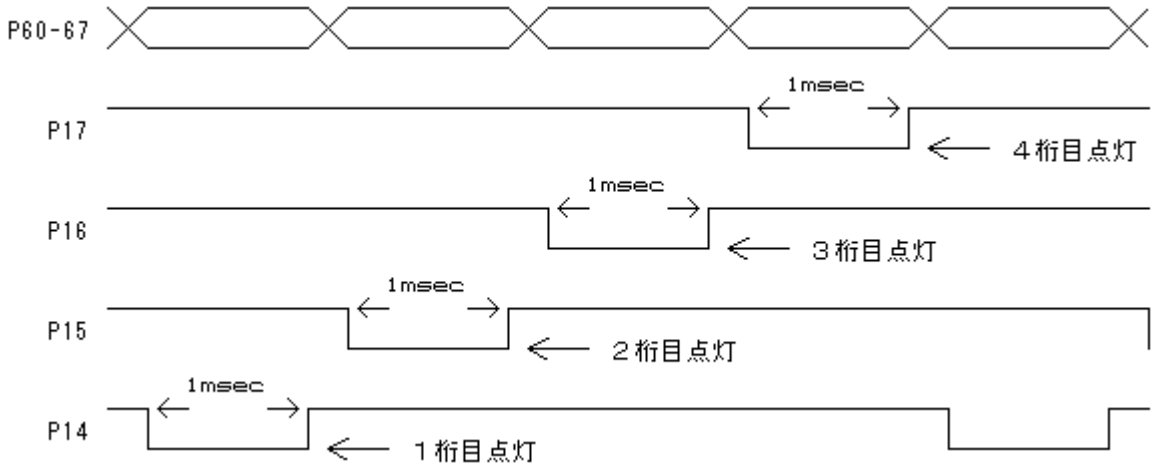
```
PDR1 .equ H'FFD4
PDR6 .equ H'FFD9
PCR1 .equ H'FFE4
PCR6 .equ H'FFE9

_main:
     bsr       INIPIO          ;PIO ｲﾆｼｬﾗｲｽﾞ
_main_00:
     mov.b     #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bra       _main_00

INIPIO:
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 6 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR6       ;ﾎﾟｰﾄ 60-67:out
     rts
```

＜Dscan_1＞

2004-2005 Toyo-linx,Co.,Ltd.

1 桁の表示ができたら次に 4 桁を時分割により表示させるプログラムを考えましょう。

ある一つの桁を表示している間は他の桁を消灯し、順に 4 桁繰り返し行います。タイミングチャートで表すと図 8-2-2 のようになります。



＜図 8-2-2 ダイナミック表示タイミングチャート＞

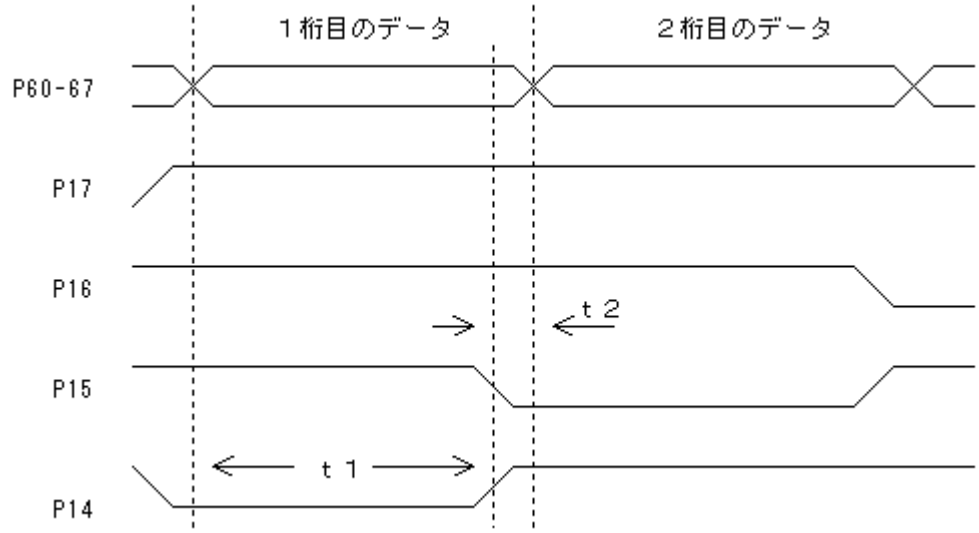
図 8-2-2 のタイミングチャートをもとに“1234”を表示するようなプログラムを作ってみます。＜Dscan_2＞

```
_main:
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
_main_00:
     mov.b     #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     mov.b     #H'DD,r1l       ;ﾎﾟｰﾄ 15:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01001111,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:3
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     mov.b     #H'BB,r1l       ;ﾎﾟｰﾄ 16:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01011011,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:2
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     mov.b     #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'00000110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:1
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     bra       _main_00

INIPIO:
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 6 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR6       ;ﾎﾟｰﾄ 60-67:out
     rts

TIMER1m:
     mov.l     #H'D02,er6
TIMER1m_00:
     dec.l     #1,er6
     bne       TIMER1m_00
     rts
```

＜Dscan_2＞

2004-2005 Toyo-linx,Co.,Ltd.

タイミングチャートのようにプログラムを作りました。一見目的通り点灯しているように見えますが、スキップ実行でプログラムをトレースしてみて下さい。一つ前の桁の数が表示されてからその桁の数が表示されているのが分かると思います。何故このように一つ前の桁の数を表示してしまうかというと、P14～17 のスキャンを切り替える際、桁は切り替わっているのに P6 の表示データは切り替わっていない為です。

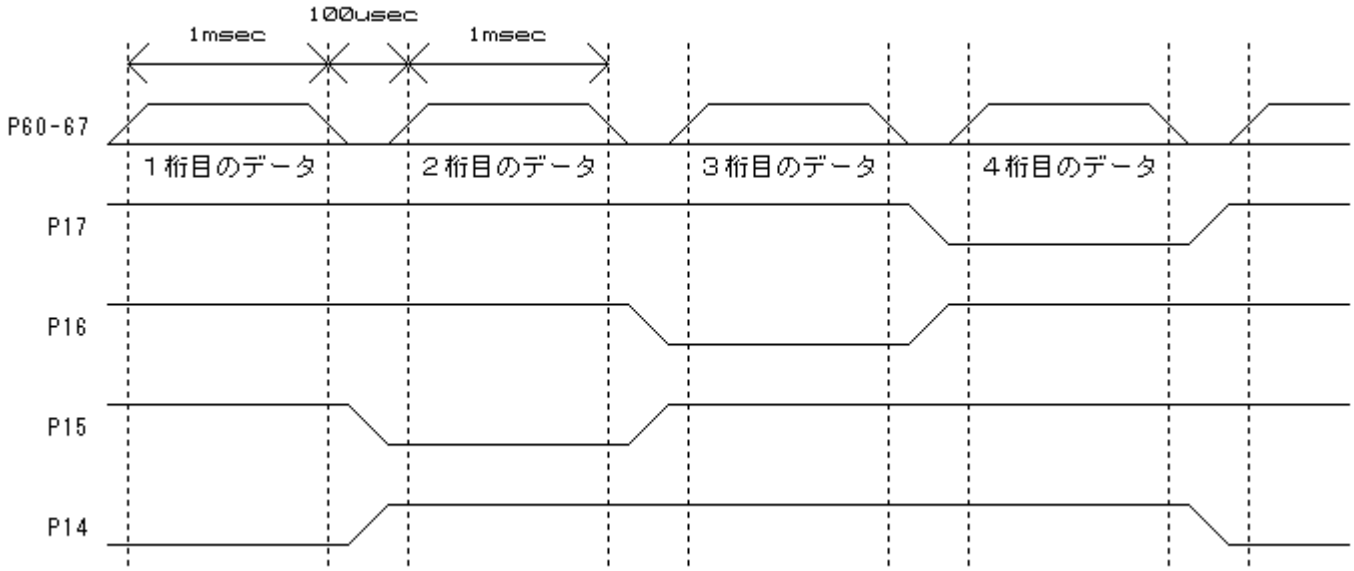


＜図 8-2-3　Dscan_2 のタイミングチャート＞

その様子を示したのが図 8-2-3 です。細かく見ていくとまず P14 が“L”の時 P6 のデータを表示します（t1）。次にスキャンを移動し P15を“L”にすると、P15のためのデータに変更するまでの時間、P14のデータを表示してしまうわけです（t2）。このことを解決する為に次の 2 つのことを考えてプログラムを作り直しました。＜Dscan_3＞

1. 桁スキャン P14～17 を変更する前に P6 のデータを“L”にして LED を一旦消灯する。
2. 1 の状態を安定させる為にタイマーを入れる。タイマーの時間は点灯している時間に対して十分に短くし、見かけ上 LED が消灯していることが分からない位の 100μsecとする。

図 8-2-4 はそのタイミングチャートです。



＜図 8-2-4　Dscan_3 のタイミングチャート＞

2004-2005 Toyo-linx,Co.,Ltd.

```
_main:
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
_main_00:
     mov.b     #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER100u:16

     mov.b     #H'DD,r1l       ;ﾎﾟｰﾄ 15:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01001111,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:3
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER100u:16

     mov.b     #H'BB,r1l       ;ﾎﾟｰﾄ 16:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01011011,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:2
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER100u:16

     mov.b     #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'00000110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:1
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     not       r0l             ;反転
     mov.b     r0l,@PDR6       ;出力
     bsr       TIMER100u:16

     bra       _main_00

INIPIO:
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 6 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR6       ;ﾎﾟｰﾄ 60-67:out
     rts

TIMER100u:
     mov.l     #H'14A,er6
TIMER100u_00:
     dec.l     #1,er6
     bne       TIMER100u_00
     rts

TIMER1m:
     mov.l     #H'D02,er6
TIMER1m_00:
     dec.l     #1,er6
     bne       TIMER1m_00
     rts
```

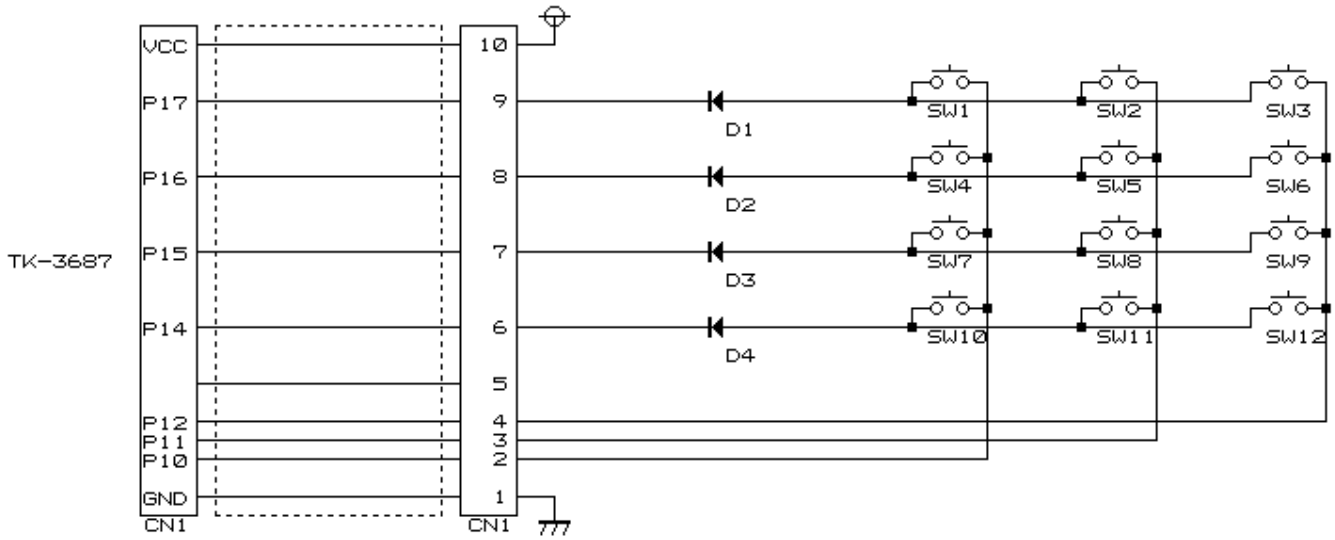
<Dscan_3>

2004-2005 Toyo-linx,Co.,Ltd.

8-3 マトリックスキー

ここではマトリックスキーの読み込みプログラムを紹介します。入力の結果を“8-2ダイナミック表示”で製作した7セグメントに表示しますのでまだ作成されていない方は先に表示部を作ってください。8-2同様部品は付属していませんので回路図を見て揃えてください。尚、オプションとして部品一式・ドキュメントをセットにした“**7セグメントLED表示&キー入力キット:B6086**”を用意しています。

図 8-3-1 はマトリックスキーのみ抜き出した回路図です。P14～17 がスキャンライン、P10～12 が読み込む為の入力ラインです。キーを読み込むには読み込みたいキーのスキャンラインを Low にしてポートを読みます。TK-3687 でポートはプルアップされているのでキーが押されたビットは Low、押されていなければ Hi となります。



＜図 8-3-1 マトリックスキー回路図＞

まずはスキャンを行わずにキーを読んでみましょう。スキャンライン P17 のみを Low にして SW1～3 のいずれかが押されたらセグメントに表示します。スキャンラインは表示と共通なのでセグメントの表示は最上桁になります。

```
_main:
     bsr       INIPIO          ;PIO ｲﾆｼｬﾗｲｽﾞ
_main_00:
     mov.b     #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力

     mov.b     @PDR1,r0l       ;ｷｰ入力
     not       r0l             ;負論理なので反転
     and.b     #H'07,r0l       ;ｷｰの有効 bit は 3bit
     bne       _main_02        ;eq=押されていない
     mov.b     #H'FF,r0l       ;ｾｸﾞﾒﾝﾄ消灯
     mov.b     r0l,@PDR6       ;表示
     bra       _main_00
```

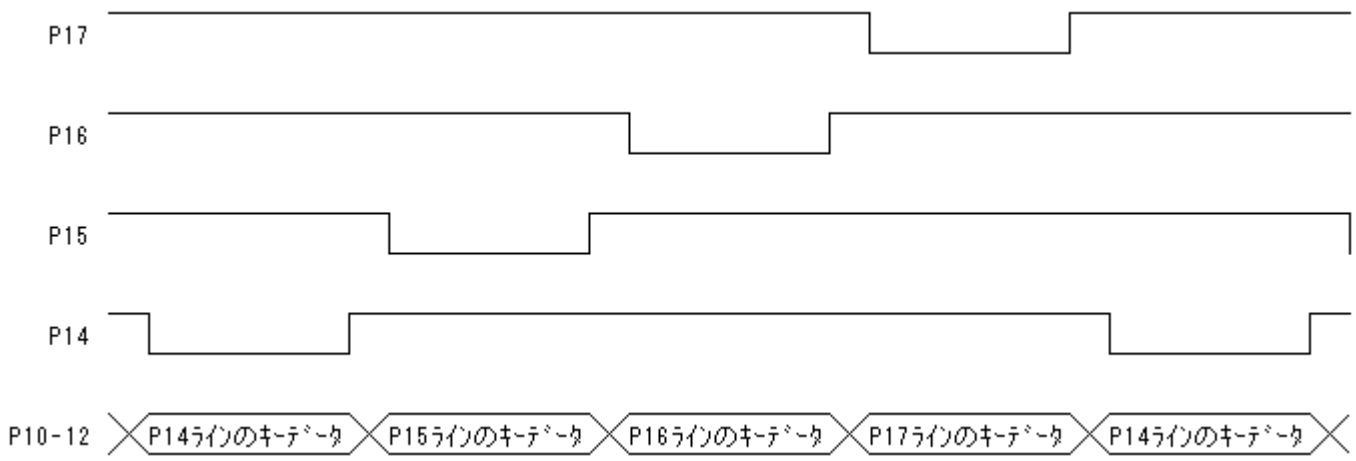
```
_main_02:
     mov.b     #B'01011100,r0l ;ｾｸﾞﾒﾝﾄ表示
     not       r0l             ;負論理なので反転
     mov.b     r0l,@PDR6       ;表示
     bra       _main_00

INIPIO:
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 6 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR6       ;ﾎﾟｰﾄ 60-67:out
     rts
```

＜matrixkey_1＞

2004-2005 Toyo-linx,Co.,Ltd.

1 ラインの読み込みができたら次は各ラインをスキャンしてキーを読み込みましょう。

1 つのスキャンラインを Low にしてキーを読み込み、1 ライン読み込んだら次のスキャンラインを Low に、これをスキャンライン分繰り返します。スキャンでのキー読み込みタイミングチャートを図 8-3-2 に示します。



＜図 8-3-2 スキャンでのキー読み込みタイミングチャート＞

上記のタイミングを元に各キーを読み込むプログラムを作ってみましょう。それぞれのキー番号をセグメントの最下桁に表示させます。マトリックスキーの読み込みを分かり易くするためにダイナミック表示は行わずに表示させます。レジスタ R1L がスキャンデータ、R1H はループの回数、ここではスキャンラインが 4 本あるので 4 回ループさせます。そして R2 にはキー番号をセットします。キー番号は 1 から始まってスキャンラインが移動するたびに+3していきます。キーは番号の若い方が優先順位が高く、幾つかのキーが押された場合は優先順位の高い方を表示します。

```
_main:
      bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
      mov.b     #H'00,r3l       ;R3L=ｾｸﾞﾒﾝﾄﾃﾞｰﾀ
_main_00:
      mov.b     #H'FF,r0l       ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
      mov.b     r0l,@PDR6       ;出力
      mov.b     #H'77,r1l       ;R1L=ｽｷｬﾝﾃﾞｰﾀ
      mov.b     #D'4,r1h        ;R1H=ﾙｰﾌﾟ回数
      mov.w     #D'1,r2         ;R2=ｷｰ番号
_main_02:
      mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
      mov.b     @PDR1,r0l       ;ｷｰ読み込み
      not       r0l             ;反転
      and.b     #H'07,r0l       ;ｷｰは押されていた?
      bne       _main_04        ;ne = 押されていた
      add.w     #D'3,r2         ;ｷｰ番号 +3
      rotr.b    r1l             ;次のｽｷｬﾝへ
      dec.b     r1h             ;ﾙｰﾌﾟ回数 -1
      bne       _main_02
      bra       _main_10        ;ﾙｰﾌﾟ終了
_main_04:                       ;ｷｰ検出
      btst      #0,r0l          ;bit0 ﾁｪｯｸ
      bne       _main_08
      btst      #1,r0l          ;bit1 ﾁｪｯｸ
      beq       _main_06
      inc.w     #1,r2           ;ｷｰ番号 +1
      bra       _main_08
_main_06:
      inc.w     #2,r2           ;ｷｰ番号 +2
_main_08:
      xor.w     e2,e2
      mov.b     @(SEG_TBL,er2),r3l   ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ ｹﾞｯﾄ
_main_10:                       ;表示ﾌﾟﾛｸﾞﾗﾑ
      mov.b     #H'EE,r0l
      mov.b     r0l,@PDR1       ;表示用ｽｷｬﾝ出力
      mov.b     r3l,r0l
      not       r0l
      mov.b     r0l,@PDR6       ;表示ﾃﾞｰﾀ 出力
      bsr       TIMER1m         ;表示時間 = 1msec
      mov.b     #H'FF,r0l       ;ｽｷｬﾝ off
      mov.b     r0l,@PDR1
      bra       _main_00

INIPIO:
      mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ1 ｲﾆｼｬﾗｲｽﾞ
      mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ14-17:out
      mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ6 ｲﾆｼｬﾗｲｽﾞ
      mov.b     r0l,@PCR6       ;ﾎﾟｰﾄ60-67:out
      rts

TIMER1m:
      mov.l     #H'D02,er6
?000:
      dec.l     #1,er6
      bne       ?000
      rts

SEG_TBL:                        ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ ﾃｰﾌﾞﾙ
 .data.b    H'3F,H'06,H'5B,H'4F,H'66   ; 0 1 2 3 4
 .data.b    H'6D,H'7D,H'07,H'7F,H'67   ; 5 6 7 8 9
 .data.b    H'77,H'7C,H'39,H'5E,H'79   ; A b C d E
 .data.b    H'71                       ; F
```

＜matrixkey_2＞

2004-2005 Toyo-linx,Co.,Ltd.

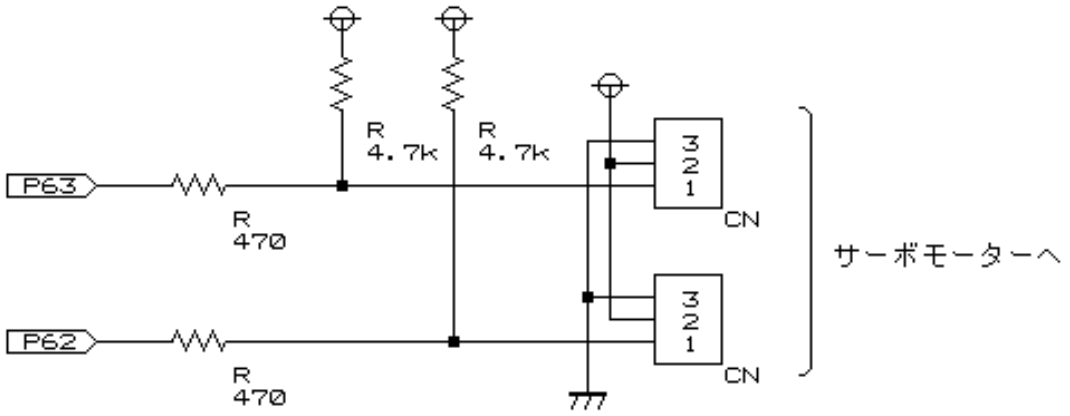
サーボモータは角度に相当する幅のパルスを数 10msec 毎に出力する事で、任意の角度へ移動・保持する事ができます。サーボの角度を示すパルス幅はメーカによって多少差はありますが、大体 700～2300μsec 程度です。中点を保持したいならば 1500μsec のパルスを数 10msec 毎に出力します。

タイマ Z にはタイマカウンタ(TCNT)とジェネラルレジスタ(GR)とを比較しポートを制御する"コンペアマッチによる波形出力機能"があります。ジェネラルレジスタは 1 チャンネルに 4 本(タイマ Z は 2 チャンネルあるので計 8 本)ありそれぞれ独立して比較する事が可能です。

ここでは、"コンペアマッチによる波形出力機能"を利用してサーボモータをコントロールします。パルス幅をタイマ Z のコンペアマッチで作り、出力間隔をタイマ B1 で管理します。尚、出力間隔は 10msec とします。また、一定角度を保持するだけでは面白くないので、ジェネラルレジスタにセットする値をテーブル化し、0.5 秒毎にジェネラルレジスタの値を更新して一連の動作をさせることにします。更新間隔の 0.5 秒もタイマ B1 で管理します。

■ ハード

タイマ Z の出力はポート 6 との兼用です。使用するポートはソフトの項目で記しますが P62・P63 を使用します。サーボモータとのインターフェースは次のようになります。



【 サーボモータとの接続インターフェース 】

挿入されている抵抗 470Ω は位置情報出力機能(ポジションキャプチャ)を有しているサーボモータ(近藤化学社製サーボモータなど)の出力信号からポートを保護する為のものです。尚、サーボモータと TK-3687 を接続する部品キットをご用意していますので詳しくは弊社までお問い合わせ下さい。

■ ソフト

①タイマ B1

タイマ B1 を約 1msec でカウントアップするように設定し、そのオーバーフローを割り込みで検出、各タイミングに利用します。そこでタイマ B1 を"オートリロードタイマ"で動作させ、クロックセレクトを '内部クロックφ/512' とします。タイマロードレジスタ(TLB1)にセットする値は、

1msec÷('内部クロックφ/512' ＝25.6μsec)≒39

アップカウンタなので、

256－39＝217

となります。

割り込み毎にカウントを行ない、10 カウント(10msec)したらタイマ Z の出力、500 カウント(0.5sec)したらジェネラルレジスタ更新を行ないます。ジェネラルレジスタの更新は、テーブルをサーチしてジェネラルレジスタを書き替

2004-2005 Toyo-linx,Co.,Ltd.

えます。尚、直接 10msec を作らずに 1msec にしているのは、プログラムを追加するなどした場合、他で流用する事を考えての事です。

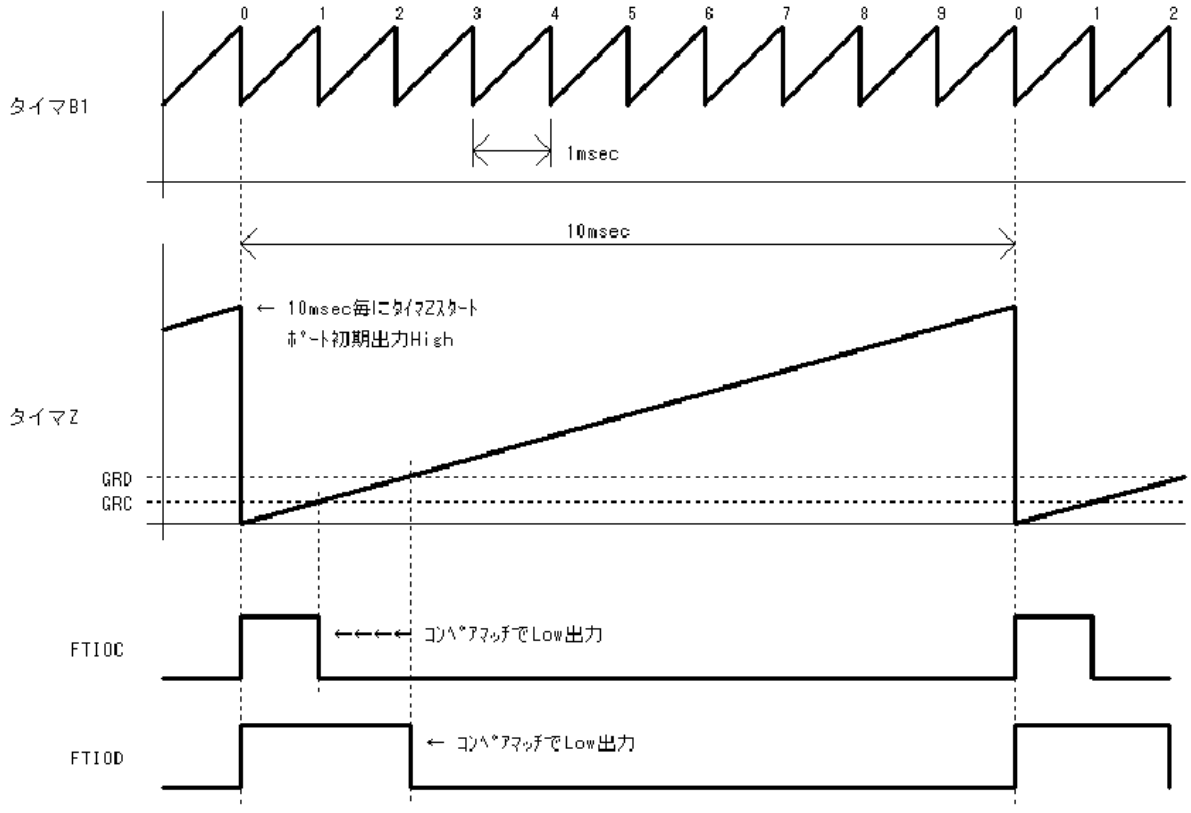
②タイマ Z

タイマ B1 の 10msec タイミングでタイマ Z をスタートさせます。ここで使用するチャンネルは ch0、ジェネラルレジスタは GRC_0・GRD_0 とします。冒頭で述べたようにタイマ 2 チャンネルと全てのジェネラルレジスタを使用すれば 8bit 制御する事ができますが、まずは 2bit で制御を行ないましょう。

カウントはフリーランニングカウントで動作させます。10msec でリスタートがかかるのでタイマプリスケーラは‘内部クロック: φ/4 でカウント’にします。これならオーバーフローするまでに約 13.1msec かかるのでそれまでにはリスタートされます。タイマ Z の動作は、コンペアマッチが発生するまでは High を出力しておくようにタイマアウトプットコントロールレジスタ(TOCR)をセット、コンペアマッチで Low を出力するようにタイマ I/O コントロールレジスタ(TIORC_0)をセットし、ジェネラルレジスタ GRC_0、GRD_0 に値を設定してカウント動作を開始させます。
例えば 1500μsec のパルスを得たいのならば、ジェネラルレジスタにセットする値は、

1500μsec÷(φ/4＝200nsec)＝7500

をセットします。ジェネラルレジスタにセットするカウント値をテーブル化するわけですが、セットする値そのままでは実際のパルス幅が直感的に分かりません。そこでテーブルにはμsec 単位で入力し、ジェネラルレジスタにセットする前にプログラムでカウント値に直すようにしました。1μsec＝5 カウント(1μsec÷(φ/4＝200nsec)＝5)なのでテーブルデータを 5 倍してジェネラルレジスタにセットします。以下にタイマ Z、タイマ B1 と出力の関係を示します。



【 タイマ Z・タイマ B1 と出力の関係 】

以上を踏まえて作成したプログラムリストを次頁に示します。

2004-2005 Toyo-linx,Co.,Ltd.

