マイコン実習マニュアル
R8C/38A 版
演習解答例

第 1.07 版
2015.04.20
ジャパンマイコンカーラリー実行委員会
株式会社日立ドキュメントソリューションズ

注 意 事 項 (rev.6.0J)

著作権

・本マニュアルに関する著作権はジャパンマイコンカーラリー実行委員会に帰属します。
・本マニュアルは著作権法および、国際著作権条約により保護されています。

禁止事項

ユーザーは以下の内容を行うことはできません。
・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと
・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること
・本マニュアルの一部または全部を改変、除去すること
・本マニュアルを無許可で翻訳すること
・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

転載、複製

本マニュアルの転載、複製については、文書によるジャパンマイコンカーラリー実行委員会の事前の承諾が必要です。

責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したものですが万一本マニュアルの記述誤りに起因する損害が生じた場合でも、ジャパンマイコンカーラリー実行委員会はその責任を負いません。

その他

・本マニュアルに記載の情報は本マニュアル発行時点のものであり、ジャパンマイコンカーラリー実行委員会は、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、最新の内容を確認いただきますようお願いします。
・すべての商標および登録商標は、それぞれの所有者に帰属します。

連絡先

株式会社 日立ドキュメントソリューションズ
〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー
E-mail:himdx.m-carrally.dd@hitachi.com

目　次

1. I/Oポートの入出力(プロジェクト:io)

1.1. 演習(1)

■問題:ポート 0 に LED 部、ポート 2 にスライドスイッチ部を接続して、スライドスイッチの状態を LED へ出力しなさい。

■解答例

メインプログラムの「**太字**」を修正します。

```
034 void main( void )
035 {
036     unsigned char d;
037
038     init();                             /* 初期化                          */
039
040     while( 1 ) {
041         d = p2;
042         p0 = d;
043     }
044 }
```

スライドスイッチの「ON/OFF」情報は、P2 レジスタから読み込みます。P2 レジスタの値を変数「d」へ代入します。変数「d」へ代入された値を P0 レジスタに代入し、LED へ出力します。

ポートの入出力設定をします。「**太字**」を修正します。

```
061     /* ポートの入出力設定 */
062     prc2 = 1;                           /* PD0のプロテクト解除             */
063     pd0 = 0xff;                         /* 7-0:スイッチ                    */
064     pd1 = 0xd0;                         /* 5:RXD0 4:TXD0 3-0:DIP SW        */
065     pd2 = 0x00;                         /* 7-0:LED                         */
066     pd3 = 0xff;                         /*                                 */
```

ポート 0 に LED 部を接続するため「**出力**」に、ポート 2 にスライドスイッチ部を接続するため「**入力**」に、それぞれ設定します。入力は"0"、出力は"1"に設定しますので、63 行目の PD0 は「**0xff**」、65 行目の PD2 は「**0x00**」に設定します。

1.2. 演習(2)

■問題:(1)のとき、スライドスイッチの状態 bit7～bit0 を読み込み、読み込んだ値の bit7～bit4 は強制的に"0"にして bit3～bit0 はそのままの値を LED へ出力しなさい(bit7～bit4 の LED は常に消灯、bit3～bit0 はディップスイッチの状態により点灯／消灯する)。

■解答例　**※(1)の解答例の状態から説明します。**

メインプログラムの「**太字**」を追加します。

```
040     while( 1 ) {
041         d = p2;
042         p0 = d & 0x0f;
043     }
```

スライドスイッチの「ON/OFF」情報が格納されている変数「d」の上位ビット(bit7～4)を強制的に"0"にするため、論理演算の論理積(AND)を使います。論理積をC言語で表現するときは記号の「＆」(アンパサンド)を使います。

42 行目の変数「d」の値と「0x0f」の AND 処理を行うことによって、上位 4bit は"0"になり、下位 4bit のスライドスイッチの値だけを取り出すことができます。

	bit	7	6	5	4	3	2	1	0	
	変数 d	0	1	0	1	0	1	0	1	← スライドスイッチの値
AND)	0x0f	0	0	0	0	1	1	1	1	← マスク値
	0x05	0	0	0	0	0	1	0	1	← 下位 4bit の値

このように、下位 4bit の値だけを取り出し、上位 4bit は強制的に"0"にすることができます。

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「11.6.1　特定のビットを"0"にする(マスク処理)」を参照してください。

1.3. 演習(3)

■問題:(1)のとき、スライドスイッチの状態 bit7～bit0 を読み込み、読み込んだ値の bit3～bit0 は強制的に"1"にして bit3～bit0 はそのままの値を LED へ出力しなさい(bit3～bit0 の LED は常に点灯、bit7～bit4 はディップスイッチの状態により点灯／消灯する)。

■解答例　**※(1)の解答例の状態から説明します。**

メインプログラムの「**太字**」を追加します。

```
040     while( 1 ) {
041         d = p2;
042         p0 = d | 0x0f;
043     }
```

スライドスイッチの「ON/OFF」情報が格納されている変数「d」の下位ビット(bit3～0)を強制的に"1"にするため、論理演算の論理和(OR)を使います。論理和をC言語で表現するときは記号の「｜」(パイプライン)を使います。

42 行目の変数「d」の値と「0x0f」の OR 処理を行うことによって、下位 4bit は"1"になり、上位 4bit のスライドスイッチの値だけを取り出すことができます。

	bit	7	6	5	4	3	2	1	0	
	変数 d	0	1	0	1	0	1	0	1	← スライドスイッチの値
OR)	0x0f	0	0	0	0	1	1	1	1	
	0x5f	0	1	0	1	1	1	1	1	← 上位 4bit の値

このように、上位 4bit の値だけを取り出し、下位 4bit は強制的に"1"にすることができます。

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「11.6.2　特定のビットを"1"にする」を参照してください。

1.4. 演習(4)

■問題:(1)のとき、スライドスイッチの状態 bit7～bit0 を読み込み、読み込んだ値をすべて反転して LED へ出力しなさい。

■解答例　**※(1)の解答例の状態から説明します。**

メインプログラムの「**太字**」を追加します。

```
040     while( 1 ) {
041         d = p2;
042         p0 = ~d;
043     }
```

スライドスイッチの「ON/OFF」情報が格納されている変数「d」の値をすべて反転させます。反転させるために論理演算の否定(NOT)を使います。否定をC言語で表現するときは記号の「~」(チルダ)を使います。

42 行目の変数「d」の前に「~」を付けて NOT 処理を行います。このようにするとスライドスイッチの値をすべて反転させることができます。

	bit	7	6	5	4	3	2	1	0	
NOT)	変数 d	0	1	0	1	0	1	0	1	← スライドスイッチの値
	0xaa	1	0	1	0	1	0	1	0	← 反転された値

このように、"1"・"0"の値をすべて反転することができます。

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「11.6.4 全ビット反転する(NOT 演算)」を参照してください。

1.5. 演習(5)

■問題:(1)のとき、スライドスイッチの状態 bit7～bit0 を読み込み、bit4 が"1"なら LED に(1111 0000)を、bit4 が"0"なら LED に(0000 1111)を出力しなさい。ただし、スライドスイッチの bit4 以外は"1"か"0"か分からないものとする。

■解答例　**※(1)の解答例の状態から説明します。**

メインプログラムを「**太字**」に修正します。

```
040     while( 1 ) {
041         d = p2;
042         if( ( d & 0x10 ) == 0x10 ){
043             p0 = 0xf0;
044         }else{
045             p0 = 0x0f;
046         }
047     }
```

