



目 次

1 はじめに、、、 1
2 組み立て(キットでお買い求めのお客様へ) 2
3 メモリーマップ・・・・・・5
4 簡易モニタ"ハイパーH8" 6
5 無償版コンパイラのインストール 11
6 プログラムの作成と確認 12
7 実習プログラム
8 応用プログラム 25
付録-1 FDT のダウンロード 52
付録-2 FDT での書き込み手順・・・・・53
付録-3 エミュレータを使ったダウンロードと実行方法 58
付録-4 HEW をよりよい環境にするために 64
付録-5 回路図
付録-6 部品表
付録-7 無償評価版コンパイラ 70

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。

# 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。

#### ■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。





XLED

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 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







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。



+側の印





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。



### 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。

アドレス		マップ				
H'0000						
L'DEFE	£= /	内蔵 ROM (56kByte)				
	:					
		未使用				
H'E800	ユーザ割り込みベクタ					
H'E860	resetPRG					
H'EA00	Р					
		ユーザ RAM エリア	内蔵 RAM (2kByte)			
H'EFFF						
		未使用	未使用			
H'F700 H'F77F	内音	ß I/O レジスタ	内部 I/O レジスタ			
H'F780	変数領域		内蔵 RAM			
		ユーザ RAM エリア ※E7 使用時、ユーザ使用不可	(1kByte) フラッシュメモリ書き換え用			
			ワークエリア			
H'FB80						
H'FD80	Stack	ユーザ RAM エリア				
		_ , ivin _ , ,	内蔵 RAM (11:Posto)			
H'FE00		: ヽイパー <sup></sup>	(IKDyte)			
<b>អ</b> 'ជ្ជច7ច	5					
H'FF80	 古立	RT/OLジスタ				
H'FFFF			へくくく ○口 ゴロドン			

<TK-3687 メモリーマップ>

### 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 **ヘダイレクト**(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。

接続の設定 ? 🔀			
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●	初設定 ? ×	] 10プロパティ	? X
名前を入力し、アイコンを選んでください。	38400bps	-トの設定	
- 1991 1997- 38400bps アイコンの:	諸番考の暗報を入りしてたとい。 /地域番号(Q): 日本 (81)	ビット/秒(图): 38400	
😼 🍣 🌭 🖳 🍪 😺 🔊	外局番(E): 03	データ ピット(D): 8	
	話番号(P): 続方法(N): COM1	パリティ(空) なし	
<u>OK</u> キャンセル	OK キャンセル	ストップ ビット( <u>S</u> ): 1	
-		」 フロー制御(E): Xon/Xoff ▼	
		既定値に戻す(日)	
	-	<b></b> きゃンセル 道用(	<u>A</u> )

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。

HyperCom1のプロパティ	? X
接続の設定設定	
- ファンクション キー、方向キー、Ctrl キーの使い方 ● 友ーミナル キー(①) ○ Windows キー(W)	
BackSpace キーの送信方法 の Ctrl+H(C) の Delete(D) の Ctrl+H、スペース、Ctrl+H(H)	
Iミュレーション(E): ANSIW _ ターミナルの設定(G)	
Telnet ターミナル ID( <u>N</u> ): VT100	
バッファの行数(B): 500 🚞	
┣ 接続/切断時に音を鳴らす(P)	
<u> エンコード方法</u> ゆ ASCII 設定( <u>A</u> )	
OK	211

ASCII 設定 ?×
ASCII の送信
□ 行末に政行文字を付ける(S)
ローカル エコーする(E)
ディレイ (行)(上): 0 ミリ秒
ディレイ (文字)(2): 0 11秒
- ASCII の受信
☑ 着信データに改行文字を付ける(A)
□ 着信データを強制的に 7 ビット ASCII (こする(E)
□ 右端で折り返す(₩)
<u> </u>

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。

餋 HyperCom1 - ハイパーターミナル	
ファイル(E) 編集(E) 表示(Y) 通信(C) 転送(T) ヘル	<b>ク</b> 伯)
D 🖻 👩 🚨 🖻 🖆	
Hyper Monitor Program. for H8/3687F Copyright(C)2003 by TOYO-LINX,Co.,LT < [?] = Command Help > H8>_	D.

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。

🦓 38400Бр	s - 1/1/K	ーターミナル					
ファイル(E)	編集(E)	表示⊙	通信©	転送①	ヘルプ(田)		
新しい接着 開く( <u>0</u> )…	続( <u>N</u> )						
上書き保	存( <u>S</u> )						
名前を付	けて保存(4	<u>4</u> )					
ページ設3 印刷( <u>P</u> )	定( <u>U</u> )						
プロパティ	(R)						

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 > 7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。

アクティブ デスクトップ( <u>D</u> )		۲
アイコンの整列(1) 等間隔に整列(1) 最新の情報に更新(E)		۲
貼り付け( <u>P</u> )		
ショートカットの貼り付け(S)		
元に戻す-削除(U)	Ctrl+Z	
新規作成(W)		۲
プロパティ( <u>R</u> )		



※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。

<mark>後 HyperCom1 - ハイパーターミナル</mark> ファイル(E) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)		
H8>L Waiting for HEX File	HyperComi - ハイパーターミナル     ファイル(2) 編集(2) 表示(2) 通信(2) 転送(2) ヘルブ(2)     ファイルの送信(3).     ファイルの送信(3).     ファイルの受信(3).     ファイルの受信(3).     ファイルの受信(3).     ファイルの送(3(2).     キャブチャして印刷(2)     キャブチャして印刷(2)	

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=17ル\#\_piochk.mot(CD-ROM ドライブが E の場合)です。

テキスト ファイルの送信					?×
ファイルの場所の	joiochk		•	🗢 🗈 💣 📰	
び 履歴 デスクトップ	画像 の FDT書き込み手、 で FDT書き込み手、	順と動作確認.doc 順と動作確認.pdf			
マイ ドキュメント マイ ドキュメント マイ コンピュータ					
マイ ネットワーク	ファイル名( <u>N</u> ): ファイルの種類( <u>T</u> ):	_piochk.mot すべてのファイル (*.*)		• •	開(@) キャンセル

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。

<mark>終</mark> HyperCor ファイル(E)	m1 - ハイパーターミナル 編集(E) 表示(M) 通信(C) 転送(D ヘルブ(E)	<u> </u>
D 🖻 🧧		
H8>L	Waiting for HEX File ************ File Name [_piochk.mot] Load Address [00E800-00E8C5] Finish!	
H8>G	Run Address [00E840] Running	

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。

ByperCom1 - ハイパーターミナル     ファイル(E) 編集(E) 表示(M) 通信(E) 転送(E) ヘルブ(H)	×
<u> </u>	
H8>G Run Address [00E840] Running //////////////////////////////////	

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。

9	HyperCo	m1 - ハイパーターミナル				_	
-	/ア1ル( <u>E</u> ) ▶  - >   -		転送(1) ヘルフ	Ψ			
	H8>G	Run Address [00E84 Running	0]				
	11111		<< Reset!	Program St	op >> /////	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
	H8>T	[00E840]       [10         [00E846]       [10         [00E848]       [10         [00E884]       [10         [00E884]       [10N         [00E888]       [10N         [00E888]       [10N         [00E888]       [10N         [00E888]       [10N         [00E888]       [10N         [00E886]       [10N         [00E886]       [10N         [00E886]       [10         [00E886]       [10         [00E886]       [10         [00E886]       [10         [00E892]       [10	7A07         0000           0780            4036            01F0         6500           F8FF            38E8            38E4            38E5            38E4            38E8            38E4            38E8            38E8            38E8            38E8	FE00	MOV.L BRA BRA MOV.B MOV.B MOV.B MOV.B MOV.B MOV.B MOV.B MOV.B	#0000FE00,ER7 #80,CCR E880 ER0,ER0 #FF,R0L R0L,@FFE8 R0L,@FFE8 R0L,@FFE4 R0L,@FFE4 R0L,@FFD8 R0L,@FFD8 R0L,@FFD8 R0L,@FFD8	

#### トレースを終了するには"/"を入力します。

H8>_	L00E88AJ [00E88C] [00E88C] [00E890] [00E890] [00E892] [00E894]	L 10NJ [10N] [10X] [10Z] [10Z] [10Z]	38E4 38EA 1588 38D8 38DB 38D4				MOV.B MOV.B XOR.B MOV.B MOV.B MOV.B	RUL,@FFEA ROL,@FFEA ROL,@FFD8 ROL,@FFD8 ROL,@FFD8 ROL,@FFD4	
接続 0:00:31	ANSIW	38400 8-N-1	SCRO	L CA	S NU	M Fr	2. II-	を印	_

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。

# ■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。

4	🏷 HyperCom1 -	M	(ーターミナル		×
	ファイル(E) 編集	€( <u>E</u> )	表示(V) 通信(C) 転送()	D ヘルプ(H)	
[	12 🕫	\$ <u> </u>			
Г	r				
L	H8>?				
L	*	***	K Command Help ****	***************************************	
L		L:	ELoad HEX File	Select HEX(.mot)File by Menu Bar.	
L		G :	: Program Run	[G],[Gxxxx],[Gxxxx,xxxx],[G,xxxx]	
L		T :	Program Trace	[T],[TR],[Tn],[TRn]	
L		S :	=Skip Trace	LSJ,LSRJ,LSnJ,LSRnJ (n=1-999d)	
L		_	<b>D D</b> .	Enteri=(Skip) race : 'R'=Register : '/'=Escape	
L	1 3	U :	: Dump Data		
L		14 S	- Write Data		
L		к.	- Kegister view		
L		г. ე.	- Module Register		
L		;	- Command Herp	(v:Address / r:Register / d:Date )	
L	-		- Cancer : System Info ===:	( X.Aduress / T.Negrster / U.Data /	
L		CPU	- Gystem Inno H8 3687E / Hear R/	M &rea:[E800->EEE:E780->EDEE]	
	*	***	****		
	H8>				

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。

HyperCom1 - ハイパーターミナル     ファイル(E) 編集(E) 表示(V) 通信(C) 転送(T) ヘルブ(U)     「」(C2) (の) ス) …□(P) [10]	
H8>LG Waiting for HEX File **********************************	

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。

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。



無頃版コンパイノ 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。

# 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。

(Phigh-performance Embedded Workshop ファイル(F) 編集(F) 表示(V) ブロジェクト(P) オ	ブション(0) ドルド(B) ツール(T) ウィ	ッドウ(W) ヘルプ(H)		>
	n 0 🕅 🗖 🙀	■ 4 6 8	5    *F	T
* * * * * * * * * * * * * * * *		- × 🔿 🗔 🗷		
Project Files	DC2     T39a2:     で 勝須70岁2017     で 勝須70岁2017	- <u>リスペースの</u> 作気©! <u>&gt; シオ・プラスペースを閉く@):</u> aang.]¥sang.] Iwos ウスペースを参照する@)	<b>アメ</b> <u> 〇K</u> キャンセル アドミニストレーション(例)	
T	iontrol f			
1/71				NUM

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

新規プロジェクトワークスペース	? ×	
フークスへ ス名(W):	OK	①ワークスペース名を入力
samp_01	بالتلين مستلج	
<del>)ロジェ</del> クト名(P):		
samp_01		
ディレクトリ(型):	$\smile$	
C:¥HEW2_tMogi¥samp_U1		▶ ②作業フォルダを指定
CPUファミリ( <u>C</u> ):		
H85,H87300		
ツールチェイン(①):		
Hitachi H8S,H8/300 Standard		
プロジェクトタイプ( <u>R)</u> :		
Assembly Application		
Empty Application		
- Library		

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。

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。



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'1' 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

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'.!'

>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<u>第 1 の関門で、</u> <u>プログラムが通るかどうかが問題です</u>)。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

Samp_01 - High-performance Em     シ ファイル(E) 編集(E) 表示(V)	bedded Workshop - (wettblarc) フロブェブト化 オプション(の) ビルド(の) ツール(の ウィンドウ(の) ヘルブ(り)	_ D ×
Camp () Camp () Cam	Petug     Debug session     Petug       FILE     :vecttbl.src       DATE     :Tue, 0ct 07, 2003       DESCRIPTION :Initialize of Vector Table       CPU TYPE       This file is generated by Hitachi Project Generator (Ver.2.1).	A
	. include "vect.inc" .section VECTBL.data .export .RESET_Vectors .reseTVectors: :<> : OP Ower On Reset .data.w. Power(ON Reset)>> :<> ::: <td>&gt;</td>	>
Tojec	INT_Vectors:	T
C:WHEW2_tMogiVsamp_01Vsamp_0 Phase Tiny/SLP Assembler fin Phase Tiny/SLP OptLinker sta Phase Tiny/SLP OptLinker fin Duild Fisicked	liketakat, src isinad srtine ished	×
0 Errors, 0 Varnings	ie: main Catel	<b>_</b>

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。

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

2	
Hyper M for H8, Copyria < [2] :	Monitor Program. 73887F ght(C)2003 by TOYO-LINX,Co.,LTD. = Command Help. >
H8>L	Waiting for HEX File ********** File Name [samp_01.mot] Load Address [00E800-00EA19] Finish!
H8>DEA(	00         +0         +1         +2         +3         +4         +5         +6         +7         +8         +9         +A         +B         +C         +D         +E          ASCII         CODE            EA00         7A         70         00         00         00         78         6A         08         FD         00         6A         09         FD         1          ASCII         CODE          X        , j,, EA30         00

 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 EAOE 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

HyperCon	12 - ハイパーターミナル	. 🗆 🗡
7711/E)	編集(E) 表示(W) 通信(E) 転送(E) ヘルズ(E) J の L polypolycati	
H8>RPC	EA00 PC [00EA00]	
H8>R	PC [00EA00] IUHUNZYC SP [FE00] CCR [10000000] ER0 [00000000] ER1 [00000000] ER2 [00000000] ER3 [00000000] ER4 [00000000] ER5 [00000000] ER6 [00000000] ER7 [0000FE00]	
H8>T	[00EA00] [10] 7A07 0000 FE00       FE00        MOV.L       #0000FE00,ER7         [00EA06] [10] 0780         LDC       #80.CCR         [00EA08] [10] 6A08 FD00         MOV.B       @FD00,R0L         [00EA00] [10Z] 6A08 FD01        MOV.B       @FD01,R1L         [00EA00] [10Z] 6A98 FD02        MOV.B       @FD01,R0L         [00EA10] [10Z] 6A88 FD02        MOV.B       R0L.@FD02         [00EA16] [10Z] 5A00 EA08        JMP       @00EA08         [00EA03] [10Z] 6A08 FD00        MOV.B       @FD00,R0L	
H8>TR	[00EA08] [10Z] 6A08 FD00 MOV.B       @FD00,R0L         ER0 [0000000] ER1 [0000000] ER2 [0000000]       ER3 [0000000]         ER4 [00000000] ER5 [0000000] ER6 [0000000]       ER7 [0000FE00]         [00EA02] 10Z] 6A09 FD01 MOV.B       @FD01,R1L         ER0 [0000000] ER1 [0000000] ER2 [0000000]       ER3 [00000000]         ER0 [00000000] ER1 [0000000]       ER2 [00000000]         ER0 [00000000] ER3 [0000000]       ER2 [00000000]         ER0 [00000000] ER3 [00000000]       ER2 [00000000]	
 接続 0:10:07	ANSIW 38400 8-N-1 SCROLL CAPS NUM キャ・ エコーを印	

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。 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。

4月576日 - 1107 - 1141 10日 編集(11) 表示(11)

加算命令が実行された時点で、ROL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 EAOE 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。

D# 63 08 M 111000056] ER1 [00000078] ER5 [00000000] ERE [000000000] 000000000 ER3 ER7 684 **CONTRACTOR** [00EAHA] [10--END [8888800002] EN4 [888880000] 88 F002 ROL. 0FD0 ER1 ER5 Teer 0078 ERC [000000000 ERC [000000000 00000 ERT 0 5400 ERI 1000 0000000 682 ER2 ERE 0000000000 FRI 121 0781 ER: [00000000] ER2 🏘 com1\_57.6Kb - /// 建氰色 筆出所 理睬的 輕強的 化口花 그도 83 - 22 2 HESEEARD, GADE Ray Address (DEEARD) -> [DEEARD] Ray ing... DODEARD] GAND EARD Read() PC (DEEARD] GGR [10--N---) 5400 EAH ----- ----**BILEACO** H8>DFD00 
 +C
 +D
 +E