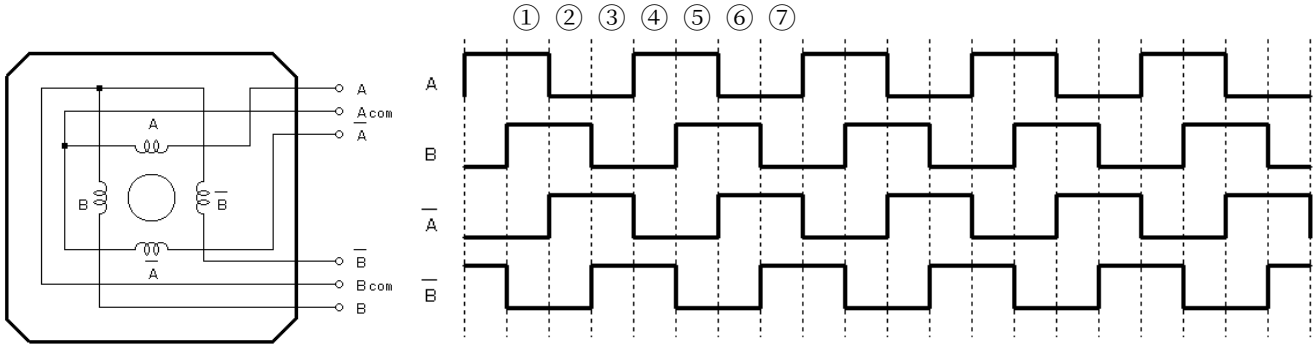
```
;---------------------------------------------------------
;
;   FILE        :TK3687servo_00.src
;   DATE        :Thu, Nov 25, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;
;---------------------------------------------------------

     .include  "io3687F_equ.inc"

     .export   _main
     .export   INTRTB1
     .export   INTRTZ0

;***********************************************************
;    定数
;***********************************************************
;-----  ｼｰｹﾝｽ関連  ----------------------------------------
SEQMAX_CNST     .equ D'64 ;ｼｰｹﾝｽﾃﾞｰﾀ数
SEQTMNG_CNST    .equ H'0500     ;ｼｰｹﾝｽ更新ﾀｲﾐﾝｸﾞ(BCD ﾃﾞｰﾀ･msec
指定)

;***********************************************************
;    メインプログラム
;***********************************************************
;※ﾘｽﾄにｱﾄﾞﾚｽを表したい時は locate を有効にする
     .section  P,code,locate=H'EA00

_main:
;----- イニシャライズ -------------------------------------
     bsr   INITTB1:16      ;TimerB1 ｲﾆｼｬﾗｲｽﾞ
     bsr   INITTZ0_0C:16         ;TimerZ0 ｺﾝﾍﾟｱﾏｯﾁ
     bsr   INITINTR:16           ;割り込み ｲﾆｼｬﾗｲｽﾞ

     mov.w#WORK_AREA,r3          ;ﾜｰｸｴﾘｱ ｸﾘｱ
     mov.w#WORK_AREAE-WORK_AREA-1,r1
     xor.br0l,r0l
     jsr  @FILL

     mov.w@SEQDT_TBL0,r0
     mov.wr0,@FTIOD0_CNST
     mov.w@SEQDT_TBL1,r0
     mov.wr0,@FTIOC0_CNST

     andc #H'7F,CCR        ;割り込み許可

;----- メインループ ---------------------------------------
LOOP_00:
     mov.b@TB1_FG,r0l            ;TimerB1 割り込み検出？
     bne  LOOP_10:16
     bra  LOOP_00

;----- TimerB1 1ms割り込み検出 ------------
LOOP_10:
     xor.br0l,r0l
     mov.br0l,@TB1_FG

     mov.w@TB1_CNT,r0            ;TB1_CNT を BCD でｲﾝｸﾘﾒﾝﾄ
     bsr  BCDINCW:16
     mov.wr0,@TB1_CNT

     cmp.w#SEQTMNG_CNST,r0       ;ｼｰｹﾝｽ更新ﾀｲﾐﾝｸﾞ
     bcc  LOOP_13

LOOP_11:
     and.b#H'0F,r0l        ;10msec 経過ならｻｰﾎﾞ出力へ
     beq  LOOP_14
     bra  LOOP_16

LOOP_13:                   ;ｼｰｹﾝｽの変更
     xor.wr0,r0            ;ｶｳﾝﾀ ｸﾘｱ
     mov.wr0,@TB1_CNT

     bsr  SEQCNTINC:16           ;ｼｰｹﾝｽﾎﾟｲﾝﾀ ｲﾝｸﾘﾒﾝﾄ
     shll.w    r0                ;ｼｰｹﾝｽﾃﾞｰﾀﾃｰﾌﾞﾙからﾃﾞｰﾀ取得
     xor.we0,e0
     mov.ler0,er1
     mov.w@(SEQDT_TBL0,er1),r0
     mov.wr0,@FTIOD0_CNST
     mov.w@(SEQDT_TBL1,er1),r0
     mov.wr0,@FTIOC0_CNST

LOOP_14:
     bsr  INITTZ0_0C:16

     mov.w@FTIOD0_CNST,r0 ;ｼﾞｪﾈﾗﾙﾚｼﾞｽﾀ:GRD_0 設定
     xor.we0,e0
     mov.w#5,r1
     mulxu.w   r1,er0            ;x5 TZ0:CLK/4 1usec=5count
     mov.wr0,@GRD_0

     mov.w@FTIOC0_CNST,r0 ;ｼﾞｪﾈﾗﾙﾚｼﾞｽﾀ:GRC_0 設定
     xor.we0,e0
     mov.w#5,r1
     mulxu.w   r1,er0            ;x5 TZ0:CLK/4 1usec=5count
     mov.wr0,@GRC_0

     mov.b@TOCR,r0l        ;FTIOD0 High 出力
     or.b #H'0C,r0l
     mov.b@TSTR,r0h        ;TZ:ch0 ｽﾀｰﾄ
     bset #0,r0h
     mov.br0l,@TOCR
     mov.br0h,@TSTR

     mov.b#B'11111000,r0l
     mov.br0l,@TIER_0

LOOP_16:
     bra  LOOP_00

;***********************************************************
;    割り込み処理
;***********************************************************
;---------------------------------------------------------
;    タイマＢ１割り込み（１msec 割り込み）
;---------------------------------------------------------
INTRTB1:
     push.l    er0

     mov.b#B'00011111,r0l ;割り込みﾌﾗｸﾞﾚｼﾞｽﾀ 2 TB1 ｸﾘｱ
     mov.br0l,@IRR2

     mov.b#H'01,r0l        ;割り込みﾌﾗｸﾞ ｾｯﾄ
     mov.br0l,@TB1_FG

     pop.ler0
     rte

;---------------------------------------------------------
;    タイマＺ０割り込み
;---------------------------------------------------------
INTRTZ0:
     push.l    er0

     mov.b@TSR_0,R0L ;IMFD ｸﾘｱ
     mov.b#B'11100111,r0l
     mov.bR0L,@TSR_0

     pop.ler0
     rte

;***********************************************************
;    サブルーチン
;***********************************************************
;---------------------------------------------------------
```


2004-2005 Toyo-linx,Co.,Ltd.

```
;          データで埋める
;---------------------------------------------------------
;R3=ｱﾄﾞﾚｽ / R1=ﾃﾞｰﾀ数 / R0L=ﾃﾞｰﾀ
FILL:
     mov.br0l,@r3    ;ｱﾄﾞﾚｽ R3 から R1 だけ R0L で埋める
     inc.w#1,r3
     dec.w#1,r1
     bne  FILL
     rts

;---------------------------------------------------------
;          シーケンス番号＋１
;---------------------------------------------------------
SEQCNTINC:
     mov.w@SEQ_CNT,r0
     inc.w#1,r0
     cmp.w#SEQMAX_CNST,r0
     bcs  SEQCNTINC_00
     xor.wr0,r0
SEQCNTINC_00:
     mov.wr0,@SEQ_CNT
     rts

;---------------------------------------------------------
;          ＢＣＤデータ＋１（0000～9999）
;---------------------------------------------------------
BCDINCW:
     add.b#1,r0l
     daa  r0l
     bcc  BCDINCW_00
     add.b#1,r0h
     daa  r0h
BCDINCW_00:
     rts

;---------------------------------------------------------
;    割り込みイニシャライズ
;---------------------------------------------------------
INITINTR:
     mov.b#B'00111111,r0l ;TimerB1 割り込み許可
     mov.br0l,@IENR2
     rts

;---------------------------------------------------------
;    タイマＺ０イニシャライズ : GRD_0 ｱｳﾄﾌﾟｯﾄ ｺﾝﾍﾟｱﾏｯﾁ
;---------------------------------------------------------
INITTZ0_0C:
     mov.b@TSTR,r0l       ;TZ0 Stop
     bclr #0,r0l
     mov.br0l,@TSTR

     xor.wr0,r0           ;TCNT_0 ｸﾘｱ
     mov.wr0,@TCNT_0
     mov.b#B'00000010,r0l ;TCNT ｸﾘｱ:禁止 CLK*4
     mov.br0l,@TCR_0
     mov.b#B'10011001,r0l ;GRD_0,GRC_0 ｺﾝﾍﾟｱﾏｯﾁ Low 出力
     mov.br0l,@TIORC_0
     mov.b@TOCR,r0l       ;FTIOD0,C0 Low
     and.b#H'F3,r0l
     mov.br0l,@TOCR
     mov.b@TOER,r0l       ;FTIOD0,C0 出力許可
     and.b#H'F3,r0l
     mov.br0l,@TOER
     rts

;---------------------------------------------------------
;    タイマＢ１イニシャライズ
;---------------------------------------------------------
INITTB1:
     mov.b#B'11111010,r0l ;ｵｰﾄﾘﾛｰﾄﾞ,CLK/512
     mov.br0l,@TMB1
     mov.b#D'255-39,r0l        ;ﾘﾛｰﾄﾞ値 CLK/512x39=1msec
     mov.br0l,@TCB1
     rts

;*********************************************************
;          シーケンスデータテーブル・エリア
;*********************************************************
;    0640=-90°: 1060=-45°: 1480=0°: 1900=+45°: 2320=+90°
;    0550=-MIN°                              : 2550=+MAX°
SEQDT_TBL0:
;手を上下にぶらぶら
     .data.w   D'0640,D'0640,D'2550,D'2550,D'2550,D'0640
     .data.w   D'0640,D'0640,D'2550,D'2550,D'2550
;ガッツポーズ
     .data.w   D'1270,D'1270,D'1690,D'1690,D'1270
     .data.w   D'1270,D'1690,D'1690,D'1270,D'1270
;手を上下にぶらぶら
     .data.w   D'0640,D'0640,D'2550,D'2550,D'2550,D'0640
     .data.w   D'0640,D'0640,D'2550,D'2550,D'2550
;ガッツポーズ
     .data.w   D'1270,D'1270,D'1690,D'1690,D'1270
     .data.w   D'1270,D'1690,D'1690,D'1270,D'1270
;深呼吸
     .data.w   D'0640,D'0640,D'0640,D'1480,D'1480,D'1480
     .data.w   D'1480,D'0640,D'0640
;深呼吸
     .data.w   D'0640,D'0640,D'0640,D'1480,D'1480
     .data.w   D'1480,D'1480,D'0550,D'0640,D'0640
     .data.w   D'0640,D'0640,D'0640

SEQDT_TBL1:
;手を上下にぶらぶら
     .data.w   D'1480,D'1480,D'1480,D'1900,D'1900,D'1900
     .data.w   D'1060,D'1060,D'1060,D'1900,D'1900
;ガッツポーズ
     .data.w   D'2500,D'2500,D'2100,D'2100,D'2500
     .data.w   D'2500,D'2100,D'2100,D'2500,D'2500
;手を上下にぶらぶら
     .data.w   D'1480,D'1480,D'1480
     .data.w   D'1900,D'1900,D'1900,D'1060,D'1060,D'1060
     .data.w   D'1900,D'1900
;ガッツポーズ
     .data.w   D'2500,D'2500,D'2100,D'2100,D'2500,D'2500
     .data.w   D'2100,D'2100,D'2500,D'2500
;深呼吸
     .data.w   D'1060,D'1060,D'1060,D'1480,D'1480,D'1480
     .data.w   D'1480,D'1060,D'1060
;深呼吸
     .data.w   D'1060,D'1060,D'1060,D'1480,D'1480,D'1480
     .data.w   D'1480,D'1060,D'1060,D'1060,D'1480,D'1480
     .data.w   D'1480

;*********************************************************
;          データ・エリア
;*********************************************************
;※ﾘｽﾄにｱﾄﾞﾚｽを表したい時は locate を有効にする
     .section  D,data,locate=H'F780

WORK_AREA:
;----- 動作ﾓｰﾄﾞ ----------------
SEQ_CNT:  .res.w    1          ;ｼｰｹﾝｽ番号
;----- ﾀｲﾏ B1 -------------------
TB1_FG:   .res.b    1          ;TimerB1 割り込みﾌﾗｸﾞ
     .align    2
TB1_CNT:  .res.w    1          ;ﾊﾟﾙｽ出力間隔 ｶｳﾝﾀ
;----- ﾀｲﾏ Z0 -------------------
GRD0_CNT: .res.w    1
FTIOC0_CNST:   .res.w    1          ;FTIOC0 出力時間
FTIOD0_CNST:   .res.w    1          ;FTIOD0 出力時間

WORK_AREAE:
;=========================================================
          .end
```


2004-2005 Toyo-linx,Co.,Ltd.

パルス(ステッピング)モータは、各相のコイルにパルスを順次切り替え励磁することでモータを回転させることができます。励磁には幾つか方式がありますが、ここでは『2 相励磁』方式で駆動します。以下にユニポーラ駆動での 2 相励磁方式のパルスを示します。



【 2 相励磁 】

2 相励磁方式は、名前の通り常に 2 つの相が励磁される方式です。1 相づつ励磁する『1 相励磁』に比べ力(トルク)が強くまた振動しにくい為、一般的に使われます。マイコンでパルスモータを回転させるには、上記波形のようにパルスを順次切り替えて出力していきます。出力データは、

①:1100b(Ch)→②:0110b(6h)→③:0011b(3h)→④1001b(9h)→⑤:1100b(Ch)※①と同じ→以下繰り返し

つまりデータをロテートして出力していけばモータを回転させることができるわけです。出力は 4bit で足りますがプログラム中のデータは 1byte としておきます。つまり、

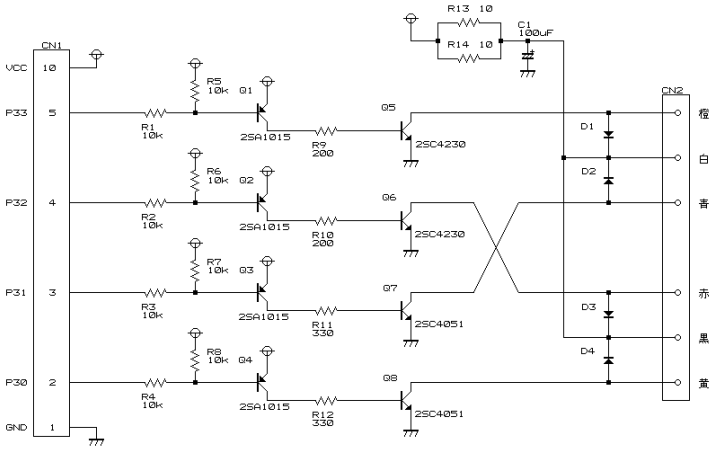
11001100b(CCh)→01100110b(66h)→00110011b(33h)→10011001b(99h)→11001100b(CCh)・・・

と、1byte データなら単純にロテートすればよい訳です。ちなみにロテートの方向を逆にすれば逆回転となります。

さて、ただ一方行に定速で回転させるだけなら上記のパターンを一定間隔で出力するだけでよいのですが、今回は回転パターンをテーブルデータとして持たせ、回転方向、回転速度、角度を自由にコントロールできるようにします。

■ ハード

パルスモータに多くの電流を流す為、ハードは次のようにしました。



【 パルスモータとの接続インターフェース 】

尚、TK-3687 とパルスモータを接続するキットをご用意しています。詳しくは弊社までお問い合わせ下さい。

■ ソフト

プログラムの流れとしてはタイマでパルスの出力間隔を作り、出力タイミングになったら回転データテーブルから回

2004-2005 Toyo-linx,Co.,Ltd.

転方向、回転速度(パルスの出力間隔)、角度(ステップ数)をそれぞれセットします。

タイマ V

タイマ V でパルスの出力間隔を作ります。タイマのカウントだけでは間隔が早過ぎるので、タイマで一定間隔の割り込みを作り、その割り込み処理内で出力間隔をカウントするようにします。前述した回転パターンの回転速度には、このパルス出力間隔がセットされます。割り込み間隔は 100μsec としました。100μsec 毎に割り込みをかけるにはタイマ V を次のようにセットします。

TCRV0＝01001010b(内部クロックφ／16・コンペアマッチ A でカウントクリア＆割り込み)

TCRV1＝11100010b(内部クロックφ／16・外部カウント禁止)

TCORA＝125d(φ／16×125＝100μsec)

回転データテーブル

回転データテーブルは、回転方向・回転速度(パルスの出力間隔)・角度(ステップ数)の 3 つのデータを 1 単位としてテーブル化します。各データは次の通りです。

	内容	セットする値
回転方向	回転する方向、もしくは停止を指定	停止＝0／左回転＝1／右回転＝2
回転速度 (パルスの出力間隔)	1ステップ毎の間隔を指定 間隔＝割り込み間隔(100μsec)×値	例えば値を 60 とすると、60×100μsec＝6msec の間隔で 1 ステップづつ動きます。
角度 (ステップ数)	動かす角度を指定 角度＝1ステップ辺りの角度×ステップ数	例えば1ステップ 1.8° のパルスモータで 90° 動かす場合にセットする値は、90÷1.8＝50 です。

回転方向は値によってロテート命令の向きを決定します。尚、停止はロテートせずに今までのデータを出力します。回転速度、つまりパルスの出力間隔は割り込み内でカウントするウェイト値です。この値が小さければパルスの出力間隔は短い＝早く回転、となり、逆に値が大きくなる程、回転速度は遅くなります。角度は動作させるステップ数です。ステップ数×値で移動する角度が求められます。また、ステップ数×パルスの出力間隔でその角度に達するまでに要する時間を求める事が出来ます。パルスの出力間隔にピンとこない場合は移動に要する時間から値を決めてもよいでしょう。例えば、1 ステップ 1.8° のパルスモータで 90° 移動を 1sec で完了したければ、

移動時間＝ステップ数×パルスの出力間隔(＝割り込み間隔×値 n)

移動時間＝1sec、ステップ数＝90÷1.8＝50、割り込み間隔＝100μsec から、

1sec＝50×100μsec×n

n＝1sec／(50×100μsec)

n＝200

と求める事が出来ます。

以上を踏まえて作成したプログラムリストを次頁に示します。

```
;---------------------------------------------------------------
;
;   FILE        :plsmotor_00.src
;   DATE        :Tue, Oct 19, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------------

     .include   "io3687F_equ.inc"
```

```
     .export        _main
     .export        INTRTV

;*************************************************************
;    メインプログラム
;*************************************************************
     .section   P,code;,locate=H'EA00

_main:
     bsr        INITTV:16           ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
     bsr        INITPIO:16          ;PIO ｲﾆｼｬﾗｲｽﾞ

     mov.w      #WORK_AREA,r3       ;ﾜｰｸｴﾘｱ ｸﾘｱ
     mov.w      #WORK_AREAE-WORK_AREA-1,r1
     xor.b      r0l,r0l
```

```
    bsr     FILL:16

    mov.b   #H'33,r0l       ;ﾊﾟﾙｽﾓｰﾀへの出力ﾃﾞｰﾀ ｾｯﾄ
    mov.b   r0l,@PMDT

    mov.w   @PMSD_TBL+0,r0  ;回転方向初期値 ｾｯﾄ
    mov.w   r0,@ROT_FLG
    mov.w   @PMSD_TBL+2,r0  ;回転速度初期値 ｾｯﾄ
    mov.w   r0,@TVWAIT_CNT
    mov.w   r0,@TVWAIT_CNST
    mov.w   @PMSD_TBL+4,r0  ;回転角度(ｽﾃｯﾌﾟ数)初期値 ｾｯﾄ
    mov.w   r0,@PMSTEP_CNT

    andc    #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;----- メインループ ----------------------------------------
LOOP_00:
    bra     LOOP_00

;*********************************************************
;    割り込み処理
;*********************************************************
;---------------------------------------------------------
;    タイマV割り込み
;---------------------------------------------------------
INTRTV:
    push.l  er0
    push.l  er1

    mov.w   @TVWAIT_CNT,r0  ;回転速度の管理
    dec.w   #1,r0
    mov.w   r0,@TVWAIT_CNT
    bne     INTRTV_FF       ;ｶｳﾝﾄｱｯﾌﾟしていなければ終了

INTRTV_10:                  ;ｶｳﾝﾄｱｯﾌﾟ
    mov.b   @PMDT,r0l       ;ﾊﾟﾙｽﾓｰﾀを 1 ｽﾃｯﾌﾟ動作
    mov.w   @ROT_FLG,r1     ; ROT_FLG=0 : 停止
    beq     INTRTV_14       ;         =1 : 左ﾛｰﾃｰﾄ
    cmp.w   #2,r1           ;         =2 : 右ﾛｰﾃｰﾄ
    beq     INTRTV_12
    rotl.b  r0l
    bra     INTRTV_14
INTRTV_12:
    rotr.b  r0l
INTRTV_14:
    mov.b   r0l,@PMDT
    mov.b   r0l,@PDR3       ;ﾊﾟﾙｽﾓｰﾀへ出力

INTRTV_20:                  ;ｽﾃｯﾌﾟ数の管理
    mov.w   @PMSTEP_CNT,r0
    dec.w   #1,r0
    mov.w   r0,@PMSTEP_CNT
    beq     INTRTV_30       ;規定ｽﾃｯﾌﾟ数終えたら次へ移行
    mov.w   @TVWAIT_CNST,r0 ;回転速度値を再ｾｯﾄ
    mov.w   r0,@TVWAIT_CNT  ; 規定ｽﾃｯﾌﾟ数まで繰り返す
    bra     INTRTV_FF

INTRTV_30:                  ;次のｼｰｹﾝｽへ
    mov.w   @PMSEQ_CNT,r0   ;ｼｰｹﾝｽ番号の更新
    inc.w   #1,r0
    cmp.w   #PMSD_SIZE,r0
    bcs     INTRTV_32
    xor.w   r0,r0           ;全ｼｰｹﾝｽを終えたら元に戻る
INTRTV_32:
    mov.w   r0,@PMSEQ_CNT

    bsr     X6:16           ;ｼｰｹﾝｽﾃﾞｰﾀを読み出す
    mov.w   @(PMSD_TBL,er0),r1   ;回転方向のｾｯﾄ
    mov.w   r1,@ROT_FLG
    inc.l   #2,er0
    mov.w   @(PMSD_TBL,er0),r1   ;回転速度のｾｯﾄ
    mov.w   r1,@TVWAIT_CNT
    mov.w   r1,@TVWAIT_CNST
    inc.l   #2,er0
    mov.w   @(PMSD_TBL,er0),r1   ;回転角度(ｽﾃｯﾌﾟ数)のｾｯﾄ
    mov.w   r1,@PMSTEP_CNT

INTRTV_FF:
    bclr    #6,@TCSRV

    pop.l   er1
    pop.l   er0
    rte

;*********************************************************
;    サブルーチン
;*********************************************************
;---------------------------------------------------------
;         ＲＯ×６
;---------------------------------------------------------
X6:
    xor.w   e0,e0           ;2
    mov.l   er0,er1         ;2   ;ER1=ER0
    add.l   er0,er0         ;2   ;ER0+ER0=2ER0
    add.l   er1,er0         ;2   ;2ER0+ER1(=ER0)=3ER0
    add.l   er0,er0         ;2   ;3ER0+3ER0=6ER0
    rts                     ;8

;---------------------------------------------------------
;         データで埋める
;---------------------------------------------------------
;R3  = ｱﾄﾞﾚｽ
;R1  = ﾃﾞｰﾀ数
;R0L = ﾃﾞｰﾀ
FILL:
    mov.b   r0l,@r3         ;R0L ﾃﾞｰﾀで埋める
    inc.w   #1,r3
    dec.w   #1,r1
    bne     FILL
    rts

;---------------------------------------------------------
;    タイマVイニシャライズ
;---------------------------------------------------------
INITTV:
    mov.b   #D'125,r0l      ;CLK*16*125=100usec
    mov.b   r0l,@TCORA
    mov.b   #B'01001010,r0l ;CLK*16 RA ｺﾝﾍﾟｱﾏｯﾁ INTR&CLR
    mov.b   r0l,@TCRV0
    rts

;---------------------------------------------------------
;    ＰＩＯイニシャライズ
;---------------------------------------------------------
INITPIO:
    mov.b   #H'FF,r0l       ;PIO3 all Output
    mov.b   r0l,@PCR3
    rts

;*********************************************************
;         パルスモータ出力値データテーブル・エリア
;*********************************************************
;-----  ｼｰｹﾝｽﾃﾞｰﾀ ﾃｰﾌﾞﾙ  ---------------------------------
;.data.w  D'1  ;回転方向 0:停止/1:左ﾛｰﾃｰﾄ/2:右ﾛｰﾃｰﾄ
;.data.w  D'60 ;回転速度 100usec * 60 = 6msec
;.data.w  D'50 ;回転角度(ｽﾃｯﾌﾟ数) 1.8 * 50 = 90°
PMSD_TBL:           ;PulseMotorSequenceData Table
    .data.w   D'1,D'60,D'50
    .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
    .data.w   D'1,D'60,D'50
    .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
    .data.w   D'1,D'60,D'50
    .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
    .data.w   D'1,D'60,D'50
    .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec

    .data.w   D'2,D'60,D'50
    .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
```


2004-2005 Toyo-linx,Co.,Ltd.

```
     .data.w   D'2,D'60,D'50
     .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
     .data.w   D'2,D'60,D'50
     .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
     .data.w   D'2,D'60,D'50
     .data.w   D'0,D'1000,D'5 ;wait 100msec*5=500msec
PMSD_TBLE:
PMSD_SIZE       .equ (PMSD_TBLE-PMSD_TBL)/6


;************************************************************
;          データ・エリア
;************************************************************
     .section  D,data;,locate=H'F780
```

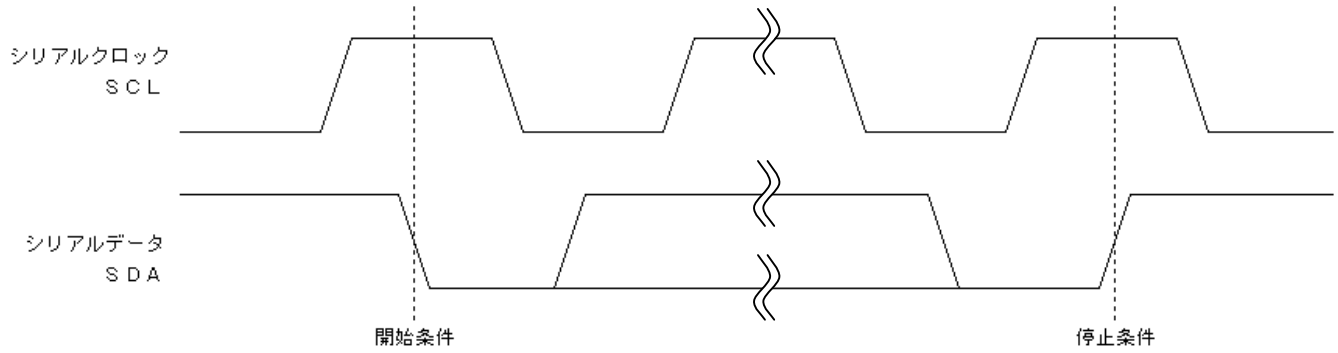
```
WORK_AREA:
;----- ﾊﾟﾙｽﾓｰﾀ -----------------
PMDT:           .res.b    1          ;ﾊﾟﾙｽﾓｰﾀ出力ﾃﾞｰﾀ
     .align     2
ROT_FLG:        .res.w    1          ;回転方向ﾌﾗｸﾞ
PMSTEP_CNT:     .res.w    1          ;ｽﾃｯﾌﾟ数ｶｳﾝﾀ
PMSEQ_CNT:      .res.w    1          ;ｼｰｹﾝｽ番号

;----- ﾀｲﾏV --------------------
TVWAIT_CNT:     .res.w    1          ;wait ｶｳﾝﾀ
TVWAIT_CNST:    .res.w    1          ;wait ｶｳﾝﾀ初期値
WORK_AREAE:

;============================================================
          .end
```

8-6 I²C バス仕様 EEPROM のリード／ライト

TK-3687 には I²C バス(Inter IC Bus)というデバイス同士の接続を想定した 2 線シリアルインターフェースが装備されています。ここではこの I²C バスを使用して EEPROM のリードライトを行ないます。

I²C バス仕様の EEPROM は各社から発売されていますが、いずれも仕様に則っているので基本的にはどのメーカを選んでも問題はありません。容量によってアドレスの指定が変わる程度です。ここでは 16kbit(2048×8)の EEPROM を使用します。

I²C バスはシリアルクロック‘SCL’を出力し、そのタイミングに合わせてシリアルデータ‘SDA’を出力します。通信シーケンスには通信の開始を示す“開始条件”と終了を示す“停止条件”が必要です。開始条件を発行するには SCL が High 状態の時に SDA を立ち下げます。また、停止条件を発行するには SCL が High の時に SDA を立ち上げます。データはこの開始条件と停止条件の間で送受信されます。



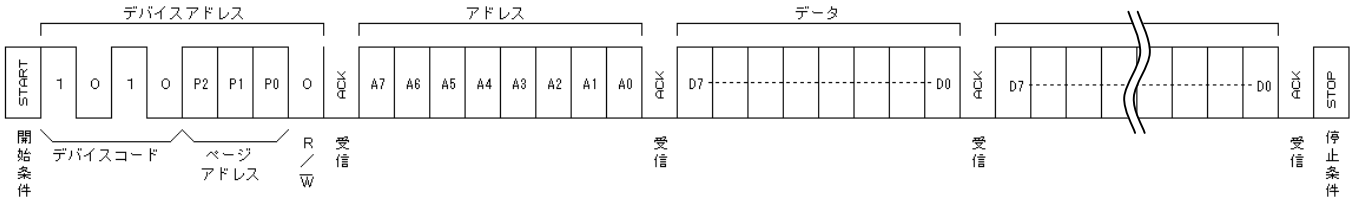
【 開始条件と停止条件 】

それでは具体的に EEPROM とのデータのやり取りを調べていきます。送受信するデータはデータ 8 ビット＋返信1ビットの計 9 ビットで構成されています。返信 1 ビットは“ACK(acknowledge)”といいデータを送信した場合には確認として相手から送り返されてくるビットです。また、逆にデータを受信した場合にはシーケンスに従って ACK をセットし送り返さなければなりません。一見面倒なように感じますが、H8/3687F に搭載されている I²C バスコントローラが ACK の取り込み、及び返答を自動で行なってくれます。

2004-2005 Toyo-linx,Co.,Ltd.

■ EEPROM への書き込み

まずは、書き込み動作を調べてみましょう。以下に書き込み時のシーケンスを示します。



【 書き込みシーケンス 】

開始条件発行後、まずデバイスアドレスを指定（送信）します。このデバイスアドレスは、デバイスを指定する 4 ビットのデバイスコード、EEPROM 内のメモリページを指定する 3 ビットのページアドレス（メーカによってはページではなくブロックと表現する場合もあります）、リード／ライトを示す 1 ビット、の計 8 ビットです。デバイスコードは EEPROM の場合 B'1010 となります。ページアドレスは EEPROM 内のアドレス A8～A10 に相当します。尚、EEPROM の容量は 16Kbit（2048×8bit）ですのでアドレスは H'000～H'7FF となります。リード／ライトビットは書き込みの場合、0 をセットします。

7	6	5	4	3	2	1	0
← デバイスコード →				← ページアドレス →			リード/ライト
1	0	1	0	P2	P1	P0	$\overline{R/W}$

【 デバイスアドレスのフォーマット 】

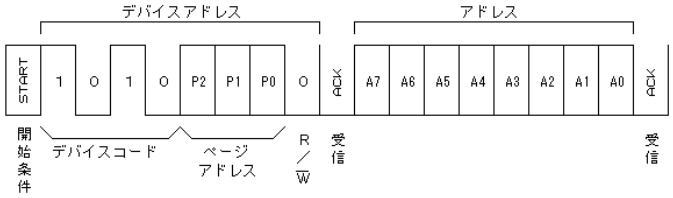
デバイスアドレスを送信したら返ってくる"ACK"をみて EEPROM が使用可能かをチェックします。EEPROM が応答可能な状態の場合は ACK は Low が返ってきます。しかし、何らかの原因（例えば書き込み動作中など）で応答できない場合 ACK は High（この状態を 'NACK' といいます）が返ってきます。

ACK＝Low なら、次にアドレス・書き込みデータと順に送信していきます。書き込みデータは連続して送信することで EEPROM 内でアドレスがインクリメントし、16 バイトまで書き込む事ができます。但しアドレスのインクリメントは下位 4 ビットしか行なわれないのでページがまたがる場合には注意が必要です。データを全て送信したら最後に停止条件を発行します。停止条件を発行すると EEPROM は書き込み動作に入ります。書き込み動作中は EEPROM にデータを送信しても何も応答がありません。そこでこの応答が無いのを利用して書き込み終了を確認する事ができます。つまり、開始条件を発行後デバイスアドレスを送信し返ってくる ACK を確認します。ACK=Low なら応答があったということなので書き込みは終了していることになります。逆に ACK＝High なら書き込み中と判断する事ができます。

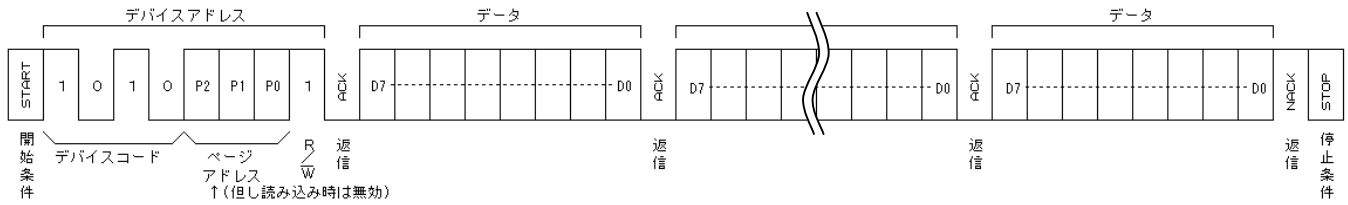
2004-2005 Toyo-linx,Co.,Ltd.

