9. I/Oポートの入出力（プロジェクト：io）

9.1 概要

本章では、マイコンのクロックの切り替え、I/O ポートの入出力設定、データの出力、データの入力方法を説明します。

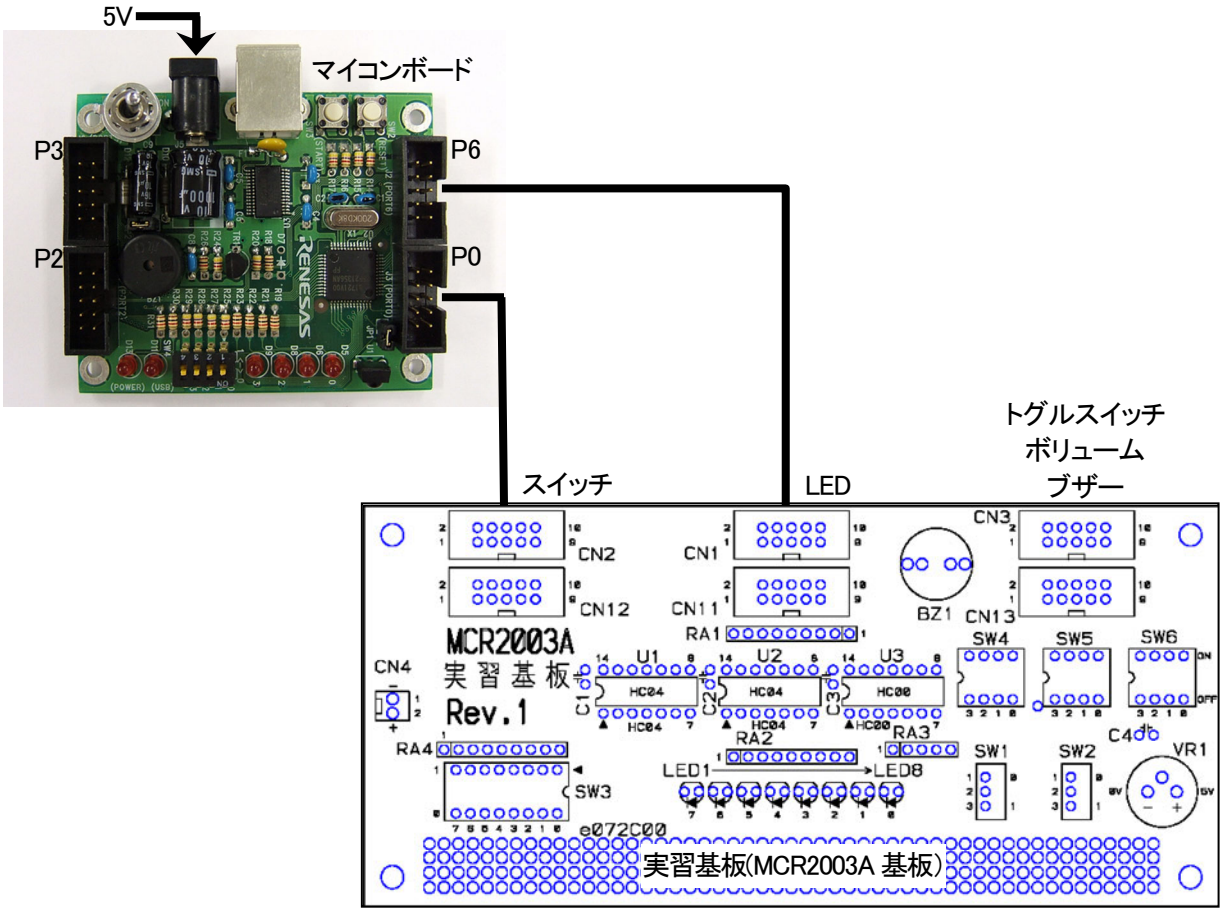
9.2 接続

■使用ポート

マイコンのポート	接続内容
P0 (J3)	実習基板のディップスイッチ部など、入力機器を接続します。マイコンカーラリーのセンサ基板 Ver.4 も可能です。
P6 (J2)	実習基板の LED 部など、出力機器を接続します。

■接続例

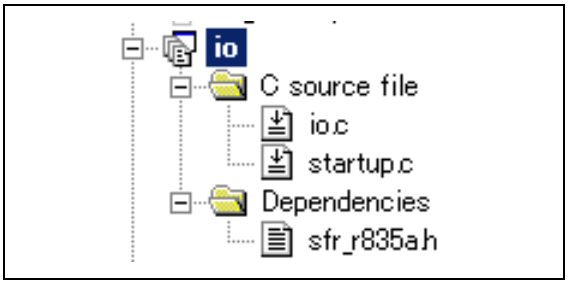
実習基板を使ったときの接続例を次に示します。



■操作方法

実習基板のディップスイッチ(SW3)を ON/OFF すると、それに合わせて LED(LED1～8)が ON/OFF します。入力したデータを出力する、制御の基本中の基本です。

9.3 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	io.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。

9.4 プログラム「io.c」

```
 1 :  /***********************************************************************/
 2 :  /* 対象マイコン  R8C/35A                                               */
 3 :  /* ﾌｧｲﾙ内容      データの入力、出力                                    */
 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 :  入力：P0_7-P0_0(ディップスイッチなど)
11 :  出力：P6_7-P6_0(LEDなど)
12 :
13 :  ポート0に入力した状態を、ポート6に出力します。
14 :  ポート0にセンサ基板、ポート6に実習基板のLED部分を接続すれば
15 :  センサ基板のチェックになります。
16 :  */
17 :
18 :  /*=====================================*/
19 :  /* インクルード                        */
20 :  /*=====================================*/
21 :  #include "sfr_r835a.h"                 /* R8C/35A SFRの定義ファイル     */
22 :
23 :  /*=====================================*/
24 :  /* シンボル定義                        */
25 :  /*=====================================*/
26 :
27 :  /*=====================================*/
28 :  /* プロトタイプ宣言                    */
29 :  /*=====================================*/
30 :  void init( void );
31 :
```

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

    init();                             /* 初期化                       */

    while( 1 ) {
        d = p0;
        p6 = d;
    }
}

/************************************************************************/
/* 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 = 0x00;                         /* スイッチなど入力             */
    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 = 0x80;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
    pd5 = 0x40;                         /* 7:DIP SW                     */
    pd6 = 0xff;                         /* LEDなど出力                  */
}

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

9.5 プログラムの解説

9.5.1 ヘッダファイルの取り込み

```
21 :  #include "sfr_r835a.h"                    /* R8C/35A SFR の定義ファイル     */
```

「#include」はインクルードと読み、外部のファイルを取り込む命令です。取り込むファイルは「＜　＞」、または「”　”」で囲い、その中に記述します。違いを下表に示します。

＜　＞	通常の include フォルダから取り込むファイルを探します。 標準ライブラリとしてルネサス統合開発環境が用意しているファイルは、こちらを使います。 例えば、stdio.h , stdlib.h , math.h　などです。ちなみに、これらのファイルは、 C:¥Program Files¥Renesas¥Hew¥Tools¥Renesas¥nc30wa¥v545r00¥inc30 フォルダにあります。波線部分は、ルネサス統合開発環境のバージョンにより異なります。
“　”	まず、カレント・フォルダ（Cプログラムファイルがあるフォルダ）を探して、そこに無ければ通常の include フォルダから取り込むファイルを探します。自分で作ったファイルは、こちらを使います。

ここでは、「sfr_r835a.h」を取り込んでいます。このヘッダファイルは、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。ちなみにこのファイルは、次のフォルダにあります。

C:¥Workspace¥common_r8c35a

9.5.2 init関数(クロックの選択)

init 関数で、R8C/35A マイコンに内蔵されている機能の初期化を行います。「init」とは、「initialize(イニシャライズ)」の略で、初期化の意味です。io.c では、init 関数内でクロックの選択とI/Oポートの入出力設定を行います。ここでは、クロックの選択について説明します。

```
 50 :  void init( void )
 51 :  {
 52 :      int i;
 53 :
 54 :      /* クロックをXINクロック(20MHz)に変更 */
 55 :      prc0  = 1;                    /* プロテクト解除                */
 56 :      cm13  = 1;                    /* P4_6,P4_7をXIN-XOUT端子にする*/
 57 :      cm05  = 0;                    /* XINクロック発振               */
 58 :      for(i=0; i<50; i++ );         /* 安定するまで少し待つ(約10ms) */
 59 :      ocd2  = 0;                    /* システムクロックをXINにする   */
 60 :      prc0  = 0;                    /* プロテクトON                  */
```

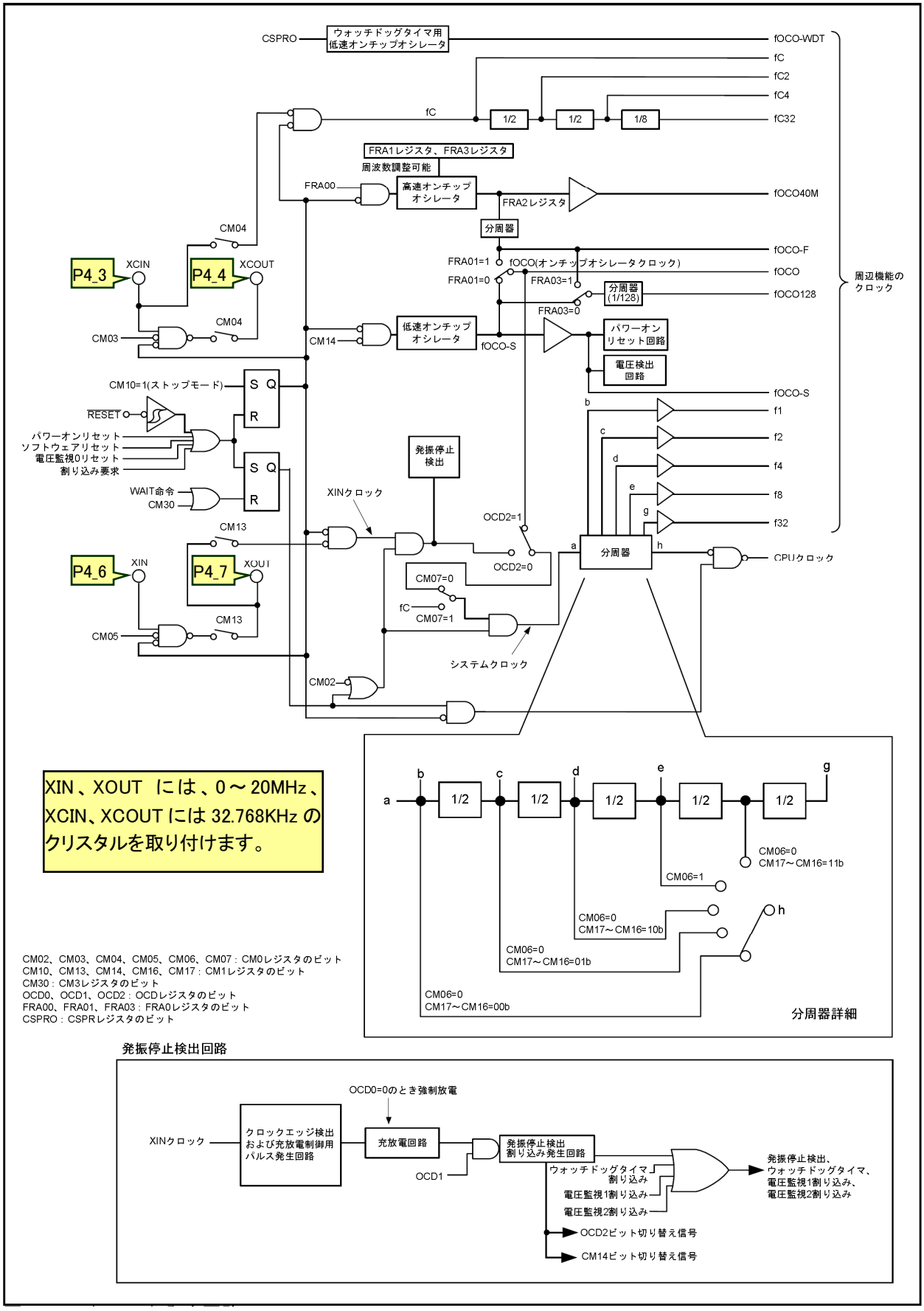
(1) R8C/35Aのクロック

R8C/35A は、2 種類のオシレータ(発振器)が内蔵されています。また、2 種類のクリスタルを外付けすることができます。どのクロックを使うかは、プログラムで設定します。下表にまとめます。