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00

 00
 00
 00
 00
 ASCII CODE --+F 60 +0 FD00 56 FD10 88 FD20 78 FD20 FD20 FD20 FD20 FD20 ¥1..... 00 00 00 00 00 00 00 885

GAPS NUM

通信の 転送の ヘルプロ

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。

ANSW

補持 02552

38400 8-N-1

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。

• • • • • • • • •

#### 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。

# 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。

; FILE	:samp	_02.src	
; DAIE	INU,	Jul 31, 2003	
· CPU TYP	PF :H8/3	687	
;			, 
; This fi	le is prog	ramed by toyolir	nx/tMogi
; samp_	<u>0</u> 1の復習		
; モニ?	タでメモリの	)データ書き換える 、	をする代わりに,転送命令で
; 入七·	ノに香さ込e 、DATAO 7	J₀ Q\DATA1_2/	
, 30- · ADD S	>DATAD,70	み合わせる。	
; 56+	·78>D/	ATA4	
; 56+	-78-34>D	ATA5	
; 演算約	詰果をポート	- 5 へLED表示する	·•
;			
; ; 課題[	 こ挑戦!		
; 課題1	. ADD,SUB実	行後のゼロ,キャ	」 ーフラッグの変化を追う。
; 課題2	. ADD , SUB俞	命令をAND,OR命令	こ置き換える。
; 課題3	. ADD , SUB俞	命令をXOR,NOT命令	に置き換える。
; 課題4	•		l
,			
;	export	 main	
;	.export	a	
DATAO	.equ	h'fd00	
DATA1	.equ	DATA0+1	
DATA2	.equ	DATA0+2	
DATAA	.equ	DATAO+3	
	.equ		
	.equ	DATA0+5 b'ffd8	
PCR5	.equ	PDR5+h'10	
	. oqu		
;			
_main:	mov.b	#h'ff,r31	;PORT5を出力に設定
	mov.b	r31,@PCR5	
	move	#b'56 r41	
	mov b	#11 30,141 r/l @DATAO	,50@DATAUL L9F
	mov b	#h'78 r4l	·78をDATA1にセット
	mov.b	r41.@DATA1	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	mov.b	#h'34,r41	;34をDATA2にセット
	mov.b	r4I,@DATA2	
loop 01:	mov.b	@DATAO.rOI	
·	mov.b	@DATA1,r1I	
	mov.b	@DATA2,r2I	
	add.b	r11,r01	;r1l+r0l=>r0l
	mov.b	rOI,@DATA4	;r01をDATA4に格納
	mov.b	rOI,@PDR5	; ″ PDR5に出力

sub.b	r21,r01	;r01-r21=>r01
mov.b	rOI,@DATA5	;r0lをDATA5に格納
mov.b	rOI,@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 PCR5 .equ h'ffe8 .equ PDR5 h'ffd8 wait1000 .equ h'140000 ;-----Initialize Routine -----\_main: mov.b #b'00000001,ROL ;ポート5,ビット0を出力にする設定値 mov.b ROL,@PCR5 ;ポートコントロールレジスタに書く xor.b ROL,ROL ;ROLをO ;----- Main\_Loop -----loop\_01: #b'00000001,R0L ;ポート5,ビット0を1に or.b mov.b ROL,@PDR5 ;LEDは点灯する @wait1s jsr and.b #b'11111110,ROL ;ポート5,ビット0を0に mov.b ROL,@PDR5 ;LEDは消灯する

	jsr	@wait1s	
	bra	loop_01	;loop_01へ無条件ブランチする
;		Sub-Routine	9
wait1s:	mov	#wait1000 EP6	;1秒待つウェート
wait1s_	01:	#wall1000,ER0	
	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)

;		
, , , ,	DATE :Thu, J DESCRIPTION :Main P CPU TYPE :H8/368	4.src ul 31, 2003 rogram 7
;	This file is progra	med by toyolinx/tMogi
, ; ;	回転命令(ローテー PDR1,5を使って,	ト)の演習 , 電飾をイメージさせる。
, , , , , , , , ,	h'03>ROL,h ROLのキャリー ROHのキャリー ROL>PDR1 ,	'00>ROH 付き左回転 付き左回転 ROH>PDR5
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	課題に挑戦! 課題1.パターンの 課題2.逆回転 課題3.全消灯,全点 課題4.PDR7を追加 課題5.	変更 気灯の間欠動作
,	.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	
; main:		Initialize	
	mov.b mov.b mov.b	#h'ff,ROL ROL,@PCR1 ROL,@PCR5	;ROLを出力に設定
	mov.b mov.b andc	#b'00000011,ROL #b'00000000,ROH #b'11111110,CCR	;バイナリでR0Lにセット ;バイナリでR0Hにセット ;キャリーを0にセット
;		Mian_Loop	
Toop_01:	rotx1.b rotx1.b	ROL ROH	;ROL左回転 ;ROH左回転
	mov.b mov.b	ROL,@PDR1 ROH,@PDR5	;PDR1に出力 ;PDR5に出力
	jsr	@wait	
	jmp	@loop_01 Sub_Boutino	
, wait:	mov.l	#wait200,ER6	;0.2秒待つループ
wait_01:	dec.l bne rts	#1,ER6 wait_01	;ER6を - 1 ;0でなければ , Ioop_02分岐
,	ond		

.end

### 8 応用プログラム

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。





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'FF	D8	; ポートデータレジスタ_5	_r	main_10:		;*** スイッチが押された ***
PCR5	.equ h'FF	E8	;ボートコントロールレジスタ_5		bclr	#4,@PDR5	; 7 <sup>·</sup> <sup>·</sup> <sup>-</sup> on
					bsr	Wait_100m	;嗚動時間
	.export	_main			bset	#4,@PDR5	; <b>ブサー</b> off
					bset	#0,r11	; 从1ッナの状態セット 1=00
_maı	n: /=\u = /-	*			bra	_main_00	
;	1_yt71/	(					*** フノイナヤナヤ テナン ***
	mov.b	#H'F0,r01	; PIO 1_2771X	_n	main_20:		
	mov.b	r01,@PCR5	; P50-53:in / P54-57:out		xor.w	r1,r1	; メイッナの状態セット 0=off
					bra	_main_00	
	bset	#4,@PDR5	; <b>7 7</b> - off				
	xor.w	r1, r1	; スイッチの状態 1ニシャラ1ス	;-	10msec	ባ፤1ト	
				Wa	ait_10m:		
;	メインループ				mov.I	#H'8235,er6	
_mai	n_00:		;*** X1vf 1st U-F ***	Wa	ait10m_00:		
	btst	#0,@PDR5	; スイッチ リート		dec.I	#1,er6	
	beq	_main_02	; 0=on / 1=off		bne	Wait10m_00	
	bra	_main_20			rts		
_mai	n_02:		;*** ワンショット動作 ***	;-	100msec	; ウェイト	
	MOV.W	r1,e1	;今迄のスイッチの状態をチェック	Wa	ait_100m:		
	beq	_main_04	; 1=on / 0=off		mov.l	#H'51612,er6	
	bra	_main_00		Wa	ait100m_00:		
					dec.l	#1,er6	
_mai	n_04:		;*** X1yf 2nd U-F ***		bne	Wait100m_00	
	bsr	Wait_10m	; チャタリング除去		rts		
	btst	#0,@PDR5	; スイッチ リード				
	beq	_main_10	; 0=on / 1=off		. end		
	bra	_main_02					



TK-3687 ユーザーズマニュアル 2004-2005 Toyo-linx, Co., Ltd.

# 8-1-2 スイッチとブザー く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 へー定間隔の 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	;ポートデータレジスタ 5		btst	#0,@PDR5	;2nd 入力チェック
PCR5		eau	h'FFE8	:ポートコントロールレジスタ 5		bea	loop 20	:入力があれば Bz 鳴動
		1		,		bra		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	-		0122220	.チェカリンル* 『今十中国(40maaa)		bra	1000_00	
CHATK_CNOT	•	equ	D 33330	, デャッシング 际 云 时间(1005ec)				
BZ_CNST	•	equ	D'49996	;BZ 間欠時間(15msec)	Toop_20:			;BZ 嗚舠
						bnot	#4,@PDR5	;Bz トグル出力
	.expor	t	_mair	า		bsr	WAITBz	;Bz 間欠時間
main:						mov.w	#1,r1	: フラグ セット
-	mov.b		#H'F0.r01	·P50-53:入力		bra		
	mov b		rOL @PCR5	·P54_57·出力		0.0		
	1100.0		101, @1003	,134-37.四75	. I.			
				<b>N</b> (1)	; 7	799777际云0	り為りWart	
	bset		#4,@PDR5	;Bz off	WATTCHATR	:		;6
	xor.w		r1, r1	;フラク゛クリア		mov.l	#CHATR_CNS	ST,er6 ;6
						bra	WAIT_00	
loop_00:								
. –	btst		#0.@PDR5	:1st 入力チェック	: B:	z 間欠動作時	特間の為の wa	ait
	hea		Loop 10	·入力あれば押し続けのチェックへ	WAITBz			·6
	boq		1000_10	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		mov 1 #P7	CNST ore	,0
				> 十一一		110V.1#DZ_	0101,010	,0
Toop_02:				;入刀無し				
	bset		#4,@PDR5	;Bz off	; Wa	ait loop -		
	xor.w		r1, r1	; フラグ クリア	WAIT_00:			
	bra		loop_00			dec.l	#1,er6	;2
			·			bne	WAIT 00	:4
Loop 10.						rts		· 8
1000_10.	mov w		r1 o1	・畑し、結けチェック		1.0		,0
	hov.W							
	bhe		100p_20	, 1世し続け 4 ら bZ - 馬勤	,=======			
						.end		
	bsr		WAITCHATR	;チャタリング除去				

## ■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>

PDR5	. (	equ	h'FFD8	;ポートデータレジスタ 5		btst	#0,@PDR5	;2nd 明るさチェック
PCR5	. (	equ	h'FFE8	;ポートコントロールレジスタ_5		bne	loop_20	;暗ければ Bz 鳴動
						bra	loop_00	
CHATR_CNST	Г.	equ	D'333330	;チャタリンク <sup>゙</sup> 除去時間(100msec)				
Bz_CNST	. (	equ	D'49996	;Bz 間欠時間(15msec)	loop_20:			;Bz 鳴動
						bnot	#4,@PDR5	;Bz <b>トグル</b> 出力
	.export	t	_mair	1		bsr	WAITBz	;Bz 間欠時間
_main:						MOV.W	#1,r1	; フラク゛ セット
	mov.b		#H'F0,r01	;P50-53:入力		bra	loop_00	
	mov.b		r01,@PCR5	;P54-57:出力			. –	
					; Ŧ	ャタリング 除去(	の為の wait	
	bset		#4,@PDR5	;Bz off	WAITCHATR			;6
	xor.w		r1, r1	; 7 <b>5</b> 7° 717		mov.l	#CHATR_CNS	ST,er6 ;6
						bra	WAIT_00	
loop 00:								
. –	btst		#0,@PDR5	;1st 明るさチェック	; B	z 間欠動作問	時間の為の wa	ait
	bne		loop_10	; 暗ければ鳴り続けのチェックへ	WAITBz:			;6
			. –			mov.I#Bz	CNST,er6	;6
loop_02:				;明るい		_		
. –	bset		#4,@PDR5	;Bz off	; w	ait loop –		
	xor.w		r1, r1	; フラグ クリア	WAIT_00:			
	bra		loop 00		_	dec.l	#1,er6	;2
			. –			bne	WAIT 00	:4
loop 10:						rts	_	;8
. –	mov.w		r1,e1	;鳴り続けチェック				,
	bne		loop 20	; 暗いままなら Bz 鳴動	;=======			
					,	. end		
	bsr		WAITCHATR	;チャタリング除去				

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。

### 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ー式・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	not rOI ;反転 mov.b rOI,@PDR6 ;出力 bra _main_00
_main:	INIPIO:
bsr INIPIO ;PIO イニシャライズ	mov.b #H'F0,r01 ;ポート1 イニシャライス <sup>*</sup>
_main_00:	mov.b r0I,@PCR1 ;ポート14-17:out
mov.b #H'EE,r1I ;ポート 14:Low	mov.b #H'FF,r01 ;ポート6 イニシャライス <sup>*</sup>
mov.b r1I,@PDR1 ;スキャン出力	mov.b r0I,@PCR6 ;ポート60-67:out
mov.b #B'01100110,r0I ;セグメントデータ:4	rts

<Dscan\_1>

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 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>

_mair	ו:				mov.b	#H'77,r1I	; ポート 17 : Low
	bsr	INIPIO:16	;PIO 1=>+>71		mov.b	r11,@PDR1	; スキャン出力
_mair	_00:				mov.b	#B'00000110,r01	; セグメントデータ:1
	mov.b	#H'EE,r1I	;ポート 14:Low		not	r0l	;反転
	mov.b	r1I,@PDR1	;スキャン出力		mov.b	rOl,@PDR6	;出力
	mov.b	#B'01100110,r01	; セク゛メントテ゛ータ:4		bsr	TIMER1m:16	
	not	r0l	;反転		bra	_main_00	
	mov.b	rOI,@PDR6	;出力				
	bsr	TIMER1m:16		INIP	10:		
	mov.b	#H'DD,r1I	;ポート 15:Low		mov.b	#H'F0,r01	;ポート 1 イニシャライズ
	mov.b	r11,@PDR1	;スキャン出力		mov.b	rOI,@PCR1	;ポ -ト 14-17:out
	mov.b	#B'01001111,r01	;セグメントデータ:3		mov.b	#H'FF,rOI	;ポート᠖イニシャライズ
	not	r0l	;反転		mov.b	r01,@PCR6	;ポート60-67:out
	mov.b	rOI,@PDR6	;出力		rts		
	bsr	TIMER1m:16					
	mov.b	#H'BB,r1I	;ポート 16:Low	TIME	R1m:		
	mov.b	r1I,@PDR1	;スキャン出力		mov.l	#H'D02,er6	
	mov.b	#B'01011011,r01	; セク゛メントテ゛ータ:2	TIME	R1m_00:		
	not	r0l	;反転		dec.l	#1,er6	
	mov.b	rOI,@PDR6	;出力		bne	TIMER1m_00	
	bsr	TIMER1m:16			rts		

<Dscan\_2>

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 100 μ sec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>

_main	:					
	bsr	INIPIO:16	;PI0 1=>+>12	mov.b	#H'77,r1I	; ポート 17 : Low
_main	_00:			mov.b	r1I,@PDR1	; スキャン出力
	mov.b	#H'EE,r1I	; ポ ート 14: Low	mov.b	#B'00000110,r01	; セク゛メントテ゛ータ : 1
	mov.b	r1I,@PDR1	;スキャン出力	not	r01	;反転
	mov.b	#B'01100110,r01	; セク゛メントテ゛ータ:4	mov.b	rOI,@PDR6	;出力
	not	r0l	;反転	bsr	TIMER1m:16	
	mov.b	rOI,@PDR6	;出力	xor.b	r01,r01	;セグメントデータ:消灯
	bsr	TIMER1m:16		not	r01	;反転
	xor.b	r01, r01	;セグメントデータ:消灯	mov.b	r01,@PDR6	;出力
	not	r0l	;反転	bsr	TIMER100u:16	
	mov.b	rOl,@PDR6	;出力			
	bsr	TIMER100u:16		bra	_main_00	
	mov.b	#H'DD.r1I	:ポート 15:Low	INIPIO:		
	mov.b	r1L,@PDR1	: スキャン出力	mov.b	#H'F0,r01	;ポート 1 イニシャライズ
	mov.b	#B'01001111.r0l	: セク メントデータ:3	mov.b	rOI.@PCR1	:ポート 14-17:out
	not	rOl	;反転	mov.b	#H'FF.rOI	:ポート 6 イニシャライス゛
	mov.b	rOI,@PDR6	;出力	mov.b	rOI,@PCR6	; ポ−ト 60-67 : out
	bsr	TIMER1m:16	,	rts	,	
	xor.b	r01, r01	;セグメントデータ:消灯			
	not	r0l	;反転	TIMER100u:		
	mov.b	rOI,@PDR6	;出力	mov.l	#H'14A,er6	
	bsr	TIMER100u:16		TIMER100u_00:		
				dec.I	#1,er6	
	mov.b	#H'BB,r1I	; ポ ート 16: Low	bne	TIMER100u_00	
	mov.b	r1l,@PDR1	;スキャン出力	rts		
	mov.b	#B'01011011,r01	; セク゛メントテ゛ータ:2			
	not	rOl	;反転	TIMER1m:		
	mov.b	rOl,@PDR6	;出力	mov.l	#H'D02,er6	
	bsr	TIMER1m:16		TIMER1m_00:		
	xor.b	r01, r01	;セグメントデータ:消灯	dec.I	#1,er6	
	not	rOl	;反転	bne	TIMER1m_00	
	mov.b	rOI,@PDR6	;出力	rts		
	bsr	TIMER100u:16				

<Dscan\_3>

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ー式・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 カラインです。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 _main_00: mov.b mov.b	INIPIO #H'77,r11 r11,0PDR1	;PI0 1ニシャラ1ズ ;ポート 17:Low ;スキャン出力	_main_02: mov.b not mov.b bra	#B'01011100,r01 r01 r01,@PDR6 _main_00	; セグ メント表示 ; 負論理なので反転 ; 表示
mov.b not and.b bne mov.b mov.b bra	@PDR1,r01 r01 #H'07,r01 _main_02 #H'FF,r01 r01,@PDR6 _main_00	;+-入力 ;負論理なので反転 ;+-の有効 bit は 3bit ;eq=押されていない ;セグメント消灯 ;表示	INIPIO: mov.b mov.b mov.b mov.b rts	#H'FO,rOI rOI,@PCR1 #H'FF,rOI rOI,@PCR6	;ポート 1 イニシャライズ ;ポート 14-17:out ;ポート 6 イニシャライズ ;ポート 60-67:out