■ EEPROM の読み込み

次に読み込み動作を説明します。以下に読み込み時のシーケンスを示します。



【 読み込みシーケンス・ダミーライト 】



【 読み込みシーケンス・データリード 】

まず読み出すアドレスを指定する為に読み出しアドレスをダミーライトします。ダミーライトは書き込み時同様、開始条件を発行し、リード／ライトビットを 0 にセットしたデバイスアドレスを送信、ついでアドレスを送信します。ここまで送信したら再度開始条件を発行します。ダミーライトに停止条件は不要です。今度はリード／ライトビットをリードの 1 にセットしたデバイスアドレスを送信します。送信後は SCL に合わせて EEPROM からデータが出力されるのでそれを取り込みます。読み込みの場合はデータが相手から送られてくるのでACKをこちらから出力しなくてはなりません。読み込むデータが続いてある場合は ACK＝0 で、終了する場合は ACK＝1で応答します。ACK＝1で応答後、停止条件を発行して通信を終了します。読み込みの際には書き込み時のような 16 バイトという制約はありません。H'000～H'7FF まで全て連続して読み込む事ができます。尚、アドレス H'7FF 以降を読み出すとアドレスは H'000 に戻ります。それでは実際のプログラムに必要なレジスタを調べていきましょう。

SCL と SDA の入出力タイミングは全て H8/3687 が行なってくれますので、プログラムではレジスタに所定のデータをセットしたりビットを操作するのがメインになります。EEPROM と通信を行なうにあたり主要なレジスタを解説します。

■ ICCR2:I²C バスコントロールレジスタ 2

ICCR2 はバスの開放状態チェック、及び開始／停止条件の発行を行ないます。

7	6	5	4	3	2	1	0
BBSY	SCP	SDAO	SDAOP	SCLO	－	I²CRST	－

開始条件を発行するにあたりバスが他で使用されていない（開放されている）事が条件となります。そこで ICCR2 のビット 7:BBSY でバスが開放されているか確認します。BBSY=1 の場合、バスは他のデバイスが使用中ですのでBBSY=0になるまで待ちます。BBSY=0 でバスが開放されている事を確認できたら開始条件を発行します。開始条件を発行するには ICCR2 のビット 7:BBSY を 0、ビット 6:SCP を 1 にセットします。データの送受信を終え終了条件を発行するには BBSY=0・SCP=0 をセットします。尚、このビットセットは MOV 命令で行ないます。

2004-2005 Toyo-linx,Co.,Ltd.

■ **ICSR:I²C バスステータスレジスタ**

ICSR は各ビットをチェックする事で送受信の完了を知ることができます。

7	6	5	4	3	2	1	0
TDRE	TEND	RDRF	NACKF	STOP	AL/OVE	AAS	ADZ

データを送信する前にはビット 6:TEND をチェックし、データが送信可能な状態かをチェックします。TEND=1 ならデータ送信可能な状態なので送信データレジスタ:ICDRT に送信データをセットして送信を行ないます。

データの受信完了をチェックするにはビット 5:RDRF をチェックします。RDRF=1 でデータ受信完了を示しますので受信データレジスタ:ICDRR から受信したデータを読み取ります。

■ **ICIER:I²C バスインタラプトイネーブルレジスタ**

ICIER は受信 ACK ビットの確認、及び送信 ACK ビットの設定を行ないます。

7	6	5	4	3	2	1	0
TIE	TEIE	RIE	NAKIE	STIE	ACKE	ACKBR	ACKBT

データ送信時、受信した ACK ビットの状態を確認するには ICIER のビット 1:ACKBR を確認します。受信した ACK ビットがそのままセットされているので ACKBR=0 なら ACK 受信、ACKBR=1 なら NACK 受信となります。

データ受信時に送信する ACK ビットの極性はレジスタ ICIER のビット 0:ACKBT をセットします。データ受信時に ACK を返すのなら ACKBT=0 を、NACK を返すのなら ACKBT=1 をセットします。

■ **ICDRT:I²C バス送信データレジスタ**

ICDRT は送信データを格納するレジスタです。

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

ICSR のビット 6:TEND=1 なら送信データをこの ICDRT にセットします。後は I²C バスコントローラが SCL・SDA を制御しデータを送信します。

■ **ICDRR:I²C バス受信データレジスタ**

ICDRR は受信データが格納されるレジスタです。また、受信動作のトリガにもなっています。

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

受信動作はこのレジスタを読む事で開始されます。つまり、データを受信するには一度 ICDRR をダミーリードします。すると I²C バスコントローラは SCL を出力してデバイスからのデータ取り込み動作を開始します。レジスタ ICSR のビット 5:RDRF=1 なら受信動作完了です。続けてデータを受信する場合にはそのまま ICDRR をリードします。受信したデータを読み取るのと同時に次の受信動作が開始されます。最後のデータを受信するにはレジスタ ICIER のビット 0:ACKBT=1(NACK 返答)をセットして ICDRR をリードします。リードと同時に最後のデータの取

2004-2005 Toyo-linx,Co.,Ltd.

り込みを行ない NACK を返答してこれ以降データの受信が無い事をデバイスに通達します。このまま最後のデータを ICDRR から読み出してしまうとまた SCL を出力してしまうので、停止条件を発行し通信を終了してから最後のデータを ICDRR からリードします。

以下に EEPROM にデータを書き込み、書き込んだデータを読み出してポート P1 に出力するプログラムを示します。必要なパラメータをセットして書き込みのサブルーチン“EEPRSQWR”または読み込みサブルーチン“EEPRSQRD”をコールすれば EEPROM へリード／ライトを行ないます。セットするパラメータは次の通りです。

サブルーチン名	内　　容
EEPRSQWR	EEPROM へデータを書き込む
EEPRSQRD	EEPROM のデータを取り込む

変数名	サイズ[Byte]	内　　容
SLAVE_ADDR	1	デバイスアドレスをセット
EEPR_ADDR	2	リード／ライトする EEPROM の先頭アドレス
SQDT_ADDR	2	ライト時・書き込むデータ元アドレス リード時・読み取ったデータのセット先アドレス
SQDT_SIZE	2	リード／ライトするデータ数

また、各サブルーチン内で開始／停止条件の発行、ACK の確認、デバイスアドレス・アドレス・データの送信、データの取り込み、全てサブルーチン単位でまとめてあるので各操作時のレジスタの扱いはリストを参照して下さい。

```
;---------------------------------------------------------
;
;   FILE        :sampleIIC_00.src
;   DATE        :Wed, Dec 22, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;                                                  Y.Kikuchi
;---------------------------------------------------------
; ■I2Cテストプログラム
;   I2Cバス仕様の EEPROM にデータを書き込み、同一アドレスを
;   読み込んでポートに表示する。
;---------------------------------------------------------

     .include   "io3687F_equ.inc"
     .export    _main

;*********************************************************
;    定数
;*********************************************************
SLAVEADDR_CNST  .equ H'A0 ;bit7-4 : Device Type Identifier
                          ;         1010 = EEPROM
                          ;bit3-1 : Chip Enable
                          ;         000  = EEPROM pin3-1 ADDR
                          ;bit0   : Read or Write
                          ;         (1)     (0)
WRITEADDR_CNST  .equ H'0000     ;書き込み先アドレス
WRITEDATA_CNST  .equ H'55       ;書き込みデータ
READADDR_CNST   .equ H'0000     ;読み込み先アドレス


;*********************************************************
;    メインプログラム
;*********************************************************
     .section   P,code,locate=H'EA00 ;※locate で絶対ｱﾄﾞﾚｽ

_main:
;----- イニシャライズ -----------------------------------
     bsr        INITPIO:16      ;PIO ｲﾆｼｬﾗｲｽﾞ

     mov.w      #WORK_AREA,r3   ;ﾜｰｸｴﾘｱ ｸﾘｱ
     mov.w      #WORK_AREAE-WORK_AREA-1,r1
     xor.b      r0l,r0l
     bsr        FILL:16

;----- メインループ -------------------------------------
     mov.b      #WRITEDATA_CNST,r0l  ;書き込みﾃﾞｰﾀｾｯﾄ
     mov.b      r0l,@WR_DATA
;.....  EEPROMのｸﾘｱ  .....................................
     mov.b      #SLAVEADDR_CNST,r0l  ;ｽﾚｰﾌﾞｱﾄﾞﾚｽ ｾｯﾄ
     mov.b      r0l,@SLAVE_ADDR
     mov.w      #WRITEADDR_CNST,r0   ;EEPROM ｱﾄﾞﾚｽ ｾｯﾄ
     mov.w      r0,@EEPR_ADDR
     mov.w      #H'800,r0            ;ﾃﾞｰﾀ数
     mov.w      r0,@SQDT_SIZE
     bsr        EEPRCLR:16           ;EEPROM のｸﾘｱ
;.....  EEPROMへ書き込み  ..............................
     mov.b      #SLAVEADDR_CNST,r0l  ;ｽﾚｰﾌﾞｱﾄﾞﾚｽ ｾｯﾄ
     mov.b      r0l,@SLAVE_ADDR
     mov.w      #WRITEADDR_CNST,r0   ;EEPROM ｱﾄﾞﾚｽ ｾｯﾄ
     mov.w      r0,@EEPR_ADDR
     mov.w      #WR_DATA,r0          ;書き込むﾃﾞｰﾀ元ｱﾄﾞﾚｽ
     mov.w      r0,@SQDT_ADDR
     mov.w      #D'1,r0              ;ﾃﾞｰﾀ数
     mov.w      r0,@SQDT_SIZE
     bsr        EEPRSQWR:16          ;ｼｰｹﾝｼｬﾙ書き込み
;.....  EEPROMから読み出し  ..............................
     mov.b      #SLAVEADDR_CNST,r0l  ;ｽﾚｰﾌﾞｱﾄﾞﾚｽ ｾｯﾄ
```

```
     mov.b     r0l,@SLAVE_ADDR
     mov.w     #READADDR_CNST,r0    ;EEPROM ｱﾄﾞﾚｽ ｾｯﾄ
     mov.w     r0,@EEPR_ADDR
     mov.w     #RD_DATA,r0          ;読み出したﾃﾞｰﾀのｾｯﾄ先
     mov.w     r0,@SQDT_ADDR
     mov.w     #D'1,r0              ;ﾃﾞｰﾀ数
     mov.w     r0,@SQDT_SIZE
     bsr       EEPRSQRD:16          ;ｼｰｹﾝｼｬﾙ読み出し

     mov.b     @RD_DATA,r0l         ;ﾃﾞｰﾀをﾎﾟｰﾄに出力する
     mov.b     r0l,@PDR1
     bclr      #0,@PDR2
     btst      #3,r0l
     beq       main_FF
     bset      #0,@PDR2
main_FF:

     bra       $    ;EEPROM の書き替え回数には限りがある
                    ;ので、ループせずにここで停止


;************************************************************
;    割り込み処理
;************************************************************


;************************************************************
;    サブルーチン
;************************************************************
;------------------------------------------------------------
;          EEPROM へシーケンシャル書き込み
;------------------------------------------------------------
;    SLAVE_ADDR : デバイス(スレーブ)アドレス
;    EEPR_ADDR  : EEPROM アドレス
;    SQDT_ADDR  : 書き込むデータの元アドレス
;    SQDT_SIZE  : 書き込むデータ数
;------------------------------------------------------------
EEPRSQWR:
EEPRSQWR_00:
;..... 書き換え動作終了? ..............................
EEPRSQWR_01:
     bsr       WRCHKI2C:16     ;書き換え動作終了ﾁｪｯｸ
     bcs       EEPRSQWR_01
     mov.b     r1l,r1l
     bne       EEPRSQWR_01

;..... ページライト ....................................
EEPRSQWR_10:
     bsr       INITI2C:16      ;ｲﾆｼｬﾗｲｽﾞ
     mov.w     @EEPR_ADDR,r0
     mov.w     r0,@ROM_ADDR

EEPRSQWR_11:
     bsr       STRTI2C:16      ;ｽﾀｰﾄｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQWR_11
EEPRSQWR_12:
     bsr       DASI2C:16       ;ﾃﾞﾊﾞｲｽｱﾄﾞﾚｽ
     bcs       EEPRSQWR_12
EEPRSQWR_13:
     bsr       WAITACKI2C:16   ;ｱｸﾉﾘｯｼﾞ待ち
     bcs       EEPRSQWR_13
     bne       EEPRSQWR_20
EEPRSQWR_14:
     bsr       MASI2C:16       ;ﾒﾓﾘｱﾄﾞﾚｽ(ﾜｰﾄﾞｱﾄﾞﾚｽ)
     bcs       EEPRSQWR_14

     mov.w     @EEPR_ADDR,r2
     mov.w     @SQDT_ADDR,r3
     mov.w     #0,e3
     mov.w     @SQDT_SIZE,r4
     mov.b     #16,r5l         ;1ﾍﾟｰｼﾞのｻｲｽﾞ
EEPRSQWR_15:
     mov.b     @er3,r0l
     mov.b     r0l,@ROM_DATA
EEPRSQWR_16:
     bsr       WDSI2C:16       ;ﾃﾞｰﾀﾗｲﾄ
     bcs       EEPRSQWR_16
     inc.w     #1,r2
     inc.l     #1,er3
     dec.w     #1,r4
     beq       EEPRSQWR_17
     dec.b     r5l             ;1ﾍﾟｰｼﾞ分ﾗｲﾄ完了?
     bne       EEPRSQWR_15     ; No→ｼﾞｬﾝﾌﾟ
EEPRSQWR_17:
     mov.w     r2,@EEPR_ADDR
     mov.w     r3,@SQDT_ADDR
     mov.w     r4,@SQDT_SIZE

EEPRSQWR_18:
     bsr       STOPI2C:16      ;ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQWR_18

     mov.w     @SQDT_SIZE,r0   ;全ﾃﾞｰﾀﾗｲﾄ完了?
     bne       EEPRSQWR_00     ; No→ｼﾞｬﾝﾌﾟ

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

EEPRSQWR_20:
EEPRSQWR_21:
     bsr       STOPI2C:16      ;ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQWR_21
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;------------------------------------------------------------
;          EEPROM からのシーケンシャル読み出し
;------------------------------------------------------------
;    SLAVE_ADDR : デバイス(スレーブ)アドレス
;    EEPR_ADDR  : EEPROM アドレス
;    SQDT_ADDR  : 読み出したデータのセット先アドレス
;    SQDT_SIZE  : 読み出すデータ数
;------------------------------------------------------------
EEPRSQRD:
EEPRSQRD_00:
;..... 書き換え動作終了? ..............................
EEPRSQRD_01:
     bsr       WRCHKI2C:16     ;書き換え動作終了ﾁｪｯｸ
     bcs       EEPRSQRD_01
     mov.b     r1l,r1l
     bne       EEPRSQRD_01

;..... シーケンシャルリード ............................
EEPRSQRD_10:
     bsr       INITI2C:16      ;ｲﾆｼｬﾗｲｽﾞ
     mov.w     @EEPR_ADDR,r0
     mov.w     r0,@ROM_ADDR

EEPRSQRD_11:
     bsr       STRTI2C:16      ;ｽﾀｰﾄｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQRD_11
EEPRSQRD_12:
     bsr       DASI2C:16       ;ﾃﾞﾊﾞｲｽｱﾄﾞﾚｽ
     bcs       EEPRSQRD_12
EEPRSQRD_13:
     bsr       WAITACKI2C:16   ;ｱｸﾉﾘｯｼﾞ待ち
     bcs       EEPRSQRD_13
     bne       EEPRSQRD_20:16
EEPRSQRD_14:
     bsr       MASI2C:16       ;ﾒﾓﾘｱﾄﾞﾚｽ(ﾜｰﾄﾞｱﾄﾞﾚｽ)
     bcs       EEPRSQRD_14

     mov.b     @SLAVE_ADDR,r0l ;ﾘｰﾄﾞ命令ｺｰﾄﾞに変更
     bset      #0,r0l
     mov.b     r0l,@SLAVE_ADDR
```


2004-2005 Toyo-linx,Co.,Ltd.

```
EEPRSQRD_16:
     bsr       RSTRTI2C:16     ;ﾘｽﾀｰﾄｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQRD_16
EEPRSQRD_17:
     bsr       DASI2C:16       ;ﾃﾞﾊﾞｲｽｱﾄﾞﾚｽ
     bcs       EEPRSQRD_17
EEPRSQRD_18:
     bsr       WAITACKI2C:16   ;ｱｸﾉﾘｯｼﾞ待ち
     bcs       EEPRSQRD_18
     bne       EEPRSQRD_20
EEPRSQRD_19:
     bsr       MSTRMDI2C:16    ;ﾏｽﾀ受信ﾓｰﾄﾞに切り替え
     bcs       EEPRSQRD_19

     mov.w     @EEPR_ADDR,r2
     mov.w     @SQDT_ADDR,r3
     mov.w     #0,e3
     mov.w     @SQDT_SIZE,r4
     cmp.w     #1,r4           ;ﾃﾞｰﾀ数=1ﾊﾞｲﾄ?
     beq       EEPRSQRD_1b     ; Yes

     bsr       RDSI2C:16       ;ﾃﾞｰﾀ受信指定
     mov.b     @ICDRR,r0l      ;ﾀﾞﾐｰﾘｰﾄﾞ,受信動作開始

EEPRSQRD_1a:
     bsr       RDDTI2C:16      ;ﾃﾞｰﾀﾘｰﾄﾞ
     mov.b     r1l,@er3
     inc.w     #1,r2
     inc.l     #1,er3
     dec.w     #1,r4
     cmp.w     #1,r4           ;残り 1 バイト?
     bhi       EEPRSQRD_1a     ; No→ｼﾞｬﾝﾌﾟ
     bsr       LRDSI2C:16      ;最終ﾃﾞｰﾀ受信指定
     bra       EEPRSQRD_1c

EEPRSQRD_1b:
     bsr       LRDSI2C:16      ;最終ﾃﾞｰﾀ受信指定
     mov.b     @ICDRR,r0l      ;ﾀﾞﾐｰﾘｰﾄﾞ,受信動作開始

EEPRSQRD_1c:
     bsr       RDFDTI2C:16     ;最終ﾃﾞｰﾀﾘｰﾄﾞ
     mov.b     r1l,@er3
     inc.w     #1,r2
     inc.l     #1,er3
     dec.w     #1,r4
     mov.w     r2,@EEPR_ADDR
     mov.w     r3,@SQDT_ADDR
     mov.w     r4,@SQDT_SIZE

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

EEPRSQRD_20:
EEPRSQRD_21:
     bsr       STOPI2C:16      ;ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRSQRD_21
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;-------------------------------------------------------
;          EEPROM のクリア
;-------------------------------------------------------
;    SLAVE_ADDR : デバイス(スレーブ)アドレス
;    EEPR_ADDR  : EEPROM アドレス
;    SQDT_SIZE  : クリアするデータ数
;-------------------------------------------------------
EEPRCLR:
     mov.b     #0,r0l               ;00h でｸﾘｱする
     mov.b     r0l,@EEPR_DATA
     bsr       EEPRFILL
     rts
```

```
;-------------------------------------------------------
;          EEPROM のデータフィル
;-------------------------------------------------------
;    SLAVE_ADDR : デバイス(スレーブ)アドレス
;    EEPR_DATA  : EEPROM にセットするデータ
;    EEPR_ADDR  : EEPROM アドレス
;    SQDT_SIZE  : セットするデータ数
;-------------------------------------------------------
EEPRFILL:
EEPRFILL_00:
;..... 書き換え動作終了? ..............................
EEPRFILL_01:
     bsr       WRCHKI2C:16     ;書き換え動作終了ﾁｪｯｸ
     bcs       EEPRFILL_01
     mov.b     r1l,r1l
     bne       EEPRFILL_01

;..... ページライト ....................................
EEPRFILL_10:
     bsr       INITI2C:16      ;ｲﾆｼｬﾗｲｽﾞ
     mov.w     @EEPR_ADDR,r0
     mov.w     r0,@ROM_ADDR

EEPRFILL_11:
     bsr       STRTI2C:16      ;ｽﾀｰﾄｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRFILL_11
EEPRFILL_12:
     bsr       DASI2C:16       ;ﾃﾞﾊﾞｲｽｱﾄﾞﾚｽ
     bcs       EEPRFILL_12
EEPRFILL_13:
     bsr       WAITACKI2C:16   ;ｱｸﾉﾘｯｼﾞ待ち
     bcs       EEPRFILL_13
     bne       EEPRFILL_20
EEPRFILL_14:
     bsr       MASI2C:16       ;ﾒﾓﾘｱﾄﾞﾚｽ(ﾜｰﾄﾞｱﾄﾞﾚｽ)
     bcs       EEPRFILL_14

     mov.b     @EEPR_DATA,r0l  ;ｾｯﾄするﾃﾞｰﾀ
     mov.b     r0l,@ROM_DATA
     mov.w     @EEPR_ADDR,r2
     mov.w     @SQDT_SIZE,r4
     mov.b     #16,r5l         ;1ﾍﾟｰｼﾞのｻｲｽﾞ
EEPRFILL_15:
EEPRFILL_16:
     bsr       WDSI2C:16       ;ﾃﾞｰﾀﾗｲﾄ
     bcs       EEPRFILL_16
     inc.w     #1,r2
     dec.w     #1,r4
     beq       EEPRFILL_17
     dec.b     r5l             ;1ﾍﾟｰｼﾞ分ﾗｲﾄ完了?
     bne       EEPRFILL_15     ; No→ｼﾞｬﾝﾌﾟ
EEPRFILL_17:
     mov.w     r2,@EEPR_ADDR
     mov.wV    r4,@SQDT_SIZE

EEPRFILL_18:
     bsr       STOPI2C:16      ;ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRFILL_18

     mov.w     @SQDT_SIZE,r0   ;全ﾃﾞｰﾀﾗｲﾄ完了?
     bne       EEPRFILL_00     ; No→ｼﾞｬﾝﾌﾟ

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

EEPRFILL_20:
EEPRFILL_21:
     bsr       STOPI2C:16      ;ｽﾄｯﾌﾟｺﾝﾃﾞｨｼｮﾝ
     bcs       EEPRFILL_21
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts
```


2004-2005 Toyo-linx,Co.,Ltd.

```
;-------------------------------------------------------
;          開始条件発行
;-------------------------------------------------------
;戻り値 : Cy (ｷｬﾘｰﾌﾗｸﾞ)
;         Cy=0 : OK
;         Cy=1 : NG バスが開放されていない(他で使用されている)
STRTI2C:
     mov.b     @ICCR2,r0l      ;ﾊﾞｽは開放されているか？
     btst      #7,r0l          ; 1=Busy
     bne       STRTI2C_10      ; ﾊﾞｽﾋﾞｼﾞｰなら Cy=1 で終了

STRT2I2C:
     mov.b     #B'10110101,r0l ;ﾏｽﾀ送信ﾓｰﾄﾞ
     mov.b     r0l,@ICCR1

     mov.b     #H'80,r0l       ;開始条件発行
     mov.b     r0l,@ICCR2      ; BBSY=1 / SCP=0

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

STRTI2C_10:
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;-------------------------------------------------------
;          停止条件発行
;-------------------------------------------------------
STOPI2C:
     mov.b     @STOPI2C_STA,r0l
     bne       STOPI2C_00

     mov.b     @ICSR,r0l       ;送信完了ﾁｪｯｸ
     and.b     #H'C0,r0l       ;TDRE[7]=1,TEND[6]=1 送信完了
     xor.b     #H'C0,r0l
     bne       STOPI2C_10      ;送信未完なら Cy=1 で終了

     xor.b     r0l,r0l         ;停止条件発行
     mov.b     r0l,@ICCR2      ; BBSY[7]=0,SCP[6]=0

     mov.b     @ICSR,r0l       ;ﾄﾗﾝｽﾐｯﾄｴﾝﾄﾞﾋﾞｯﾄ ｸﾘｱ
     bclr      #6,r0l          ; TEND=0
     mov.b     r0l,@ICSR

     mov.b     #1,r0l
     mov.b     r0l,@STOPI2C_STA

STOPI2C_00:
     mov.b     @ICSR,r0l       ;停止条件検出待ち
     btst      #3,r0l          ; STOP[3]=1 なら検出
     beq       STOPI2C_10

     xor.b     r0l,r0l
     mov.b     r0l,@STOPI2C_STA

     bsr       WAIT4u:16       ;4usec 待ち

     andc      #H'FE,CCR       ;Cy=0 : OK
     nop
     rts

STOPI2C_10:
     orc       #H'01,CCR       ;Cy=1 : NG
     nop
     rts

;-------------------------------------------------------
;          開始条件再発行
;-------------------------------------------------------
;戻り値 : Cy (ｷｬﾘｰﾌﾗｸﾞ)
;         Cy=0 : OK
;         Cy=1 : NG 送信未完了
RSTRTI2C:
     mov.b     @ICSR,r0l       ;送信完了ﾁｪｯｸ
     and.b     #H'C0,r0l       ;TDRE[7]=1,TEND[6]=1 なら完了
     xor.b     #H'C0,r0l
     bne       RSTRTI2C_10     ;送信未完なら Cy=1 で終了

     mov.b     #H'80,r0l       ;開始条件発行
     mov.b     r0l,@ICCR2      ; BBSY=1 / SCP=0

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

RSTRTI2C_10:
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;-------------------------------------------------------
;          デバイスアドレス設定
;-------------------------------------------------------
DASI2C:
     mov.b     @SLAVE_ADDR,r0l ;ｽﾚｰﾌﾞｱﾄﾞﾚｽの作成
     mov.w     @ROM_ADDR,r1    ; ROM_ADDR の A8,A9,A10 を
     shal.b    r1h             ;SLAVE_ADDR の bit5,6,7 へ
     and.b     #H'0e,r1h
     or.b      r0l,r1h         ;結合
     mov.b     r1h,r1l         ;送信ﾃﾞｰﾀは R1L へ
     bra       TXDI2C:16       ;送信

;-------------------------------------------------------
;          ライトデータ設定
;-------------------------------------------------------
WDSI2C:
     mov.b     @ROM_DATA,r1l   ;R1L=書き込みﾃﾞｰﾀ
     bra       TXDI2C:16       ;送信

;-------------------------------------------------------
;          データ送信
;-------------------------------------------------------
;引数   : R1L (送信データ)
;戻り値 : Cy (ｷｬﾘｰﾌﾗｸﾞ)
;         Cy=0 : OK
;         Cy=1 : NG 未送信データがある為送信できず
TXDI2C:
     mov.b     @ICSR,r0l       ;送信ﾊﾞｯﾌｧは空いているか？
     btst      #7,r0l          ; いなければ Cy=1 で抜ける
     beq       TXDI2C_10

     mov.b     r1l,@ICDRT      ;ﾃﾞｰﾀ送信

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

TXDI2C_10:
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;-------------------------------------------------------
;          メモリアドレス設定
;-------------------------------------------------------
MASI2C:
     mov.w     @ROM_ADDR,r1    ;R1=ROM ｱﾄﾞﾚｽ
     bsr       TXDI2C:16       ;ｱﾄﾞﾚｽ送信
     bcs       MASI2C_10       ;Cy=0 で送信完了
                               ; Cy=1 は送信 NG なので抜ける

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
MASI2C_10:                     ;NG の場合 Cy=1
     rts
```


2004-2005 Toyo-linx,Co.,Ltd.

```
;---------------------------------------------------------
;          ＡＣＫ待ち
;---------------------------------------------------------
;戻り値 : Cy (ｷｬﾘｰﾌﾗｸﾞ)
;       : Z (ｾﾞﾛﾌﾗｸﾞ)
;         Cy=0 : OK / Z=0 : ACK=0
;                  / Z=1 : ACK=1
;         Cy=1 : NG 転送が完了していない
WAITACKI2C:
     mov.b     @ICSR,r0l       ;転送完了＆転送準備完了ﾁｪｯｸ
     btst      #6,r0l
     beq       WAITACKI2C_10   ; 完了していなければ
                               ; Cy=1 で抜ける
     mov.b     @ICIER,r0l      ;ICIER(#2,ACKBR)=ACK ﾃﾞｰﾀ
     and.b     #H'02,r0l       ;ACK=0: Z / ACK=1:NZ

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

WAITACKI2C_10:
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;---------------------------------------------------------
;          マスタ受信モード切り替え
;---------------------------------------------------------
MSTRMDI2C:
     mov.b     @ICSR,r0l       ;送信完了ﾁｪｯｸ
     btst      #6,r0l          ;ICSR(#6,TEND)=1 なら送信終了
     beq       WAITACKI2C_10   ; 完了していなければ
                               ; Cy=1 で抜ける
     bclr      #6,r0l          ; 送信完了ﾌﾗｸﾞ ｸﾘｱ
     mov.b     r0l,@ICSR

     mov.b     @ICCR1,r0l      ;ﾏｽﾀ受信ﾓｰﾄﾞ切替
     bclr      #4,r0l          ; ICCR2(#4,TRS)=0
     bset      #5,r0l          ; ICCR2(#5,MST)=1
     mov.b     r0l,@ICCR1

     mov.b     @ICSR,r0l       ;TDRE ｸﾘｱ
     bclr      #7,r0l          ; ICSR(#7,TDRE)=0
     mov.b     r0l,@ICSR

     andc      #H'FE,CCR       ;OK : Cy=0
     nop
     rts

MSTRMDI2C_10:
     orc       #H'01,CCR       ;NG : Cy=1
     nop
     rts

;---------------------------------------------------------
;          最終データリード
;---------------------------------------------------------
RDFDTI2C:
     mov.b     @ICSR,r0l       ;受信ﾃﾞｰﾀ転送待ち
     btst      #5,r0l          ; ICSR(#5,RDRF)=1 で転送完了
     beq       RDFDTI2C

     mov.b     @ICCR2,r0l      ;停止条件発行
     and.b     #H'3F,r0l       ; ICCR2(#7,BBSY)=0
     mov.b     r0l,@ICCR2      ; ICCR2(#6,SCP) =0

RDFDTI2C_00:
     mov.b     @ICSR,r0l       ;停止条件生成待ち
     btst      #3,r0l          ; ICSR(#3,STOP)=1 まで待つ
     beq       RDFDTI2C_00

     mov.b     @ICDRR,r1l      ;最終ﾃﾞｰﾀ ﾘｰﾄﾞ
     mov.b     @ICCR1,r0l      ;ｽﾚｰﾌﾞ 受信ﾓｰﾄﾞに設定
     and.b     #H'9F,r0l       ; ICCR1(#6,RCVD)=0
     mov.b     r0l,@ICCR1      ; ICCR1(#5,MST) =0

     andc      #H'FE,CCR
     nop
     rts

;---------------------------------------------------------
;          最終データ読み込み指定
;---------------------------------------------------------
LRDSI2C:
     mov.b     @ICIER,r0l      ;ﾃﾞｰﾀ受信後 NACK を返送する
     bset      #0,r0l               ; ICIER(#0,ACKBT)=1
     mov.b     r0l,@ICIER

     mov.b     @ICCR1,r0l      ;ﾃﾞｰﾀ受信後の受信を禁止
     bset      #6,r0l               ; ICCR1(#6,RCVD)=1
     mov.b     r0l,@ICCR1

     rts

;---------------------------------------------------------
;          データリード
;---------------------------------------------------------
RDDTI2C:
     mov.b     @ICSR,r0l       ;受信ﾃﾞｰﾀ転送待ち
     btst      #5,r0l          ; ICSR(#5,RDRF)=1 で転送完了
     beq       RDDTI2C
     mov.b     @ICDRR,r1l      ;ﾃﾞｰﾀ ﾘｰﾄﾞ
     rts

;---------------------------------------------------------
;          データ読み込み指定
;---------------------------------------------------------
RDSI2C:
     mov.b     @ICIER,r0l      ;ﾃﾞｰﾀ受信後 ACK を返送する
     bclr      #0,r0l          ; ICIER(#0,ACKBT)=0
     mov.b     r0l,@ICIER
     rts

;---------------------------------------------------------
;          書き込み終了チェック
;---------------------------------------------------------
WRCHKI2C:
     mov.b     @IIC_STA,r0l
     beq       WRCHKI2C_00
     cmp.b     #1,r0l
     beq       WRCHKI2C_10
     cmp.b     #2,r0l
     beq       WRCHKI2C_20
     cmp.b     #3,r0l
     beq       WRCHKI2C_30
     cmp.b     #4,r0l
     beq       WRCHKI2C_40

     xor.b     r0l,r0l              ;ｽﾃｰﾀｽ ｸﾘｱ
     mov.b     r0l,@IIC_STA

     orc       #H'01,CCR       ;NG : Cy=1
     rts

;-----  I2C ｲﾆｼｬﾗｲｽﾞ  -----
WRCHKI2C_00:
     bsr       INITI2C:16      ;I2C ｲﾆｼｬﾗｲｽﾞ
     mov.b     #1,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@IIC_STA

;-----  開始条件発行  -----
WRCHKI2C_10:
     bsr       STRTI2C:16      ;開始条件発行
     bcs       WRCHKI2C_FF
     mov.b     #2,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@IIC_STA
```