スライドスイッチの「ON/OFF」情報が格納されている変数「d」の値からbit4 のスライドスイッチの NO/OFF 情報を取り出します。取り出す方法として、bit4 以外の bit を「1.2　演習(2)」で行った、bit を強制的に"0"にする方法使って"0"にします。42 行目の「**(d & 0x10)**」と記述すると bit4 の"1"・"0"の情報だけを取り出すことができます。後は、42 行目～46 行目の if 文で比較し、bit4 が"1"なら LED に(1111 0000)を、bit4 が"0"なら LED に(0000 1111)を出力するようにします。

1.6. 演習(6)

■問題:(1)のとき、スライドスイッチの状態 bit7～bit0 を読み込み、読み込んだ値の bit5～2 のみを反転して、その他の bit はそのままの値を LED へ出力しなさい。

■解答例　**※(1)の解答例の状態から説明します。**

メインプログラムの「**太字**」を追加します。

```
040     while( 1 ) {
041         d = p2;
042         p0 = d ^ 0x3c;
043     }
```

スライドスイッチの「ON/OFF」情報が格納されている変数「d」の値の特定のビットだけ反転させます。特定のビットを反転させるため、論理演算の排他的論理和(XOR)を使います。排他的論理和をC言語で表現するときは記号の「^」(アクサンシルコンフレックス)を使います。

42 行目の変数「d」の前に「^」を付けて XOR 処理を行います。このようにすることでスライドスイッチの値の特定のビットのみを反転させることができます。

	bit	7	6	5	4	3	2	1	0	
	変数 d	0	1	0	1	0	1	0	1	← スライドスイッチの値
XOR)	0x3c	0	0	1	1	1	1	0	0	
	0x69	0	1	1	0	1	0	0	1	← bit5～bit2 の値のみ反転

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「11.6.3　特定のビットを反転する」を参照してください。

2. I/Oポートの入出力2(プロジェクト:io2)

2.1. 演習(1)

■問題:98 行目、100 行目を「Px_y」の形式に書き直しなさい。

■解答例

ポートのシンボル名は、下記のように定義されています。

px_y : ポート x の bit y

98 行目、100 行目を「**太字**」のように修正します。

```
 91 :  /************************************************************************/
 92 :  /* マイコンボードのLED出力                                               */
 93 :  /* 引数    0:消灯 1:点灯                                                */
 94 :  /************************************************************************/
 95 :  void led_out_m( unsigned char led )
 96 :  {
 97 :      if( led ) {
 98 :          p4_5 = 1;                   /* LED点灯                        */
 99 :      } else {
100 :          p4_5 = 0;                   /* LED消灯                        */
101 :      }
102 :  }
```

3. ソフトウェアによるタイマ(プロジェクト:timer1)

3.1. 演習(1)

■問題:次の状態をポート 2 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

■解答例

メインプログラムを「**太字**」に修正します。

```
038     while( 1 ) {
039         p2 = 0xf0;
040         timer( 500 );
041         p2 = 0x0f;
042         timer( 500 );
043         p2 = 0x00;
044         timer( 250 );
045     }
```

問題文に書かれている点灯パターンと点灯時間をメインプログラムに作成することでできます。

3.2. 演習(2)

■問題:次の状態を RY_R8C38 ボードの LED に出力するようにしなさい。

① "1"を 0.1 秒間
② "0"を 0.9 秒間

■解答例

マイコンボード上の LED に出力する関数がないため、関数のプロトタイプ宣言をします。「**太字**」を追加します。

```
025 /*====================================*/
026 /* プロトタイプ宣言                   */
027 /*====================================*/
028 void init( void );
029 void timer( unsigned long timer_set );
030 void led_out_m( unsigned char led );
```

メインプログラムを「**太字**」に修正します。

```
039     while( 1 ) {
040         led_out_m( 1 );
041         timer( 100 );
042         led_out_m( 0 );
043         timer( 900 );
044     }
```

問題文に書かれている点灯パターンと点灯時間をメインプログラムに作成することでできます。

マイコンボード上の LED 出力用の関数を作成します。「**太字**」を追加します。

```
091 /************************************************************************/
092 /* マイコンボードのLED出力                                              */
093 /* 引数    0:消灯 1:点灯                                                */
094 /************************************************************************/
095 void led_out_m( unsigned char led )
096 {
097     if( led ) {
098         p4 |= 0x20;                 /* LED点灯                          */
099     } else {
100         p4 &= 0xdf;                 /* LED消灯                          */
101     }
102 }
103
104 /************************************************************************/
105 /* end of file                                                          */
106 /************************************************************************/
```

RY_R8C38 ボード上の LED に出力させるための関数です。RY_R8C38 ボード上の LED は、P4_5 に接続されているため、点灯させたい場合は、論理和(OR)を使ってポート 4 の bit5 だけを強制的に"1"にします。消灯させたい場合は、論理積(AND)を使ってポート 4 の bit5 だけを強制的に"0"にします。

前述の関数では、P4_5 の 1bit を制御するのに、1 バイト単位でデータを処理しました。1 バイト単位で処理をするためには、論理和(OR)や論理積(AND)の論理演算を使わなければいけませんでした。

1bit 単位で制御できたならどうでしょうか。1bit 単位で制御する関数を紹介します。「**太字**」に修正します。

```
091 /************************************************************************/
092 /* マイコンボードのLED出力                                              */
093 /* 引数    0:消灯 1:点灯                                                */
094 /************************************************************************/
095 void led_out_m( unsigned char led )
096 {
097     if( led ) {
098         p4_5 = 1;                   /* LED点灯                          */
099     } else {
100         p4_5 = 0;                   /* LED消灯                          */
101     }
102 }
103
104 /************************************************************************/
105 /* end of file                                                          */
106 /************************************************************************/
```

プログラムが少しシンプルになりました。98 行と 100 行の「p4_5」は、ポート 4 の bit5 という意味になります。P4_5 を直接指定していますので、LED を点灯させたければ"1"を設定し、消灯させたければ"0"を設定します。

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「11.7　ビット単位でデータを入力、出力する」を参照してください。

4. 割り込みによるタイマ(プロジェクト:timer2)

4.1. 演習(1)

■問題:タイマ RB の割り込み間隔を 100μs にして、動作確認しなさい。ただし、timer 関数がある 45 行、47 行、49 行は 1 秒にすること(「timer(10000)」にする)。

■解答例

まず、割り込み周期を設定します。下記の式に、当てはめて算出します。

(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
(TRBPRE＋1)×(TRBPR＋1)＝(**100×10^{-6}**)　／(50×10^{-9})
(TRBPRE＋1)×(TRBPR＋1)＝2,000
B　C　A

A は、65536 以下なのでタイマ RB カウントソースは OK です。B と C の値を見つけます。今回は、B=20、C=100 にします。よって、

B ＝TRBPRE+1＝20 、 TRBPRE＝20-1
C ＝TRBPR+1＝100 、 TRBPR＝100-1

timer 関数は 1 あたり、100μs になります。45 行、47 行、49 行は、1 秒のタイマにするので、

1／(100×10^{-6})＝10000

を timer 関数に設定します。

メインプログラムを「**太字**」に修正します。

```
 35 :  /************************************************************************/
 36 :  /* メインプログラム                                                     */
 37 :  /************************************************************************/
 38 :  void main( void )
 39 :  {
 40 :      init();                          /* 初期化                        */
 41 :      asm(" fset I ");                 /* 全体の割り込み許可            */
 42 :
 43 :      while( 1 ) {
 44 :          p2 = 0x55;
 45 :          timer( 10000 );
 46 :          p2 = 0xaa;
 47 :          timer(10000 );
 48 :          p2 = 0x00;
 49 :          timer(10000 );
 50 :      }
 51 :  }

中略

 56 :  void init( void )
 57 :  {
 58 :      int i;
中略
 88 :      trbmr  = 0x00;                   /* 動作モード、分周比設定        */
 89 :      trbpre = 20-1;                   /* プリスケーラレジスタ          */
```

```
  90 :      trbpr  = 100-1;                  /* プライマリレジスタ          */
  91 :      trbic  = 0x07;                   /* 割り込み優先レベル設定      */
  92 :      trbcr  = 0x01;                   /* カウント開始                */
```

