13. ソフトウェアによるタイマ（プロジェクト：timer1）

13.1 概要

本章では、時間稼ぎをする関数(timer 関数)を使い、LED の光り方を 1 秒ごとに替える方法を説明します。LED をたくさん制御すれば、電飾などに応用可能です。ただし、時間の測り方は簡易的な方法を使っているので正確ではありません。正確なタイマが必要な場合は、プロジェクト「timer2」を参照してください。

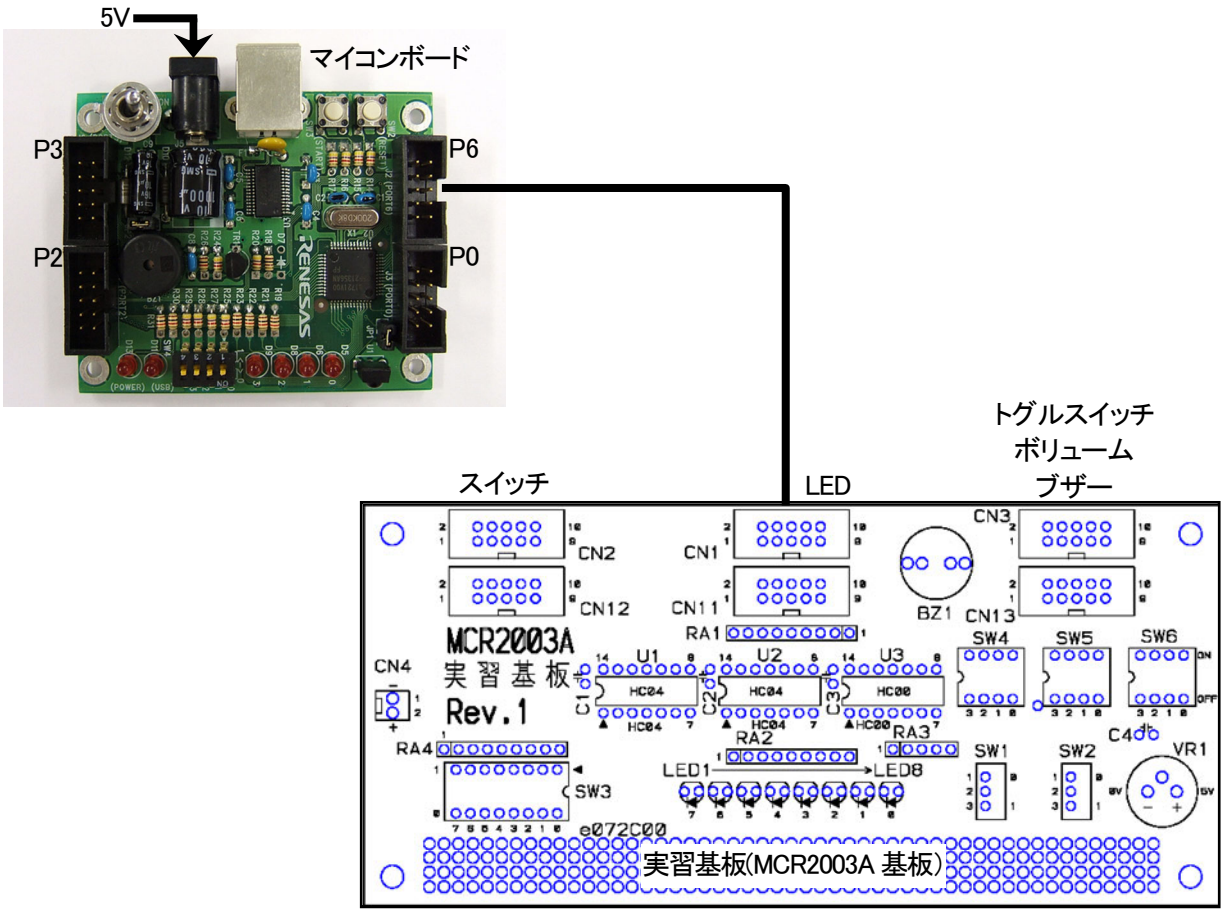
13.2 接続

■使用ポート

マイコンのポート	接続内容
P6 (J2)	実習基板の LED 部など、出力機器を接続します。

■接続例

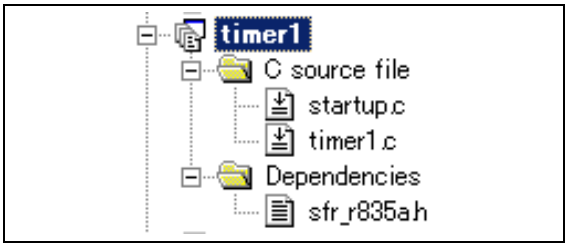
実習基板を使ったときの接続例を次に示します。



■操作方法

操作は特にありません。電源を入れるとLEDが点滅します。LEDの点滅の仕方をよく観察してください。

13.3 プロジェクトの構成



	ファイル名	内容
1	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。このファイルは共通で、どのプロジェクトもこのファイルから実行されます。
2	timer1.c	実際に制御するプログラムが書かれています。R8C/35A の内蔵周辺機能(SFR)の初期化も行います。
3	sfr_r835a.h	R8C/35A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Registers)を定義したファイルです。

13.4 プログラム「timer1.c」

```
 1 :  /***********************************************************************/
 2 :  /* 対象マイコン R8C/35A                                                */
 3 :  /* ﾌｧｲﾙ内容     ソフトウェアタイマ                                     */
 4 :  /* バージョン   Ver.1.20                                               */
 5 :  /* Date         2010.04.19                                             */
 6 :  /* Copyright    ルネサスマイコンカーラリー事務局                       */
 7 :  /*              日立インターメディックス株式会社                       */
 8 :  /***********************************************************************/
 9 :  /*
10 :  出力：P6_7-P6_0(LEDなど)
11 :
12 :  ポート6に繋いだLEDを1秒間隔で点滅させます。
13 :  タイマはループによるソフトウェアタイマを使用します。
14 :  */
15 :
16 :  /*=====================================*/
17 :  /* インクルード                        */
18 :  /*=====================================*/
19 :  #include "sfr_r835a.h"                 /* R8C/35A SFRの定義ファイル     */
20 :
21 :  /*=====================================*/
22 :  /* シンボル定義                        */
23 :  /*=====================================*/
24 :
25 :  /*=====================================*/
26 :  /* プロトタイプ宣言                    */
27 :  /*=====================================*/
28 :  void init( void );
29 :  void timer( unsigned long timer_set );
30 :
```

```
/************************************************************************/
/* メインプログラム                                                     */
/************************************************************************/
void main( void )
{
    unsigned char d;

    init();                             /* 初期化                       */

    while( 1 ) {
        p6 = 0x55;
        timer( 1000 );
        p6 = 0xaa;
        timer( 1000 );
        p6 = 0x00;
        timer( 1000 );
    }
}

/************************************************************************/
/* R8C/35A スペシャルファンクションレジスタ(SFR)の初期化                */
/************************************************************************/
void init( void )
{
    int i;

    /* クロックをXINクロック(20MHz)に変更 */
    prc0  = 1;                          /* プロテクト解除               */
    cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
    cm05  = 0;                          /* XINクロック発振              */
    for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
    ocd2  = 0;                          /* システムクロックをXINにする  */
    prc0  = 0;                          /* プロテクトON                 */

    /* ポートの入出力設定 */
    prc2 = 1;                           /* PD0のプロテクト解除          */
    pd0 = 0xe0;                         /* 7-5:LED 4:MicroSW 3-0:Sensor */
    p1  = 0x0f;                         /* 3-0:LEDは消灯                */
    pd1 = 0xdf;                         /* 5:RXD0 4:TXD0 3-0:LED        */
    pd2 = 0xfe;                         /* 0:PushSW                     */
    pd3 = 0xfb;                         /* 4:Buzzer 2:IR                */
    pd4 = 0x83;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
    pd5 = 0x40;                         /* 7:DIP SW                     */
    pd6 = 0xff;                         /* LEDなど出力                  */
}

/************************************************************************/
/* タイマ本体                                                           */
/* 引数   タイマ値 1=1ms                                                */
/************************************************************************/
void timer( unsigned long timer_set )
{
    int i;

    do {
        for( i=0; i<1240; i++ );
    } while( timer_set-- );
}

/************************************************************************/
/* end of file                                                          */
/************************************************************************/
```