2004-2005 Toyo-linx,Co.,Ltd.

```
;-----  ﾃﾞﾊﾞｲｽ ｱﾄﾞﾚｽ ｾｯﾄ  -----
WRCHKI2C_20:
     bsr       DASI2C:16       ;ﾃﾞﾊﾞｲｽ ｱﾄﾞﾚｽ 送信
     bcs       WRCHKI2C_FF
     mov.b     #3,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@IIC_STA

;-----  ACK 待ち  -----
WRCHKI2C_30:
     bsr       WAITACKI2C:16   ;ACK を待つ
     bcs       WRCHKI2C_FF
     bne       WRCHKI2C_32     ;ACK/NACK ﾁｪｯｸ
     mov.b     #0,r1l          ;ACK : R1L=0
     bra       WRCHKI2C_34

WRCHKI2C_32:
     mov.b     #1,r1l          ;NACK : R1L=1
WRCHKI2C_34:
     mov.b     #4,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@IIC_STA

;-----  終了条件発行  -----
WRCHKI2C_40:
     bsr       STOPI2C:16
     bcs       WRCHKI2C_FF

     xor.b     r0l,r0l         ;ｽﾃｰﾀｽ ｸﾘｱ
     mov.b     r0l,@IIC_STA

     andc      #H'FE,CCR       ;OK : Cy=0 / R1L=ACK

WRCHKI2C_FF:                   ;NG : Cy=1
     rts

;---------------------------------------------------------
;          データで埋める
;---------------------------------------------------------
;R3  = ｱﾄﾞﾚｽ
;R1  = ﾃﾞｰﾀ数
;R0L = ﾃﾞｰﾀ
FILL:
     mov.b     r0l,@r3    ;ｱﾄﾞﾚｽ R3 から R1 だけ R0L で埋める
     inc.w     #1,r3
     dec.w     #1,r1
     bne       FILL
     rts

;---------------------------------------------------------
;          ４μsec Ｗａｉｔ
;---------------------------------------------------------
WAIT4u:
     push.l    ER6
     mov.w     #20*10/6,R6
     bra       WAIT:16

;---------------------------------------------------------
;          Ｗａｉｔ
;---------------------------------------------------------
WAIT:
     dec.w     #1,R6
     bne       WAIT
     pop.l     ER6
     rts

;---------------------------------------------------------
;          ＩＩＣイニシャライズ
;---------------------------------------------------------
INITI2C:
     mov.b     #B'10000000,r0l ;ｽﾚｰﾌﾞｱﾄﾞﾚｽ=1000000
     mov.b     r0l,@SAR

     mov.b     #B'10000101,r0l ;ｽﾚｰﾌﾞ受信ﾓｰﾄﾞ 200kbps
     mov.b     r0l,@ICCR1
     mov.b     #B'00110000,r0l ;MSB ﾌｧｰｽﾄ NoWait 9bit ﾌｫｰﾏｯﾄ
     mov.b     r0l,@ICMR

     rts

;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INITPIO:
     xor.b     r0l,r0l         ;初期出力設定
     mov.b     r0l,@PDR1       ;ﾎﾟｰﾄ1 : 初期出力 Low
     mov.b     r0l,@PDR2       ;ﾎﾟｰﾄ2 : 初期出力 Low

     mov.b     #H'FF,r0l       ;入出力設定
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ1 : 全て出力
     mov.b     r0l,@PCR2       ;ﾎﾟｰﾄ2 : 全て出力

     rts


;*********************************************************
;          データ・エリア
;*********************************************************
          .section   D,data,locate=H'F780

WORK_AREA:
WR_DATA:       .res.b    D'16 ;書き込みﾃﾞｰﾀ
RD_DATA:       .res.b    D'16 ;読み出しﾃﾞｰﾀ

SLAVE_ADDR:    .res.b    1    ;ﾃﾞﾊﾞｲｽ(ｽﾚｰﾌﾞ)ｱﾄﾞﾚｽ
EEPR_DATA:     .res.b    1    ;EEPROM ﾃﾞｰﾀ
          .align    2
EEPR_ADDR:     .res.w    1    ;EEPROM ｱﾄﾞﾚｽ
SQDT_ADDR:     .res.w    1    ;読み出したﾃﾞｰﾀのｾｯﾄ先
                              ;書き込むﾃﾞｰﾀ元ｱﾄﾞﾚｽ
SQDT_SIZE:     .res.w    1    ;R/W するﾃﾞｰﾀ数

ROM_ADDR:      .res.w    1
ROM_DATA:      .res.b    1
WRCHKI2C_STA:  .res.b    1
STOPI2C_STA:   .res.b    1
MASI2C_STA:    .res.b    1
IIC_STA:       .res.b    1
          .align    2
WORK_AREAE:

;=========================================================
          .end
```

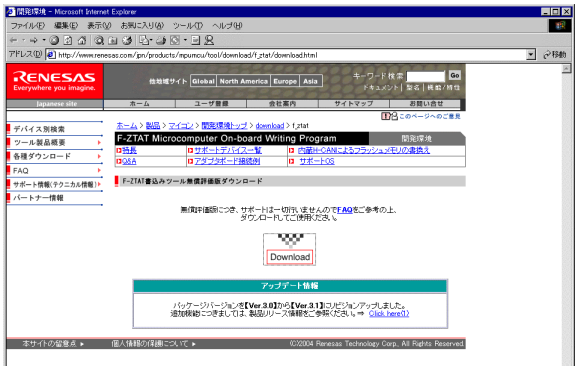

2004-2005 Toyo-linx,Co.,Ltd.

付録-1 FDT のダウンロード

ユーザが作成したプログラムを flashROM に書き込む、又はハイパーH8 を再度書き込む場合には“FDT(Flash Development Toolkit)”を使用します。付録-1 では FDT のダウンロード方法を、付録-2 では FDT のセッティングと書き込み手順、また複数ファイルの書き込み手順を紹介します。尚 FDT をダウンロード済みの際は次頁へお進み下さい。

■FDT のダウンロード

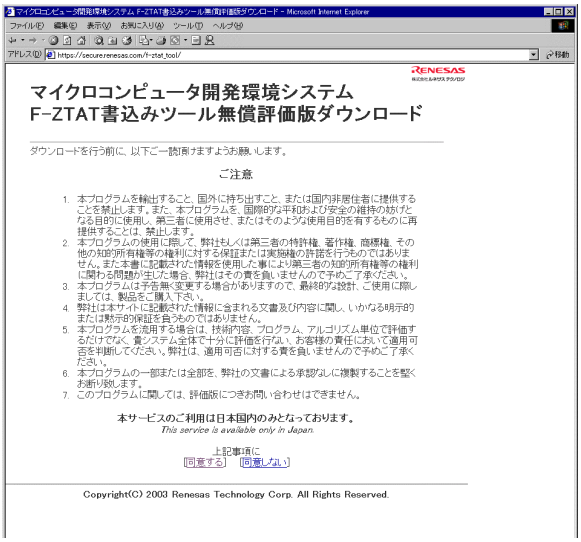
1. FDT は以下のサイトからダウンロードして下さい。
   http://www.renesas.com/jpn/products/mpumcu/tool/download/f_ztat/download.html
   （2004 年 4 月現在）



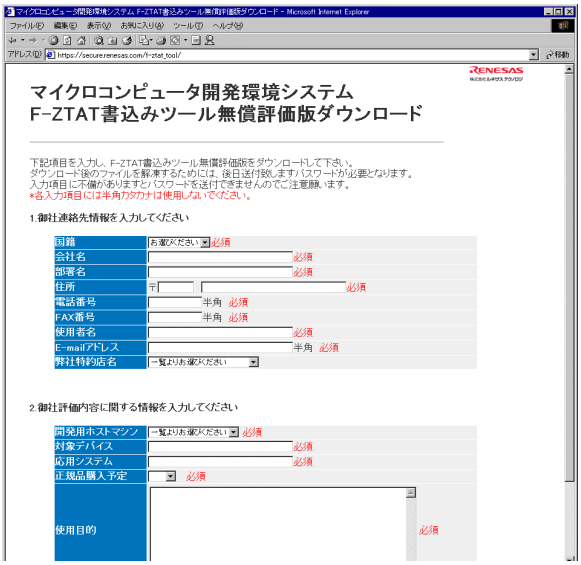
2. ダウンロードするには 

をクリックします。

3. 注意事項が記されたページへ移動するので内容を読み、同意した上で[同意する]をクリックします。



4. ユーザ情報を入力し、プログラムをダウンロードします。入力したメールアドレスに、プログラムを解凍する際必要なパスワードが送られてくるので入力事項に間違いが無い様、注意して下さい。



2004-2005 Toyo-linx,Co.,Ltd.

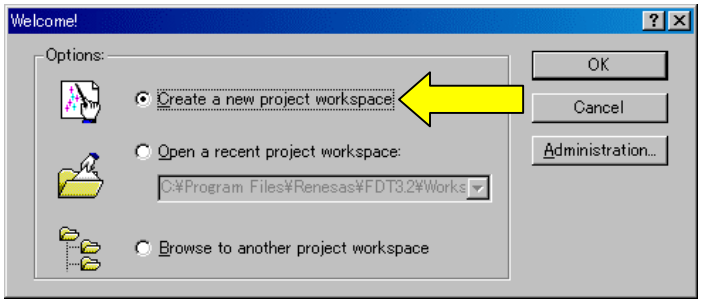
付録-2 FDT での書き込み手順

H8 書き込みツール“Flash Development Toolkit（FDT）”を用いて FDT のセッティングから TK-3687 へプログラム書き込みまで、順を追って説明していきます。

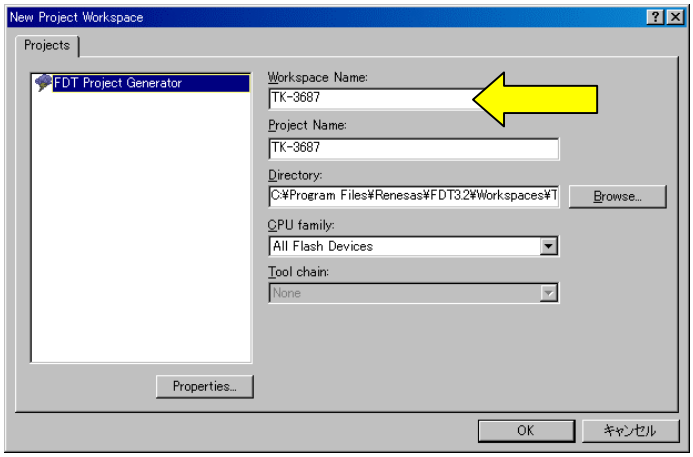
■FDT のセッティング（ワークスペースとプロジェクトの立ち上げ）

1. スタートメニューから“**Flash Development Toolkit 3.2**”を起動します。

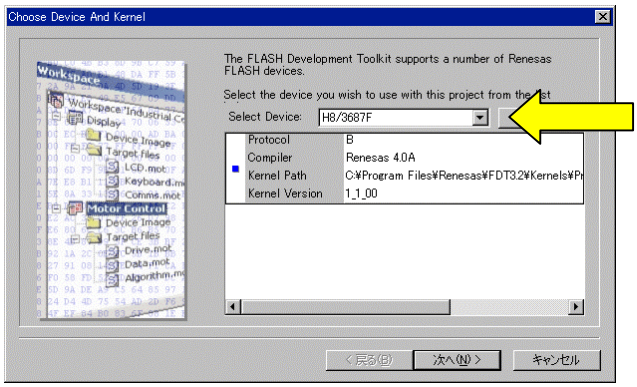
2. 右図のようなダイアログが開くので、“**Create a new project Workspace**”を選択して OK をクリックします。



3. “**Workspace Name**”を決定します。名前は自由に決めて結構です（ここでは TK-3687 としています）。またワークスペースを作成するディレクトリを指定したい場合は“**Directory:**”の Browse... をクリックしディレクトリを指定して下さい。よければ OK をクリックし次へ進みます。

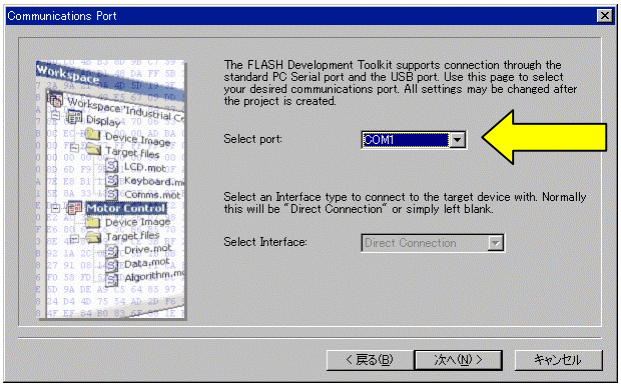


4. デバイスを選択します。“**Select Device:**”の欄で“**H8/3687F**”を選択し、次へ(N) > をクリックします。

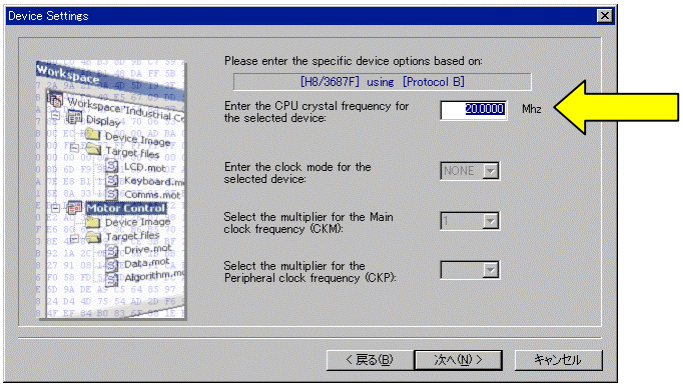


2004-2005 Toyo-linx,Co.,Ltd.

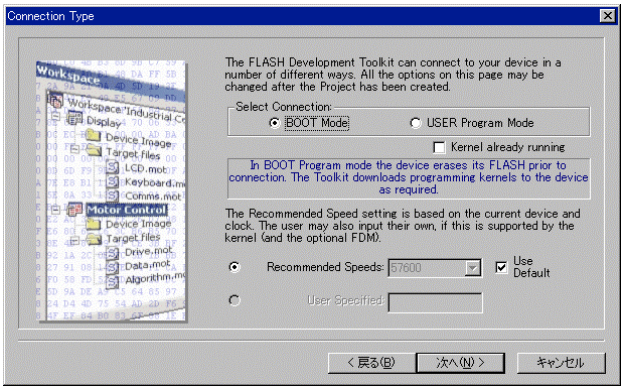
5. 使用する Com ポートを選択します。“**Select port:**”で接続する Com ポートを選択し、次へ(N) > をクリックします。



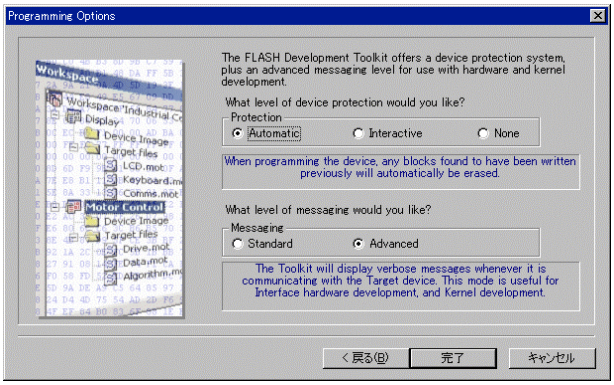
6. CPU のクロックを入力します。“**Enter the CPU crystal frequency …**”の欄に実装されているクロックの周波数“**20.00**”MHz を入力し、次へ(N) > をクリックします。



7. この後出てくる項目は入力・変更の必要はないので 次へ(N) > をクリックします。



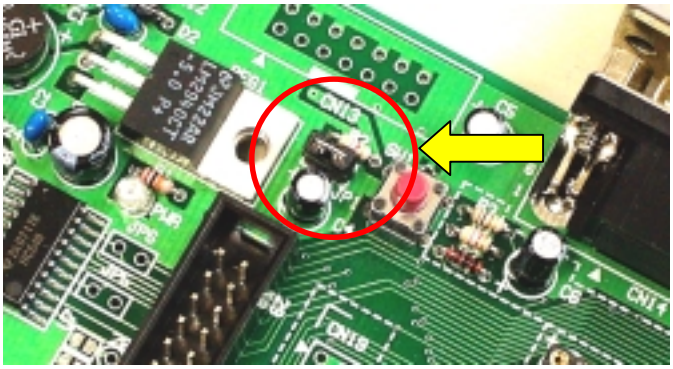
ここでも変更は無いので 完了 をクリックします。以上でワークスペースとプロジェクトの立ち上げは完了です。



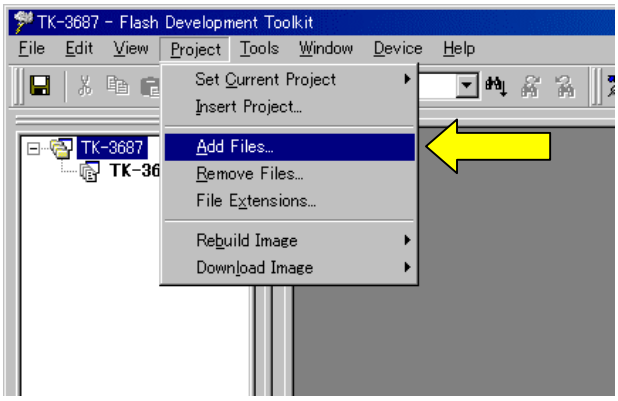
2004-2005 Toyo-linx,Co.,Ltd.

■ファイルのダウンロード

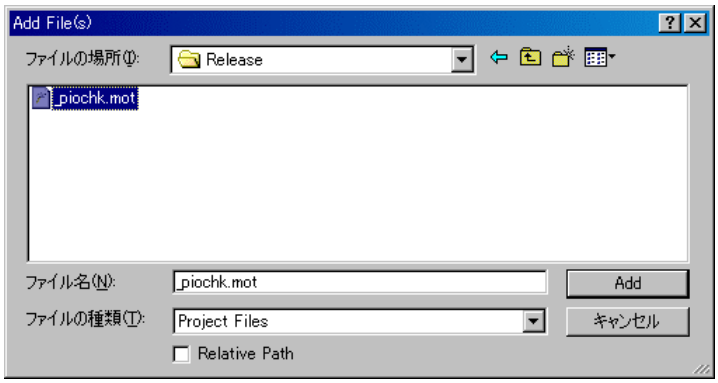
1. まず TK-3687 とパソコンとを接続します。基板上のジャンパ **JP1 をショート**して、RS-232C ケーブルでパソコンと接続し電源を投入、又はリセットをして下さい（ブートモードで起動）。



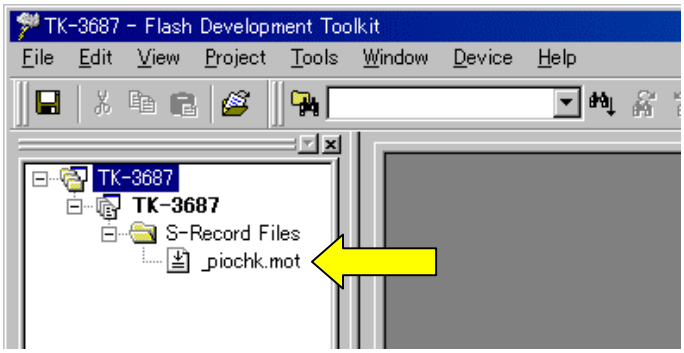
2. 次にダウンロードするファイルをプロジェクトに追加します。メニューバーから“**Project > Add Files...**”を選択します。



3. ダウンロードするファイルのフォルダを開き、該当するファイルを選択して Add をクリックして下さい。

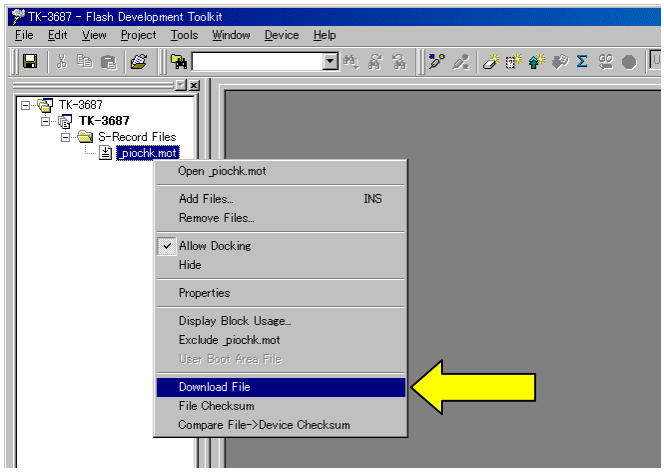


4. 以上でファイルが追加されました。画面左のルートディレクトリ内“**S-Record Files**”に選択したファイルが追加されたのを確認して下さい。（右の画面では 3 で指定したファイル“_piochk.mot”が追加されています）

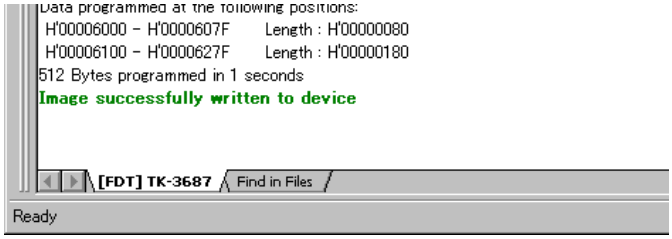


2004-2005 Toyo-linx,Co.,Ltd.

5. 追加したファイルをデバイスへダウンロードします。追加されたターゲットファイルを**右クリック**し、“Download File”を選択すると、ダウンロードを開始します。



6. 右図の“Image successfully written to device”のメッセージが表示されれば終了です。基板のジャンパ JP1 を外し、リセットスイッチを押して下さい。ダウンロードしたプログラムが走り始めます（通常モード）。



■ターゲットファイルの追加

ターゲットファイルを追加しダウンロード時に選択する事が出来ます。ファイルを追加するには前述の**■ファイルのダウンロード**の**2，3**を繰り返し行って下さい。TK-3687 へのダウンロードは今まで同様ダウンロードしたいファイルを右クリックでし“Download File to Device”を選択して下さい。

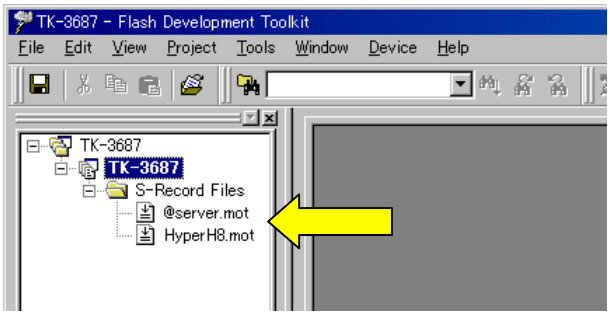
■複数のファイルをダウンロードする（デバイスイメージの作成とダウンロード）

複数のファイルをダウンロードするには、そのダウンロードファイルを一つにまとめた“デバイスイメージ”を作成する必要があります。以下にファイルの追加からデバイスイメージの作成、ダウンロードまでの手順を説明します。

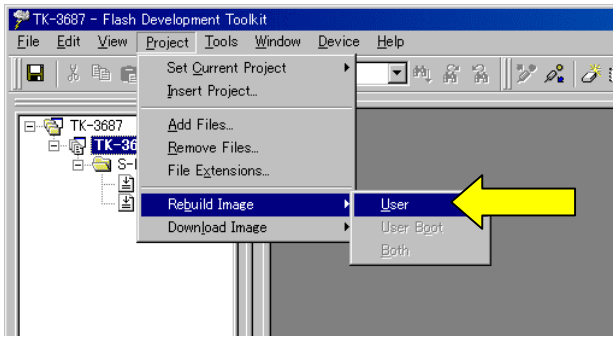
1. まずダウンロードするファイルを追加します。**■ファイルのダウンロード**の項目**2，3**を繰り返してターゲットファイルを追加して下さい。但し追加するファイルは次の条件を満たしている必要があります。

★それぞれのアドレスが重複していないこと

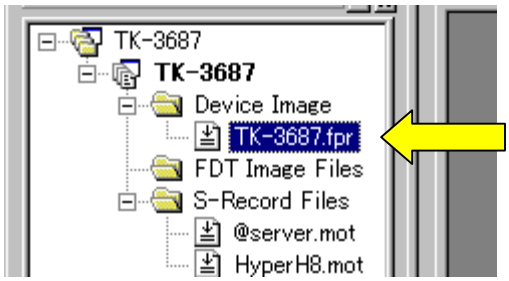
★フラッシュメモリのサイズを越えていないこと



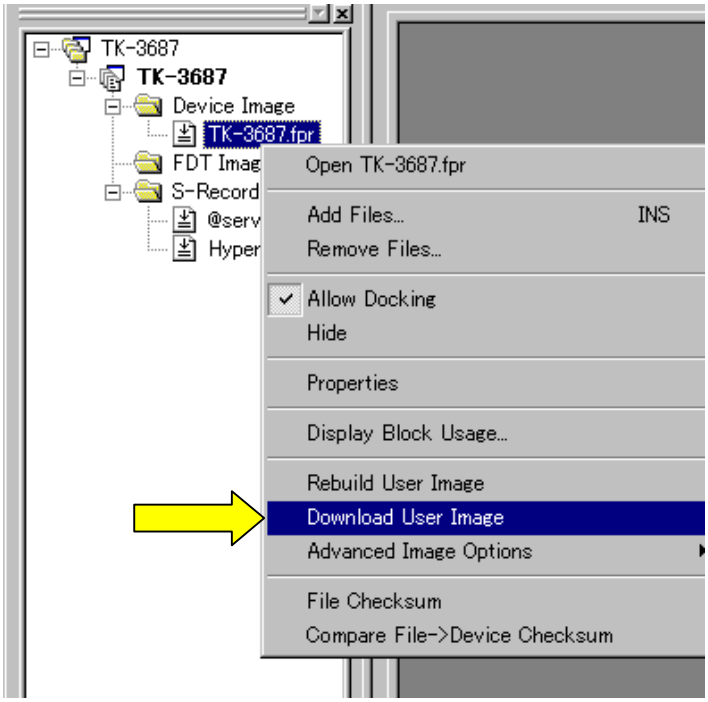
2. 次にダウンロードする複数のファイルを一つにまとめた“デバイスイメージ”を作成します。メニューバーから“**Project ＞ Rebuild Image ＞ User**”を選択して下さい。



“**Image Build Succeeded:...User Image added to workspace**”のメッセージが表示され、画面左ルートディレクトリ“Device Image”フォルダ下にデバイスイメージ“**(プロジェクト名).fpr**”が作成されます。(右図では TK-3687.fpr です。)



3. 最後に作成したデバイスイメージを TK-3687 にダウンロードします。作成されたデバイスイメージファイルを右クリックし“**Download User Image**”を選択して下さい。



以上で複数ファイルのダウンロードは完了です。

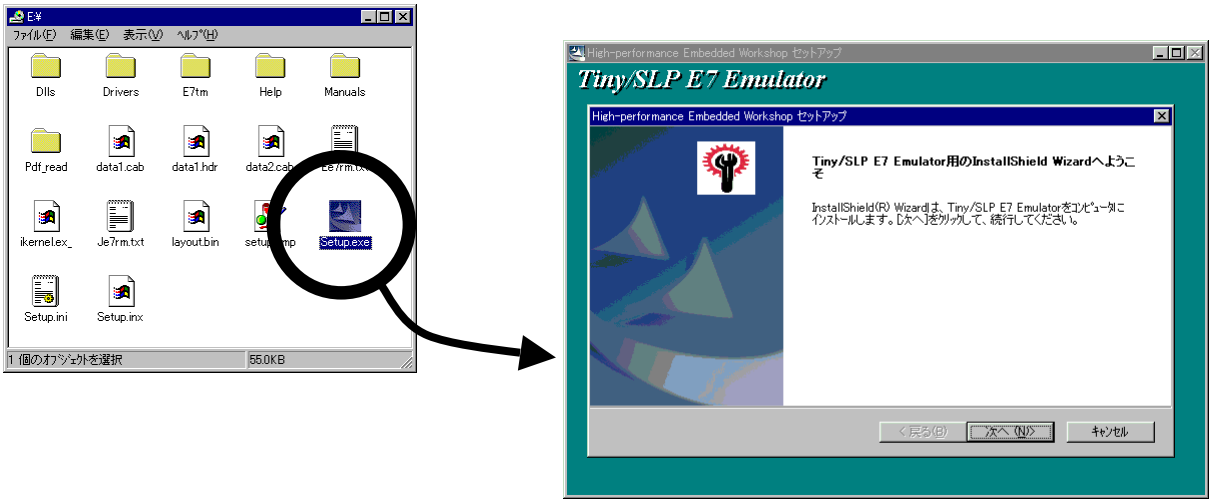
付録-3 エミュレータを使ったダウンロードと実行方法

作成したプログラムを“E7”でダウンロード・実行してみましょう。その前に，“E7”のエミュレータソフトのインストールはお済みでしょうか。まだの方は以下の手順でインストールしてください。

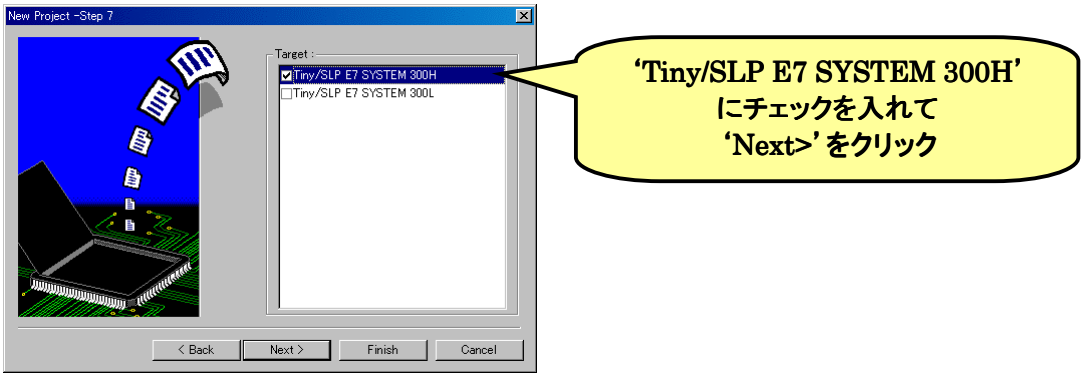
まず最初に，インストールされている**無償版コンパイラ“HEW”のバージョンが Version2.2(release15)以降**かご確認ください。それより前のバージョンで“E7”は動作しません。5 章を参照し，最新版の“HEW”をインストールしてください。(“E7“の CD-ROM のバージョンが Version2.0.00 以降の時はエミュレータソフトのインストールと同時に最新の“HEW”もインストールされます。)

次に，“E7”の CD の中の‘setup.exe’を実行して下さい。あとは，インストールウィザードに従いインストールします。このとき，**無償版コンパイラの“HEW”がインストールされているフォルダと同一フォルダに**，“E7”の“HEW”をインストールしてください。ビルドとデバッグを一つの“HEW”から操作する事ができるようになります。

さらにここで“E7”とパソコンを接続する際の各種設定(USBドライバ，E7 のファームウェア等)を済ませます。設定方法の詳細については“E7”のマニュアルをご覧下さい。

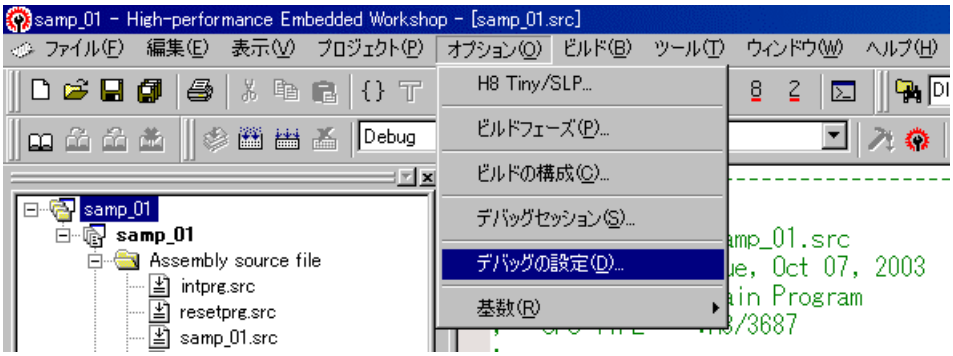


E7 のインストールを終えたら、デバッグするプログラムを作成します。既に作成されているプログラムをデバッグしたい場合も新たにプロジェクトを作り直して下さい。これは E7 を使用するのに必要な情報が含まれていないからです。プログラムの作成は 6 章を参照して下さい。E7 をインストールするとプロジェクト立ち上げ時、下のような画面が出ますので、‘Tiny/SLP E7 SYSTEM 300H’にチェックを入れ、‘Next >’をクリックします。

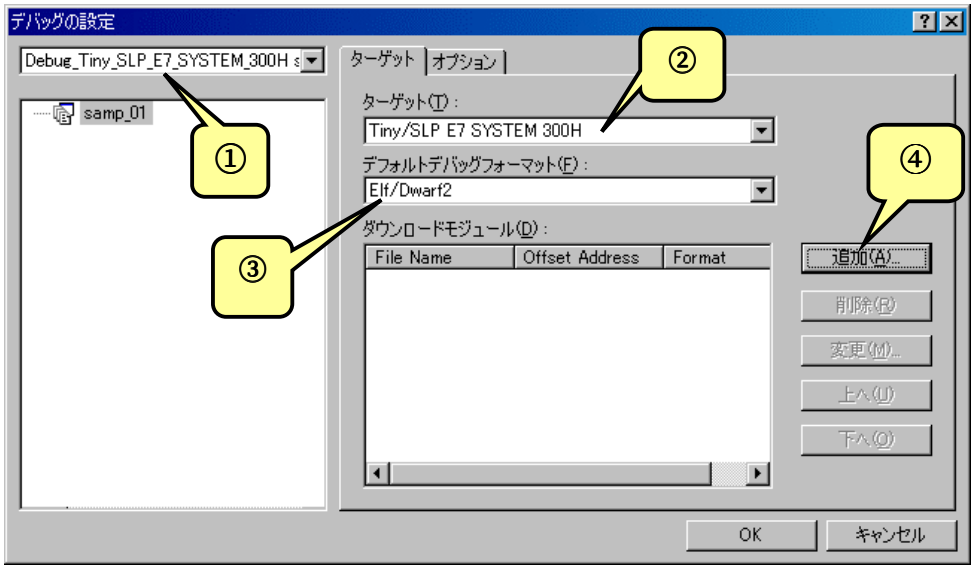


2004-2005 Toyo-linx,Co.,Ltd.

