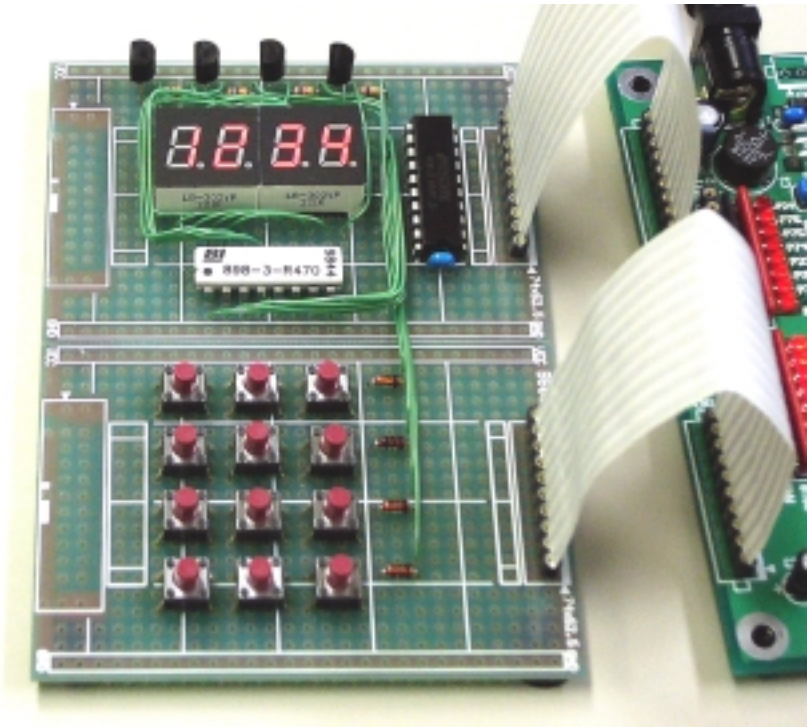
－TK-3687Option－

ダイナミックスキャン

Version 1.20



目　次

Toyo-linx,Co.,Ltd.

1 はじめに、、、

このキットはダイナミック方式でのセグメント表示とマトリックスキーの入力をハード・ソフト共に学習していきます。

ダイナミック表示とは、多桁のセグメント表示を行う際1つのドライバで複数個のセグメントを時分割により駆動する方式です。人間の目の反応より十分速く表示を切り替えれば各セグメントが同時に点灯しているように見えます。ダイナミック表示を行うことによって 1 つの桁が点灯している間、他の桁は消灯しているので消費電力を小さくすることができ、また信号線や IC の数を減らすことができます。

マトリックスキーもダイナミック表示同様少ない信号線でより多くのキーを読み込む方式です。例えば 64 個のキーを設けたい場合、8bit の入力と 8bit のスキャン出力、計 16 本の信号線で 8bit×8 本の 64 キーを設けることができます。キーを読むにはスキャンを切り替えて 8bit 入力を繰り返します。こちらも人間が気づかない速さでスキャンをする事で全てのキーを読み込んでいるかのよう動作します。

学習の流れは、、、

ハードの組み立て
↓
ダイナミック表示のプログラミング
↓
マトリックスキーのプログラミング
↓
応用プログラム

の順で説明していきます。学習を進めていくに当たり次のものを用意して下さい。

・組み立て

ハンダごて、ハンダ、ニッパ、ワイヤストリッパ、ピンセット

・プログラム

High-performance Embedded Workshop(HEW)、もしくは Motorola 形式 HEX ファイル(*.mot)を生成するアセンブラ(本文内で掲載しているリストは HEW を使用して作成しています)、D-Sub9pin ストレートケーブル

2 ハードの組み立て

各プログラムを作成する前に付属のユニバーサル基板にハードを組み上げます。プリント基板と違いユニバーサル基板は全ての配線を自分で結線しなければなりませんが、回路図を見ながら部品をハンダ付けしていく事によってハードの内容をより理解しやすくなります。

まず、工具を揃えましょう。組み立てるのに必要な工具は次の通りです。

ハンダごて、ハンダ、ニッパ、ワイヤストリッパ、ピンセット

次に部品の確認を行ないます。梱包されている 7 セグメント LED の種類によって部品表、回路図、実装図が異なります。部品の入っている袋に貼ってあるシールに“アノードコモン”と記されていた場合は、4～6ページを、“カソードコモン”と記されていた場合は、7～9ページを参照して下さい。それぞれの部品表と照らし合わせて全ての部品がそろっているか確認をします。確認のできた部品は部品表にチェックを入れておくと良いでしょう。

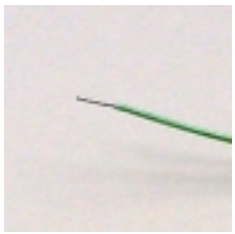
部品の確認が済みましたら、いよいよ組み立てです。それぞれの回路図・実装図をよく見ながらハンダ付けを行って下さい。ハンダ付けによるケーブル配線は大変なので、電源や GND、交差しない信号線等はハンダ面でメッキ線や部品のリード線を流用して接続します（完成写真を参照）。ハンダ面のみでは配線しきれない信号線はラッピングケーブルで配線していきます。尚、ラッピングケーブルでの配線が初めての方は以下に示した結線方法を参照して下さい。

TK-3687 とオプション基板とは 10 芯接続ケーブルで接続します。10 芯接続ケーブルはオプション基板とは直付けに、TK-3687 とは丸ピンソケットを介して接続しますので、間違えてオプション基板に丸ピンソケットを実装しないよう注意して下さい。

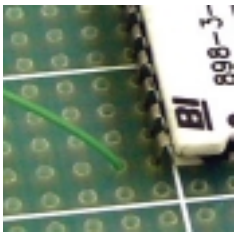
TK-3687 の JP5,JP6 は、右図のようにラッピングケーブルでジャンパします。

■ラッピングケーブルでの結線方法

ハンダ面で結線されない配線はラッピングケーブルで結線していきます。ここではラッピングケーブルでの結線の仕方を説明します。

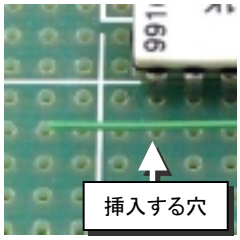


1.
被覆を剥きます
(5mm 位)



2.
部品面からに穴に通します

3.
ハンダ付けする部品の足にラッピングケーブルを絡げてからハンダ付けします



4.
結線先までケーブルを這わせ、挿入する穴から 3 つ先の穴辺りでカットします
1～3 の手順でハンダ付けします

■動作チェック

全てのハンダ付けが終了したら、最後にもう一度結線に間違えが無いかよく確認して下さい。また TK-3687 上のジャンパ JP5,JP6 にジャンパが差してあるか確認して下さい。間違えが無いようなら動作確認を行います。パソコンとTK-3687を接続して内蔵の簡易モニタを起動します。モニタでファイルのロードと実行のコマンド“LG”を入力し、CD-ROM に収録されているチェックプログラム“checkprg.mot”をロード・実行します。ファイルの場所は、CD-ROM:¥TK-3687¥ｵﾌﾟｼｮﾝ¥ﾀﾞｲﾅﾐｯｸｽｷｬﾝ¥ﾌﾟﾛｸﾞﾗﾑ¥checkprg.mot です。

最初は表示のチェックです。0～F までの表示を下桁から順に行います。各桁がちゃんと 0～F の表示になっているか確認して下さい。例えばある桁だけ表示しない場合は各セグメントのコモン端子(pin5or10)が導通していない可能性があります。また、あるセグメントだけ点灯しない、もしくは他のセグメントと同時に点灯している場合はそのセグメントに対応するピンの接続を確認して下さい。

次にキーのチェックを行います。0～F の表示が各桁終了すると“PUSH”と表示されてから左から 2 番目の桁に“1”と表示されます。左 2 桁に押すキー番号が表示されますので、その指示通りにキーを押していってください。表示は指示したキーが押されると次の番号を示します。もし、対応するキーを押しているのに次に進まない場合はそのキーの接続を再度確認して下さい。

最後に“ End”と表示されればチェック完了です。

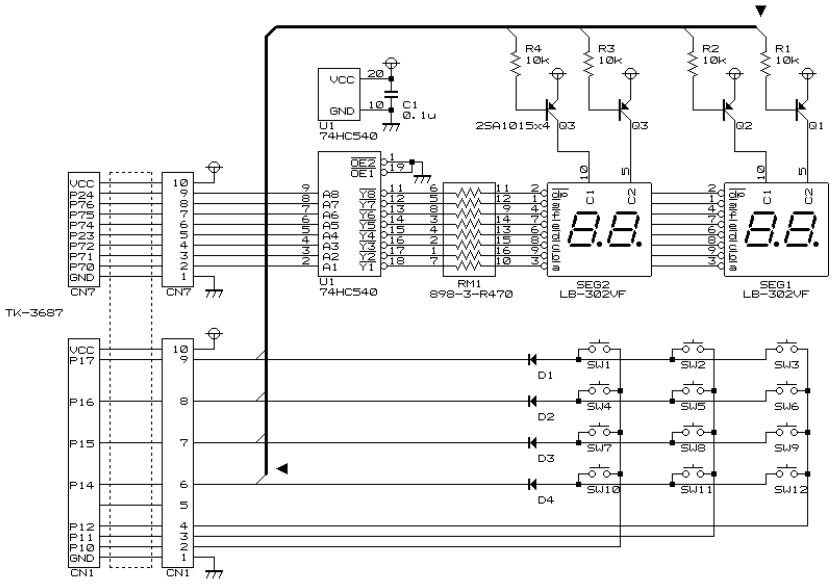
■部品（アノードコモンの場合）

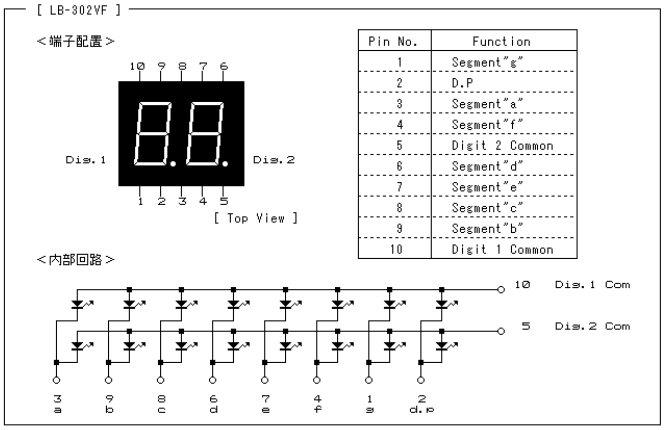
TK-3687オプション　ダイナミックスキャン　部品表（※アノードコモン）	全数：	1

	部品番号	型名，規格	メーカー	数量	全数	備考
1	U1	74HC540		1	1	
2	Q1,2,3,4	2SA1015		4	4	※1
3	SEG1,2	LB-302VF	ROHM	2	2	※1
4	D1,2,3,4	1S1588相当		4	4	※1
5	R1,2,3,4	10kΩ		4	4	
6	RM1	898-3-R470	BI	1	1	
7	C1	0.1μF（積セラ）		1	1	
8	SW1-12	SKHHAK/AM/DC	ALPS	12	12	
9	CN1,7	10pin丸ピンソケット		2	2	20pinを2つに切断　※2
10	基板	B6082	東洋リンクス	1	1	2枚を1枚として使用
11	ゴム足			4	4	
12	ラッピングケーブル	2m		1	1	結線用
13	メッキ線	Vcc,GND等半田面結線用　※3				
14						
15	10芯接続ケーブル	A10-N100-A	Hitaltech	2	2	CN1,7接続用
16						

※1　相当品を使用する場合があります。
※2　TK-3687に実装します。
※3　ラッピングケーブルの被覆を剥し2本をよじって使用します。また抵抗の足も流用できます。

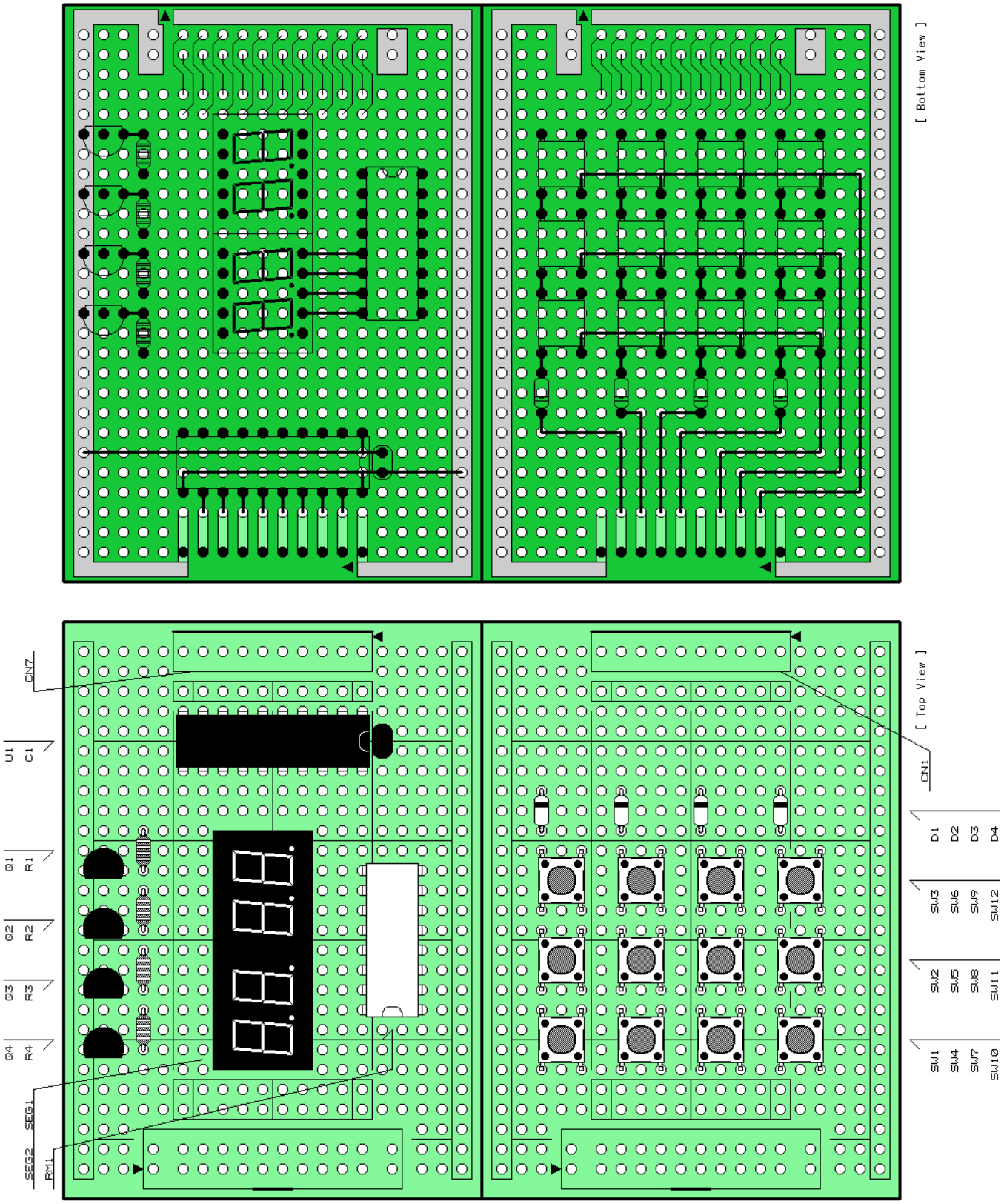
■回路図（アノードコモンの場合）



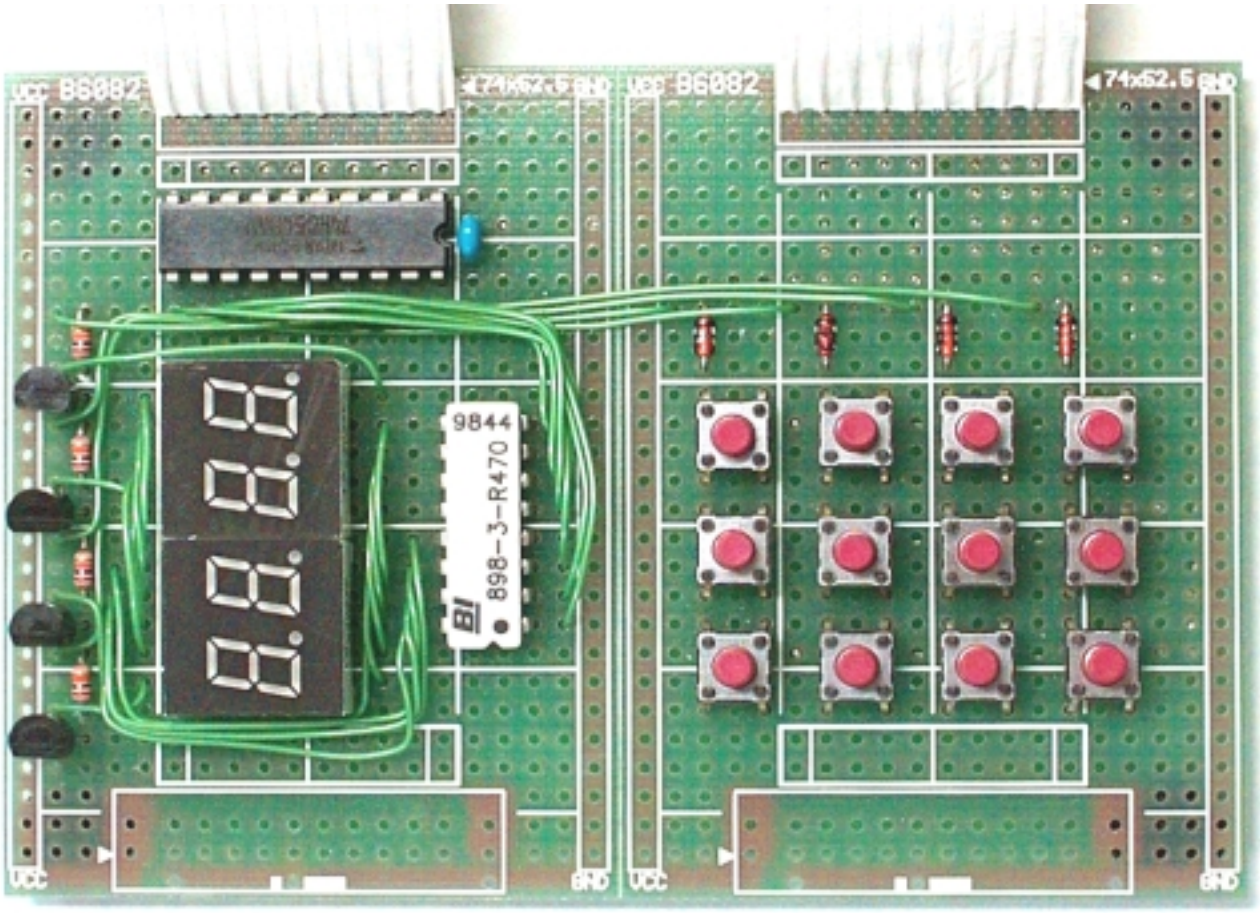


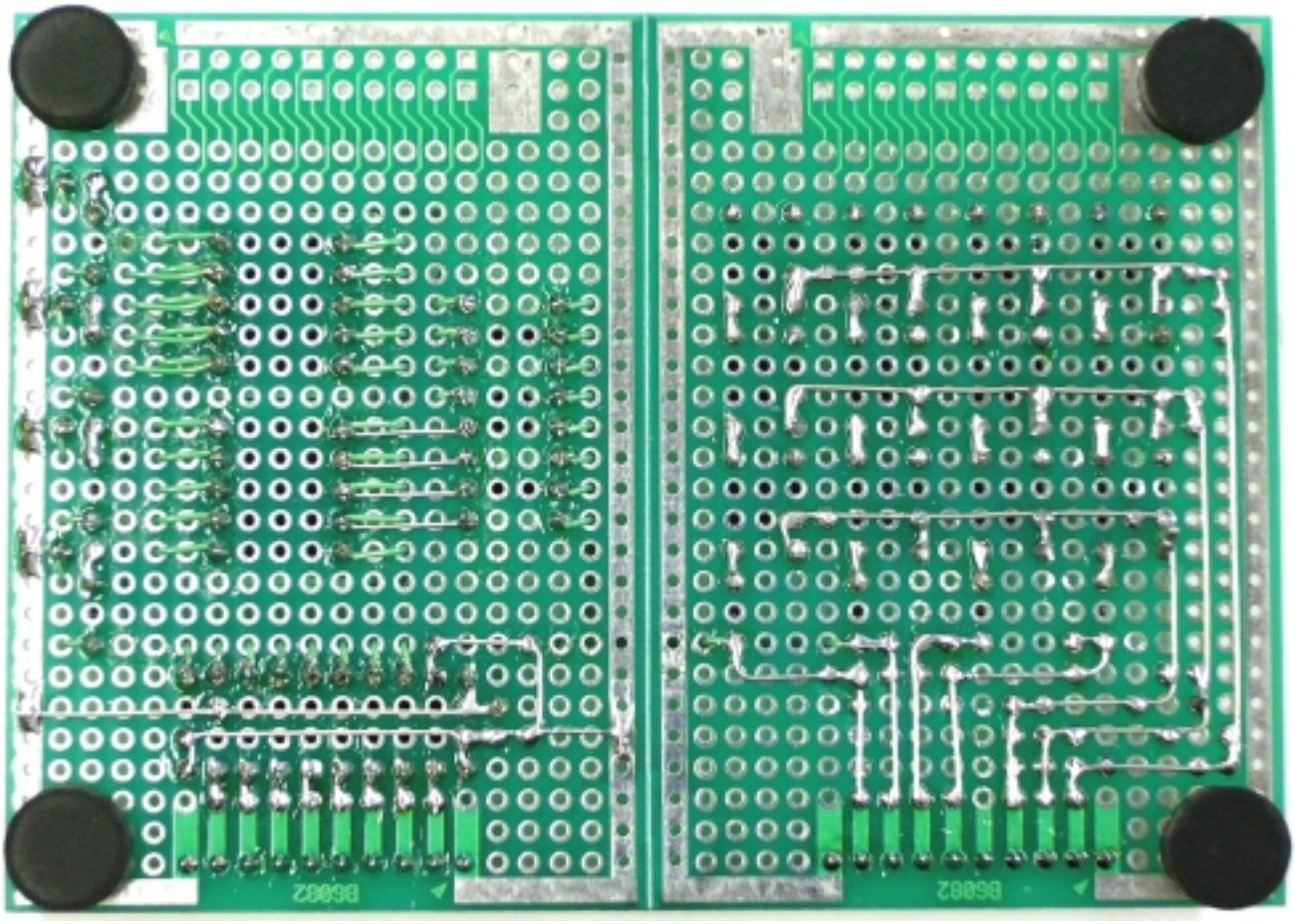
Pin No.	Function
1	Segment"g"
2	D.P
3	Segment"a"
4	Segment"f"
5	Digit 2 Common
6	Segment"d"
7	Segment"e"
8	Segment"c"
9	Segment"b"
10	Digit 1 Common