<matrixkey\_1>

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。



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。

bsr         INIPIO:16         :PIO 4:5y747	_main:				mov.b @(SEG_TBL,er2),r31 ; ቲታ አንኑም – ቃ ታ ット		
mov.b         #H'00,r31         :R3L=bf'x/hf'-9         mov.b         #H'FE,r01           _main_00:           mov.b         r01,@PDR6         :BJT           mov.b         r01,@PDR6         :BJT         mov.b         r01,@PDR6         :BJT           mov.b         r01,@PDR6         :BJT         mov.b         r01,@PDR6         :BJT           mov.b         r01,@PDR6         :BJT         mov.b         r01,@PDR6         :BJT           mov.b         #H'77,r11         :R1L=J+7) <sup>T</sup> (DJ         mov.b         r01, @PDR6         :BJT           mov.b         #D'1,r2         :R2=F-#B         mov.b         r01, @PDR6         :BJT           _main_02:          mov.b         r11, @PDR1         :At+bJT         mov.b         r01, @PDR1         :At+y 'DT           mov.b         r01, @PDR1         :At+DJT         mov.b         mov.b         r01, @PDR1         :At+y 'DT           mov.b         r01, @PDR1         :At+DJT         mov.b         r01, @PDR1         :At+2y F/AT           mov.b         r01, @PDR1         :At+DT         mov.b         r01, @PDR1         :At+2y F/AT           mov.b         r01, @PDR1         :At+DT         mov.b         r01, @PDR1 <td></td> <td>bsr</td> <td>INIPIO:16</td> <td>;PIO 1=&gt;+&gt;12</td> <td>_main_10:</td> <td></td> <td>;表示プログラム</td>		bsr	INIPIO:16	;PIO 1=>+>12	_main_10:		;表示プログラム
		mov.b	#H'00,r31	; R3L=セク <sup>*</sup> メントテ <sup>*</sup> ータ	mov.b	#H'EE,rOI	
mov.b #H'FF,r01 : セヴメントデー9:消灯 mov.b r31,r01 mov.b r01,@PDR6 : 出力 not r01 mov.b #H'77,r11 :R1L=Xt+y5´-9 mov.b #H'77,r11 :R1L=Xt+y5´-9 mov.b #D'4,r1h :R1H=h-7`回数 bs r TIMER1m :表示時間 = 1msec mov.w #D'1,r2 :R2=t-番号 mov.b #H'FF,r01 : ;Xt+y off mov.b v11,@PDR1 :Xt+y出力 bra	_main_00:		mov.b	rOI,@PDR1	;表示用スキャン出力		
mov.b         r01, @PDR6         :出力         not         r01           mov.b         #H'77, r11         :R1L=X+?/-, r-j         mov.b         mov.b         r01, @PDR6         :表示; r)         t)           mov.b         #D'4, r1h         :R1H=II-?         imov.b         bsr         TIMER1m         :表示; r)         t)           mov.b         #D'1, r2         :R2=#-番号         mov.b         rDI, @PDR1         :X+?         mov.b         rDI, @PDR1           mov.b         @PDR1, r01         :A+izH=X-TC1/L?         mov.b         r01, @PCR1         :ボ - 1 1 1.?         :XPY-7/4           mov.b         #D'1, r2         :F-番号 +3         mov.b         r01, @PCR1         :ボ - 1 1 4.?>P7/3/           mov.b         #D'1, r2         :F-番号 +3         mov.b         r01, @PCR1         :ボ - 1 4.?>P7/3/           mov.b         #D'1, r2         :F-番号 +3         mov.b         r01, @PCR6         :ボ - 1 4.?>P7/3/           mov.b         r01, @PCR6         :ボ - 1 6.?>P7/3/         mov.b         r01, @PCR6         :ボ - 1 6.?>P7/3/           mov.b         r01, @PCR6         :ボ - 1 6.?>P7/3/         mov.b         r01, @PCR6         :ボ - 1 6.?>P7/3/           mov.b         mov.b         r01, @PCR6         :ボ - 1 6.?>P7/3/         mo		mov.b	#H'FF,rOl	;セグメントデータ:消灯	mov.b	r31,r01	
mov.b         #H'77,r11         :R1L=At+ンデ-9         mov.b         r01,@PDR6         :表示i+3           mov.b         #D'4,r1h         :R1H=JL-ブ 回数         bsr         TIMER1m         :表示i+3           mov.b         #D'1,r2         ;R2=1-番号         bsr         TIMER1m         :表示i+3           mov.b         #D'1,r2         ;R2=1-番号         mov.b         #H'77,r11         :At+y           mov.b         #D'1,r2         ;R2=1-番号         mov.b         #H'FF,r01         :At+y           mov.b         #DR1         :At+y         #H'77,r11         :Fis#3x3A+         mov.b         #D'1,r01           mov.b         #DR1,r01         :+fis#3x3A+         mov.b         #H'F0,r01         :ボ + 1 4:2y #74           mov.b         #M'03,r2         :+fis#6 +3         mov.b         #H'FF,r01         :ボ + 1 4:2y #74           add.w         #D'3,r2         :+fis#6 +3         mov.b         #I'FF,r01         :ボ + 1 4:17:out           add.w         #D'3,r2         :+fis#6 +3         mov.b         #I'FF,r01         :ボ + 6 4:2y #74',           bra         _main_02         Imov.b         r01,@PCR6         :ボ + 1 6:0; 0:0; 0:0;           _main_04:         :#figH         'mov.b         r01,@PCR6         :ボ + 1 6:0;		mov.b	rOl,@PDR6	;出力	not	r0l	
mov.b         #D'4,r1h         :R1H=J-ブ 回数         bsr         TIMER1m         :表示時間 = 1msec           mov.w         #D'1,r2         ;R2=4-番号         mov.b         #H'FF,r01         ;Xi+y off           mov.b         r01, @PDR1         ;Xi+y Unfn         bra        main_00           mov.b         @PDR1,r01         :4-tiBAbAA         bra        main_00           mov.b         @PDR1,r01         :4-tiBAbAA         mov.b         mov.b         #H'FF,r01         ;Xi+y off           mov.b         @PDR1,r01         :4-tiBAbAA         mov.b         mov.b         mov.b         #H'FO,r01         :# - + 14-17:out           and.b         #H'07,r01         :+ - tiBH = 10         mov.b         mov.b         #H'FO,r01         :# - + 14-17:out           and.b         #H'07,r01         :+ - tiBH = 13         mov.b         #H'FF,r01         :# - + 14-17:out           add.w         #D'3,r2         :+ - tiBH = 13         mov.b         #OI, @PCR6         :# - + 14-17:out           dec.b         r11         : XroAt+y^         mov.b         r01, @PCR6         :# - + 60-67:out          main_02         :+ - fit         mov.l         #H'FF,r06         :# - + 60-67:out         mov.b          main_04         :-		mov.b	#H'77,r1l	;R1L=スキャンデータ	mov.b	r01,@PDR6	;表示データ 出力
mov.w         #D'1,r2         ;R2=十番号         mov.b         #H'FF,r01         ;Atv2 off           _main_02:         mov.b         r01,@PDR1         imov.b         r01,@PDR1           mov.b         @PDR1,r01         ;Atv2出力         bra         _main_00           mov.b         @PDR1,r01         ;+istxibA         bra         _main_00           mov.b         #H'G7,r01         ;+istxibA         mov.b         #H'F0,r01         ;i -h 1 f2>b747           and.b         #H'O7,r01         ;+istxibA         mov.b         #H'FF,r01         ;i -h 1 f2>b747           and.b         #H'O7,r01         ;+istxibA         mov.b         #H'FF,r01         ;i -h 1 f2>b747           and.b         #H'O7,r01         ;+istxibA         mov.b         #H'FF,r01         ;i -h 1 f2>b747           and.b         #H'O3,r2         ;+istxibA         mov.b         #H'FF,r01         ;i -h 1 f2>b747           and.b         #H'D3,r2         ;+istxibA         mov.b         #H'FF,r01         ;i -h 1 f2>b747           and.b         #H'D3,r2         ;+istxibA         mov.b         #H'FF,r01         ;i -h 6 f2>b747           and.b         #Inf         ;bord         ;i -f b4         mov.b         r01,@PCR6         ;i -f 0-6-7:out </td <td></td> <td>mov.b</td> <td>#D'4,r1h</td> <td>;R1H=<b>ル-フ</b>゚回数</td> <td>bsr</td> <td>TIMER1m</td> <td>;表示時間 = 1msec</td>		mov.b	#D'4,r1h	;R1H= <b>ル-フ</b> ゚回数	bsr	TIMER1m	;表示時間 = 1msec
main_02:       mov.b       r01,@PDR1       ;X+y出力       mov.b       r01,@PDR1         mov.b       @PDR1,r01       ;+i読み込み       bra      main_00         not       r01       ;反転       INIPIO:         ad.b       #H'07,r01       ;+i描Pされていた?       mov.b       #0'.6         bne      main_04       :ne = 押されていた?       mov.b       #0'.6       #1'.FF,r01       ;ボ - h 14 - 17:out         add.w       #D'3,r2       ;+i番号 +3       mov.b       #1'FF,r01       ;ボ - h 6 - 12:yラ74,*         ictr.b       r11       ;次のスキッハへ       mov.b       #1'FF,r01       ;ボ - h 60-67:out         dec.b       r1h       ;トj*のスキャハへ       mov.b       r01,@PCR6       ;ボ - h 60-67:out         bce       _main_02       mov.b       r11'FF,r01       ;ボ - h 60-67:out       mov.b        main_02       #1       ;ho?       mov.b       r01,@PCR6       ;ボ - h 60-67:out        main_04:       :ho?       #1       rts       mov.l       #H'D02,er6         bfst       #0,r01       ;bit0 fruy       2000:       mov.l       #1.002,er6         bfst       #1,r01       ;bit1 fruy       bre       2000       rts        main_06:      m		MOV.W	#D'1,r2	;R2=キー番号	mov.b	#H'FF,rOI	;スキャン off
mov.b       r11,@PDR1       : スキャン出力       bramain_00         mov.b       @PDR1,r01       : +i読み込み       INIPIO:         not       r01       :反転       INIPIO:         ad.b       #H'07,r01       : +it押されていた?       mov.b       #H'F0,r01       : #i +1 1:2 * p747         bbe       _main_04       :ne = 押されていた?       mov.b       #I'F0,r01       : #i +1 1:2 * p747         add.w       #D'3,r2       : #i 番号 +3       mov.b       r01, @PCR1       : #i +14-17:out         add.w       #D'3,r2       : #i 番号 +3       mov.b       r01, @PCR6       : #i +14-17:out         add.w       #D'3,r2       : #i 番号 +3       mov.b       r01, @PCR6       : #i +4:17:out         add.w       #D'3,r2       : #i 番号 +3       mov.b       r01, @PCR6       : #i +4:17:out         bra       _main_02       rotr.b       r01, @PCR6       : #i +6:0-67:out       rotr.b         bra       _main_02       rotr.b       rotr.b       rotr.b       r01, @PCR6       : #i +6:0-67:out         bra       _main_08       rotr.b       ibit0 frug       mov.l       #H'BO2,er6       rotr.b       rotr.b       rotr.b       rotr.b       rotr.b       rotr.b       rotr.b       rotr.b       rotr.b	_main_02:			mov.b	r0I,@PDR1		
mov.b         @PDR1,r01         : 非読み込み (反転)         INIPIO:           and.b         #H'07,r01         : 卡は押されていた?         mov.b         #H'F0,r01         : ボート14-17:out           bne         _main_04         : ne = 押されていた         mov.b         r01,@PCR1         : ボート14-17:out           add.w         #D'3,r2         : 十番号 +3         mov.b         r01,@PCR6         : ボート60-67:out           add.w         #D'3,r2         : 十番号 +3         mov.b         r01,@PCR6         : ボート60-67:out           add.w         #D'3,r2         : 十番号 +3         mov.b         r01,@PCR6         : ボート60-67:out           dec.b         r11         : 次のスキャンへ         mov.b         r01,@PCR6         : ボート60-67:out           bne         _main_02		mov.b	r1I,@PDR1	;スキャン出力	bra	_main_00	
not r0l ;反転 and.b #H'07,r0l ;+-は押されていた? bnemain_04 ;ne = 押されていた? add.w #D'3,r2 ;+-番号 +3 rotr.b r11 ;次のスキャンへ dec.b r1h ;ル-ブ@数 -1 rts bnemain_02 bramain_02 bramain_03 btst #0,r0l ;bit0 チェック btst #1,r0l ;bit1 チェック beqmain_06 inc.w #1,r2 ;+-番号 +1 bramain_08 btst #1,r0l ;bit1 fェック bramain_08 inc.w #1,r2 ;+-番号 +2 main_08: m		mov.b	@PDR1,r0I	;キー読み込み			
and.b #H'07,r01 ;キーは押されていた? mov.b #H'F0,r01 ;ボーh 1 イi2ytライズ bne _main_04 ;ne = 押されていた mov.b r01,@PCR1 ;ボーh 1 4i2ytライズ mov.b r01,@PCR1 ;ボ-h 1 4i2ytライズ mov.b r01,@PCR1 ;ボ-h 1 4i2ytライズ mov.b r01,@PCR6 ;ボ-h 6 fi2ytライズ mov.b r01,@PCR6 ;ボ-h 6 fi2ytライズ mov.b r01,@PCR6 ;ボ-h 6 fi2ytライズ mov.b r01,@PCR6 ;ボ-h 60-67:out rts bne _main_02 bra _main_02 bra _main_02 btst #0,r01 ;bit0 fiyy bne _main_08 btst #1,r01 ;bit1 fiy beq _main_06 inc.w #1,r2 ;t-番号 +1 bra _main_08 _main_06: _main_06: _main_06: _main_08: _main_08: _xor.w e2,e2 : + 番号 +2 _amain_08: _xor.w e2,e2 : + 番号 +2		not	r0l	;反転	INIPIO:		
bne       _main_04       ;ne = 押されていた       mov.b       r01,@PCR1       ;ホ -h 14-17:out         add.w       #D'3,r2       ;+番号 +3       mov.b       #H'FF,r01       ;ň -h 64       fightspic(1)         add.w       #D'3,r2       ;+番号 +3       mov.b       r01,@PCR1       ;ň -h 64       fightspic(1)         add.w       #D'3,r2       ;+番号 +3       mov.b       r01,@PCR6       ;ň -h 64       fightspic(1)         add.w       #D'3,r2       ;h-J' 回数 -1       mov.b       r01,@PCR6       ;ň -h 60-67:out         dec.b       r1h       ;ル-J' 回数 -1       rts       mov.b       r01,@PCR6       ;ň -h 60-67:out         bne       _main_02		and.b	#H'07,r01	;キーは押されていた?	mov.b	#H'F0,r01	;ポート 1 イニシャライズ
add.w #D'3,r2 ;+-番号 +3 mov.b #H'FF,r01 ;ボ-h 6 12シャライズ rotr.b r11 ;次のスキャンへ dec.b r1h ;ル-ブ回数 -1 mov.b r01,0PCR6 ;ボ-h 60-67:out mov.b r01,0PCR6 ;ボ-h 60-67:out rts TIMER1m: mov.l #H'D02,er6 inc.w #1,r01 ;bit1 fry/ beq _main_08 _main_0		bne	_main_04	;ne = 押されていた	mov.b	rOI,@PCR1	;ポート 14-17:out
rotr.b r11 ;次のスキャンヘ dec.b r1h ;ループ回数 -1 bne _main_02 bra _main_10 ;ループ級了 _main_04: ;キ-検出 mov.b r01,0PCR6 ;ポ-ト 60-67:out rts TIMER1m: mov.l #H'D02,er6 7000: dec.l #1,er6 bne ?000 bne _main_08 btst #1,r01 ;bit1 チェック beq _main_06 inc.w #1,r2 ;+-番号 +1 bra _main_08 _main_06: inc.w #2,r2 ;+-番号 +2 _main_08: xor.w e2,e2 ; trase +2		add.w	#D'3,r2	;キー番号 +3	mov.b	#H'FF,rOI	;ポート 6 イニシャライズ
dec.b       r1h       ; ル-プ回数 -1       rts         bne       _main_02          bra       _main_10       ; ル-プ終了       TIMER1m:         _main_04:       ; キ-検出       mov.l       #H'D02,er6         btst       #0,r01       ; bit0 fɪyp       ?000:          btst       #0,r01       ; bit1 fɪyp       ?000:          btst       #1,r01       ; bit1 fɪyp       dec.l       #1,er6         btst       #1,r2       ; +m番号 +1       bne       ?000         bra       _main_06       rts           _main_06:		rotr.b	r1l	;次のスキャンへ	mov.b	r01,@PCR6	;ポート 60-67:out
bne       _main_02         bra       _main_10       ; \n-7° 終了         _main_04:       ; \n-7° 終了         _main_04:       ; \n-7° 終了         btst       #0,r01       ; bit0 f1yp         bne       _main_08         btst       #1,r01       ; bit1 f1yp         beq       _main_06         inc.w       #1,r2       ; +-番号 +1         bra       _main_08         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_06:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         _main_08:       .data.b       H'71       ; F		dec.b	r1h	; <b>ルーフ</b> ゚回数 -1	rts		
bra _main_10       ; ル-プ終了       TIMER1m:         _main_04:       ; キ-検出       mov.l #H'D02,er6         btst       #0,r01       ; bit0 fryp         bne       _main_08       @dec.l #1,er6         btst       #1,r01       ; bit1 fryp         beq       _main_06       rts         inc.w       #1,r2       ; +一番号 +1         bra _main_08       SEG_TBL:       ; ½ヴ メントデ -タ テ -ブル         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_06:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         .data.b       H'71       ; F		bne	_main_02				
_main_04:       ; ++検出       mov.l #H'D02,er6         btst       #0,r01 <td; bit0="" f1yp<="" td="">       ?000:         bne       _main_08       dec.l #1,er6         btst       #1,r01       <td; bit1="" f1yp<="" td="">       bne         beq       _main_06       rts         inc.w       #1,r2       ; +-番号 +1         bra       _main_08       SEG_TBL:       ; tb' xン h5' - 4 7 - 7' h         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_08:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         .data.b       H'71       ; F</td;></td;>		bra	_main_10	;ループ終了	TIMER1m:		
btst       #0,r01       ;bit0 f1yp       ?000:         bne       _main_08       dec.l       #1,er6         btst       #1,r01       ;bit1 f1yp       bne       ?000         beq       _main_06       rts       rts         inc.w       #1,r2       ;f+番号 +1       seG_TBL:       ;tp* x>h5* -9 7-7*         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_08:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         .data.b       H'71       ; F	_mair	_04:		;キー検出	mov.l	#H'D02,er6	
bne       _main_08       dec.l #1,er6         btst       #1,r0l       ;bit1 f1yp       bne       ?000         beq       _main_06       rts       rts         inc.w       #1,r2       ;f+番号 +1       SEG_TBL:       ;tyf xyhテ -9 テ-7 ル         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_08:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         _main_08:       .data.b       H'71       ; F		btst	#0,r01	;bit0 ቻェック	?000:		
btst       #1,r0l       ;bit1 f1yp       bne       ?000         beq       _main_06       rts         inc.w       #1,r2       ;f-番号 +1         bra       _main_08       SEG_TBL:       ;tp* x>hテ* -9 テ -7 ル         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         _main_08:       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         _wor.w       e2,e2       .data.b       H'71       ; F		bne	_main_08		dec.I	#1,er6	
beq     _main_06     rts       inc.w     #1,r2     ;+番号 +1       bra     _main_08       _main_06:     .data.b     H'3F,H'06,H'5B,H'4F,H'66     ; 0 1 2 3 4       .main_06:     .data.b     H'3F,H'06,H'5B,H'4F,H'66     ; 0 1 2 3 4       .main_08:     .data.b     H'6D,H'7D,H'07,H'7F,H'67     ; 5 6 7 8 9       .main_08:     .data.b     H'77,H'7C,H'39,H'5E,H'79     ; A b C d E       .xor.w     e2,e2     .data.b     H'71     ; F		btst	#1,r0I	;bit1 ቻェック	bne	?000	
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		beq	_main_06		rts		
bra _main_08       SEG_TBL:       ; たグ・メントデ・-ダ・テーブル         _main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         inc.w       #2,r2       ; 非番号 +2       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         _xor.w       e2,e2       .data.b       H'71       ; F		inc.w	#1,r2	;キー番号 +1			
_main_06:       .data.b       H'3F,H'06,H'5B,H'4F,H'66       ; 0 1 2 3 4         inc.w       #2,r2       ; f+番号 +2       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         _xor.w       e2,e2       .data.b       H'71       ; F		bra	_main_08		SEG_TBL:	; セク゛:	メントテ゛ータ テーフ゛ル
inc.w       #2,r2       ;非番号 +2       .data.b       H'6D,H'7D,H'07,H'7F,H'67       ; 5 6 7 8 9         _main_08:       .data.b       H'77,H'7C,H'39,H'5E,H'79       ; A b C d E         xor.w       e2,e2       .data.b       H'71       ; F	_mair	_06:			.data.b H'3	F,H'06,H'5B,H'4F,	H'66 ; 0 1 2 3 4
_main_08: xor.w e2,e2 .data.b H'77,H'7C,H'39,H'5E,H'79 ; A b C d E .data.b H'71 ; F		inc.w	#2,r2	;キー番号 +2	.data.b H'6	D,H'7D,H'07,H'7F,	H'67 ; 5 6 7 8 9
xor.w e2,e2 .data.b H'71 ; F	_mair	_08:			.data.b H'7	7,H'7C,H'39,H'5E,	H'79; A b C d E
		xor.w	e2,e2		.data.b H'7	1	; F