	XIN クロック 発信回路	XCIN クロック 発信回路	高速オンチップ オシレータ (内蔵クロック)	低速オンチップ オシレータ (内蔵クロック)	ウォッチドッグタイマ用 低速オンチップオシレータ (内蔵クロック)
接続端子	XIN(P4_6)、 XOUT(P4_7) に接続	XCIN(P4_3)、 XCOUT(P4_4) に接続	マイコン内蔵	マイコン内蔵	マイコン内蔵
クロック周波数	0～20MHz	32.768KHz	約 40MHz	約 125KHz	約 125KHz
リセット後の状態	停止	停止	停止	**発振**	停止、発振は ROM の特定の番地に書き込む値で決めます
備考	**ミニマイコンカー Ver.2 では 20MHz のクリスタルを実装しています。リセット後、プログラムでこのクロックに切り替えます。**	取り付けていません。	**R8C/35A には搭載されていません(R8C/35C には搭載されています)。**	リセット直後のみ使用します。すぐに、XIN に切り替えます。	使用しません。

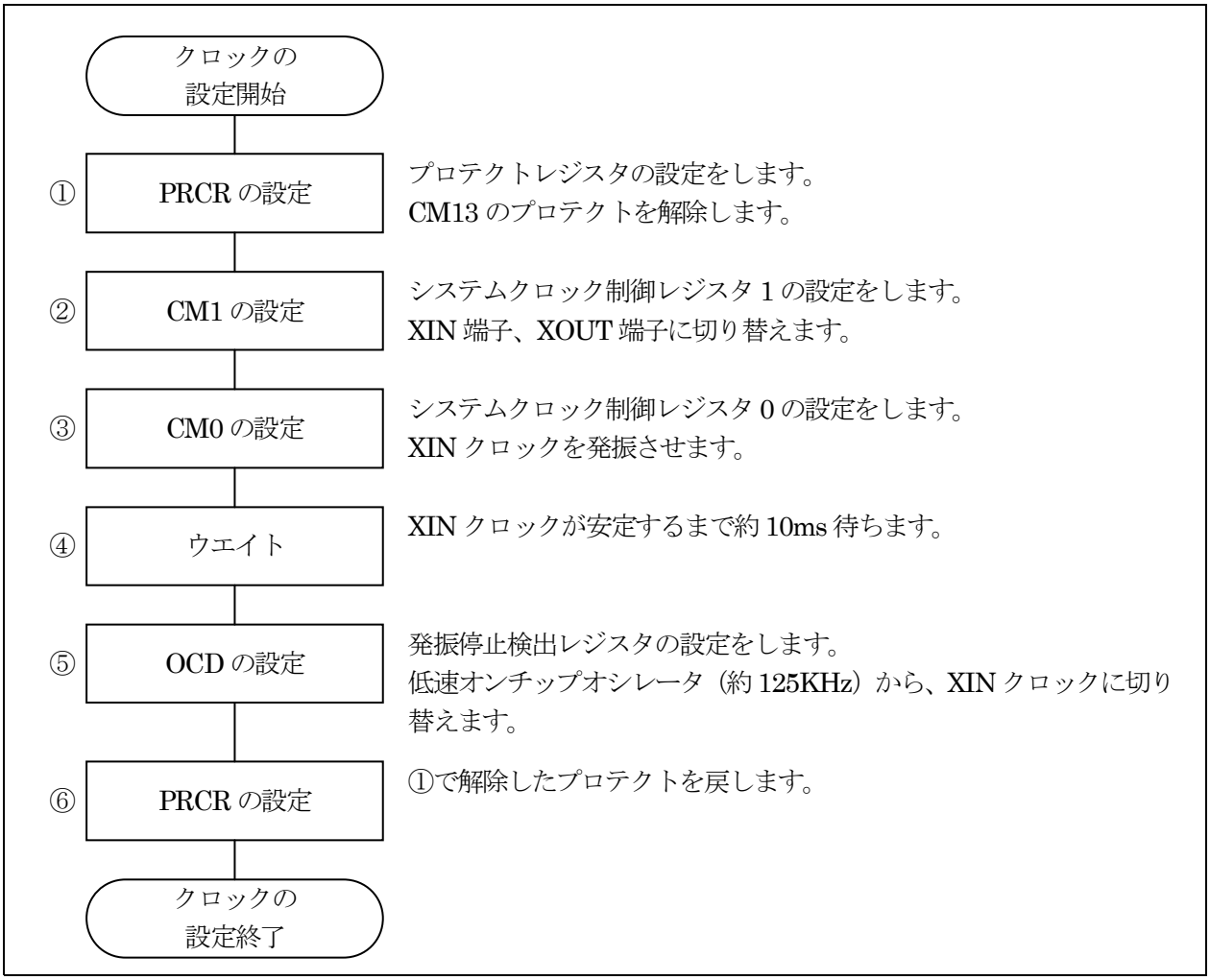
(2) ブロック図

R8C/35A のクロック発生回路を下記に示します。図の状態は、リセット直後の状態です。



(3) クロックの設定

R8C/35A マイコンは起動時、低速オンチップオシレータ(約 125KHz)で動作しています。init 関数内のプログラムで XIN クロック発振回路に切り替えて、20MHz で動作させます。レジスタの設定手順を下記に示します。



①プロテクトレジスタ(PRCR:Protect register)の設定(書き込み許可)

R8C/35A マイコンには、プログラムが暴走したときに備え重要なレジスタは簡単に書き替えられないように保護するレジスタがあります。プロテクトレジスタ(PRCR)は、プロテクトがかかっているレジスタの書き込みを禁止、または許可するレジスタです。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7～4		″0000″を設定	変更なし
bit3	プロテクトビット 3 prc3	OCVREFCR、VCA2、VD1LS、VW0C、VW1C、VW2C レジスタへの書き込み許可 0:書き込み禁止 1:書き込み許可	変更なし
bit2	プロテクトビット 2 prc2	PD0 レジスタへの書き込み許可 0:書き込み禁止 1:書き込み許可(注 1)	変更なし
bit1	プロテクトビット 1 prc1	PM0、PM1 レジスタへの書き込み許可 0:書き込み禁止 1:書き込み許可	変更なし
bit0	プロテクトビット 0 prc0	CM0、CM1、CM3、OCD、FRA0、FRA1、FRA2、FRA3 レジスタへの書き込み許可 0:書き込み禁止 1:書き込み許可	1

注 1. PRC2 ビットは“1”を書いた後、任意の番地に書き込みを実行すると、“0”になります。他のビットは“0”になりませんので、プログラムで“0”にしてください。

これからCM1、CM0、OCD の設定をします。これらのレジスタはプロテクトレジスタ(PRCR)で保護されており、値を変更するには保護を解除します。今回、これらのレジスタを書き込み許可するために bit0 を″1″にします。bit0 のみ OR 演算で″1″にします。下記にプログラムを示します。

```
prcr |= 0x01;
```

もう1つ、設定方法があります。上表の左から2列目に「下:シンボル」とありますが、各レジスタのビットに名前を付けています。プロテクトレジスタ(PRCR)の bit0 は、表より「**prc0**」という名前が付いています。よって下記のようにプログラムすることもできます。今回のプログラムでは、「prc0」として bit0 を″1″にしています。

```
55 :      prc0  = 1;                    /* プロテクト解除              */
```

②システムクロック制御レジスタ 1(CM1:System clock control register1)の設定

XIN 端子、XOUT 端子に切り替えます。CM1 レジスタは、PRCR レジスタの PRC0 ビットを“1”(書き込み許可)にした後で書き換えてください。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7,6	CPU クロック分周比選択ビット 1(注 1) bit7:cm17 bit6:cm16	00:分周なしモード 01:2 分周モード 10:4 分周モード 11:16 分周モード	変更なし
bit5		″1″を設定	変更なし
bit4	低速オンチップオシレータ発振停止ビット(注 3、4) cm14	0:低速オンチップオシレータ発振 1:低速オンチップオシレータ停止	変更なし
bit3	ポート XIN-XOUT 切り替えビット(注 5) cm13	0:入出力ポート P4_6、P4_7 1:XIN-XOUT 端子	1
bit2	XCIN-XCOUT 内蔵帰還抵抗選択ビット cm12	0:内蔵帰還抵抗有効 1:内蔵帰還抵抗無効	変更なし
bit1	XIN-XOUT 内蔵帰還抵抗選択ビット cm11	0:内蔵帰還抵抗有効 1:内蔵帰還抵抗無効	変更なし
bit0	全クロック停止制御ビット(注 2) cm10	0:クロック発振 1:全クロック停止(ストップモード)	変更なし

注 1. CM06 ビットが“0”(CM16、CM17 ビット有効)の場合、CM16～CM17 ビットは有効となります。
注 2. CM10 ビットが“1”(ストップモード)の場合、内蔵している帰還抵抗は無効となります。
注 3. CM14 ビットは OCD2 ビットが“0”(XIN クロック選択)のとき、“1”(低速オンチップオシレータ停止)にできます。OCD2 ビットを“1”(オンチップオシレータクロック選択) にすると、CM14 ビットは“0”(低速オンチップオシレータ発振)になります。“1”を書いても変化しません。
注 4. 電圧監視 1 割り込み、電圧監視 2 割り込みを使用する場合(デジタルフィルタを使用する場合)、CM14 ビットを“0”(低速オンチップオシレータ発振)にしてください。
注 5. CM13 ビットはプログラムで一度“1”にすると、“0”にはできません。

ミニマイコンカーVer.2 は、P4_6、P4_7 の端子に外付けクリスタル(20MHz)が接続されています。この端子を XIN 端子、XOUT 端子に切り替えます。CM1 の bit3 を″1″にします。今回は、ビット指定で CM13 を″1″にします。

```
56 :        cm13  = 1;                          /* P4_6,P4_7 を XIN-XOUT 端子にする*/
```

③システムクロック制御レジスタ 0(CM0:System clock control register0)の設定

XIN クロックを発振させます。CM0 レジスタは、PRCR レジスタの PRC0 ビットを“1”(書き込み許可)にした後で書き換えてください。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7	XIN、XCIN クロック選択ビット(注 7) cm07	0:XIN クロック 1:XCIN クロック	変更なし
bit6	CPU クロック分周比選択ビット 0(注 4) cm06	0:CM1 レジスタの CM16、CM17 ビット有効 1:8 分周モード	変更なし
bit5	XIN クロック(XIN-XOUT)停止ビット(注 1、3) cm05	0:発振 1:停止(注 2)	0
bit4	ポート、XCIN-XCOUT 切り替えビット(注 5) cm04	0:入出力ポート P4_3、P4_4 1:XCIN、XCOUT 端子(注 6)	変更なし
bit3	XCIN クロック停止ビット cm03	0:発振 1:停止	変更なし
bit2	ウェイトモード時周辺機能クロック停止ビット cm02	0:ウェイトモード時、周辺機能クロックを停止しない 1:ウェイトモード時、周辺機能クロックを停止する	変更なし
bit1,0		″00″を設定	変更なし

注 1. CM05 ビットは高速オンチップオシレータモード、低速オンチップオシレータモードにするとき XIN クロックを停止させるビットです。XIN クロックが停止したかどうかの検出には使えません。XIN クロックを停止させる場合、次のようにしてください。
(1) OCD レジスタの OCD1～OCD0 ビットを“00b”にする。
(2) OCD2 ビットを“1”(オンチップオシレータクロック選択)にする。

注 2. 外部クロック入力時には、クロック発振バッファだけ停止し、クロック入力は受け付けられます。

注 3. CM05 ビットが“1”(XIN クロック停止) かつ CM1 レジスタの CM13 ビットが“0”(P4_6、P4_7) の場合のみ、P4_6、P4_7 は入出力ポートとして使用できます。

注 4. ストップモードへの移行時、CM06 ビットは“1”(8 分周モード)になります。

注 5. CM04 ビットはプログラムで“1”にできますが、“0”にできません。

注 6. XCIN クロックを使用する場合、CM04 ビットを“1”、PINSR レジスタの XCSEL ビットを“1”にしてください。また、ポート P4_3、P4_4 は入力ポートで、プルアップなしにしてください。

注 7. CM04 ビットを“1”(XCIN-XCOUT 端子)にし、XCIN クロックの発振が安定した後に、CM07 ビットを“0”から“1”(XCIN クロック)にしてください。