13.5 プログラムの解説

13.5.1 timer関数(時間稼ぎ)

timer 関数は、実行した行で時間稼ぎをする関数です。

```
 81 :  void timer( unsigned long timer_set )
 82 :  {
 83 :      int i;
 84 :
 85 :      do {
 86 :          for( i=0; i<1240; i++ );        この行で 1ms の時間稼ぎ
 87 :      } while( timer_set-- );
 88 :  }
```

86 行	この行で、1ms の時間稼ぎをします。i を 1 足して 1240 以下なら for の次の命令を実行します。今回は、命令がないので、何もせずに終わります。また i を 1 足して・・・ を繰り返し、i が 1240 になったら次の行へ移ります。この繰り返しが 1ms になります。1240 という数字は、実測です。 ※1240 について この数値は、 ・ルネサス統合開発環境のバージョン(コンパイラのバージョン) ・ツールチェインの設定 ・クリスタルの値 によって違います。今回の条件固有の数値と覚えておくと良いでしょう。

85 行、
87 行
	do {
	　命令
	} while(条件);
	として、条件が成り立つ間、命令を実行し続けます。今回の条件は、timer_set 変数を-1 ずつして、0 になったら終了です。timer_set 変数は、関数の引数です。例えば、
	timer(1000);
	と実行したなら、timer_set 変数には 1000 が代入され、do～while 文が 1000 回実行されることになります。

使い方を次に示します。

```
timer( 時間稼ぎする時間[ms] );
```

カッコの中には、時間稼ぎをする時間を ms 単位で代入します。1 秒にしたいなら、1 秒＝1000ms なので、1000 を代入します。

13.5.2 main関数

```
34 :  void main( void )
35 :  {
36 :      unsigned char d;
37 :
38 :      init();                                 /* 初期化                          */
39 :
40 :      while( 1 ) {
41 :          p6 = 0x55;
42 :          timer( 1000 );
43 :          p6 = 0xaa;
44 :          timer( 1000 );
45 :          p6 = 0x00;
46 :          timer( 1000 );
47 :      }
48 :  }
```

41 行	ポート 6 に 0x55(0101 0101)を出力します。
42 行	timer 関数で 1000ms(＝1 秒)の時間稼ぎをします。
43 行	ポート 6 に 0xaa(1010 1010)を出力します。
44 行	timer 関数で 1000ms(＝1 秒)の時間稼ぎをします。
45 行	ポート 6 に 0x00(0000 0000)を出力します。
46 行	timer 関数で 1000ms(＝1 秒)の時間稼ぎをします。

※命令の実行時間について

プログラムの 1 命令は、数百 ns(ナノ秒)から数μs(マイクロ秒)という非常に短い時間で終わります。逆に言うと、短くても時間がかかるということで、何十万回も繰り返すと秒単位の時間となります。timer 関数は、何もしないことを何千回も繰り返すことによって、長い時間、時間稼ぎをしています。

main 関数のそれぞれの行の実行時間を、下記に示します。

```
34 :  void main( void )
35 :  {
36 :      unsigned char d;
37 :
38 :      init();                 ←init関数内の命令を実行する時間かかる(数百μs程度)
39 :
40 :      while( 1 ) {
41 :          p6 = 0x55;          ←数μs
42 :          timer( 1000 );      ←約1000ms
43 :          p6 = 0xaa;          ←数μs
44 :          timer( 1000 );      ←約1000ms
45 :          p6 = 0x00;          ←数μs
46 :          timer( 1000 );      ←約1000ms
47 :      }
48 :  }
```

13.6 演習

(1) 次の状態をポート 6 の LED に出力するようにしなさい。

① 1111 0000 を 0.5 秒間

② 0000 1111 を 0.5 秒間

③ 0000 0000 を 0.25 秒間

(2) 次の状態をマイコンボードの LED に出力するようにしなさい。

① 0101 を 0.2 秒間

② 1010 を 0.2 秒間

14. 割り込みによるタイマ(プロジェクト:timer2)

14.1 概要

本章は、動作はプロジェクト「timer1」と同じですが、時間の測り方を R8C/35A 内蔵のタイマ RB を使い、正確に時間を計ります。具体的には、タイマ RB で 1ms ごとに割り込みを発生させ、その回数で時間を計ります。

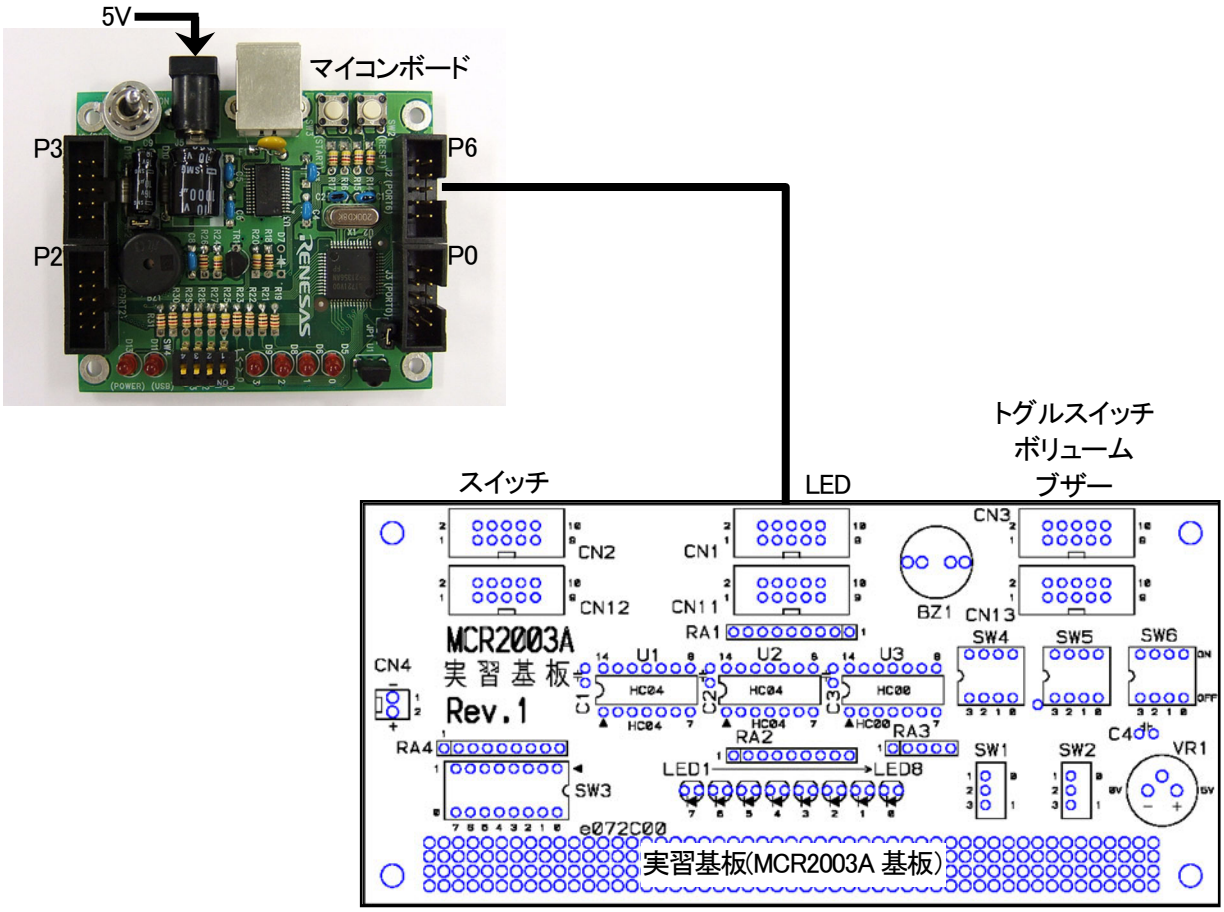
14.2 接続

■使用ポート

マイコンのポート	接続内容
P6 (J2)	実習基板の LED 部など、出力機器を接続します。

■接続例

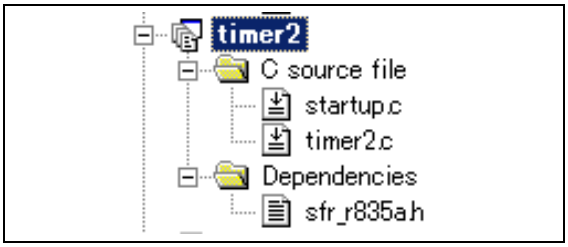
実習基板を使ったときの接続例を次に示します。



■操作方法

操作は特にありません。電源を入れるとLEDが点滅します。LEDの点滅の仕方をよく観察してください。

14.3 プロジェクトの構成



	ファイル名	内容
1	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。このファイルは共通で、どのプロジェクトもこのファイルから実行されます。
2	timer2.c	実際に制御するプログラムが書かれています。R8C/35A の内蔵周辺機能(SFR)の初期化も行います。
3	sfr_r835a.h	R8C/35A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Registers)を定義したファイルです。