4.2. 演習(2)

■問題:タイマ RB の割り込み間隔を 10μs にして、動作確認しなさい。ただし、timer 関数がある 45 行、47 行、49 行は 1 秒にすること(「timer(100000)」にする)。

■解答例

まず、割り込み周期を設定します。下記の式に、当てはめて算出します。

(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
(TRBPRE+1)×(TRBPR+1)=(**10×10^{-6}**)　　／(50×10^{-9})
(TRBPRE+1)×(TRBPR+1)=200
B　　C　　A

A は、65536 以下なのでタイマ RB カウントソースは OK です。B と C の値を見つけます。今回は、B=20、C=10 にします。よって、

B =TRBPRE+1=20 、 TRBPRE=20-1
C =TRBPR+1=10 、 TRBPR=10-1

timer 関数は 1 あたり、10μs になります。45 行、47 行、49 行は、1 秒のタイマにするので、

$1/(10\times10^{-6})=100000$

を timer 関数に設定します。

メインプログラムを「**太字**」に修正します。

```
  35 :  /***********************************************************************/
  36 :  /* メインプログラム                                                    */
  37 :  /***********************************************************************/
  38 :  void main( void )
  39 :  {
  40 :      init();                          /* 初期化                      */
  41 :      asm(" fset I ");                 /* 全体の割り込み許可          */
  42 :
  43 :      while( 1 ) {
  44 :          p2 = 0x55;
  45 :          timer( 100000 );
  46 :          p2 = 0xaa;
  47 :          timer(100000 );
  48 :          p2 = 0x00;
  49 :          timer(100000 );
  50 :      }
  51 :  }

中略

  56 :  void init( void )
  57 :  {
  58 :      int i;
中略
```

```
 88 :      trbmr  = 0x00;                  /* 動作モード、分周比設定        */
 89 :      trbpre = 20-1;                  /* プリスケーラレジスタ          */
 90 :      trbpr  = 10-1;                  /* プライマリレジスタ            */
 91 :      trbic  = 0x07;                  /* 割り込み優先レベル設定        */
 92 :      trbcr  = 0x01;                  /* カウント開始                  */
```

4.3. 演習(3)

■問題:タイマ RB の割り込み間隔を 1μs にして、動作確認しなさい。ただし、timer 関数がある 45 行、47 行、49 行は 1 秒にすること(「timer(1000000)」にする)。

■解答例

まず、割り込み周期を設定します。下記の式に、当てはめて算出します。

(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
(TRBPRE＋1)×(TRBPR＋1)＝(**1×10^{-6}**)　　／(50×10^{-9})
(TRBPRE＋1)×(TRBPR＋1)＝20
B　　C　　A

A は、65536 以下なのでタイマ RB カウントソースは OK です。B と C の値を見つけます。今回は、B=2、C=10 にします。よって、

B ＝TRBPRE+1＝2 、 TRBPRE＝2-1
C ＝TRBPR+1＝10 、 TRBPR＝10-1

timer 関数は 1 あたり、1μs になります。45 行、47 行、49 行は、1 秒のタイマにするので、

1／(1×10^{-6})＝1000000

を timer 関数に設定します。

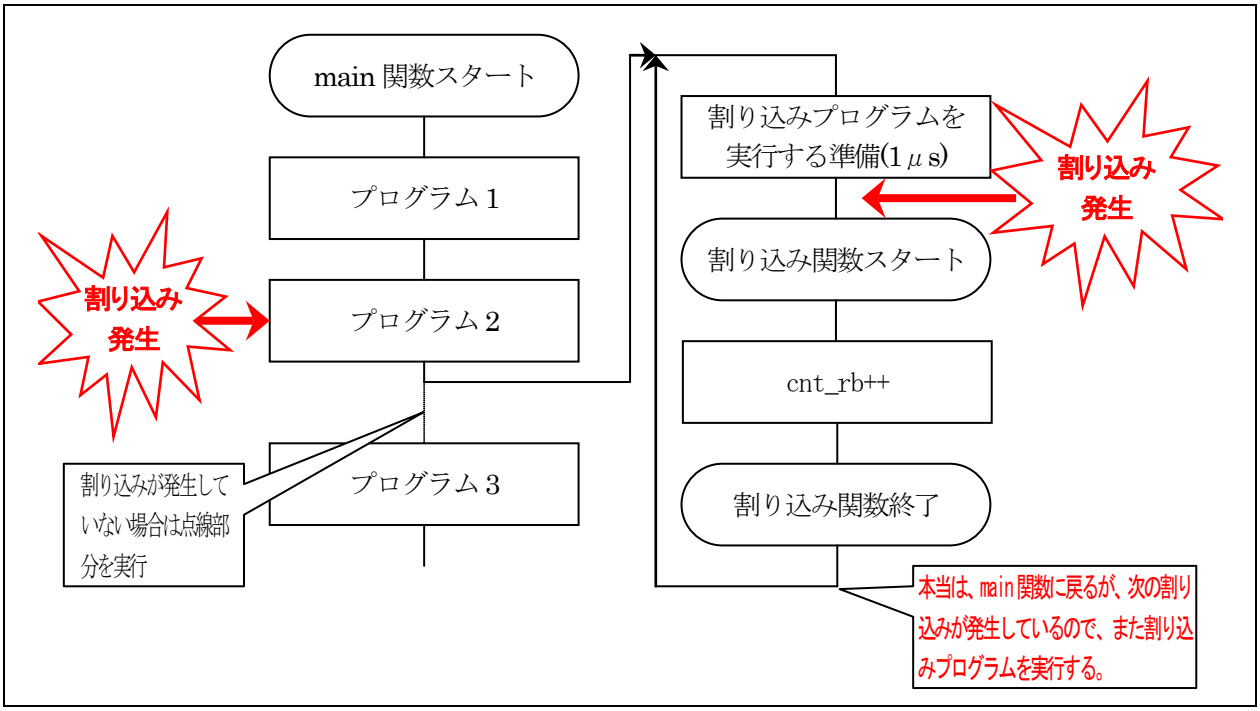
メインプログラムを「**太字**」に修正します。

```
 35 :  /************************************************************************/
 36 :  /* メインプログラム                                                     */
 37 :  /************************************************************************/
 38 :  void main( void )
 39 :  {
 40 :      init();                         /* 初期化                        */
 41 :      asm(" fset I ");                /* 全体の割り込み許可            */
 42 :
 43 :      while( 1 ) {
 44 :          p2 = 0x55;
 45 :          timer( 1000000 );
 46 :          p2 = 0xaa;
 47 :          timer( 1000000 );
 48 :          p2 = 0x00;
 49 :          timer( 1000000 );
 50 :      }
 51 :  }

中略
```

```
  56 :  void init( void )
  57 :  {
  58 :      int i;
中略
  88 :      trbmr  = 0x00;                    /* 動作モード、分周比設定        */
  89 :      trbpre = 2-1;                     /* プリスケーラレジスタ          */
  90 :      trbpr  = 10-1;                    /* プライマリレジスタ            */
  91 :      trbic  = 0x07;                    /* 割り込み優先レベル設定        */
  92 :      trbcr  = 0x01;                    /* カウント開始                  */
```