XIN クロック(XIN-XOUT)停止ビット(CM05)を″0″にして、XIN クロックを発振させます。CM0 の bit5 を″0″にします。今回は、ビット指定で CM05 を″0″にします。

```
57 :        cm05   = 0;                          /* XIN クロック発振              */
```

④ウエイト

③で XIN クロックを発振させました。発振が安定するまで 10ms 程度、待ちます。
プログラムでは、for 文を使って、「50 回分何もせずに繰り返す」命令を実行します。実測で約 10ms です。

```
58 :      for(i=0; i<50; i++ );                  /* 安定するまで少し待つ(約 10ms) */
```

プログラムの実行時間は、机上での計算はできないので、実測する必要があります。
実測の方法は、プログラム実行前に空いているポートの端子を"1"にして、プログラム実行後にその端子を"0"にします。端子の"1"の時間を、オシロスコープなどで測定すると実行時間が分かります。
ポート 6 を使って実行時間を計測するプログラム例を、下記に示します。ポート 6 はすべて空き端子とします。

```
    p6  = 0x00;                         最初に実行 ポート 6 の値をすべて"0"にする
    pd6 = 0xff;                         最初に実行 ポート 6 の入出力設定  すべて出力

～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～

    p6  = 0xff;                         // ポート 6 の値をすべて"1"にする
    for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約 10ms) */
    p6  = 0x00;                         // ポート 6 の値をすべて"0"にする
```

⑤発振停止検出レジスタ(OCD:Oscillation stop detection register)の設定

システムクロックを低速オンチップオシレータ(約 125KHz)から、XIN クロックに切り替えます。OCD レジスタは、PRCR レジスタの PRC0 ビットを“1”(書き込み許可)にした後、書き換えてください。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7～4		″0000″を設定	変更なし
bit3	クロックモニタビット(注 4、5) ocd3	0:XIN クロック発振 1:XIN クロック停止	変更なし
bit2	システムクロック選択ビット(注 3) ocd2	0:XIN クロック選択(注 6) 1:オンチップオシレータクロック選択(注 2)	0
bit1	発振停止検出割り込み許可ビット ocd1	0:禁止(注 1) 1:許可	変更なし
bit0	発振停止検出有効ビット(注 6) ocd0	0:発振停止検出機能無効(注 1) 1:発振停止検出機能有効	変更なし

注 1. ストップモード、高速オンチップオシレータモード、低速オンチップオシレータモード(XIN クロック停止) に移行する前に OCD1～OCD0 ビットを“00b”に設定してください。
注 2. OCD2 ビットを“1”(オンチップオシレータクロック選択)にすると、CM14 ビットは“0”(低速オンチップオシレータ発振)になります。
注 3. OCD2 ビットは、OCD1～OCD0 ビットが“11b”のときに XIN クロック発振停止を検出すると、自動的に“1”(オンチップオシレータクロック選択)に切り替わります。また、OCD3 ビットが“1”(XIN クロック停止)のとき、OCD2 ビットに“0”(XIN クロック選択)を書いても変化しません。
注 4. OCD3 ビットは OCD0 ビットが“1”(発振停止検出機能有効)のとき有効です。
注 5. OCD1～OCD0 ビットが“00b”のとき OCD3 ビットは“0”(XIN クロック発振)になり、変化しません。
注 6. 発振停止検出後、XIN クロックが再発振した場合の切り替え手順は、R8C/35A ハードウェアマニュアルの「図 9.10 低速オンチップオシレータから XIN クロックへの切り替え手順」を参照してください。

先の設定で、XIN クロック(XIN-XOUT)停止ビット(CM05)を″0″にして、XIN クロックを発振させました。システムクロックを低速オンチップオシレータ(約 125KHz)から、XIN クロックに切り替えます。ここから、動作が 20MHz になります。OCD の bit2 を″0″にします。今回は、OCD2 を″0″にします。

```
59 :        ocd2  = 0;                          /* システムクロックを XIN にする  */
```

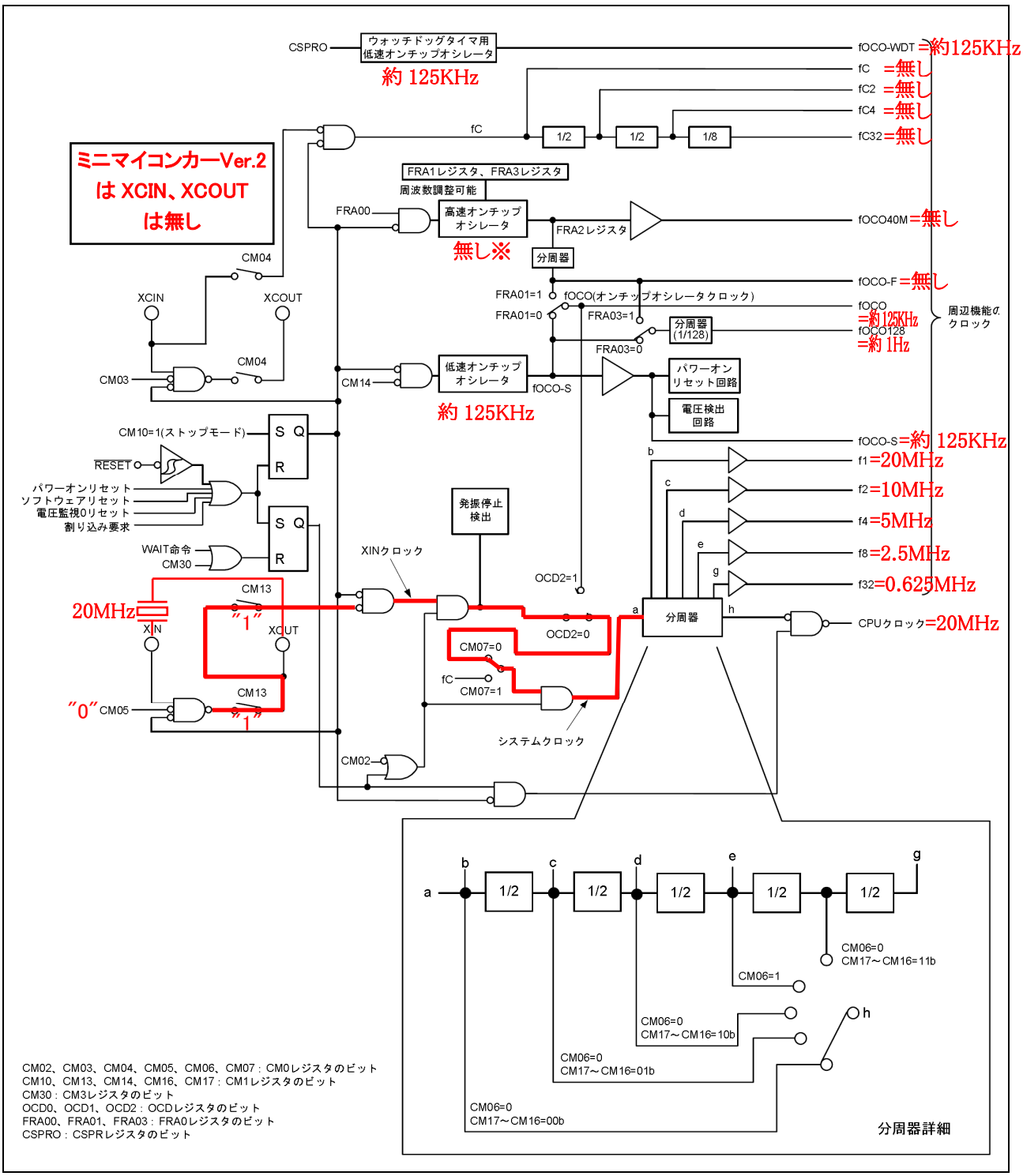
⑥プロテクトレジスタ(PRCR:Protect register)の設定(書き込み禁止)

①で書き込みを許可しました。それを、元に戻します(書き込み禁止にします)。

```
60 :        prc0  = 0;                          /* プロテクト ON                  */
```

(4) 設定後のクロックの状態

今回の設定をしたときの、クロック発生回路を下記に示します。



※R8C/35A には、高速オンチップオシレータは搭載されていません。R8C/35C には搭載されています。

9.5.3 init関数(I/Oポートの入出力設定)

init 関数内でクロックの選択と、I/O ポートの入出力設定を行っています。ここでは I/O ポートの入出力について説明します。プログラムを下記に示します。

```
 62 :      /* ポートの入出力設定 */
 63 :      prc2 = 1;                          /* PD0のプロテクト解除          */
 64 :      pd0 = 0x00;                        /* スイッチなど入力             */
 65 :      p1  = 0x0f;                        /* 3-0:LEDは消灯                */
 66 :      pd1 = 0xdf;                        /* 5:RXD0 4:TXD0 3-0:LED        */
 67 :      pd2 = 0xfe;                        /* 0:PushSW                     */
 68 :      pd3 = 0xfb;                        /* 4:Buzzer 2:IR                */
 69 :      pd4 = 0x80;                        /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
 70 :      pd5 = 0x40;                        /* 7:DIP SW                     */
 71 :      pd6 = 0xff;                        /* LEDなど出力                  */
 72 :  }
```

(1) I/Oポートとは

I/O ポートは、Input／Output Port の略で、直訳すると「**入力や出力を行う港**」という意味です。今回は「入力、出力をまとめて行う場所」というような意味合いです。I/O ポートの入出力設定は、プログラムの初めに行います。

(2) ポートの入出力設定

R8C/35A には、ポート 0～ポート 6 まで 7 本のポートがあります。基本的には 1 ポート 8 ビットですが、8 ビットないポートもあります。それぞれの端子を入力用として使用するか、出力用として使用するか、表にまとめてみます。

考え方は、次のとおりです。

・ディップスイッチなど、外部からの信号をマイコンへ入力する場合、端子を「入力」にします。

・LED など、マイコンから外部へ信号を出力する場合、端子を「出力」にします。

今回の実習でのポートの一覧を下記に示します。ゴシック体部分は、実習基板を接続した場合の内容です。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**	**ディップスイッチ入力**
1	未接続	未接続	RxD0 入力	TxD0 出力	マイコンボード上の LED3 出力	マイコンボード上の LED2 出力	マイコンボード上の LED1 出力	マイコンボード上の LED0 出力
2	未接続	未接続	未接続	未接続	未接続	未接続	未接続	マイコンボード上のタクトスイッチ入力
3	未接続	未接続	未接続	マイコンボード上のブザー出力	未接続	マイコンボード上の赤外線受光 IC 入力	未接続	未接続
4	クリスタル出力	クリスタル入力	マイコンボード上の SW2 入力	マイコンボード上の SW1 入力	マイコンボード上の SW0 入力	Vcc 入力		
5	マイコンボード上の SW3 入力	未接続						
6	**LED 出力**	**LED 出力**	**LED 出力**	**LED 出力**	**LED 出力**	**LED 出力**	**LED 出力**	**LED 出力**

※表の斜線の bit は、端子がない bit です。
※リセット後は、全て入力ポートです。

ポートの入出力方向の設定は、**「ポート Pi 方向レジスタ」(i は 0～6 の数字が入ります)で行い、レジスタ名は「pd0～pd6」の 7 個あります**。最後の数字がポート番号です。例えばポート 0 は「PD0」となります。
ポート Pi 方向レジスタの設定方法を下記に示します。

①外部へ信号を出力する端子は、"1"を設定する
②外部から信号を入力する端子は、"0"を設定する
③未接続の端子は、プルアップ抵抗、またはプルダウン抵抗を接続して入力("0")にするか、
　何も接続せずに出力("1")にする。今回は、後者で設定する
④端子が無い場合(表の斜線部分)は、"0"にする

①～④の考え方を基に、"1"、"0"で書き換えた表を下記に示します。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	16 進数
0	0	0	0	0	0	0	0	0	0x00
1	1	1	0	1	1	1	1	1	0xdf
2	1	1	1	1	1	1	1	0	0xfe
3	1	1	1	1	1	0	1	1	0xfb
4	1	0	0	0	0	0	0	0	0x80
5	0	1	0	0	0	0	0	0	0x40
6	1	1	1	1	1	1	1	1	0xff