■実装図（アノードコモンの場合）



■完成写真（アノードコモンの場合）





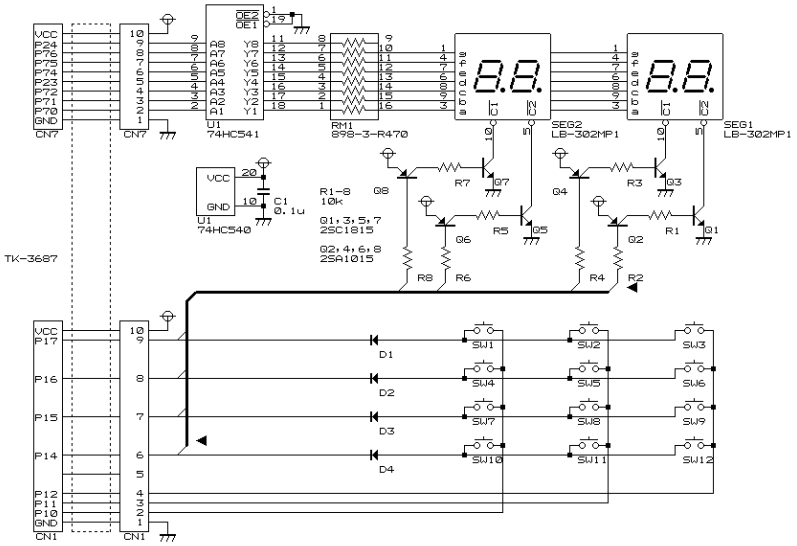
■部品（カソードコモンの場合）

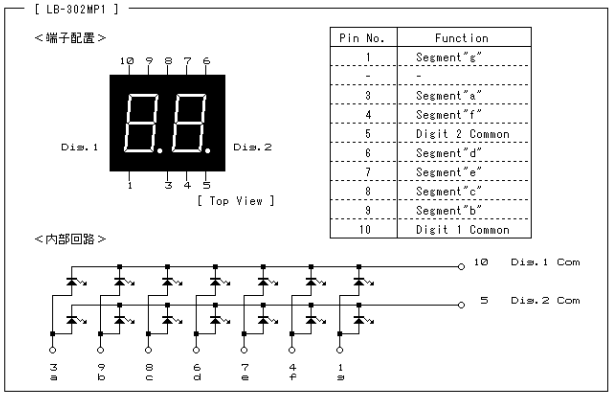
TK-3687オプション　ダイナミックスキャン　部品表(※カソードコモン)	全数：	1

	部品番号	型名，規格	メーカー	数量	全数	備考
1	U1	74HC541		1	1	
2	Q1,3,5,7	2SC1815		4	4	※1
3	Q2,4,6,8	2SA1015		4	4	※1
4	SEG1,2	LB-302MP1	ROHM	2	2	※1
5	D1,2,3,4	1S1588相当		4	4	※1
6	R1～8	10kΩ		8	8	
7	RM1	898-3-R470	BI	1	1	
8	C1	0.1μF（積セラ）		1	1	
9	SW1-12	SKHHAK/AM/DC	ALPS	12	12	
10	CN1,7	10pin丸ピンソケット		2	2	20pinを2つに切断　※2
11	基板	B6082	東洋リンクス	1	1	2枚を1枚として使用
12	ゴム足			4		
13	ラッピングケーブル	2m		1	1	結線用
14	メッキ線	Vcc,GND等半田面結線用　※3				
15						
16	10芯接続ケーブル	A10-N100-A	Hitaltech	2	2	CN1,7接続用
17						

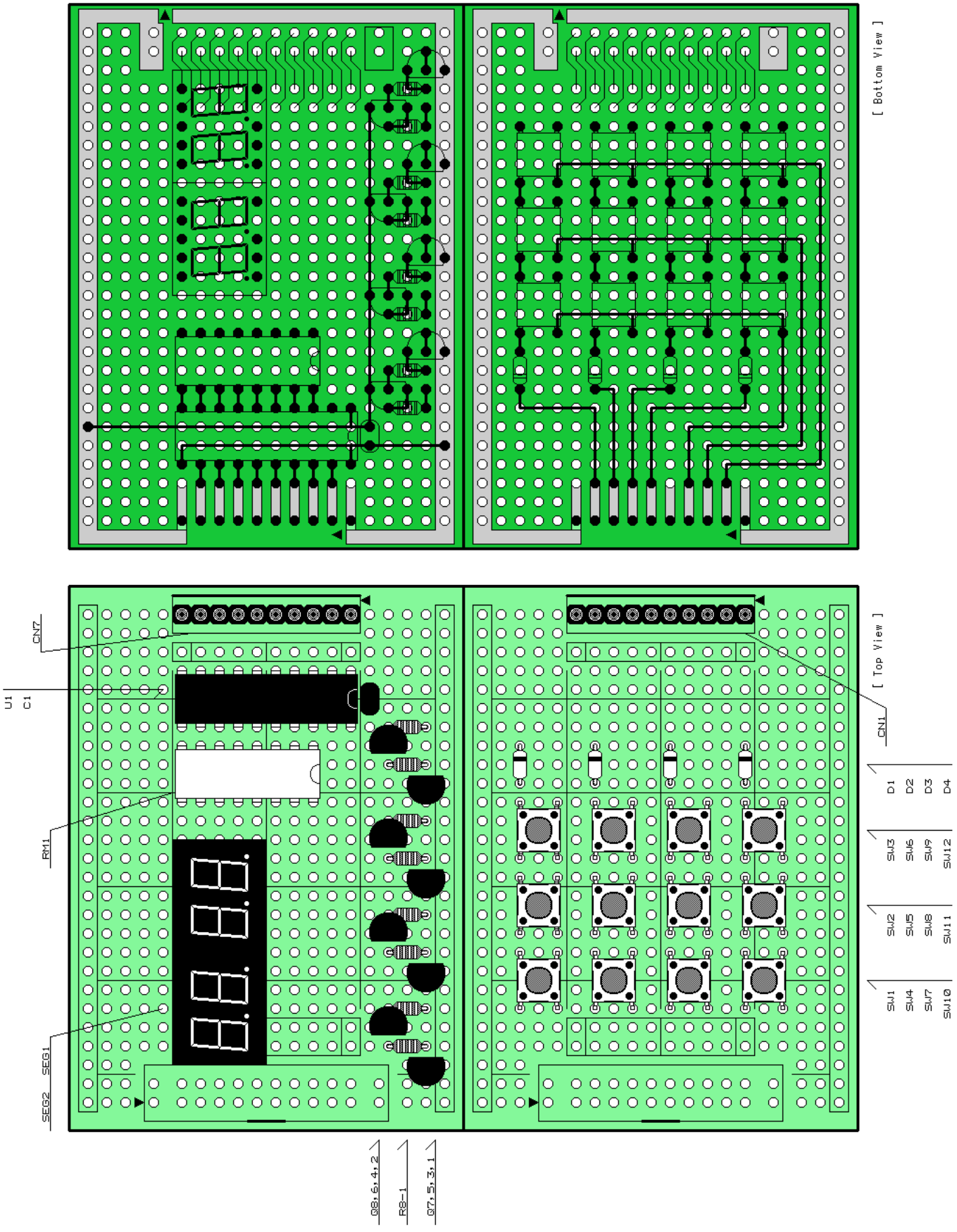
※1　相当品を使用する場合があります。
※2　2つはTK-3687に実装します。
※3　ラッピングケーブルの被覆を剥し2本をよじって使用します。また抵抗の足も流用できます。

■回路図（カソードコモンの場合）

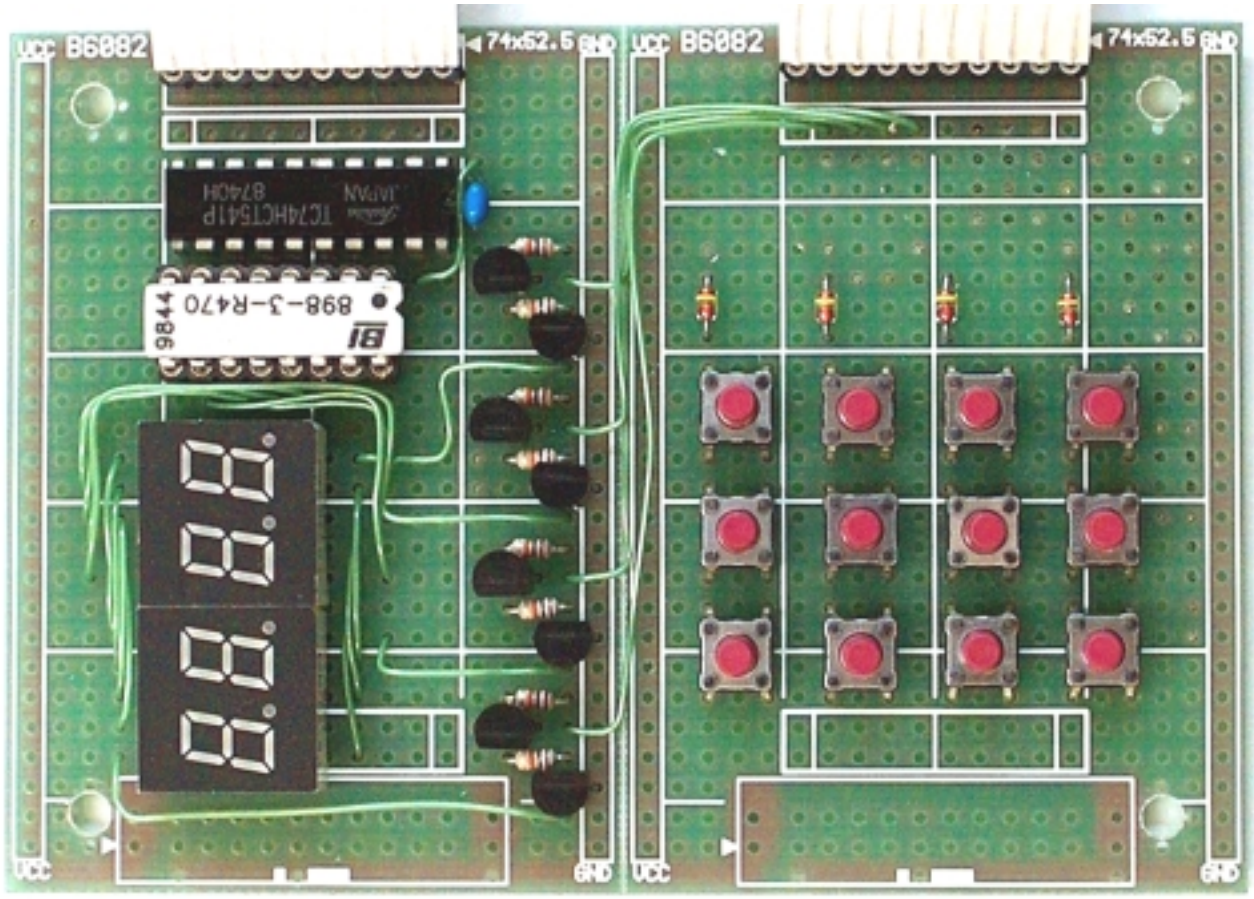


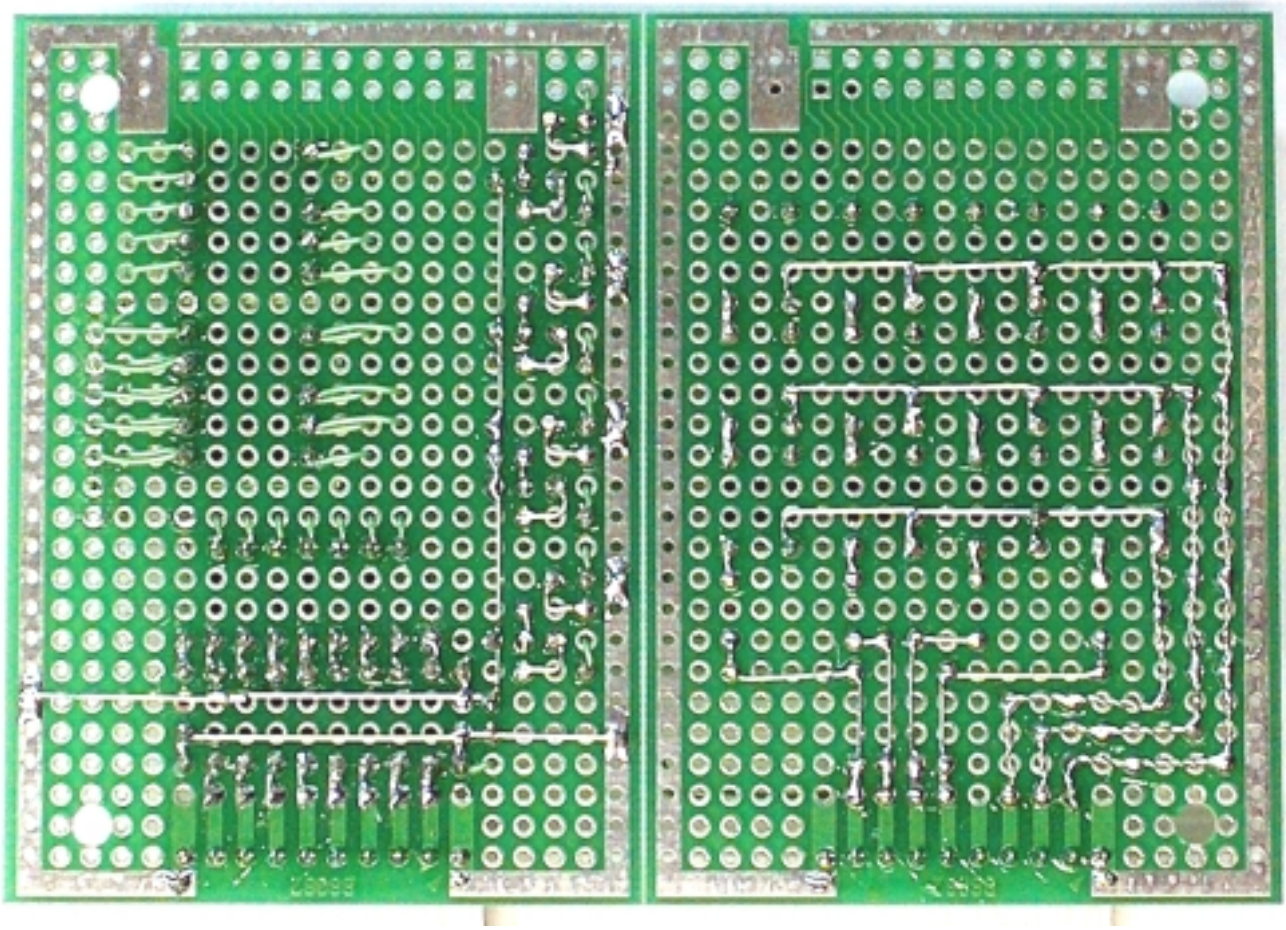


■実装図（カソードコモンの場合）



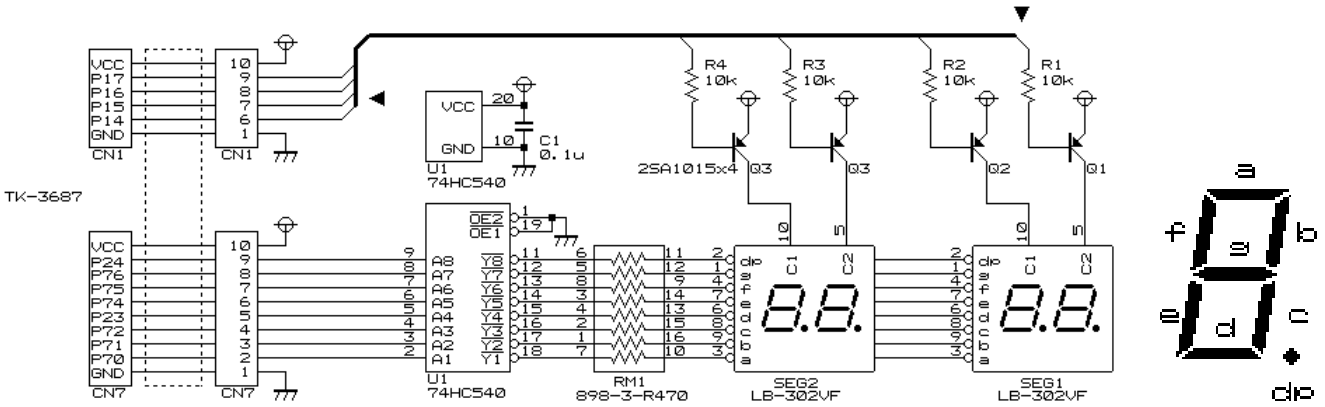
■完成写真（カソードコモンの場合）





3 ダイナミック表示のプログラム

図 3-1 は表示部分を抜き出した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。使用している P7 ポートには P73,P77 がありませんのでその分を P23,P24 で補っています。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。出力するデータとしては LED のドライバとして実装されている 74HC540 が論理を反転しますので正論理となりますので B’00000110,P23=0,P24=0 となります。



＜図 3-1 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 P7 及び P23,P24 は点灯させたいビットを“Hi” 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＞

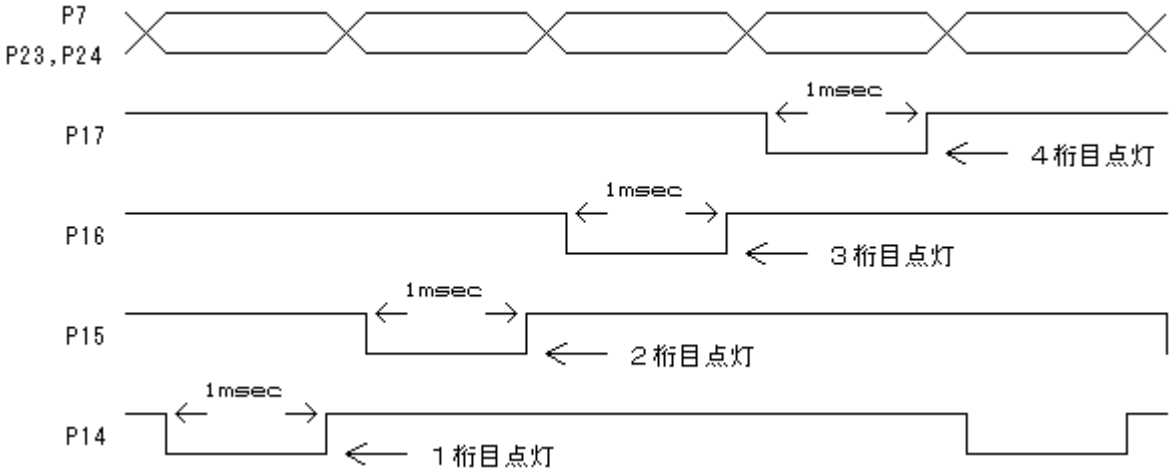
```
;---------------------------------------------------------
;   FILE        :Dscan_1.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;             and programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;---------------------------------------------------------

          .include   "io3687F_equ.inc"
          .export    _main

;=========================================================
;          メインプログラム
;=========================================================
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr        INIPIO          ;PIO ｲﾆｼｬﾗｲｽﾞ
;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------
MAIN_LOOP:
     mov.b      #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b      r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b      #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     mov.b      r0l,@PDR7       ;出力
     bclr       #3,@PDR2
     bclr       #4,@PDR2
     bra        MAIN_LOOP
```

```
;=========================================================
;          サブルーチンプログラム
;=========================================================
;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INIPIO:
     mov.b      #H'02,r0l
     mov.b      r0l,@PMR1
     mov.b      #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b      #H'07,r0l
     mov.b      r0l,@PUCR1
     mov.b      #H'1D,r0l       ;ﾎﾟｰﾄ 2 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR2       ;P23,P24:out
     mov.b      #H'FF,r0l       ;ﾎﾟｰﾄ 7 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR7       ;ﾎﾟｰﾄ 70-77:out
     rts

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

＜リスト 3-1 Dscan_1.src＞

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、それを 4 桁繰り返し行います。タイミングチャートで表すと図 3-2 のようになります。



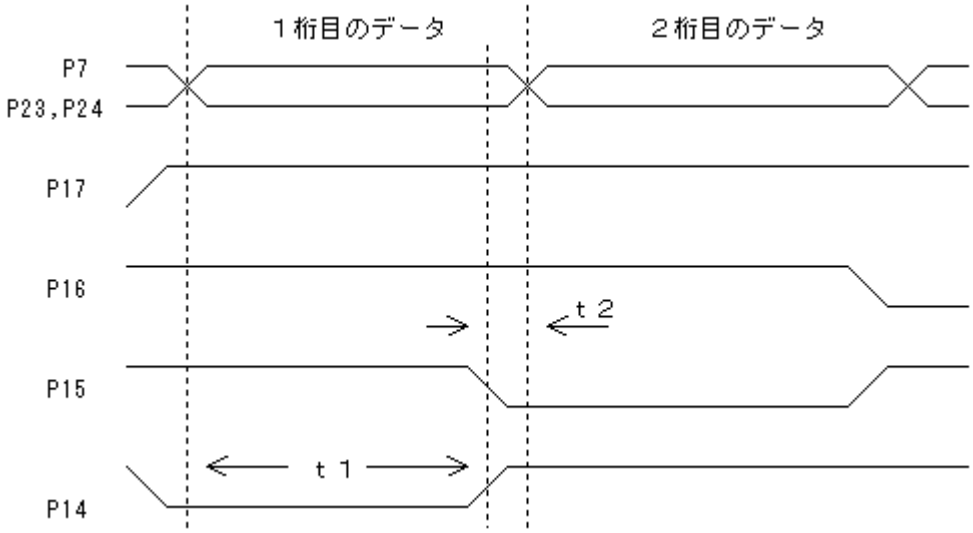
＜図 3-2 ダイナミック表示タイミングチャート＞

図 3-2 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＞