ただし、今回のプログラムは実行しても、正しく LED が点灯しません。
今回のプログラムの動きを、下図に示します。



割り込み関数を終了した時点で、既に次の割り込みが発生しているので main 関数を実行せずにすぐに割り込みプログラムを実行します。このように永遠に main 関数が実行されず割り込みプログラムだけが実行されることになります。

結論は、R8C マイコンでは、1μs ごとの割り込みは対応できないことになります。

10μs ごとの割り込みも、割り込みプログラムの実行時間が 9μs 以上になると同様のことがおきます。

R8C マイコンでは、割り込み間隔をできれば 100μs 以上にして、割り込みプログラムは割り込み周期以内に終わるようにしてください。

5. 7 セグメントLEDの表示(プロジェクト:seven_seg)

5.1. 演習(1)

■問題:RY_R8C38 ボードのディップスイッチ値(0～15)を、7 セグメント LED に表示するようにしなさい。

■解答例

RY_R8C38 ボード上にあるディップスイッチの値を読み込む関数がないため、関数のプロトタイプ宣言をします。「**太字**」を追加します。

```
026 /*====================================*/
027 /* プロトタイプ宣言                   */
028 /*====================================*/
029 void init( void );
030 void timer( unsigned long timer_set );
031 unsigned char dipsw_get( void );
```

割り込みプログラムを「**太字**」に修正します。

```
109 /************************************************************************/
110 /* タイマRB 割り込み処理                                                */
111 /************************************************************************/
112 #pragma interrupt intTRB(vect=24)
113 void intTRB( void )
114 {
115     int i;
116
117     cnt_rb++;
118
119     // 7セグメントLEDの表示値の更新
120     seg_out = dipsw_get();
121
122     // 7セグメントLED 表示処理
123     if( p0 & 0x01 ){
```

7 セグメント LED に表示したい値をグローバル変数「seg_out」に代入します。今回はディップスイッチの値を代入します。

ディップスイッチの関数を作成します。「**太字**」を追加します。

```
138 /************************************************************************/
139 /* ディップスイッチ値読み込み                                           */
140 /* 戻り値 スイッチ値 0～15                                              */
141 /************************************************************************/
142 unsigned char dipsw_get( void )
143 {
144     unsigned char sw;
145
146     sw = p1 & 0x0f;                 /* P1_3～P1_0読み込み           */
147
148     return  sw;
149 }
150
151 /************************************************************************/
152 /* end of file                                                          */
153 /************************************************************************/
```

ディップスイッチは、ポート 1 の bit3～0 に接続されているため、ポート 1 の上位ビット(bit7～4)を強制的に"0"にして下位ビット(bit3～0)の値を取り出します。

5.2. 演習(2)

■問題:「a」～「f」までの 7 セグメント LED データを作り、「00」～「ff」(10 進数で 0～255)まで 16 進数表記で 0.1 秒ごとに表示内容が＋1するようにしなさい。

■解答例

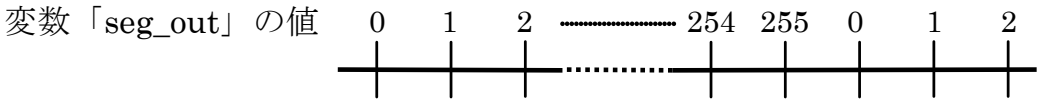
グローバル配列変数「seg_data」に、「a」～「f」のデータを追加します。「**太字**」を追加します。

```
040 const unsigned char seg_data[] = {
041 /*    0    1    2    3    4    5    6    7    8    9 */
042     0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f,    ←コンマを追加
043 /*    A    b    C    d    E    F                              */
044     0x77, 0x7c, 0x39, 0x5e, 0x79, 0x71
045 };
```

割り込みプログラムを「**太字**」に修正します。

```
121     // 7セグメントLEDの表示値の更新
122     cnt_7seg++;
123     if( cnt_7seg >= 100 ) {
124         // 100msに1回実行
125         cnt_7seg = 0;
126         seg_out++;
127     }
128
129     // 7セグメントLED 表示処理
130     if( p0 & 0x01 ) {
131         // 10の桁(DIG1)表示
132         p0 &= 0xfe;
133         i = seg_out >> 4;
134         p2 = seg_data[ i ];
135         p0 |= 0x02;
136     } else {
137         // 1の桁(DIG2)表示
138         p0 &= 0xfd;
139         i = seg_out & 0x0f;
140         p2 = seg_data[ i ];
141         p0 |= 0x01;
142     }
```

122 行～127 行は、7 セグメント LED に表示する値がグローバル変数「seg_out」に格納されます。変数「seg_out」に格納された値を 0.1 秒間隔でカウン トアップします。0～255 までカウントアップすると自動的に変数「seg_out」の値は 0 に戻ります。変数「seg_out」は、「unsigned char」型の 1 バイトデータですので 255 の次は 0 になり、オーバーフローを起こします。



133 行は、変数「seg_out」に格納されている上位 4bit の値を取り出し、その値を 7 セグメント LED に出力します。
139 行は、変数「seg_out」に格納されている下位 4bit の値を取り出し、その値を 7 セグメント LED に出力します。

6. 外部割込み（プロジェクト：int_interrupt）

6.1. 演習（1）

■問題：$\overline{\text{INT3}}$端子に立ち上がりエッジの信号が入力されると、割り込みが発生するようにしなさい。

■解答例

$\overline{\text{INT3}}$の外部割込みレジスタを「**太字**」に修正します。

```
092     int0ic = 0x00;                  /* INT0割り込み優先レベル設定    */
093     int1ic = 0x00;                  /* INT1割り込み優先レベル設定    */
094     int2ic = 0x00;                  /* INT2割り込み優先レベル設定    */
095     int3ic = 0x17;                  /* INT3割り込み優先レベル設定    */
096     int4ic = 0x00;                  /* INT4割り込み優先レベル設定    */
```

$\overline{\text{INT3}}$端子の割り込みが発生するタイミングを「立ち下がりエッジ」から「立ち上がりエッジ」に変更しますので、「int3ic」レジスタの bit4 を"1"に設定します。設定値は「0x17」になります。

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「16.5.1(3)⑨ $\overline{\text{INT3}}$割り込み制御レジスタの設定」を参照してください。

6.2. 演習（2）

■問題：$\overline{\text{INT1}}$割り込みを P3_2 端子に設定して、立ち下がりエッジで割り込みがかかるようにしなさい。割り込みプログラムはサンプルプログラムと同じとする。
（実習基板 Ver.2 の SW11 は、左から「下下下下　下上下下」にしてください。）

■解答例

外部割込み端子を P3_3 から P3_2 に変更するため、P3_2 を入力設定にします。レジスタの値を「**太字**」に修正します。

```
070     /* ポートの入出力設定 */
071     prc2 = 1;                       /* PD0のプロテクト解除           */
072     pd0 = 0x00;                     /*                               */
073     pd1 = 0xd0;                     /* 5:RXD0 4:TXD0 3-0:DIP SW      */
074     pd2 = 0xff;                     /* 7-0:LED                       */
075     pd3 = 0xfb;                     /* 3:パルス入力                  */
076     p4  = 0x00;                     /* P4_5のLED:初期は消灯          */
077     pd4 = 0xb8;                     /* 7:XOUT 6:XIN 5:LED 2:VREF     */
```