C 言語は 2 進数表記はできないので、10 進数か 16 進数に変換します。通常は、2 進数を 16 進数に変換する方が簡単なため、16 進数に変換します。

表より、ポート P0 方向レジスタ(PD0)～ポート P6 方向レジスタ(PD6)に設定する値を、下記に示します。

ポート	方向レジスタ名	設定値
0	PD0	0x00
1	PD1	0xdf
2	PD2	0xfe
3	PD3	0xfb
4	PD4	0x80
5	PD5	0x40
6	PD6	0xff

プログラムを下記に示します。

```
63 :        prc2 = 1;                   /* PD0のプロテクト解除              */
64 :        pd0 = 0x00;                 /* スイッチなど入力                 */
65 :        p1  = 0x0f;                 /* 3-0:LEDは消灯                    */
66 :        pd1 = 0xdf;                 /* 5:RXD0 4:TXD0 3-0:LED            */
67 :        pd2 = 0xfe;                 /* 0:PushSW                         */
68 :        pd3 = 0xfb;                 /* 4:Buzzer 2:IR                    */
69 :        pd4 = 0x80;                 /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
70 :        pd5 = 0x40;                 /* 7:DIP SW                         */
71 :        pd6 = 0xff;                 /* LEDなど出力                      */
```

63 行は、次のページで説明します。

※PD0 を設定するときの注意点

R8C/35A マイコンには、プログラムが暴走したときに備え重要なレジスタは簡単に書き換えられないように保護するレジスタがあります。

PD0～PD6 の中で、PD0 だけは保護(プロテクト)されており、保護を解除しないと書き換えることができません。PD0 を書き換える前に、プロテクトレジスタ(PRCR)の bit2 を"1"にします。このビットは「sfr_r835a.h(R8C/35A のレジスタ定義ファイル)」で「PRC2」と定義されており、「PRC2」と記述すると PRCR の bit2 の意味になります。プログラムを下記に示します。

```
prc2 = 1;      ←PD0の書き換えを許可(PRC2=PRCRのbit2の意味です)
pd0 = 0x00;    ←その後、書き換える
```

PRC2 を"1"にした後、次に何かの命令を実行すると PRC2 は自動的に"0"(書き換え不可)になります。そのため、**必ず PRC2 を"1"にした次の行で PD0 を設定してください**。プログラムで PRC2 を"0"に戻す必要はありません。

(3) データの出力

出力に設定した端子からデータを出力するときの方法を説明します。

データを出力するときは、**「ポート Pi レジスタ」(i は 0～6 の数字が入ります)で行い、レジスタ名は「P0～P6」の 7 個あります**。最後の数字がポート番号です。例えばポート 0 は「P0」となります。

ポート Pi レジスタの設定値は、次のようになります。

①端子から"1"(5V)を出力したい場合は、"1"を設定する
②端子から"0"(0V)を出力したい場合は、"0"を設定する
③入力端子は、"0"、"1"のどちらを設定しても構わないが、"1"だと意味があるように思われるため
　"0"を設定しておく
④端子が無い場合は、"0"にする

例えば、ポート 3 の出力端子から次のように電圧を出力したいとします。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
3	5V	0V	0V	0V	5V	入力	0V	5V

5V 部分を"1"に、0V 部分を"0"に、入力部分を"0"にすれば、ポート 3 に設定する値になります。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
3	1	0	0	0	1	0	0	1

2 進数で「1000 1001」、16 進数に直すと「0x89」になります。プログラムは、次のように記述します。

```
p3 = 0x89;
```

(4) データの入力

入力に設定した端子からデータを入力するときの方法を説明します。

データを入力するときは、**「ポート Pi レジスタ」(i は 0～6 の数字が入ります)で行い、レジスタ名は「P0～P6」の 7 個あります**。最後の数字がポート番号です。例えばポート 0 は「P0」となります(レジスタ名は出力と同じです)。

ポートPiレジスタの値を読み込めば、端子の状態が分かります。端子の状態とポートPiレジスタ関係は、次のようになります。

①端子に 5V が入力されている場合は、"1"が読み込まれる
②端子に 0V が入力されている場合は、"0"が読み込まれる
③出力端子を読み込むと、現在出力している状態が読み込まれる
④端子が無い場合は、不定("0"か"1"か分からない状態)が読み込まれる

例えば、ポート 0 には次の電圧が入力されているとします(例なので今回の演習とは関係ありません)。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	出力端子 (0V 出力)	出力端子 (0V 出力)	出力端子 (5V 出力)	出力端子 (5V 出力)	5V	5V	0V	0V

ポート0を読み込んだときに入力される値は、5V 部分は"1"が、0V 部分は"0"が読み込まれます。また、出力端子は現在出力している状態が読み込まれます。よって次のようになります。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	1	1	1	1	0	0

2 進数で「0011 1100」、16 進数に直すと「0x3c」になります。次のようにプログラムすると、変数 c には、0x3c が代入されます。

```
c = p0;      // 変数cには0x3cが代入される
```

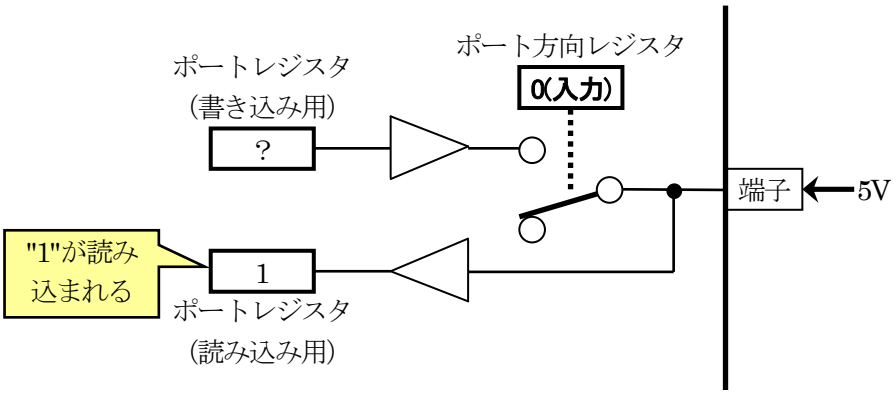
(5) 出力設定にするときのポート方向レジスタレジスタ(PD0～PD6)とポートレジスタ(P0～P6)の設定手順

ポート方向レジスタ(PD0～PD6)とポートレジスタ(P0～P6)を設定するとき、どちらを先に設定すればよいのか説明します。マイコンリセット後の初期値を下記に示します。

レジスタ	リセット後の初期値
ポート方向レジスタ(PD0～PD6)	0x00(入力設定)
ポートレジスタ(P0～P6)	不定(どのような値か分からない)

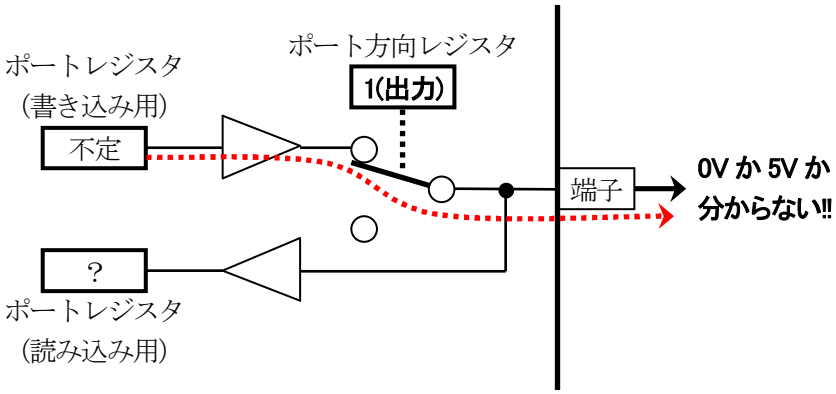
●入力設定の場合

リセット後は入力設定なので、ポートレジスタの値を読み込むと端子に入力されている状態が読み込まれます。

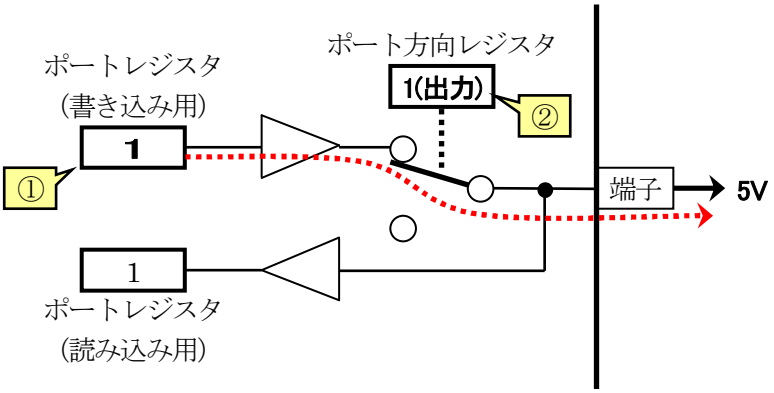


●出力設定の場合

リセット後、ポート方向レジスタを"1"にすると、出力設定になり、ポートレジスタの値が出力されます。ただし、**ポートレジスタの値はリセット後、不定です。そのため、出力設定にすると"0"が出力されるか"1"が出力されるか分かりません。**



そのため、**出力設定するときは、①ポートレジスタに出力したい値を設定してから、②ポート方向レジスタで出力に設定します。**



例えば、マイコンボード上の LED(ポート 1 の bit3～0)は、"0"で点灯、"1"で消灯します。ポート方向レジスタを設定する前に、ポートレジスタに"1"を設定してからポート方向レジスタを設定します。

```
    p1  = 0x0f;                     /* 3-0:LEDは消灯                */
    pd1 = 0xdf;                     /* 5:RXD0 4:TXD0 3-0:LED        */
```

(6) 入力時の電圧について

入力ポートの場合、"1"と判断する電圧(下表の V_{IH})と"0"と判断する電圧(下表の V_{IL})を、下表に示します。

記号	項目				測定条件	規格値			単位
						最小	標準	最大	
Vcc/AVcc	電源電圧					1.8	—	5.5	V
Vss/AVss	電源電圧					—	0	—	V
VIH	"H" 入力電圧	CMOS入力以外				0.8Vcc	—	Vcc	V
		CMOS入力	入力レベル切り替え機能(I/Oポート)	入力レベル選択：0.35Vcc	4.0V ≦ Vcc ≦ 5.5V	0.5Vcc	—	Vcc	V
					2.7V ≦ Vcc < 4.0V	0.55Vcc	—	Vcc	V
					1.8V ≦ Vcc < 2.7V	0.65Vcc	—	Vcc	V
				入力レベル選択：0.5Vcc	4.0V ≦ Vcc ≦ 5.5V	0.65Vcc	—	Vcc	V
					2.7V ≦ Vcc < 4.0V	0.7Vcc	—	Vcc	V
					1.8V ≦ Vcc < 2.7V	0.8Vcc	—	Vcc	V
				入力レベル選択：0.7Vcc	4.0V ≦ Vcc ≦ 5.5V	0.85Vcc	—	Vcc	V
					2.7V ≦ Vcc < 4.0V	0.85Vcc	—	Vcc	V
					1.8V ≦ Vcc < 2.7V	0.85Vcc	—	Vcc	V
VIL	"L" 入力電圧	CMOS入力以外				0	—	0.2Vcc	V
		CMOS入力	入力レベル切り替え機能(I/Oポート)	入力レベル選択：0.35Vcc	4.0V ≦ Vcc ≦ 5.5V	0	—	0.2Vcc	V
					2.7V ≦ Vcc < 4.0V	0	—	0.2Vcc	V
					1.8V ≦ Vcc < 2.7V	0	—	0.2Vcc	V
				入力レベル選択：0.5Vcc	4.0V ≦ Vcc ≦ 5.5V	0	—	0.4Vcc	V
					2.7V ≦ Vcc < 4.0V	0	—	0.3Vcc	V
					1.8V ≦ Vcc < 2.7V	0	—	0.2Vcc	V
				入力レベル選択：0.7Vcc	4.0V ≦ Vcc ≦ 5.5V	0	—	0.55Vcc	V
					2.7V ≦ Vcc < 4.0V	0	—	0.45Vcc	V
					1.8V ≦ Vcc < 2.7V	0	—	0.35Vcc	V

電源電圧(Vcc)は 5.0V、入力レベル選択は 0.5Vcc です。よって、下記のように判断します。

"1"と判断する電圧は表より
　　0.65Vcc＝0.65×5.0＝3.25V　　∴3.25V～5.0V の電圧なら"1"と判断
"0"と判断する電圧は表より
　　0.4Vcc＝0.4×5.0＝2.0V　　∴0～2.0V の電圧なら"0"と判断