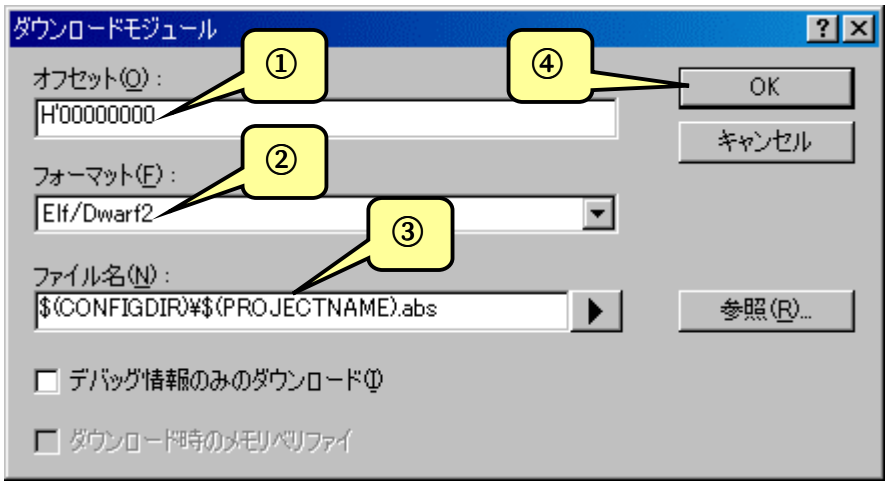
次に、HEW からデバッグするプロジェクトを起動し“E7 エミュレータ起動時の設定”を行ないます。メニューからオプション→デバッグの設定を選択して下さい。



すると，‘デバッグの設定’ダイアログボックスが開きます。①‘デバッガセッション’ドロップダウンリストボックスから**‘Debug_Tiny_SLP_E7_SYSTEM_300H_session’**を選びます。②‘ターゲット’ドロップダウンリストボックスから**‘Tiny/SLP E7 SYSTEM 300H’**を選びます。③‘デフォルトデバッグフォーマット’ドロップダウンリストボックスから**‘Elf/Dwarf2’**を選びます。④‘ダウンロードモジュール’を追加するため‘追加’ボタンをクリックします。

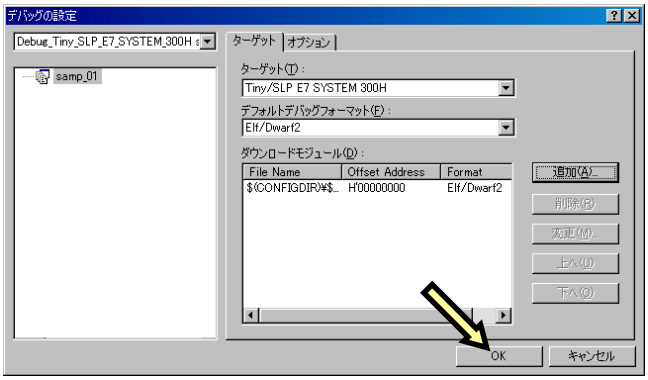


‘ダウンロードモジュール’ダイアログボックスが開きます。①‘オフセット’を**‘H’00000000’**に設定，②‘フォーマット’ドロップダウンリストボックスから**‘Elf/Dwarf2’**を選択，③‘ファイル名’に**‘$(CONFIGDIR)¥$(PROJICTNAME).abs’**を入力して，④‘OK’をクリックします。



2004-2005 Toyo-linx,Co.,Ltd.

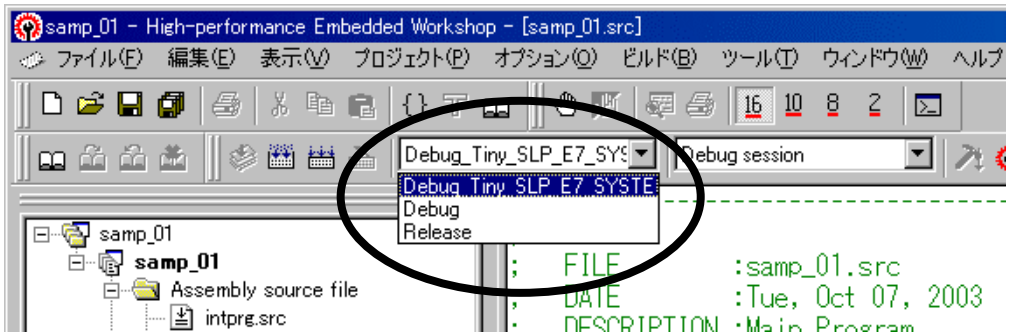
そうすると，‘デバッグの設定’ダイアログボックスは次のようになります。‘OK’をクリックしてダイアログボックスを閉じます。



これで，全ての設定は終わりました。次は，いよいよ“HEW”をデバッグモードにして“E7”に接続します。まず，TK-3687 の電源はオフにしてください。“E7”とパソコンを USB ケーブルで接続します。そして，“E7”と TK-3687 の CN13（14 ピンコネクタ）を付属のケーブルで接続します。なお，**この時点ではまだ TK-3687 の電源はオフのままにしておいてください。**

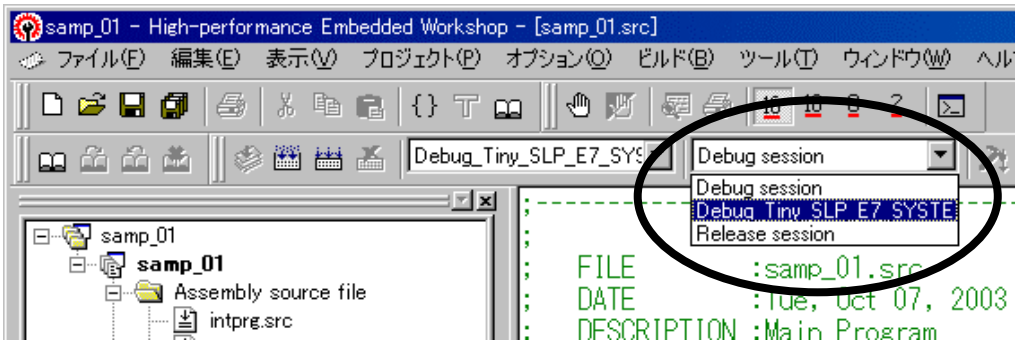
“E7”は CPU のフラッシュメモリにプログラムを書き込んでデバッグを行います。6 章で作成した“samp_01”は RAM 上にプログラムが割り当てられているので、フラッシュ ROM 上にプログラムを割り付けなおします。

ツールバーの中、左側のリストボックスから‘Debug_Tiny_SLP_E7_SYSTEM’を選択します。

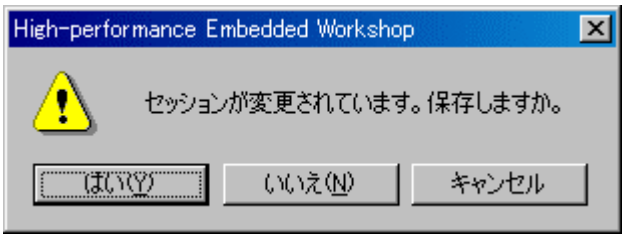


これで“E7”用にプログラムがフラッシュ ROM 上に割り当てられました。尚、3 章‘メモリマップ’で示したようにユーザが使用できない RAM 領域もありますので、“E7”を使用する時はその領域を使わないよう注意して下さい。

次にその右側にあるリストボックスから‘Debug Tiny SLP E7 SYSTEM 300H session’を選択します。そうすると，“E7”との接続が開始されます。



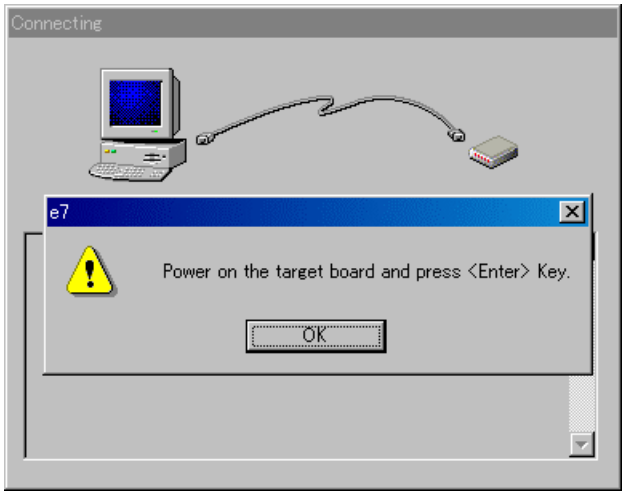
右のような警告が出ますが，‘はい’をクリックします。



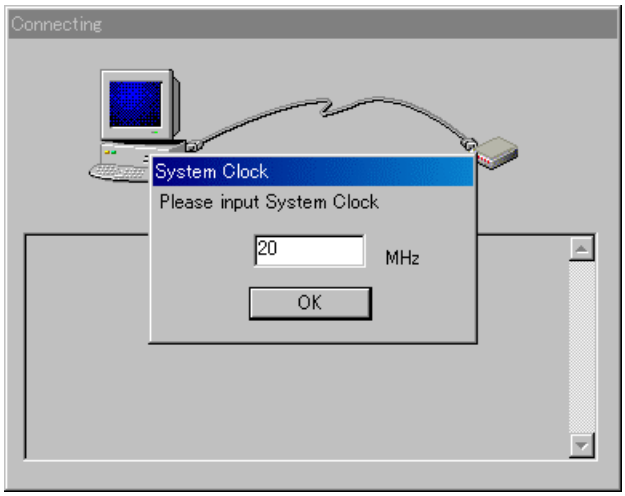
2004-2005 Toyo-linx,Co.,Ltd.

‘Select Emulator mode’ダイアログボックスが開きます。内容が右図の通りか確認して‘OK’をクリックします。

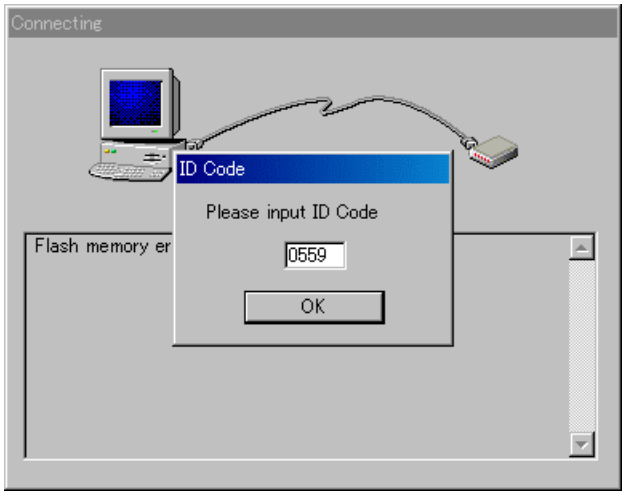
右のダイアログボックスが出たら TK-3687 の電源をオンします。それから，‘Enter’キーを押すか，‘OK’をクリックします。



システムクロックの周波数を入力します。TK-3687 の場合は 20MHz です。入力したら‘OK’をクリックします。

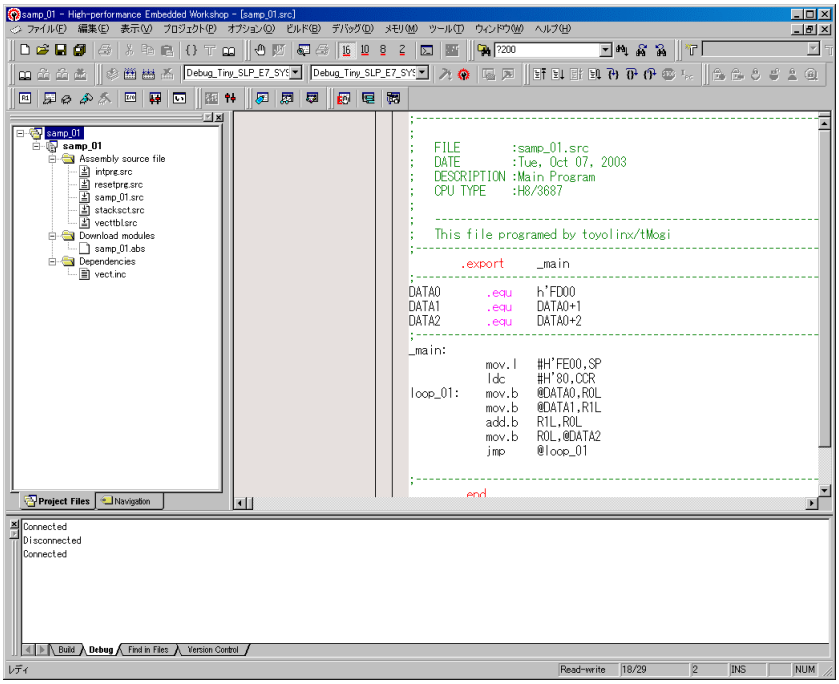


任意の ID コードを入力します。右図では 0559 となっていますが、0 でも構いません。入力したら‘OK’をクリックします。

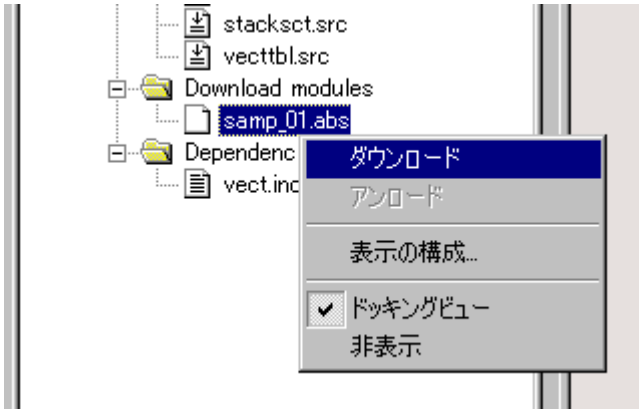


2004-2005 Toyo-linx,Co.,Ltd.

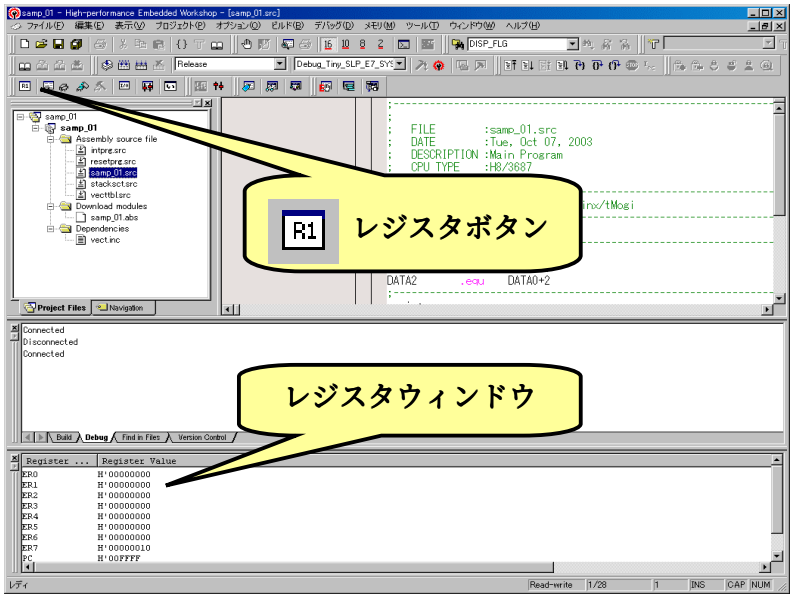
図のように，アウトプットウィンドウに‘Connected’と表示されたら“E7”との接続完了です。



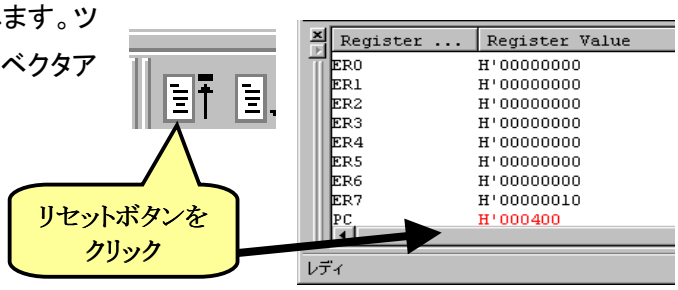
しかし，このときにはまだプログラムはダウンロードされていません。ワークスペースウィンドウの Download modules 内にある‘プロジェクト名.abs’を右クリックしてメニューを出し，‘ダウンロード’をクリックします。これで，TK-3687 に実装されている‘H8/3687’のフラッシュメモリにプログラムが書き込まれました。



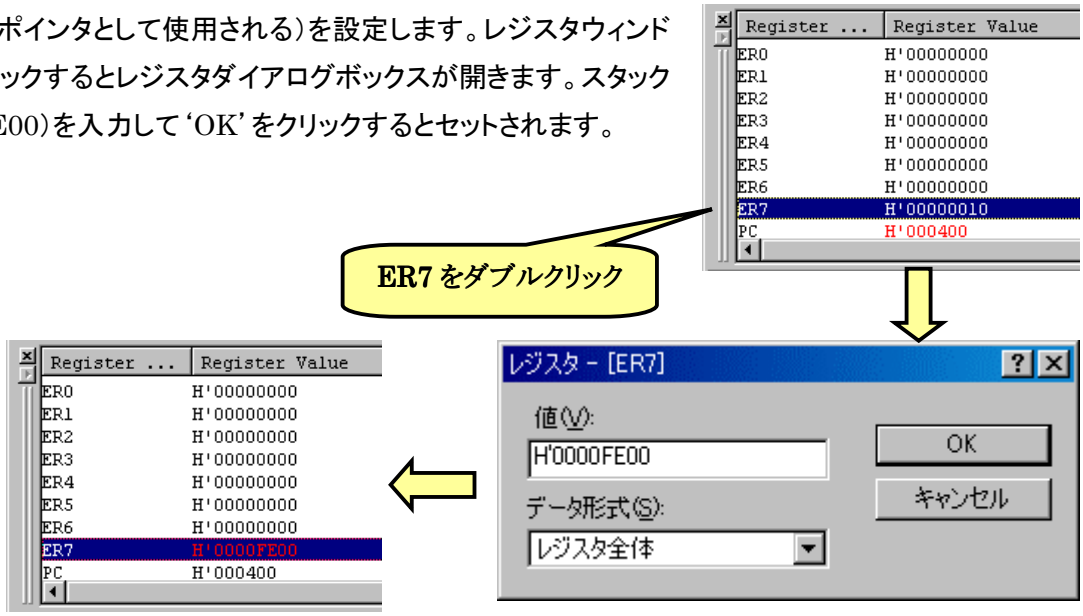
続いて，プログラムを実行してみましょう。この時点ではプログラムカウンタもスタックポインタも不定のため，あらためて指定する必要があります。ツールバーのレジスタボタンをクリックして，レジスタウィンドウを開きます。



まず，PC（プログラムカウンタ）を先頭アドレスに設定します。ツールバーのリセットボタンをクリックします。すると，リセットベクタアドレスの値（この場合 H’000400）がセットされます。



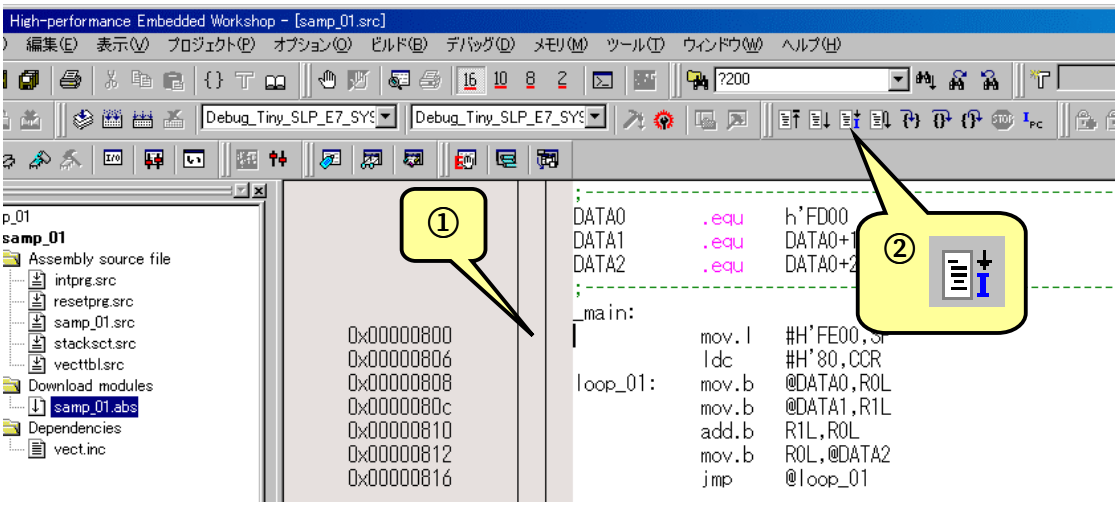
次に ER7（スタックポインタとして使用される）を設定します。レジスタウィンドウの ER7 をダブルクリックするとレジスタダイアログボックスが開きます。スタックポインタの初期値（H’FE00）を入力して‘OK’をクリックするとセットされます。



ここまでセットしたら，本格的なデバッグが可能になります。一例として‘カーソルまで実行’の手順を説明します。

①プログラムを停止したい行にテキストカーソルをおく。

②ツールバーの‘カーソルまで実行’ボタンをクリックする。



“E7”はその他にも，ステップ実行，ブレークポイント設定，メモリリード・ライト，I/O リード・ライト，C のラベルによる変数のウォッチ機能など，本格的なデバッグに必要な機能を十分に備えています。詳細については，ヘルプ又はユーザーズマニュアルをご覧下さい。

“E7”でダウンロードした時点で‘H8/3687’のフラッシュメモリにプログラムが書き込まれていますので，“E7”を接続しないで TK-3687 の電源をオンにすると書き込まれたプログラムを実行します。

2004-2005 Toyo-linx,Co.,Ltd.

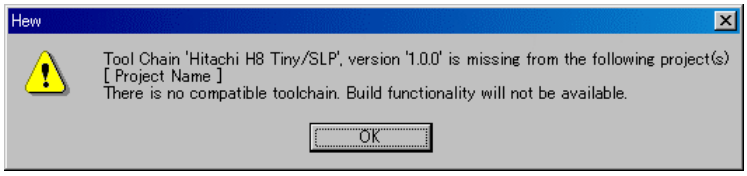
付録-4 HEW をよりよい環境にするために

HEW を使用する上で、知っていれば便利なことや役に立つ情報です。

2004-2005 Toyo-linx,Co.,Ltd.

Topics-1　旧バージョンで作成したプロジェクトの開き方

古いバージョンの HEW で作成したプロジェクトを新しいバージョンの HEW で開こうとした場合、次のダイアログが出てプロジェクトとして開けない場合があります。

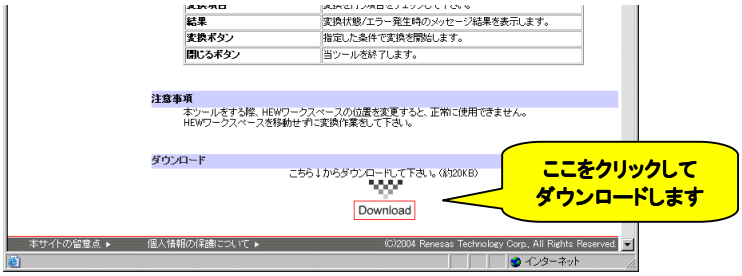


旧バージョンのプロジェクトはツールチェーンオプションコンバータ（Toc_ip）を使用する事で開けるようになります。以下に Toc_ip での変換手順を示します。

1．Toc_ip をダウンロードする

Toc_ip は Renesas の web ページからダウンロードする事ができます。

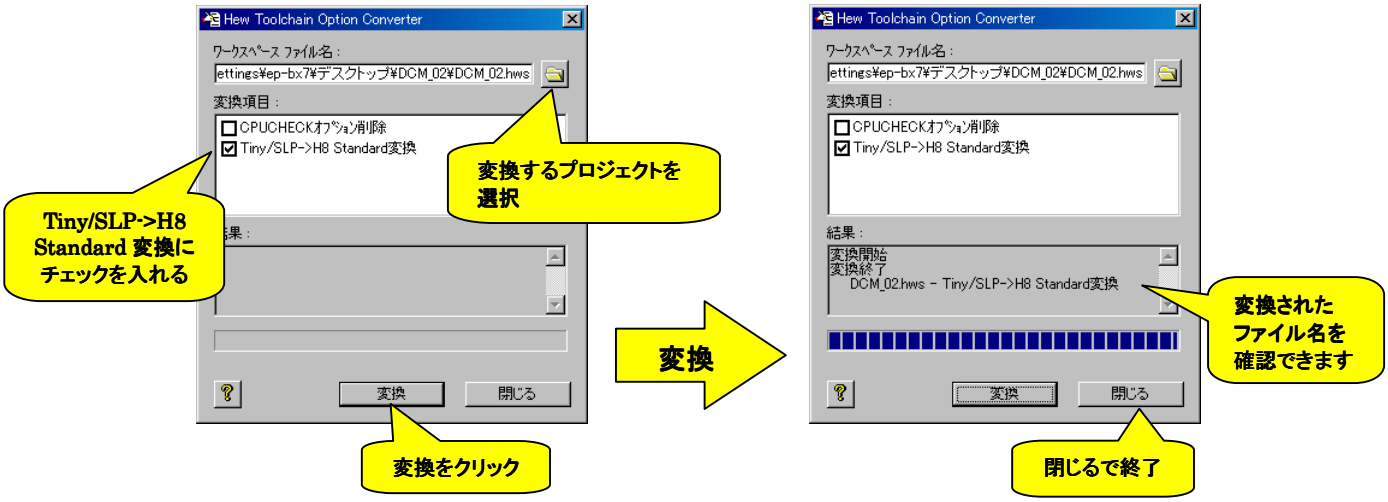
URL ： http://www.renesas.com/jpn/products/mpumcu/tool/crosstool/support_tool/hew_conv.html
（2004.11 現在）



2．Toc_ip を起動してプロジェクトを変換する

ダウンロードしたファイルは Zip 形式です。解凍したフォルダの中にプログラム“Toc.exe”があるのでダブルクリックして起動します。

起動すると次の画面が出るので、変換するワークスペースファイルを指定し、変換項目の“Tiny/SLP->H8 Standard 変換”にチェックを入れます。最後に“変換”ボタンをクリックすれば変換終了です。



尚、ワークスペースが移動されていると変換できませんので、必ず元のワークスペース（プロジェクト立ち上げ時に指定したディレクトリ）で変換を行なって下さい。

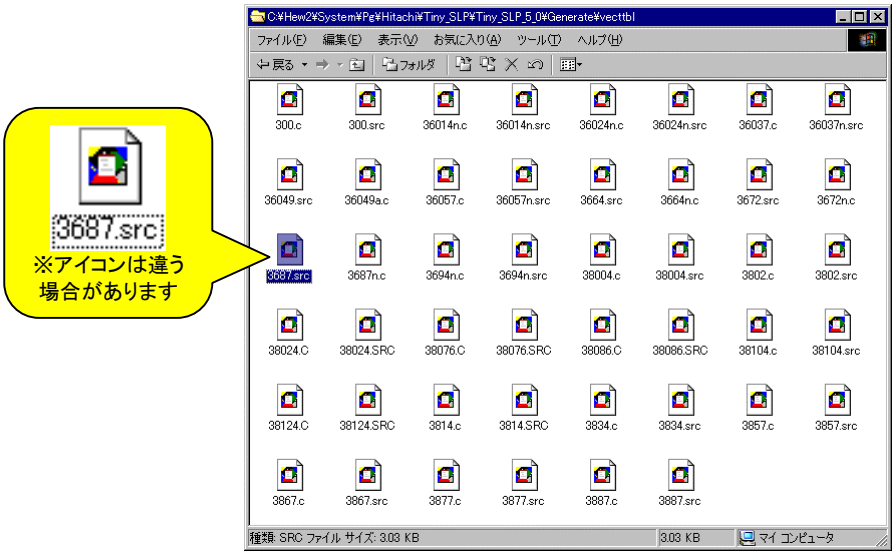
Topics-2　vecttbl.src の置換作業を省略する

『6 章・プログラムの作成と確認－10（P.15）』でプロジェクト立ち上げ時に vecttbl.src の置換作業を行なっていましたが、HEW の元ファイルを変更する事で以後の置換作業を無くす事ができます。以下に元ファイルの変更方法を示します。

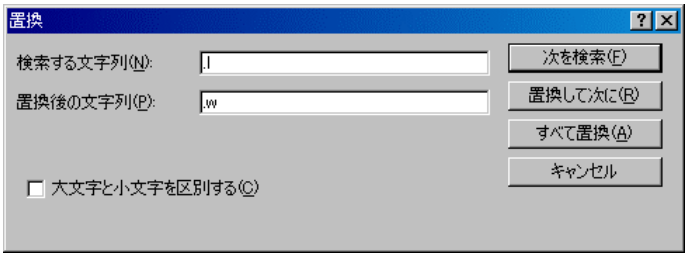
1．次のファイルをエディタで開いて下さい。

C:¥Hew2¥System¥Pg¥Hitachi¥Tiny_SLP¥Tiny_SLP_5_0¥Generate¥vecttbl¥3687.src

※下線（___）の箇所は HEW をインストールした先に合わせます。



2．開いたら以前行なった置換作業同様、‘.I’ を ‘.w’ に置換します。



※この時、置換する文字以外は変更しないで下さい。ここで行なった変更は以後自動生成される vecttbl.src 全てに反映されます。

3．変換し終えたら、保存をして終了です。

これで以後プロジェクト立ち上げ時に自動生成される vecttbl.src は全て ‘.w’ になります。

2004-2005 Toyo-linx,Co.,Ltd.