```
;---------------------------------------------------------
;   FILE        :Dscan_2.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;---------------------------------------------------------

          .include   "io3687F_equ.inc"
          .export    _main

;=========================================================
;          メインプログラム
;=========================================================
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------
MAIN_LOOP:
     mov.b     #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     mov.b     #H'DD,r1l       ;ﾎﾟｰﾄ 15:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01001111,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:3
     mov.b     r0l,@PDR7       ;出力
     bset      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     mov.b     #H'BB,r1l       ;ﾎﾟｰﾄ 16:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01011011,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:2
     mov.b     r0l,@PDR7       ;出力
     bset      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     mov.b     #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'00000110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:1
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     bra       MAIN_LOOP

;=========================================================
;          サブルーチンプログラム
;=========================================================
;---------------------------------------------------------
;          1msec タイマ
;---------------------------------------------------------
TIMER1m:                       ;8
     mov.l     #H'D02,er6      ;6
TIMER1m_00:
     dec.l     #1,er6          ;2
     bne       TIMER1m_00      ;4
     rts                       ;8

;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INIPIO:
     mov.b     #H'02,r0l
     mov.b     r0l,@PMR1
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'07,r0l
     mov.b     r0l,@PUCR1
     mov.b     #H'1D,r0l       ;ﾎﾟｰﾄ 2 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR2       ;P23,P24:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 7 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR7       ;ﾎﾟｰﾄ 70-77:out
     rts

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

＜リスト 3-2 Dscan_2.src＞

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表示データは切り替わっていない為です。

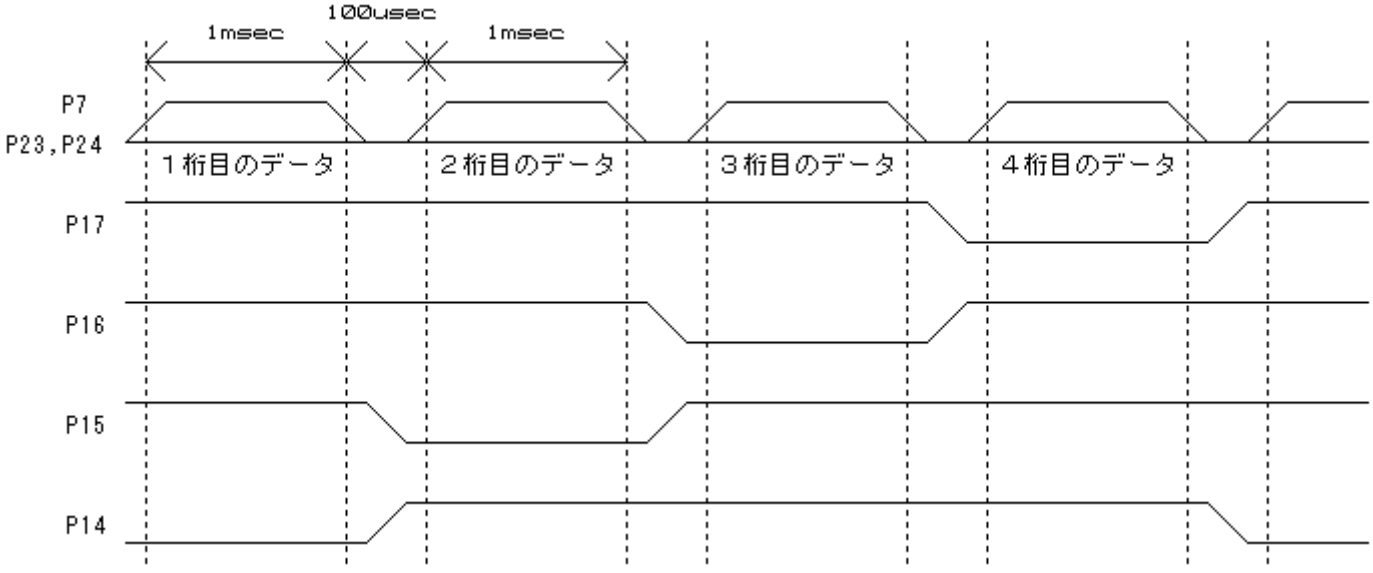


＜図 3-3　Dscan_2 のタイミングチャート＞

その様子を示したのが図 3-3 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 を変更する前に表示データを“L”にして LED を一旦消灯する。
2. 1 の状態を安定させる為にタイマーを入れる。タイマーの時間は点灯している時間に対して十分に短くし、見かけ上 LED が消灯していることが分からない位の 100μsecとする。

図 3-4 はそのタイミングチャートです。



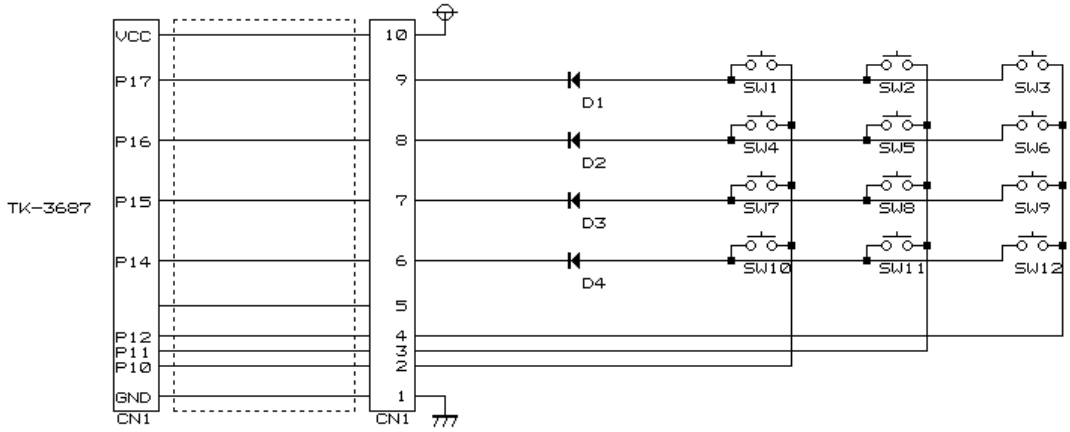
＜図 3-4　Dscan_3 のタイミングチャート＞

```
;---------------------------------------------------------
;
;   FILE        :Dscan_3.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;             and generated by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------

          .include  "io3687F_equ.inc"

          .export   _main

;=========================================================
;          メインプログラム
;=========================================================
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ

;-----  ﾒｲﾝﾙｰﾌﾟ  -------------------
MAIN_LOOP:
     mov.b     #H'EE,r1l       ;ﾎﾟｰﾄ 14:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01100110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:4
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER100u:16

     mov.b     #H'DD,r1l       ;ﾎﾟｰﾄ 15:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01001111,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:3
     mov.b     r0l,@PDR7       ;出力
     bset      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER100u:16

     mov.b     #H'BB,r1l       ;ﾎﾟｰﾄ 16:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'01011011,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:2
     mov.b     r0l,@PDR7       ;出力
     bset      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER100u:16

     mov.b     #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     #B'00000110,r0l ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:1
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER1m:16
     xor.b     r0l,r0l         ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     bsr       TIMER100u:16

     bra       MAIN_LOOP

;=========================================================
;          サブルーチンプログラム
;=========================================================
;---------------------------------------------------------
;          1msec タイマ
;---------------------------------------------------------
TIMER1m:                       ;8
     mov.l     #H'D02,er6      ;6
TIMER1m_00:
     dec.l     #1,er6          ;2
     bne       TIMER1m_00      ;4
     rts                       ;8

;---------------------------------------------------------
;          100μsec タイマ
;---------------------------------------------------------
TIMER100u:                     ;8
     mov.l     #H'14A,er6      ;6
TIMER100u_00:
     dec.l     #1,er6          ;2
     bne       TIMER100u_00    ;4
     rts                       ;8

;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INIPIO:
     mov.b     #H'02,r0l
     mov.b     r0l,@PMR1
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'07,r0l
     mov.b     r0l,@PUCR1
     mov.b     #H'1D,r0l       ;ﾎﾟｰﾄ 2 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR2       ;P23,P24:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ 7 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR7       ;ﾎﾟｰﾄ 70-77:out
     rts

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

<リスト 3-3 Dscan_3(Dynamic.src)>

4 マトリックスキーのプログラム

図 4-1 はマトリックスキーのみ抜き出した回路図です。P14～17 がスキャンライン、P10～12 が読み込む為の入力ラインです。キーを読むには読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。
スキャンラインに接続されているダイオードは電流の逆流を防止します。例えば P14 が Low で SW7,SW10 が共に押された場合、ダイオードが無いと P14・P15 がショートしてしまいます。このようなキーを同時に押された際に生じる電流の逆流をダイオードが防いでくれます。



＜図 4-1 マトリックスキー回路図＞

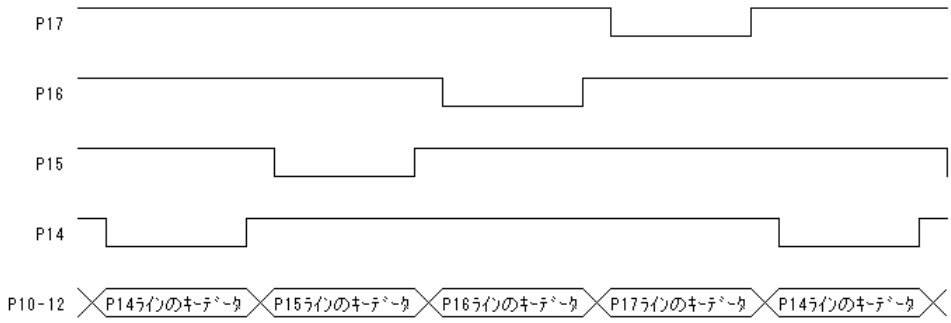
まずはスキャンを行わずにキーを読んでみましょう。スキャンライン P17 のみを Low にして SW1～3 のいずれかが押されたらセグメントに表示します。スキャンラインは表示と共通なのでセグメントの表示は最上桁になります。

```
;---------------------------------------------------------
;
;   FILE        :matrixkey_1.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;             and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------
          .include   "io3687F_equ.inc"

          .export    _main

;=========================================================
;          メインプログラム
;=========================================================
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr        INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------
MAIN_LOOP:
     mov.b      #H'77,r1l       ;ﾎﾟｰﾄ 17:Low
     mov.b      r1l,@PDR1       ;ｽｷｬﾝ出力

     mov.b      @PDR1,r0l       ;ｷｰ入力
     not        r0l             ;負論理なので反転
     and.b      #H'07,r0l       ;ｷｰの有効 bit は 3bit
     bne        MAINLOOP_00     ;押されていたらｾｸﾞﾒﾝﾄに表示
     xor.b      r0l,r0l         ;ｾｸﾞﾒﾝﾄ消灯
     mov.b      r0l,@PDR7       ;表示
     bclr       #3,@PDR2
     bclr       #4,@PDR2
     bra        MAIN_LOOP

MAINLOOP_00:
     mov.b      #B'01011100,r0l ;ｾｸﾞﾒﾝﾄ表示
     mov.b      r0l,@PDR7       ;表示
     bset       #3,@PDR2
     bclr       #4,@PDR2
     bra        MAIN_LOOP

;=========================================================
;          サブルーチンプログラム
;=========================================================
;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INIPIO:
     mov.b      #H'02,r0l
     mov.b      r0l,@PMR1
     mov.b      #H'F0,r0l       ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR1       ;ﾎﾟｰﾄ 14-17:out
     mov.b      #H'07,r0l
     mov.b      r0l,@PUCR1
     mov.b      #H'1D,r0l       ;ﾎﾟｰﾄ 2 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR2       ;P23,P24:out
     mov.b      #H'FF,r0l       ;ﾎﾟｰﾄ 7 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR7       ;ﾎﾟｰﾄ 70-77:out
     rts

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

＜リスト 4-1 matrixkey_1.src＞

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 4-2 に示します。



＜図 4-2　スキャンでのキー読み込みタイミングチャート＞

上記のタイミングを元に各キーを読み込むプログラムを作ってみましょう。それぞれのキー番号をセグメントの最下桁に表示させます。マトリックスキーの読み込みを判り易くするためにダイナミック表示は行わずに表示します。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。

```
;-------------------------------------------------------
;
;   FILE        :matrixkey_2.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;-------------------------------------------------------

     .include  "io3687F_equ.inc"

     .export   _main

;=======================================================
;          メインプログラム
;=======================================================
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
     mov.b     #H'00,r3l       ;R3L=ｾｸﾞﾒﾝﾄﾃﾞｰﾀ

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------
MAIN_LOOP:
     mov.b     #H'FF,r0l       ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ:消灯
     mov.b     r0l,@PDR7       ;出力
     bclr      #3,@PDR2
     bclr      #4,@PDR2

     mov.b     #H'77,r1l       ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     mov.b     #D'4,r1h        ;R1H=ﾙｰﾌﾟ回数
     mov.w     #D'1,r2         ;R2=ｷｰ番号
MAINLOOP_10:
     mov.b     r1l,r0l
     and.b     #H'F0,r0l
     mov.b     r0l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     @PDR1,r0l       ;ｷｰ読み込み
     not       r0l             ;反転
     and.b     #H'07,r0l       ;ｷｰは押されていた?
     bne       MAINLOOP_12     ;ne = 押されていた
     add.w     #D'3,r2         ;ｷｰ番号 +3
     rotr.b    r1l             ;次のｽｷｬﾝへ
     dec.b     r1h             ;ﾙｰﾌﾟ回数 -1
     bne       MAINLOOP_10
     bra       MAINLOOP_20     ;ﾙｰﾌﾟ終了
MAINLOOP_12:                   ;ｷｰ検出
     btst      #0,r0l          ;bit0 ﾁｪｯｸ
     bne       MAINLOOP_16
     btst      #1,r0l          ;bit1 ﾁｪｯｸ
     beq       MAINLOOP_14
     inc.w     #1,r2           ;ｷｰ番号 +1
     bra       MAINLOOP_16
MAINLOOP_14:
     inc.w     #2,r2           ;ｷｰ番号 +2
MAINLOOP_16:
     xor.w     e2,e2
     mov.b     @(SEG_TBL,er2),r3l   ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀ ｹﾞｯﾄ

MAINLOOP_20:                   ;表示ﾌﾟﾛｸﾞﾗﾑ
     mov.b     #H'EE,r0l
     mov.b     r0l,@PDR1       ;表示用ｽｷｬﾝ出力
     mov.b     r3l,r0l
     mov.b     r0l,@PDR7       ;表示ﾃﾞｰﾀ 出力
     btst      #3,r0l
     bne       MAINLOOP_22
     bclr      #3,@PDR2
     bra       MAINLOOP_24
MAINLOOP_22:
     bset      #3,@PDR2

MAINLOOP_24:
     btst      #7,r0l
```

＜リスト 4-2　matrixkey_2.src (1/2)＞

```
     bne       MAINLOOP_26
     bclr      #4,@PDR2
     bra       MAINLOOP_28
MAINLOOP_26:
     bset      #4,@PDR2

MAINLOOP_28:
     bsr       TIMER1m          ;表示時間 = 1msec
     mov.b     #H'FF,r0l        ;ｽｷｬﾝ off
     mov.b     r0l,@PDR1

     bra       MAIN_LOOP

;/////  PIO ｲﾆｼｬﾗｲｽﾞ  ////////////////////////////////////////
INIPIO:
     mov.b     #H'02,r0l
     mov.b     r0l,@PMR1
     mov.b     #H'F0,r0l        ;ﾎﾟｰﾄ 1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1        ;ﾎﾟｰﾄ 14-17:out
     mov.b     #H'07,r0l
     mov.b     r0l,@PUCR1
     mov.b     #H'1D,r0l        ;ﾎﾟｰﾄ 2 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR2        ;P23,P24:out
     mov.b     #H'FF,r0l        ;ﾎﾟｰﾄ 7 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR7        ;ﾎﾟｰﾄ 70-77:out
     rts
```

```
;/////  TIMER1msec  ////////////////////////////////////
TIMER1m:                      ;8
     mov.l     #H'D02,er6     ;6
TIMER1m_00:
     dec.l     #1,er6         ;2
     bne       TIMER1m_00     ;4
     rts                      ;8

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

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

＜リスト 4-2　matrixkey_2.src (2/2)＞

5 応用プログラム

応用プログラムしてダイナミック表示とマトリックスキーを組み合わせた時計プログラムを作成しましょう。時刻のカウントには内蔵の RTC を使用し、表示は 4 桁ですので時分をセグメントで表示します。秒は最下位桁の dp を点滅させて表現します。時計ですので時刻設定が出来なくてはなりません。そこでマトリックスキーを使用して時刻設定ができるようにします。

いきなり時計のプログラムに入るには少し敷居が高いのでまずはダイナミック表示とマトリックスキーを組み合わせてみましょう。キーが押されると今までの表示を左へシフトし押されたキーの番号を下桁に表示します。ここで必要なプログラムは大きく分けると“キー入力”、“表示データのシフト”、“ダイナミック表示”の3つになります。それぞれの働きを分りやすくする為にサブルーチン化しましょう。キー入力を“KEYIN”、表示データシフトを“DSHIFT”、ダイナミック表示を“DISP”とします。今まで作成したプログラムはレジスタのみで行っていましたが今回はそれぞれサブルーチン化されるので必要なデータはメモリ経由でやり取りを行います。

それぞれのサブルーチンを考えてみましょう。

“KEYIN”

基本的には＜matrixkey_2＞と同じ処理ですが、ここではワンショット動作を追加します。これはキーを押した際始めの一度のみキーを受け付ける処理です。押しつづけても表示が流れてしまわないようにします。処理内容としてはキーの押し続けを検出し押され続けていたらその入力は無視します。キーの入力があったらキー番号をバッファ“KEY_NO”にセットしフラグ“KEY_FG”を 1 にセットします。

“DSHIFT”

各桁毎に表示バッファを持たせておき、バッファのデータを入れ替えシフトしていきます。“KEY_FG”が 1 だったら表示バッファをシフトして“KEY_NO”を下桁にセットします。シフト作業を行ったら“KEY_FG”は必ずクリアします。

“DISP”

今までは1ループで全桁を表示していましたがそれでは表示処理に時間がかかりすぎてしまいます。ここでは1ループで1桁表示とします。“DISP”をコールするとスキャンビットを移動してその桁のデータを表示します。輝度をある程度保つ為にウェイトをいれて点灯時間を確保しておきます。

上記を踏まえて作成したのが＜KEYIN_DISP＞です。リストを掲載します。

```
;------------------------------------------------------
;
;   FILE        :KEYIN_DISP.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;------------------------------------------------------

          .include   "io3687F_equ.inc"
          .export         _main

;======================================================
;         メインプログラム
;======================================================
```

```
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------
     bsr        INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
     xor.l      er0,er0         ;ﾃﾞｰﾀのｸﾘｱ
     mov.w      r0,@KEY_FG
     mov.l      er0,@DISP_BUF
     mov.b      #H'77,r0l       ;ｽｷｬﾝﾃﾞｰﾀ ｾｯﾄ
     mov.b      r0l,@SCAN_DT
     mov.l      #DISP_BUF,er0   ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ ｾｯﾄ
     mov.l      er0,@DISPDT_ADR

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------
MAIN_LOOP:
     bsr        KEYIN:16        ;ｷｰ入力
     bsr        DSHIFT:16       ;表示ﾃﾞｰﾀｼﾌﾄ
     bsr        DISP:16         ;表示
     bra        MAIN_LOOP
```

＜リスト 5-1　KEYIN_DISP.src　(1/3)＞

```
;=========================================================
;          サブルーチンプログラム
;=========================================================
;---------------------------------------------------------
;          KEYIN
;---------------------------------------------------------
KEYIN:
     mov.b     #H'77,r1l       ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     mov.b     #D'4,r1h        ;R1H=ﾙｰﾌﾟ回数
     mov.w     #D'1,r2         ;R2=ｷｰ番号
KEYIN_00:
     mov.b     r1l,r0l
     and.b     #H'F0,r0l
     mov.b     r0l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     @PDR1,r0l       ;ｷｰ読み込み
     not       r0l             ;反転
     and.b     #H'07,r0l       ;ｷｰは押されていた?
     bne       KEYIN_02        ;ne = 押されていた
     add.w     #D'3,r2         ;ｷｰ番号 +3
     rotr. b   r1l             ;次のｽｷｬﾝへ
     dec.b     r1h             ;ﾙｰﾌﾟ回数 -1
     bne       KEYIN_00
     bra       KEYIN_10        ;ﾙｰﾌﾟ終了
KEYIN_02:                      ;ｷｰ検出
     btst      #0,r0l          ;bit0 ﾁｪｯｸ
     bne       KEYIN_06
     btst      #1,r0l          ;bit1 ﾁｪｯｸ
     beq       KEYIN_04
     inc.w     #1,r2           ;ｷｰ番号 +1
     bra       KEYIN_06
KEYIN_04:
     inc.w     #2,r2           ;ｷｰ番号 +2
KEYIN_06:
     mov.w     @KEY_NO,r0      ;ｷｰの押し続けﾁｪｯｸ
     cmp.w     r0,r2
     beq       KEYIN_12        ;押し続け ﾌﾗｸﾞをｸﾘｱ
     mov.w     r2,@KEY_NO      ;入力有り ｷｰ番号ｾｯﾄ
     mov.w     #H'1,r0
     mov.w     r0,@KEY_FG      ;ｷｰﾌﾗｸﾞ ｾｯﾄ
     rts
KEYIN_10:
     xor.w     r0,r0           ;ｷｰ番号 ｸﾘｱ
     mov.w     r0,@KEY_NO
KEYIN_12:
     xor.w     r0,r0           ;ｷｰﾌﾗｸﾞ ｸﾘｱ
     mov.w     r0,@KEY_FG
     rts