外部割込みのレジスタの値を「**太字**」に修正します。

```
085     /* INT0～4割り込み設定(今回はINT3を設定) */
086     intsr  = 0x08;                  /* INT1～3の入力端子設定          */
087     inten  = 0x04;                  /* INT0～3の外部入力許可設定      */
088     inten1 = 0x00;                  /* INT4の外部入力許可設定         */
089     intf   = 0x0c;                  /* INT0～3の入力フィルタ選択      */
090     intf1  = 0x00;                  /* INT4の入力フィルタ選択         */
091
092     int0ic = 0x00;                  /* INT0割り込み優先レベル設定     */
093     int1ic = 0x07;                  /* INT1割り込み優先レベル設定     */
094     int2ic = 0x00;                  /* INT2割り込み優先レベル設定     */
095     int3ic = 0x00;                  /* INT3割り込み優先レベル設定     */
096     int4ic = 0x00;                  /* INT4割り込み優先レベル設定     */
```

86 行は、INT 割り込み入力選択レジスタ「intsr」で $\overline{\text{INT1}}$の入力端子を P3_2 に割り当てます。
87 行は、$\overline{\text{INT1}}$入力極性選択ビットは片側エッジを選択し、$\overline{\text{INT1}}$入力を許可に設定します。
89 行は、$\overline{\text{INT1}}$の入力フィルタを「あり」に設定します。
93 行の $\overline{\text{INT1}}$割り込みの設定は、立ち下がりエッジを選択し、割り込み優先レベルを 7 に設定します。
95 行は、先程まで使用していた $\overline{\text{INT3}}$割り込みを禁止にします。

外部割込みの関数を「**太字**」のように修正します。

```
125 /***********************************************************************/
126 /* INT3 割り込み処理                                                   */
127 /***********************************************************************/
128 #pragma interrupt intINT1(vect=25)
129 void intINT1( void )
130 {
131     cnt_int3++;
132     p2 = cnt_int3;
133 }
```

外部割込みが $\overline{\text{INT3}}$から $\overline{\text{INT1}}$に変更になったため、ベクタ番号を修正します。割り込み関数名は修正しなくても良いのですが、何の割り込み関数なのか分からなくなってしまうため関数名も修正します。

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「16.5.1(3) $\overline{\text{INT3}}$割り込みの設定」を参照してください。

7. A/D コンバータ(単発モード)(プロジェクト:ad)

7.1. 演習(1)

■問題:ポート 6 に実習基板 Ver.2 の LED 部を接続して、A/D 変換値の上位 8bit をその LED へ出力しなさい。

■解答例

A/D 変換値の出力先をポート 6 に変更します。メインプログラムを「**太字**」に修正します。

```
040     while( 1 ) {
041         ad = get_ad7();
042         ad = ad >> 2;
043         p6 = ad;
044     }
```

7.2. 演習(2)

■問題:(1)の状態で、アナログ入力端子を P0_1 端子に変更して、A/D 変換した値を LED へ出力しなさい。

■解答例　**※(1)の解答例の状態から説明します。**

アナログ入力端子 P0_1 の A/D 変換値を読み込むための関数がないため、関数のプロトタイプ宣言をします。「int get_ad7(void)」は使用しないため、「int get_ad7(void)」を「**太字**」に修正します。

```
025 /*======================================*/
026 /* プロトタイプ宣言                     */
027 /*======================================*/
028 void init( void );
029 int get_ad6( void );
```

メインプログラムを「**太字**」に修正します。

```
040     while( 1 ) {
041         ad = get_ad6();
042         ad = ad >> 2;
043         p6 = ad;
044     }
```

「get_ad7」関数は使いませんので、「get_ad7」関数のプログラムを少し修正して「get_ad6」関数を作成します。「**太字**」に修正します。

```
078 /************************************************************************/
079 /* A/D値読み込み(AN7)                                                   */
080 /* 引数    なし                                                         */
081 /* 戻り値 A/D値 0～1023                                                 */
082 /************************************************************************/
083 int get_ad6( void )
084 {
085     int i;
086
087     /* A/Dコンバータの設定 */
088     admod   = 0x03;                 /* 単発モードに設定                 */
089     adinsel = 0x06;                 /* 入力端子AN7(P0_0)を選択          */
090     adcon1  = 0x30;                 /* A/D動作可能                      */
091     asm(" nop ");                   /* φADの1サイクルウエイト入れる*/
092     adcon0  = 0x01;                 /* A/D変換スタート                  */
093
094     while( adcon0 & 0x01 );         /* A/D変換終了待ち                  */
095
096     i = ad6;
097
098     return i;
099 }
```

83 行は、関数名なので変更の必要はありませんが、どのアナログ入力端子を読み込んでいるか、わからないため関数名も修正します。

89 行は、AN7(P0_0)端子から AN6(P0_1)端子に変更します。

96 行は、アナログ入力端子を変更すると変換結果を読み込むレジスタが変わります。「ad7」から「ad6」に修正します。

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「17.5.2(3) A/D コンバータの設定」を参照してください。

8. A/D コンバータ(繰り返しモード 0)(プロジェクト:ad_kurikaeshi)

8.1. 演習(1)

■問題:ポート 6 に LED 基板(実習基板 Ver.2 の LED 部など)を接続して、A/D 変換値の上位 8bit をその LED へ出力しなさい。

■解答例

A/D 変換値の出力先をポート 6 に変更します。メインプログラムを「**太字**」に修正します。

```
040     while( 1 ) {
041         ad = get_ad7();
042         ad = ad >> 2;
043         p6 = ad;
044     }
```

8.2. 演習(2)

■問題:(1)の状態で、アナログ入力端子を P0_1 端子に変更して、LED へ出力しなさい。

■解答例　**※(1)の解答例の状態から説明します。**

アナログ入力端子 P0_1 の A/D 変換値を読み込む関数がないため、関数のプロトタイプ宣言をします。「int get_ad7(void)」は使用しないため、「int get_ad7(void)」を「**太字**」に修正します。

```
025 /*====================================*/
026 /* プロトタイプ宣言                   */
027 /*====================================*/
028 void init( void );
029 int get_ad6( void );
```

メインプログラムを「**太字**」に修正します。

```
040     while( 1 ) {
041         ad = get_ad6();
042         ad = ad >> 2;
043         p6 = ad;
044     }
```

A/D 変換のレジスタの値を「**太字**」に修正します。

```
077     /* A/Dコンバータの設定 */
078     admod   = 0x13;                 /* 繰り返しモード0に設定          */
079     adinsel = 0x06;                 /* 入力端子AN7(P0_0)を選択        */
080     adcon1  = 0x30;                 /* A/D動作可能                    */
081     asm(" nop ");                   /* φADの1サイクルウエイト入れる*/
082     adcon0  = 0x01;                 /* A/D変換スタート                */
```

79 行は、入力端子を AN7(P0_0)から AN6(P0_1)に変更します。

「get_ad7」関数は使いませんので、「get_ad7」関数のプログラムを少し修正して「get_ad6」関数を作成します。「**太字**」に修正します。

```
085 /************************************************************************/
086 /* A/D値読み込み(AN7)                                                   */
087 /* 引数    なし                                                         */
088 /* 戻り値 A/D値 0～1023                                                 */
089 /************************************************************************/
090 int get_ad6( void )
091 {
092     int i;
093
094     /* 繰り返しモード0は、自動的に繰り返すので、結果を読み込むだけ */
095     i = ad6;
096
097     return i;
098 }
```