<matrixkey\_2>

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 700~2300 µ sec 程度です。中 点を保持したいならば 1500 µ sec 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 $\phi$ /512'=25.6 $\mu$  sec)=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
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:  $\phi/4$  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 1500  $\mu$  sec 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、

 $1500 \,\mu \, \text{sec} \div (\phi/4 = 200 \,\text{nsec}) = 7500$ 

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 $\mu$  sec 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。1 $\mu$  sec=5 カウント(1 $\mu$  sec÷( $\phi$ /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。

;-----; :TK3687servo\_00.src FILE ; :Thu, Nov 25, 2004 ; DATE 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 INTRTZO .export 定数 ;----- シーケンス関連 ------.equ D'64 ;シーケンスデータ数 SEQMAX\_CNST SEQTMNG\_CNST .equ H'0500 ;シーケンス更新タイミング(BCD データ・msec 指定) \*\*\*\*\*\* メインプログラム \* リストにアド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 לבאָקלג bsr INITTZ0\_0C:16 ;TimerZ0 ביא<sup>°</sup> דעיד bsr INITINTR:16 ;割り込み イニシャライズ mov.w#WORK\_AREA,r3 ;ワークエリア クリア mov.w#WORK\_AREAE-WORK\_AREA-1,r1 xor.br01,r01 jsr @FILL mov.w@SEQDT TBL0.r0 mov.wr0,@FTIODO\_CNST mov.w@SEQDT\_TBL1,r0 mov.wr0,@FTIOCO\_CNST andc #H'7F,CCR ;割り込み許可 ;----- メインループ -----LOOP\_00: mov.b@TB1\_FG,rOI ;TimerB1 割り込み検出? bne L00P\_10:16 bra LOOP\_00 ;----- TimerB1 1ms 割り込み検出 ------L00P\_10: xor.br01,r01 mov.brOl,@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グ bcc LOOP\_13 L00P\_11: and.b#H'OF,rOI ;10msec 経過ならサーボ出力へ beq LOOP\_14 bra LOOP\_16 L00P\_13: ; シーケンスの変更

; カウンタ りリア

xor.wr0,r0

mov.wr0,@TB1\_CNT bsr SEQCNTINC:16 : シーケンスホ゜インタ インクリメント ; シーケンスデータテーブルからデータ取得 shll.w r0 xor.we0,e0 mov.ler0,er1 mov.w@(SEQDT\_TBL0,er1),r0 mov.wr0,@FTIODO\_CNST mov.w@(SEQDT\_TBL1,er1),r0 mov.wr0,@FTIOCO\_CNST L00P\_14: bsr INITTZ0\_0C:16 mov.w@FTIODO\_CNST,r0;ジェネラルレジスタ: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@FTIOCO\_CNST,r0;ジェネラルレジスタ:GRC\_0 設定 xor.we0,e0 mov.w#5.r1 ;x5 TZ0:CLK/4 1usec=5count mulxu.w r1,er0 mov.wr0,@GRC\_0 mov.b@TOCR,r01 ;FTIOD0 High 出力 or.b #H'0C,r01 mov.b@TSTR,rOh ;TZ:ch0 X9-1 bset #0,r0h mov.br01,@TOCR mov.br0h,@TSTR mov.b#B'11111000,r01 mov.br01,@TIER\_0 L00P\_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;割り込みフラグレジスタ2 TB1 クリア mov.br01,@IRR2 mov.b#H'01,r01 ;割り込みフラグ セット mov.brOl,@TB1\_FG pop.ler0 rte :-----タイマZ0割り込み INTRTZ0: push.l er0 mov.b@TSR\_0,ROL; IMFD /// mov.b#B'11100111,r0I mov.bROL,@TSR\_0 pop.ler0 rte サブルーチン

```
データで埋める
:
;R3=アト・レス / R1=デ・ータ数 / ROL=デ・ータ
FILL:
    mov.br0l,@r3 ;アドレス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,r01
    daa r01
    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,r01;TimerB1 割り込み許可
    mov.br01,@IENR2
    rts
    ÷ -
    タイマZ0イニシャライズ : GRD_0 アウトプット コンペアマッチ
;
INITTZO OC:
    mov.b@TSTR,r01
                     ;TZO Stop
    bclr #0.r01
    mov.br01,@TSTR
                     ; TCNT_0 ካሀፖ
    xor.wr0,r0
    mov.wr0,@TCNT_0
    mov.b#B'00000010,r01;TCNT クリア:禁止 CLK*4
    mov.br01.@TCR 0
    mov.b#B'10011001,r01;GRD_0,GRC_0 コンヘ アマッチ Low 出力
    mov.br01,@TIORC_0
                     ;FTIOD0,C0 Low
    mov.b@TOCR.rOl
    and b#H'F3 r01
    mov.br01,@TOCR
    mov.b@TOER,rOI
                     ;FTIOD0,C0 出力許可
    and.b#H'F3.r0I
    mov.br01,@TOER
    rts
:----
    タイマ B 1 イニシャライズ
INITTB1:
    mov.b#B'11111010,r0l; オートリロート, CLK/512
    mov.br01,@TMB1
                         ;リロード値 CLK/512x39=1msec
    mov.b#D'255-39,r01
```

```
mov.br01,@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
            D'1270, D'1270, D'1690, D'1690, D'1270
    .data.w
            D'1270, D'1690, D'1690, D'1270, D'1270
    .data.w
;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
            D'1060, D'1060, D'1060, D'1900, D'1900
    .data.w
;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
             D'1900, D'1900, D'1900, D'1060, D'1060, D'1060
    .data.w
    .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
            D'1060, D'1060, D'1060, D'1480, D'1480, D'1480
    .data.w
    .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・レスを表したい時は locate を有効にする
    .section D,data,locate=H'F780
WORK_AREA:
;----- 動作モード ------
SEQ_CNT: .res.w 1
                          ;シーケンス番号
;----- 917 B1 -----
TB1_FG: .res.b 1
                          :TimerB1 割り込みフラグ
    .align 2
TB1_CNT: .res.w 1
                           ;パルス出力間隔 カウンタ
;----- 917 Z0 -----
GRDO_CNT: .res.w 1
FTIOCO_CNST: .res.w
                                :FTIOCO 出力時間
                      1
FTIODO_CNST:
             .res.w
                      1
                                ;FTIOD0 出力時間
WORK_AREAE:
.end
```

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、

 $11001100b(CCh) \rightarrow 01100110b(66h) \rightarrow 00110011b(33h) \rightarrow 10011001b(99h) \rightarrow 11001100b(CCh) \cdots$ 

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、ただー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

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 100 μ sec としました。100 μ sec 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 $\phi / 16$ ・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)

 $TCORA = 125d(\phi \swarrow 16 \times 125 = 100 \,\mu \text{ sec})$ 

#### 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	1ステップ毎の間隔を指定	例えば値を 60 とすると、60×100 µ sec=6msec の間隔
(パルスの出力間隔)	間隔=割り込み間隔(100 µ sec)×値	で1ステップづつ動きます。
角度	動かす角度を指定	例えば1ステップ 1.8°のパルスモータで 90°動かす場
(ステップ数)	角度=1ステップ辺りの角度×ステップ数	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、

 $1 \sec = 50 \times 100 \,\mu \sec \times n$ 

 $n = 1 \sec (50 \times 100 \,\mu \sec)$ 

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	.export .export	_main INTRTV	
; DATE :Tue, Oct 19, 2004 ; DESCRIPTION :Main Program	; メインプロ		****
; CPU TYPE :H8/3687 ;	, .section	P,code;,locate=H'	EA00
; This file is generated by Hitachi Project Generator (Ver.2.1 ; and programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.	_main:		
; ;	bsr bsr	INITTV:16 INITPI0:16	;917 V 1=>+ラ1ズ ;PIO 1=>+ラ1ズ
.include "io3687F_equ.inc"	mov.w mov.w	#WORK_AREA,r3 #WORK_AREAE-WORK_	; <b>ባ-ሳェリア ሳリア</b> AREA-1,r1

	bsr	FILL:16		mov.w	v (	(PMSD_TBL,er0)	, r1	;回転角度(ステップ数)のセット
	mov b	#∐'33 r0l	・11°    75-タヘの出力データ セット	MOV.W	V	r1,@PMSTEP_CNT		
	mov.b	r01,@PMDT	,л ихт-э, хоэщэээ -э сэн	INTRTV_FF:		10 0 <b>700</b> 01		
				bclr	7	#6,@ICSRV		
	MOV.W	@PMSD_TBL+0,r0	;回転万问初期值 セット					
	MOV.W	r0,@R01_FLG		pop.I	l 6	er1		
	MOV.W	@PMSD_TBL+2,r0	;四転速度初期値でツト	pop.I	I €	erO		
	MOV.W	r0,@IVWAII_CNI		rte				
	MOV.W	r0,@IVWAII_CNSI			******	*****	*****	****
	MOV.W	@PMSD_IBL+4,r0	;回転用度(メテッノ 釵)初期値 セット	;······	·····			
	MOV.W	ru,@PMSTEP_CNT		; サノル	ルーナン	/ • • • • • • • • • • • • • • • • • • •	*****	****
			· \$\$110`} 7.75 7° "	,				
	andc	#H'/F,CCR	;割り込み1ィーノル	;		·····		
	J Z M	-		;	RUX	6		
;	- メイノル	- )		;				
LUUP_	_00:			X6:		-0 -0		
	DIA	L00P_00		XOL.W	ve		, Z	- EP1_EPO
. * * * *	******	****	*****	n.vom			, Z	, ER 1=ERU
	実用パンコの	τœ		adu. I		ero, ero	, Z . 2	, ERU+ERU=2ERU - 2EPO+EP1(_EPO)_2EPO
, .* * * *	***********	** <b>*</b> ********	*****	add. I			, 2 • 2	· 3ER0+3ER0-6ER0
, 				auu.i		510,610	, Z • Q	, JERUTJERU=OERU
;	タイマV割	  り込み		113			,0	
;	 V:			;	 データ	 'で埋める		
	push.l	er0		;				
	push.I	er1		;R3 = 71 t	<i>ل</i> ر			
				; R1 = $\bar{\tau}^* - 4$	夕数			
	mov.w	@TVWAIT CNT,r0	:回転速度の管理	; ROL = $\overline{\tau}^*$ -	9			
	dec.w	#1,r0	,	FILL:				
	mov.w	r0,@TVWAIT_CNT		mov.b	)	rOl,@r3	; ROL	データで埋める
	bne	INTRTV_FF	;カウントアップしていなければ終了	inc.w	v ‡	#1,r3	,	
		-	,	dec.w	v ‡	¥1, r1		
INTRT	V_10:		; カウントアップ	bne	F	FILL		
	mov.b	@PMDT,r01	;パルスモータを 1 ステップ動作	rts				
	mov.w	@ROT_FLG, r1	;ROT_FLG=0 : 停止					
	beq	INTRTV_14	; =1:左示-ト	;				
	cmp.w	#2,r1	; =2 : 右厅-ト	; タイ <b>マ</b>	ママイニ	ニシャライズ		
	beq	INTRTV_12		;				
	rotl.b	r01		INITTV:				
	bra	INTRTV_14		mov.b	) ‡	#D'125,r0l	;CLK*	16*125=100usec
INTRT	V_12:			mov.b	)	rOI,@TCORA		
	rotr.b	r0l		mov.b	o ‡	#B'01001010,r0l	;CLK*	*16 RA コンヘ゜アマッチ INTR&CLR
INTRT	V_14:			mov.b	<b>)</b>	rOI,@TCRVO		
	mov.b	rOI,@PMDT		rts				
	mov.b	r01,@PDR3	;パルスモータへ出力					
				;				
INTRT	V_20:		;ステップ数の管理	; PIC	ロイニシ	シャライズ		
	MOV.W	<pre>@PMSTEP_CNT,r0</pre>		;				
	dec.w	#1,r0		INITPIO:				
	MOV.W	r0,@PMSTEP_CNT		mov.b	<b>)</b>	#H'FF,rOI	;PI03	3 all Output
	beq	INTRTV_30	;規定ステップ数終えたら次へ移行	mov.b	) (	rOI,@PCR3		
	MOV.W	@TVWAIT_CNST,r0	;回転速度値を再セット	rts				
	MOV.W	rO,@TVWAIT_CNT	; 規定ステップ数まで繰り返す					
	bra	INTRTV_FF		· * * * * * * * * * * * * * * * * * * *	*****	*****	****	*******
				;	パルス	モータ出力値デ・	ータテ	ーブル・エリア
INTRT	V_30:		;次のシーケンスへ	• * * * * * * * * * * * * * * * * * * *	*****	*****	****	*****
	MOV.W	<pre>@PMSEQ_CNT, r0</pre>	;シーケンス番号の更新	; <i>&gt;</i> -!	・ケンスディー	タテーブル		
	inc.w	#1,r0		;.data.w	D'1	;回転方向0:停止	:/1:左	.ロテート/2:右ロテート
	cmp.w	#PMSD_SIZE,r0		;.data.w	D'60	;回転速度 100use	ec * (	60 = 6msec
	bcs	INTRTV_32		;.data.w	D'50	;回転角度(ステップ	数) 1	.8 * 50 = 90 °
	xor.w	r0,r0	;全シーケンスを終えたら元に戻る	PMSD_TBL:		;PulseMoto	rSequ	enceData Table
INTRT	V_32:			.data	a.w [	D'1,D'60,D'50		
	MOV.W	r0,@PMSEQ_CNT		.data	a.w [	D'0,D'1000,D'5	;wait	100msec*5=500msec
				.data	a.w [	D'1,D'60,D'50		
	bsr	X6:16	;シーケンスデータを読み出す	.data	a.w [	D'0,D'1000,D'5	;wait	100msec*5=500msec
	MOV.W	@(PMSD_TBL,er0)	,r1 ;回転方向のセット	.data	a.w [	D'1,D'60,D'50		
	MOV.W	r1,@ROT_FLG		.data	a.w [	D'0,D'1000,D'5	;wait	100msec*5=500msec
	inc.l	#2,er0		.data	a.w [	D'1,D'60,D'50		
	MOV.W	@(PMSD_TBL,er0)	,r1 ;回転速度のセット	.data	a.w [	D'0,D'1000,D'5	;wait	100msec*5=500msec
	MOV.W	r1,@TVWAIT_CNT						
	MOV.W	r1,@TVWAIT_CNST		.data	a.w [	D'2,D'60,D'50		
	inc.	#2.er0		data	a.w [	)'0.D'1000.D'5	wait	100msec*5=500msec