;---------------------------------------------------------
;          DSHIFT
;---------------------------------------------------------
DSHIFT:
     mov.w     @KEY_FG,r0      ;ｷｰﾌﾗｸﾞ ﾁｪｯｸ
     bne       DSHIFT_00
     rts
DSHIFT_00:
     xor.w     r0,r0           ;ｷｰﾌﾗｸﾞ ｸﾘｱ
     mov.w     r0,@KEY_FG
     mov.l     @DISP_BUF,er0   ;表示ﾃﾞｰﾀ 読み込み
     mov.w     r0,r1           ;左ｼﾌﾄ
     mov.w     e0,r0
     mov.b     r0l,r0h
     mov.b     r1h,r0l
     mov.w     r0,e0
     mov.b     r1l,r0h
     xor.l     er1,er1
     mov.b     @KEY_NO+1,r1l   ;ｾｸﾞﾒﾝﾄﾃﾞｰﾀに変換
     mov.b     @(SEG_TBL,er1),r0l
     mov.l     er0,@DISP_BUF   ;表示ﾃﾞｰﾀの更新
     rts

;---------------------------------------------------------
;          DISP
;---------------------------------------------------------
DISP:
     mov.l     @DISPDT_ADR,er3 ;ER3=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b     @SCAN_DT,r1l    ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     inc.l     #1,er3          ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ+1
     rotr.b    r1l             ;ｽｷｬﾝ移動
     bcs       DISOP_00        ;一巡したか？
     mov.l     #DISP_BUF,er3   ;一巡ならｱﾄﾞﾚｽ ﾘｾｯﾄ
DISOP_00:
     mov.l     er3,@DISPDT_ADR
     mov.b     r1l,@SCAN_DT
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     @er3,r0l        ;表示ﾃﾞｰﾀ読み込み
     mov.b     r0l,@PDR7       ;表示ﾃﾞｰﾀ出力
     btst      #3,r0l
     bne       DISOP_02
     bclr      #3,@PDR2
     bra       DISOP_04
DISOP_02:
     bset      #3,@PDR2
DISOP_04:
     btst      #7,r0l
     bne       DISOP_06
     bclr      #4,@PDR2
     bra       DISOP_08
DISOP_06:
     bset      #4,@PDR2
DISOP_08:
     bsr       TIMER1m         ;表示時間=1msec
     xor.b     r0l,r0l
     mov.b     r0l,@PDR7
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     mov.b     #H'FF,r0l       ;消灯
     mov.b     r0l,@PDR1
     rts

;---------------------------------------------------------
;          1msec タイマ
;---------------------------------------------------------
TIMER1m:
     mov.l     #H'D02,er6
TIMER1m_00:
     dec.l     #1,er6
     bne       TIMER1m_00
     rts

;---------------------------------------------------------
;          ＰＩＯイニシャライズ
;---------------------------------------------------------
INIPIO:
     mov.b     #H'02,r0l
     mov.b     r0l,@PMR1
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ14-17:out
     mov.b     #H'07,r0l
     mov.b     r0l,@PUCR1
     mov.b     #H'1D,r0l       ;ﾎﾟｰﾄ2 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR2       ;P23,P24:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ7 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR7       ;ﾎﾟｰﾄ70-77:out
     rts
```

<リスト 5-1 KEYIN_DISP.src (2/3)>

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

```
;---------------------------------------------------------
;           データ・エリア
;---------------------------------------------------------
            .section   D,data,locate=H'F780
KEY_FG:           .res.w     1          ;ｷｰ入力ﾌﾗｸﾞ
KEY_NO:           .res.w     1          ;ｷｰ番号
DISP_BUF:         .res.l     1          ;表示ﾃﾞｰﾀﾊﾞｯﾌｧ
DISPDT_ADR:       .res.l     1          ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ
SCAN_DT:          .res.b     1          ;ｽｷｬﾝﾃﾞｰﾀﾊﾞｯﾌｧ
            .align     2

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

<リスト 5-1　KEYIN_DISP.src　(3/3)>

ダイナミック表示とマトリックスキーの組み合わせが出来たなら、次にRTCを組み合わせてみましょう。RTCを読み取りその値をダイナミック表示します。メインループにRTC読み取りサブルーチン“RTC_read”と読み取ったデータを表示データに変換するサブルーチン“DispDT_set”を追加します。“KEYIN_DISP”で使用していた表示データシフトサブルーチン“DSHIFT”は削除します。

プログラムを追加する前に使用するRTCの各レジスタを簡単に説明しておきます。尚、より詳しい説明については H8/3687 ハードウェアマニュアルを参照してください。

・RTC コントロールレジスタ 1（RTCCR1）

時計タイマの動作停止／動作開始及びリセットを制御します。

7	6	5	4	3	2	1	0
RUN	12/24	PM	RST	-	-	-	-
1	1	0	0	0	0	0	0

bit7：RUN　0=RTC 動作停止 ／ 1=RTC 動作開始
bit6：12/24　0=12 時間動作 ／ 1=24 時間動作
bit5：PM　12 時間動作時に 0=AM ／ 1=PM
bit4：RST　0=通常動作 ／ 1=リセット ※1 を書き込んだら必ず 0 を書き込むこと

・クロックソースセレクトレジスタ（RTCCSR）

クロックソースの選択を行います。

7	6	5	4	3	2	1	0
-	RCS6	RCS5	-	RCS3	RCS2	RCS1	RCS0
0	0	0	0	1	0	0	0

bit6：RCS6 ┐ クロック出力選択
bit5：RCS5 ┘ ※作成するプログラムではクロックを出力する必要は無いでデフォルトの B’00 を書き込みます。

bit3：RCS3 ┐
bit2：RCS2 │ クロックソース選択
bit1：RCS1 │ ※作成するプログラムではサブクロックを選択するので B’1000 を書き込みます。
bit0：RCS0 ┘

・**秒／分／時データレジスタ（RSECDR／RMINDR／RHRDR）**

秒／分／時のカウントを行います。

7	6	5	4	3	2	1	0
BSY	SC12	SC11	SC10	SC03	SC02	SC01	SC00

bit7：BSY　1=RTC ビジー（データ更新中） ※0 の時にデータを読み込みます。

bit7：SC12
bit6：SC11　10 位データ　※データは BCD コード。
bit5：SC10

bit3：SC03
bit2：SC02　1 位データ　※データは BCD コード。
bit1：SC01
bit0：SC00

■RTC 読み取りサブルーチン“RTC_read”

RTC から時間を読み取るには秒／分／時いずれかのデータレジスの bit7 をチェックし、bit7 が 0 のときに読み取ります。しかし、ただ bit7 が 0 だからといって読み込んでいては処理に無駄があります。また、読み込んでいる間にデータが更新される可能性もあるので、時間データを読み込むタイミングは bit7 が 1→0 に変換した瞬間が一番望ましいです。そこで bit7=1 の時にセットされるフラグを一つ設けます。Bit7=0 の時、フラグが 1 であれば bit7 が 1→0 に変化したタイミングですので時間を読み出します。この時フラグは bit7=0 なのでクリアしておきます。そうすることによって次に bit7=0 を検出してもフラグは 0 なので 1→0 のタイミングにはなりませんので無駄な読み込みを防げます。それらを踏まえた RTC 読み取りサブルーチン＜RTC_read＞を以下に示します。このサブルーチンは bit7 のフラグに RTCRD_FG、読み取った時間データはバッファ Time に格納しています。

```
;-------------------------------------------------------
;          時間の読み取り(RTC Read)
;-------------------------------------------------------
RTC_read:
     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     btst      #7,@er3         ;RTC Busy?
     bne       RTCread_02      ;bit7:1 = Busy
     mov.b     @RTCRD_FG,r0l   ;今まで Busy だった?
     beq       RTCread_04      ;RTCRD_FG:0=NonBusy
     xor.w     r0,r0           ;RTCRD_FG:0=NonBusy Set
     mov.b     r0l,@RTCRD_FG

     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     mov.l     #Time+5,er2     ;ER2=Time+5=Time_SL
     mov.b     #3,r1l          ;R2L=loop count
RTCread_00:
     mov.b     @er3,r0l
     and.b     #H'0F,r0l
     mov.b     r0l,@er2
     subs.l    #1,er2
     mov.b     @er3,r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     mov.b     r0l,@er2
     subs.l    #1,er2
     adds.l    #1,er3
     dec.b     r1l
     bne       RTCread_00
     mov.b     @dp_FG,r0l
     xor.b     #H'FF,r0l
     mov.b     r0l,@dp_FG
     RTS
RTCread_02:
     mov.b     #H'01,r0l       ;RTCRD_FG=1 set
     mov.b     r0l,@RTCRD_FG   ;1→0 を検出する為
RTCread_04:
     RTS
```

＜リスト 5-2　サブルーチン“RTC_read”＞

■表示データ変換サブルーチン“DispDT_set”

RTC から読み取ったデータは BCD コードなので表示するにはセグメント変換しなければなりません。そこで表示データに変換するサブルーチン“DispDT_set”を表示の前に追加します。表示データのセットは後で追加する別のモードでも使用できるので汎用性のある作りにしておきます。ここでは汎用レジスタ ER3 を変換元データアドレスとし、ER3 をセットしてからサブルーチンをコールします。“DispDT_set”のリストを示します。

```
;---------------------------------------------------------
;          表示データのセット
;---------------------------------------------------------
DispDT_set:
     mov.l     #Disp_DT,er2    ;ER2=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b     #4,r4l          ;R4L=ﾙｰﾌﾟ回数
DispDTset_00:
     xor.l     er0,er0
     mov.b     @er3,r0l
     mov.b     @(SEG_TBL,er0),r0l   ;ｾｸﾞﾒﾝﾄ変換
     mov.b     r0l,@er2        ;表示ﾃﾞｰﾀｾｯﾄ
     adds.l    #1,er2          ;ｱﾄﾞﾚｽ+1
     adds.l    #1,er3
     dec.b     r4l             ;ﾙｰﾌﾟ回数-1
     bne       DispDTset_00

     mov.b     @Disp_1,r0l
     bset      #7,r0l
     mov.b     @dp_FG,r0h
     beq       DispDTset_02
     bclr      #7,r0l
DispDTset_02:
     mov.b     r0l,@Disp_1

     rts
```

＜リスト 5-3　サブルーチン“DispDT_set”＞

“RTC_read”、“DispDT_set”を追加したプログラムリスト＜watch_1＞を以下に示します。文中で説明はしていませんが、“dp_FG”を使用して最下位桁の dp で秒の点滅を行っています。それほど難しくはないと思いますのでリスト中から読み取ってみてください。

```
;---------------------------------------------------------
;
;   FILE        :watch_1.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;            and Programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------

     .include  "io3687F_equ.inc"

     .export   _main

;---------------------------------------------------------
;          メインプログラム
;---------------------------------------------------------
_main:
;-----  イニシャライズ  ----------------------------------
     bsr       INIPIO:16       ;PIO ｲﾆｼｬﾗｲｽﾞ
     bsr       INIRTC:16       ;RTC ｲﾆｼｬﾗｲｽﾞ

     xor.l     er0,er0         ;ﾃﾞｰﾀのｸﾘｱ
     mov.l     er0,@Disp_DT
     mov.b     #H'77,r0l       ;ｽｷｬﾝﾃﾞｰﾀ ｾｯﾄ
     mov.b     r0l,@SCAN_DT
     mov.l     #Disp_DT,er0    ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ ｾｯﾄ
     mov.l     er0,@DISPDT_ADR
```

```
;-----  メインループ  ------------------------------------
_main_loop:                    ;ﾒｲﾝﾙｰﾌﾟ
     bsr       RTC_read:16     ;時間読み取り
     mov.l     #Time,er3       ;ER3=時間ﾃﾞｰﾀｱﾄﾞﾚｽ
     bsr       DispDT_set:16   ;表示ﾃﾞｰﾀｾｯﾄ
     bsr       DISP:16         ;表示
     bra       _main_loop

;---------------------------------------------------------
;          時間の読み取り(RTC Read)
;---------------------------------------------------------
RTC_read:
     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     btst      #7,@er3         ;RTC Busy?
     bne       RTCread_02      ;bit7:1 = Busy
     mov.b     @RTCRD_FG,r0l   ;今まで Busy だった?
     beq       RTCread_04      ;RTCRD_FG:0=NonBusy
     xor.w     r0,r0           ;RTCRD_FG:0=NonBusy Set
     mov.b     r0l,@RTCRD_FG

     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     mov.l     #Time+5,er2     ;ER2=Time+5=Time_SL
     mov.b     #3,r1l          ;R2L=loop count
RTCread_00:
     mov.b     @er3,r0l
     and.b     #H'0F,r0l
     mov.b     r0l,@er2
     subs.l    #1,er2
     mov.b     @er3,r0l
     shlr.b    r0l
     shlr.b    r0l
```

＜リスト 5-4　watch_1(watch.src) (1/3)＞

```
     shlr.b     r0l
     shlr.b     r0l
     mov.b      r0l,@er2
     subs.l     #1,er2
     adds.l     #1,er3
     dec.b      r1l
     bne        RTCread_00
     mov.b      @dp_FG,r0l
     xor.b      #H'FF,r0l
     mov.b      r0l,@dp_FG
     RTS
RTCread_02:
     mov.b      #H'01,r0l       ;RTCRD_FG=1 set
     mov.b      r0l,@RTCRD_FG   ;1→0 を検出する為
RTCread_04:
          RTS

;---------------------------------------------------------
;          表示データのセット
;---------------------------------------------------------
DispDT_set:
     mov.l      #Disp_DT,er2    ;ER2=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b      #4,r4l          ;R4L=ﾙｰﾌﾟ回数
DispDTset_00:
     xor.l      er0,er0
     mov.b      @er3,r0l
     mov.b      @(SEG_TBL,er0),r0l   ;ｾｸﾞﾒﾝﾄ変換
     mov.b      r0l,@er2        ;表示ﾃﾞｰﾀｾｯﾄ
     adds.l     #1,er2          ;ｱﾄﾞﾚｽ+1
     adds.l     #1,er3
     dec.b      r4l             ;ﾙｰﾌﾟ回数-1
     bne        DispDTset_00

     mov.b      @Disp_1,r0l
     bset       #7,r0l
     mov.b      @dp_FG,r0h
     beq        DispDTset_02
     bclr       #7,r0l
DispDTset_02:
     mov.b      r0l,@Disp_1

     rts

;---------------------------------------------------------
;          DISP
;---------------------------------------------------------
DISP:
     mov.l      @DISPDT_ADR,er3 ;ER3=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b      @SCAN_DT,r1l    ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     inc.l      #1,er3          ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ+1
     rotr.b     r1l             ;ｽｷｬﾝ移動
     bcs        DISP_00         ;一巡したか？
     mov.l      #Disp_DT,er3    ;一巡ならｱﾄﾞﾚｽ ﾘｾｯﾄ
DISP_00:
     mov.l      er3,@DISPDT_ADR
     mov.b      r1l,@SCAN_DT
     mov.b      r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b      @er3,r0l        ;表示ﾃﾞｰﾀ読み込み
     mov.b      r0l,@PDR7       ;表示ﾃﾞｰﾀ出力
     btst       #3,r0l
     bne        DISP_02
     bclr       #3,@PDR2
     bra        DISP_04
DISP_02:
     bset       #3,@PDR2
DISP_04:
     btst       #7,r0l
     bne        DISP_06
     bclr       #4,@PDR2
     bra        DISP_08
DISP_06:
     bset       #4,@PDR2
DISP_08:
     bsr        TIMER1m:16      ;表示時間=1msec
     xor.b      r0l,r0l
     mov.b      r0l,@PDR7
     bclr       #3,@PDR2
     bclr       #4,@PDR2
     mov.b      #H'FF,r0l       ;消灯
     mov.b      r0l,@PDR1
     rts

;/////  TIMER1msec  ////////////////////////////////////////
TIMER1m:
     mov.l      #H'D02,er6
TIMER1m_00:
     dec.l      #1,er6
     bne        TIMER1m_00
     rts

;/////  PIO ｲﾆｼｬﾗｲｽﾞ  ////////////////////////////////////////
INIPIO:
     mov.b      #H'02,r0l
     mov.b      r0l,@PMR1
     mov.b      #H'F0,r0l       ;ﾎﾟｰﾄ1 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR1       ;ﾎﾟｰﾄ14-17:out
     mov.b      #H'07,r0l
     mov.b      r0l,@PUCR1
     mov.b      #H'1D,r0l       ;ﾎﾟｰﾄ2 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR2       ;P23,P24:out
     mov.b      #H'FF,r0l       ;ﾎﾟｰﾄ7 ｲﾆｼｬﾗｲｽﾞ
     mov.b      r0l,@PCR7       ;ﾎﾟｰﾄ70-77:out
     rts

;/////  RTC ｲﾆｼｬﾗｲｽﾞ  ////////////////////////////////////////
INIRTC:
     mov.b      @RSECDR,r0l
     btst       #7,r0l
     bne        INIRTC
     mov.b      @RTCCR1,r0l     ;RTC ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ1 ﾘｰﾄﾞ
     bclr       #7,r0l          ;RUN=0 : 動作停止
     mov.b      r0l,@RTCCR1
     bclr       #4,r0l          ;RST=0 : ﾘｾｯﾄ解除
     mov.b      r0l,@RTCCR1

     mov.b      #B'00001000,r0l ;ｸﾛｯｸｿｰｽ : RTC
     mov.b      r0l,@RTCCSR

     mov.b      @RTCCR1,r0l     ;RTC ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ ﾘｰﾄﾞ
     or.b       #B'01000000,r0l ;24 時間ﾓｰﾄﾞ / AM
     mov.b      r0l,@RTCCR1

     bset       #7,r0l          ;RUN=1 : 動作開始
     mov.b      r0l,@RTCCR1
     rts

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

<リスト 5-4　watch_1(watch.src) (2/3)>

```
;-------------------------------------------------------
;          データ・エリア
;-------------------------------------------------------
     .section   D,data,locate=H'F780
;-----  表示  ----------------------------------------
Disp_DT:                        ;表示ﾃﾞｰﾀ ﾊﾞｯﾌｧ
Disp_1000:      .res.b     1
Disp_100:       .res.b     1
Disp_10:        .res.b     1
Disp_1:         .res.b     1
DISPDT_ADR:     .res.l     1    ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ
SCAN_DT:        .res.b     1    ;ｽｷｬﾝﾃﾞｰﾀﾊﾞｯﾌｧ
dp_FG:          .res.b     1    ;dp の点滅 FG
RTCRD_FG:       .res.b     1    ;RTC
          .align    2
```

```
;-----  時間  ----------------------------------------
Time:
Time_HH:        .res.b     1
Time_HL:        .res.b     1
Time_MH:        .res.b     1
Time_ML:        .res.b     1
Time_SH:        .res.b     1
Time_SL:        .res.b     1
          .align    2

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

＜リスト 5-4　watch_1(watch.src) (3/3)＞

ここまで出来たら時刻設定の機能を追加しプログラムを完成させましょう。まずキーの割り当てを考えましょう。キーの並びが 4×3 と電話の配列に似ているので数字の割り当ては電話と同じにします。時刻設定モードに移行する“モードキー”は電話の＊キー、SW10 に割り当てます。このモードキーは間違えて時刻設定モードにした時のキャンセルキーも兼ねています。残った SW12 を設定した時刻を有効にする Enter キーとしましょう。割り当てた結果は以下のようになります。

SW1 1	SW2 2	SW3 3
SW4 4	SW5 5	SW6 6
SW7 7	SW8 8	SW9 9
SW10 Mod	SW11 0	SW12 Ent

0～1　・・・　時刻設定モード時、時刻入力

Mod　・・・　時刻表示時　時刻設定モードへ移行
　　時刻設定モード時　時刻表示へ戻る（入力は無効）

Ent　・・・　時刻設定モード時　入力時刻設定

＜キーの割り当てと機能＞

キー入力には前に作成した“KEYIN”をそのまま流用します。キーを検出したら SW10 が押されたかを判定し、SW10 であれば時刻設定処理に入ります。時刻設定処理はメインループのように独立したもう一つの処理ループを作り、その中で各キーの入力と入力されたキーの表示、そして SW12・Ent キーが押されたら RTC の再設定を行います。RTC の再設定は RTCCR1 で動作を停止させてから RSECDR、RMINDR、RHRDR に設定時刻をセットし、再び RTCCR1 で動作を再開させます。