では、上記の範囲に入っていない 2.0～3.25V の電圧が入力された場合はどうなるのでしょうか。この場合は、"0"と判断するか"1"と判断するか分かりません。ある時は"0"と判断し、ある時は"1"と判断するかもしれません。そのため、この範囲の電圧は加えないようにします。

(7) 出力時の電圧、電流について

出力ポートの場合、流せる電流を下表に示します。

$I_{OH(sum)}$	“H”尖頭総出力電流	全端子の$I_{OH(peak)}$の総和		—	—	−160	mA
$I_{OH(sum)}$	“H”平均総出力電流	全端子の$I_{OH(avg)}$の総和		—	—	−80	mA
$I_{OH(peak)}$	“H”尖頭出力電流	駆動能力Low時		—	—	−10	mA
		駆動能力High時		—	—	−40	mA
$I_{OH(avg)}$	“H”平均出力電流	駆動能力Low時		—	—	−5	mA
		駆動能力High時		—	—	−20	mA
$I_{OL(sum)}$	“L”尖頭総出力電流	全端子の$I_{OL(peak)}$の総和		—	—	160	mA
$I_{OL(sum)}$	“L”平均総出力電流	全端子の$I_{OL(avg)}$の総和		—	—	80	mA
$I_{OL(peak)}$	“L”尖頭出力電流	駆動能力Low時		—	—	10	mA
		駆動能力High時		—	—	40	mA
$I_{OL(avg)}$	“L”平均出力電流	駆動能力Low時		—	—	5	mA
		駆動能力High時		—	—	20	mA
$f_{(XIN)}$	XINクロック入力発振周波数		2.7V ≦ Vcc ≦ 5.5V	—	—	20	MHz
			1.8V ≦ Vcc < 2.7V	—	—	5	MHz
$f_{(XCIN)}$	XCINクロック入力発振周波数		1.8V ≦ Vcc ≦ 5.5V	—	32.768	50	kHz
fOCO40M	タイマRC、タイマRDのカウントソース(注3)		2.7V ≦ Vcc ≦ 5.5V	32	—	40	MHz
fOCO-F	fOCO-F周波数		2.7V ≦ Vcc ≦ 5.5V	—	—	20	MHz
			1.8V ≦ Vcc < 2.7V	—	—	5	MHz
—	システムクロック周波数		2.7V ≦ Vcc ≦ 5.5V	—	—	20	MHz
			1.8V ≦ Vcc < 2.7V	—	—	5	MHz
$f_{(BCLK)}$	CPUクロック周波数		2.7V ≦ Vcc ≦ 5.5V	—	—	20	MHz
			1.8V ≦ Vcc < 2.7V	—	—	5	MHz

注1. 指定のない場合は、Vcc = 1.8V～5.5V、Topr = −20℃～85℃(Nバージョン)/−40℃～85℃(Dバージョン)です。
注2. 平均出力電流は100 msの期間内での平均値です。
注3. fOCO40MはVcc = 2.7V～5.5Vの範囲で、タイマRC、タイマRDのカウントソースとして使用することができます。

″1″(5V)にした端子から流すことのできる電流は、5mA です($I_{OH(avg)}$)。また、マイコン全体では、″1″(5V)にした端子から流すことのできる電流は 80mA です($I_{OH(sum)}$)。

″0″(0V)にした端子から流すことのできる電流は、5mA です($I_{OL(avg)}$)。また、マイコン全体では、″0″(0V)にした端子から流すことのできる電流は 80mA です($I_{OL(sum)}$)。

出力ポートの場合、"1"と"0"の実際の出力電圧を下表に示します。

記号	項目		測定条件		規格値 最小	規格値 標準	規格値 最大	単位
VOH	"H" 出力電圧		駆動能力High VCC = 5V	IOH = －20mA	Vcc－2.0	—	Vcc	V
			駆動能力Low VCC = 5V	IOH = －5mA	Vcc－2.0	—	Vcc	V
VOL	"L" 出力電圧		駆動能力High VCC = 5V	IOL = 20mA	—	—	2.0	V
			駆動能力Low VCC = 5V	IOL = 5mA	—	—	2.0	V
VT+-VT-	ヒステリシス	INT0、INT1、INT2、INT3、INT4、KI0、KI1、KI2、KI3、TRAIO、TRBO、TRCIOA、TRCIOB、TRCIOC、TRCIOD、TRDIOA0、TRDIOB0、TRDIOC0、TRDIOD0、TRDIOA1、TRDIOB1、TRDIOC1、TRDIOD1、TRCTRG、TRCCLK、ADTRG、RXD0、RXD1、RXD2、CLK0、CLK1、CLK2、SSI、SCL、SDA、SSO			0.1	1.2	—	V
		RESET			0.1	1.2	—	V
IIH	"H" 入力電流		VI = 5V		—	—	5.0	μA
IIL	"L" 入力電流		VI = 0V		—	—	－5.0	μA
RPULLUP	プルアップ抵抗		VI = 0V		25	50	100	kΩ
RfXIN	帰還抵抗	XIN			—	0.3	—	MΩ
RfXCIN	帰還抵抗	XCIN			—	8	—	MΩ
VRAM	RAM保持電圧		ストップモード時		1.8	—	—	V

注1. 指定のない場合は、4.2V≦Vcc≦5.5V、Topr = －20℃～85℃(Nバージョン)/－40℃～85℃(Dバージョン)、f(XIN) = 20MHzです。

"1"出力は、(Vcc-2.0)～Vcc の電圧が出力されます。Vcc は 5.0V なので、3.0～5.0V が出力されます。マイコンに接続されている回路は、この電圧で動作するように回路設計する必要があります。

"0"出力は、0～2V が出力されます。マイコンに接続されている回路は、この電圧で動作するように回路設計する必要があります。

9.5.4 main関数

```
 35 :  void main( void )
 36 :  {
 37 :      unsigned char d;
 38 :
 39 :      init();                              /* 初期化                    */
 40 :
 41 :      while( 1 ) {
 42 :          d = p0;
 43 :          p6 = d;
 44 :      }
 45 :  }
```

「io.c」は、main 関数から実行されます。

39 行	init 関数を実行します。
41 行	while( )は、カッコの中が成り立ったなら(真なら)中カッコ(波カッコ)の中を実行、成り立たないなら(偽なら)中カッコを抜ける命令です。今回は()の中が「1」です。これは、C 言語では成り立つという意味になり、常に成り立つ状態です。41 行目の最後の「{」から、44 行目の「}」の間のプログラムが無限に繰り返されます。要は、42 行、43 行が無限に繰り返されます。
42 行	変数 d にポート 0 の状態を読み込み、保存します。
43 行	ポート d の値を、ポート 6 に出力します。

9.6 ビット操作

9.6.1 特定のビットを"0"にする(マスク処理)

特定のビットを強制的に"0"にするには、AND 演算を行います。
例えば、ポート 0 の bit7～4 はそのままにして、bit3～0 を強制的に"0"にしたいとします。表にまとめます。

bit	7	6	5	4	3	2	1	0
状態	そのまま	そのまま	そのまま	そのまま	"0"にする	"0"にする	"0"にする	"0"にする

そのままの部分を「1」に、"0"にする部分を「0」に置き換えます。

bit	7	6	5	4	3	2	1	0
変換後	1	1	1	1	0	0	0	0
16 進数	F				0			

ポート 0 と変換後の値を AND 演算することにより、bit3～0 の値を強制的に"0"にすることができます。AND 演算の記述は C 言語では"&"(アンパサンド)を用います。C 言語で記述すると次のようになります。

```
d = p0 & 0xf0;
p6 = d;                    ←結果をポート6へ出力
```

例えば、ポート 0 の値が 0x55 なら、結果(変数 d の値)は次のようになります。

bit	7	6	5	4	3	2	1	0
P0 の値	0	1	0	1	0	1	0	1
AND 値	1	1	1	1	0	0	0	0
結果	0	1	0	1	0	0	0	0

AND 値が"1"なら、結果は P0 の値になる　　AND 値が"0"なら、結果は"0"になる

■マスクについて

ポートの特定のビットが"1"か"0"かチェックしたいとします。**1 ポートの単位は 8bit のため、1bit だけチェックすることはできません**(ビットフィールドという方法を使えばできますが、ここでは無しにします)。**必ず 8bit まとめてのチェックとなります**。
例えば、ポート 0 にディップスイッチ基板が繋がっていて、bit7 が"1"かどうかチェックしたい場合、次のようにすればいいように思えます。

```
if( p0 == 0x80 ) {
    /* bit 7 が “1” ならこの中を実行 */
}
```

しかし、ポート 0 の bit6～0 がどのような値になっているか分かりません。例えば、bit7 が“1”、bit0 も”1”ならポート 0 の値は 2 進数で"1000 0001"、16 進数に直すと 0x81 となります。そのため、次のようになります。

```
if( p0 == 0x80 ) {      ←p0 は 0x81 なので式は成り立たない
    /* bit 7 は “1” だが、 この中を実行しない！！ */
}
```

bit7 は"1"ですが、bit0も"1"なので、式は成り立たずカッコの中を実行しません。このようなときは、**チェック不要なビットを AND 演算で強制的に"0"にします。これを「マスク」(覆い隠す)といいます。マスクは、AND 処理でチェックに不要なビットを強制的に"0"にすることです。AND 処理する値をマスク値といいます**。
今回、チェックしたい bit は 7、その他はチェックしません。表にまとめます。

bit	7	6	5	4	3	2	1	0
チェックする？	する	しない	しない	しない	しない	しない	しない	しない

マスク値は、チェックする bit を"1"に、チェックしない bit を"0"にするだけです。

bit	7	6	5	4	3	2	1	0
マスク値	1	0	0	0	0	0	0	0

プログラムでは、チェックする値(今回はポート 0)とマスク値を AND 演算した結果をチェックします。結果は、bit6～0 が"0"であることが分かっているので、bit7 だけ"1"か"0"かをチェックすればいいことになります。
bit7 が"0"かどうか調べるなら、プログラムは次のようになります。

```
if( (p0 & 0x80) == 0x00 ) {
    /* bit 7 が “0” ならこの中を実行 */
}
```

P0 が 0x71 なら、次のような計算になります。

```
    P0        0111 0001
    マスク値  1000 0000  (AND
    結果      0000 0000
```

bit7 は"0"なので式は成り立ち、カッコの中が実行されます。

bit7 が"1"かどうか調べるなら、プログラムは次のようになります。

```
if( (p0 & 0x80) == 0x80 ) {
    /* bit７が“1”ならこの中を実行 */
}
```

P0 が 0xaf なら、次のような計算になります。

```
P0        1010 1111
マスク値  1000 0000  (AND
結果      1000 0000
```

bit7 は"1"なので式は成り立ち、カッコの中が実行されます。

9.6.2 特定のビットを"1"にする

特定のビットを強制的に"1"にするには、OR 演算を行います。
例えば、ポート 0 の bit7～4 はそのままにして、bit3～0 を強制的に"1"にしたいとします。表にまとめます。

bit	7	6	5	4	3	2	1	0
状態	そのまま	そのまま	そのまま	そのまま	"1"にする	"1"にする	"1"にする	"1"にする

そのままの部分を「0」に、"1"にする部分を「1」に置き換えます。

bit	7	6	5	4	3	2	1	0
変換後	0	0	0	0	1	1	1	1
16 進数	0				f			

ポート 0 と変換後の値を OR 演算することにより、bit3～0 の値を強制的に"1"にすることができます。OR 演算の記述は C 言語では"|"(縦線)を用います。C 言語で記述すると次のようになります。

```
d = p0 | 0x0f;
p6 = d;                ←結果をポート6へ出力
```

例えば、ポート 0 の値が 0x55 なら、結果(変数 d の値)は次のようになります。

bit	7	6	5	4	3	2	1	0
P0 の値	0	1	0	1	0	1	0	1
OR 値	0	0	0	0	1	1	1	1
結果	0	1	0	1	1	1	1	1