90 行は、関数名なので変更の必要はありませんが、どのアナログ入力端子を読み込んでいるか、わからないため関数名も修正します。

96 行は、アナログ入力端子を変更すると変換結果を読み込むレジスタが変わります。「ad7」から「ad6」に修正します。

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「18.5.2 init 関数(A/D コンバータの設定)」を参照してください。

9. A/D コンバータ(繰り返し掃引モード)(プロジェクト:ad_kurikaeshi_souin)

9.1. 演習(1)

■問題:RY_R8C38 ボードのディップスイッチの値 2～7 によって、AN0～5 端子から電圧を取り込むようにしなさい。このとき、取り込んだ A/D 変換値を実習基板 Ver.2 の LED へ出力するようにしなさい。

※プログラム実行時、A/D 変換する端子に合わせて、実習基板 Ver.2 のボリュームの端子切り替えスイッチ SW12 の設定も替えてください。例えば、AN5 なら SW12 の 2 を ON に、AN0 なら SW12 の 7 を ON にしてください。

■解答例

メインプログラムを「**太字**」のように修正します。

```
042     while( 1 ) {
043         switch( dipsw_get() ) {
044         case 2:
045             ad = ad5;                   /* AD5(P0_2)取得              */
046             ad = ad >> 2;
047             p2 = ad;
048             break;
049         case 3:
050             ad = ad4;                   /* AD4(P0_3)取得              */
051             ad = ad >> 2;
052             p2 = ad;
053             break;
054         case 4:
055             ad = ad3;                   /* AD3(P0_4)取得              */
056             ad = ad >> 2;
057             p2 = ad;
058             break;
059         case 5:
060             ad = ad2;                   /* AD2(P0_5)取得              */
061             ad = ad >> 2;
062             p2 = ad;
063             break;
064         case 6:
065             ad = ad1;                   /* AD1(P0_6)取得              */
066             ad = ad >> 2;
067             p2 = ad;
068             break;
069         case 7:
070             ad = ad0;                   /* AD0(P0_7)取得              */
071             ad = ad >> 2;
072             p2 = ad;
073             break;
074         }
075     }
```

RY_R8C38 ボードのディップスイッチの値によって、AN5(P0_2)～AN0(P0_7)のどのアナログ入力端子の変換結果をポート 2 へ出力するか選択するプログラムです。

アナログ入力端子は、AN5(P0_2)～AN0(P0_7)端子です。P0_1～P0_0 は使用しないため、出力設定にします。ポートの入出力設定を「**太字**」に修正します。

```
093     /* ポートの入出力設定 */
094     prc2 = 1;                       /* PD0のプロテクト解除          */
095     pd0 = 0x03;                     /* 7-0:ボリュームなどのｱﾅﾛｸﾞ電圧*/
096     pd1 = 0xd0;                     /* 5:RXD0 4:TXD0 3-0:DIP SW     */
```

AN7(P0_0)～AN0(P0_7)の 8 端子がアナログ入力端子に設定されていますので、AN5(P0_2)～AN0(P0_7)の 6 端子をアナログ入力端子に設定します。レジスタの設定を「**太字**」に修正します。

```
108     /* A/Dコンバータの設定 */
109     admod   = 0x33;                 /* 繰り返し掃引モードに設定     */
110     adinsel = 0x20;                 /* 入力端子P0の8端子を選択      */
111     adcon1  = 0x30;                 /* A/D動作可能                  */
112     asm(" nop ");                   /* φADの1サイクルウエイト入れる*/
113     adcon0  = 0x01;                 /* A/D変換スタート              */
```

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「19.5.2 init 関数(A/D コンバータの設定)」を参照してください。

9.2. 演習(2)

■問題:(1)の状態から A/D 変換の結果を 7 セグメント LED へ出力するようにしなさい。7 セグメント LED は、16 進数で表示するようにしなさい。

■解答例　**※(1)の解答例の状態から説明します。**

7 セグメント LED に出力するデータを格納する変数と 7 セグメント LED に表示する値を格納する変数を宣言します。グローバル変数の宣言がないため、「**太字**」を追加します。

```
033 /*====================================*/
034 /* グローバル変数の宣言               */
035 /*====================================*/
036 unsigned char seg_out;               /* 7セグメントLEDに表示する値   */
037
038 const unsigned char seg_data[] = {
039 /*    0    1    2    3    4    5    6    7    8    9 */
040     0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f,
041 /*    A    b    C    d    E    F                        */
042     0x77, 0x7c, 0x39, 0x5e, 0x79, 0x71
043 };
```

メインプログラムを「**太字**」に修正します。

```
052     init();                             /* 初期化                       */
053     asm(" fset I ");                    /* 全体の割り込み許可           */
054
055     while( 1 ) {
056         switch( dipsw_get() ) {
057         case 2:
058             ad = ad5;                   /* AD5(P0_2)取得                */
059             ad = ad >> 2;
060             seg_out = ad;
061             break;
062         case 3:
063             ad = ad4;                   /* AD4(P0_3)取得                */
064             ad = ad >> 2;
065             seg_out = ad;
066             break;
067         case 4:
068             ad = ad3;                   /* AD3(P0_4)取得                */
069             ad = ad >> 2;
070             seg_out = ad;
071             break;
072         case 5:
073             ad = ad2;                   /* AD2(P0_5)取得                */
074             ad = ad >> 2;
075             seg_out = ad;
076             break;
077         case 6:
078             ad = ad1;                   /* AD1(P0_6)取得                */
079             ad = ad >> 2;
080             seg_out = ad;
081             break;
082         case 7:
083             ad = ad0;                   /* AD0(P0_7)取得                */
084             ad = ad >> 2;
085             seg_out = ad;
086             break;
087         }
088     }
```

53 行は、割り込み関数を使用するため、全体割り込み許可をします。

60、65、70、75、80、85 行は、AN5(P0_2)～AN0(P0_7)の A/D 変換結果を1バイト(10 進数で 0～255)データに変換し、グローバル変数「seg_out」に格納します。

1msの割り込みをタイマ RB で作ります。init 関数内に「**太字**」を追加します。

```
128     /* タイマRBの設定 */
129     /* 割り込み周期 = 1 / 20[MHz]     * (TRBPRE+1) * (TRBPR+1)
130                     = 1 / (20*10^-6) * 200         * 100
131                     = 0.001[s] = 1[ms]
132     */
133     trbmr  = 0x00;                      /* 動作モード、分周比設定       */
134     trbpre = 200-1;                     /* プリスケーラレジスタ         */
135     trbpr  = 100-1;                     /* プライマリレジスタ           */
136     trbic  = 0x07;                      /* 割り込み優先レベル設定       */
137     trbcr  = 0x01;                      /* カウント開始                 */
138 }
```

※タイマ RB の設定については、「マイコン実習マニュアル R8C/38A 版」の「14.5.2 init 関数(タイマ RB の設定)」を参照してください。

割り込み関数を作成します。「**太字**」を追加します。

```
/************************************************************************/
/* タイマRB 割り込み処理                                                */
/************************************************************************/
#pragma interrupt intTRB(vect=24)
void intTRB( void )
{
    int i;

    // 7セグメントLED 表示処理
    if( p0 & 0x01 ) {
        // 10の桁(DIG1)表示
        p0 &= 0xfe;
        i = seg_out >> 4;
        p2 = seg_data[ i ];
        p0 |= 0x02;
    } else {
        // 1の桁(DIG2)表示
        p0 &= 0xfd;
        i = seg_out & 0x0f;
        p2 = seg_data[ i ];
        p0 |= 0x01;
    }
}
```