Topics-3　絶対アドレスの指定とリストファイルの出力設定

ハイパーH8 使用時に、ブレイク実行や変数参照など絶対アドレスを知りたい場合がでてきます。ここでは絶対アドレスの指定とリストファイルの出力設定を記します。

■　絶対アドレスの指定

通常、そのままプログラムを入力するとアドレスは相対アドレスで指定されています。このままでリストファイルを出力してしまうと全て相対でアドレスが記されてしまいます。そこで各セクションの先頭に次の擬似命令を入力します。

例）P セクションの場合

.section　　P,code,locate=H'EA00

'locate=アドレス' を入力することでそのセクションは絶対アドレス指定となります。

```
479                  25  ;***********************************
480                  26  ;       メインプログラム
481                  27  ;***********************************
482                  28
483  0000            29  _main:
484  0000 5C0000D8   30          bsr     INITTV:16
485  0004 5C0000DE   31          bsr     INITPIO:16
486                  32
487  0008 79030000   33          mov.w   #WORK_AREA,r3
488  000C 7901000B   34          mov.w   #WORK_AREAE-WORK_AR
489  0010 1588       35          xor.b   r0l,r0l
490  0012 5C0000BC   36          bsr     FILL:16
```

【 相対アドレス 】

相対アドレスで指定されているのでプログラムアドレスが相対値になっている

```
479                  25  ;***********************************
480                  26  ;       メインプログラム
481                  27  ;***********************************
482  EA00            28      .section    P,code,locate=H'EA00
483                  29
484  EA00            30  _main:
485  EA00 5C0000D8   31          bsr     INITTV:16
486  EA04 5C0000DE   32          bsr     INITPIO:16
487                  33
488  EA08 79030000   34          mov.w   #WORK_AREA,r3
489  EA0C 7901000B   35          mov.w   #WORK_AREAE-WORK_AR
490  EA10 1588       36          xor.b   r0l,r0l
```

【 絶対アドレス 】

locate で絶対アドレスを指定しているのでプログラムアドレスが絶対値になっている

■　リストファイルの出力設定

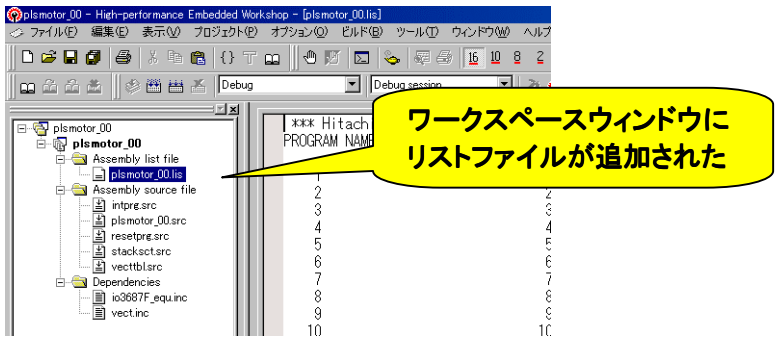
次にリストファイルの出力設定を行ないます。

1. メニューバーから、"オプション → H8 Tiny/SLP"を選択
2. "Assembly"タブをクリックし、'Category:' プルダウンメニューから"List"を選択
3. "Generate list file"ボックスにチェックを入れ、"OK"をクリック
4. ビルドする

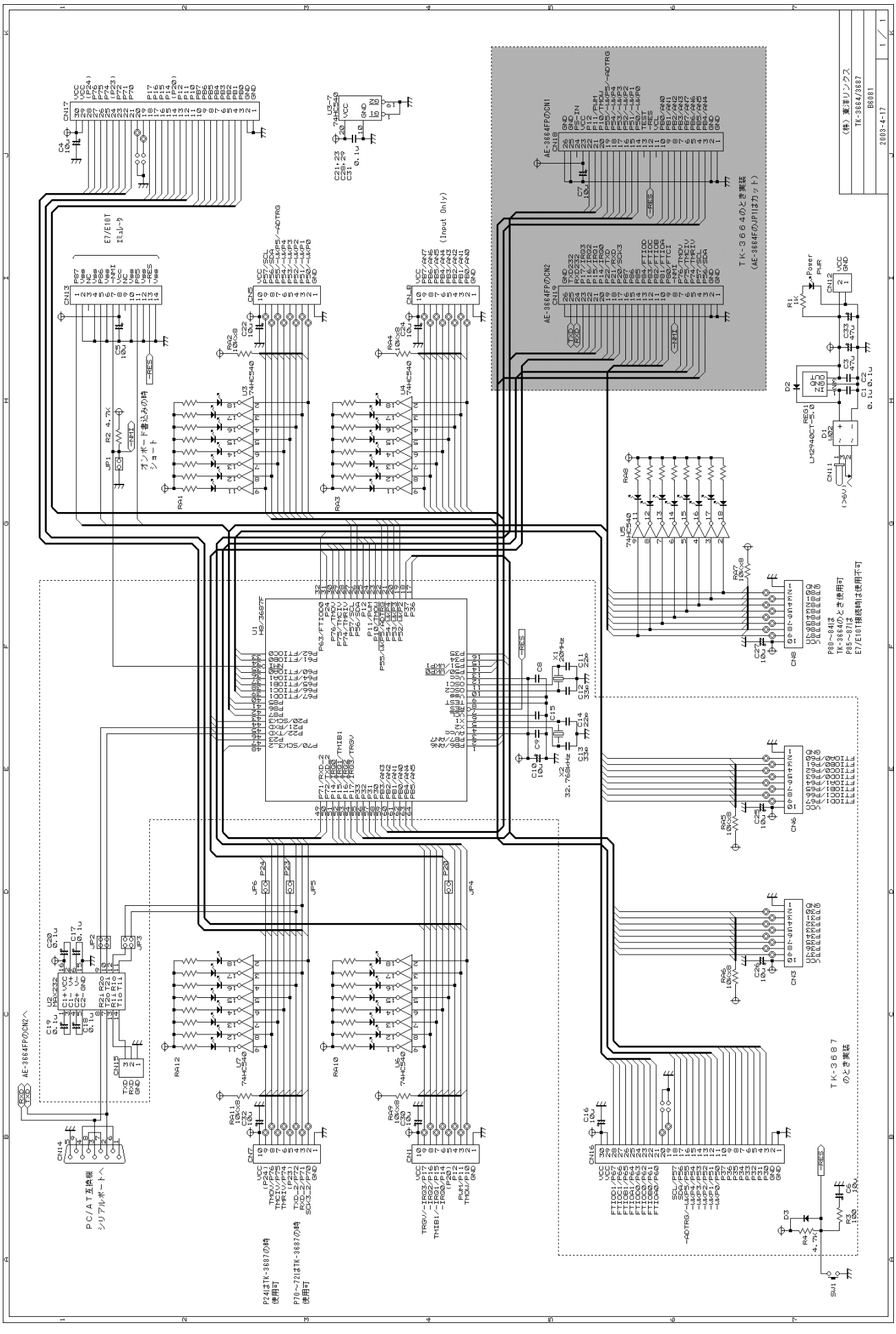
上記作業で選択しているセッションのフォルダ（例えば Debug Session なら Debug フォルダ）にリストファイル（.lis）が生成されます。次にすぐ参照できるようにプロジェクトにファイルを追加しておきましょう。

5. メニューバーから、"プロジェクト → ファイルの追加"を選択
6. 参照したいリストファイル（.lis ファイル）を選択して"追加"をクリック

するとワークスペースウィンドウに選択したリストファイルが追加されます。リストを参照したい時はワークスペースウィンドウのリストファイルをダブルクリックすれば開くことができます。



2004-2005 Toyo-linx,Co.,Ltd.



2004-2005 Toyo-linx,Co.,Ltd.

TK-3687部品表	台数:	1

	部品番号	型名，規格	メーカ	使用数	数量	備考
1	U1	H8/3687F (Flat)	ルネサス	1	1	実装済み
2	U2	MAX232ACSE (Flat)	MAXIM	1	1	実装済み
3	U3,4,~~5~~,6,7	74HC540 (Flat)		4	5	実装済み(U5を除く)
4						
5	REG1	LM2940CT-5.0	NS	1	1	
6						
7	X1	20MHz		1	1	水晶又はセラロック
8	X2	32.768kHz		1	1	
9	D1	W02		1	1	
10	D2	D1N20	新電元	1	1	REG1保護
11	D3	1SS133-T72	ROHM	1	1	
12	P50-57,PB0-7 ~~P80-87~~,10-17,P20 P70-76,P23,P24	HLMP-6001#A04	HP	8 ※1	10 ※1	連結LEDモジュール P80-P87は実装しません
13	PWR(LED)			1	1	パワーオン表示
14						
15	R1	4.7k～10kΩ (茶黒橙金)		1	1	PWR LED の輝度で調整
16	R2,4	4.7kΩ (黄紫赤金)		2	2	
17	R3	100Ω (茶黒茶金)		1	1	
18	RA1,3,10,12	M9-1-561～102J	BI	4	4	560～1kΩ LEDの輝度で調整
19	RA2,4,5,6,7,9,11	M9-1-103J～473J	BI	7	7	10k～47kΩ
20						
21	C~~11~~,14	22pF		1	2	※2
22	C~~12~~,13	33pF		1	2	※2
23	C1,2,8,9,15,21 23,~~28~~,29,31	0.1μF (積セラ)		9	10	C28は実装しません
24	C17,18,19,20	0.1μF		4	4	
25	C4,5,6,7,10,16,22 24,25,26,27,30,32	10μF/16V (電解)		13	13	
26	C3,33	47～100μF/16V (電解)		2	2	
27						
28	SW1	SKHHAK/AM/DC	ALPS	1	1	
29	JP1,4-6	2pin		4	8	※3
30	CN11	DCジャック(2.1φ)		1	1	
31	CN14	D-Sub9pin		1	1	メス,ライトアングル
32	CN16,17	Box,30pin		0	2	※4
33	CN1,3,5,6,7,8 CN-B	B10P-SHF-1AA	JST	0	7	※4
34	CN12	B2P-SHF-1AA	JST	0	1	※4(5V供給用)
35	CN13	Box,14pin,ライトアングル		1	1	E7接続用
36	ゴム足	B-21	タカチ	4	4	
37						
38	ベース基板	B6081(TK-3664/3687)	東洋リンクス	1	1	
39						

・相当品を使用する場合があります
※1 8連の時は1/2個
※2 X1セラロック使用時はC11,C12実装せず
※3 丸ピンソケットを2pin×4ヶに切断して使用
※4 付属していません

2004-2005 Toyo-linx,Co.,Ltd.

付録-7 無償評価版コンパイラ

ルネサステクノロジは現在，High-performancr Embedded Wprkshop V. 4（HEW4）に対応した無償評価版コンパイラを公開しています（Tiny/SLP 無償版コンパイラはなくなりました）。無償評価版コンパイラは，はじめてコンパイルした日から 60 日間は製品版と同等の機能と性能のままで試用できます。61 日目以降はリンクサイズが 64K バイトまでに制限されますが，H8/3687 はもともとアクセスできるメモリサイズが 64K までバイトなので，この制限は関係ありません。また，無償評価版コンパイラは製品開発では使用できないのですが，H8/300H Tiny シリーズ（H8/3687 も含まれる）では許可されています。この項では無償評価版コンパイラのダウンロードからインストール，プログラムの入力とビルド，ハイパーH8 によるダウンロードと実行までを説明します。

1. HEW の入手

TK-3687mini のプログラミングで使用するアセンブラは，High-performance Embedded Workshop V.4（HEW4）に対応した無償評価版コンパイラに含まれており，株式会社ルネサステクノロジのホームページよりダウンロードします。ダウンロードサイトの URL は以下の通りです。

株式会社ルネサステクノロジ
http://www.renesas.com/jpn/

無償評価版コンパイラ
ダウンロードサイト
http://www.renesas.com/jpn/products/mpumcu/tools/download/h8c/index.html

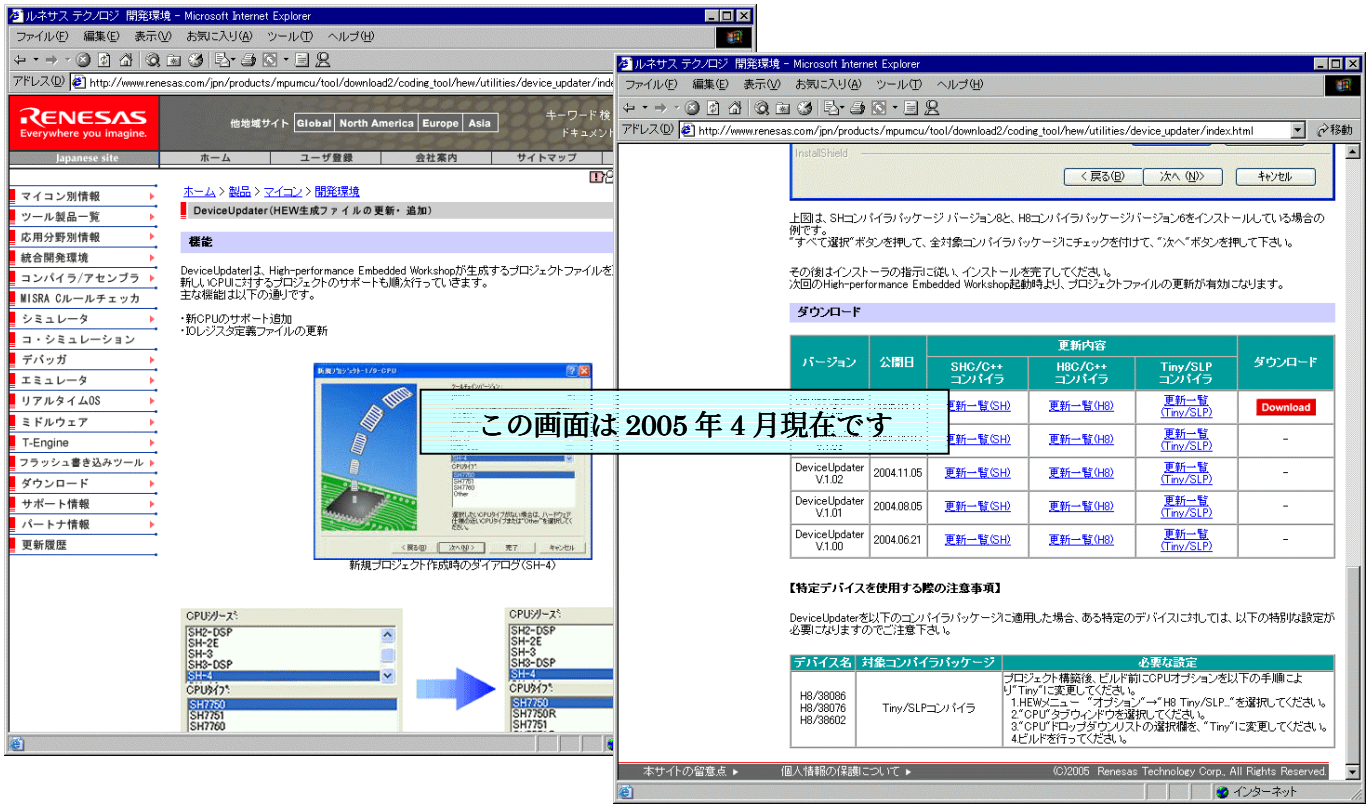
ダウンロードサイトの下の方にある「ダウンロードのページへ」をクリックして下さい。次のページで必須事項を入力してダウンロードを開始します。ダウンロード先はデスクトップにすると便利です。全部で 69．4MByte になりますので，ADSL か光回線でないと，かなり大変なのが実情です。‘ h8cv601r00．exe ’というファイルがダウンロードされます。

最新版の HEW を手に入れましょう

HEW はときどきバージョンアップされます。HEW はルネサステクノロジのマイコン全てに対応しているため，H8 シリーズはもとより，R8 シリーズや SH シリーズなど，対応するマイコンが増えるとそのたびにマイナーチェンジされるようです。また，その際に報告されていた不具合を一緒に修正することもあります。

それで，ルネサステクノロジのホームページは定期的にのぞいてみることをおすすめします。特にデバイスアップデータの情報は要注意です。

ところで，ここでダウンロードした無償評価版コンパイラには不具合があることが報告されています。それで，ルネサステクノロジが公開しているデバイスアップデータを使用して不具合を修正します。デバイスアップデータは下記の URL のサイトからダウンロードできます。



デバイスアップデータ

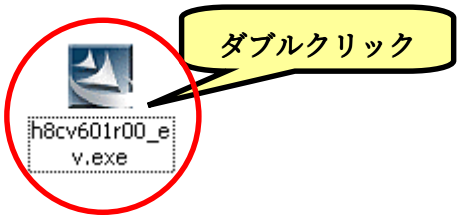
ダウンロードサイト

http://www.renesas.com/jpn/products/mpumcu/tool/download2/coding_tool/hew/utilities/device_updata/index.html

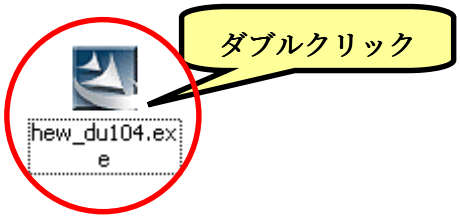
ページの下の方にある「Download」をクリックしてください。ダウンロード先はデスクトップにすると便利です。全部で 3．55MByte になります。‘ hew_du104．exe ’というファイルがダウンロードされます。

2．HEW のインストール

ダウンロード先をデスクトップにした場合で説明します。ダウンロードした‘ h8cv601r00．exe ’をダブルクリックしてください。すると，インストールが始まります。画面の指示に従ってインストールしてください。



次に，無償評価版コンパイラをアップデートします。ダウンロードした‘ ｈｅｗ_du104．exe ’をダブルクリックしてください。すると，インストールが始まります。画面の指示に従ってインストールしてください。



3. メモリマップの確認

HEW を使うときのコツの一つは，メモリマップを意識する，ということです。プログラムがどのアドレスに作られて，データはどのアドレスに配置されるか，ちょっと意識するだけで，HEW を理解しやすくなります。ハイパーH8 を使うときのメモリマップは次のとおりです。

<table>
<tr><td>0000 番地

DFFF 番地</td><td colspan="3">モニタプログラム
‘ ハイパーH8 ’</td><td>ROM/フラッシュメモリ
（56K バイト）</td></tr>
<tr><td>E000 番地
E7FF 番地</td><td colspan="3">未使用</td><td>未使用</td></tr>
<tr><td>E800 番地</td><td>VECTTBL
INTTBL</td><td>ベクタテーブル</td><td rowspan="3">ユーザ RAM エリア</td><td rowspan="3">RAM
（2K バイト）</td></tr>
<tr><td>E860 番地</td><td>ResetPRG
IntPRG</td><td>リセットプログラム
割り込みプログラム</td></tr>
<tr><td>EA00 番地

EFFF 番地</td><td>P</td><td>プログラム領域</td></tr>
<tr><td>F000 番地
F6FF 番地</td><td colspan="3">未使用</td><td>未使用</td></tr>
<tr><td>F700 番地
F77F 番地</td><td colspan="3">I/O レジスタ</td><td>I/O レジスタ</td></tr>
<tr><td>F780 番地

FB7F 番地</td><td rowspan="2">B</td><td rowspan="2">未初期化データ領域
（変数領域）</td><td rowspan="3">ユーザ RAM エリア</td><td>RAM（1K バイト）
フラッシュメモリ書換え用
ワークエリアのため，
FDT と E7 使用時は，
ユーザ使用不可</td></tr>
<tr><td>FB80 番地

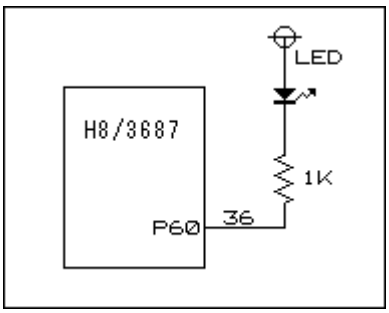
FD80 番地
FDFF 番地</td><td rowspan="3">RAM
（1K バイト）</td></tr>
<tr><td>Stack</td><td>スタック領域</td></tr>
<tr><td>FE00 番地
FF7F 番地</td><td colspan="3">ハイパーH8
ワークエリア</td></tr>
<tr><td>FF80 番地
FFFF 番地</td><td colspan="3">I/O レジスタ</td><td>I/O レジスタ</td></tr>
</table>

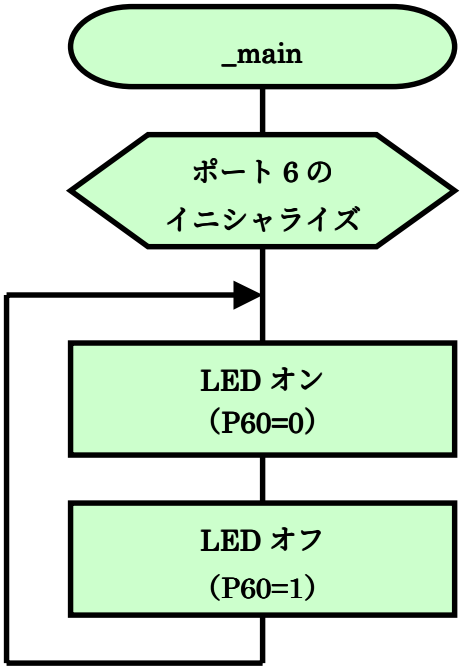
メモリマップのうちユーザ RAM エリアの部分だけが自由に使用できるエリアです。

2004-2005 Toyo-linx,Co.,Ltd.

4. プロジェクトの作成

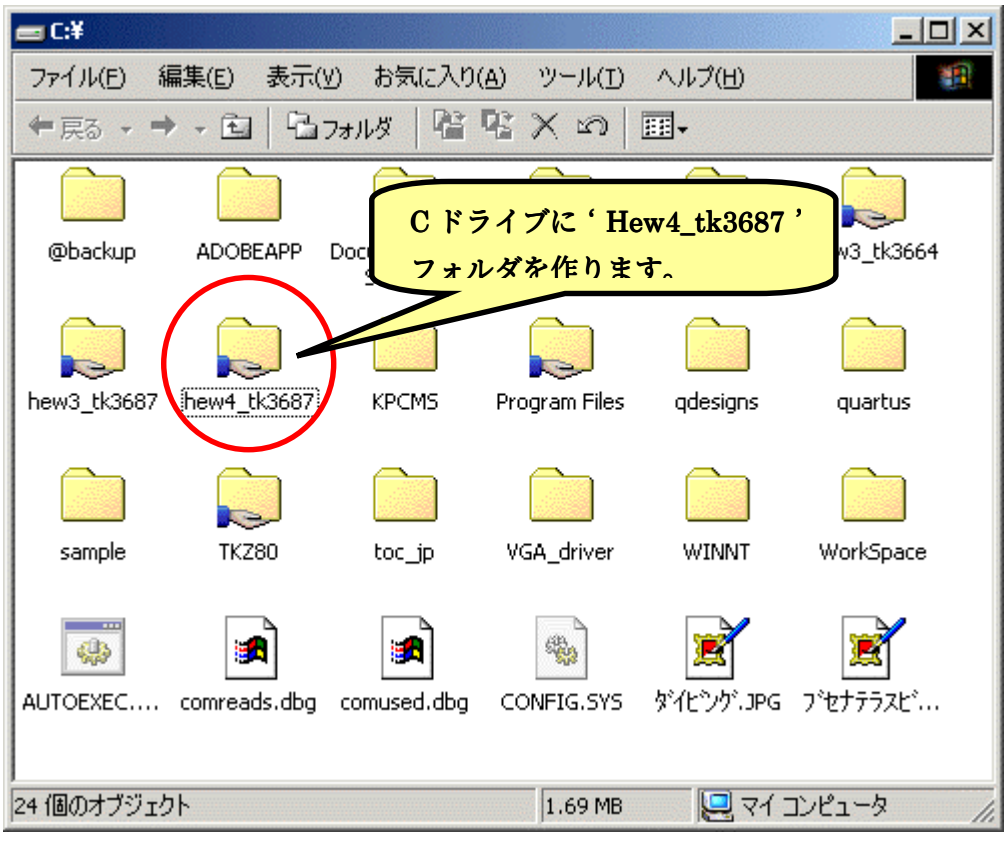
ここで作るプログラムは P60 につないだ LED を点滅させるというものです。回路図とフローチャートは次のとおりです。





HEW ではプログラム作成作業をプロジェクトと呼び, そのプロジェクトに関連するファイルは 1 つのワークスペース内にまとめて管理されます。通常はワークスペース，プロジェクト，メインプログラムには共通の名前がつけられます。この章で作るプロジェクトは‘ led ’と名付けます。以下に, 新規プロジェクト‘ led ’を作成する手順と動作確認の手順を説明します。

しかしその前に，HEW 専用作業フォルダを作っておきましょう。C ドライブに‘ Hew4_ｔｋ3687 ’を作ってください。このマニュアルのプロジェクトは全てこのフォルダに作成します。

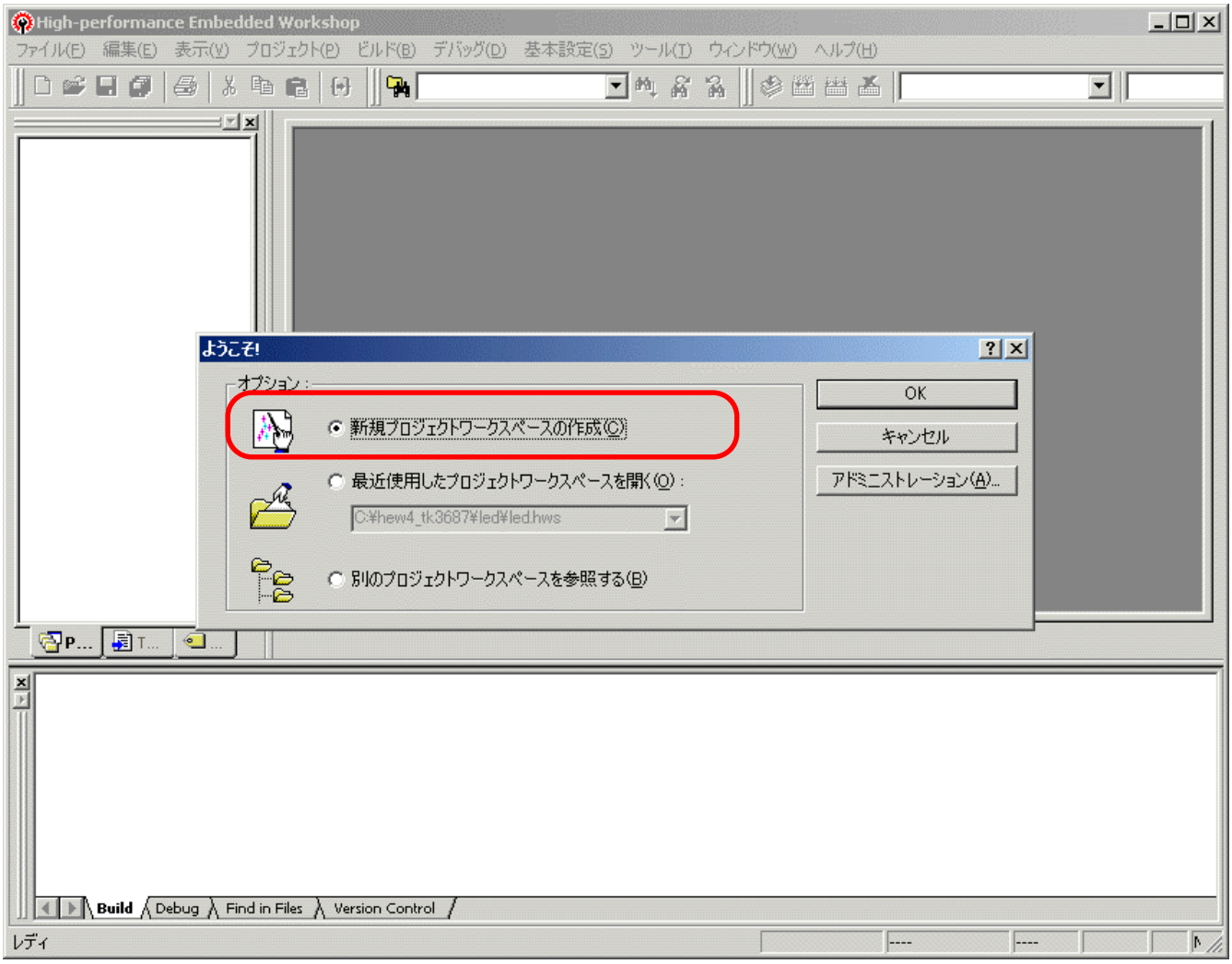


2004-2005 Toyo-linx,Co.,Ltd.

では，HEW を起動しましょう。スタートメニューから起動します。

スタート → プログラム → Renesas → High-performance Embedded Workshop → High-performance Embedded Workshop

HEW を起動すると下記の画面が現れるので，「新規プロジェクトワークスペースの作成」を選択して‘OK’をクリックします。



前に作ったプロジェクトを使うとき

その場合は,「ようこそ!」ダイアログで「最近使用したプロジェクトワークスペースを開く」を選択して‘OK’をクリックします。そのプロジェクトの最後に保存した状態で HEW

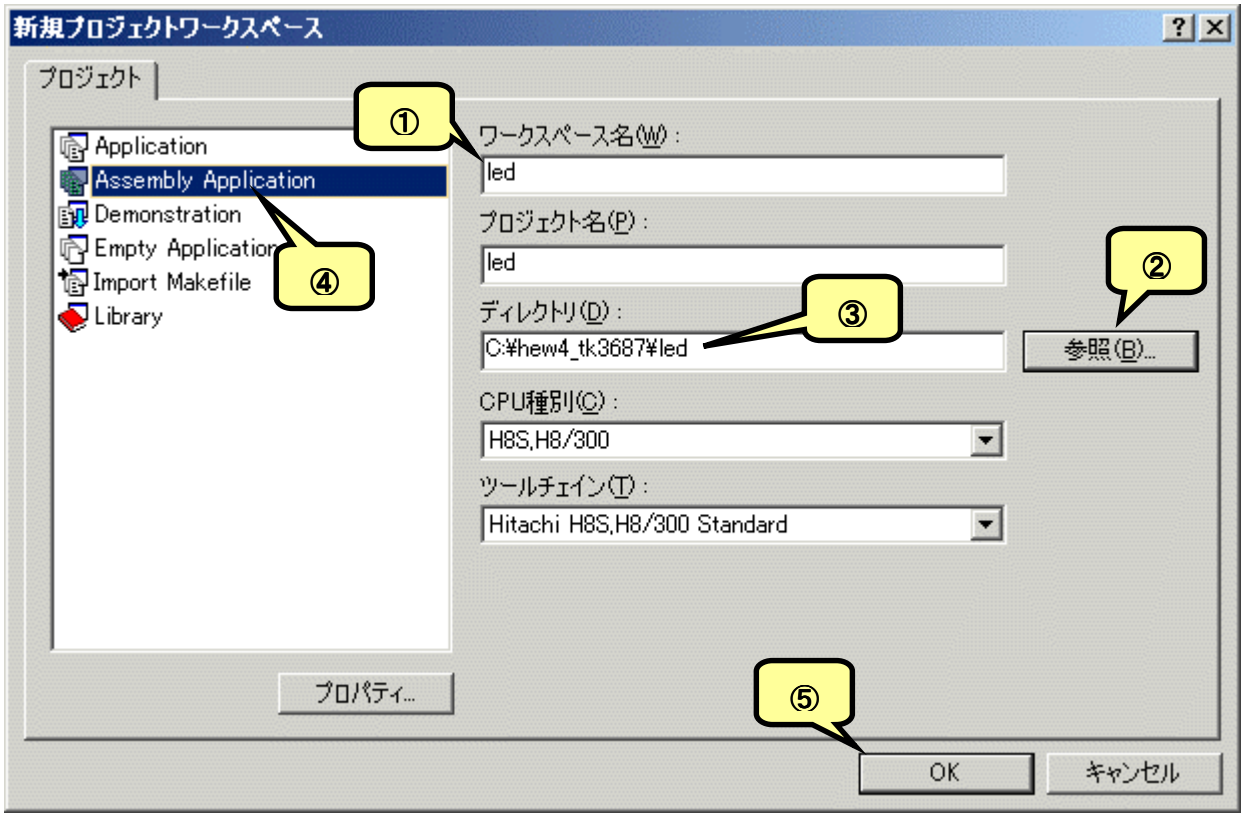
2004-2005 Toyo-linx,Co.,Ltd.