14.4 プログラム「timer2.c」

```
 1 :  /***********************************************************************/
 2 :  /* 対象マイコン R8C/35A                                                */
 3 :  /* ﾌｧｲﾙ内容     タイマRB割り込みによるタイマ                           */
 4 :  /* バージョン   Ver.1.20                                               */
 5 :  /* Date         2010.04.19                                             */
 6 :  /* Copyright    ルネサスマイコンカーラリー事務局                       */
 7 :  /*              日立インターメディックス株式会社                       */
 8 :  /***********************************************************************/
 9 :  /*
10 :  出力 : P6_7-P6_0(LEDなど)
11 :
12 :  ポート6に繋いだLEDを1秒間隔で点滅させます。
13 :  タイマはタイマRB割り込みによる正確なタイマを使用します。
14 :  */
15 :
16 :  /*=====================================*/
17 :  /* インクルード                        */
18 :  /*=====================================*/
19 :  #include "sfr_r835a.h"                 /* R8C/35A SFRの定義ファイル     */
20 :
21 :  /*=====================================*/
22 :  /* シンボル定義                        */
23 :  /*=====================================*/
24 :
25 :  /*=====================================*/
26 :  /* プロトタイプ宣言                    */
27 :  /*=====================================*/
28 :  void init( void );
29 :  void timer( unsigned long timer_set );
30 :
31 :  /*=====================================*/
32 :  /* グローバル変数の宣言                */
33 :  /*=====================================*/
34 :  unsigned long cnt_rb;                  /* タイマRB用                    */
35 :
```

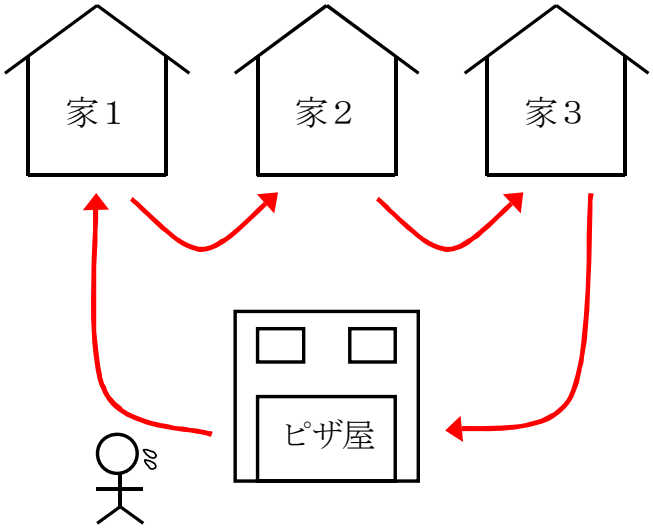
```
 36 :  /************************************************************************/
 37 :  /* メインプログラム                                                     */
 38 :  /************************************************************************/
 39 :  void main( void )
 40 :  {
 41 :      init();                             /* 初期化                       */
 42 :      asm(" fset I ");                    /* 全体の割り込み許可           */
 43 :
 44 :      while( 1 ) {
 45 :          p6 = 0x55;
 46 :          timer( 1000 );
 47 :          p6 = 0xaa;
 48 :          timer( 1000 );
 49 :          p6 = 0x00;
 50 :          timer( 1000 );
 51 :      }
 52 :  }
 53 :
 54 :  /************************************************************************/
 55 :  /* R8C/35A スペシャルファンクションレジスタ(SFR)の初期化                */
 56 :  /************************************************************************/
 57 :  void init( void )
 58 :  {
 59 :      int i;
 60 :
 61 :      /* クロックをXINクロック(20MHz)に変更 */
 62 :      prc0  = 1;                          /* プロテクト解除               */
 63 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
 64 :      cm05  = 0;                          /* XINクロック発振              */
 65 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
 66 :      ocd2  = 0;                          /* システムクロックをXINにする  */
 67 :      prc0  = 0;                          /* プロテクトON                 */
 68 :
 69 :      /* ポートの入出力設定 */
 70 :      prc2 = 1;                           /* PD0のプロテクト解除          */
 71 :      pd0 = 0xe0;                         /* 7-5:LED 4:MicroSW 3-0:Sensor */
 72 :      p1  = 0x0f;                         /* 3-0:LEDは消灯                */
 73 :      pd1 = 0xdf;                         /* 5:RXD0 4:TXD0 3-0:LED        */
 74 :      pd2 = 0xfe;                         /* 0:PushSW                     */
 75 :      pd3 = 0xfb;                         /* 4:Buzzer 2:IR                */
 76 :      pd4 = 0x83;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
 77 :      pd5 = 0x40;                         /* 7:DIP SW                     */
 78 :      pd6 = 0xff;                         /* LEDなど出力                  */
 79 :
 80 :      /* タイマRBの設定 */
 81 :      /* 割り込み周期 = 1 / 20[MHz]    * (TRBPRE+1) * (TRBPR+1)
 82 :                      = 1 / (20*10^-6) * 200        * 100
 83 :                      = 0.001[s] = 1[ms]
 84 :      */
 85 :      trbmr  = 0x00;                      /* 動作モード、分周比設定       */
 86 :      trbpre = 200-1;                     /* プリスケーラレジスタ         */
 87 :      trbpr  = 100-1;                     /* プライマリレジスタ           */
 88 :      trbic  = 0x07;                      /* 割り込み優先レベル設定       */
 89 :      trbcr  = 0x01;                      /* カウント開始                 */
 90 :  }
 91 :
 92 :  /************************************************************************/
 93 :  /* タイマ本体                                                           */
 94 :  /* 引数   タイマ値 1=1ms                                                */
 95 :  /************************************************************************/
 96 :  void timer( unsigned long timer_set )
 97 :  {
 98 :      cnt_rb = 0;
 99 :      while( cnt_rb < timer_set );
100 :  }
101 :
102 :  /************************************************************************/
103 :  /* タイマRB 割り込み処理                                                */
104 :  /************************************************************************/
105 :  #pragma interrupt intTRB(vect=24)
106 :  void intTRB( void )
107 :  {
108 :      cnt_rb++;
109 :  }
110 :
111 :  /************************************************************************/
112 :  /* end of file                                                          */
113 :  /************************************************************************/
```

14.5 プログラムの解説

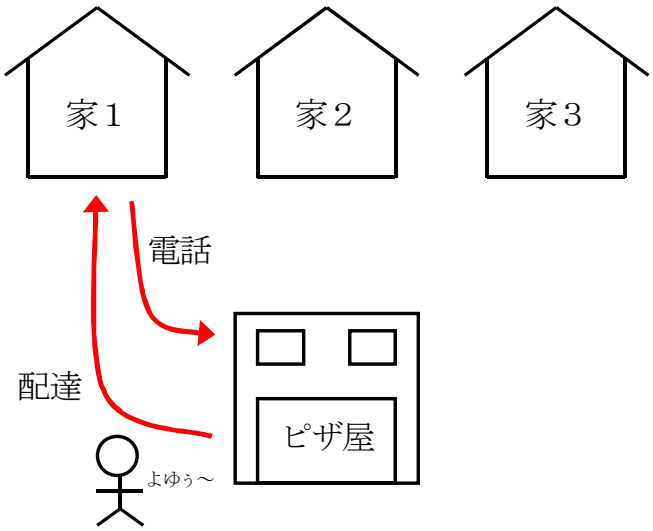
14.5.1 割り込みとは

(1) 割り込みとは

例えば、ピザ屋さんが家 1～3 に注文がないか回るとします。バイト君は、定期的に家を回らなければいけません。定期的に聞きに行くことを制御の用語で**ポーリング**といいます。回る間隔が長いと、待たせることになります。また、注文がなければ無駄足になってしまいます(下図)。



そこで、電話で注文を受けることにします。バイト君は、それぞれの家を回る必要がありません。電話の注文が来ればその家に届ければよいので作業効率が良いです(下図)。



ただし、電話を用意する必要があります。プログラムに当てはめると、割り込み設定に当たります。さらに、電話の受け答えをする必要があります。割り込みプログラムに当たります。