.data.w	D'2,D'60,D'50		WORK_AF
.data.w	D'0,D'1000,D'5 ;wait 100msec*5=500msec		;
.data.w	D'2,D'60,D'50		PMDT:
.data.w	D'0,D'1000,D'5 ;wait 100msec*5=500msec		. 6
.data.w	D'2,D'60,D'50		ROT_FLG
.data.w	D'0,D'1000,D'5 ;wait 100msec*5=500msec		PMSTEP_
PMSD_TBLE:			PMSEQ_C
PMSD_SIZE	.equ (PMSD_TBLE-PMSD_TBL)/6		
			;
		1	TVWAIT_
. * * * * * * * * * * * * * * * * * * *	***************************************		TVWAIT_
; デー	タ・エリア		WORK_AF
• * * * * * * * * * * * * * * * * * * *	***************************************		
.section	D,data;,locate=H'F780		;======

WORK_AREA:			
; パルスモータ			
PMDT:	.res.b	1	;パルスモータ出力データ
.align	2		
ROT_FLG:	.res.w	1	;回転方向フラグ
PMSTEP_CNT:	.res.w	1	; ステップ数カウンタ
PMSEQ_CNT:	.res.w	1	;シーケンス番号
; 91₹ V			
TVWAIT_CNT:	.res.w	1	;wait <u>አ</u> ሳንዎ
TVWAIT CNST:	.res.w	1	;wait カカンタ初期値
WORK_AREAE:			
_			
:=================			
ond			
	WORK_AREA:         ; パ、ルスモータ         PMDT:         .align         ROT_FLG:         PMSTEP_CNT:         PMSEQ_CNT:         ; タイマ V         TVWAIT_CNT:         TVWAIT_CNST:         WORK_AREAE:         ;	WORK_AREA:         ; パ*ルスモータ         PMDT:       .res.b         .align 2         ROT_FLG:       .res.w         PMSTEP_CNT:       .res.w         PMSEQ_CNT:       .res.w         ;	WORK_AREA:         ; Il® INZE-9         PMDT:       .res.b         .align       2         ROT_FLG:       .res.w       1         PMSTEP_CNT:       .res.w       1         PMSEQ_CNT:       .res.w       1         ;       947 V

#### 8-6 I<sup>2</sup>C バス仕様 EEPROM のリード/ライト

TK-3687 には I<sup>2</sup>C 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 I<sup>2</sup>C バスを使用して EEPROM のリードライトを行ないます。

I<sup>2</sup>C 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。

I<sup>2</sup>C 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 I<sup>2</sup>C 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。

### ■ 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
<b>▲</b> 1	デバイ: 0	<b>ג⊐—וּ</b> 1	0	<b>◄</b> •	<b>ページアドレス</b> P1	P0	リ <b>ード/ライト</b> R/W
		<b>7</b>	• • • • •				

【 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。

#### ■ 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・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:I<sup>2</sup>C バスコントロールレジスタ 2

7	6	5	4	3	2	1	0
BBSY	SCP	SDAO	SDAOP	SCLO	_	I <sup>2</sup> CRST	_

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。 ■ ICSR: I<sup>2</sup>C 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:I<sup>2</sup>C 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:I<sup>2</sup>C 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
$\mathrm{D7}$	D6	$\mathrm{D5}$	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 I<sup>2</sup>C 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:I<sup>2</sup>C 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	$\mathrm{D5}$	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 I<sup>2</sup>C 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

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 ヘリード/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 ヘデータを書き込む
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	0	ライト時・書き込むデータ元アドレス
	2	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。

;	· ************************************
	.section P.code.locate=H'EA00 ; locate で絶対アドレス
: FILE :sampleIIC 00.src	
DATE Wed Dec 22, 2004	main:
DESCRIPTION Main Program	イニシャライズ
· CDI TVDE ·H8/3687	$\int \frac{1}{\sqrt{2}} \frac{1}{$
· · · · · · · · · · · · · · · · · · ·	
, This file is generated by Hitachi Project Generator (Ver. 2.1	
and Programed by Toyo-Linx Co. Ltd. / Y Furukawa	
	IIIOV.W #WORK_AREAE-WORK_AREA-1,11
,	DSF FILL:16
, I2Cリスドノログノム - T2Cリスドグの「CDDON」にデータを書き込む。 日、フドレスを	
, IZCハスILI 你UPEPRUM にナークを書さ込み、 向一アトレスを 、	
; 読み込んでホートに表示する。	mov.b #WRITEDATA_CNST,r01 ;書き込みデータセット
;	mov.b r01,@WR_DATA
	; ΕΕΡROM の <i>/ JP</i>
.include "io3687F_equ.inc"	mov.b #SLAVEADDR_CNST,rOI ;スレーブアドレス セット
.export _main	mov.b rOI,@SLAVE_ADDR
	mov.w #WRITEADDR_CNST,r0 ;EEPROMアドレス セット
·*************************************	mov.w r0,@EEPR_ADDR
;  定数	mov.w #H'800,r0 ;データ数
. ************************************	mov.w r0,@SQDT_SIZE
SLAVEADDR_CNST .equ H'AO ;bit7-4 : Device Type Identifier	bsr EEPRCLR:16 ;EEPROM のクリア
; 1010 = EEPROM	; EEPROM へ書き込み
;bit3-1 : Chip Enable	mov.b #SLAVEADDR_CNST,rOI ;スレーフ゛アト゛レス セット
; 000 = EEPROM pin3-1 ADDR	mov.b rOI,@SLAVE_ADDR
;bit0 : Read or Write	mov.w #WRITEADDR_CNST,r0 ;EEPROMアドレス セット
; (1) (0)	mov.w r0.@EEPR ADDR
WRITEADDR_CNST .equ H'0000 ;書き込み先アドレス	mov.w #WR DATA.r0 :書き込すシデータ元アドレス
WRITEDATA CNST .equ H'55 :書き込みデータ	mov.w r0.@SQDT ADDR
READADDR CNST .egu H'0000 :読み込み先アドレス	mov w #D'1.r0 ·元 - 夕数
	mov w r0 @SODT SIZE
	hsr FFPRSOWR:16 ·シーケンシャル書き込み
. *************************************	· FEBROW から読み出し
, ・ メインプログラム	mov b #SLAV/FADDR CNST r01 · 71,-7 <sup>*</sup> 7k <sup>*</sup> 1.7 t <sub>2</sub> ·k
, , , , , , , , , , , , , , , , , , , ,	

mov.b mov.w mov.w mov.w mov.w mov.w bsr mov.b mov.b bclr btst beq bset main_FF:	rOI, @SLAVE_ADDR #READADDR_CNST, rO, @EEPR_ADDR #RD_DATA, rO rO, @SQDT_ADDR #D'1, rO rO, @SQDT_SIZE EEPRSQRD:16 @RD_DATA, rOI rOI, @PDR1 #O, @PDR2 #3, rOI main_FF #O, @PDR2	r0 ;EEPROM アドレス セット ;読み出したデータのセット先 ;データ数 ;シーケンシャル読み出し ;データをポートに出力する
bra	\$ ,EEPROM 05 ;ので、ル−	青さ皆ん回数には限りかめる −プせずにここで停止
· * * * * * * * * * * * * * * * * * * *	* * * * * * * * * * * * * * * * * * * *	*******
; 割り込み	处埋 * * * * * * * * * * * * * * * * * * *	****
,		
. * * * * * * * * * * * * * * * * *	*****	*******
; サブルー	チン	
• * * * * * * * * * * * * * * * * * *	* * * * * * * * * * * * * * * * * * *	*******
;		川 書之:) フ;
; EEP	KUM ヘシーケンシャ	ル者さ込み
	 	·ーゴ)マドレフ
· FEPR ADD	R · FEPROM アドレ	7
	R . CEFROM アドレ D ・ まきひかデー	タのテアドレフ
	「 ・ 書き込むテーク	クリカプトレス
, SQDI_SIZ	こ 音さ込むナー	Σ \$X
,		
EEPROQUE OO		
LLFN30mh_00. 圭 <b>主</b> 场·	え 動作 終了 2	
,····· 百〇 探/ FFPRSOWR 01·		
hsr	WRCHK12C-16	・書き換え動作終了チェック
bon	EEPRSOWR 01	
mov b	r11.r11	
bne	EEPRSOWR 01	
	ou01	
;ページ	ライト	
EEPRSQWR_10:		
bsr	INITI2C:16	; 1ニシャライス゛
mov.w	@EEPR_ADDR,r0	
mov.w	r0,@ROM_ADDR	
EEPRSQWR_11:		
bsr	STRTI2C:16	; スタートコンディション
bcs	EEPRSQWR_11	
EEPRSQWR_12:		
bsr	DASI2C:16	; デ バ イスアド レス
bcs	EEPRSQWR_12	
EEPRSQWR_13:		
bsr	WAITACKI2C:16	;アクノリッジ待ち
bcs	EEPRSQWR_13	
bne	EEPRSQWR_20	
EEPRSQWR_14:		
bsr	MASI2C:16	; メモリアドレス(ワードアドレス)
bcs	EEPRSQWR_14	
mov.w	<pre>@EEPR_ADDR, r2</pre>	
MOV.W	<pre>@SQDT_ADDR,r3</pre>	
mov.w	#0,e3	
MOV.W	<pre>@SQDT_SIZE,r4</pre>	
mov.b	#16,r5l	;1 ページのサイズ
EEPRSQWR_15:		
mov.b	@er3,r01	

EEPRSQWR_16: bsr	WDS12C:16	;データライト
bcs	EEPRSQWR_16	
inc.l	#1,er3	
dec.w bea	#1,r4 EEPRSQWR 17	
dec.b	r5l	;1 ページ分ライト完了?
bne EEPRSQWR_17:	EEPRSQWR_15	; NO 9 777
MOV.W	r2,@EEPR_ADDR	
mov.w	r4,@SQDT_SIZE	
EEPRSQWR_18:		
bsr	STOPI2C:16	; גראר גראר גער גער גראר גער גער גער גער
DUS	LLFNJQWN_10	
mov.w bne	<pre>@SQDT_SIZE, r0 EEPRSQWR_00</pre>	;全データライト完了? ・No ジャンプ
andc nop	#H'FE,CCR	;0K : Cy=0
rts		
EEPRSQWR_20:		
EEPRSQWR_21: bsr	STOP12C:16	; ストップ コンデ ィション
bcs	EEPRSQWR_21	·NC · Cv_1
nop	#H UI,CCK	,NO . Cy=1
rts		
;		
; EEPRO;	™からのシーケン 	ンヤル読み山し 
; EEPRO ; ; SLAVE_ADDR : EEPR ADDR	M からのシーケン  : デバイス(スレ : EEPROM アドレ	シャル読み団し  ープ)アドレス ス
; EEPRO; ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデー	シャル読み出し ープ)アドレス ス - タのセット先アドレス
; EEPR( ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ;	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー	シャル読み出し ープ)アドレス ス -タのセット先アドレス 9数
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD : FEPRSQRD 00:	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーク	シャル読み出し ープ)アドレス ス - タのセット先アドレス 9数
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD: EEPRSQRD_00: ;書き換え	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー? 動作終了?	シャル読み⊡し ープ)アドレス ス -タのセット先アドレス 9数
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーク 動作終了?	シャル読み団 ープ)アドレス ス -タのセット先アドレス 9数 
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ; 書き換え EEPRSQRD_01: bSr bCS	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー 動作終了? WRCHKI2C:16 EEPRSQRD_01	シャル読み出し ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; SQDT_SIZE ; EEPRSQRD_00: ; 書き換え EEPRSQRD_01: bsr bcs mov.b bne	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー 動作終了? WRCHKI2C:16 EEPRSQRD_01 r11,r11 FEPRSQRD_01	シャル読み団 0 ープ)アドレス ス - タのセット先アドレス 9数 
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー 動作終了? WRCHKI2C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01	シャル読み団 0 ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bcs mov.b bne ;シーケン EEPRSQRD_10:	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデー 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード	シャル読み団 - プ)アドレス ス - 夕のセット先アドレス 9数 ;書き換え動作終了チェック
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ;	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーク 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16	シャル読み団 0 ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック ;イニシャライス
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bcs mov.b bne ;シーケン EEPRSQRD_10: bsr mov.w mov.w	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR	シャル読み出し ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; SQDT_	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーク 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR	シャル読み回し ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bcs mov.b bne ;シーケン EEPRSQRD_11: bsr mov.w mov.w EEPRSQRD_11: bsr	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHKI2C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INITI2C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRTI2C:16	シャル読み団 U ーブ)アドレス ス -タのセット先アドレス 対数 ;書き換え動作終了チェック ;イニシャライズ ; イニシャライズ ; スタートコンディション
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_10: bsr mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_11: bsr bcs EEPRSQRD_11:	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11	シャル読み出し ープ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック ; イニシャライズ ; スタートコンディション
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; SQDT_	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーク 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11 DAS12C:16 EEPRSQRD_42	シャル読み出し ーブ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック ;イニシャライズ ;スタートコンディション ;デ パイスアドレス
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_11: bsr mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13:	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHKI2C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INITI2C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRTI2C:16 EEPRSQRD_12	シャル読み出し ーブ)アドレス ス -タのセット先アドレス 努数 ;書き換え動作終了チェック ;イニシャライズ ; イニシャライズ ; パーイスアド・レス
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_10: bsr mov.w mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13: bsr bcs	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11 DAS12C:16 EEPRSQRD_12 WAITACK12C:16 FEPRSQRD_13	シャル読み出し ーブ)アドレス ス -タのセット先アドレス 9数 ;書き換え動作終了チェック ;パニシャライズ ;パニシャライズ ;パーイスアドレス ;アクノリッジ待ち
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_11: bsr mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13: bsr bcs bcs bne	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11 DAS12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_13 EEPRSQRD_20:16	シャル読み出し ーブ)アドレス ス -タのセット先アドレス 対数 ;書き換え動作終了チェック ;イニシャライズ ;スタートコンディション ;デ バイスアトドレス ;アクノリッジ 待ち
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_10: bsr mov.w mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13: bsr bcs EEPRSQRD_13: bsr bcs bne EEPRSQRD_14: bsr	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11 DAS12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_13 EEPRSQRD_20:16 MAS12C:16	<ul> <li>マアル読み出し</li> <li>-ブ)アドレス</li> <li>ス</li> <li>-タのセット先アドレス</li> <li>&gt;数</li> <li>:書き換え動作終了チェック</li> <li>; 書き換え動作終了チェック</li> <li>; パニシャライズ</li> <li>; パニシャライズ</li> <li>; パーイスアト・レス</li> <li>; アウノリッジ・待ち</li> <li>: メモリアト・レス(ワート・アト・レス)</li> </ul>
; EEPRG ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_SIZE ; SQDT_	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_11 DAS12C:16 EEPRSQRD_13 EEPRSQRD_13 EEPRSQRD_20:16 MAS12C:16 EEPRSQRD_14	<ul> <li>シャル読み出し</li> <li>-ブ)アドレス</li> <li>ス</li> <li>-タのセット先アドレス</li> <li>&gt;数</li> <li>:書き換え動作終了チェック</li> <li>; (イニシャライズ</li> <li>; スタートコンディション</li> <li>; デパイスアドレス</li> <li>; アクノリッジ 待ち</li> <li>; メモリアドレス(ワードアドレス)</li> </ul>
; EEPRG ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_11: bsr mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13: bsr bcs EEPRSQRD_13: bsr bcs EEPRSQRD_14: bsr bcs mov.b	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHKI2C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャノレリード INITI2C:16 0EEPR_ADDR,r0 r0,0ROM_ADDR STRTI2C:16 EEPRSQRD_12 WAITACKI2C:16 EEPRSQRD_13 EEPRSQRD_13 EEPRSQRD_14 0SLAVE_ADDR,r0I	<ul> <li>マアル読み出し</li> <li>-ブ)アドレス</li> <li>ス</li> <li>-タのセット先アドレス</li> <li>&gt;数</li> <li>;書き換え動作終了チェック</li> <li>; イニシャライズ</li> <li>; パーイスアト・レス</li> <li>; アクノリッジ・待ち</li> <li>; メモリアト・レス(ワート・アト・レス)</li> <li>; リート・命令コート・に変更</li> </ul>
; EEPRC ; SLAVE_ADDR ; EEPR_ADDR ; SQDT_ADDR ; SQDT_ADDR ; SQDT_SIZE ; EEPRSQRD_00: ;書き換え EEPRSQRD_01: bsr bcs mov.b bne ;シーケン EEPRSQRD_10: bsr mov.w mov.w EEPRSQRD_11: bsr bcs EEPRSQRD_11: bsr bcs EEPRSQRD_12: bsr bcs EEPRSQRD_13: bsr bcs EEPRSQRD_14: bsr bcs mov.b bset mov.b	M からのシーケン : デバイス(スレ : EEPROM アドレ : 読み出したデ- : 読み出すデーイ 動作終了? WRCHK12C:16 EEPRSQRD_01 r11,r11 EEPRSQRD_01 シャルリード INIT12C:16 @EEPR_ADDR,r0 r0,@ROM_ADDR STRT12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_12 WAITACK12C:16 EEPRSQRD_14 @SLAVE_ADDR,r01 #0,r01 r01 @SLAVE_ADDR,r01	シャル読み出し ーブ)アドレス ス -タのセット先アドレス 9数 ; 書き換え動作終了チェック ; ポージャライズ ; スタートコンディジョン ; デ パイスアト・レス ; アクノリッジ・待ち ; メモリアト・レス(ワート・アト・レス) ; リート・命令コート・に変更