まず，①「ワークスペース名（W)」（ここでは ‘ led ’）を入力します。「プロジェクト名（P)」は自動的に同じ名前になります。

ワークスペースの場所を指定します。②右の「参照（B）…」ボタンをクリックします。そして，あらかじめ用意した HEW 専用作業フォルダ（ここでは Hew4_tk3687）を指定します。設定後,「ディレクトリ（D)」が正しいか確認して下さい。（③）

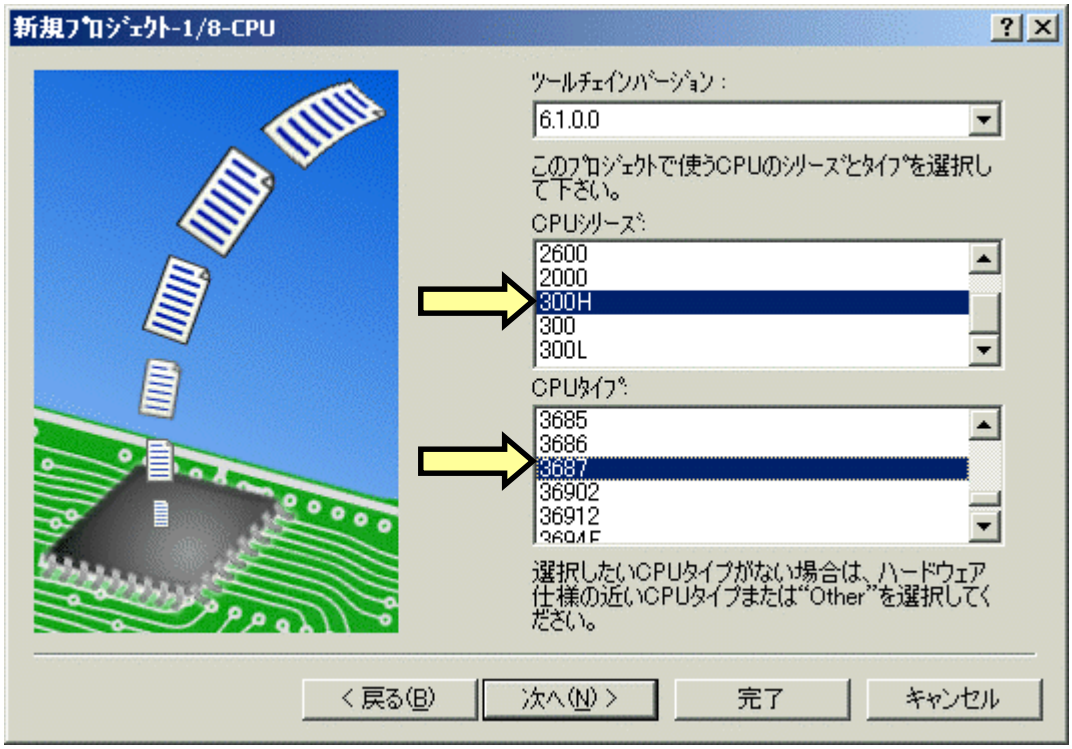
次にプロジェクトタイプを指定します。今回はアセンブラなので④「Assembly　Application」を選択します。

入力が終わったら⑤「OK」をクリックして下さい。

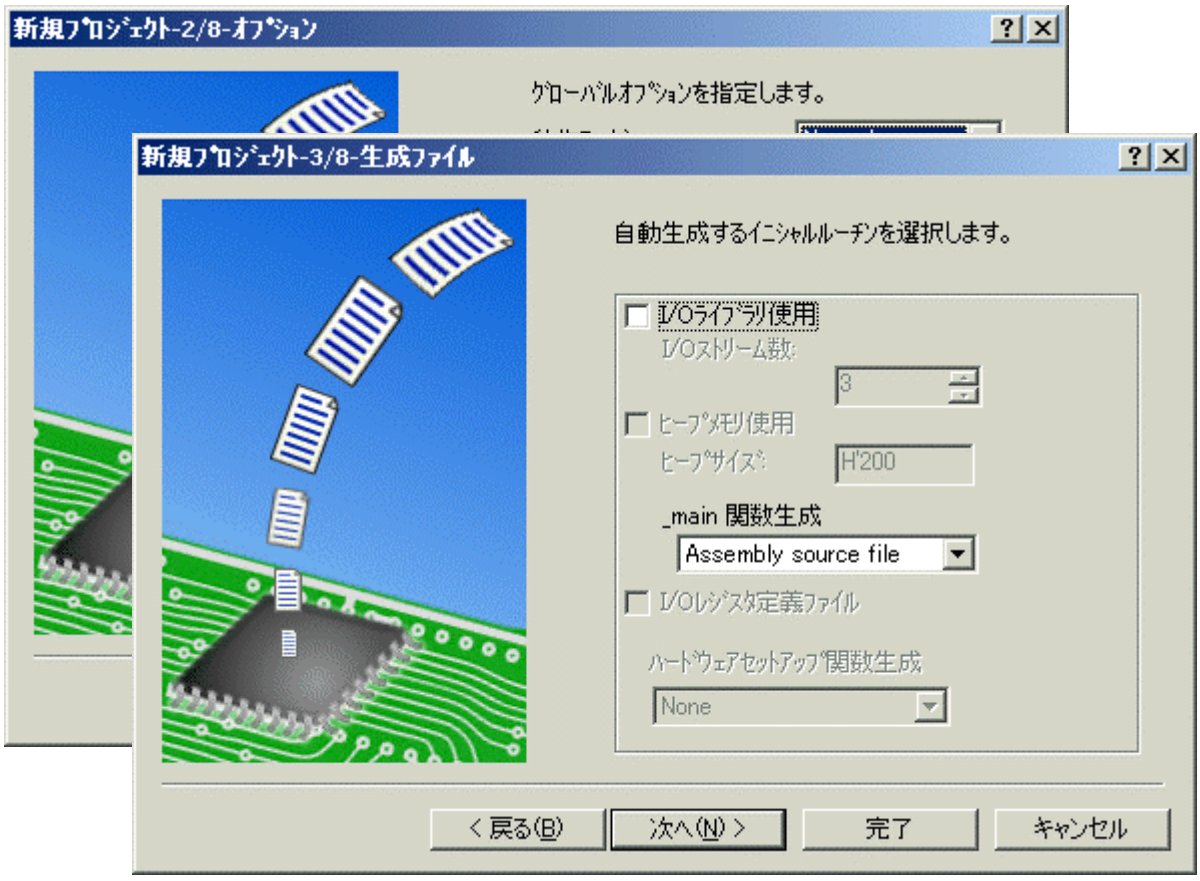


2004-2005 Toyo-linx,Co.,Ltd.

「新規プロジェクト－1／8－CPU」で，使用する CPU シリーズ（300H）と，CPU タイプ（3687）を設定し，「次へ（N）＞」をクリックします。

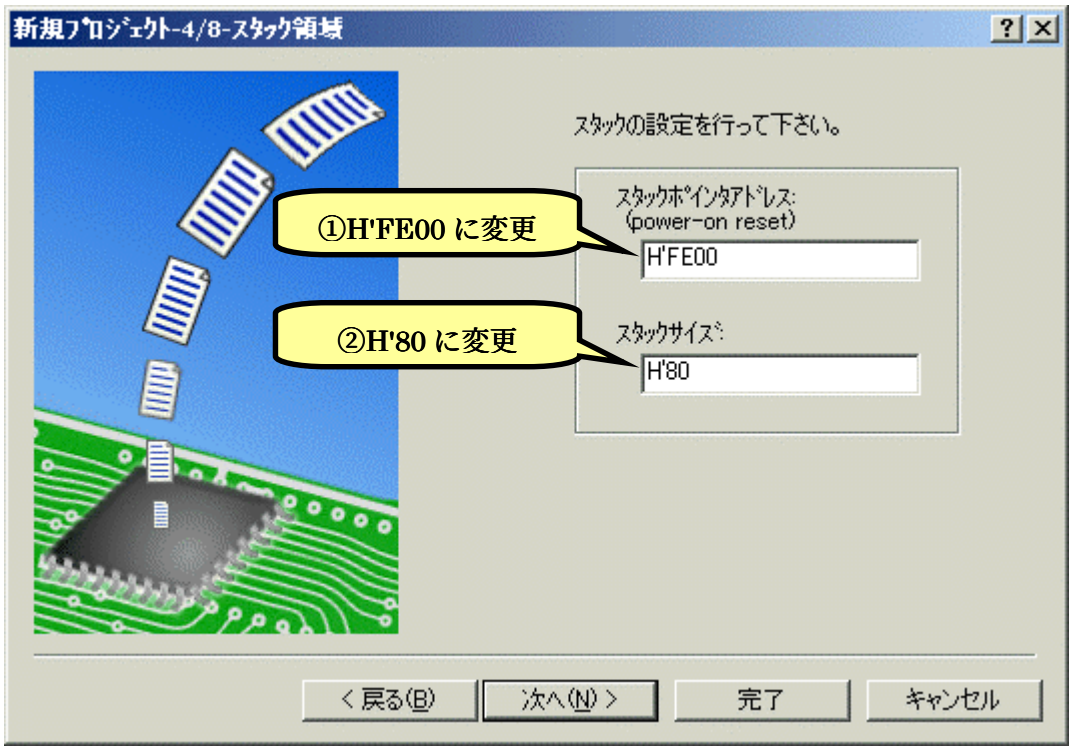


「新規プロジェクト－2／8－オプション」と「新規プロジェクト－3／8－生成ファイル」は変更しません。「次へ（N）＞」をクリックして順に次の画面に進みます。

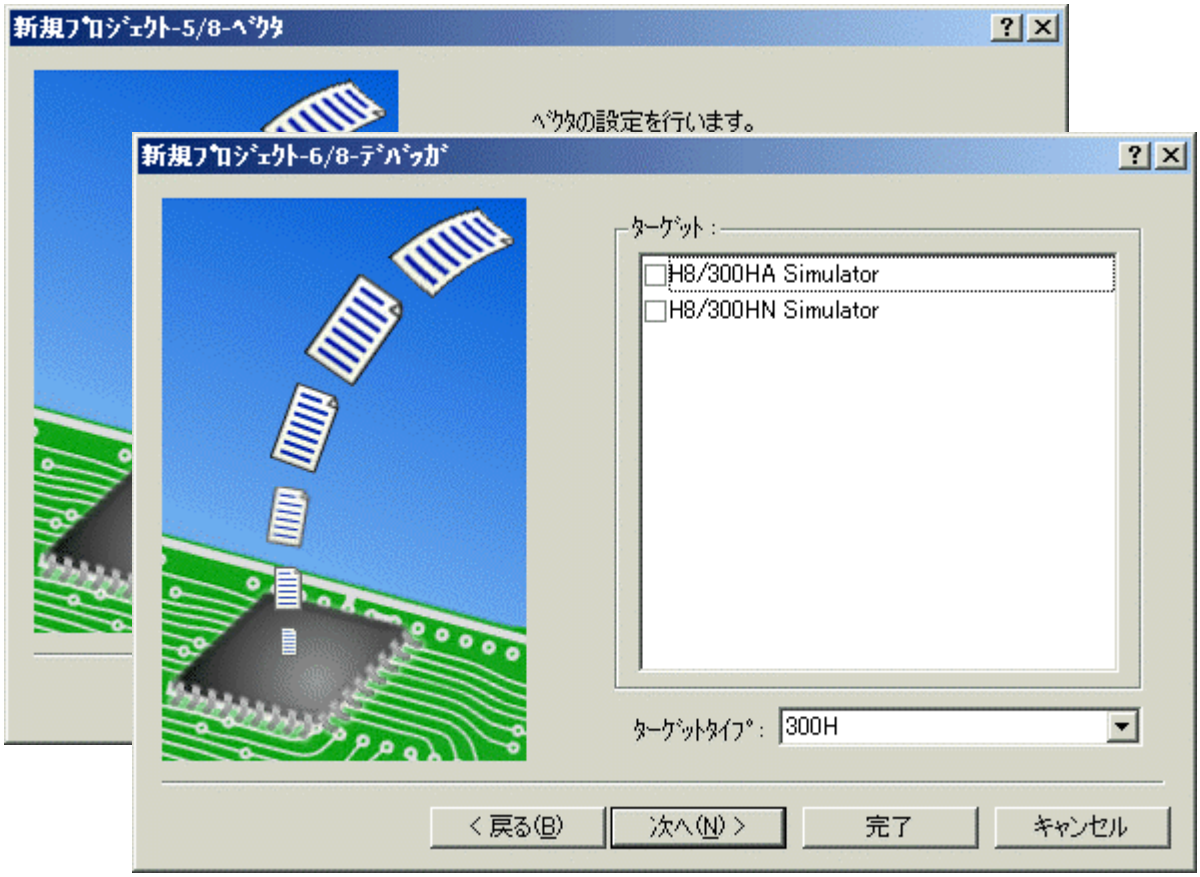


2004-2005 Toyo-linx,Co.,Ltd.

「新規プロジェクトの作成－4／8－スタック領域」でスタックのアドレスとサイズを変更します。ハイパーH8 を使用するので，①スタックポインタを H'FE00 に，②スタックサイズを H'80 にします。設定が終わったら「次へ（N）＞」をクリックします。（ハイパーH8 を使わないときは変更の必要はありません。）

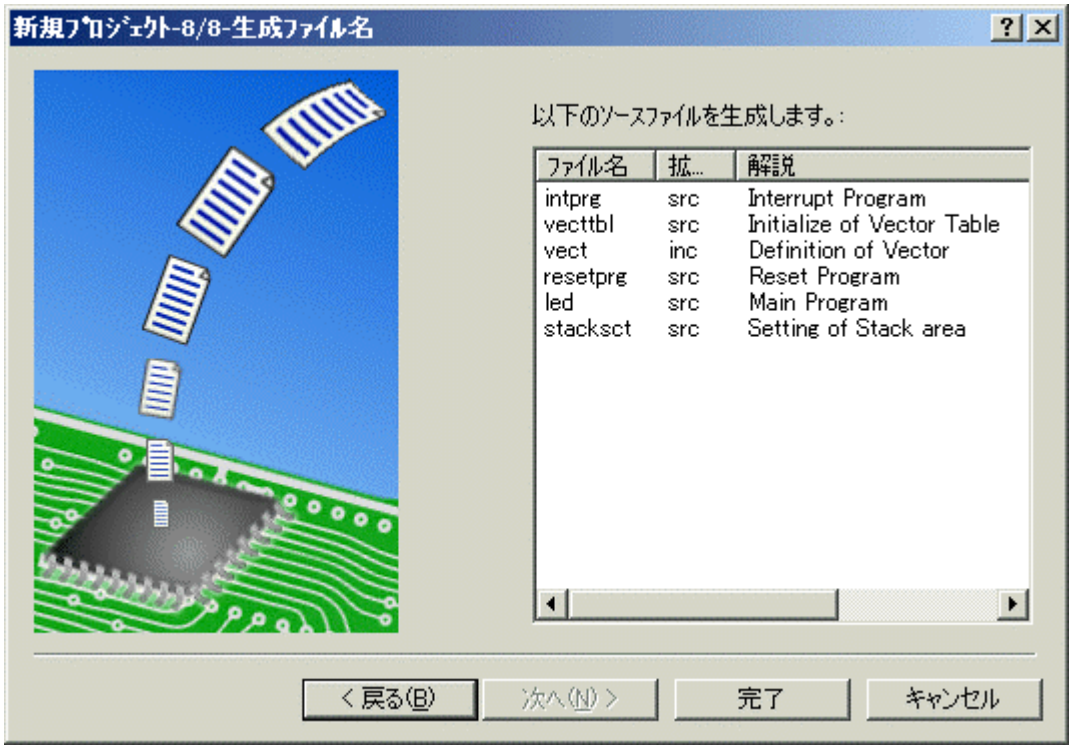


「新規プロジェクト－5／8－ベクタ」と「新規プロジェクト－6／8－デバッガ」は変更しません。「次へ（N）＞」をクリックして順に次の画面に進みます。

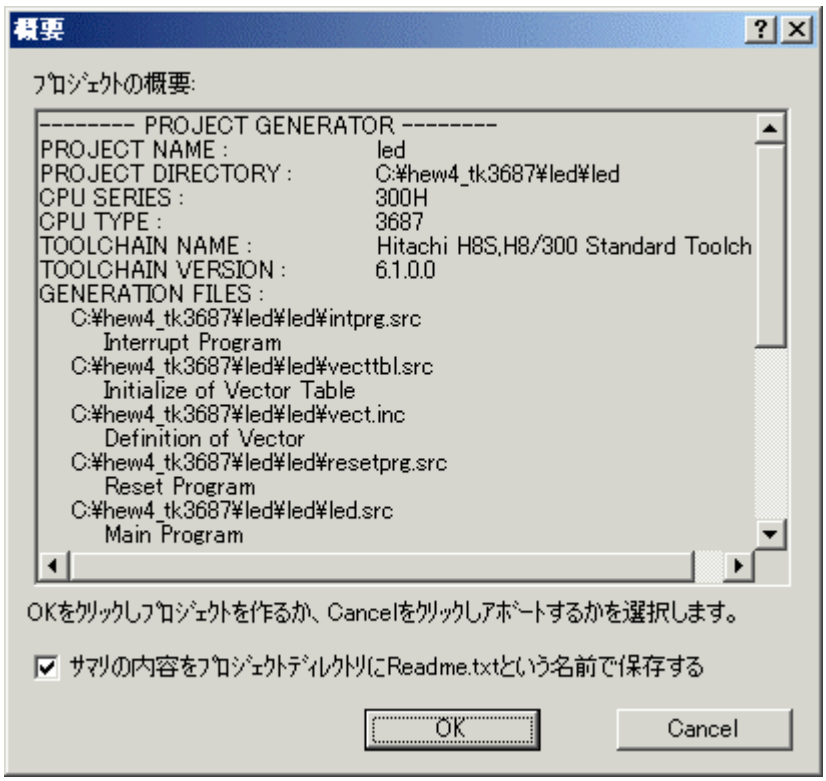


2004-2005 Toyo-linx,Co.,Ltd.

次は「新規プロジェクト－8／8－生成ファイル名」です。ここも変更しません。「完了」をクリックします。

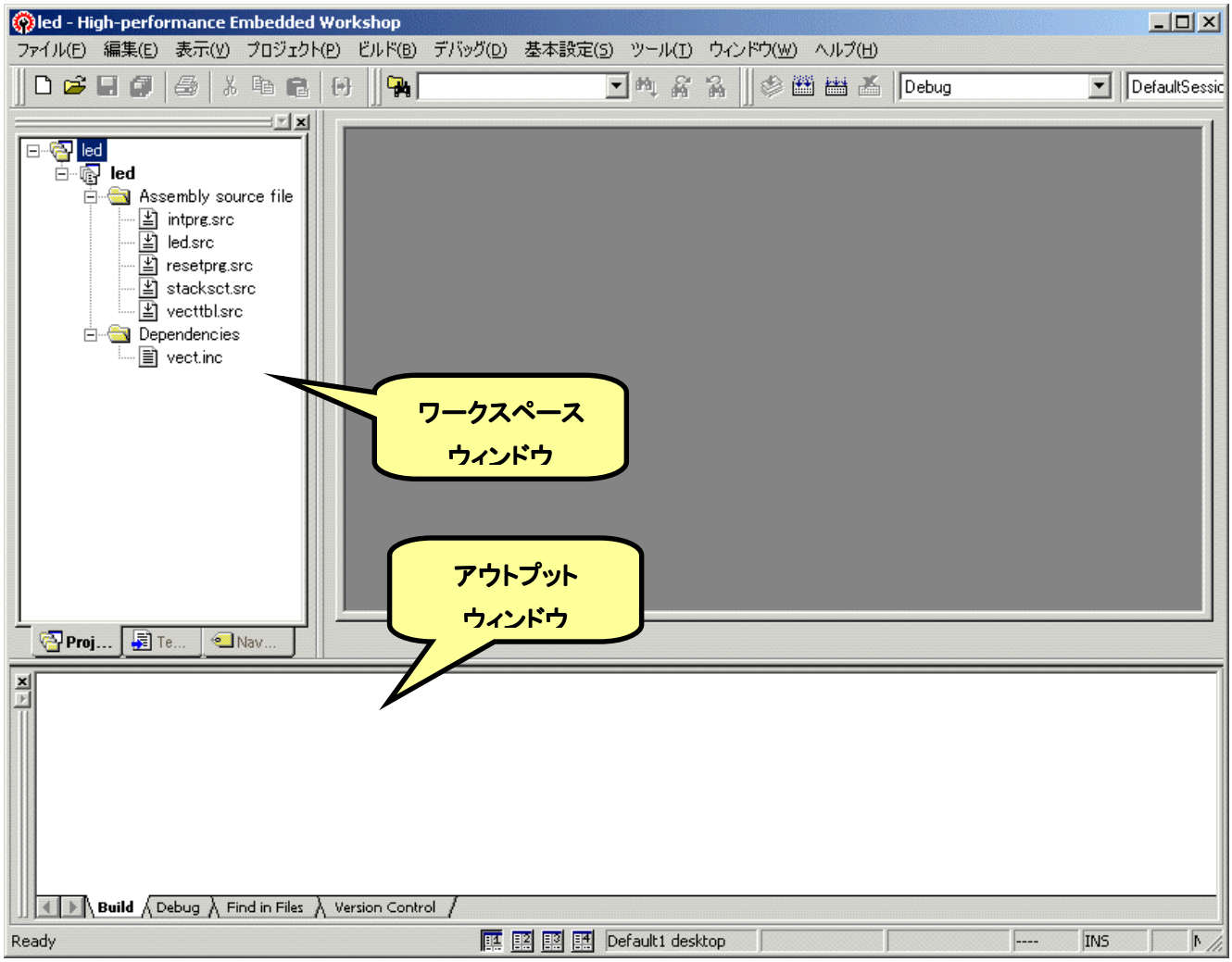


すると，「概要」が表示されるので「OK」をクリックします。



2004-2005 Toyo-linx,Co.,Ltd.

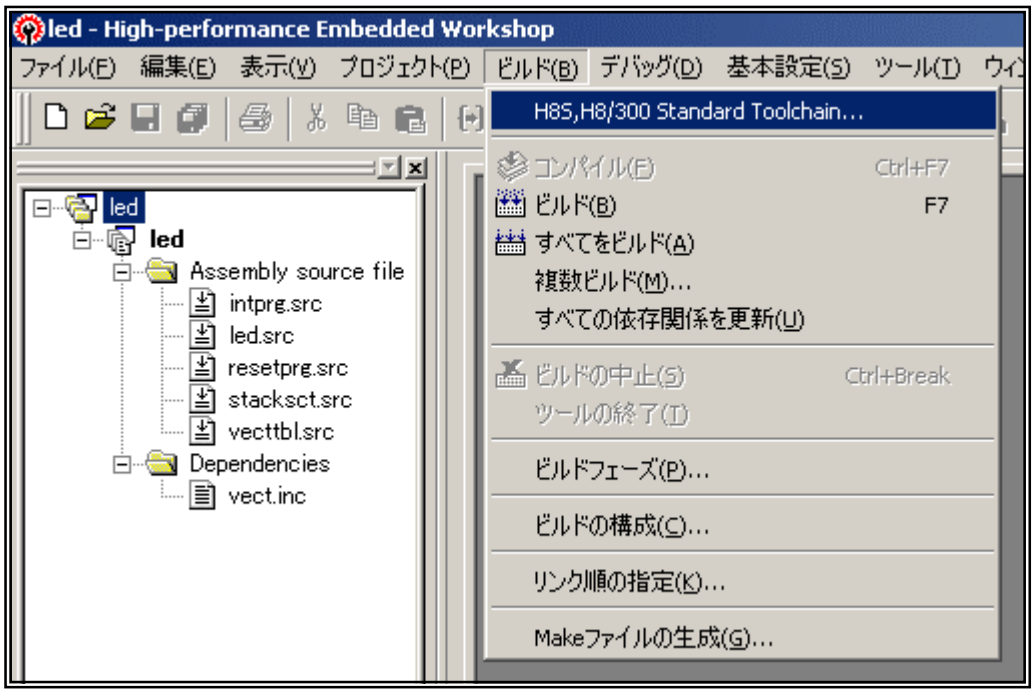
これで，プロジェクトワークスペースが完成します。HEW はプロジェクトに必要なファイルを自動生成し，それらのファイルは左端のワークスペースウィンドウに一覧表示されます。



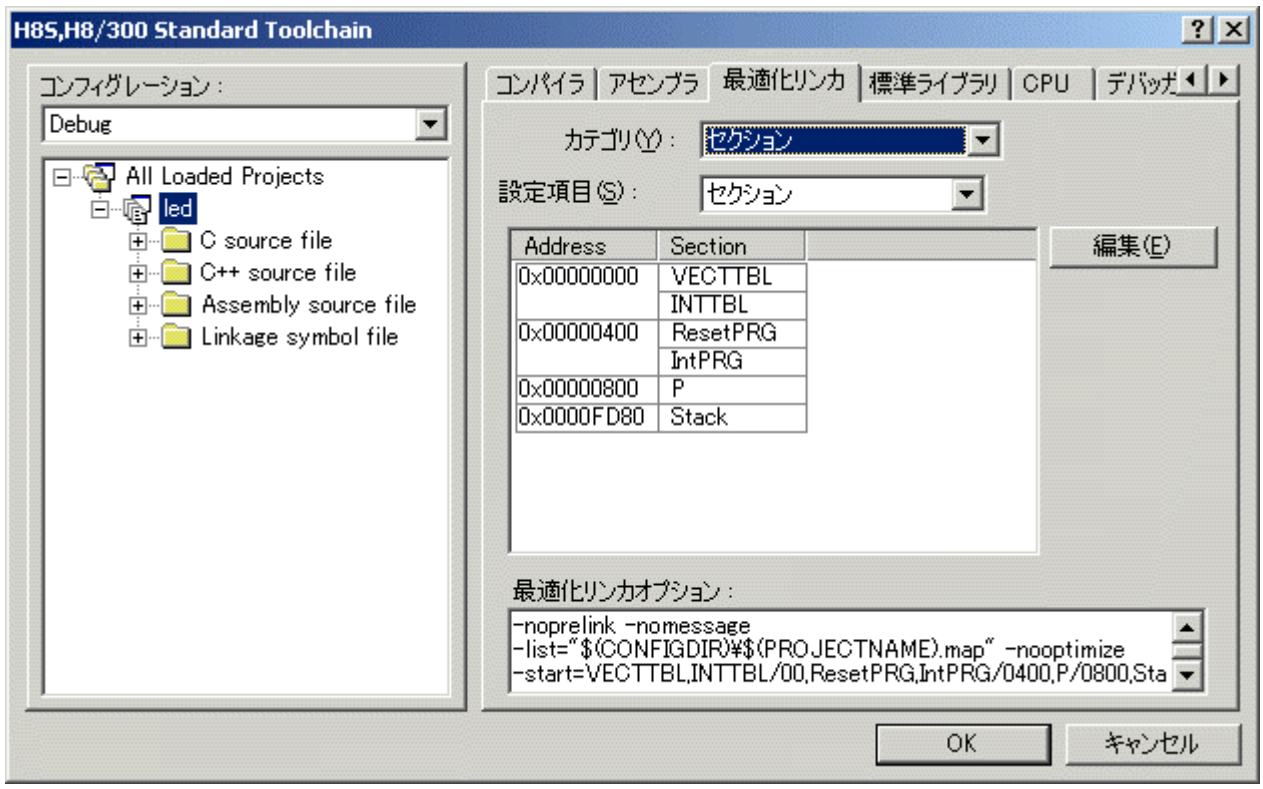
2004-2005 Toyo-linx,Co.,Ltd.

さて，これでプロジェクトは完成したのですが，ハイパーH8 を使うためにセクションを変更してプログラムが RAM 上にできるようにします。(当然ながら，ハイパーH8 を使わないときは変更する必要はなく，そのままで OK。)

下図のように，メニューバーから「H8S, H8/300 Standard Toolchain...」を選びます。

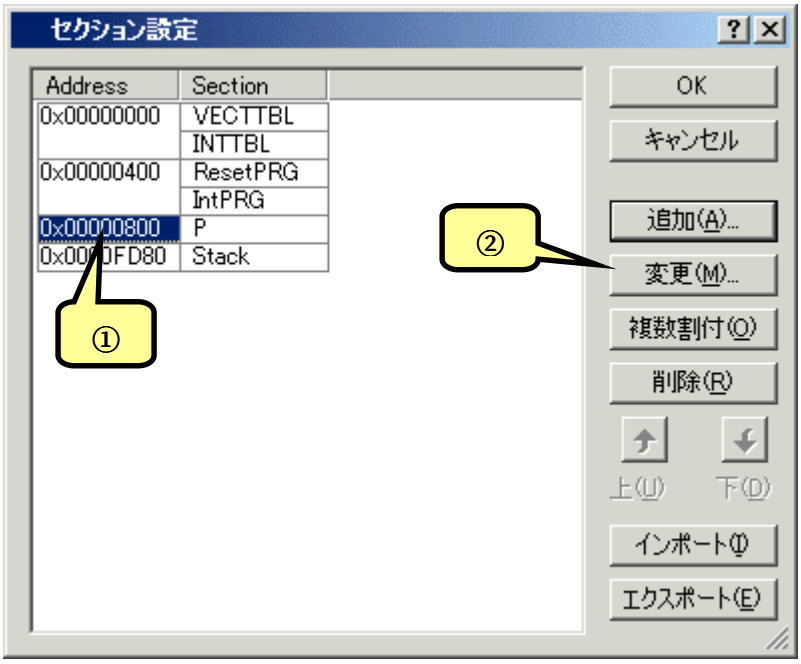


すると，「H8S, H8/300 Standard　Toolchain」ウィンドウが開きます。「最適化リンカ」のタブを選び，「カテゴリ（Y)」のドロップダウンメニューの中から「セクション」を選択します。すると，下図のような各セクションの先頭アドレスを設定する画面になります。「編集（E)」ボタンをクリックしてください。

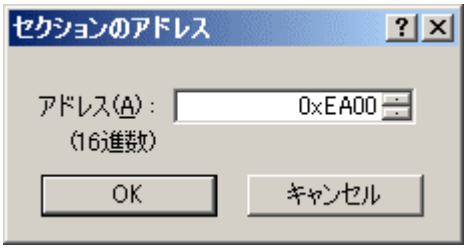


2004-2005 Toyo-linx,Co.,Ltd.