まとめると下記のようになります。

・注文がないか聞きに回る
制御の用語で「ポーリング」といいます。定期的に監視しなければいけないので、監視する部分が多いと、監視が遅れたり、監視もれが起こります。

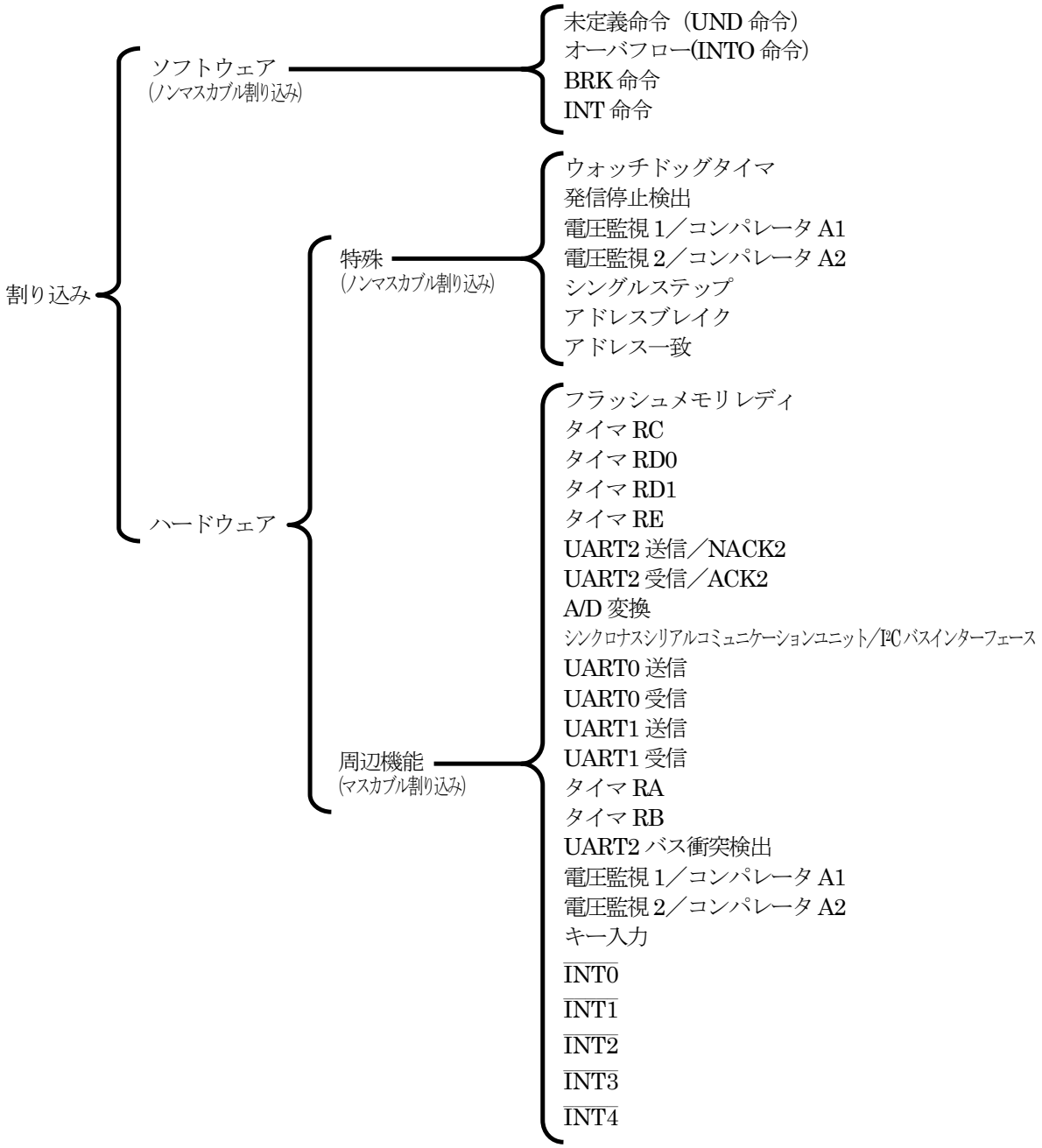
・電話で注文をうける
制御の用語（でもないですが）で「割り込み」といいます。電話のように、きっかけがあったときにだけ対処すれば良いので効率が良いです。ただし、電話の用意、電話の受け答えをする必要があります。

人で例えましたが、マイコンの場合は下記のようになります。

人間の場合		マイコンの場合
電話を用意する	→	割り込みプログラムを設定する
ベルが鳴る	→	割り込みが発生する
電話対応する	→	割り込みプログラムを実行する

(2) 割り込みの種類

R8C/35A の割り込みの種類を、下表に示します。



マスカブル割り込み	フラグレジスタ(FLG)の割り込み許可フラグ(I フラグ)による割り込みの許可(禁止)や割り込み優先レベルによる割り込み優先順位の変更が**可能**
ノンマスカブル割り込み	フラグレジスタ(FLG)の割り込み許可フラグ(I フラグ)による割り込みの許可(禁止)や割り込み優先レベルによる割り込み優先順位の変更が**不可能**

14.5.2 init関数(タイマRBの設定)

タイマ RB を使って、1ms ごとに割り込みを発生させます。

```
 80 :      /* タイマRBの設定 */
 81 :      /* 割り込み周期 = 1 / 20[MHz]    * (TRBPRE+1) * (TRBPR+1)
 82 :                     = 1 / (20*10^-6) * 200        * 100
 83 :                     = 0.001[s] = 1[ms]
 84 :      */
 85 :      trbmr  = 0x00;                    /* 動作モード、分周比設定        */
 86 :      trbpre = 200-1;                   /* プリスケーラレジスタ          */
 87 :      trbpr  = 100-1;                   /* プライマリレジスタ            */
 88 :      trbic  = 0x07;                    /* 割り込み優先レベル設定        */
 89 :      trbcr  = 0x01;                    /* カウント開始                  */
```