※割り込み関数については、本マニュアルの「4.2　演習(2)解答例」と同じプログラムを使用していますので、詳しくは、こちらを参照してください。

10. タイマ RD による PWM 波形出力(リセット同期 PWM モード)(プロジェクト:timer_rd_doukipwm)

10.1. 演習(1)

■問題:周期が出力される端子(P2_1)の PWM 出力を禁止して、通常の I/O ポートにしなさい。このとき、この端子は出力端子として"0"を出力しなさい。

■解答例

TRDIOC0(P2_1)端子の PWM 出力を禁止し、ポートの割り当てを TRDIOC0 端子は「**使用しない**」に設定します。「**太字**」に修正します。

```
072     /* タイマRD リセット同期PWMモードの設定*/
073     trdfcr  = 0x01;                 /* リセット同期PWMモードに設定  */
074     trdmr   = 0xf0;                 /* バッファレジスタ設定         */
075     trdoer1 = 0x05;                 /* 出力端子の選択               */
076     trdpsr0 = 0x48;                 /* TRDIOB0,C0,D0端子設定        */
077     trdpsr1 = 0x55;                 /* TRDIOA1,B1,C1,D1端子設定     */
078     trdcr0  = 0x23;                 /* ソースカウントの選択:f8      */
```

10.2. 演習(2)

■問題:逆相の端子(P2_3、P2_6、P2_7 の 3 つとも)の PWM 出力を禁止して、通常の I/O ポートにしなさい。このとき、この端子は出力端子として"0"を出力しなさい。
※プログラムは、(1)の状態から改造するものとする。

■解答例　**※(1)の解答例の状態から説明します。**

TRDIOD0(P2_3)端子、TRDIOC1(P2_6)端子、TRDIOD1(P2_7)端子の出力を禁止し、ポートの割り当てを TRDIOD0、TRDIOC1、TRDIOD1 端子は「**使用しない**」に設定します。「**太字**」に修正します。

```
072     /* タイマRD リセット同期PWMモードの設定*/
073     trdfcr  = 0x01;                 /* リセット同期PWMモードに設定  */
074     trdmr   = 0xf0;                 /* バッファレジスタ設定         */
075     trdoer1 = 0xcc;                 /* 出力端子の選択               */
076     trdpsr0 = 0x08;                 /* TRDIOB0,C0,D0端子設定        */
077     trdpsr1 = 0x05;                 /* TRDIOA1,B1,C1,D1端子設定     */
078     trdcr0  = 0x23;                 /* ソースカウントの選択:f8      */
```

10.3. 演習(3)

■問題:P2_4 端子も P2_2 端子と同様に、ディップスイッチに合わせて PWM 信号が出力されるようにしなさい。出力の仕方は、サンプルプログラム 38 行目の「39998 * dipsw_get() / 15」を使用することとする。
※プログラムは、(2)の状態から改造するものとする。

■解答例　**※(2)の解答例の状態から説明します。**

P2_4 端子のデューティ比の設定は、TRDGRA1 レジスタです。そのバッファレジスタである TRDGRC1 レジスタに値を設定します。メインプログラムに「**太字**」を追加します。

```
037     while( 1 ) {
038         trdgrd0 = 39998 * dipsw_get() / 15;
039         trdgrc1 = 39998 * dipsw_get() / 15;
040     }
```

10.4. 演習(4)

■問題:P2_5 端子も P2_2 端子と同様に、ディップスイッチに合わせて PWM 信号が出力されるようにしなさい。出力の仕方は、サンプルプログラム 38 行目の「39998 * dipsw_get() / 15」を使用することとする。
※プログラムは、(3)の状態から改造するものとする。

■解答例　**※(3)の解答例の状態から説明します。**

P2_5 端子のデューティ比の設定は、TRDGRB1 レジスタです。そのバッファレジスタである TRDGRD1 レジスタに値を設定します。メインプログラムに「**太字**」を追加します。

```
037     while( 1 ) {
038         trdgrd0 = 39998 * dipsw_get() / 15;
039         trdgrc1 = 39998 * dipsw_get() / 15;
040         trdgrd1 = 39998 * dipsw_get() / 15;
041     }
```

10.5. 演習(5)

■問題:P2_4 端子、P2_5 端子の PWM 出力を禁止して、通常の I/O ポートにしなさい。このとき、この端子は出力端子として"0"を出力しなさい。
※プログラムは、(4)の状態から改造するものとする。

■解答例　**※(4)の解答例の状態から説明します。**

P2_4 端子、P2_5 端子は PWM の出力を禁止にしますので、「**太字**」の 2 行は不要となります。「**太字**」の 2 行を削除します。

```
037     while( 1 ) {
038         trdgrd0 = 39998 * dipsw_get() / 15;
            ~~trdgro1 = 39998 * dipsw_get() / 15;~~     ←行を削除
            ~~trdgrd1 = 39998 * dipsw_get() / 15;~~     ←行を削除
039     }
```

TRDIOA1(P2_4)端子、TRDIOB1(P2_5)端子の出力を禁止し、ポートの割り当てを TRDIOA1 端子、TRDIOB1 端子は「**使用しない**」に設定します。「**太字**」に修正します。

```
072     /* タイマRD リセット同期PWMモードの設定*/
073     trdfcr  = 0x01;                 /* リセット同期PWMモードに設定   */
074     trdmr   = 0xf0;                 /* バッファレジスタ設定          */
075     trdoer1 = 0xfc;                 /* 出力端子の選択                */
076     trdpsr0 = 0x08;                 /* TRDIOB0,C0,D0端子設定         */
077     trdpsr1 = 0x00;                 /* TRDIOA1,B1,C1,D1端子設定      */
078     trdcr0  = 0x23;                 /* ソースカウントの選択:f8       */
```

※詳しくは、「マイコン実習マニュアル R8C/38A 版」の「22.6.1(3) ③～⑤」を参照してください。

11. タイマ RD による PWM 波形出力(PWM モード)(プロジェクト:timer_rd_pwm6)

11.1. 演習(1)

■問題:問題点を解決したプログラムに改造しなさい。

■解答例

メインプログラムに「**太字**」を追加します。

```
041     while( 1 ) {
042         if( dipsw_get() == 0 ) {
043             // trdgra0,trdgra1と同じにすると0%出力
044             pwm = 39999;
045         } else if( dipsw_get() == 15 ) {
046             // trdgra0,trdgra1より大きくすると100%出力
047             pwm = 40000;
048         } else {
049             // 1～14なら割合に応じてPWM出力する
050             pwm = (long)39999 * dipsw_get() / 15;
051         }
052
053         // trdgrb0を設定するとき
054         while( trd0 <= trdgrb0 && trdgra0 > trdgrb0 );
055         trdgrb0  = pwm;                 /* P2_2のON幅設定              */
056         // trdgrc0を設定するとき
057         while( trd0 <= trdgrc0 && trdgra0 > trdgrc0 );
058         trdgrc0  = pwm;                 /* P2_1のON幅設定              */
059         // trdgrd0を設定するとき
060         while( trd0 <= trdgrd0 && trdgra0 > trdgrd0 );
061         trdgrd0  = pwm;                 /* P2_3のON幅設定              */
062         // trdgrb1を設定するとき
063         while( trd1 <= trdgrb1 && trdgra1 > trdgrb1 );
064         trdgrb1  = pwm;                 /* P2_5のON幅設定              */
065         // trdgrc1を設定するとき
066         while( trd1 <= trdgrc1 && trdgra1 > trdgrc1 );
067         trdgrc1  = pwm;                 /* P2_6のON幅設定              */
068         // trdgrb1を設定するとき
069         while( trd1 <= trdgrd1 && trdgra1 > trdgrd1 );
070         trdgrd1  = pwm;                 /* P2_7のON幅設定              */
071     }
```