時刻設定も含め完成させたプログラムが＜watch_2＞です。以下にリストを示します。

```
;-------------------------------------------------------
;
;   FILE        :watch_2.src
;   DATE        :Thu, Sep 02, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;
;-------------------------------------------------------
;       ダイナミックスキャン・キーマトリックス応用プログラム
;               内蔵RTC使用、時計プログラム
;
;時分表示の時計プログラムです。秒は1桁目の dp の点滅で表示しま
;す。 mod キーを押すと時刻設定モードになりますので 0～9 のキーで時
;刻を入力して下さい。入力時刻をセットするには ent キーを押して下さ
;い。入力した時刻で時間を刻み始めます。尚、秒は 00 からです。時刻
;設定をキャンセルして今までの時刻表示に戻るには、再度 mod キーを
;押して下さい。時刻設定モードで ent を押さない限り時刻は変更されま
;せん。
;
;■キー配列と役割
; +---+---+---+ mod :時刻表示時      :時刻設定モードへ
; l 1 l 2 l 3 l      時刻設定モード時:時刻設定モードキャンセル
; +---+---+---+
; l 4 l 5 l 6 l ent :時刻表示時      :ー
; +---+---+---+     :時刻設定モード時:時刻設定決定
; l 7 l 8 l 9 l
; +---+---+---+ 0～9:時刻表示時      :ー
; lmodl 0 lentl     :時刻設定モード時:時刻入力
; +---+---+---+
;


          .include   "io3687F_equu.inc"

     .export         _main

;-------------------------------------------------------
;           メインプログラム
;-------------------------------------------------------
_main:
;-----  イニシャライズ  ----------------------------------
     bsr        INIPIO:16       ;PIO イニシャライズ
     bsr        INIRTC:16       ;RTC イニシャライズ

     xor.l      er0,er0         ;データのクリア
     mov.w      r0,@KEY_FG
     mov.l      er0,@Disp_DT
     mov.b      #H'77,r0l       ;スキャンデータ セット
     mov.b      r0l,@SCAN_DT
     mov.l      #Disp_DT,er0    ;表示データアドレス セット
     mov.l      er0,@DISPDT_ADR

;-----  時刻表示モード  ----------------------------------
MAIN_LOOP:                      ;メインループ
     bsr        KEYIN:16        ;キー入力
     mov.w      @KEY_FG,r0      ;キー入力有り?
     beq        MAINLOOP_00
     mov.w      @KEY_No,r0
     cmp.w      #H'A,r0         ;設定キー?
     beq        SETTING_LOOP    ;設定キーならセッティングモードへ
MAINLOOP_00:
     bsr        RTC_read:16     ;時間読み取り
     mov.l      #Time,er3       ;ER3=時間データアドレス
     bsr        DispDT_set:16   ;表示データセット
     bsr        DISP:16         ;表示
     bra        MAIN_LOOP
```

```
;-----  セッティングモード  ------------------------------
SETTING_LOOP:
     mov.l      @Time,er0
     mov.l      er0,@KEY_BUF
SETTINGLOOP_00:
     bsr        KEYIN:16        ;キー入力
     mov.w      @KEY_FG,r0
     beq        SETTINGLOOP_02
     mov.w      @KEY_No,r0
     cmp.w      #H'A,r0         ;ESC?
     beq        MAIN_LOOP
     cmp.w      #H'C,r0
     beq        SETTINGLOOP_10  ;set?
     bsr        KeyNo_set:16
SETTINGLOOP_02:
     mov.l      #KEY_BUF,er3    ;ER3=Key バッファ アドレス
     bsr        DispDT_set:16   ;表示データセット
     bsr        DISP:16
     bra        SETTINGLOOP_00

;-----  時刻のセット  ------------------------------------
SETTINGLOOP_10:
     mov.l      @KEY_BUF,er0    ;時間に合わない入力はクリア
     cmp.w      #H'050A,e0      ;時のチェック
     bcs        SETTINGLOOP_12
     xor.w      e0,e0
SETTINGLOOP_12:
     cmp.b      #H'06,r0h       ;分のチェック
     bcs        SETTINGLOOP_14
     xor.b      r0h,r0h
SETTINGLOOP_14:
     mov.l      er0,@KEY_BUF    ;
     mov.l      er0,@Time       ;

     mov.l      #Time,er3       ;ER3=時間データアドレス
     bsr        DispDT_set:16   ;表示データセット

     mov.b      @RSECDR,r0l
     btst       #7,r0l          ;RTC busy?
     bne        SETTINGLOOP_10
     mov.b      @RTCCR1,r0l     ;RTC コントロールレジスタ 1 リード
     bclr       #7,r0l          ;RUN=0 : 動作停止
     mov.b      r0l,@RTCCR1
     bset       #4,r0l          ;RST=1 : リセット
     mov.b      r0l,@RTCCR1
     bclr       #4,r0l          ;RST=0 : リセット解除
     mov.b      r0l,@RTCCR1
     mov.b      #B'00001000,r0l ;クロックソース : RTC
     mov.b      r0l,@RTCCSR
     mov.l      @Time,er0
     shll.b     r0h
     shll.b     r0h
     shll.b     r0h
     shll.b     r0h
     or.b       r0h,r0l
     mov.b      r0l,@RMINDR     ;分データレジスタ セット
     mov.w      e0,r0
     shll.b     r0h
     shll.b     r0h
     shll.b     r0h
     shll.b     r0h
     or.b       r0h,r0l
     mov.b      r0l,@RHRDR      ;時データレジスタ セット
     xor.b      r0l,r0l         ;秒データレジスタ セット
     mov.b      r0l,@RSECDR
     mov.b      @RTCCR1,r0l     ;RTC コントロールレジスタ リード
     or.b       #B'01000000,r0l ;24 時間モード / AM
     mov.b      r0l,@RTCCR1
```

<リスト 5-5 watch_2.sec (1/3)>

```
     bset      #7,r0l          ;RUN=1 : 動作開始
     mov.b     r0l,@RTCCR1

     bra       MAIN_LOOP

;---------------------------------------------------------
;          KEYIN
;---------------------------------------------------------
KEYIN:
     mov.b     #H'FF,r3l       ;scan off data
     mov.b     r3l,@PDR1       ;scan off
     mov.b     #H'77,r1l       ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     mov.b     #D'4,r1h        ;R1H=ﾙｰﾌﾟ回数
     mov.w     #D'1,r2         ;R2=ｷｰ番号
KEYIN_00:
     mov.b     r1l,r0l
     and.b     #H'F0,r0l
     mov.b     r0l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     @PDR1,r0l       ;ｷｰ読み込み
     mov.b     r3l,@PDR1       ;scan off
     not       r0l             ;反転
     and.b     #H'07,r0l       ;ｷｰは押されていた?
     bne       KEYIN_02        ;ne = 押されていた
     add.w     #D'3,r2         ;ｷｰ番号 +3
     rotr.b    r1l             ;次のｽｷｬﾝへ
     dec.b     r1h             ;ﾙｰﾌﾟ回数 -1
     bne       KEYIN_00
     bra       KEYIN_10        ;ﾙｰﾌﾟ終了
KEYIN_02:                      ;ｷｰ検出
     btst      #0,r0l          ;bit0 ﾁｪｯｸ
     bne       KEYIN_06
     btst      #1,r0l          ;bit1 ﾁｪｯｸ
     beq       KEYIN_04
     inc.w     #1,r2           ;ｷｰ番号 +1
     bra       KEYIN_06
KEYIN_04:
     inc.w     #2,r2           ;ｷｰ番号 +2
KEYIN_06:
     mov.w     @KEY_No,r0      ;ｷｰの押し続けﾁｪｯｸ
     cmp.w     r0,r2
     beq       KEYIN_12        ;押し続け ﾌﾗｸﾞをｸﾘｱ
     mov.w     r2,@KEY_No      ;入力有り ｷｰ番号ｾｯﾄ
     mov.w     #H'1,r0
     mov.w     r0,@KEY_FG      ;ｷｰﾌﾗｸﾞ ｾｯﾄ
     rts
KEYIN_10:
     xor.w     r0,r0           ;ｷｰ番号 ｸﾘｱ
     mov.w     r0,@KEY_No
KEYIN_12:
     xor.w     r0,r0           ;ｷｰﾌﾗｸﾞ ｸﾘｱ
     mov.w     r0,@KEY_FG
     rts

;---------------------------------------------------------
;          ｷｰ番号のｾｯﾄ
;---------------------------------------------------------
KeyNo_set:
     mov.b     @KEY_100,r0h
     mov.b     @KEY_10,r0l
     mov.w     r0,e0
     mov.w     @KEY_No,r0
     cmp.w     #H'B,r0
     bne       KeyNoset_00
     xor.w     r0,r0
KeyNoset_00:
     mov.b     @KEY_1,r0h
     mov.l     er0,@KEY_BUF
     rts
```

```
;---------------------------------------------------------
;          時間の読み取り(RTC Read)
;---------------------------------------------------------
RTC_read:
     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     btst      #7,@er3         ;RTC Busy?
     bne       RTCread_02      ;bit7:1 = Busy
     mov.b     @RTCRD_FG,r0l   ;今まで Busy だった?
     beq       RTCread_04      ;RTCRD_FG:0=NonBusy
     xor.w     r0,r0           ;RTCRD_FG:0=NonBusy Set
     mov.b     r0l,@RTCRD_FG

     mov.l     #RSECDR,er3     ;ER3=RSECDR=H'F728
     mov.l     #Time+5,er2     ;ER2=Time+5=Time_SL
     mov.b     #3,r1l          ;R2L=loop count
RTCread_00:
     mov.b     @er3,r0l
     and.b     #H'0F,r0l
     mov.b     r0l,@er2
     subs.l    #1,er2
     mov.b     @er3,r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     mov.b     r0l,@er2
     subs.l    #1,er2
     adds.l    #1,er3
     dec.b     r1l
     bne       RTCread_00
     mov.b     @dp_FG,r0l
     xor.b     #H'FF,r0l
     mov.b     r0l,@dp_FG
     RTS
RTCread_02:
     mov.b     #H'01,r0l       ;RTCRD_FG=1 set
     mov.b     r0l,@RTCRD_FG   ;1→0 を検出する為
RTCread_04:
     RTS

;---------------------------------------------------------
;          表示データのｾｯﾄ
;---------------------------------------------------------
DispDT_set:
     mov.l     #Disp_DT,er2    ;ER2=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b     #4,r4l          ;R4L=ﾙｰﾌﾟ回数
DispDTset_00:
     xor.l     er0,er0
     mov.b     @er3,r0l
     mov.b     @(SEG_TBL,er0),r0l   ;ｾｸﾞﾒﾝﾄ変換
     mov.b     r0l,@er2        ;表示ﾃﾞｰﾀｾｯﾄ
     adds.l    #1,er2          ;ｱﾄﾞﾚｽ+1
     adds.l    #1,er3
     dec.b     r4l             ;ﾙｰﾌﾟ回数-1
     bne       DispDTset_00

     mov.b     @Disp_1,r0l
     bset      #7,r0l
     mov.b     @dp_FG,r0h
     beq       DispDTset_02
     bclr      #7,r0l
DispDTset_02:
     mov.b     r0l,@Disp_1

     rts

;---------------------------------------------------------
;          DISP
;---------------------------------------------------------
```

<リスト 5-5 watch_2.sec (2/3)>

```
DISP:
     mov.l     @DISPDT_ADR,er3 ;ER3=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     mov.b     @SCAN_DT,r1l    ;R1L=ｽｷｬﾝﾃﾞｰﾀ
     inc.l     #1,er3          ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ+1
     rotr.b    r1l             ;ｽｷｬﾝ移動
     bcs       DISP_00         ;一巡したか？
     mov.l     #Disp_DT,er3    ;一巡ならｱﾄﾞﾚｽ ﾘｾｯﾄ
DISP_00:
     mov.l     er3,@DISPDT_ADR
     mov.b     r1l,@SCAN_DT
     mov.b     r1l,@PDR1       ;ｽｷｬﾝ出力
     mov.b     @er3,r0l        ;表示ﾃﾞｰﾀ読み込み
     mov.b     r0l,@PDR7       ;表示ﾃﾞｰﾀ出力
     btst      #3,r0l
     bne       DISP_02
     bclr      #3,@PDR2
     bra       DISP_04
DISP_02:
     bset      #3,@PDR2
DISP_04:
     btst      #7,r0l
     bne       DISP_06
     bclr      #4,@PDR2
     bra       DISP_08
DISP_06:
     bset      #4,@PDR2
DISP_08:
     bsr       TIMER1m:16      ;表示時間=1msec
     xor.b     r0l,r0l
     mov.b     r0l,@PDR7
     bclr      #3,@PDR2
     bclr      #4,@PDR2
     mov.b     #H'FF,r0l       ;消灯
     mov.b     r0l,@PDR1
     rts

;/////  TIMER1msec  ////////////////////////////////////////
TIMER1m:
     mov.l     #H'D02,er6
TIMER1m_00:
     dec.l     #1,er6
     bne       TIMER1m_00
     rts

;/////  PIO ｲﾆｼｬﾗｲｽﾞ  //////////////////////////////////////
INIPIO:
     mov.b     #H'02,r0l
     mov.b     r0l,@PMR1
     mov.b     #H'F0,r0l       ;ﾎﾟｰﾄ1 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR1       ;ﾎﾟｰﾄ14-17:out
     mov.b     #H'07,r0l
     mov.b     r0l,@PUCR1
     mov.b     #H'1D,r0l       ;ﾎﾟｰﾄ2 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR2       ;P23,P24:out
     mov.b     #H'FF,r0l       ;ﾎﾟｰﾄ7 ｲﾆｼｬﾗｲｽﾞ
     mov.b     r0l,@PCR7       ;ﾎﾟｰﾄ70-77:out
     rts

;/////  RTC ｲﾆｼｬﾗｲｽﾞ  //////////////////////////////////////
INIRTC:
     mov.b     @RSECDR,r0l
     btst      #7,r0l
     bne       INIRTC
     mov.b     @RTCCR1,r0l     ;RTC ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ1 ﾘｰﾄﾞ
     bclr      #7,r0l          ;RUN=0 : 動作停止
     mov.b     r0l,@RTCCR1
     bclr      #4,r0l          ;RST=0 : ﾘｾｯﾄ解除
     mov.b     r0l,@RTCCR1

     mov.b     #B'00001000,r0l ;ｸﾛｯｸｿｰｽ : RTC
     mov.b     r0l,@RTCCSR

     mov.b     @RTCCR1,r0l     ;RTC ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ ﾘｰﾄﾞ
     or.b      #B'01000000,r0l ;24 時間ﾓｰﾄﾞ / AM
     mov.b     r0l,@RTCCR1

     bset      #7,r0l          ;RUN=1 : 動作開始
     mov.b     r0l,@RTCCR1
     rts

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

;-------------------------------------------------------
;          ﾃﾞｰﾀ･ｴﾘｱ
;-------------------------------------------------------
          .section  D,data,locate=H'F780
;-----  ｷｰ入力  ----------------------------------------
KEY_FG:         .res.w    1     ;ｷｰ入力ﾌﾗｸﾞ
KEY_No:         .res.w    1     ;ｷｰ番号
KEY_BUF:
KEY_1000:       .res.b    1
KEY_100:        .res.b    1
KEY_10:         .res.b    1
KEY_1:          .res.b    1
          .align    2

;-----  表示  ------------------------------------------
Disp_DT:                       ;表示ﾃﾞｰﾀ ﾊﾞｯﾌｧ
Disp_1000:      .res.b    1
Disp_100:       .res.b    1
Disp_10:        .res.b    1
Disp_1:         .res.b    1
DISPDT_ADR:     .res.l    1     ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ
SCAN_DT:        .res.b    1     ;ｽｷｬﾝﾃﾞｰﾀﾊﾞｯﾌｧ
dp_FG:          .res.b    1     ;dp の点滅 FG
RTCRD_FG:       .res.b    1     ;RTC
          .align    2
;-----  時間  ------------------------------------------
Time:
Time_HH:        .res.b    1
Time_HL:        .res.b    1
Time_MH:        .res.b    1
Time_ML:        .res.b    1
Time_SH:        .res.b    1
Time_SL:        .res.b    1
          .align    2

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

<リスト 5-5 watch_2.sec (3/3)>

より現実的な時計プログラム

前章で RTC を使用した時計プログラムを作成しました。この章ではより現実的な時計プログラムを作成しましょう。これで実務で要求される力が身に付きます。

まず、世の中にあるデジタル時計を思い浮かべてみて下さい。前章で作成したようなテンキーで時刻を設定するような時計はまずありません。よく見られるのは時刻設定ボタンで時刻設定モードにし、時／分ボタンで時刻を設定する方法、もしくは、時→分と順に＋／－ボタンで時刻を合わせていく方法です。これなら必要なスイッチは3つ程度で済みます。また、設定した時刻になったらブザーやメロディを鳴らすアラーム機能もよく見ます。

これから作成するプログラムはこれら世の中に出回っている時計をお手本に次の機能を備えるのもとします。

■ 現在時刻の設定

マトリックスキーに時刻設定キーと時キー、分キーを割り付け時刻を設定します。時刻設定キーを暫らく押すと現在時刻設定モードになり、時／分キーで時刻を合わせ時刻設定キーで確定とします。尚、時刻設定キーを暫らく押すのは押し間違え等で容易に時刻設定モードに行かせない為です。

■ タイマ機能

アラーム機能でも良いのですがせっかくマイコンを使っているので、ON 時刻と OFF 時刻の2つの時刻をセットできるようにし、それぞれの時間が来たらポート出力を ON／OFF するタイマ機能を搭載することにします。この信号を利用して例えばソリッドステートリレー（SSR）などで外部機器を制御すれば任意の時間に機器を ON／OFF する事ができるようになります。ON／OFF 時刻の設定にはそれぞれ ON 時刻キー／OFF 時刻キーを割り付ける事にします。各時刻の設定方法は現在時刻の設定に倣い、暫らく押し続けることで設定モードに移行、時／分キーで ON／OFF 時刻を合わせ時刻設定キーで確定とします。

■ 早送り機能

時刻を設定するのに使用するキーは時／分のみですので、それぞれのキーが押されたら時刻を＋1進めます。しかし、1、2コ進める程度なら問題ありませんが20、30進めるとなるといちいちその数分だけキーを押すのは大変です。そこで、押し続けると自動的に時刻を進める機能を入れます。また、一定時間以上押しつづけると進める早さを早くする“早送り機能”も盛り込みます。

■ 秒の表現

前章のプログラムでは dp で秒を表していました。しかし dp で秒を表す方法は一般的ではありません。それに点滅の周期が2秒（点灯・消灯時間がそれぞれ1秒づつ）なのも違和感を感じます。そこで今回は1秒周期で点滅する信号を作りそれを秒の表示とします。1秒毎にピッ、ピッと光るイメージです。この信号を時・分の間にある ‘:’ に見立てて、空いているポート 5 に 2bit 出力します。

■ 各時刻設定モード中の表現

各時刻設定時には現在設定中であることを示さなければなりません。そうしないと現在時刻の表示と区別がつかないからです。空いているポートの LED を点灯させて区別する方法もありますが、ここではよりはっきりと区別する為に 7segLED を点滅させる事にします。こうすることで現在設定中であることがひと目で判断できます。では ON／OFF 時刻設定の区別はどのようにすればよいでしょう。7seg 上での表現は限られていますし、このために 1bit もしくは 2bit 新たに使用して区別させるのも無駄な感じがします。そこで設定時には使用する必

要のない秒表現の 2bit を使用して ON／OFF 設定の区別をつける事にします。

これで大体のイメージが出来上がりました。使用するキーは、時刻設定キー・ON 時刻キー・OFF 時刻キー・時キー・分キー、の計5つです。各キーの動作及び割り付けを以下のようにしました。

SW1 ー	SW2 ー	SW3 ー
SW4 ー	SW5 ー	SW6 ー
SW7 ON 時刻	SW8 OFF 時刻	SW9 ー
SW10 分	SW11 時	SW12 時刻 設定

	ワンショット	押し続け
時刻設定キー	決定 ※	現在時刻設定
ON 時刻キー	ON 時刻確認表示	ON 時刻設定
OFF 時刻キー	OFF 時刻確認表示	OFF 時刻設定
時キー	時＋1 ※	時早送り ※
分キー	分＋1 ※	分早送り ※

※時刻設定時のみ

＜キーの割り付けと各動作＞

タイマの出力、そして秒の点滅表示はポート5に出力して、LED で状態を表示します。

P57	P56	P55	P54	P53	P52	P51	P50
※ タイマ B1 割り込み 確認	※ RTC 割り込み 確認	秒表示 OFF 時刻 表示／設定	秒表示 ON 時刻 表示／設定	ー	ー	ー	ON/OFF タイマ出力

※デバッグの為に出力した信号です

＜ポート 5 の出力＞

■ プログラム

さて今回のプログラムでは行うことがだいぶ増えてきました。前章までのようにメインループに全てを盛り込んでしまうと少々複雑になってしまいます。また秒や 7seg の点滅、キーの押し続けなど時間を管理する必要があります。そこで今回はタイマ割り込みを使用します。また RTC の更新検出も割り込みで行います。メインループ、各割り込み処理のリストを抜粋しそれぞれ説明していきます。

● メインループ

```
;-----  ﾒｲﾝﾙｰﾌﾟ  ----------------------------------------
MAIN_LOOP:
     jsr       @KEYACT         ;キー入力と対応処理
     mov.b     @CP_FLG,r0l     ;CP_FLG=1 なら MAINLOOP_10 へ
     bne       MAINLOOP_10
MAINLOOP_02:
     jsr       @SETDISP        ;表示ﾃﾞｰﾀ ｾｯﾄ
     bra       MAIN_LOOP

MAINLOOP_10:
     mov.b     @MODE,r0l
     btst      #7,r0l          ;時刻設定中なら MAINLOOP_02
     bne       MAINLOOP_02
     xor.b     r0l,r0l         ;CP_FLG ｸﾘｱ
     mov.b     r0l,@CP_FLG
     jsr       @RLY            ;時刻判定と外部機器制御
     bra       MAIN_LOOP
```

“KEYACT”

このサブルーチン‘KEYACT’ではキーが押されていたらその押されたキーに対応する動作を行います。また、キー押し続けの管理も行います。尚、キーの検出は後述のタイマ B1 割り込み内で行っています。

“RLY”

現在時刻と ON／OFF 時刻を比較し P50 の出力を決定します。常に比較する必要はなく、時刻が更新されたタイミング時のみ比較を行います。尚、ON 時刻=OFF 時刻の場合は比較を行いません。

“SETDISP”

表示データをセットします。ダイナミックスキャンは後述のタイマ B1 内で行っているのでこのサブルーチン内ではバッファに各表示データをセットするのみです。また、設定モード時の点滅や秒の点滅の時間管理もこのサブルーチン内で行っています。

- **RTC 割り込み**

```
;------------------------------------------------------------
;          ＲＴＣ割り込み
;------------------------------------------------------------
;割り込み間隔：1sec