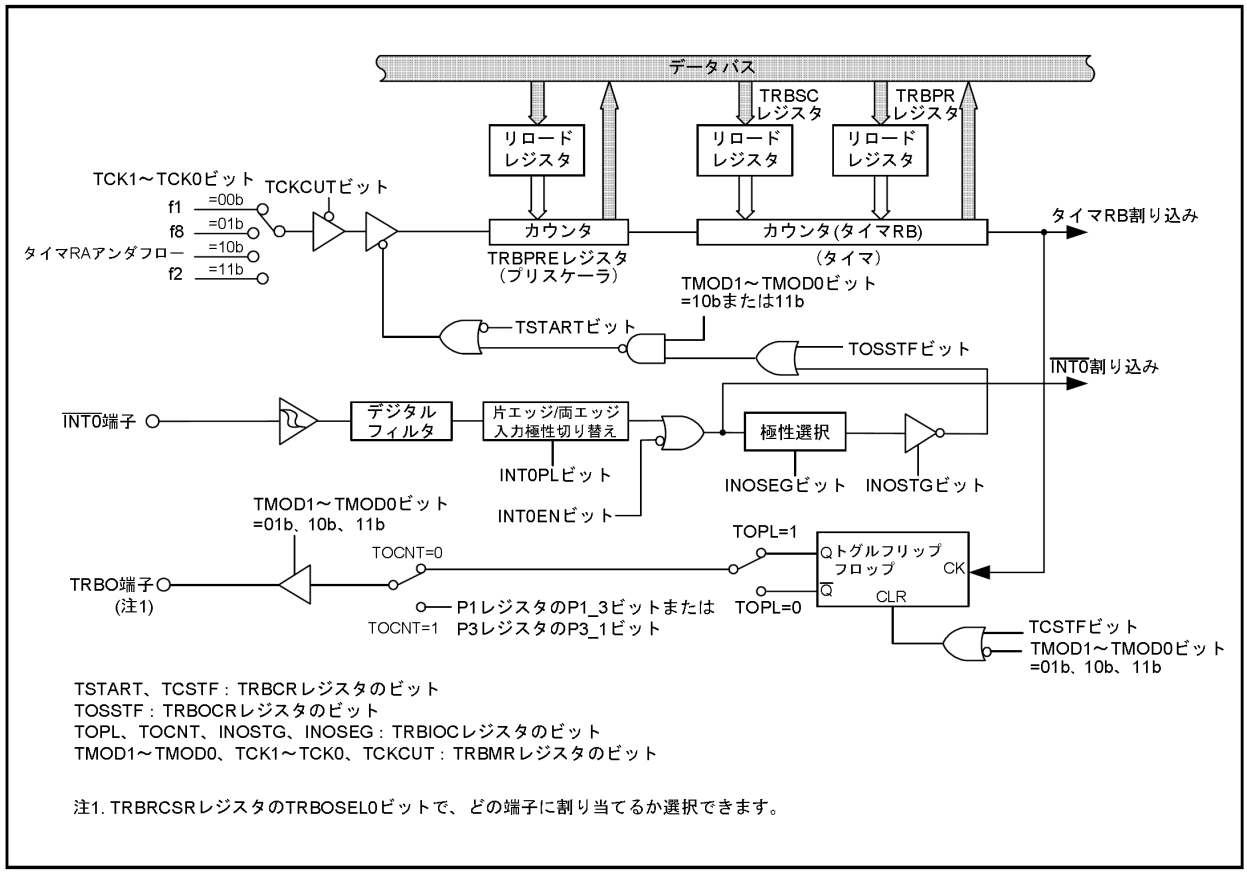
(1) タイマRBとは

R8C/35A には、タイマ RB というタイマが 1 個内蔵されています。タイマ RB には、次の 4 種類のモードがあります。

モード	詳細
タイマモード	内部カウントソース(周辺機能クロックまたはタイマ RA のアンダフロー)をカウントするモードです。
プログラマブル 波形発生モード	任意のパルス幅を連続して出力するモードです。
プログラマブル ワンショット発生モード	ワンショットパルスを出力するモードです。
プログラマブルウェイト ワンショット発生モード	ディレイドワンショットパルスを出力するモードです。

本プロジェクトでは、タイマモードを使い、1ms ごとに割り込みを発生させます。

(2) タイマRBのブロック図



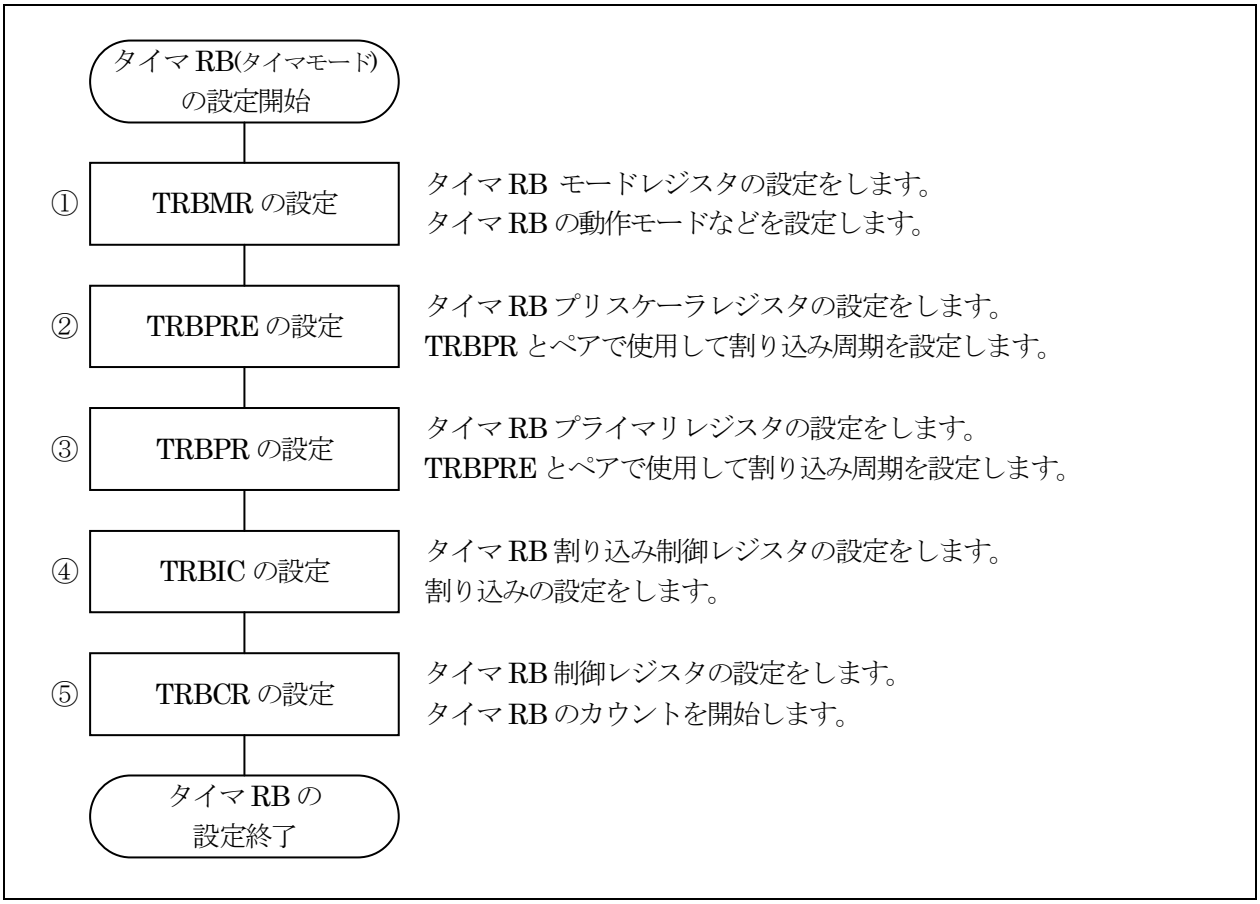
※タイマ RB の端子構成

端子名	割り当てる端子	入出力	機能
TRBO	P1_3 または P3_1	出力	パルス出力(プログラマブル波形発生モード、 プログラマブルワンショット発生モード、 プログラマブルウェイトワンショット発生モード)

今回は、1ms ごとの割り込みを発生させるだけなので、端子は使いません。

(3) タイマRBの設定(タイマモード)

今回は、タイマ RB をタイマモードで使用して、1ms ごとに割り込みを発生させるように設定にします。レジスタの設定手順を下記に示します。



①タイマ RB モードレジスタ(TRBMR:Timer RB mode register)の設定

タイマ RB のモードを設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7	タイマ RB カウントソース遮断ビット(注 1) tckcut_trbmr	0:カウントソース供給 1:カウントソース遮断 供給するので"0"を設定します。	0
bit6		"0"を設定	0
bit5,4	タイマ RB カウントソース選択ビット(注 1) bit5:tck1_trbmr bit4:tck0_trbmr	00:f1 (1/20MHz＝50ns) 01:f8 (8/20MHz＝400ns) 10:タイマ RA のアンダフロー 11:f2 (2/20MHz＝100ns) f1 を選択します。	00
bit3	タイマ RB 書き込み制御ビット(注 2) twrc_trbmr	0:リロードレジスタとカウンタへの書き込み 1:リロードレジスタのみ書き込み "0"を設定します。	0
bit2		"0"を設定	0
bit1,0	タイマ RB 動作モード選択ビット(注 1) bit1:tmod1_trbmr bit0:tmod0_trbmr	00:タイマモード 01:プログラマブル波形発生モード 10:プログラマブルワンショット発生モード 11:プログラマブルウェイトワンショット発生モード タイマモードで動作させるので"00"を設定します。	00

注 1. TMOD1～TMOD0 ビット、TCK1～TCK0 ビット、TCKCUT ビットは、TRBCR レジスタの TSTART ビットと TCSTF ビットが共に“0”(カウント停止)のときに変更してください。

注 2. TWRC ビットは、タイマモードのとき“0”または“1”が選択できます。プログラマブル波形発生モード、プログラマブルワンショット発生モード、プログラマブルウェイトワンショット発生モードでは“1”(リロードレジスタのみ書き込み)にしてください。

タイマ RB モードレジスタ(TRBMR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	**0**	0	**0**	**0**	**0**	0	**0**	**0**
16 進数	O				O			

※タイマRBモードレジスタ(TRBMR)のタイマRBカウントソース選択ビットの設定方法

タイマ RB モードレジスタ(TRBMR)のタイマ RB カウントソース選択ビット(bit5,4)で、タイマ RB プリスケーラレジスタ(TRBPRE)、タイマ RB プライマリレジスタ(TRBPR)がどのくらいの間隔で+1 するか設定します。
タイマ RB モードレジスタ(TRBMR)のタイマ RB カウントソース選択ビットの値と、割り込み間隔の関係を下記に示します。