OR 値が"0"なら、結果は P0 の値になる　　OR 値が"1"なら、結果は"1"になる

9.6.3 特定のビットを反転する

特定のビットを反転する(″0″なら″1″に、″1″なら″0″にする)には、XOR(exclusive or、排他的論理和)演算を行います。

例えば、ポート 0 の bit7～4 はそのままにして、bit3～0 を反転したいとします。表にまとめます。

bit	7	6	5	4	3	2	1	0
状態	そのまま	そのまま	そのまま	そのまま	反転する	反転する	反転する	反転する

そのままの部分を「0」に、反転する部分を「1」に置き換えます。

bit	7	6	5	4	3	2	1	0
変換後	0	0	0	0	1	1	1	1
16 進数	0				f			

ポート 0 と変換後の値を XOR 演算することにより、bit3～0 の値を反転することができます。XOR 演算の記述は C 言語では″^″(アクサンシルコンフレックス)を用います。C 言語で記述すると次のようになります。

```
d = p0 ^ 0x0f;
p6 = d;                    ←結果をポート6へ出力
```

例えば、ポート 0 の値が 0x55 なら、結果(変数 d の値)は次のようになります。

bit	7	6	5	4	3	2	1	0
P0 の値	0	1	0	1	0	1	0	1
XOR 値	0	0	0	0	1	1	1	1
結果	0	1	0	1	1	0	1	0

XOR 値が″0″なら、結果は P0 の値になる　　XOR 値が″1″なら、結果は反転する

9.6.4 全ビット反転する(NOT演算)

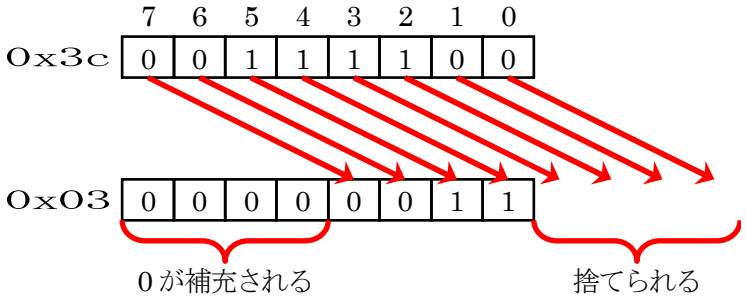
全ビット反転するなら、C 言語では″~″(チルダ)を用います。C 言語で記述すると次のようになります。

```
d = ~p0;
p6 = d;                    ←結果をポート6へ出力
```

反転したい値の先頭に″~″を付けます。

9.6.5 ビットシフトする

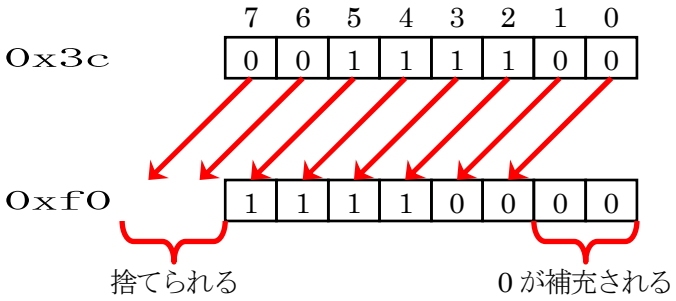
ビットシフトとは、各ビットを右や左にずらすことです。
例えば、ポート 0 を 4 ビット右にシフトするとします。ポート 0 が 0x3c なら、結果は 0x03 になります。



右シフトの記述は C 言語では">>"を用います。C 言語で記述すると次のようになります。

```
d = p0 >> 4;
        └──シフトする数
p6 = d;                 ←結果をポート6へ出力
```

例えば、ポート 0 を 2 ビット左にシフトするとします。ポート 0 が 0x3c なら、結果は 0xf0 になります。



左シフトの記述は C 言語では"<<"を用います。C 言語で記述すると次のようになります。

```
d = p0 << 2;
        └──シフトする数
p6 = d;                 ←結果をポート6へ出力
```

9.7 ビット単位でデータを入力、出力する

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「sfr_r835a.h」は、ポート Pi レジスタ(P0～P6)をビット単位で設定できるよう定義しています。
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

■ポートにデータを出力する

例えば、ポート 2 の bit0 に"1"を設定したい場合のプログラムを下記に示します。

```
p2_0 = 1;
```

■ポートからデータを入力する

例えば、ポート 0 の bit2 の値を変数 c に代入するプログラムを書きに示します。

```
c = p0_2;
```

例えば、ポート 0 の bit5 の状態を確認する場合のプログラムを書きに示します。

```
if( p0_5 == 1 ) {
    p0_5 が"1"ならこの部分を実行
} else {
    p0_5 が"0"なら、この部分を実行
}
```

9.8 演習

(1) ポート 0 に LED 部、ポート 6 にディップスイッチ部を接続して、ディップスイッチの状態を LED へ出力しなさい。

(2) (1)のとき、入力した値の bit7～4 を強制的に"0"にして、LED へ出力しなさい。

(3) (1)のとき、入力した値の bit3～0 を強制的に"1"にして、LED へ出力しなさい。

(4) (1)のとき、ディップスイッチの値をすべて反転して LED へ出力しなさい。

(5) (1)のとき、ディップスイッチの 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、ディップスイッチの bit4 以外は"1"か"0"か分からないものとする。

10. I/Oポートの入出力 2(プロジェクト:io2)

10.1 概要

本章では、マイコンボード上にあるディップスイッチ(4bit)と LED(4bit)を使って、I/O ポートの入出力を行う方法を説明します。ポートが分かれているため、ビット操作を行って入出力を行います。

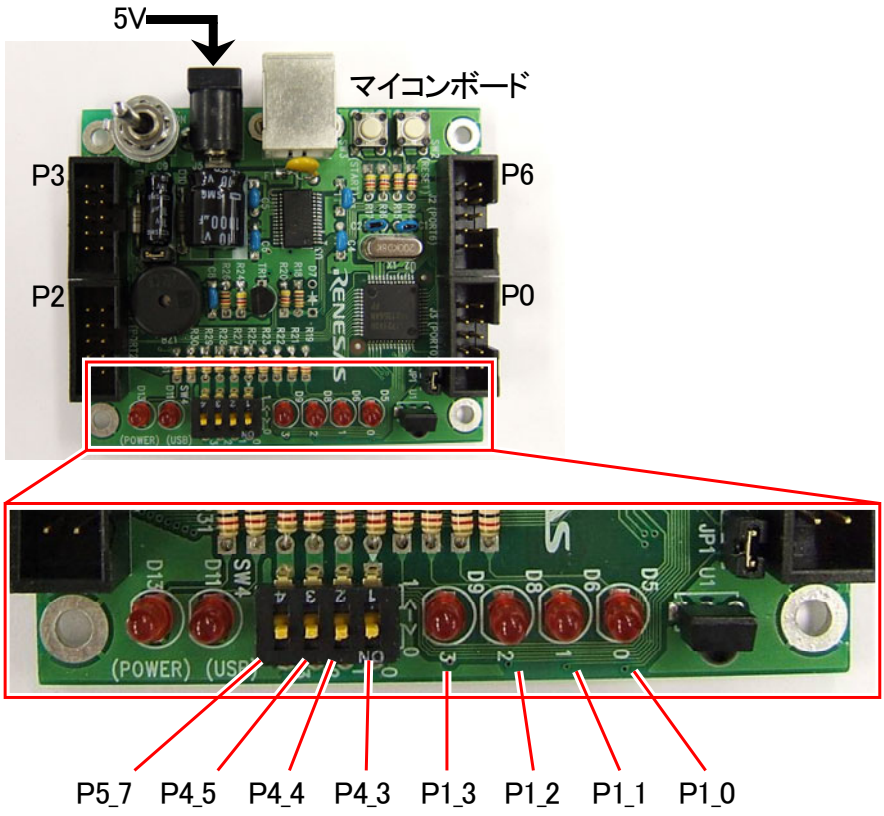
10.2 接続

■使用ポート

マイコンのポート	接続内容
P5_7、P4_5、P4_4、P4_3	マイコンボード上のディップスイッチです。
P1_3、P1_2、P1_1、P1_0	マイコンボード上の LED です。

■接続例

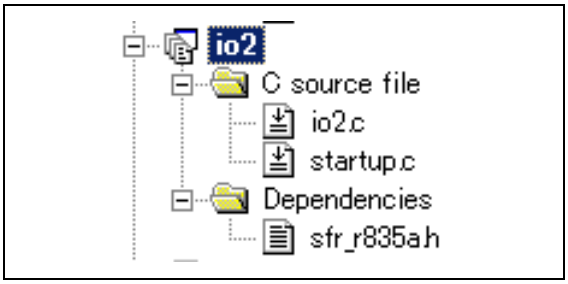
マイコンボードだけで実習できます。



■操作方法

マイコンボードのディップスイッチ(SW4)を ON/OFF すると、それに合わせて LED(D9,D8,D6,D5)が点灯／消灯します。

10.3 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	io2.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。

10.4 プログラム「io2.c」

```
 1 :  /***********************************************************************/
 2 :  /* 対象マイコン  R8C/35A                                               */
 3 :  /* ﾌｧｲﾙ内容      データの入力、出力                                    */
 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 :  入力：マイコンボードのディップスイッチ(4bit)
11 :  出力：マイコンボードのLED(4bit)
12 :
13 :  マイコンボードのディップスイッチ(4bit)から入力した状態を、
14 :  マイコンボードのLED(4bit)に出力します。
15 :  */
16 :
17 :  /*====================================*/
18 :  /* インクルード                       */
19 :  /*====================================*/
20 :  #include "sfr_r835a.h"                 /* R8C/35A SFRの定義ファイル     */
21 :
22 :  /*====================================*/
23 :  /* シンボル定義                       */
24 :  /*====================================*/
25 :
26 :  /*====================================*/
27 :  /* プロトタイプ宣言                   */
28 :  /*====================================*/
29 :  void init( void );
30 :  unsigned char dipsw_get( void );
31 :  void led_out( unsigned char led );
32 :
```

```
 33 :  /************************************************************************/
 34 :  /* メインプログラム                                                     */
 35 :  /************************************************************************/
 36 :  void main( void )
 37 :  {
 38 :      unsigned char d;
 39 :
 40 :      init();                             /* 初期化                       */
 41 :
 42 :      while( 1 ) {
 43 :          d = dipsw_get();
 44 :          led_out( d );
 45 :      }
 46 :  }
 47 :
 48 :  /************************************************************************/
 49 :  /* R8C/35A スペシャルファンクションレジスタ(SFR)の初期化                */
 50 :  /************************************************************************/
 51 :  void init( void )
 52 :  {
 53 :      int i;
 54 :
 55 :      /* クロックをXINクロック(20MHz)に変更 */
 56 :      prc0  = 1;                          /* プロテクト解除               */
 57 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
 58 :      cm05  = 0;                          /* XINクロック発振              */
 59 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
 60 :      ocd2  = 0;                          /* システムクロックをXINにする  */
 61 :      prc0  = 0;                          /* プロテクトON                 */
 62 :
 63 :      /* ポートの入出力設定 */
 64 :      prc2 = 1;                           /* PD0のプロテクト解除          */
 65 :      pd0 = 0xe0;                         /* 7-5:LED 4:MicroSW 3-0:Sensor */
 66 :      p1  = 0x0f;                         /* 3-0:LEDは消灯                */
 67 :      pd1 = 0xdf;                         /* 5:RXD0 4:TXD0 3-0:LED        */
 68 :      pd2 = 0xfe;                         /* 0:PushSW                     */
 69 :      pd3 = 0xfb;                         /* 4:Buzzer 2:IR                */
 70 :      pd4 = 0x83;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
 71 :      pd5 = 0x40;                         /* 7:DIP SW                     */
 72 :      pd6 = 0xff;
 73 :  }
 74 :
 75 :  /************************************************************************/
 76 :  /* ディップスイッチ値読み込み                                           */
 77 :  /* 戻り値 スイッチ値 0～15                                              */
 78 :  /************************************************************************/
 79 :  unsigned char dipsw_get( void )
 80 :  {
 81 :      unsigned char sw, sw1, sw2;
 82 :
 83 :      sw1 = (p5>>4) & 0x08;               /* ディップスイッチ読み込み3    */
 84 :      sw2 = (p4>>3) & 0x07;               /* ディップスイッチ読み込み2,1,0*/
 85 :      sw = sw1 | sw2;                     /* P5とP4の値を合わせる         */
 86 :
 87 :      return  sw;
 88 :  }
 89 :
 90 :  /************************************************************************/
 91 :  /* マイコン部のLED出力                                                  */
 92 :  /* 引数    スイッチ値 0～15                                             */
 93 :  /************************************************************************/
 94 :  void led_out( unsigned char led )
 95 :  {
 96 :      unsigned char data;
 97 :
 98 :      led = ~led;
 99 :      led &= 0x0f;
100 :      data = p1 & 0xf0;
101 :      p1 = data | led;
102 :  }
103 :
104 :  /************************************************************************/
105 :  /* end of file                                                          */
106 :  /************************************************************************/
```