RTCINT:
     push.l    er0

     mov.b     #H'01,r0l       ;ﾌﾗｸﾞ ｾｯﾄ
     mov.b     r0l,@RTC_FLG
     mov.b     r0l,@CP_FLG

     bnot      #6,@PDR5        ;ﾃﾞﾊﾞｯｸﾞ用 P56 ﾄｸﾞﾙ出力

     bclr      #6,@IRR1        ;IRRTA ｸﾘｱ

     pop.l     er0

     rte
```

‘RTC_FLG’のセット

このフラグをセットする事で表示の更新が行われます。

‘CP_FLG’のセット

このフラグをセットする事で現在時刻と ON／OFF 時刻の比較が行われます。

‘P56’への出力

デバッグの為のトグル出力です。

‘IRRTA’のクリア

続けて RTC 割り込みをかけるために、レジスタ IRR1 の bit6:IRRTA をクリアします。

- **タイマ B1 割り込み**

```
;---------------------------------------------------------
;          タイマＢ１割り込み
;---------------------------------------------------------
;割り込み間隔：1msec

TB1INT:
     push.l    er0
     push.l    er1
     push.l    er2
     push.l    er3

     jsr       @TMRCNT         ;ｿﾌﾄﾀｲﾏ管理
     jsr       @KEYIN2         ;ｷｰ入力
     jsr       @DISP           ;表示

     bnot      #7,@PDR5        ;ﾃﾞﾊﾞｯｸﾞ用 P57 ﾄｸﾞﾙ出力

     bclr      #5,@IRR2        ;IRRTB1 ｸﾘｱ

     pop.ler3
     pop.ler2
     pop.ler1
     pop.ler0

     rte
```

“TMRCNT”

プログラム中で使用しているソフトタイマのカウントを行います。ソフトタイマは以下のような構造のデクリメントタイマバッファを複数個持たせ時間を管理します。

+0	ステータス（タイマの状態）	0 = ストップ 1 = カウント動作（ワンショット） 2 = カウント動作（エンドレス）
+1	フラグ（カウントアップの有無）	0 = カウント中 1 = カウントアップ
+2	カウント初期値	2byte データ カウントアップ時にこの初期値がカウンタにロードされます。
+3		
+4	カウンタ	2byte データ デクリメントカウンタ
+5		
+6	予備	ー
+7		ー

まずステータスをチェックし、1 又は 2 であればカウンタの値をデクリメントします。デクリメントした結果カウンタの値が 0 になったらフラグをセットしカウントアップが発生した事を示します。また、ステータスをチェックしワンショット動作（ステータス=1）なら次回はカウントしないのでタイマを停止（ステータス=0）します。

メインでソフトタイマを動作させるには、カウンタ、カウンタ初期値のセット→フラグをクリア→ステータスのセット、の順で任意のソフトタイマバッファにセットすれば動作を開始します。また、カウントアップを知るにはフラグをチェックします。今回ソフトタイマを使用するのは、

- キーのチャタリング待ち時間
- キーの押し続け時間
- 設定モード時点滅や ON／OFF 時刻表示の時間
- 秒の点灯時間

の計 4 つです。

“KEYIN2”

マトリックスキーの読み取りを行います。今まで作成したプログラムではサブルーチン内ですべてのキーをスキャンして読み込んでいましたが、このKEYIN2では割り込み単位毎に処理を進めていきます。つまり、KEYIN2 を1CALL すると1スキャンラインのキーを読み取ります。全スキャンライン読み込んだらキー入力の有無を判定し、入力があればチャタリング除去の為 50msec 後にダブルリードを行いキーの入力を確定します。

“DISP”

ダイナミックスキャンで 7segLED を表示します。表示時間は割り込み間隔の 1msec となります。

‘P57’への出力

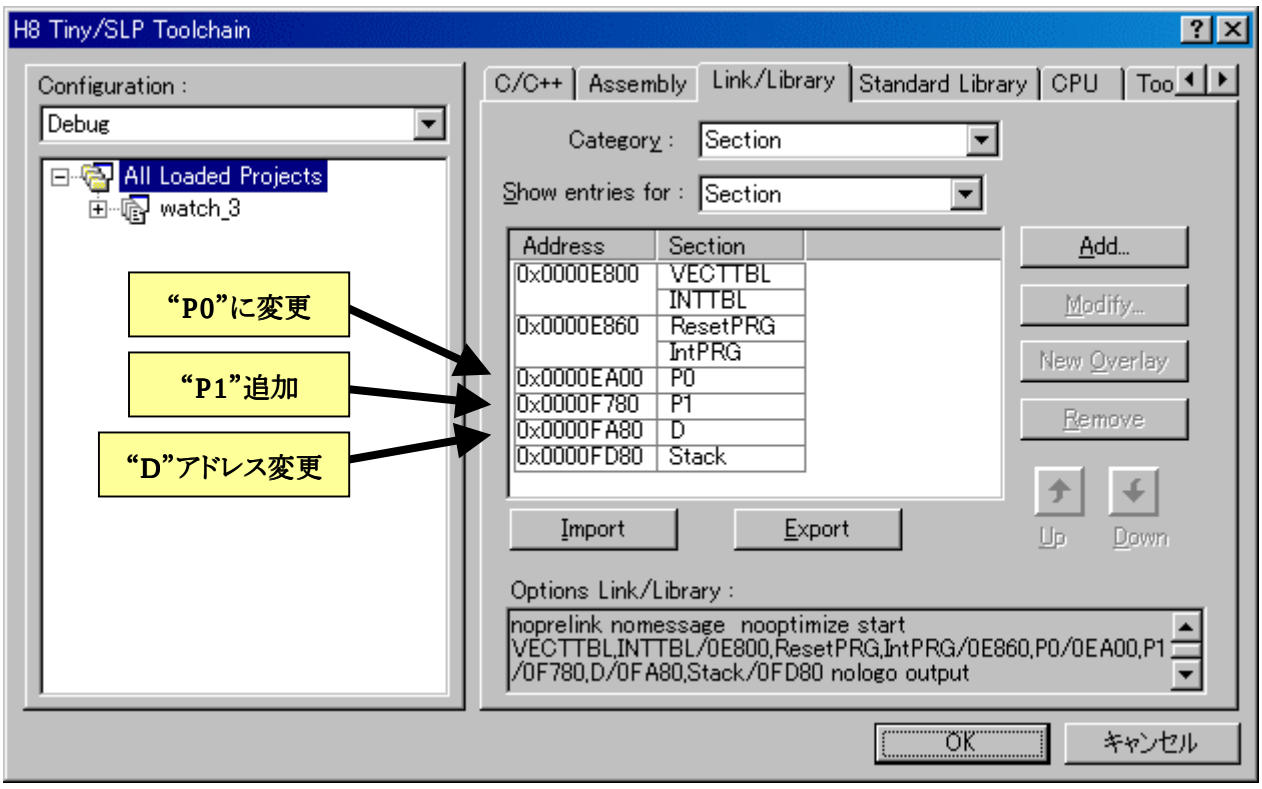
デバッグの為のトグル出力です。

‘IRRTB1’のクリア

続けてタイマ B1 割り込みをかけるために、レジスタ IRR2 の bit5:IRRTB1 をクリアします。

次頁に作成した‘watch_3’のリストを示します。

尚、プログラムサイズが E800h～EFFFh で収まらなかったので、新たに先頭アドレス F780h のセクション“P1”を作り、収まらなかったプログラムをセクション“P1”に割り付けました。又、セクション“P1”で F780h 番地から使用してしまうのでデータセクション‘D’の先頭アドレスを FA80h に変更しました。尚、“P”セクションを連番にする為に“P0”とします。プログラムのロケーションを変更するにはセクションで区切る方法以外に‘.org’命令で変更する方法もありますが、この命令を使用してでき上がる mot ファイルは‘.org’で指定する前のアドレスから‘.org’で指定したアドレスまで全て 00h で埋められてしまいます。例えば watch_3 ですと F000h～F77Fh まで全て 00h です。このエリアは内部 I/O レジスタですのでハイパーH8 でロードした際、不要なデータを I/O レジスタに書き込んでしまいます。ですので、プログラムエリアが跨ぐ場合はセクションで区切って下さい。変更・追加したセクションを以下に示します。



```
;-------------------------------------------------------
;
;   FILE        :watch_3.src
;   DATE        :Wed, Aug 25, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;           and programed by Toyo-linx,co.,ltd. / Y.Furukawa.
;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; ２４時間時計・デイリータイマ付き
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; ■機能
; ・２４時間時計
; ・ＯＮ／ＯＦＦタイマ（デイリータイマ）
;     設定したＯＮ／ＯＦＦ時間になると、ポート P50 からレベル
;     信号を出力して外部機器を制御する。ON で High、OFF で Low。
; ・キー割り付け
;   ＋－－－＋－－－＋－－－＋  SW 7 : ON 時刻キー
;   |       |       |       |     ON 時刻の表示と ON 時刻設定
;   | SW 1  | SW 2  | SW 3  |  SW 8 : OFF 時刻キー
;   ＋－－－＋－－－＋－－－＋   OFF 時刻の表示と OFF 時刻設定
;   |       |       |       |  Sw12 : ＳＥＴキー
;   | SW 4  | SW 5  | SW 6  | 各設定時刻の決定と現在時刻設定
;   ＋－－－＋－－－＋－－－＋  SW10 : 分キー
;   |ON 時刻|OF 時刻|       |          時刻設定時、分を＋１
;   | SW 7  | SW 8  | SW 9  |  SW11 : 時キー
;   ＋－－－＋－－－＋－－－＋          時刻設定時、時を＋１
;   |  分   |  時   |ＳＥＴ|
;   | SW10  | SW11  | SW12  |
;   ＋－－－＋－－－＋－－－＋
; ・各時刻設定
;   以下のキーを長押しすることによって時刻設定モードになる。
;     現在時刻  ：ＳＥＴキー
;     ＯＮ時刻  ：ON 時刻キー
;     ＯＦＦ時刻：OFF 時刻キー
; ・時／分の早送り機能
;     時刻設定は送る方向しかないが、その代わりに押し続けると
;     送る早さが変わる“早送り”機能を備えている。
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; ※セクションの指定について
;   プログラムサイズが長く H'E800～H'EFFF では収まらなかった
; ので、コードセクションを“P0”“P1”と二つに分け、P0 を H'E800
; ～、P1 を H'F780～と割り付けました。またそれに伴い“D”セク
; ションのアドレスも H'FA80 に変更しています。
;
;-------------------------------------------------------

;=======================================================
;         履歴
;=======================================================
;2004.08.25          watch_3 作成、ｵﾘｼﾞﾅﾙからの移植
;     09.06          記述修正
;     09.13          ﾛｹｰｼｮﾝ変更とそれに伴う命令変更(bsr->bsr
;     09.17          P1 ｾｸｼｮﾝ追加 ｾｸｼｮﾝ P1,code,H'F780～
;                    それに伴い D ｾｸｼｮﾝ変更 Section'D'= H'FA80

        .include  "io3687F_equ.inc"

        .export         _main
        .export         RTCINT
        .export         TB1INT

;=======================================================
;         定数
;=======================================================
;-----  時/分キー押し続けに関する定数  ------------------
KEY_WAITCNST1   .equ D'1000     ;ｽﾛｰﾘﾋﾟｰﾄまでの wait 時間(msec
KEY_WAITCNST2   .equ D'500      ;ｽﾛｰﾘﾋﾟｰﾄの wait 時間(msec)
KEY_WAITCNST3   .equ D'100      ;ﾌｧｰｽﾄﾋﾟｰﾄの wait 時間(msec)
KEY_WAITCNST4   .equ D'1000     ;ON/OFF 設定ﾓｰﾄﾞ移行時間(msec
KEY_RPTCNST     .equ D'5        ;ﾌｧｰｽﾄﾘﾋﾟｰﾄまでのｶｳﾝﾄ定数

;-----  表示時間に関する定数  ----------------------------
DISP_DISPCNST1  .equ D'5000     ;時刻確認表示時間(msec)
DISP_DISPCNST2  .equ D'1        ;時刻設定後の確認表示時間
DISP_BLINKCNST1 .equ D'250      ;時刻設定時点滅間隔時間(msec
DISP_CLNCNST    .equ D'100      ;“：”点灯時間(msec)

;-----  ソフトタイマに関する定数  ------------------------
TMR_USECNST     .equ D'4        ;使用するｿﾌﾄﾀｲﾏの数

;=======================================================
;         メインプログラム
;=======================================================
    .section  P0,code
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ----------------------------------------
    jsr       @INITPIO        ;PIO ｲﾆｼｬﾗｲｽﾞ
    jsr       @INITTB1        ;ﾀｲﾏ B1 ｲﾆｼｬﾗｲｽﾞ
    jsr       @INITTZ         ;ﾀｲﾏ Z ｲﾆｼｬﾗｲｽﾞ
    jsr       @INITRTC        ;RTC ｲﾆｼｬﾗｲｽﾞ
    jsr       @INITINTR       ;割り込み ｲﾆｼｬﾗｲｽﾞ

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

    mov.w     #SEG_BUF+2,r3   ;ﾀﾞｲﾅﾐｯｸｽｷｬﾝ ｲﾆｼｬﾗｲｽﾞ
    mov.w     r3,@SEGDT_ADR
    mov.b     #B'11101110,r0l
    mov.b     r0l,@SCAN_DT

    mov.b     #H'01,r0l
    mov.b     r0l,@SET_PNT    ;時刻設定位置 ｲﾆｼｬﾗｲｽﾞ
    mov.b     r0l,@RTC_FLG

    andc      #H'7F,CCR       ;割り込み許可

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------------------------------
MAIN_LOOP:
    jsr       @KEY            ;キー入力と対応処理
    mov.b     @CP_FLG,r0l     ;CP_FLG=1 なら MAINLOOP_10 へ
    bne       MAINLOOP_10
MAINLOOP_02:
    jsr       @SETDISP        ;表示ﾃﾞｰﾀ ｾｯﾄ
    bra       MAIN_LOOP

MAINLOOP_10:
    mov.b     @MODE,r0l
    btst      #7,r0l          ;時刻設定中なら MAINLOOP_02
    bne       MAINLOOP_02
    xor.b     r0l,r0l         ;CP_FLG ｸﾘｱ
    mov.b     r0l,@CP_FLG
    jsr       @RLY            ;時刻判定と外部機器制御
    bra       MAIN_LOOP

;=======================================================
;         割り込み処理
;=======================================================
;-------------------------------------------------------
;         ＲＴＣ割り込み
;-------------------------------------------------------
;割り込み間隔：1sec
RTCINT:
    push.l    er0
```

<リスト 5-6 watch_3.sec (1/11)>

```
     mov.b     #H'01,r0l       ;ﾌﾗｸﾞ ｾｯﾄ
     mov.b     r0l,@RTC_FLG
     mov.b     r0l,@CP_FLG

     bnot      #6,@PDR5        ;ﾃﾞﾊﾞｯｸﾞ用 P56 ﾄｸﾞﾙ出力

     bclr      #6,@IRR1        ;IRRTA ｸﾘｱ

     pop.l     er0

     rte

;-------------------------------------------------------
;          タイマＢ１割り込み
;-------------------------------------------------------
;割り込み間隔：1msec

TB1INT:
     push.l    er0
     push.l    er1
     push.l    er2
     push.l    er3

     jsr       @TMRCNT         ;ｿﾌﾄﾀｲﾏ管理
     jsr       @KEYIN2         ;ｷｰ入力
     jsr       @DISP           ;表示

     bnot      #7,@PDR5        ;ﾃﾞﾊﾞｯｸﾞ用 P57 ﾄｸﾞﾙ出力

     bclr      #5,@IRR2        ;IRRTB1 ｸﾘｱ

     pop.ler3
     pop.ler2
     pop.ler1
     pop.ler0

     rte


;=======================================================
;          サブルーチンプログラム
;=======================================================
;-------------------------------------------------------
;          リレー制御
;-------------------------------------------------------
RLY:
     mov.b     @TIME_HR,r0h    ;各時刻ﾘｰﾄﾞ
     mov.b     @TIME_MIN,r0l   ; R0=現在時刻
     mov.b     @ONTIM_HR,r1h   ; R1=ON 時刻
     mov.b     @ONTIM_MIN,r1l  ; R2=OFF 時刻
     mov.b     @OFFTIM_HR,r2h
     mov.b     @OFFTIM_MIN,r2l

     cmp.w     r2,r1           ;ON 時刻<>OFF 時刻
;    beq       RLY_01          ;ON 時刻=OFF 時刻ならﾘﾚｰ on
     bcc       RLY_10
                               ;OFF 時刻＞ON 時刻
     cmp.w     r2,r0           ;現在時刻<>OFF 時刻
     bcc       RLY_00          ;現在時刻＞OFF 時刻ならﾘﾚｰ off
     cmp.w     r1,r0           ;現在時刻<>ON 時刻
     bcs       RLY_00          ;現在時刻＜ON 時刻ならﾘﾚｰ off
     bra       RLY_01          ;それ以外はﾘﾚｰ on

RLY_10:                        ;ON 時刻＞OFF 時刻
     cmp.w     r1,r0           ;現在時刻<>ON 時刻
     bcc       RLY_01          ;現在時刻＞ON 時刻ならﾘﾚｰ on
     cmp.w     r2,r0           ;現在時刻<>OFF 時刻
     bcs       RLY_01          ;現在時刻＜OFF 時刻ならﾘﾚｰ on
                               ;それ以外はﾘﾚｰ off
;-----  ﾘﾚｰ OFF  ---------------
RLY_00:
     bclr      #0,@PDR5
     rts

;-----  ﾘﾚｰ ON  ----------------
RLY_01:
     bset      #0,@PDR5
     rts

;-------------------------------------------------------
;          キー動作
;-------------------------------------------------------
KEYACT:                        ;KEY ACTion
     mov.b     @KEY_FLG,r0l    ;ｷｰ入力 ﾁｪｯｸ
     bne       KEYACT_000      ; 入力があったら KEYACT_000 へ
;-----  ｷｰ入力無し  ------------
     xor.b     r0l,r0l         ;ｷｰｽﾃｰﾀｽ ｸﾘｱ
     mov.b     r0l,@KEY_STA
     rts

;-----  ｷｰ入力有り  ------------
KEYACT_000:
     mov.w     #KEY_BUF,r3
     mov.b     @r3,r0l

     btst      #0,r0l          ;分ｽｲｯﾁ?
     bne       KEYACT_100:16
     btst      #1,r0l          ;時ｽｲｯﾁ?
     bne       KEYACT_100:16

     btst      #2,r0l          ;SET ｽｲｯﾁ?
     bne       KEYACT_200:16

     inc.w     #1,r3
     mov.b     @r3,r0l

     btst      #0,r0l          ;ON ｽｲｯﾁ?
     bne       KEYACT_300:16
     btst      #1,r0l          ;OFF ｽｲｯﾁ?
     bne       KEYACT_400:16

     rts

;-----  分／時スイッチ  ---------------------------------
KEYACT_100:
     mov.b     @MODE,r0l
     btst      #7,r0l
     bne       KEYACT_102      ;時刻設定中なら KEYACT_102 へ
     rts

KEYACT_102:
     mov.b     @KEY_STA,r0l    ;ｷｰｽﾃｰﾀｽ ﾘｰﾄﾞ
     beq       KEYACT_110      ;KEY_STA=0
     cmp.b     #1,r0l
     beq       KEYACT_120      ;KEY_STA=1
     cmp.b     #2,r0l
     beq       KEYACT_130      ;KEY_STA=2
     cmp.b     #3,r0l
     beq       KEYACT_140      ;KEY_STA=3
     bra       KEYACT_110

;-----  KEY_STA=0:初回受付  --------------------
KEYACT_110:
     mov.b     #1,r0l          ;ｽﾃｰﾀｽ=1 ｾｯﾄ
     mov.b     r0l,@KEY_STA

     mov.w     #TMR_KEY,r3     ;ﾀｲﾏｾｯﾄ&ｽﾀｰﾄ
```

<リスト 5-6 watch_3.sec (2/11)>

```
      mov.w     #KEY_WAITCNST1,E0    ;ｽﾛｰﾘﾋﾟｰﾄまでの wait 時間
      mov.b     #H'01,r0l
      jsr       @SETTMR

      bra       KEYACT_160      ;時刻±１へ

;-----  KEY_STA=1:ﾘﾋﾟｰﾄまでの待ち  -------------
KEYACT_120:
      mov.w     #TMR_KEY+1,r3   ;ﾀｲﾏ ｶｳﾝﾄｱｯﾌﾟ ﾁｪｯｸ
      mov.b     @r3,r0l
      bne       KEYACT_122      ;ｶｳﾝﾄｱｯﾌﾟ なら KEYACT_122 へ
      rts

KEYACT_122:
      mov.b     #2,r0l          ;ｽﾃｰﾀｽ=2 ｾｯﾄ
      mov.b     r0l,@KEY_STA
      xor.b     r0l,r0l         ;ﾘﾋﾟｰﾄ回数 ｸﾘｱ
      mov.b     r0l,@KEY_RPTCNT
      bra       KEYACT_134

;-----  KEY_STA=2:ｽﾛｰﾘﾋﾟｰﾄ  --------------------
KEYACT_130:
      mov.w     #TMR_KEY+1,r3   ;ﾀｲﾏ ｶｳﾝﾄｱｯﾌﾟ ﾁｪｯｸ
      mov.b     @r3,r0l
      bne       KEYACT_132      ;ｶｳﾝﾄｱｯﾌﾟ なら KEYACT_132 へ
      rts

KEYACT_132:
      mov.b     @KEY_RPTCNT,r0l ;ﾘﾋﾟｰﾄ回数+1
      inc.b     r0l
      mov.b     r0l,@KEY_RPTCNT
      cmp.b     #KEY_RPTCNST,r0l     ;定数回ﾘﾋﾟｰﾄしたら終了
      bcc       KEYACT_136
KEYACT_134:
      mov.w     #TMR_KEY,r3     ;ﾘﾋﾟｰﾄ間隔(ｽﾛｰ) ﾀｲﾏｾｯﾄ
      mov.w     #KEY_WAITCNST2,E0    ;ｽﾛｰﾘﾋﾟｰﾄの wait 時間
      mov.b     #H'01,r0l
      jsr       @SETTMR
      bra       KEYACT_160      ;時刻±１へ