E	EPRSQRD_16:		
	bsr	RSTRT12C:16	;リスタートコンテ゛ィション
	bcs	EEPRSQRD 16	
F	EPRSORD 17		
	bor	DACIOC. 10	·=* #* /ファレ゙   フ
	DST	DAS120.10	, 7 // 12/1 /22
	bcs	EEPRSQRD_17	
E	EPRSQRD_18:		
	bsr	WAITACKI2C:16	・アクノリッジ待ち
	bec	EEDDSODD 19	,
	DUS	ELFROUND_10	
	bne	EEPRSQRD_20	
E	EPRSQRD_19:		
	bsr	MSTRMD12C:16	;マスタ受信モードに切り替え
	hcs	EEPRSORD 19	
	000		
	MOV.W	@EEPR_ADDR, r2	
	MOV.W	<pre>@SQDT_ADDR, r3</pre>	
	mov.w	#0.e3	
	mov w	@SODT_SIZE_r4	
		#1 r/	・デューク米ケー1 川、イトク
	Cilip.w	#1,14	, <b>J</b> = <b>J</b> = <b>J A A A</b>
	beq	EEPRSQRD_1b	; Yes
	bsr	RDSI2C:16	:データ受信指定
	mov b	@ICDPP rOI	がミニード。受信動作開始
	1100.0	GIODIAN, I'O'	,7、71,又旧到旧册如
E	EPRSQRD_1a:		
	bsr	RDDT12C:16	: データリード
	mov b	r1L @er3	
	inc.w	#1 r2	
	Inc.w	#1,12	
	inc.l	#1,er3	
	dec.w	#1,r4	
	CMD.W	#1.r4	:残り1バイト?
	bhi	EEDRSORD 12	
	biii		
	DST	LRUSI2C:16	,取於丁一》又后拍正
	bra	EEPRSQRD_1c	
E	EPRSQRD 1b:		
	her		- 是终于 - ゆ受信指定
	031		
	MOV.D	@ICDRR, FUI	;タ ミーリート,芆信動作用炤
E	EPRSQRD_1c:		
	bsr	RDFDT12C:16	:最終データリード
	mov h	r11 @or3	
		#4 #0	
	Inc.w	#1,12	
	INC.I	#1,er3	
	dec.w	#1,r4	
	mov.w	r2.@EEPR ADDR	
	mov w	r3 @SODT ADDR	
	110 <b>V</b> . W		
	MOV.W	14, @SQD1_SIZE	
	andc	#H'FE,CCR	;0K : Cy=0
	non		
	rte		
	113		
E	EPRSQRD_20:		
E	EPRSQRD_21:		
	bsr	STOP 12C:16	: ストップ コンデ ィション
	bee	EEDRSOPD 24	,
	005		
	orc	#H'UT,CCR	;NG : Cy=1
	nop		
	rts		
,	FEDD		
;	EEPRO		
;			
;	SLAVE_ADDF	≀: デバイス(スレ	/ーブ)アドレス
	EEPR ADDR	: EEPROM アドレ	ス
	SODT SIZE	クリアすスデ	
,	0001_012L		~ **
,			
E	EPROLR:		
	mov.b	#0,r01	;00h でクリアする
	mov.b	rOI,@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: WRCHKI2C:16 ;書き換え動作終了チェック bsr bcs EEPRFILL\_01 mov.b r11, r11 EEPRFILL\_01 bne ;..... ページライト ..... EEPRFILL\_10: bsr INITI2C:16 ;イニシャライズ mov.w @EEPR\_ADDR, r0 rO,@ROM\_ADDR mov.w EEPRFILL\_11: STRT12C:16 ; スタートコンデ ィション bsr EEPRFILL\_11 bcs EEPRFILL\_12: DASI2C:16 ; デ バ イスアド レス bsr EEPRFILL\_12 bcs EEPRFILL\_13: bsr WAITACKI2C:16 ; アクノリッシ 待ち bcs EEPRFILL\_13 EEPRFILL\_20 bne EEPRFILL\_14: bsr MASI2C:16 ; メモリアドレス(ワードアドレス) EEPRFILL\_14 bcs @EEPR\_DATA,rOI ;セットするデータ mov.b rol,@ROM\_DATA mov.b mov.w @EEPR\_ADDR, r2 mov.w @SQDT\_SIZE, r4 ;1 ページ のサイズ mov.b #16,r51 EEPRFILL\_15: EEPRFILL\_16: bsr WDS12C:16 ;データライト bcs EEPRFILL\_16 inc.w #1,r2 dec.w #1, r4 beq EEPRFILL\_17 ;1 ページ分ライト完了? dec.b r5l bne EEPRFILL\_15 ; No ジャンプ EEPRFILL\_17: mov.w r2,@EEPR\_ADDR mov.wV r4,@SQDT\_SIZE EEPRFILL 18: STOP12C:16 ; ストップ コンデ ィション bsr bcs EEPRFILL\_18 @SQDT\_SIZE,r0 ;全データライト完了? mov.w bne EEPRFILL\_00 ; No ジャンプ #H'FE,CCR andc ;0K : Cy=0 nop rts EEPRFILL\_20: EEPRFILL\_21: STOP12C:16 ; ストップ コンデ ィション bsr bcs EEPRFILL\_21 ;NG : Cy=1 orc #H'01,CCR nop rts

;開			; Cy=' RSTRTI2C:	1 : NG 送信禾完了	
;			mov.b	@ICSR,r0I	;送信完了チェック
;戻り値:Cy · Cv-	(キャリーフラク ) -0 ・ 0K		and.b	#H'CO,rOI #H'CO,rOI	;TDRE[7]=1,TEND[6]=1 なら完了
; Cy:	=1 : NG バスが開	放されていない(他で使用されてい	bne	RSTRTI2C_10	;送信未完なら Cy=1 で終了
STRT12C:			mov.b	#H'80,r01	;開始条件発行
mov.b	@ICCR2,r01	;バスは開放されているか?	mov.b	rOI,@ICCR2	; BBSY=1 / SCP=0
btst bne	#7,r01 STRT12C_10	; 1=Busy ; バスビジーなら Cy=1 で終了	andc nop	#H'FE,CCR	;0K : Cy=0
STRT212C:			rts		
mov.b mov.b	#B'10110101,r r0I,@ICCR1	이 ; रスタ送信モート	RSTRTI2C_10: orc	#H'01.CCR	:NG : Cv=1
mov.b mov.b	#H'80,r01 r01,@ICCR2	;開始条件発行 ; BBSY=1 / SCP=0	nop rts	·	
andc nop	#H'FE,CCR	;0K : Cy=0	; : デル	(イスアドレス設定	
rts			;		
CTDT 100 40.			DASI2C:		
orc	#H'01.CCR	NG CV=1	mov.b	@SLAVE_ADDR, ru @ROM_ADDR.r1	「, スレー) パト レスの11 F カス ・ ROM ADDR の A8, A9, A10 を
nop		,	shal.b	r1h	; SLAVE_ADDR Ø bit5,6,7 ^
rts			and.b	#H'0e, <b>r1</b> h	
			or.b	r0l,r1h	
; / / / / / / / / / / / / / / / /			mov.b	r1h,r1l	; 送信ァー9は R1L へ · 送信
;			bia	170120.10	, 20日
STOP12C:	ACTODIOC CTA	-01	;		
bne	STOP12C_STA,	101	;	1 トナーク設定	
			WDSI2C:		
mov.b	@ICSR,r01	;送信完了チェック ·TDPE171-1 TEND161-1 送信完了	mov.b	<pre>@ROM_DATA, r11 TYDL2C:16</pre>	;R1L=書き込みデータ ·洋信
xor.b	#H'CO, rOI	,IDKE[/]=1,IEND[0]=1 医信光 ]	Dia	170120.10	,这店
bne	STOP12C_10	;送信未完なら Cy=1 で終了	;	 	
xor.b	r01,r01	;停止条件発行	;	- 9 区旧 	
mov.b	rOI,@ICCR2	; BBSY[7]=0,SCP[6]=0	;引数 : R1L ;戻り値 : Cy	(送信データ) (キャリーフラグ)	
mov.b	@ICSR,r0I	; トランスミットエント゛ヒ゛ット クリア	; Cy=0		クジナフサンジニスナープ
mov.b	#6, r01 r01,@ICSR	; IEND=U	; Cy= TXDI2C:	1:NG 木达信テー	クかめる為达信でさ9 ・送信パッファけ容いているか?
mov.b	#1,r0l		btst	#7,r01	; いなければ Cy=1 で抜ける
mov.b	r01,@STOP12C_	_STA	beq	TXDI2C_10	
STOP12C_00:			mov.b	r1I,@ICDRT	;データ送信
mov.b btst	#3,r01	,1字正宗11+ (快山行ら) : STOP[3]=1 なら検出	ando	#H'FF.CCR	:0K : Cv=0
beq	STOP I 2C_10		nop	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	, , , , , , , , , , , , , , , , , , , ,
			rts		
xor.b mov.b	r01,r01 r01,@STOP12C_	STA	TXDI2C_10:		
bsr	WAIT4u:16	;4usec 待ち	orc	#H*01,CCR	;NG : Cy=1
andc	#H'FE,CCR	;Cy=0 : OK	rts		
nop rts			;;	 Eリアドレス設定	
			;		
STOPI2C_10:	#U'04 COD	· (),-1 · NC	MASI2C:		
nop	#H UT,CCR	;Cy=1 : NG	mov.w bsr	TXDI2C:16	;KI=ROMプトレム ・アドレスi送信
rts			bcs	MAS12C_10	;Cy=0 で送信完了
					; Cy=1 は送信 NG なので抜ける
;開	始条件再発行		andc	#H'FE,CCR	;OK : Cy=0
;			nop		
;戻り値 : Cy	(キャリーフラク <sup>・</sup> ) - 0 ・ 0ド		MASI2C_10:		;NG の場合 Cy=1
, Cy=	-0 . UK		115		

mov.b @ICCR1,r0I and b #H'9F,r01 rOI,@ICCR1 mov.b ACK待ち ;戻り値 : Cy (キャリーフラグ) andc #H'FE,CCR : Z (ゼロフラグ) nop Cy=0 : OK / Z=0 : ACK=0 ; rts / Z=1 : ACK=1 ; Cy=1 : NG 転送が完了していない WAITACKI2C: 最終データ読み込み指定 @ICSR,r01 ;転送完了&転送準備完了チェック mov.b #6,r01 LRDS12C: btst WAITACKI2C\_10 ; 完了していなければ @ICIER, rOI beq mov.b ; Cy=1 で抜ける bset #0,r01 mov.b @ICIER, rOI ;ICIER(#2,ACKBR)=ACK データ mov.b rOI,@ICIER #H'02,r01 ;ACK=0: Z / ACK=1:NZ and b @ICCR1,r01 mov.b andc #H'FE,CCR ;OK : Cy=0 #6,r01 bset rOI,@ICCR1 nop mov.b rts rts WAITACKI2C\_10: #H'01.CCR ;NG : Cy=1 orc nop データリード rts RDDT12C @ICSR,r01 mov.b マスタ受信モード切り替え btst #5,r01 RDDT12C ·----beq MSTRMDI2C: mov.b @ICDRR, r1I @ICSR,r01 ;送信完了チェック mov.b rts btst #6,r01 ;ICSR(#6,TEND)=1 なら送信終了 WAITACKI2C\_10 ; 完了していなければ bea ; Cy=1 で抜ける データ読み込み指定 ;送信完了フラグクリア bclr #6,r01 r01,@ICSR RDS12C: mov.b @ICIER, rOI mov.b mov.b @ICCR1,r0I ;マスタ受信モード切替 bclr #0,r01 rOI,@ICIER #4.r0l ; ICCR2(#4,TRS)=0 bclr mov.b bset #5,r01 ; ICCR2(#5,MST)=1 rts mov.b rOI,@ICCR1 ;-----@ICSR.r01 :TDRE クリア mov.b 書き込み終了チェック ;-----#7,r01 ; ICSR(#7,TDRE)=0 bclr mov.b rOI,@ICSR WRCHK12C: @IIC STA.rOI mov.b #H'FE,CCR WRCHKI2C\_00 andc ;OK : Cy=0 beq cmp.b #1, r0l nop WRCHKI2C\_10 rts beq cmp.b #2,r01 MSTRMDI2C\_10: WRCHKI2C\_20 beq #H'01,CCR orc ;NG : Cy=1 cmp.b #3,r01 beq WRCHKI2C\_30 nop #4,r01 rts cmp.b beq WRCHK12C\_40 最終データリード xor.b r01,r01 : mov.b rOl,@IIC\_STA RDFDT12C: @ICSR,r01 mov.b ;受信データ転送待ち #H'01,CCR orc btst #5,r01 ; ICSR(#5,RDRF)=1 で転送完了 rts RDFDT12C beq :---- 12C 1=>+>f12 -----@ICCR2,r01 ;停止条件発行 WRCHKI2C\_00: mov.b INITI2C:16 #H'3F,r01 ; ICCR2(#7,BBSY)=0 and b bsr mov.b r01.@ICCR2 ; ICCR2(#6,SCP) =0 mov.b #1.r0l rOI,@IIC\_STA mov.b RDFDT12C\_00: @ICSR,r01 ;----- 開始条件発行 -----;停止条件生成待ち mov.b btst #3,r01 ; ICSR(#3,STOP)=1 まで待つ WRCHKI2C\_10: STRT12C:16 beq RDFDT12C\_00 bsr WRCHKI2C\_FF bcs @ICDRR,r1I ;最終データリート mov.b #2,r01 mov.b mov.b rOI,@IIC\_STA