10.5 プログラムの解説

10.5.1 dipsw_get関数

dipsw_get 関数は、マイコンボードの 4bit のディップスイッチの値を読み込む関数です。

```
75 :  /************************************************************************/
76 :  /* ディップスイッチ値読み込み                                           */
77 :  /* 戻り値 スイッチ値 0～15                                              */
78 :  /************************************************************************/
79 :  unsigned char dipsw_get( void )
80 :  {
81 :      unsigned char sw, sw1, sw2;
82 :
83 :      sw1 = (p5>>4) & 0x08;              /* ディップスイッチ読み込み3     */
84 :      sw2 = (p4>>3) & 0x07;              /* ディップスイッチ読み込み2,1,0*/
85 :      sw = sw1 | sw2;                    /* P5とP4の値を合わせる          */
86 :
87 :      return  sw;
88 :  }
```

マイコンボードには 4bit のディップスイッチが搭載されています。左から順にマイコンの P5_7、P4_5、P4_4、P4_3 の各 bit に接続されています。

ポートがばらばらなので、ビット演算を使って、次の図のようにビットを移動させて bit3～0 になるようにします。これを行うのが、dipsw_get 関数です。dipsw_get 関数を呼ぶと、0～15 の値が返ってきます。ディップスイッチを上にすると"1"、下(ON と書いてある側)にすると"0"です。

まず、変数 sw1 にポート 5(P5)の値を読み込みます。

```
83 :      sw1 = (p5>>4) & 0x08;              /* ディップスイッチ読み込み3     */
                 ①     ②
```

① ポート 5(P5)の値を 4bit 右シフトします。
② ①の値を 0x08 でマスクします。0x08 は、「0000 1000」なので bit3 のみ有効に、他は強制的に"0"にします。

次に、変数 sw2 にポート 4(P4)の値を読み込みます。

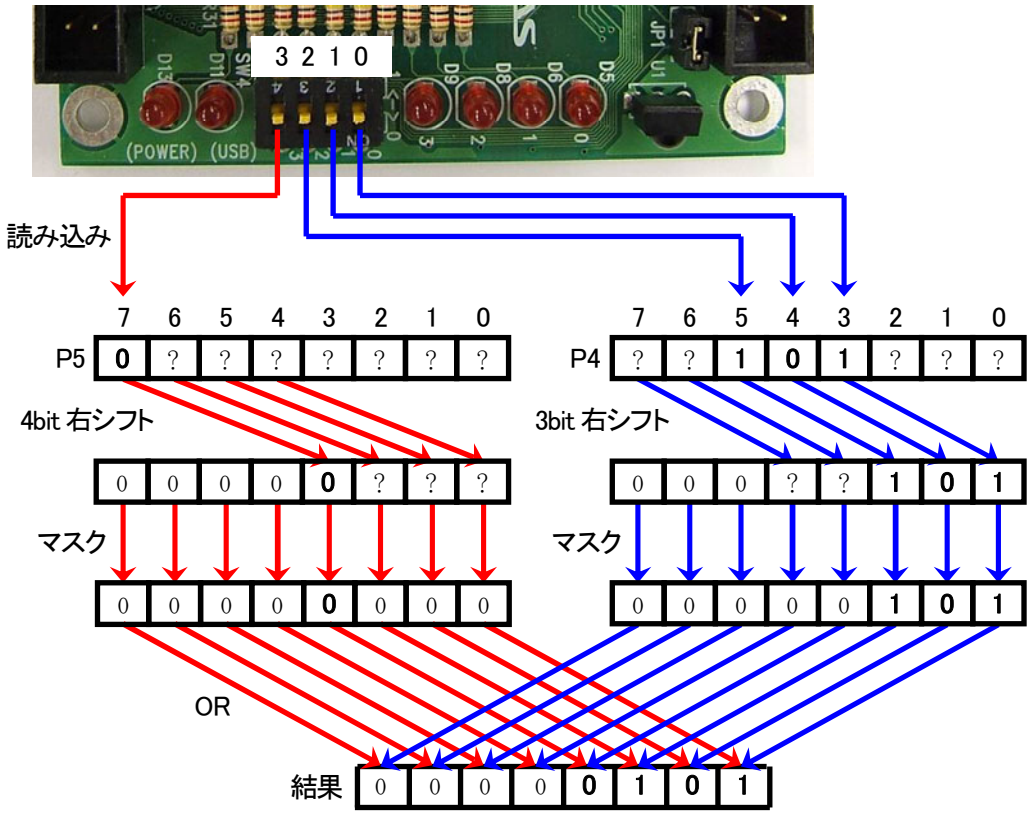
```
84 :      sw2 = (p4>>3) & 0x07;              /* ディップスイッチ読み込み2,1,0*/
                 ①     ②
```

① ポート 4(P4)の値を 3bit 右シフトします。
② ①の値を 0x07 でマスクします。0x07 は、「0000 0111」なので bit2～0 のみ有効に、他は強制的に"0"にします。

最後に、sw1 と sw2 の値を OR 演算で合わせます。

```
85 :        sw = sw1 | sw2;                    /* P5とP4の値を合わせる          */
```

ディップスイッチが"0101"(下上下上)のときの、dipsw_get 関数の動きを下図に示します。



10.5.2 led_out関数

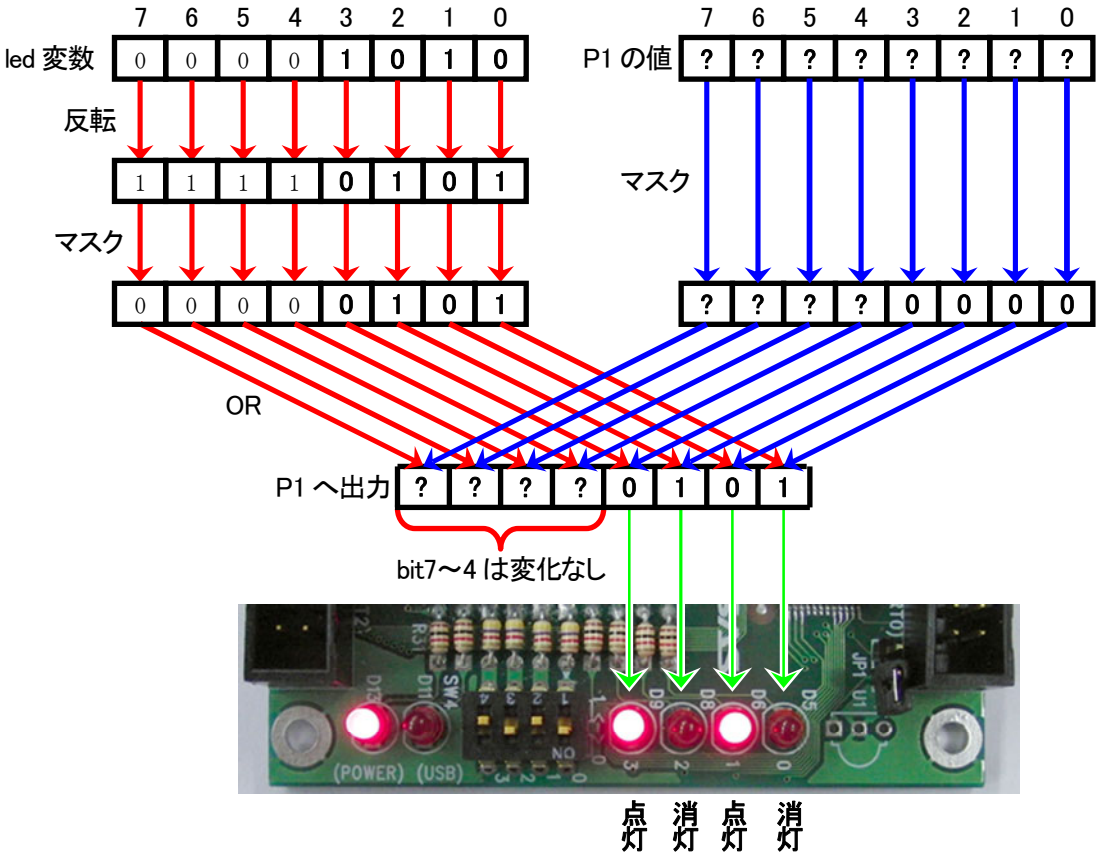
led_out 関数は、マイコンボードの 4 個の LED を点灯／消灯させる関数です。

```
 90 :  /*************************************************************************/
 91 :  /* マイコン部のLED出力                                                   */
 92 :  /* 引数    スイッチ値 0～15                                              */
 93 :  /*************************************************************************/
 94 :  void led_out( unsigned char led )
 95 :  {
 96 :      unsigned char data;
 97 :
 98 :      led = ~led;
 99 :      led &= 0x0f;
100 :      data = p1 & 0xf0;
101 :      p1 = data | led;
102 :  }
```

マイコンボードには 4 個の LED が搭載されています。左から順にマイコンの P1_3、P1_2、P1_1、P1_0 の各ビットに接続されています。

LED はポート 1 の bit3～0 に接続されていますので、ポート 1 の bit7～4 には影響がないようにします。また、LED は″1″で消灯、″0″で点灯なので、反転させます。これを行うのが、led_out 関数です。

led_out 関数に引数 10 を代入したときの動きを、下図に示します。



led 変数に入力されている値は、"0"で消灯、"1"で点灯です。実際の LED は"1"で消灯、"0"で点灯なので、引数の led 変数を反転させます。

```
 98 :       led = ~led;
```

LED は bit3～0 に接続されています。関係ない bit7～4 を"0"にしておきます。

```
 99 :       led &= 0x0f;
```

次に、LED が繋がっているポート 1(P1)の値を読み込みます。このとき、LED のある bit3～0 は"0"にしておきます。bit7～4 の値は現在のポート 1(P1)の値のままにしておきます。この値を変数 data に代入します。

```
100 :       data = p1 & 0xf0;
```

最後に、data と led の値を OR 演算で合わせ、ポート 1(P1)へ出力します。

```
101 :       p1 = data | led;
```

10.5.3 main関数

```
 36 :  void main( void )
 37 :  {
 38 :      unsigned char d;
 39 :
 40 :      init();                                 /* 初期化                          */
 41 :
 42 :      while( 1 ) {
 43 :          d = dipsw_get();
 44 :          led_out( d );
 45 :      }
 46 :  }
```

main 関数は次のような動作をします。

43 行	変数 d にマイコンボード上のディップスイッチの値を読み込みます。
44 行	マイコンボード上の LED に変数 d の値を出力します。

結果、マイコンボード上のディップスイッチの値を、マイコンボード上の LED へ出力します。

10.6 演習

本演習では、
ディップスイッチ…マイコンボード上のディップスイッチ
LED…マイコンボード上の LED
とする。

(1) ディップスイッチの値を反転させて、LED へ出力しなさい。

(2) ディップスイッチの bit0 が"1"なら LED へ"1010"、"0"なら LED へ"0101"を出力するようにしなさい。ただし、bit0 以外は"0"か"1"かは分からないものとする。

11. プッシュスイッチの情報入力（プロジェクト：pushsw）

11.1 概要

本章では、マイコンボードのプッシュスイッチ(SW3)の値を読みこむ方法を説明します。

※製作マニュアルでは、SW3 をタクトスイッチと説明していますが、マイコンカー関連のマニュアルでは、本スイッチを「プッシュスイッチ」と説明しています。そのため、本マニュアルでも「プッシュスイッチ」という名称で説明します。