KEYACT_136:
      mov.b     #3,r0l          ;ｽﾃｰﾀｽ=2 ｾｯﾄ
      mov.b     r0l,@KEY_STA
      bra       KEYACT_142      ;ﾌｧｰｽﾄﾘﾋﾟｰﾄへ

;-----  KEY_STA=3:ﾌｧｰｽﾄﾘﾋﾟｰﾄ  ------------------
KEYACT_140:
      mov.w     #TMR_KEY+1,r3   ;ﾀｲﾏ ｶｳﾝﾄｱｯﾌﾟ ﾁｪｯｸ
      mov.b     @r3,r0l
      bne       KEYACT_142      ;ｶｳﾝﾄｱｯﾌﾟ なら KEYACT_142 へ
      rts

KEYACT_142:
      mov.w     #TMR_KEY,r3     ;ﾘﾋﾟｰﾄ間隔(ﾌｧｰｽﾄ) ﾀｲﾏｾｯﾄ
      mov.w     #KEY_WAITCNST3,e0    ;ﾌｧｰｽﾄﾋﾟｰﾄの wait 時間
      mov.b     #H'01,r0l
      jsr       @SETTMR
      bra       KEYACT_160      ;時刻±１へ

;-----  時刻±１  -----------------------------
KEYACT_160:
      mov.b     @MODE,r0l
      cmp.b     #H'80,r0l       ;80h=時刻設定
      beq       KEYACT_161
      cmp.b     #H'81,r0l       ;81h=ON 時刻設定
      beq       KEYACT_162
      mov.w     #OFFTIM_BUF,r3  ;R3=OFF 時刻 BUF
      bra       KEYACT_163

KEYACT_161:
      mov.w     #TIME_BUF,r3    ;R3=時刻 BUF
      bra       KEYACT_163

KEYACT_162:
      mov.w     #ONTIM_BUF,r3   ;ON 時刻 BUF
KEYACT_163:

      mov.w     #KEY_BUF,r2
      mov.b     @r2,r0l

      btst      #1,r0l
      bne       KEYACT_166
KEYACT_164:
      jsr       @INCMIN         ;分＋１
      bra       KEYACT_168

KEYACT_166:
      jsr       @INCHR          ;時＋１
;     bra       KEYACT_168
;-----  表示更新ﾌﾗｸﾞ&点滅ﾀｲﾏｾｯﾄ  -----     ;他から CALL される
KEYACT_168:
      mov.b     #H'01,r0l       ;表示の更新
      mov.b     r0l,@DISP_FLG   ; ﾌﾗｸﾞ ｾｯﾄ

      mov.w     #TMR_DISP,r3    ;点滅時間ｾｯﾄ
      mov.w     #D'250,e0       ;250d=0.25sec
      mov.b     #H'01,r0l
      jsr       @SETTMR
      rts

;-----  ＳＥＴスイッチ  ----------------------------------
KEYACT_200:
      mov.b     @KEY_STA,r0l    ;ｷｰｽﾃｰﾀｽ ﾁｪｯｸ
      beq       KEYACT_202:16   ;ｽﾃｰﾀｽ=0 初回ｷｰ入力
      cmp.b     #1,r0l
      beq       KEYACT_210:16   ;ｽﾃｰﾀｽ=1 押し続け wait
      cmp.b     #4,r0l
      beq       KEYACT_220:16   ;ｽﾃｰﾀｽ=4 ﾜﾝｼｮｯﾄ動作
      bra       KEYACT_202:16

;-----  初回ｷｰ入力  ------------
KEYACT_202:
      mov.b     @MODE,r0l
      btst      #7,r0l
      bne       KEYACT_204      ;時刻設定中なら KEYACT_204 へ
                                ;通常動作
      mov.b     #1,r0l          ;ｷｰｽﾃｰﾀｽ=1:押し続けのﾁｪｯｸへ
      mov.b     r0l,@KEY_STA

      mov.w     #TMR_KEY,r3     ;押し続け時間ﾀｲﾏ ｾｯﾄ
      mov.w     #D'3000,e0      ;3sec

      mov.b     #H'01,r0l
      jsr       @SETTMR
      rts

KEYACT_204:                     ;時刻設定動作
      mov.b     @MODE,r0l       ;設定ﾓｰﾄﾞ終了
      and.b     #H'7F,r0l
      mov.b     r0l,@MODE
      beq       KEYACT_206      ;現在時刻設定なら KEYACT_206

      mov.b     #H'01,r0l       ;表示ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@DISP_FLG

      mov.w     #TMR_DISP,r3    ;設定終了後表示時間ﾀｲﾏ ｾｯﾄ
      mov.w     #DISP_DISPCNST2,E0
      mov.b     #H'01,r0l
```

＜リスト 5-6 watch_3.sec (3/11)＞

```
      jsr       @SETTMR
      bra       KEYACT_208

KEYACT_206:                       ;時刻ｾｯﾄ
      jsr       @RTCSTOP          ;RTC 動作停止
      mov.b     @TIME_HR,r0l      ;時読み取り
      mov.b     @TIME_WK,r0h      ;週読み取り
      mov.w     r0,e0
      mov.b     @TIME_SEC,r0l     ;秒読み取り
      mov.b     @TIME_MIN,r0h     ;分読み取り
      jsr       @RTCTSET          ;RTC 時刻設定
      jsr       @RTCRUN           ;RTC 動作開始

KEYACT_208:
      mov.b     #4,r0l            ;ﾜﾝｼｮｯﾄ動作
      mov.b     r0l,@KEY_STA
      rts

;-----  押し続け wait  ----------
KEYACT_210:                       ;押し続けﾀｲﾑｱｯﾌﾟ ﾁｪｯｸ
      mov.w     #TMR_KEY+1,r3
      mov.b     @r3,r0l
      bne       KEYACT_214:16     ;ﾀｲﾑｱｯﾌﾟなら時刻設定へ
KEYACT_212:
      rts

KEYACT_214:                       ;時刻設定
      mov.b     @MODE,r0l
      btst      #7,r0l
      bne       KEYACT_212        ;設定ﾓｰﾄﾞ中なら抜ける

      mov.b     #H'80,r0l         ;MODE=80h 時刻設定ﾓｰﾄﾞ
      mov.b     r0l,@MODE

      jsr       @KEYACT_168       ;表示更新ﾌﾗｸﾞ&点滅ﾀｲﾏ ｾｯﾄ

      mov.b     #4,r0l            ;ﾜﾝｼｮｯﾄ動作
      mov.b     r0l,@KEY_STA
      bra       KEYACT_212

;-----  ﾜﾝｼｮｯﾄ動作  ------------
KEYACT_220:                       ;何もしない
      rts

;-----  ＯＮ／ＯＦＦスイッチ  ------------------------------
;-----  ON ｽｲｯﾁ  ----------------------------------------
KEYACT_300:
      mov.b     #H'01,r0h         ;MODE=1 ON 時刻表示
      bra       KEYACT_402

;-----  OFF ｽｲｯﾁ  ---------------------------------------
KEYACT_400:
      mov.b     #H'02,r0h         ;MODE=2 OFF 時刻表示

KEYACT_402:
      mov.b     @MODE,r0l         ;設定ﾓｰﾄﾞ中は無視
      btst      #7,r0l
      beq       KEYACT_404
KEYACT_403:
      rts

KEYACT_404:
      mov.b     @KEY_STA,r0l
      beq       KEYACT_410        ;初回ｷｰ入力
      cmp.b     #1,r0l
      beq       KEYACT_420        ;押し続け wait
      cmp.b     #4,r0l
      beq       KEYACT_430        ;ﾜﾝｼｮｯﾄ動作
      beq       KEYACT_410

;-----  初回ｷｰ入力  ------------
KEYACT_410:
      mov.b     r0h,@MODE         ;動作ﾓｰﾄﾞ ｾｯﾄ
      mov.b     #1,r0l            ;次のｽﾃｰﾀｽへ KEY_STA=1
      mov.b     r0l,@KEY_STA

      mov.b     #H'01,r0l         ;表示ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@DISP_FLG

      mov.w     #TMR_DISP,r3      ;表示時間ﾀｲﾏ ｾｯﾄ
      mov.w     #DISP_DISPCNST1,e0
      mov.b     #H'01,r0l
      jsr       @SETTMR

      mov.w     #TMR_KEY,r3       ;押し続け時間ﾀｲﾏ ｾｯﾄ
      mov.w     #KEY_WAITCNST4,e0
      mov.b     #H'01,r0l
      jsr       @SETTMR
      rts

KEYACT_422:                       ;ON/OFF 時刻設定ﾓｰﾄﾞへ移行
      bset      #7,r0h            ;設定ﾓｰﾄﾞ bit ｾｯﾄ
      mov.b     r0h,@MODE

      mov.b     #4,r0l            ;次のｽﾃｰﾀｽへ KEY_STA=4
      mov.b     r0l,@KEY_STA

      jsr       @KEYACT_168       ;表示更新ﾌﾗｸﾞ&点滅ﾀｲﾏ ｾｯﾄ
      rts

;-----  ﾜﾝｼｮｯﾄ動作  ------------
KEYACT_430:
      rts                       ;何もしない

;------------------------------------------------------------
;          時刻＋１
;------------------------------------------------------------
;R3=時刻 BUF ｱﾄﾞﾚｽ
;-----  秒＋１  ----------------------------------------
INCSEC:
      bra       INCMIN_00

;-----  分＋１  ----------------------------------------
INCMIN:
      inc.w     #1,r3
INCMIN_00:
      mov.b     #H'60,r1l         ;加算上限 59h(BCD)
      bra       TIMEINC

;-----  時＋１  ----------------------------------------
INCHR:
      inc.w     #2,r3
      mov.b     #H'24,r1l         ;加算上限 23h(BCD)
      bra       TIMEINC

;-----  週＋１  ----------------------------------------
INCWK:
      add.w     #3,r3
      mov.b     #7,r1l            ;加算上限 6 (0:日曜 -> 6:土曜)
      bra       TIMEINC

;-----  時刻＋１  ----------------------------
TIMEINC:
      xor.b     r1h,r1h           ;ｵｰﾊﾞｰﾌﾛｰした時の戻り値

      mov.b     @r3,r0l           ;時刻ﾘｰﾄﾞ
      add.b     #1,r0l            ;時刻+1
      daa       r0l               ;BCD 補正
      cmp.b     r1l,r0l           ;ｵｰﾊﾞｰﾌﾛｰ？
```

<リスト 5-6 watch_3.sec (4/11)>

```
     bcs       TIMEINC_00      ; CC=ｵｰﾊﾞｰﾌﾛｰ
     mov.b     r1h,r0l         ; 00 に戻す
TIMEINC_00:
     mov.b     r0l,@r3         ;時刻ｾｯﾄ

     rts

;---------------------------------------------------------
;          キー入力２（スキャン）
;---------------------------------------------------------
KEYIN2:
     mov.b     @KEYIN_STA,r0l  ;ｽﾃｰﾀｽﾁｪｯｸ
     beq       KEYIN2_00       ;0=1st ﾘｰﾄﾞ
     cmp.b     #1,r0l
     beq       KEYIN2_10       ;1=wait
     cmp.b     #2,r0l
     beq       KEYIN2_20       ;2=2nd ﾘｰﾄﾞ

;-----  1st ﾘｰﾄﾞ  ---------------
KEYIN2_00:
     mov.b     @SCAN_DT,r0l
     btst      #0,r0l
     bne       KEYIN2_02       ;ｽｷｬﾝ bit=0 以外なら KEYIN2_02
     mov.w     #KEY1st_BUF,r3  ;ｽｷｬﾝ bit=0 ｱﾄﾞﾚｽ･ﾌﾗｸﾞｲﾆｼｬﾗｲｽﾞ
     mov.w     r3,@KEYDT_ADR
     xor.b     r0l,r0l
     mov.b     r0l,@KEYIN_FLG
KEYIN2_02:
     jsr       @KEYREAD        ;ﾎﾟｰﾄ ﾘｰﾄﾞ
     jsr       @KEYSETBUF      ;ﾎﾟｰﾄの状態を BUF へｾｯﾄ
     and.b     r0l,r0l
     beq       KEYIN2_04
     mov.b     #H'01,r0l       ;ｷｰ入力があったらﾌﾗｸﾞｾｯﾄ
     mov.b     r0l,@KEYIN_FLG
KEYIN2_04:
     mov.b     @SCAN_DT,r0l
     btst      #3,r0l
     beq       KEYIN2_06       ;全ｽｷｬﾝ終了したら KEYIN2_06 へ
     rts

KEYIN2_06:
     mov.b     @KEYIN_FLG,r0l
     beq       KEYIN2_40:16    ;ｷｰの入力が無かったら終了

     mov.b     #H'01,r0l       ;次のｽﾃｰﾀｽへ : ｽﾃｰﾀｽ=1
     mov.b     r0l,@KEYIN_STA

     mov.w     #TMR_KEYIN,r3   ;wait ﾀｲﾏｾｯﾄ
     mov.w     #D'50,e0
     mov.b     #H'01,r0l
     jsr       @SETTMR
     rts

;-----  wait  ------------------
KEYIN2_10:
     mov.w     #TMR_KEYIN+1,r3 ;ﾀｲﾑｱｯﾌﾟ ﾁｪｯｸ
     mov.b     @r3,r0l
     bne       KEYIN2_14       ;ﾀｲﾑｱｯﾌﾟしていれば KEYIN2_14
KEYIN2_12:
     rts

KEYIN2_14:
     mov.b     @SCAN_DT,r0l
     btst      #0,r0l
     bne       KEYIN2_12       ;ｽｷｬﾝ bit=0 でなければ抜ける

     mov.b     #2,r0l          ;次のｽﾃｰﾀｽへ : KEYIN_STA=2
     mov.b     r0l,@KEYIN_STA

     mov.w     #KEY2nd_BUF,r3  ;ｱﾄﾞﾚｽ･ﾌﾗｸﾞ ｲﾆｼｬﾗｲｽﾞ
     mov.w     r3,@KEYDT_ADR
;-----  2nd ﾘｰﾄﾞ  --------------
KEYIN2_20:
     jsr       @KEYREAD        ;ﾎﾟｰﾄ ﾘｰﾄﾞ
     jsr       @KEYSETBUF      ;ﾎﾟｰﾄの状態を BUF へｾｯﾄ
     and.b     r0l,r0l
     beq       KEYIN2_22       ;ｷｰ入力が無ければ KEYIN2_22 へ
     mov.b     #H'01,r0l       ;ｷｰ入力あり ﾌﾗｸﾞｾｯﾄ
     mov.b     r0l,@KEYIN_FLG
KEYIN2_22:
     mov.b     @SCAN_DT,r0l
     btst      #3,r0l
     beq       KEYIN2_24       ;全ｽｷｬﾝ終了したら KEYIN2_24 へ
     rts

KEYIN2_24:
     mov.b     @KEYIN_FLG,r0l;ﾌﾗｸﾞが立ってなければ入力無し
     beq       KEYIN2_40:16

     mov.w     #KEY1st_BUF,r2  ;1st ﾘｰﾄﾞと 2nd ﾘｰﾄﾞ 比較
     mov.w     #KEY2nd_BUF,r3
     mov.b     #4,r1l
KEYIN2_26:
     mov.b     @R2,r0l
     mov.b     @r3,r0h
     cmp.b     r0h,r0l
     bne       KEYIN2_40       ;不一致なら終了
     inc.w     #1,r2
     inc.w     #1,r3
     dec.b     r1l
     bne       KEYIN2_26

     mov.b     #4,r1l          ;入力確定
     mov.w     #KEY2nd_BUF,r2  ; ｷｰﾃﾞｰﾀを KEY_BUF へｾｯﾄ
     mov.w     #KEY_BUF,r3
KEYIN2_28:
     mov.b     @r2,r0l
     mov.b     r0l,@r3
     inc.w     #1,r2
     inc.w     #1,r3
     dec.b     r1l
     bne       KEYIN2_28

     xor.b     r0l,r0l         ;ｽﾃｰﾀｽ ｸﾘｱ
     mov.b     r0l,@KEYIN_STA
     mov.b     #01,r0l         ;ﾌﾗｸﾞ ｾｯﾄ
     mov.b     r0l,@KEY_FLG
     rts

;-----  入力無し､又は不一致  ---
KEYIN2_40:
     xor.b     r0l,r0l         ;ｽﾃｰﾀｽ･ﾌﾗｸﾞ ｸﾘｱ
     mov.b     r0l,@KEYIN_STA
     mov.b     r0l,@KEY_FLG
     rts

;---------------------------------------------------------
;          キーの状態をポートから読む
;---------------------------------------------------------
KEYREAD:
     mov.b     @PDR1,r0l
     not.b     r0l
     and.b     #H'07,r0l
     rts

;---------------------------------------------------------
;          キーの状態をバッファへセット
;---------------------------------------------------------
```

<リスト 5-6 watch_3.sec (5/11)>

```
KEYSETBUF:
      mov.w     @KEYDT_ADR,r3
      mov.b     r0l,@r3
      inc.w     #1,r3
      mov.w     r3,@KEYDT_ADR
      rts

;-------------------------------------------------------
;          表示セット
;-------------------------------------------------------
SETDISP:
      mov.b     @MODE,r0l       ;各ﾓｰﾄﾞ毎に表示をｾｯﾄ
      cmp.b     #H'00,r0l       ;00=時刻表示
      beq       SETDISP_000:16
      cmp.b     #H'01,r0l       ;01=ON 時刻表示
      beq       SETDISP_100:16
      cmp.b     #H'02,r0l       ;02=OFF 時刻表示
      beq       SETDISP_200:16
      cmp.b     #H'80,r0l       ;80=時刻設定ﾓｰﾄﾞ
      beq       SETDISP_800:16
      cmp.b     #H'81,r0l       ;81=ON 時刻設定ﾓｰﾄﾞ
      beq       SETDISP_900:16
      cmp.b     #H'82,r0l       ;82=OFF 時刻設定ﾓｰﾄﾞ
      beq       SETDISP_A00:16
      rts

;-----  時刻表示  ----------------------------------------
SETDISP_000:
      mov.b     @RTC_FLG,r0l
      bne       SETDISP_006     ;ﾌﾗｸﾞが立っていたら表示更新
SETDISP_002:
      mov.w     #TMR_COLON+1,r3 ;":"点灯時間ﾁｪｯｸ
      mov.b     @r3,r0l
      bne       SETDISP_004     ;ﾀｲﾑｱｯﾌﾟなら":"消灯
      rts

SETDISP_004:
      xor.b     r0l,r0l
      mov.b     r0l,@r3         ;ﾀｲﾑｱｯﾌﾟﾌﾗｸﾞ ｸﾘｱ

      bclr      #4,@PDR5        ;":"消灯
      bclr      #5,@PDR5
      rts

SETDISP_006:
      mov.b     @RSECDR,r0l     ;RTC BUSY ﾁｪｯｸ
      btst      #7,r0l
      bne       SETDISP_002     ;BUSY なら SETDISP_002 へ

      xor.b     r0l,r0l         ;ﾌﾗｸﾞ ｸﾘｱ
      mov.b     r0l,@RTC_FLG

      jsr       @RTCREAD        ;RTC 時刻 読み取り
      mov.w     #TIME_BUF,r3    ;時刻ｾｸﾞﾒﾝﾄ変換
      jsr       @CONVSEG

      bset      #4,@PDR5        ;":"点灯
      bset      #5,@PDR5

      mov.w     #TMR_COLON,r3   ;":"点灯時間ﾀｲﾏ ｾｯﾄ
      mov.w     #DISP_CLNCNST,e0
      mov.b     #H'01,r0l
      jsr       @SETTMR
      rts

;-----  ON 時刻表示  ----------------------------------------
SETDISP_100:
      mov.b     #H'10,r0h
      mov.w     #ONTIM_BUF,r3
      bra       SETDISP_210

;-----  OFF 時刻表示  ----------------------------------------
SETDISP_200:
      mov.b     #H'20,r0h
      mov.w     #OFFTIM_BUF,r3

SETDISP_210:
      mov.b     @DISP_FLG,r0l   ;DISP_FLG=1 : 表示ﾃﾞｰﾀｾｯﾄ
      bne       SETDISP_212

      mov.w     #TMR_DISP+1,r3  ;表示時間経過ﾁｪｯｸ
      mov.b     @r3,r0l
      bne       SETDISP_214
      rts

SETDISP_212:
      xor.b     r0l,r0l         ;ﾌﾗｸﾞ ｸﾘｱ
      mov.b     r0l,@DISP_FLG

      orc.b     #H'80,CCR       ;割り込み禁止
      mov.b     @PDR5,r0l
      and.b     #H'CF,r0l
      or.b      r0h,r0l
      mov.b     r0l,@PDR5
      andc      #H'7F,CCR       ;割り込み許可
      jsr       @CONVSEG        ;ON/OFF 時刻ｾｸﾞﾒﾝﾄ変換
      rts

SETDISP_214:
      xor.b     r0l,r0l         ;表示時間経過 MODE=00 時刻表示
      mov.b     r0l,@MODE
      rts

;-----  時刻設定モード  ----------------------------------
SETDISP_800:
      mov.b     #H'30,r0h
      mov.w     #TIME_BUF,r4
      bra       SETDISP_A10

;-----  ON 時刻設定モード  ----------------------------------
SETDISP_900:
      mov.b     #H'10,r0h
      mov.w     #ONTIM_BUF,r4
      bra       SETDISP_A10

;-----  OFF 時刻設定モード  ----------------------------------
SETDISP_A00:
      mov.b     #H'20,r0h
      mov.w     #OFFTIM_BUF,r4
      bra       SETDISP_A10

SETDISP_A10:
      orc.b     #H'80,CCR       ;割り込み禁止
      mov.b     @PDR5,r0l       ;":"表示 ｾｯﾄ
      and.b     #H'CF,r0l
      or.b      r0h,r0l
      mov.b     r0l,@PDR5
      andc      #H'7F,CCR       ;割り込み許可

      mov.b     @DISP_FLG,r0l
      bne       SETDISP_A20     ;変更があったら強制的に表示

      mov.w     #TMR_DISP+1,r3  ;表示点滅ﾀｲﾏ ﾀｲﾑｱｯﾌﾟ確認
      mov.b     @r3,r0l
      bne       SETDISP_A12     ;ﾀｲﾑｱｯﾌﾟなら表示の変更
      rts
```

<リスト 5-6 watch_3.sec 6/11)>