TRBMR bit5,4	内容
00	タイマ RB プリスケーラレジスタ(TRBPRE)がカウントアップする時間を、f1 に設定します。時間は、 f1/20MHz＝1/20MHz＝50ns 設定できる割り込み間隔の最大は、 50ns×65,536＝**3.2768ms** よって、この時間以内の割り込み間隔を設定する場合は"00"を設定、これ以上の割り込み間隔を設定したい場合は次以降の値を検討します。
11	タイマ RB プリスケーラレジスタ(TRBPRE)がカウントアップする時間を、f2 に設定します。時間は、 f2/20MHz＝2/20MHz＝100ns 設定できる割り込み間隔の最大は、 100ns×65,536＝**6.5536ms** よって、この時間以内の割り込み間隔を設定する場合は"11"を設定、これ以上の割り込み間隔を設定したい場合は次以降の値を検討します。
01	タイマ RB プリスケーラレジスタ(TRBPRE)がカウントアップする時間を、f8 に設定します。時間は、 f8/20MHz＝8/20MHz＝400ns 設定できる割り込み間隔の最大は、 400ns×65,536＝**26.2144ms** よって、この時間以内の割り込み間隔を設定する場合は"01"を設定します。**これ以上の割り込み間隔を設定することはできません**。これ以上の割り込み間隔を設定しなくても良いように、プログラム側で工夫してください。

今回は、割り込み間隔を 1ms にします。
"00"の設定…最大の割り込み間隔は 3.2768ms、今回設定したい 1ms の割り込み間隔を設定できるので OK

よって、"00"を設定します。

②タイマ RB プリスケーラレジスタ(TRBPRE:Timer RB prescaler register)の設定
③タイマ RB プライマリレジスタ(TRBPR:Timer RB Primary Register)の設定

タイマ RB プリスケーラレジスタ(TRBPRE)とタイマ RB プライマリレジスタ(TRBPR)はペアで使い、割り込み周期を設定します。
TRBPRE と TRBPR の値を計算する式を、下記に示します。

タイマ RB 割り込み要求周期＝タイマ RB カウントソース×(TRBPRE＋1)×(TRBPR＋1)

TRBPRE と TRBPR を左辺に移動します。

(TRBPRE＋1)×(TRBPR＋1)＝タイマ RB 割り込み要求周期／タイマ RB カウントソース

今回、設定する割り込み周期は 1ms です。タイマ RB カウントソースとは、タイマ RB モードレジスタ(TRBMR)の bit5,4 に設定している内容で、今回は f1(50ns)です。よって、

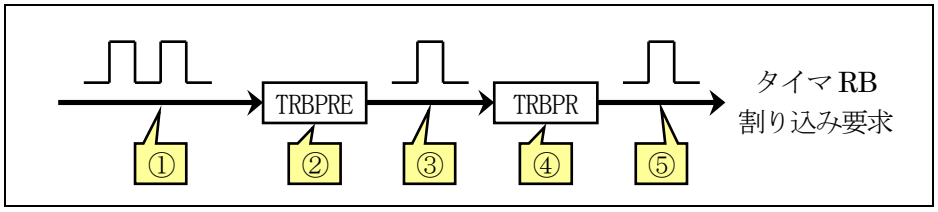
$(TRBPRE＋1)×(TRBPR＋1)＝(1×10^{-3})／(50×10^{-9})$
$\underline{(TRBPRE＋1)}×\underline{(TRBPR＋1)}＝\underline{20,000}$
B C A

次の条件になるよう、A、B、C 部分を設定してください。

A…65,536 以下にする必要があります。65,537 以上の場合、カウントソースを長い時間に設定し直してください。
B…1～256 以下になるよう、値を設定してください。値は整数です。
C…1～256 以下になるよう、値を設定してください。値は整数です。

今回、A は 20,000 なので、A の条件は満たしています。
B、C を決める公式はありません。B×C が、20,000 になるような数字を見つけてください。
例えば、B=200 とすると、
200×C＝20,000
∴C＝100
となります。
B＝TRBPRE＋1　∴**TRBPRE**＝200－1＝**199**
C＝TRBPR＋1　∴**TRBPR**　＝100－1＝**99**
を設定します。

このときの動作を下記に示します。



①	タイマ RB モードレジスタ(TRBMR)の RB カウントソース選択ビットで設定したパルスが入力されます。今回は f1 を選択しているので、次のパルスが入力されます。 f1=1/20MHz=50ns
②	タイマ RB プリスケーラレジスタ(TRBPRE)はダウンカウントです。設定した値からスタートし、1 つずつ値が減っていき 0 の次は設定値になります。例えば 9 を設定したなら 9→8→7→6→5→4→3→2→1→0→9→8・・・となります。このように 0 も含めてカウントするため、10 回カウントしたければ、1 小さい値の 9 を設定します。 今回は、199 を設定します。そのため、199→198→・・・→2→1→0→199→198・・・、とカウントされます。
③	タイマ RB プリスケーラレジスタ(TRBPRE)が 0→199 になった瞬間、1 パルス出力されます。これは TRBPRE に 200 パルス入ると 1 パルス出力されるということです。パルスが出力される間隔は、次のようになります。 50ns(入力されるパルスの間隔)×200＝10,000ns＝10μs
④	タイマ RB プライマリレジスタ(TRBPR)はダウンカウントです。設定値した値からスタート、0 の次は設定値になります。 今回は 99 を設定します。そのため、99→98→・・・→2→1→0→99→98・・・、とカウントされます。TRBPR には 10μs ごとにパルスが入力されます。要は、10μs ごとに、TRBPR が-1 されます。
⑤	タイマ RB プライマリレジスタ(TRBPR)が 0→99 になった瞬間、1 パルス出力されます。これは TRBPR に 100 パルス入ると 1 パルス出力されるということです。パルスが出力される間隔は、次のようになります。 10μs(入力されるパルスの間隔)×100＝1,000μs＝**1ms**

このように、TRBPR から 1ms ごとにパルスが出力されます。**このパルスが割り込みを発生させるきっかけになります**。要は、タイマ RB によって 1ms ごとに割り込みを発生させます。

タイマ RB プリスケーラレジスタ(TRBPRE)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	**199**							

タイマ RB プライマリレジスタ(TRBPR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	**99**							

④タイマ RB 割り込み制御レジスタ(TRBIC：Timer RB interrupt control register)の設定

タイマ RB の割り込み関係の設定をします。

設定 bit	上：ビット名 下：シンボル	内容	今回の内容
bit7～4		"0"を設定	0000
bit3	割り込み要求ビット ir_trbic	0：割り込み要求なし 1：割り込み要求あり **割り込みが発生すると自動で"1"になります。割り込みプログラムプログラムを実行すると自動的に"0"になります。設定は、"0"にします。**	0
bit2～0	割り込み優先レベル選択ビット bit2: ilvl2_trbic bit1: ilvl1_trbic bit0: ilvl0_trbic	000：レベル 0 (割り込み禁止) 001：レベル 1 010：レベル 2 011：レベル 3 100：レベル 4 101：レベル 5 110：レベル 6 111：レベル 7 他の割り込みが同時に発生した場合、どの割り込みを優先させるか設定します。レベルの高い割り込みが優先されます。割り込みを 2 つ以上使う場合は、どれを優先させるかここで決めます。今回の割り込みは、タイマ RB だけなのでレベル 1～7 のどれを設定しても構いません。一応、レベルのいちばん高い"111"を設定します。	111

タイマ RB 割り込み制御レジスタ(TRBIC)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	0	0	0	0	**1**	**1**	**1**
16 進数	0				7			

⑤タイマ RB 制御レジスタ(TRBCR:Timer RB Control Register)の設定

タイマ RB のカウント動作を開始するよう設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7～3		"00000"を設定	00000
bit2	タイマ RB カウント強制停止ビット(注 1、2) tstop_trbcr	"1"を書くとカウントが強制停止します。 読んだ場合、その値は"0"になります。	0
bit1	タイマ RB カウントステータスフラグ(注 1) tcstf_trbcr	0:カウント停止 1:カウント中(注 3) カウント中かどうかチェックするフラグです。書き込みは無効です。書き込むときは"0"を設定します。	0
bit0	タイマ RB カウント開始ビット(注 1) tstart_trbcr	0:カウント停止 1:カウント開始 タイマ RB のカウントを開始するので"1"を設定します。設定した瞬間から、カウントが開始されます。	1

注 1. TSTART、TCSTF、TSTOP ビットの使用上の注意事項については、ハードウェアマニュアルの「18.7 タイマ RB 使用上の注意」を参照してください。

注 2. TSTOP ビットに"1"を書くと、TRBPRE レジスタ、TRBSC レジスタ、TRBPR レジスタ、TSTART ビット、TCSTF ビット、TRBOCR レジスタの TOSSTF ビットがリセット後の値になります。