; ル-ブ 受信モード に設定

; ICCR1(#6,RCVD)=0

; ICCR1(#5,MST) =0

;デー9受信後 NACK を返送する

;データ受信後の受信を禁止

;受信データ転送待ち

;データ\_リード

; ICIER(#0,ACKBT)=1

; ICCR1(#6,RCVD)=1

; ICSR(#5, RDRF)=1 で転送完了

;データ受信後 ACK を返送する

; ICIER(#0,ACKBT)=0

; ステータス クリア

;NG : Cy=1

;12C 1=>v512

;次のステータスへ

;開始条件発行

;次のステータスへ

;	- デバイ <b>ス</b>	アドレス セット	
WRCH	(12C_20:	DASI 20.40	·=* II* / 7 7 7 1* 1 7 1* / =
	bsr	WRCHK12C FF	;7 /\ 1/、 / ト レノ 达信
	mov.b	#3,r01	;次のステータスへ
	mov.b	rol,@IIC_STA	
:	ACK 待?	5	
, WRCHM	<12C_30:	-	
	bsr	WAITACKI2C:16	;ACK を待つ
	DCS hne	WRCHKIZC_FF	· ACK / NACK ችተማ <b>ታ</b>
	mov.b	#0,r1l	;ACK : R1L=0
	bra	WRCHK12C_34	
WRCH	(12C_32:		
	mov.b	#1,r1l	;NACK : R1L=1
WRCHM	(12C_34:	#4 r01	· '/2 (D.1=_01 )
	mov.b	r01,@IIC_STA	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
; WRCHM	·- 終了条作 (12C 40·	十発行	
	bsr	STOP I 2C: 16	
	bcs	WRCHK12C_FF	
	xor.b	r01, r01	· አテータス クリア
	mov.b	rOI,@IIC_STA	,
	ando	#H'EE CCR	-OK - CV-O / R11-ACK
	ando	#11 1 L,00K	, or : oy-o / rre-nor
WRCHM	(I2C_FF: rts		;NG : Cy=1
;	デー	タで埋める	
;			
;R3 :R1	= パト レス = データ数		
;ROL	$= \overline{\tau}^{*} - 9$		
FILL:		-01 @-0 .71°1	7.02 사이 이 하며 하며 주면 바 7
	mov.b inc.w	rui,@r3 ;/ri #1.r3	从 R3 から R1 たけ R0L で埋める
	dec.w	#1,r1	
	bne	FILL	
	rts		
;			
;	4 µ :	sec Wait	
WAIT4	4u:		
	push.I	ER6	
	MOV.W	#20*10/6,R6	
	bra	WAII:16	
;		• .	
; :	Wa	1 t	
WAIT:			
	dec.w	#1,R6	
	pop.l	ER6	
	rts		
;	 т т	( <i>X</i> - 2) - = <i>X</i> - 7	
;	ц т т		
INITI	120:	UD 140000000	
	mov.b	#B 1000000, r01 r01,@SAR	; XV-J J'F VX=100000
	mov.b mov.b	#B'10000101,r0l r0l,@ICCR1	; ルーノ 受信t-ト 200kbps

#B'00110000,r01 ;MSB 7ァースト NoWait 9bit 7ォーマット mov.b rOI,@ICMR mov.b rts PIOイニシャライズ . . . . . . . . . . . INITPIO: xor.b rOl,rOl ;初期出力設定 mov.b rOl,@PDR1 ;ポート1 : 初期出力 Low mov.b rOl,@PDR2 ;ポート2 : 初期出力 Low mov.b #H'FF,r01 mov.b r01\_epop ;入出力設定 mov.b rOI,@PCR1 mov.b rOI,@PCR2 ;ポート1: 全て出力 ;ポート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 ;書き込みデータ RD\_DATA: .res.b D'16 ;読み出しデータ D'16 ;書き込みデータ EEPR\_DATA: . res.b 1 ; EEPROM データ .align 2 1 1 EEPR\_ADDR: .res.w ; EEPROM **ア**ト・レス ;読み出したデータのセット先 SQDT\_ADDR: .res.w ;書き込むデータ元アドレス ;R/W するデー9数 SQDT\_SIZE: .res.w 1 ROM\_ADDR: .res.w 1 .res.b ROM\_DATA: 1 WRCHK12C\_STA: .res.b WRCHKIZU\_STA: .res.b STOPI2C\_STA: .res.b 1 STOPI20\_517... MASI2C\_STA: .res.b 1 .align 2 WORK\_AREAE : .end

## 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

 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)



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。

🖉 開発環境 - Microsoft Intern	rt Explorer	_ 0
ファイル(E) 編集(E) 表示(	y) お残に入り(a) ツール(D) ヘルプ(B)	
+·+·@ 2 4 0	1 4 5 J 0 - 2 2	
アドレス(D) 🛃 http://www.ren	eses.com/jpn/products/mpumcu/tool/download/f_gtat/download.html	▼ ∂B
RENESAS Everywhere you imagine.	推動増サイト Global North America Europe Asia キーワード校本 G ドキュメント なる 単数/別の	
Japanese site	ホーム ユーザ登録 会社案内 サイトマップ お問い合せ	
デバイス別検索	<u>ホーム</u> > 製品 > <u>マイコン</u> > <mark>勤発環境トップ</mark> > <u>download</u> > f_ztat	e
ツール製品概要 🕨	F-ZTAT Microcomputer On-board Writing Program 關発環境	
各種ダウンロード 🕨		_
FAQ 🕨		
サポート情報(テクニカル情報)	F-ZTAT書込みツール無償評価版ダウンロード	
//	Midphadics20, 196-192-1939, 294-071-2022-294-0.1.	
	アップデート情報	
	マシケージパージョンを【Ver 3.0】から【Ver 3.0】コノビジルンアップしました。 通知機能につきましては、製品リリース確認をご参照くださし、⇒ Olek here(1)	
本サイトの留意点▶	個人情報の保護について » 00/2004 Renesas Technology Corp., All Rights Reserve	sd

	交開発環境システム F-ZTAT書込みツール無償料価価ダウンロード - Microsoft Internet Explorer	
E/ 6638.0		
? 1 ?	7ロコンビュータ開発環境システム	
-21	AI者込みツール無損評価版ダリンロート	
ランロー	ドを行う前に、以下ご一読頂ナますようお願いします。	
	ご注意	
1.	本プログラムを輸出すること、国外に持ち出すこと、または国内非原性者に提供する ことを禁止します、また、太ブログラムを、国際の以下相称は7次全企の推得の旅力と 7.46日的に使用。第二者に使用させ、またはそのよう2.使用日的を有するものに再 発起することが「案」します。	
2.	本プログラムの使用に開いて、弊社もしくは第三者の特許権。著作権、商標権、その 他の知動所有権等の権利に対する保証されは実防権の許諾を行われのでありま せん。また本書に記載された情報を使用した事により第三者の知動所有権等の権利 に関わる問題が生じた場合、弊社はその責き負いませんので予めこ了承びさい。	
3. 4	本プログラムは予告無(変更する場合がありますので、最終的な設計、ご使用に際し ましては、製品をご購入下さい。 整計は本サイルに記載された情報の今まわろ文書及び内容に開し、いかねる明示的	
5.	または黙示的保証を負うものではありません。 本ブログラムを流用する場合は、技術内容、ブログラム、アルゴリズム単位で評価す	
	るだけでなく、貴システム全体で十分に評価を行ない、お客様の責任において適用可 否を判断してください。弊社は、適用可否に対する責を負いませんので予めこ了承く たか、	
6.	へこい。 本プログラムの一部または全部を、弊社の文書による承認なしに複製することを堅く お新り級します。	
7.	このプログラムに関しては、評価版につきお問い合わせはできません。	
	本サービスのご利用は日本国内のみとなっております。 This service is available only in Japan.	
	上記事項に 同意する) [同意しれ、]	
	Copyright(C) 2003 Renesas Technology Corp. All Rights Reserved.	

 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 カ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。

マイクロコン	/ピュータ開発環:	境システム		Cht.649379/19
ZTAT書)	込みツール無償	評価版ダウン	ロード	
「記酒日を入力」。F=7	TAT書込みツール毎借時価版を分	ウンロードレズ下れい。		
でウンロード後のファイ) 、力項目に不備がありま	しを解凍するためには、後日送付 きすとバスワードを送付できません	設しますパスワードが必要。 のでご注意願います。	こなります。	
各入力項目には半角力	ゆカナは使用しないでください。			
御社連絡先情報を入	カしてください			
国籍	お遊びださい 🗷 必須			
会社名		必須		
部署名		必須		
住所	<b>T</b>	必須		
電話番号	半角 必須			
FAX番号	半角 必須			
使用者名		必須		
E-mailアドレス		半角 必須		
弊社特約店名	一覧よりお泳びべださい 🖃			
御社評価内容に関す	る情報を入力してください			
開発用ホフトフィ				
対象デバイフ		2/18		
内康ノフニレ		2/18		
		02010		
応用ノスノム 正規品購入予定				
正規品購入予定				
正規品購入予定			×	

#### 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 ヘプログ 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)

- スタートメニューから"Flash Development Toolkit 3.2"を起動します。
- Flash Development Toolkit 3.2
- 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。



"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
 後クリックし次へ進みます。

New Project Workspace Projects		<u>? ×</u>
FDT Project Generator	Workspace Name:         TK~3687         Project Name:         TK~3687         Directory:         C*Program Files¥Renesas¥FDT32*Workspaces¥T         CPU family:         All Flash Devices         Tool chain:         None	Browse
	ОК	キャンセル

 デバイスを選択します。"Select Device:"の欄 で"H8/3687F"を選択し、 次へ(№)> をクリッ クします。

Workspace	The FLASH Developm FLASH devices. Select the device you Select Device: HB	nent Toolkit supports a number of Renesas wish to use with this project from the st 1/3687F
Connes mod	Protocol Compiler Kernel Path Kernel Version	B Renesas 4.0A CVProgram Files¥Renesas¥FDT3.2¥Kernels¥P 1.1.00
8 24 04 40 75 54 40 20 76 5 9 4F EF 64 80 83 6F 77 1F 1		

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(い)> をクリックします。

nmunications Port		×
Workspace Industrial Co	The FLASH Development Tookit supports connection throu standard PC Serial port and the USB port. Use this page to your desired communications port. All settings may be chan the project is created.	gh the select ged after
Construction of the second sec	Select port	
S Comms.mot	Select an Interface type to connect to the target device with this will be "Direct Connection" or simply left blank.	h. Normally
St.         def         Target files           9 22 1A 22 or 30 Drive mot         Drive mot           9 22 1A 22 or 30 Drive mot         Drive mot           7 91 06 1S Drive mot         Drive mot           7 92 0A 20 S To         Drive mot           9 30 Drive mot         Drive mot           9 4 Drive Mot         Drive mot	Select Interface: Direct Connection 💌	
	< 戻る(8) 次へ(10) >	キャンセル

 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(い)> をクリックします。



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

Workspace 40 DA FF SB	The FLASH Development Toolkit can connect to your device in a number of different ways. All the options on this page may be changed after the Project has been created.
S Display	Select Connection: © BOOT Mode C USER Program Mode
Target, files Target, files LCD.mot S. Commes.met Commes.met Device Image Target files	In BOOT Program mode the device erases its FLASH prior to connection. The Toolkit downloads programming kernels to the device as required. The Recommended Speed setting is based on the current device and clock. The user may also input their own, if this is supported by the kernel (and the cotional FDM).
22 IA 20 - 51 Drive, mot 27 91 08 - 51 Data, mot 50 94 08 4 51 Algorithm, mot 50 94 08 4 55 54 40 20 75	Recommended Speeds: 57600      ✓ User Default     User Specified

ここでも変更は無いので<u><sup>完了</sup></u>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。



## ■ファイルのダウンロード

 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)。



 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。

📌 TK-3687 - Flash	Development Toolkit	
<u>F</u> ile <u>E</u> dit <u>V</u> iew	Project Tools Window	<u>D</u> evice <u>H</u> elp
<b> </b>   X h C	Set <u>O</u> urrent Project <u>I</u> nsert Project	, <b>⊥</b> ₩ % % ]}
E@ TK-3687	<u>A</u> dd Files	
🕞 ТК-36	<u>R</u> emove Files	
	File E <u>x</u> tensions	
	Re <u>b</u> uild Image	•
	Down <u>l</u> oad Image	•

 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。

Add File(s)			<u>?×</u>
ファイルの場所型:	🔁 Release	🛨 🗢 💽	·
piochk.mot			
」 ファイル名(N):	_piochk.mot		Add
ファイルの種類(工):	Project Files	L	キャンセル
	🗖 Relative Path	-	

 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)

🏸 ТК	-3687	- Flash	Developn	nent Too	lkit			
<u>F</u> ile	<u>E</u> dit	⊻iew	<u>P</u> roject	<u>T</u> ools	<u>W</u> indow	<u>D</u> evice	<u>H</u> elp	
	8	ta 🖪		<b>%</b>			<b>▲</b> 8,91	8
¢	<mark>≧ TK</mark> ⊡®	-3687 TK-36 	<b>87</b> Record Fil _piochk.m					

5. 追加したファイルをデバイスヘダウンロードしま 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。



 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。 <u>基板のジャンパ JP1 を外し</u>、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)。

	Uata programmed at the tollowing positions:						
	H'00006000 - H'0000607F Length : H'00000080						
	H'00006100 - H'0000627F Length : H'00000180						
	512 Bytes programmed in 1 seconds						
	Image successfully written to device						
Ш							

FDT] TK-3687 Find in Files

#### Ready

#### ■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。

- 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



 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。)



 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。



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 オーマット'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。

デバッグの設定		?×
Debug_Tiny_SLP_E7_SYSTEM_300H	ターゲット オプション	
samp_01	ターゲット( <u>T</u> ): Tiny/SLP E7 SYSTEM 300H デフォルトデバッグフォーマット( <u>E</u> ): Elf/Dwarf2	
	ダウンロードモジュール( <u>D</u> ): File Name Offset Address Format	<b>通知(A)</b>
		変更( <u>())</u> 上へ( <u>(</u> )) 下へ())
	ОК	キャンセル

'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'0000000'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。



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。

デバッグの設定		? ×
Debug_Tiny_SLP_E7_SYSTEM_300H s	ターゲット オプション	
- 🖓 samp.01	ターゲット①: 「Tryy/SLP E7 SYSTEM 300H デフォルトデパッグフォーマット(E): [EH/Dwart2 ダウンロードモジュール①: File Name Offset Address Format \$(CONFIGDIR)¥\$H0000000 EH/Dwart2 	

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。



'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。

Select Emulator mode			
<u>D</u> evice	H8/3687F		
Mode	<ul> <li>Download emulator firmware</li> <li>Does not download emulator firmware</li> <li>Writing Elash memory</li> </ul>		
	(ОК]		

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。



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。

Samp_01 - High-performance Embedded Workshop - [s-	amp_01.src]	_ 🗆 ×
⊘ ファイル(E) 編集(E) 表示(型) ブロジェクト(P) オブシ	risン(Q) ビルド(B) デバッグ(D) メモリ(M) ツール(D) ウィンドウ(M) ヘルプ(H)	_ # ×
🗅 🐸 🖬 🕼   🎸 X 🖻 🖻   O T 🖬	🕙 💯 🕼 🕼 🖺 😫 🙎 🖾 🚟 🛛 🎇 DISP_FLG 💽 🔍 🖧 🖓 🕇	<b>T</b> 1
🛛 🕰 🖾 🏥 🕸 🛗 🏧 🔏 🖪 Release	▼ Debug_Tiny_SLP_E7_SYS▼ 入 😧 国 同 目 計 副 ひ ひ ひ … 👘 魚 …	5 5 2 9
🔟 🗸 🖉 🄝 💭 🖬 🖬 🖽 📫		
	FILE         ::same_01.src           DATE         :Tue. Oct 07, 2003           DESURPTION         :Main Program           CPU TYPE         :#8/387           R1         レジスタボタン           DATA2         .erou	
Project Files Naviation	1	×
All Connected Disconnected Connected	レジスタウィンドウ	
Register         Register         Value           FR0         M'0000000         B'2           Br1         M'0000000         B'2           Br2         H'0000000         B'2           Br4         M'0000000         B'2           Br5         H'0000000         B'2           Br4         H'0000000         B'2           Br5         H'0000000         B'2		
ER7 H'0000010 PC H'00FFFF		×
レディ	Read-write 1/28 1 INS	CAP NUM

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。

59.19			
	Register		Register Value
シークタア 11	ERO	I	H'00000000
	ER1	H	1'00000000
	ER2	H	H'00000000
	ER3	H	1'00000000
$\wedge$	ER4	H	H'00000000
	ER5	H	1'00000000
	ER6	H	1'00000000
	ER7	H	H'00000010
リセットボタンを	PC.	I	1'000400
b11 b	1		
<u> </u>	1 = 1		
	074		



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。



"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。

## 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。

Topics-1	旧バージョンで作成したプロジェクトの開き方	65
Topics-2	vecttbl.src の置換作業を省略する	66
Topics-3	リストファイルの出力設定とファイルの追加・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	67

#### 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)

AM40	REMOTIONED CONTRACTOR NO.
結果	変換状態/エラー発生時のメッセージ結果を表示します。
変換ボタン	指定した条件で変換を開始します。
閉じるボタン	当ツールを終了します。
注意事項	
本ツールをする際、	、HEWワークスペースの位置を変更すると、正常に使用できません。 フを移動せずに変換作業者にて下すい。
1	A CHARLE & COULD FOR COULD BY
ダウンロード	
2.00H 1	
	こちら↓からダウンロードして下さい。(約20KB) ここをクリックして
	こちら↓からダウンロードしてきは、(約20KB) ここをクリックして
本サイトの倉意点・ 個人情報の保護について、	こちら 1 からダウンロードして下さい。(約20KB) Download (0)2024 Reveasa Technology Corp. All Rights Paserved
本サイトの留意点 ▶ 個人情報の保護について。	CEEをクリックして Download (0)2004 Revesas Technology Corp. All Rights Reserved