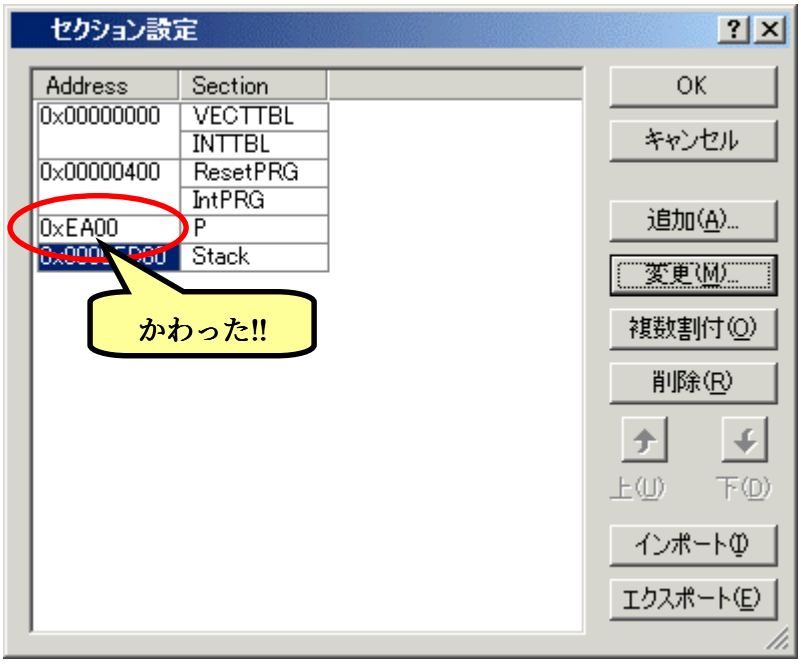
「セクション設定」ダイアログが開きます。それでは,「1. メモリマップの確認」で調べたメモリマップにあわせて設定していきましょう。最初に‘ P ’Section のアドレスを変更します。デフォルトでは 800 番地になっていますね。①‘ 0x00000800 ’というところをクリックして下さい。それから,②「変更（M）…」をクリックします。



そうすると,「セクションのアドレス」ダイアログが開きます。‘ P ’Section は EA00 番地から始まりますので, 右のように入力して‘ OK ’をクリックします。

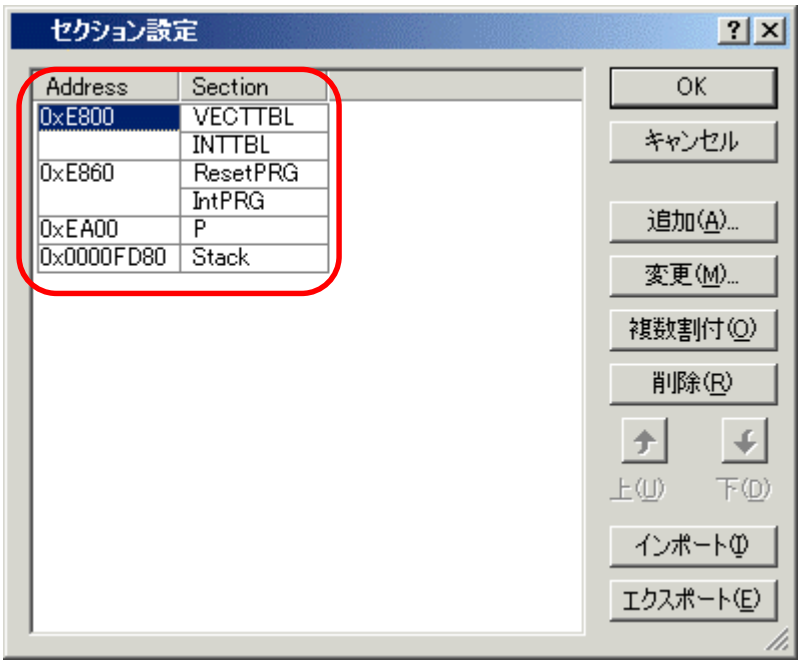


すると…



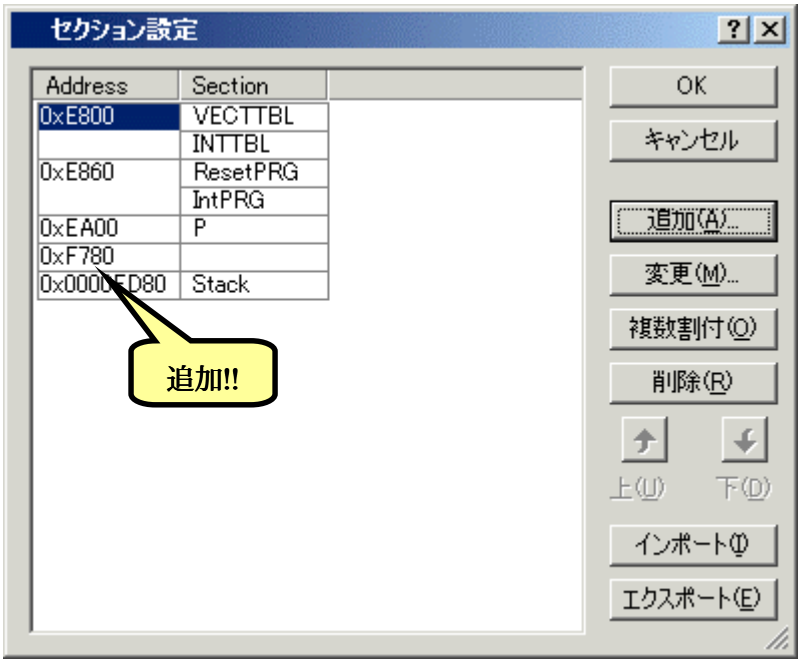
2004-2005 Toyo-linx,Co.,Ltd.

同じように, 他のセクションも変更しましょう。



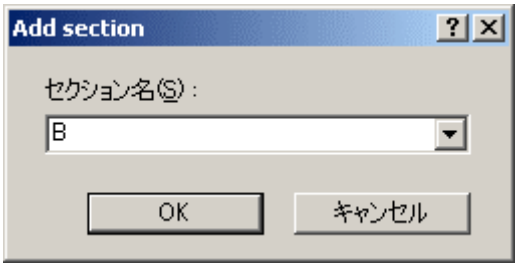
次は‘ B ’Section を追加します。どこでもかまわないのでアドレスをクリックして,「追加 (A) ...」をクリックして下さい。そうすると,「セクションのアドレス」ダイアログが開きます。‘ B ’Section は F780 番地から始まりますので，右のように入力して「OK」をクリックします。

Address に 0xF780 が追加されました。

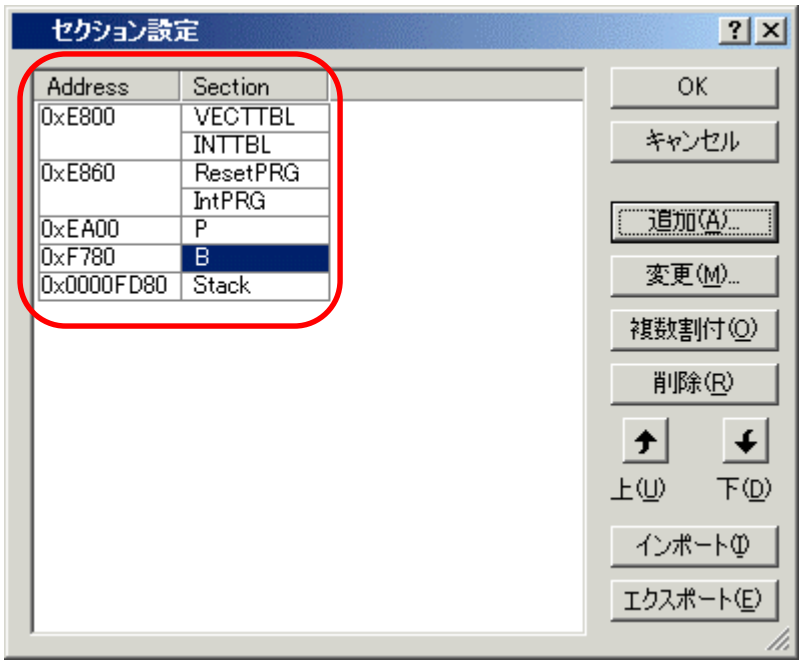


2004-2005 Toyo-linx,Co.,Ltd.

0xF780 番地の Section をクリックして，さらに‘ Add… ’をクリックして下さい。‘ Add Section ’ダイアログが開きます。‘ Section name ’のドロップダウンメニューの中から‘ B ’を選択し，「OK」をクリックします。

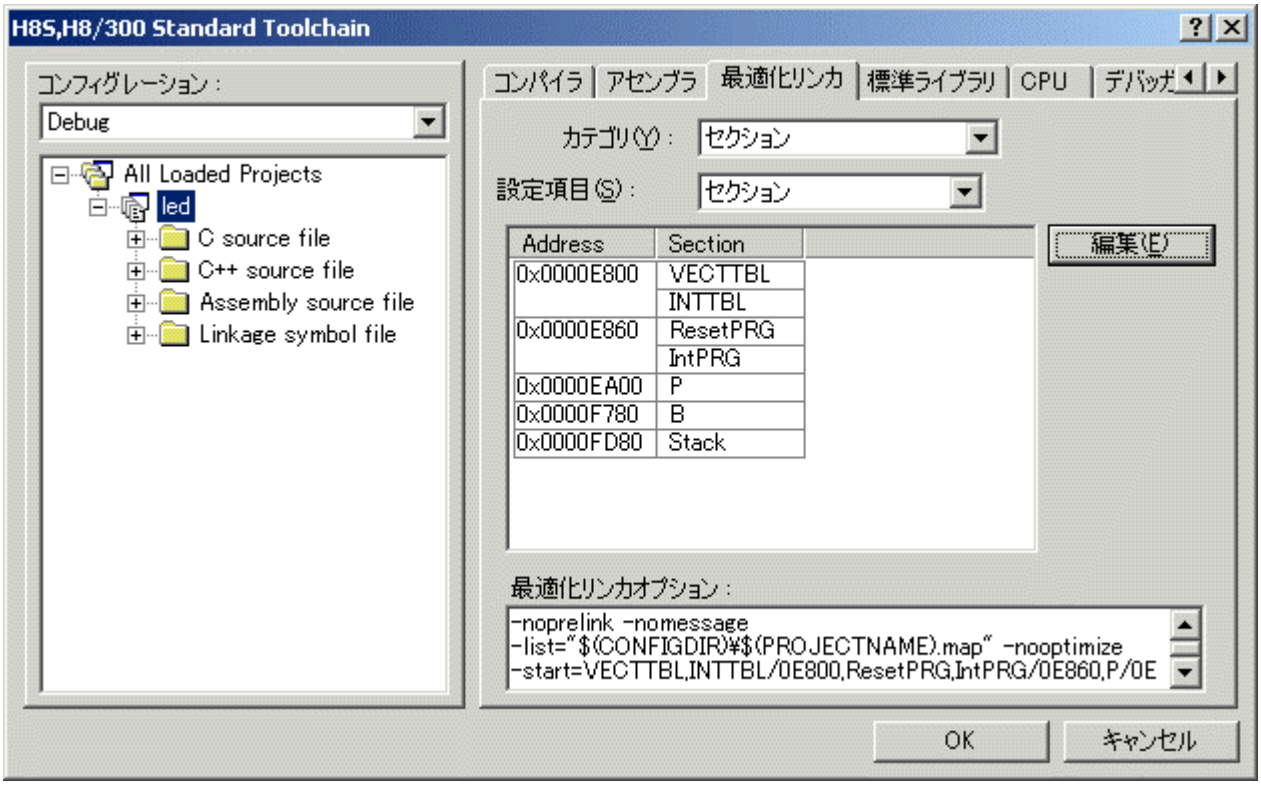


メモリマップと同じように Section が指定されていることを確認します。ちゃんと設定されていたら「OK」をクリックします。



セクション設定の保存

次回のために今修正したセクション情報を保存することができます。下段の「エクスポート（E)」ボタンをクリックしてください。保存用のダイアログが開きますので好きな名前を付けて保存します。次回は「インポート（I)」ボタンをクリックすると保存したセクション設定を呼び出すダイアログが開きます。(お



もう一度確認してから「OK」をクリックして‘ H8S，H8/300 Standard Toolchain ’ウィンドウを閉じます。

2004-2005 Toyo-linx,Co.,Ltd.

5. プログラムの入力

HEW のワークスペースウィンドウの ‘ led. src ’ をダブルクリックしてください。すると，自動生成された ‘ led. src ’ ファイルが開きます。

```
;------------------------------------------------------------------------
;                                                                       |
;   FILE        :led.src                                                |
;   DATE        :Wed, Apr 20, 2005                                      |
;   DESCRIPTION :Main Program                                           |
;   CPU TYPE    :H8/3687                                                |
;                                                                       |
;   This file is generated by Renesas Project Generator (Ver.4.0).      |
;                                                                       |
;------------------------------------------------------------------------



        .export     _main
_main:
        rts
        .end
```

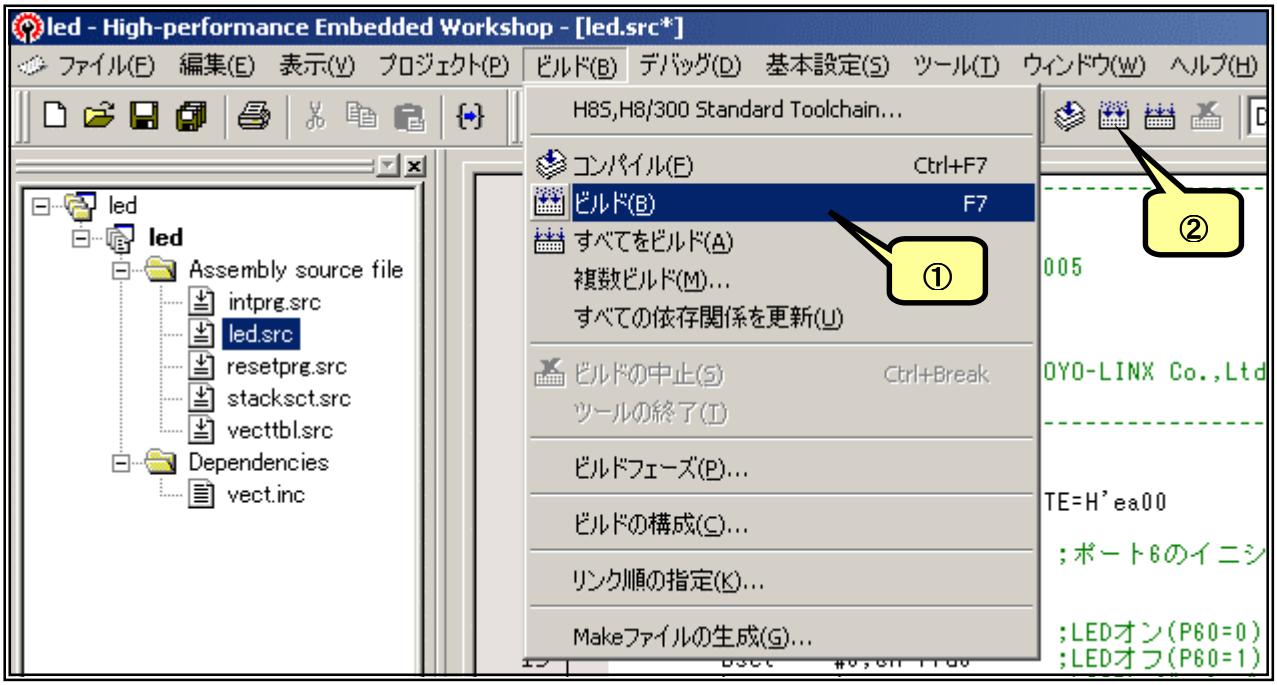
このファイルに追加・修正していきます。コーディングしたソースリストを参考に，HEW のアセンブラの文法で追加・修正したあとのリストは次のとおりです。このとおり入力してみてください。なお，アセンブラの文法については，HEW をインストールしたときに一緒にコピーされる「H8S，H8/300 シリーズ C/C++コンパイラ，アセンブラ，最適化リンケージエディタ　ユーザーズマニュアル」の中で説明されています。

```
;------------------------------------------------------------------------
;                                                                       |
;   FILE        :led.src                                                |
;   DATE        :Wed, Apr 20, 2005                                      |
;   DESCRIPTION :Main Program                                           |
;   CPU TYPE    :H8/3687                                                |
;                                                                       |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi             |
;                                                                       |
;------------------------------------------------------------------------

        .export     _main
        .section    P,CODE,LOCATE=H'ea00
_main:
        mov.b   #H'01,r0l       ;ポート6のイニシャライズ
        mov.b   r0l,@H'ffe9
loop:
        bclr    #0,@H'ffd9      ;LEDオン(P60=0)
        bset    #0,@H'ffd9      ;LEDオフ(P60=1)
        bra     loop            ;LOOPにジャンプ

        .end
```

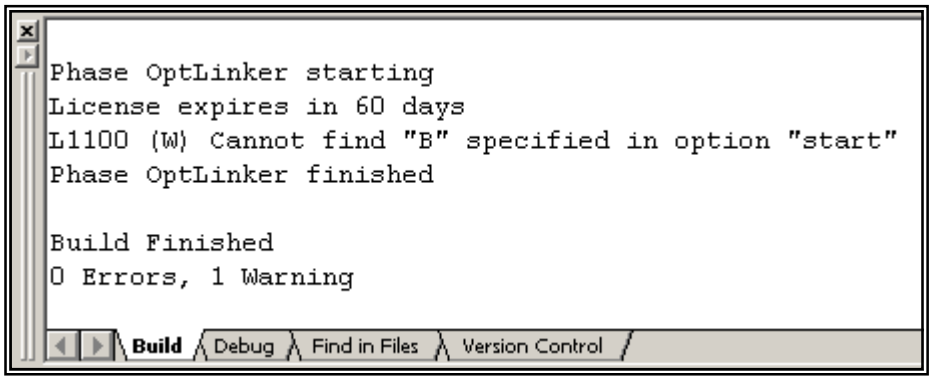

2004-2005 Toyo-linx,Co.,Ltd.

6. ビルド!!

では，アセンブルしてみましょう。HEW ではこの作業をビルドと呼んでいます。ファンクションキーの［F7］を押すか，図のように①メニューバーから「ビルド」を選ぶか，②ツールバーのビルドのアイコンをクリックして下さい。



アセンブルが終了するとアウトプットウィンドウに結果が表示されます。文法上のまちがいがないかチェックされ，なければ「0　Errors」と表示されます。



エラーがある場合はソースファイルを修正します。アウトプットウィンドウのエラー項目にマウスカーソルをあててダブルクリックすると，エラー行に飛んでいきます（このあたりの機能が統合化環境の良いところですね。）ソースファイルと前のページのリストを比べてまちがいなく入力しているかもう一度確認して下さい。

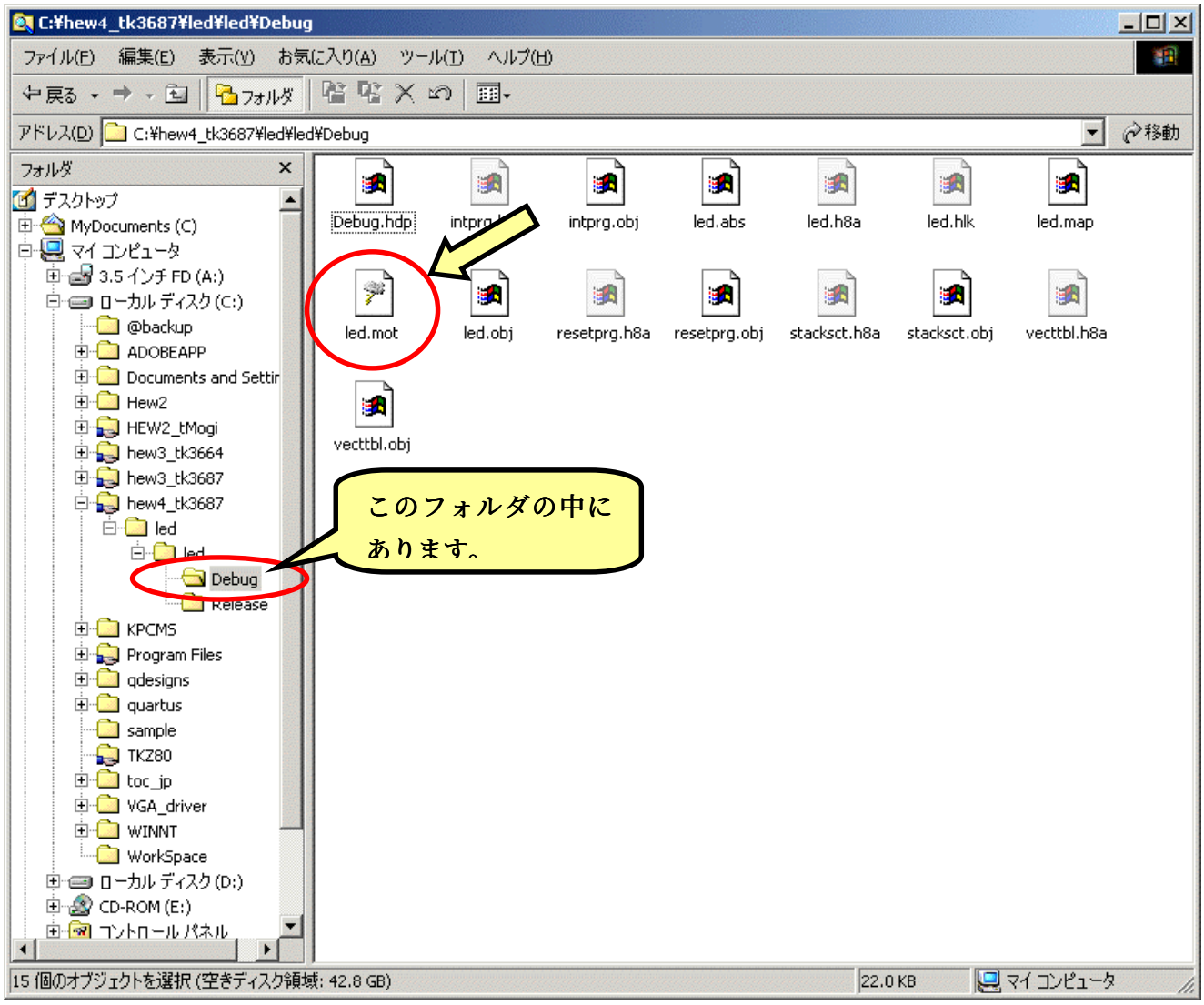
さて，図では「1　Warning」と表示されています。これは「まちがいではないかもしれないけど，念のため確認してね」という警告表示です。例えばこの図の「L1100(W) Cannot find "B" specifind in option "start"」は，B セクションを設定したのに B セクションのデータがないとき表示されます。今回のプログラムでは B セクションは使っていませんので，この警告が出ても何も問題ありません。

もっとも，Warninng の中には動作に影響を与えるものもあります。「H8S，H8/300 シリーズ　C/C++ コンパイラ，アセンブラ，最適化リンケージエディタ　ユーザーズマニュアル」の 607 ページからアセンブラのエラーメッセージが，621 ページから最適化リンケージエディタのエラーメッセージが載せられていますので，問題ないか必ず確認して下さい。

2004-2005 Toyo-linx,Co.,Ltd.

7. ダウンロードとトレース実行

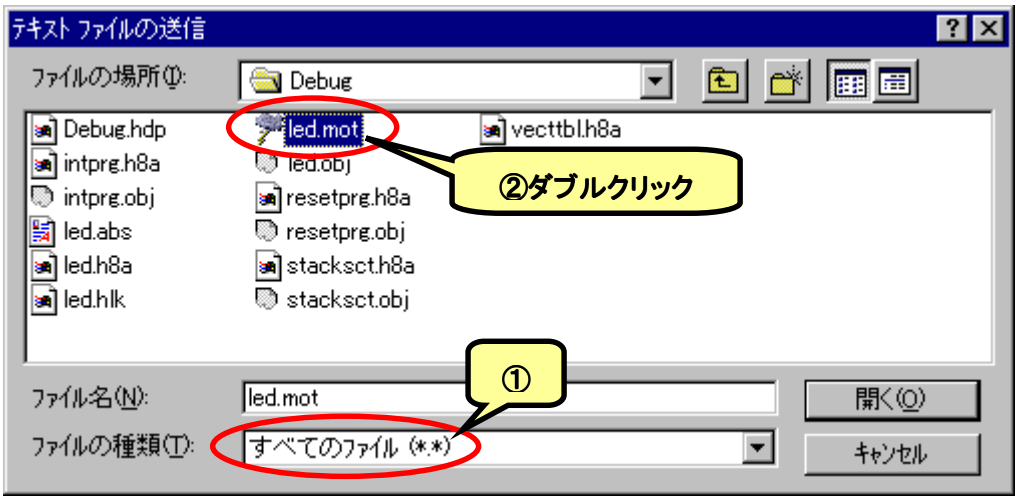
アセンブルすると‘led．mot’というファイルが作られます。拡張子が‘．mot’のファイルは「S タイプファイル」と呼ばれていて，マシン語の情報が含まれているファイルです。このファイルは次のフォルダ内に作られます。



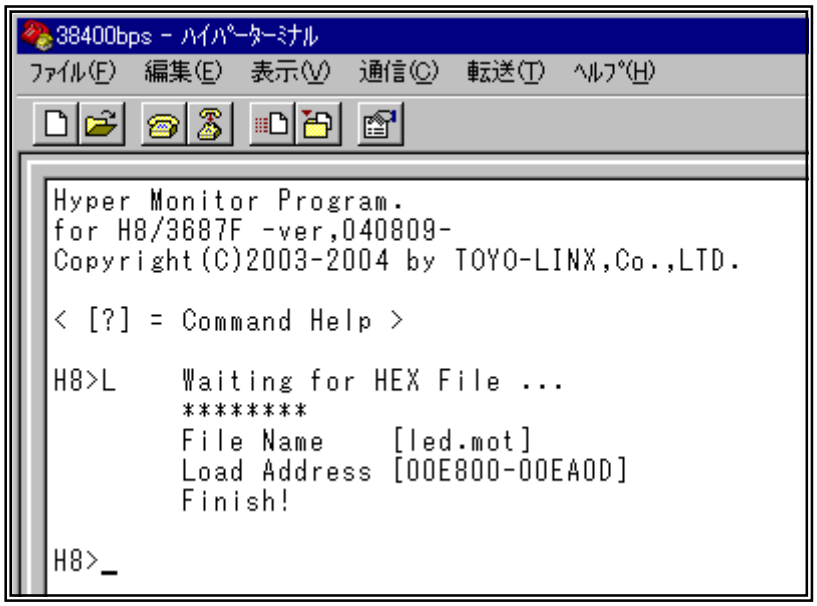
それでは，ハイパーH8 を起動して下さい。‘L’コマンドを使います。パソコンのキーボードから‘L’と入力して‘Enter’キーを押します。

メニューから「テキストファイルの送信（T）…」を選択します。

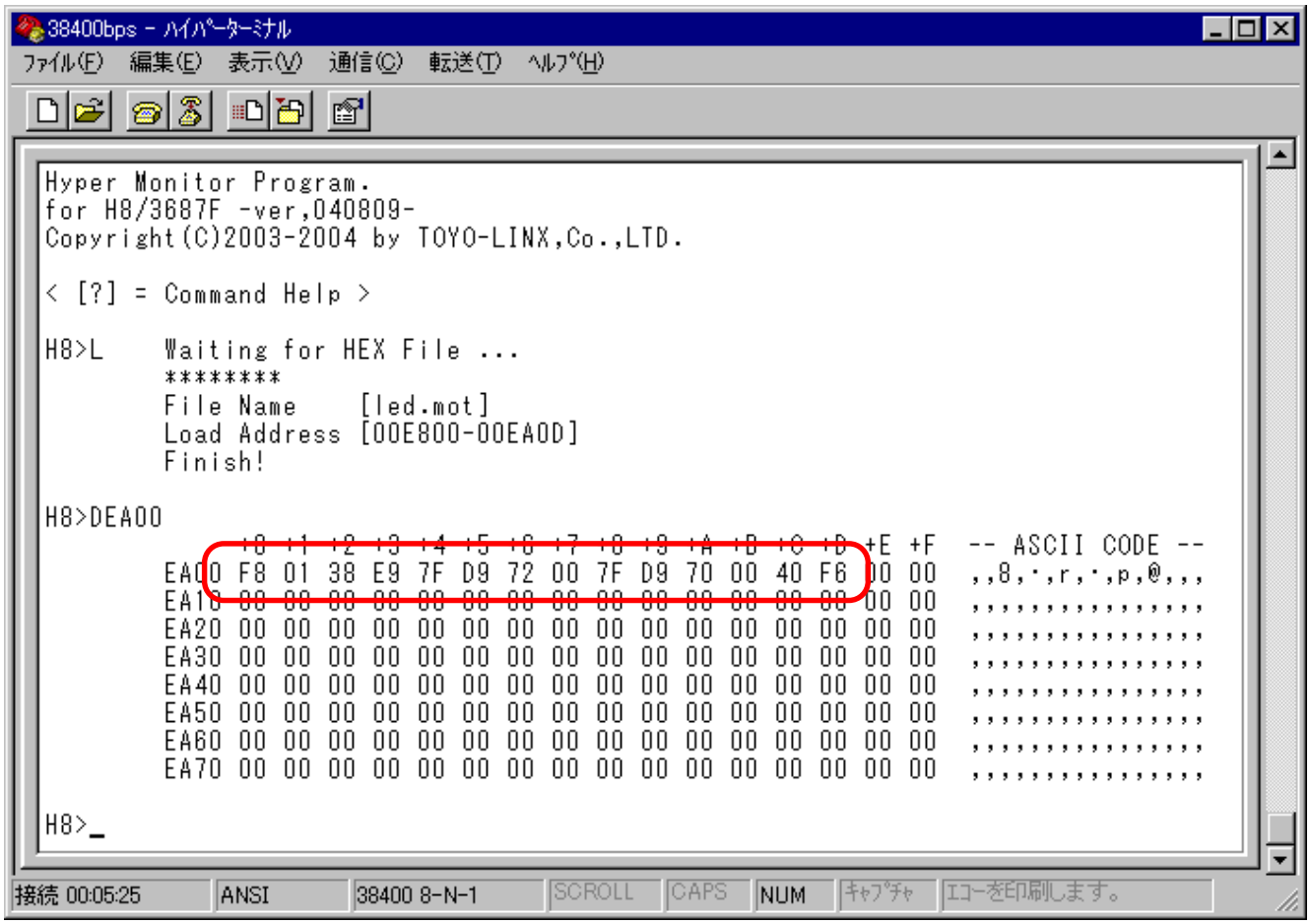
‘テキストファイルの送信’ウィンドウが開きます。①ファイルの種類を‘すべてのファイル’にして下さい。②‘led.mot’をダブルクリックします。



ダウンロードが始まります。終了すると右のように表示されます。



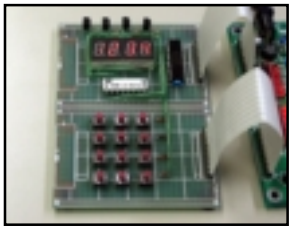
ちゃんとダウンロードできたか確認しておきましょう。パソコンのキーボードから‘ DEA00 ’と入力して‘ Enter ’キーを押します。



あとは‘ T ’コマンドでトレース実行してみてください。ちゃんと動作するでしょうか。

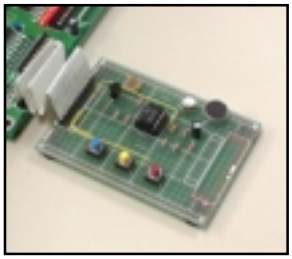
～～～ オプション品のご紹介 ～～～

★ ダイナミックスキャンによる 7 セグメント LED 表示&マトリックスキーキット：B6086（¥3,150-）



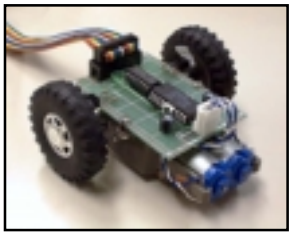
PIO を使用したダイナミックスキャンでの 7 セグメント LED 表示と 3×4 のマトリックスキー入力の学習キットです。学習内容はユニバーサル基板にハードを実装するところから始め，PIO の基本的な使い方・ダイナミックスキャンの考え方・マトリックスキーの入力方法，それらの応用として RTC を用いた時計プログラムを作成します。

★ 光センサとコンデンサマイク使用 A/D 変換キット：B6087（¥2,100-）



内蔵 A/D コンバータを使用した明るさと音を変換する学習キットです。学習内容はユニバーサル基板にハードを実装するところから始め，A/D コンンバータの基本的な使い方・光センサでの直流信号の A/D 変換・コンデンサマイクを使用した交流信号の A/D 変換とパソコンへの送信をプログラムします。又，パソコン側の受信ソフトとして，Win32API による方法と Excel の VBA による方法を示します。

★ DC モータの回転制御：B6088（¥3,780-）



ツインモータギアボックスとドライバ IC・TA7279P/AP を用いた DC モータ制御の学習キットです。フォトリフレクタによりタイヤの回転数を検出することが出来ます。学習内容はユニバーサル基板にハードを実装するところから始め，単純な PIO での制御，PWM での制御，応用として回転数一定制御をプログラムしていきます。

★ AC パワーコントローラキット：B6089（¥3,150-）

★ RS-485 実習キット：B6085（¥3,150-）

★ その他のオプション：

AC アダプタ(¥2,100-)，RS232C ケーブル(9 ピン-9 ピン，¥1,155-)，ユニバーサル基板セット（¥1,050-，A7 版，ケーブルつき） 等

（価格は全て税込価格です）

★ご質問，お問い合わせはメール又は FAX で，，，

（株）東洋リンクス

〒102-0093 東京都千代田区平河町 1-2-2 朝日ビル

TEL: 03-3234-0559 ／ FAX: 03-3234-0549

URL: http://www2.u-netsurf.ne.jp/~toyolinx

E-Mail: toyolinx@va.u-netsurf.jp

※本書の内容は将来予告無しに変更することがあります（2005 年 4 月作成）

20050422

2004-2005 Toyo-linx,Co.,Ltd.