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「23.5.2 main 関数」の「※問題点」を参照してください。

11.2. 演習(2)

■問題:TRDGRC0(P2_1)=0%、TRDGRB0(P2_2)=20%、TRDGRD0(P2_3)=40%、TRDGRB1(P2_5)=60%、
RDGRC1(P2_6)=80%、TRDGRD1(P2_7)=100%を初期 PWM 値として、1 秒ごとに 20%ずつ PWM 値を増やすプログラムを作成しなさい。ただし、100%の次は 0%とする。

■解答例

1 秒の時間をつくるタイマ関数とデューティ比をジェネラルレジスタに設定する値に変換する PWM 関数を作成します。関数のプロトタイプ宣言とタイマ関数用のグローバル変数を追加します。「**太字**」を追加します。

```
026 /*====================================*/
027 /* プロトタイプ宣言                   */
028 /*====================================*/
029 void init( void );
030 unsigned char dipsw_get( void );
031 void timer( unsigned long timer_set );
032 unsigned int PWM( int pwm );
033
034 /*====================================*/
035 /* グローバル変数の宣言               */
036 /*====================================*/
037 unsigned long cnt_rb;                  /* タイマRB用                  */
```

メインプログラムを「**太字**」に修正します。

```
044     int    pwm[6] = {0,20,40,60,80,100};
045     int    i;
046
047     init();                            /* 初期化                      */
048     asm(" fset I ");                   /* 全体の割り込み許可          */
049
050     while( 1 ) {
051         // trdgrb0を設定するとき
052         while( trd0 <= trdgrb0 && trdgra0 > trdgrb0 );
053         trdgrb0  = PWM( pwm[1] );           /* P2_2のON幅設定              */
054         // trdgrc0を設定するとき
055         while( trd0 <= trdgrc0 && trdgra0 > trdgrc0 );
056         trdgrc0  = PWM( pwm[0] );          /* P2_1のON幅設定              */
057         // trdgrd0を設定するとき
058         while( trd0 <= trdgrd0 && trdgra0 > trdgrd0 );
059         trdgrd0  = PWM( pwm[2] );          /* P2_3のON幅設定              */
060         // trdgrb1を設定するとき
061         while( trd1 <= trdgrb1 && trdgra1 > trdgrb1 );
062         trdgrb1  = PWM( pwm[3] );          /* P2_5のON幅設定              */
063         // trdgrc1を設定するとき
064         while( trd1 <= trdgrc1 && trdgra1 > trdgrc1 );
065         trdgrc1  = PWM( pwm[4] );          /* P2_6のON幅設定              */
066         // trdgrb1を設定するとき
067         while( trd1 <= trdgrd1 && trdgra1 > trdgrd1 );
068         trdgrd1  = PWM( pwm[5] );          /* P2_7のON幅設定              */
069
070         timer(1000);
071         for( i = 0; i < 6; i++ ){
072             pwm[i] += 20;
073             if( pwm[i] > 100 )pwm[i] = 0;
074         }
075     }
```

44 行の変数「pwm[6]」は、PWM を出力する 6 本のデューティ比の初期値です。20%ずつ変化させる変数になります。

45 行の変数「i」は、71～74 行の for 文で使用します。

48 行は、タイマ RB による割り込みを発生させるため、マイコン全体の割り込みを許可します。

53、56、59、62、65、68 行は、変数「pwm[x]」(xは、0～5 が入ります。)に格納されているデューティ比を PWM 関数の引数に代入し、TRD のジェネラルレジスタに設定する値に変換し、それぞれのジェネラルレジスタに値を設定します。

70 行は、timer 関数で 1 秒のウエイトを入れます。

71～74 行は、6 本の PWM 出力端子に出力するデューティ比を 20%ずつ増やし、100%の次は 0%にする処理をします。

1msの割り込みをタイマ RB で作ります。「**太字**」を追加します。

```
106     pd9 = 0x3f;                         /*                          */
107     pur0 = 0x04;                        /* P1_3～P1_0のプルアップON    */
108
109     /* タイマRBの設定 */
110     /* 割り込み周期 = 1 / 20[MHz]    * (TRBPRE+1) * (TRBPR+1)
111                     = 1 / (20*10^-6) * 200        * 100
112                     = 0.001[s] = 1[ms]
113     */
114     trbmr  = 0x00;                      /* 動作モード、分周比設定      */
115     trbpre = 200-1;                     /* プリスケーラレジスタ        */
116     trbpr  = 100-1;                     /* プライマリレジスタ          */
117     trbic  = 0x07;                      /* 割り込み優先レベル設定      */
118     trbcr  = 0x01;                      /* カウント開始                */
119
120     /* タイマRD PWMモードの設定(6本のPWM出力) */
121     trdpmr   = 0x77;                    /* 各端子PWMﾓｰﾄﾞにするかしないか*/
122     trdfcr   = 0x80;                    /* PWMモードに設定             */
```

※タイマ RB の設定については、「マイコン実習マニュアル R8C/38A 版」の「14.5.2 init 関数(タイマ RB の設定)」を参照してください。

割り込み関数とタイマ関数を作成します。「**太字**」を追加します。

```
142 /************************************************************************/
143 /* タイマRB 割り込み処理                                                */
144 /************************************************************************/
145 #pragma interrupt intTRB(vect=24)
146 void intTRB( void )
147 {
148     cnt_rb++;
149 }
150
151 /************************************************************************/
152 /* タイマ本体                                                           */
153 /* 引数    タイマ値 1=1ms                                               */
154 /************************************************************************/
155 void timer( unsigned long timer_set )
156 {
157     cnt_rb = 0;
158     while( cnt_rb < timer_set );
159 }
```

デューティ比をジェネラルレジスタに設定する値に、変換する PWM 関数を追加します。「**太字**」を追加します。

```
174 /************************************************************************/
175 /* デューティの計算                                                     */
176 /*                                                                      */
177 /************************************************************************/
178 unsigned int PWM( int pwm )
179 {
180     unsigned int pwm_out;
181
182     if( pwm <= 0 ) {
183         // trdgra0,trdgra1と同じにすると0%出力
184         pwm_out = 39999;
185     } else if( pwm >= 100 ) {
186         // trdgra0,trdgra1より大きくすると100%出力
187         pwm_out = 40000;
188     } else {
189         // 1～14なら割合に応じてPWM出力する
190         pwm_out = (long)39999 * pwm / 100;
191     }
192
193     return pwm_out;
194 }
```

※詳しくは、「マイコン実習マニュアル　R8C/38A 版」の「23.5.2 main 関数」を参照してください。

11.3. 演習(3)

■問題:(2)の初期 PWM 値から初めて、1 秒ごとに 10%ずつ PWM 値を減らすプログラムを作成しなさい。ただし、0%の次は 100%とする。

■解答例　**※(2)の解答例の状態から説明します。**

メインプログラムを「**太字**」に修正します。

```
070         timer(1000);
071         for( i = 0; i < 6; i++ ){
072             pwm[i] -= 10;
073             if( pwm[i] < 0 )pwm[i] = 100;
074         }
```

72 行は、6 本の PWM のデューティ比を 10%ずつ減らしていきます。
73 行は、デューティ比の値が 0%の次は、100%になるようにします。