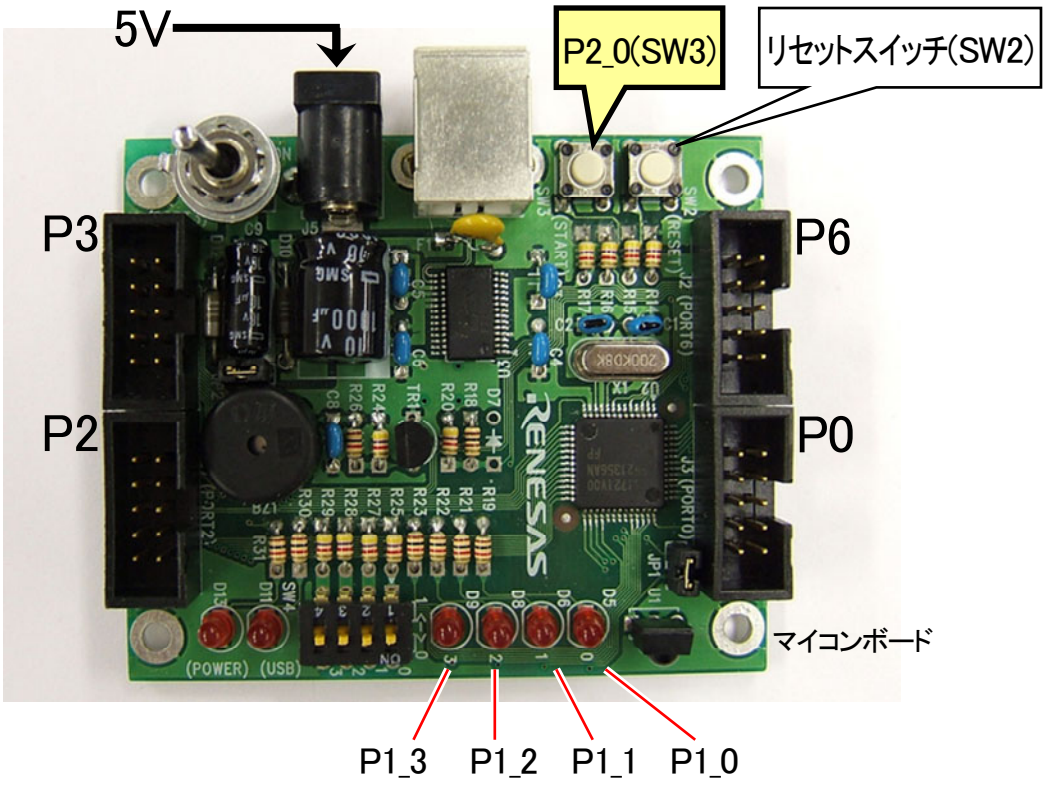
11.2 接続

■使用ポート

マイコンのポート	接続内容
P2_0	マイコンボード上のプッシュスイッチ(SW3)です。 ※SW2 はリセットスイッチで、マイコンのポートには接続されていません。
P1_3、P1_2、P1_1、P1_0	マイコンボード上の LED です。

■接続例

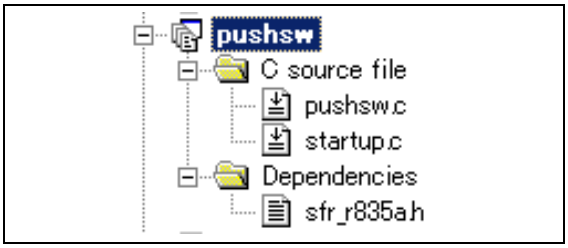
マイコンボードだけで実習できます。



■操作方法

マイコンボードのプッシュスイッチ(SW3)を ON/OFF すると、それに合わせて LED(D5)が点灯/消灯します。

11.3 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	pushsw.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。

11.4 プログラム「pushsw.c」

```
 1 :  /***********************************************************************/
 2 :  /* 対象マイコン  R8C/35A                                               */
 3 :  /* ﾌｧｲﾙ内容      プッシュスイッチの読み込み                            */
 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 :  入力：マイコンボードのプッシュスイッチSW3(P2_0)
11 :  出力：マイコンボードのLED(4bit)
12 :
13 :  マイコンボードのプッシュスイッチSW3(P2_0)から入力した状態を、
14 :  マイコンボードのLED(4bit)に出力します。
15 :  */
16 :
17 :  /*====================================*/
18 :  /* インクルード                       */
19 :  /*====================================*/
20 :  #include "sfr_r835a.h"                 /* R8C/35A SFRの定義ファイル     */
21 :
22 :  /*====================================*/
23 :  /* シンボル定義                       */
24 :  /*====================================*/
25 :
26 :  /*====================================*/
27 :  /* プロトタイプ宣言                   */
28 :  /*====================================*/
29 :  void init( void );
30 :  unsigned char pushsw_get( void );
31 :  void led_out( unsigned char led );
32 :
```

```
 33 :  /************************************************************************/
 34 :  /* メインプログラム                                                     */
 35 :  /************************************************************************/
 36 :  void main( void )
 37 :  {
 38 :      unsigned char d;
 39 :
 40 :      init();                             /* 初期化                       */
 41 :
 42 :      while( 1 ) {
 43 :          d = pushsw_get();
 44 :          led_out( d );
 45 :      }
 46 :  }
 47 :
 48 :  /************************************************************************/
 49 :  /* R8C/35A スペシャルファンクションレジスタ(SFR)の初期化                */
 50 :  /************************************************************************/
 51 :  void init( void )
 52 :  {
 53 :      int i;
 54 :
 55 :      /* クロックをXINクロック(20MHz)に変更 */
 56 :      prc0  = 1;                          /* プロテクト解除               */
 57 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
 58 :      cm05  = 0;                          /* XINクロック発振              */
 59 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
 60 :      ocd2  = 0;                          /* システムクロックをXINにする  */
 61 :      prc0  = 0;                          /* プロテクトON                 */
 62 :
 63 :      /* ポートの入出力設定 */
 64 :      prc2 = 1;                           /* PD0のプロテクト解除          */
 65 :      pd0 = 0xe0;                         /* 7-5:LED 4:MicroSW 3-0:Sensor */
 66 :      p1  = 0x0f;                         /* 3-0:LEDは消灯                */
 67 :      pd1 = 0xdf;                         /* 5:RXD0 4:TXD0 3-0:LED        */
 68 :      pd2 = 0xfe;                         /* 0:PushSW                     */
 69 :      pd3 = 0xfb;                         /* 4:Buzzer 2:IR                */
 70 :      pd4 = 0x83;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
 71 :      pd5 = 0x40;                         /* 7:DIP SW                     */
 72 :      pd6 = 0xff;
 73 :  }
 74 :
 75 :  /************************************************************************/
 76 :  /* プッシュスイッチ値読み込み                                           */
 77 :  /* 戻り値 プッシュスイッチの値 0:OFF 1:ON                               */
 78 :  /************************************************************************/
 79 :  unsigned char pushsw_get( void )
 80 :  {
 81 :      unsigned char sw;
 82 :
 83 :      sw  = ~p2;                          /* プッシュスイッチ読み込み     */
 84 :      sw &= 0x01;                         /* 不要ビットを"0"にする        */
 85 :
 86 :      return  sw;
 87 :  }
 88 :
 89 :  /************************************************************************/
 90 :  /* マイコン部のLED出力                                                  */
 91 :  /* 引数    スイッチ値 0～15                                             */
 92 :  /************************************************************************/
 93 :  void led_out( unsigned char led )
 94 :  {
 95 :      unsigned char data;
 96 :
 97 :      led = ~led;
 98 :      led &= 0x0f;
 99 :      data = p1 & 0xf0;
100 :      p1 = data | led;
101 :  }
102 :
103 :  /************************************************************************/
104 :  /* end of file                                                          */
105 :  /************************************************************************/
```

11.5 プログラムの解説

11.5.1 pushsw_get関数

マイコンボードにはプッシュスイッチが 2 個あります。機能を下記に示します。

- SW2:マイコンのリセットスイッチです。マイコンのポートには接続されていません。
- SW3:マイコンの P2_0 に接続されています。

pushsw_get 関数は、プッシュスイッチ SW3 の値を読み込む関数です。

```
75 :  /************************************************************************/
76 :  /* プッシュスイッチ値読み込み                                           */
77 :  /* 戻り値 プッシュスイッチの値 0:OFF 1:ON                               */
78 :  /************************************************************************/
79 :  unsigned char pushsw_get( void )
80 :  {
81 :      unsigned char sw;
82 :
83 :      sw  = ~p2;                          /* プッシュスイッチ読み込み     */
84 :      sw &= 0x01;                         /* 不要ビットを"0"にする        */
85 :
86 :      return  sw;
87 :  }
```

まず、変数 sw にポート 2(P2)の値を読み込みます。

```
83 :      sw  = ~ p2;                         /* プッシュスイッチ読み込み     */
                ② ①
```

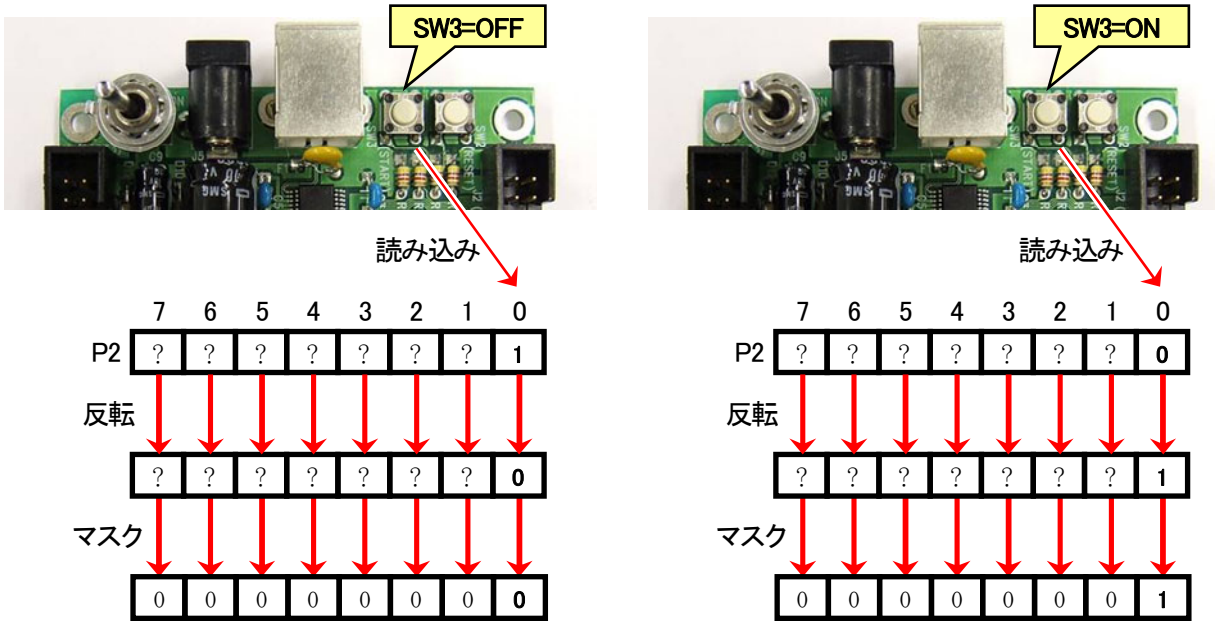
① ポート 2(P2)の値を読み込みます。
② このとき、反転させて sw 変数へ代入します。「~ (チルダ)」は C 言語で反転という意味です。

次に、変数 sw の値をマスクします。

```
84 :      sw &= 0x01;                         /* 不要ビットを"0"にする        */
```

読み込んだ値を 0x01 でマスクします。0x01 は、「0000 0001」なので bit0 のみ有効に、他は強制的に"0"にします。

プッシュスイッチが OFF のとき、ON のときの、pushsw_get 関数の動きを下図に示します。



11.5.2 main関数

```
33 :  /************************************************************************/
34 :  /* メインプログラム                                                     */
35 :  /************************************************************************/
36 :  void main( void )
37 :  {
38 :      unsigned char d;
39 :
40 :      init();                             /* 初期化                       */
41 :
42 :      while( 1 ) {
43 :          d = pushsw_get();
44 :          led_out( d );
45 :      }
46 :  }
```

main 関数は次のような動作をします。

43 行	変数 d にマイコンボード上のプッシュスイッチの値を読み込みます。
44 行	マイコンボード上の LED に変数 d の値を出力します。

結果、マイコンボード上のプッシュスイッチの値を、マイコンボード上の LED に出力します。

11.6 演習

本演習では、LED＝マイコンボード上の D9,D8,D6,D5 とする。LED=″1100″とは、左から D9=″1″、D8=″1″、D6=″0″、D5=″0″という意味とする。

(1) プッシュスイッチが OFF で D6,D5 が点灯(その他は消灯)、ON で D9,D8 が点灯(その他は消灯)するようにしなさい。

(2) ディップスイッチが押されるたびに、LED の値が″0000″(10 進数で 0)→″0001″(10 進数で 1、以下同じ)→″0010″→・・・→″1111″→″0000″と 1 つずつ増えていくようにしなさい。