```
SETDISP_A12:
     mov.b     @BLINK_FLG,r0l  ;今迄の点灯状態ﾁｪｯｸ
     bne       SETDISP_A30     ; EQ=消灯 / NE=点灯

SETDISP_A20:                   ;消灯→点灯
     mov.b     #1,r0l          ;ﾌﾗｸﾞ=1 点灯
     mov.b     r0l,@BLINK_FLG
     mov.w     r4,r3           ;表示ﾃﾞｰﾀｾｯﾄ
     jsr       @CONVSEG
     bra       SETDISP_A40

SETDISP_A30:                   ;点灯→消灯
     xor.b     r0l,r0l         ;ﾌﾗｸﾞ=0 消灯
     mov.b     r0l,@BLINK_FLG
     xor.l     er0,er0         ;設定箇所消灯
     mov.w     #SEG_BUF,r3
     inc.w     #2,r3
     mov.w     e0,@r3
     inc.w     #2,r3
     mov.w     e0,@r3

SETDISP_A40:
     xor.b     r0l,r0l         ;表示更新ﾌﾗｸﾞ ｸﾘｱ
     mov.b     r0l,@DISP_FLG

     mov.w     #TMR_DISP,r3    ;点滅用ﾀｲﾏ ｾｯﾄ
     mov.w     #D'250,e0       ;250d=0.25sec
     mov.b     #H'01,r0l
     jsr       @SETTMR
     rts

;---------------------------------------------------------
;          表示
;---------------------------------------------------------
DISP:
     jsr       @SCANOFF        ;ｽｷｬﾝ off 消灯

     mov.b     @SCAN_DT,r0l    ;ｽｷｬﾝﾋﾞｯﾄ ｼﾌﾄ
     rotl.b    r0l
     bcc       DISP_00
                               ;未ｽｷｬﾝ有り
     mov.w     @SEGDT_ADR,r3   ;R3=表示ﾃﾞｰﾀｱﾄﾞﾚｽ
     inc.w     #1,r3
     bra       DISP_02

DISP_00:                       ;全ｽｷｬﾝ終了
     mov.w     #SEG_BUF+2,r3   ;表示ﾃﾞｰﾀｱﾄﾞﾚｽ 初期値
DISP_02:
     mov.b     r0l,@SCAN_DT
     mov.w     r3,@SEGDT_ADR

     mov.b     @r3,r0l         ;表示ﾃﾞｰﾀ 出力
     mov.b     r0l,@PDR7
     mov.b     @PDR2,r0h
     and.b     #B'01110111,r0h
     btst      #3,r0l
     beq       DISP_04
     bset      #3,r0h
DISP_04:
     btst      #7,r0l
     beq       DISP_06
     bset      #7,r0h
DISP_06:
     mov.b     r0h,@PDR2

     jsr       @SCANON         ;ｽｷｬﾝ on 表示

     rts

;---------------------------------------------------------
;          スキャンＯＮ
;---------------------------------------------------------
SCANON:
     mov.b     @PDR1,r0l       ;ｽｷｬﾝﾎﾟｰﾄ ﾘｰﾄﾞ
     and.b     #H'0F,r0l       ;ｽｷｬﾝ使用ﾋﾞｯﾄ ｸﾘｱ
     mov.b     @SCAN_DT,r0h    ;ｽｷｬﾝﾃﾞｰﾀ ﾘｰﾄﾞ
     and.b     #H'F0,r0h
     or.b      r0h,r0l         ;ｽｷｬﾝﾎﾟｰﾄ+ｽｷｬﾝﾃﾞｰﾀ
     mov.b     r0l,@PDR1       ;ｽｷｬﾝ出力 (on)
     rts

;*********************************************************
;          P1 セクションの指定
;   プログラムサイズが H'E800～H'EFFF で収まらない為、ここでセ
; クションを再設定しこれ以降のプログラムをアドレス H'F780 番
; 地以降に割り付けます。
;*********************************************************
          .section   P1,code

;---------------------------------------------------------
;          スキャンＯＦＦ
;---------------------------------------------------------
SCANOFF:
     mov.b     @PDR1,r0l       ;ｽｷｬﾝﾎﾟｰﾄ ﾘｰﾄﾞ
     or.b      #H'F0,r0l       ;ｽｷｬﾝ ｸﾘｱ
     mov.b     r0l,@PDR1       ;ｽｷｬﾝ出力 (off)
     rts

;---------------------------------------------------------
;          タイマ管理
;---------------------------------------------------------
TMRCNT:
     mov.w     #TMR_BUF,r3     ;R3=BUF ｱﾄﾞﾚｽ
     mov.b     #TMR_USECNST,r1l      ;R1L=ﾀｲﾏの数
TMRCNT_00:
     push.w    r3
     mov.b     @r3,r0l         ;ｽﾃｰﾀｽ ﾁｪｯｸ
     bne       TMRCNT_10       ; ｽﾃｰﾀｽがｾｯﾄされていたらｶｳﾝﾄ
TMRCNT_02:
     pop.w     r3
     dec.b     r1l
     beq       TMRCNT_04
     add.w     #D'8,r3         ;次のｶｳﾝﾀへ
     bra       TMRCNT_00

TMRCNT_04:
     rts

;-----  ｶｳﾝﾄ  ----------------------------------------------
TMRCNT_10:
     add.w     #D'4,r3         ;R3=ｶｳﾝﾀ
     mov.w     @r3,r0
     dec.w     #1,r0           ;ｶｳﾝﾄ
     mov.w     r0,@r3
     beq       TMRCNT_20       ;ｶｳﾝﾄｱｯﾌﾟ したら TMRCNT_20 へ
     bra       TMRCNT_02

;-----  ｶｳﾝﾄｱｯﾌﾟ  ------------------------------------------
TMRCNT_20:
     push.w    r3
     dec.w     #2,r3           ;R3=ｶｳﾝﾄ初期値格納ｱﾄﾞﾚｽ
     mov.w     @r3,r0          ;ｶｳﾝﾄ初期値読み取り
     dec.w     #1,r3           ;R3=ﾌﾗｸﾞ格納ｱﾄﾞﾚｽ
     mov.b     #H'01,r0l       ;ｶｳﾝﾄｱｯﾌﾟﾌﾗｸﾞ ｾｯﾄ
     mov.b     r0l,@r3
     dec.w     #1,r3           ;R3=ｽﾃｰﾀｽ格納ｱﾄﾞﾚｽ
     mov.b     @r3,r0l         ;ROL=ｽﾃｰﾀｽ
     pop.w     r3
```

<リスト 5-6 watch_3.sec (7/11)>

```
     mov.w     e0,@r3          ;ｶｳﾝﾄ初期化
     cmp.b     #H'01,r0l       ;ｶｳﾝﾄﾘﾋﾟｰﾄ or ｽﾄｯﾌﾟ
     beq       TMRCNT_22       ;ｽﾃｰﾀｽ=01h ならｶｳﾝﾀｽﾄｯﾌﾟ
     bra       TMRCNT_02

TMRCNT_22:                     ;STA=H'01 = ｶｳﾝﾄｽﾄｯﾌﾟ
     sub.w     #D'4,r3         ;R3=ｽﾃｰﾀｽ格納位置
     xor.b     r0l,r0l         ;ｽﾃｰﾀｽ ｸﾘｱ
     mov.b     r0l,@r3
     bra       TMRCNT_02

;-------------------------------------------------------
;          タイマセット＆スタート
;-------------------------------------------------------
;R3  = ﾀｲﾏ ｱﾄﾞﾚｽ
;E0  = ｶｳﾝﾄ値
;R0L = ﾀｲﾏ ｽﾃｰﾀｽ [1:ｶｳﾝﾄ / 2:ｶｳﾝﾄ(ｴﾝﾄﾞﾚｽ)]
SETTMR:
     push.w    r3
     inc.w     #1,r3           ;ﾌﾗｸﾞ ｸﾘｱ
     xor.b     r0h,r0h
     mov.b     r0h,@r3
     inc.w     #1,r3           ;ｶｳﾝﾄ初期値ｾｯﾄ
     mov.w     e0,@r3
     inc.w     #2,r3           ;ｶｳﾝﾀｾｯﾄ
     mov.w     e0,@r3
     pop.w     r3              ;ｽﾃｰﾀｽｾｯﾄ
     mov.b     r0l,@r3         ;ｶｳﾝﾄ開始
     rts

;-------------------------------------------------------
;          セグメント変換＆セット
;-------------------------------------------------------
;R3=TIME_BUF
CONVSEG:
     mov.w     #SEG_BUF,r4
     xor.l     er0,er0
     mov.b     #3,r1l
CONVSEG_00:
     mov.b     @r3,r0l
     and.b     #H'0F,r0l
     mov.b     @(SEG_TBL,er0),r0l
     mov.b     r0l,@r4
     inc.w     #1,r4
     mov.b     @r3,r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     shlr.b    r0l
     mov.b     @(SEG_TBL,er0),r0l
     mov.b     r0l,@r4
     inc.w     #1,r4

     inc.w     #1,r3
     dec.b     r1l
     bne       CONVSEG_00
     rts

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

```
;-------------------------------------------------------
;          ＲＴＣから時刻読み取り
;-------------------------------------------------------
RTCREAD:
     mov.b     @RSECDR,r0l     ;秒
     mov.b     r0l,@TIME_SEC
     mov.b     @RMINDR,r0l     ;分
     mov.b     r0l,@TIME_MIN
     mov.b     @RHRDR,r0l      ;時
     mov.b     r0l,@TIME_HR
     mov.b     @RWKDR,r0l      ;週
     mov.b     r0l,@TIME_WK
     rts

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

;-------------------------------------------------------
;          ＰＩＯイニシャライズ
;-------------------------------------------------------
INITPIO:
     mov.b     #B'11110000,r0l ;P14-17 SCAN bit
     mov.b     r0l,@PCR1
     mov.b     #B'00011000,r0l ;P23,24 SegData
     mov.b     r0l,@PCR2
     mov.b     #B'11111111,r0l ;P5 RLY & other output
     mov.b     r0l,@PCR5
     mov.b     #B'11111111,r0l ;P7 SegData
     mov.b     r0l,@PCR7

     mov.b     #B'00000111,r0l ;PB Setting
     mov.b     r0l,@ADCSR

     rts

;-------------------------------------------------------
;          タイマＢ１イニシャライズ
;-------------------------------------------------------
INITTB1:
     mov.b     #B'11111010,r0l ;ｵｰﾄﾘﾛｰﾄﾞ
     mov.b     r0l,@TMB1
     mov.b     #D'256-39,r0l   ;998.4usec
     mov.b     r0l,@TCB1
     rts

;-------------------------------------------------------
;          タイマＺイニシャライズ
;-------------------------------------------------------
;※FTIOA0(P60)からは約 32.787kHz の出力の出力が出ています。
;サブクロックを搭載せずに RTC を使用する場合は、FTIOA0(P60)を
;X1 に接続して下さい。
INITTZ:
     mov.b     #B'00100000,r0l ;ｸﾘｱ要因:GRA / φｶｳﾝﾄ
     mov.b     r0l,@TCR_0
     mov.b     #B'10001011,r0l ;GRA:ﾄｸﾞﾙ出力
     mov.b     r0l,@TIORA_0
     mov.w     #D'305,r0             ;32.787kHz 出力
     mov.w     r0,@GRA_0
     mov.b     #B'11111101,r0l ;ch0:ｽﾀｰﾄ
```

<リスト 5-6 watch_3.sec (8/11)>

```
     mov.b     r0l,@TSTR
     mov.b     #B'11111110,r0l ;FTIOA0 出力
     mov.b     r0l,@TOER
     rts

;---------------------------------------------------------
;          ＲＴＣイニシャライズ
;---------------------------------------------------------
INITRTC:
     mov.b     @RSECDR,r0l     ;Busy?
     btst      #7,r0l
     bne       INITRTC

     mov.b     @RTCCR1,r0l
     bclr      #7,r0l          ;stop
     mov.b     r0l,@RTCCR1
     bset      #4,r0l          ;reset on
     mov.b     r0l,@RTCCR1
     bclr      #4,r0l          ;reset off
     mov.b     r0l,@RTCCR1

     xor.b     r0l,r0l         ;時間ﾚｼﾞｽﾀ ｸﾘｱ
     mov.b     r0l,@RSECDR
     mov.b     r0l,@RMINDR
     mov.b     r0l,@RHRDR
     mov.b     r0l,@RWKDR

     mov.b     #B'00001000,r0l ;RTC 動作
     mov.b     r0l,@RTCCSR
     mov.b     #B'00000001,r0l ;秒割り込み
     mov.b     r0l,@RTCCR2
     mov.b     #B'11000000,r0l ;24h ﾓｰﾄﾞ Run
     mov.b     r0l,@RTCCR1

     rts

;---------------------------------------------------------
;          ＲＴＣ動作停止
;---------------------------------------------------------
RTCSTOP:
     mov.b     @RSECDR,r0l     ;Busy?
     btst      #7,r0l
     bne       RTC_STOP
     mov.b     @RTCCR1,r0l
     bclr      #7,r0l          ;bit7:RUN=0 / stop
     mov.b     r0l,@RTCCR1
     rts

;---------------------------------------------------------
;          ＲＴＣリセット
;---------------------------------------------------------
RTCRESET:
     mov.b     @RTCCR1,r0l
     bset      #4,r0l          ;bit4:RST=1 / reset on
     mov.b     r0l,@RTCCR1
     bclr      #4,r0l          ;bit4:RST=0 / reset off
     mov.b     r0l,@RTCCR1
     rts

;---------------------------------------------------------
;          ＲＴＣ時間セット
;---------------------------------------------------------
;R0L   = 秒
;R0H   = 分
;E0(L) = 時
;E0(H) = 週
RTCTSET:
     mov.b     r0l,@RSECDR     ;秒ｾｯﾄ
     mov.b     r0h,@RMINDR     ;分ｾｯﾄ
     mov.w     e0,r0
     mov.b     r0l,@RHRDR      ;時ｾｯﾄ
     mov.b     r0h,@RWKDR      ;週ｾｯﾄ
     rts

;---------------------------------------------------------
;          ＲＴＣ動作開始
;---------------------------------------------------------
RTCRUN:
     mov.b     @RTCCR1,r0l
     bset      #7,r0l          ;bit7:RUN=1 / run
     mov.b     r0l,@RTCCR1
     rts

;---------------------------------------------------------
;          割り込みイニシャライズ
;---------------------------------------------------------
INIT_INTR:
     mov.b     #B'01010000,r0l ;RTC 割り込みｲﾈｰﾌﾞﾙ
     mov.b     r0l,@IENR1
     mov.b     #B'00111111,r0l ;ﾀｲﾏ B1 割り込みｲﾈｰﾌﾞﾙ
     mov.b     r0l,@IENR2
     rts

;=========================================================
;          ワークエリア
;=========================================================
     .section  D,data

WORK_AREA:
;-----  KEY  ---------------------------------------------
KEY_FLG         .res.b    1    ;ｷｰ入力ﾌﾗｸﾞ
KEY_STA         .res.b    1    ;ｷｰｽﾃｰﾀｽ 0:1st/3:ﾌｧｰｽﾄﾘﾋﾟｰﾄ
                               ;         1:wait/4:ﾜﾝｼｮｯﾄ
                               ;         2:ｽﾛｰﾘﾋﾟｰﾄ
KEY_DT          .res.b    1    ;ｷｰﾃﾞｰﾀ
KEY_RPTCNT      .res.b    1    ;ｽﾛｰﾘﾋﾟｰﾄ繰り返し回数

KEYIN_STA       .res.b    1    ;KEYIN ｽﾃｰﾀｽ
                               ; 0:1st ﾘｰﾄﾞ / 1:2nd ﾘｰﾄﾞ
KEY1st_DT       .res.b    1    ;1st ﾘｰﾄﾞ ﾃﾞｰﾀ

KEYIN_FLG       .res.b    1    ;ｷｰ入力有無ﾌﾗｸﾞ

KEY1st_BUF      .res.b    4    ;1st ﾘｰﾄﾞ BUF
KEY2nd_BUF      .res.b    4    ;2nd ﾘｰﾄﾞ BUF
KEY_BUF         .res.b    4    ;確定ｷｰ BUF
     .align     2
KEYDT_ADR       .res.w    1    ;ｷｰ BUF ｱﾄﾞﾚｽ 格納先

;-----  時刻  --------------------------------------------
RTC_FLG         .res.b    1    ;RTC 割り込みﾌﾗｸﾞ
CP_FLG          .res.b    1    ;ON/OFF 時刻判定指示ﾌﾗｸﾞ
TIME_BUF:
TIME_SEC        .res.b    1    ;現時時刻 秒
TIME_MIN        .res.b    1    ;現在時刻 分
TIME_HR         .res.b    1    ;現在時刻 時
TIME_WK         .res.b    1    ;現在時刻 週
ONTIM_BUF:
ONTIM_SEC       .res.b    1    ;ON 時刻 秒
ONTIM_MIN       .res.b    1    ;ON 時刻 分
ONTIM_HR        .res.b    1    ;ON 時刻 時
ONTIM_WK        .res.b    1    ;ON 時刻 週
OFFTIM_BUF:
OFFTIM_SEC      .res.b    1    ;OFF 時刻 秒
OFFTIM_MIN      .res.b    1    ;OFF 時刻 分
OFFTIM_HR       .res.b    1    ;OFF 時刻 時
OFFTIM_WK       .res.b    1    ;OFF 時刻 週
     .align     2
```

＜リスト 5-6 watch_3.sec (9/11)＞

```
;-----  設定  ----------------------------------------------
MODE            .res.b    1     ;動作状態
                                ; 00h=時刻表示
                                ; 01h=ON 時刻表示
                                ; 02h=OFF 時刻表示
                                ; bit7=1=設定モード
                                ; 80h=時刻設定モード
                                ; 81h=ON 時刻設定モード
                                ; 82h=OFF 時刻設定モード
SET_PNT         .res.b    1     ;設定箇所ﾎﾟｲﾝﾀ
                                ; 0:秒 / 1:分 / 2:時
     .align     2
;-----  設定  ----------------------------------------------
MODE            .res.b    1     ;動作状態
                                ; 00h=時刻表示
                                ; 01h=ON 時刻表示
                                ; 02h=OFF 時刻表示
                                ; bit7=1=設定モード
                                ; 80h=時刻設定モード
                                ; 81h=ON 時刻設定モード
                                ; 82h=OFF 時刻設定モード
SET_PNT         .res.b    1     ;設定箇所ﾎﾟｲﾝﾀ
                                ; 0:秒 / 1:分 / 2:時
     .align     2

;-----  表示  ----------------------------------------------
SEG_BUF         .res.b    6     ;表示ﾃﾞｰﾀﾊﾞｯﾌｧ
                                ; (ｾｸﾞﾒﾝﾄﾃﾞｰﾀ) 秒分時
SEGDT_ADR       .res.w    1     ;ｾｸﾞﾒﾝﾄﾊﾞｯﾌｧ ｱﾄﾞﾚｽ
SCAN_DT         .res.b    1     ;ｽｷｬﾝﾃﾞｰﾀ
DISP_FLG        .res.b    1     ;
BLINK_FLG       .res.b    1     ;点滅ﾌﾗｸﾞ
                                ; 0:消灯 / 1:点灯
     .align     2

;-----  ﾀｲﾏ  ----------------------------------------------
;■構造 (ﾃﾞｸﾘﾒﾝﾄ ｶｳﾝﾀ)
;ｽﾃｰﾀｽ         size=1    0:ｽﾄｯﾌﾟ / 1:ｶｳﾝﾄ / 2:ｶｳﾝﾄ(ｴﾝﾄﾞﾚｽ)
;ﾌﾗｸﾞ          size=1    1:ｶｳﾝﾄｱｯﾌﾟ
;ｶｳﾝﾄ初期値    size=2    ﾜｰﾄﾞﾃﾞｰﾀ
;ｶｳﾝﾀ          size=2    ﾜｰﾄﾞﾃﾞｰﾀ
;予備          size=2    -
TMR_BUF:
TMR_KEY:
                .res.b    2     ;ｽﾃｰﾀｽ / ﾌﾗｸﾞ
                .res.w    3     ;ｶｳﾝﾄ初期値 / ｶｳﾝﾀ / 予備
TMR_DISP:
                .res.b    2     ;ｽﾃｰﾀｽ / ﾌﾗｸﾞ
                .res.w    3     ;ｶｳﾝﾄ初期値 / ｶｳﾝﾀ / 予備
TMR_KEYIN:
                .res.b    2     ;ｽﾃｰﾀｽ / ﾌﾗｸﾞ
                .res.w    3     ;ｶｳﾝﾄ初期値 / ｶｳﾝﾀ / 予備
TMR_COLON:
                .res.b    2     ;ｽﾃｰﾀｽ / ﾌﾗｸﾞ
                .res.w    3     ;ｶｳﾝﾄ初期値 / ｶｳﾝﾀ / 予備
     .align     2

WORK_AREAE:

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

<リスト 5-6 watch_3.sec (10/11)>

```
;-------------------------------------------------------------
;
;   FILE        :intprg.src
;   DATE        :Wed, Aug 25, 2004
;   DESCRIPTION :Interrupt Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;
;-------------------------------------------------------------


    .include    "vect.inc"

     .import    RTCINT
     .import    TB1INT

    .section    IntPRG, code
;1 Reserved
_INT_Reserved1
                    ≀


;18 WKP
_INT_WKP
;19 RTC
_INT_RTC
          bra       RTCINT:16

;20 Reserved
_INT_Reserved20
                    ≀


;28 Reserved
_INT_Reserved28
;29 Timer B1
_INT_TimerB1
          bra       TB1INT:16

;30 Reserved
_INT_Reserved30
;31 Reserved
_INT_Reserved31
;32 SCI3_2
_INT_SCI3_2
    sleep
    nop
    .end
```

<リスト 5-6 watch_3.sec (11/11)>

★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 2 月作成）

20050208