注 3. タイマモード、プログラマブル波形発生モードでは、カウント中を示します。プログラマブルワンショット発生モード、プログラマブルウェイトワンショット発生モードでは、ワンショットパルスのトリガを受け付けられることを示します。

タイマ RB 制御レジスタ(TRBCR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	0	0	0	0	0	0	**1**
16 進数	**0**				**1**			

14.5.3 intTRB関数（1msごとに実行される関数）

先の設定で、タイマ RB を 1ms ごとに割り込みを発生させる設定にしました。intTRB 関数は、この割り込みが発生したときに実行される関数です。

```
105 :  #pragma interrupt intTRB(vect=24)
106 :  void intTRB( void )
107 :  {
108 :      cnt_rb++;
109 :  }
```

105 行	#pragma interrupt **割り込み処理関数名**(vect=**ソフトウェア割り込み番号**) とすることで、ソフトウェア割り込み番号の割り込みが発生したとき、割り込み処理関数名を実行します。 ソフトウェア割り込み番号の表を次ページに示します。タイマ RB 割り込みは表より、24 番です。 よって、24 番の割り込みが発生したときに intTRB 関数を実行するよう、「#pragma interrupt」で設定します。
106 行	タイマ RB 割り込みにより実行する関数です。割り込み関数は、引数、戻り値ともに指定することはできません。すなわち、「void 関数名(void)」である必要があります。
108 行	cnt_rb 変数を+1 します。この関数は 1ms ごとに実行されるので、cnt_rb は 1ms ごとに+1 されることになります。

※ソフトウェア割り込み番号

割り込み要因とソフトウェア割り込み番号の関係は、下記のとおりです。

割り込み要因	ベクタ番地(注1) 番地(L)～番地(H)	ソフトウェア割り込み番号	割り込み制御レジスタ	参照先
BRK命令(注3)	+0～+3(0000h～0003h)	0	—	R8C/Tinyシリーズ ソフトウェアマニュアル
フラッシュメモリレディ	+4～+7(0004h～0007h)	1	FMRDYIC	32. フラッシュメモリ
—(予約)		2～5	—	—
$\overline{\text{INT4}}$	+24～+27(0018h～001BFh)	6	INT4IC	11.4 $\overline{\text{INT}}$割り込み
タイマRC	+28～+31(001Ch～001Fh)	7	TRCIC	19. タイマRC
タイマRD0	+32～+35(0020h～0023h)	8	TRD0IC	20. タイマRD
タイマRD1	+36～+39(0024h～0027h)	9	TRD1IC	
タイマRE	+40～+43(0028h～002Bh)	10	TREIC	21. タイマRE
UART2送信/NACK2	+44～+47(002Ch～002Fh)	11	S2TIC	23. シリアルインタフェース(UART2)
UART2受信/ACK2	+48～+51(0030h～0033h)	12	S2RIC	
キー入力	+52～+55(0034h～0037h)	13	KUPIC	11.5 キー入力割り込み
A/D変換	+56～+59(0038h～003Bh)	14	ADIC	28. A/Dコンバータ
シンクロナスシリアルコミュニケーションユニット/I^2Cバスインタフェース(注2)	+60～+63(003Ch～003Fh)	15	SSUIC/ IICIC	25. シンクロナスシリアルコミュニケーションユニット(SSU)、 26. I^2Cバスインタフェース
—(予約)		16	—	—
UART0送信	+68～+71(0044h～0047h)	17	S0TIC	22. シリアルインタフェース(UARTi (i=0～1))
UART0受信	+72～+75(0048h～004Bh)	18	S0RIC	
UART1送信	+76～+79(004Ch～004Fh)	19	S1TIC	
UART1受信	+80～+83(0050h～0053h)	20	S1RIC	
$\overline{\text{INT2}}$	+84～+87(0054h～0057h)	21	INT2IC	11.4 $\overline{\text{INT}}$割り込み
タイマRA	+88～+91(0058h～005Bh)	22	TRAIC	17. タイマRA
—(予約)		23	—	—
タイマRB	+96～+99(0060h～0063h)	24	TRBIC	18. タイマRB
$\overline{\text{INT1}}$	+100～+103(0064h～0067h)	25	INT1IC	11.4 $\overline{\text{INT}}$割り込み
$\overline{\text{INT3}}$	+104～+107(0068h～006Bh)	26	INT3IC	
—(予約)		27	—	—
—(予約)		28	—	—
$\overline{\text{INT0}}$	+116～+119(0074h～0077h)	29	INT0IC	11.4 $\overline{\text{INT}}$割り込み
UART2バス衝突検出	+120～+123(0078h～007Bh)	30	U2BCNIC	23. シリアルインタフェース(UART2)
—(予約)		31	—	—
ソフトウェア(注3)	+128～+131(0080h～0083h)～ +164～+167(00A4h～00A7h)	32～41	—	R8C/Tinyシリーズ ソフトウェアマニュアル
—(予約)		42～49	—	—
電圧監視1/コンパレータA1	+200～+203(00C8h～00CBh)	50	VCMP1IC	6. 電圧検出回路
電圧監視2/コンパレータA2	+204～+207(00CCh～00CFh)	51	VCMP2IC	30. コンパレータA
—(予約)		52～55	—	—
ソフトウェア(注3)	+224～+227(00E0h～00E3h)～ +252～+255(00FCh～00FFh)	56～63		R8C/Tinyシリーズ ソフトウェアマニュアル

注1. INTBレジスタが示す番地からの相対番地です。
注2. SSUIICSRレジスタのIICSELビットで選択できます。
注3. Iフラグによる禁止はできません。

今回は、タイマRBを使用して割り込みを発生させるので、表より番号は24番となります。

次のように、#pragma interrupt 命令を記述して、「ソフトウェア割り込み番号 24 番が発生したときに、実行する関数は割り込み処理関数名ですよ」ということを、宣言します。

```
#pragma interrupt 割り込み処理関数名(vect=24)
```

関数名のプログラムを記述して、割り込みが発生したときに実行するプログラムを作成します。

```
void 割り込み処理関数名( void )
{
    プログラム
}
```

14.5.4 timer関数（割り込みを使った時間稼ぎ）

timer 関数は、実行した行で時間稼ぎをする関数です。プロジェクト「timer1」はソフトウェアによるタイマでした。そのため、for 文で使用した 1240 という数値を見つけるのは大変です。また、コンパイラのバージョンの違いやツールチェインの設定により、時間が変わる可能性があります。今回の timer 関数は、クリスタルの値を基準としているため、正確な計測が可能です。

```
 92 :  /************************************************************************/
 93 :  /* タイマ本体                                                           */
 94 :  /* 引数    タイマ値 1=1ms                                               */
 95 :  /************************************************************************/
 96 :  void timer( unsigned long timer_set )
 97 :  {
 98 :      cnt_rb = 0;
 99 :      while( cnt_rb < timer_set );
100 :  }
```

98 行	cnt_rb を 0 にクリアします。

99 行	cnt_rb が timer_set より小さいなら、99 行を繰り返し続けます。
	cnt_rb は割り込みプログラムで 1ms ごとに+1 されます。timer_set は、timer 関数を実行したときに引数でセットした値です。
	例えば、
	`timer( 500 );`
	と実行したなら、timer_set には 500 が入ります。
	timer 関数を実行したいちばん最初は、
	`while( 0 < 500 );`
	となり、成り立つので 99 行を繰り返します。1ms 後は、割り込みプログラムで cnt_rb が+1 されるので、
	`while( 1 < 500 );`
	となります。まだ成り立つので、99 行目を繰り返します。timer 関数を実行してから 500ms たったなら、割り込みプログラムも 500 回実行され、cnt_rb は 500 となります。
	`while( 500 < 500 );`
	成り立たなくなり、次の行へ進みます。よって、99 行を実行し終わるまで 500ms かかることになります。

14.5.5 main関数

```
 39 :  void main( void )
 40 :  {
 41 :      init();                          /* 初期化                       */
 42 :      asm( " fset I ");                /* 全体の割り込み許可           */
 43 :
 44 :      while( 1 ) {
 45 :          p6 = 0x55;
 46 :          timer( 1000 );
 47 :          p6 = 0xaa;
 48 :          timer( 1000 );
 49 :          p6 = 0x00;
 50 :          timer( 1000 );
 51 :      }
 52 :  }
```

42 行	全体の割り込みを許可する命令です。 init 関数内でタイマ RB の割り込みを許可していますが、全体の割り込みを許可しなければ割り込みは発生しません。全体の割り込みを許可する命令は、C 言語で記述することができないため、asm 命令を使ってアセンブリ言語で割り込みを許可する命令を記述しています。