 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。

<u>C:</u>¥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。

	🚖 C:¥Hew2¥S	ystem¥Pg¥Hitac	hi¥Tiny_SLP¥T	iny_SLP_5_0¥Ger	nerate¥vecttbl			_ 🗆 ×
	ファイル(E) 新	編集(E) 表示(	1) お気に入り	)( <u>A</u> ) ツール( <u>T</u> )	ヘルプ(円)			1
	〜戻る・→・白  127ヵルダ   12 12 × い  囲・							
	300.c	2 300.src	2 36014n.c	26014n.src	36024n.c	2 36024n.src	2 36037.c	36037n.src
	36049.src	26049a.c	2 36057.c	36057n.src	<b>2</b> 3664.src	3664n.c	2672.src	<b>2</b> 3672n.c
3687.src ※アイコンは違う 場合があります	5087/src	3687n.c	2694n.c	3694n.src	<b>2</b> 38004.c	38004.src	<b>2</b> 3802.c	3802.src
	38024.C	38024.SRC	28076.C	28076.SRC	<b>28086.</b> C	38086.SRC	38104.c	38104.src
	38124.C	38124.SRC	2814.c	3814.SRC	<b>2</b> 3834.c	3834.src	2857.c	3857.src
	<b>2</b> 3867.c	3867.src	2877.c	2877.src	3887.c	23887.src		
	種類 SRC ファー	(ル サイズ: 3.03 N	(B			3.03 KB	🖳 २१ 🖳	パュータ //

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。

置換		? ×
検索する文字列( <u>N</u> ):	.1	次を検索( <u>F</u> )
置換後の文字列(P):	.w	置換して)次(こ( <u>R</u> )
		すべて置換( <u>A</u> )
		キャンセル
	(Cb/19/2)/O/	

※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。

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	. * * * * * * * * * * * * * * * * * * *	******	479		25	• * * * * * * * * *	******	*****
480		26	; メインブ	゚ログラム	480		26	; ;	メインプ	ログラム
481		27	. * * * * * * * * * * * * * * * * * * *	*****	481		27	.******* ,	******	*****
482		28			482	EA00	28	.secti	ion P,	code,locate=H'EA00
483	0000	29	_main:		483		29			
484	0000 5C0000D8	30	bsr	INITTV:16	484	EA00	30	_main:		
485	0004 5C0000DE	31	bsr	INITPIO:16	485	EA00 5C0000D8	31		bsr	INITTV:16
486		32			486	EA04 5C0000DE	32		bsr	INITPIO:16
487	0008 79030000	33	MOV.W	#WORK_AREA,r3	487		33			
488	000C 7901000B	34	MOV.W	#WORK_AREAE-WORK_AR	488	EA08 79030000	34		mov.w	#WORK_AREA,r3
489	0010 1588	35	xor.b	r01,r01	489	EA0C 7901000B	35		mov.w	#WORK_AREAE-WORK_AR
490	0012 5C0000BC	36	bsr	FILL:16	490	EA10 1588	36		xor.b	r01,r01
【 相対アドレス 】						Ι	絶対アド	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				loc	ate で絶対ア	ドレン	スを指定し	している	るのでプログラムア	
が相対値になっている				ドレ	ѵスが絶対値	こなっ	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。



## 付録-5 回路図



# 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, <del>5</del> ,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	
	P50-57,PB0-7			0	10	海結150エジュ 11
12	<del>P80-87</del> ,10-17,P20	HLMP-6001#A04	HP	ŏ ∨ 1	10	遅和LEDモンユール D00_D07(+実社) ません
	P70-76,P23,P24			X	X	P00 <sup>-</sup> P0/1よ 美表しま 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 <del>11</del> ,14	22pF		1	2	<b>※</b> 2
22	C <del>12</del> ,13	33pF		1	2	<b>※</b> 2
23	C1,2,8,9,15,21	0.1 µ F (積セラ)		9	10	C28は実装しません
24	C17.18.19.20	0.1 <i>u</i> F		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	47~100 µ F/16V (電解)		2	2	
27	,	, , , , , , <b>, , , , , , , , , , , , , </b>				
28	SW1	SKHHAK/AM/DC	ALPS	1	1	
29	JP1.4-6	2pin		4	8	<u> </u>
30	CN11	DCジャック(2.1 ゆ)		1	1	
31	CN14	D-Sub9pin		1	1	メス,ライトアングル
32	CN16,17	Box,30pin		0	2	<u>%</u> 4
	CN1,3,5,6,7,8		ют		_	× A
33	CN-B	BIOP-SHF-IAA	JS1	0	/	**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

・相当品を使用する場合かめります ※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

## 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。



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。


HEW を使うときのコ<sup>···</sup>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。

0000 番地 DFFF 番地	モニタプログラム 'ハイパーH8 '			ROM/フラッシュメモリ (56K バイト)
E000 番地 E7FF 番地		未使用		未使用
E800 番地	VECTTBL INTTBL	ベクタテーブル		
E860 番地	ResetPRG IntPRG	リセットプログラム 割り込みプログラム		
EA00 番地	Ρ	プログラム領域	ユーザ RAM エリア	RAM (2K バイト)
F000 番地 F6FF 番地				未使用
F700 番地 F77F 番地	1/0 レジスタ		1/0 レジスタ	
F780 番地 FB7F 番地	В	未初期化データ領域 ( 変数領域 )	ユーザ RAM エリア	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
FB80 番地 FD80 番地	Stack	スタック領域		RAM (1K バイト)
FE00 番地 FF7F 番地				
FF80 番地 FFFF 番地	I/0 レジスタ		I/0 レジスタ	

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。

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\_t k 3687 '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。



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。



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。

High-performance Embedded Workshop	
ファイル(E) 編集(E) 表示(Y) プロジェクト(E) ビルド(B) デバッグ(D) 基本設定(S) ツール(T) ウィンドウ(W) ヘルプ(H)	
▋□☞▋▋₿ ≵௩€ ᠻ║╄ ╅╅⋩║◈ᢁᢁ┻	
● 最近使用したプロジェクトワークスペースを開く(@):     アドミニストレーション(A)       ○:¥hew4_tk:3687¥led¥ledhws	
○ 別のプロジェクトワークスペースを参照する(B)	
Build (Debug ) Find in Files Version Control	
レディ	

## 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 まず,「ワークスペース名(<u>W</u>)」(ここでは 'led ')を入力します。「プロジェクト名(<u>P</u>)」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。 右の「参照(<u>B</u>)…」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 (<u>D</u>)」が正しいか確認して下さい。()

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 「Assembly Application」を選択 します。

入力が終わったら 「OK」をクリックして下さい。

新規プロジェクトワークスペース		<u>? ×</u>
プロジェクト  Application  Assembly Application  Empty Application  Find Import Makefile  Library	ワークスペース名(W): led フロジェクト名(P): led ディレクトリ(D): ③ C:¥hew4_tk3687¥led CPU種別(Q): H8S,H8/300 マールチェイン(T): Hitachi H8S,H8/300 Standard	<b>②</b> 参照(B)
	<u>(5)</u>	
	ОК	キャンセル

「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 (<u>N</u>) >」をクリックします。

新規フ <b>゚</b> ロジェクトー1/8-CPU	<u>? ×</u>
	ツールチェインハ <sup>×</sup> ージョン: 6.1.0.0 このプロジ・ェクトで使うCPUのジリーズをタイフ を選択して下さい。 CPUジリーズ: 2600 2000 3001 300 3001 CPUタイフ <sup>*</sup> : 3685 3686 5687 36902 36912 36912 36912 36912 36912 36912 36912 36912 3694 「 選択したいCPUタイプがない場合は、ハードウェア 送知したいCPUタイプがない場合は、ハードウェア
	ださい。
< 戻る( <u>B</u> )	次へ(N)>完了キャンセル

「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。

新規プロジ	ェクト-2/8-オフ*ション	<u>?</u> ×	
		グューバルオフ♡ョンを指定します。	
	新規プロジェクトー3/8ー生成ファイル		?×
		, 自動生成するイニシャルルーチンを選択します。	
		■ 1/0517 59使用 1/0ストリーム数: 3 量	
		■ ヒーフ*メモリ使用 ヒーフ*サイス*:	
191		_main 関数生成 Assembly source file ▼	
		None N-トウェアセットアッフ 間数生成	
	<戻	る(B) 次へ(N)> 完了 キャン・	セル

「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, スタックポインタを H'FE00 に, スタックサイズを H'80 にします。設定が終 わったら「次へ(<u>N</u>) >」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。)

新規プロジェクトー4/8ースタック領域	<u>?</u>	×
H'Se	スタックの設定を行って下さい。 0 に変更 スタックホ°インタアト <sup>・</sup> しス: (power-on reset) H'FE00 スタックサイス: H'80	
< 戻る(	) 次へ(N) > 完了 キャンセル	

「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。

新規プロシ	±クト-5/8-ヘ°ንን	? ×	
	新規プロジェクト-6/8-デバゥガ	ターケ <sup>*</sup> ット : H8/300HA Simulator H8/300HN Simulator 9-ケ <sup>*</sup> ットタイフ <sup>*</sup> : 300H	? ×
	< 戻る	5(B) 次へ(N) ティン ディン	セル

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。

新規プロジェクト-8/8-生成ファイル名			<u>?×</u>
	以下のソース	ファイルを生	も成します。:
	ファイル名	拡	角罕記兑
	intprg vecttbl vect resetprg led stacksct	src src inc src src src	Interrupt Program Initialize of Vector Table Definition of Vector Reset Program Main Program Setting of Stack area
	•		
く戻る(B)	次へ( <u>N</u> ) >	]	完了 キャンセル

すると、「概要」が表示されるので「OK」をクリックします。

<b>観要</b> プロジェクトの概要:	?>
PROJECT NAME : PROJECT DIRECTORY : CPU SERIES : CPU TYPE : TOOLCHAIN NAME : TOOLCHAIN VERSION : GENERATION FILES : C¥hew4_tk3687¥led¥led¥in Interrupt Program C¥hew4_tk3687¥led¥led¥ve Initialize of Vector Tab C¥hew4_tk3687¥led¥led¥ve Definition of Vector C¥hew4_tk3687¥led¥led¥re Reset Program C¥hew4_tk3687¥led¥led¥re Main Program	led C:¥hew4_tk3687¥led¥led 300H 3687 Hitachi H8S,H8/300 Standard Toolch 6.1.0.0 htprg.src le ect.inc esetprg.src ed.src ▼
OKをワリックしプロジェクトを作るか、C	Cancelを知っりしアホートするかを選択します。 小りにReadme.txtという名前で保存する
	Cancel

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。



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(<u>F</u>)」ボタンをクリックしてください。

H85,H8/300 Standard Toolchain	<u>?×</u>
コンフィグレーション:	コンパイラ アセンブラ 最適化リンカ 標準ライブラリ CPU デバッナ・
Debug	カテゴリ(Y): <mark>ピクション</mark> 設定項目(S): セクション Address Section Address Section 「 Address Section 「 Address Section 「 (本の0000000 VECTTBL 「 NTTBL 0×000000400 ResetPRG 「 ntPRG 0×00000800 P 0×00000FD80 Stack
	最適化リンカオブション: -noprelink -nomessage -list="\$(CONFIGDIR)¥\$(PROJECTNAME).map" -nooptimize -start=VECTTBL,INTTBL/00,ResetPRG,IntPRG/0400,P/0800,Sta OK キャンセル

「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。 '0x00000800' というところをクリックして下さい。それ から、「変更(<u>M</u>)…」をクリックしま す。

Section VECTTBL INTTBL ResetPRG IntPRG P Stack		l	Of キャン 追加・	K セル (A)
VECTTBL INTTBL ResetPRG IntPRG P Stack		L	キャン 追加・	セル (A)
INTTBL ResetPRG IntPRG P Stack		L	キャン 追加・	セル (A)
ResetPRG IntPRG P Stack		ι	追加	(A)
IntPRG P Stack		L	追加	(A)
P Stack	-		追加	(#)
Stack				<u></u>
			- 亦面(	(M)
				<u></u> y
			複数割	付◎
			削除	:( <u>R</u> )
			<b>†</b>	÷
			上回	下(D)
			インポー	-ÞΦ
			エクスポ	
				<ul> <li>              ≹数割 単)除</li></ul>

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。

セクションのアドレス	<u>?×</u>
アドレス( <u>A</u> ): (16 <u>)進</u> 数)	0×EA00 🕂
ОК	キャンセル

すると...



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(<u>A</u>)...」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。

セクションのアドレス	<u>?</u> ×
アドレス( <u>A</u> ):   (16進数)	0×F780 🔆
ОК	キャンセル

Address に 0xF780 が追加されました。

セクション設	定		? ×
Address	Section		ок
0×E800	VECTTBL		
	INTTBL		キャンセル
0×E860	ResetPRG	_	
L	IntPRG		C 38 1078 S
0×EA00	Р		
UXF /80	<u>.</u>	_	変更(M)
0x0000 080	Stack		
			複数割付( <u>O</u> )
ļ	旦加!!		<u>削除(R)</u>
			<b>† </b>
			F(0) (0)
			インポートの
			エクスポート( <u>E</u> )
,			

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。

Add section	?×
セクション名(S):	
В	•
ОК	キャンセル

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(<u>E</u>)」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(<u>I</u>)」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

セクション設定	定		?)
Address	Section		ок
0×E800	VECTTBL	]	
	INTTBL	1	キャンセル
0×E860	ResetPRG	]	
	IntPRG	1	2000 TO 100 TO 100 TO 100
0×EA00	P	]	<u>适加(A)</u>
0×F780	B		亦重八心
0x0000FD80	Stack		
			複数割付(Q) 削除(R)
			インボートΦ エクスポート( <u>E</u> )

H85,H8/300 Standard Toolchain	<u>? ×</u>
コンフィグレーション: Debug     All Loaded Projects	コンパイラ アセンブラ 最適化リンカ 標準ライブラリ CPU デバッナ▲▲         カテゴリ(Y):       セクション         設定項目(S):       セクション         Address       Section         0x0000E800       VECTTBL         INTTBL       0x0000E860         0x0000E860       ResetPRG         IntPRG       0x0000F780         0x0000FD80       Stack
	ん 最適化リンカオプション: -noprelink -nomessage -list="\$(CONFIGDIR)¥\$(PROJECTNAME).map" -nooptimize -start=VECTTBL,INTTBL/0E800,ResetPRG,IntPRG/0E860,P/0E OK キャンセル

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。

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 P,CODE,LOCATE=H'ea00 .section \_main: #H'01,r0I ;ポート6のイニシャライズ mov.b rOl,@H'ffe9 mov.b loop: bclr #O,0H'ffd9 ;LEDオン(P60=0) ;LEDオラ(P60=1) ;LOOPIこジャンプ #0,0H'ffd9 hset bra loop . end

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 メニューバーから「ビルド」を選ぶか, 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

×	Phone Outlinhou staution
	Phase OptLinker Starting
	License expires in 60 days
	L1100 (W) Cannot find "B" specified in option "start"
	Phase OptLinker finished
	Build Finished
	0 Errors, 1 Warning
	• • • • • • • • • • • • • • • • • • •

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。 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 (<u>T</u>)…」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。 ファイルの
 種類を、すべてのファイ
 ル、にして下さい。 ・led.
 mot、をダブルクリック
 します。

テキスト ファイルの送信					? ×
ファイルの場所型:	🔄 Debug		- 🗈 🖻	* 📰 🖽	
🛋 Debug.hdp	/led.mot	🛋 vecttbl.h8a			
🔊 intprg.h8a	🔍 lea.obj		11		
🔘 intpre.obj	🛋 resetprg.h8a	シタノルク	199		
📓 led.abs	🔘 resetprg.obj				
📓 led.h8a	🔊 stackset.h8a				
🔊 led.hlk	🔘 stackset.obj				
ファイル名(凹):	led.mot			開<(0)	
ファイルの種類(工): (	すべてのファイル (*:	*)	<b>_</b>	キャンセル	

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。

4	▶38400bps - ハイハ°ーターミナル
7	ァイル(E) 編集(E) 表示(V) 通信(C) 転送(T) ヘルフ℃(H)
IF	
	Hyper Monitor Program. for H8/3687F -ver,040809- Copyright(C)2003-2004 by TOYO-LINX,Co.,LTD.
	< [?] = Command Help >
	H8>L Waiting for HEX File ****** File Name [led.mot] Load Address [D0E800-00EA0D] Finish!
	H8>_

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。

※38400bps - ハイハ <sup>®</sup> -ターミナル ファイル(E) 編集(E) 表テヘハ 通		×
Hyper Monitor Program for H8/3687F -ver,040 Copyright(C)2003-2004 < [?] = Command Help	m. 0809- 14 by TOYO-LINX,Co.,LTD.	
H8>L Waiting for H ******* File Name Load Address Finish!	HEX File [led.mot] [OOE800-OOEAOD]	
H8>DEADO EAOO F8 01 38 EA10 88 00 00 EA20 00 00 00 EA30 00 00 00 EA40 00 00 00 EA50 00 00 00 EA60 00 00 00 EA70 00 00 00	2       13       14       15       16       17       18       18       18       10       19       14       18       10       14       15       16       17       10	
H8>_		-
接続 00:05:25 ANSI	38400 8-N-1 SCROLL CAPS NUM キャプチャ エコーを印刷します。	11

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