14.5.6 割り込みの発生タイミング

main 関数の while(1)のカッコ内の C 言語をアセンブリ言語に変換すると、下記のようになります。

```
; while(1)のカッコ内をアセンブリ言語に
; 変換した内容

L1:
        mov.w   #0001H,R0
        jeq     L3
        mov.b   #55H,_p6_addr
        push.w  #0000H
        push.w  #03e8H
        jsr     _timer
        add.b   #04H,SP
        mov.b   #0aaH,_p6_addr
                                (右上へ続く)
```

```
(左下から続き)
        push.w  #0000H
        push.w  #03e8H
        jsr     _timer
        add.b   #04H,SP
        mov.b   #00H,_p6_addr
        push.w  #0000H
        push.w  #03e8H
        jsr     _timer
        add.b   #04H,SP
        jmp     L1
L3:
```

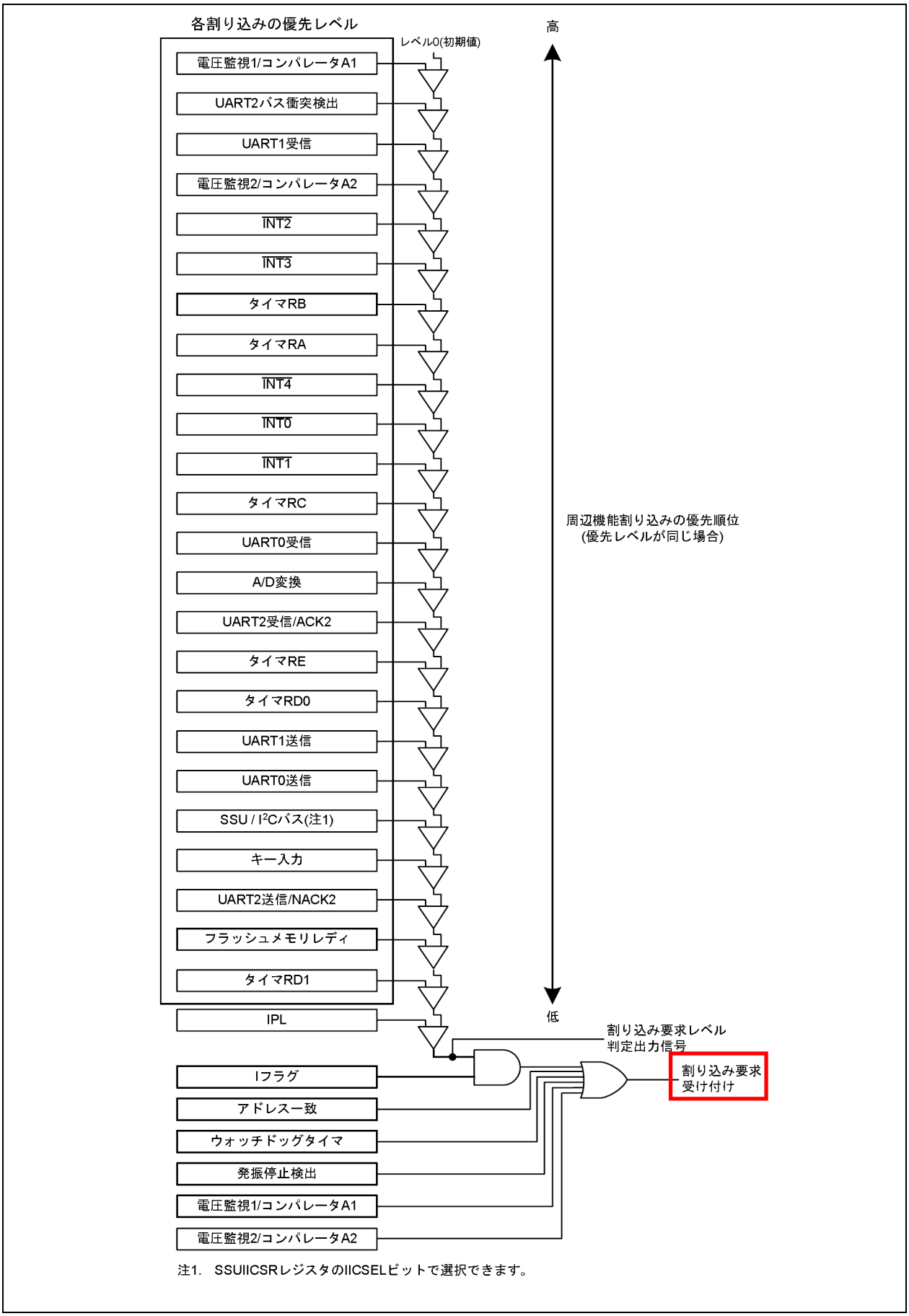
割り込みがあるかどうかは、アセンブリ言語の命令を 1 行終了後、毎回チェックが行われます。
イメージを下記に示します。

```
; while(1)のカッコ内をアセンブリ言語に
; 変換した内容

L1:
        mov.w   #0001H,R0
        割り込み要求があるかチェック
        jeq     L3
        割り込み要求があるかチェック
        mov.b   #55H,_p6_addr
        割り込み要求があるかチェック
        push.w  #0000H
        割り込み要求があるかチェック
        push.w  #03e8H
        割り込み要求があるかチェック
        jsr     _timer
        割り込み要求があるかチェック
        add.b   #04H,SP
        割り込み要求があるかチェック
        mov.b   #0aaH,_p6_addr
        割り込み要求があるかチェック
                                (右上へ続く)
```

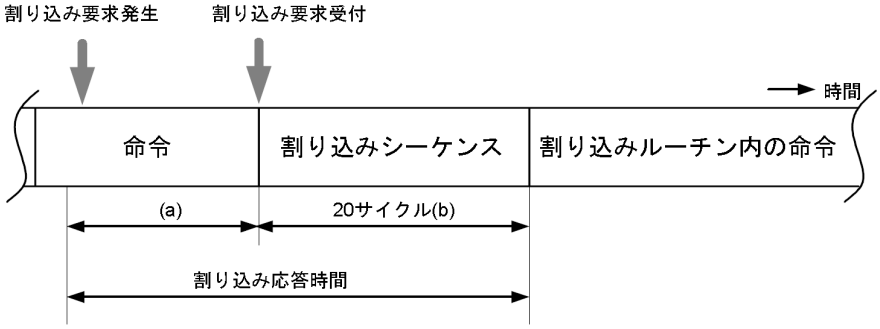
```
(左下から続き)
        push.w  #0000H
        割り込み要求があるかチェック
        push.w  #03e8H
        割り込み要求があるかチェック
        jsr     _timer
        割り込み要求があるかチェック
        add.b   #04H,SP
        割り込み要求があるかチェック
        mov.b   #00H,_p6_addr
        割り込み要求があるかチェック
        push.w  #0000H
        割り込み要求があるかチェック
        push.w  #03e8H
        割り込み要求があるかチェック
        jsr     _timer
        割り込み要求があるかチェック
        add.b   #04H,SP
        割り込み要求があるかチェック
        jmp     L1
L3:
```

割り込み要求があるかチェックの部分は、下記の「割り込み要求受け付け」信号をチェック、"1"なら割り込みありと判断します。



「割り込み要求受け付け」が"1"なら、割り込みが発生した割り込みプログラムを実行します。

割り込み応答時間を下記に示します。割り込み応答時間は、割り込み要求が発生してから割り込みルーチン内の最初の命令を実行するまでの時間です。この時間は、割り込み要求発生時点から、そのとき実行している命令が終了するまでの時間(a)と割り込みシーケンスを実行する時間(20 サイクル(b))で構成されます。



(a) 割り込み要求発生時点からそのとき実行している命令が終了するまでの時間。実行している命令によって異なります。この時間が最も長くなる命令はDIVX命令で30サイクル(ウェイトなし、除数がレジスタの場合のサイクル数)です。
(b) アドレス一致割り込み、シングルステップ割り込みは21サイクルです。

本マイコンボードは、20MHz のクリスタルで動作しています。1 サイクルは、

1/20MHz＝50ns

です。割り込みシーケンスの時間は、20 サイクルですので、

割り込みシーケンスの時間 ＝ 1 サイクル×20 ＝ 50ns×20 ＝ 1000ns ＝ 1μs

となります。

割り込みルーチンの処理が終わったら、割り込みシーケンス発生前に実行していた命令の、次の命令から実行を再開します。